Fakultät für Maschinenwesen
Lehrstuhl für Angewandte Mechanik

Dynamics and Control of Redundant Robots

Efficient Simulation, Biped Walking, Redundancy Resolution and Collision Avoidance

Dr.-Ing. Thomas Buschmann

Geboren 27.06.1978 in Bobingen, Bundesrepublik Deutschland

Vollständiger Abdruck der zur Erlangung der Lehrbefähigung an der Fakultät für Maschinenwesen der Technischen Universität München für das

Fachgebiet „Angewandte Mechanik“

eingereichten Habilitationsschrift.

Mentorat:

1. Prof. Dr.-Ing. habil. Heinz Ulbrich (Vorsitz)
2. Prof. Dr.-Ing. habil. Boris Lohmann
3. Prof. Dr.-Ing. Prof. E.h. Peter Eberhard (Universität Stuttgart)

Die Habilitation wurde am 21.03.2014 eingereicht.

Acknowledgment

First, I would like to thank Prof. Heinz Ulbrich, Prof. Boris Lohmann and Prof. Peter Eberhard for supporting me and serving as mentors during my habilitation period and Prof. Rixen for giving me the opportunity to continue my research at the Institute of Applied Mechanics.

Most of all, I want to thank my colleges and students and acknowledge their contribution to the results presented in this manuscript. Markus Schwienbacher contributed a large part of the work on recursive forward dynamics, efficient collision geometries and distance calculations. The application of these methods to the redundant agricultural manipulator was implemented and analyzed by Jörg Baur. The manipulator itself was developed by Julian Pfaff. Christoph Schütz implemented the predictive approach to inverse kinematics and analyzed the approach for different manipulator models. The chapter on walking control owes much to the collaboration with Alexander Ewald, Arndt von Twickel and Ansgar Büschges on a comprehensive review of walking control in animals and robots. The section on task-space trajectory modifications for bipedal walking is the outcome of joint work with Arne-Christoph Hildebrandt, who implemented and experimentally evaluated the ideas presented here. Robert Wittmann implemented and analyzed the "Inverse Kineto-Dynamics" method, as well as the error compensation methods and contributed ideas to the stabilization of the walking pattern generation method base on "Inverse Kineto-Dynamics." Finally, the work described here would not have been possible without the excellent biped robot prototype developed by Sebastian Lohmeier and the work on electronics and low-level control by Valerio Favot and Georg Mayr. I am also indebted to Wilhelm Miller, Walter Wöß, Simon Gerer, Philip Schneider, Tobias Schmidt and Georg König for their excellent work in manufacturing and maintaining the bipedal and agricultural robots.

I would also like to thank Jörg Baur, Alexander Ewald, Arne-Christoph Hildebrandt, Christoph Schütz, Markus Schwienbacher and Robert Wittmann and Kathrin Buschmann for proofreading the manuscript.

Contents

1 Introduction

This monograph summarizes research on the dynamics and control of robots carried out at the Institute of Applied Mechanics at the Technische Universität München, Germany. The problem of fast, robust, flexible and stable bipedal walking has served as a constant source of challenges and has motivated most of the work described here.

Humanoid walking robots are among the most complex robotic systems. The large number of Degrees of Freedom (DoFs) requires efficient methods for computing the kinematics and dynamics. These algorithms are both essential components of many planning and control methods and important tools for the design and evaluation of control systems in closed-loop simulations. Chapters 2 and 3 address these topics. They are meant to provide the theoretical background, including specialized algorithms suitable for most robotic systems. The first novel contribution in these chapters is an efficient recursive implementation of forward dynamics for systems with kinematics loops described in minimal coordinates. As a second contribution, the recursive approach is compared to efficient, structure exploiting methods that explicitly use a system mass matrix, which enables a realistic assessment of the relative merit of the two approaches.

Chapter 4 reviews current approaches to controlling bipedal robots, as well as the neural control of human posture and balance. It is intended to provide the context of this motivating application, describe what is possible with current technology and discuss some of the remaining open questions. Current state-of-the-art bipedal robots are capable of quite fast and robust locomotion, but the gap between the capabilities of humans and machines remains considerable. A large portion of current research is still at the level of generating and stabilizing basic walking patterns and successful approaches to this problem use strong simplifications and impose severe restrictions on possible motions.

Promising approaches towards improving the capabilities of bipedal robots are therefore to be found in basic research on improved methods for generating, modeling and controlling bipedal walking. This is the subject of ongoing research in the robotics group at the institute, which is aimed at improved event-based walking control (Buschmann et al. 2012a; Ewald et al. 2012; Ewald and Buschmann 2013) and better modeling and prediction of walking dynamics to enable reactive stepping control (Wittmann et al. 2014). This line of research, however, is not the subject of this monograph.

The second promising line of research is to improve the capability of generating motions that use all the robot's DoFs to achieve this task. This in turn requires efficient methods that enable real-time modification of the generated motion to account for self-

collision, obstacle avoidance and the minimization of appropriately defined objective functions. Suitable methods for this problem are discussed in chapter 5 and applications both to biped walking and to a redundant agricultural manipulator are discussed. Solving general planning and optimal control problems for systems with many DoFs still is too computationally demanding for real-time use. Real-time control systems therefore usually take a hierarchical approach, in which higher layers use coarser system descriptions, enabling longer planning horizons. For complex robots with many DoFs, any given task is likely described by fewer equations than could be satisfied with the system's number of DoFs. In such redundant configurations it is natural to plan in a suitably defined task-space of lower dimension than the configuration space. This reduces the planning effort and allows lower layers to choose a solution which is optimal with respect to some additional criteria. This approach is explored in chapter 5. Classical inverse kinematics methods for redundant manipulators form the background for the presented methods. An important contribution is a very efficient approach to avoiding obstacles and self-collisions. Redundancy resolution is extended to task-space modifications to enable walking over arbitrarily shaped obstacles. The presented method exploits all of the robot's DoFs and is therefore more general than previous approaches. Another contribution is the extension of the inverse kinematics approach to hybrid tasks including desired generalized force trajectories. This enables a novel walking pattern generation scheme for biped robots, but is not restricted to this application. The final contribution discussed in this monograph is a predictive approach to redundancy resolution, which promises to reduce many of the drawbacks of local potential field methods while remaining fast enough for real-time use.

Most work was conducted together with students or PhD candidates at the Institute of Applied Mechanics, whose contribution is acknowledged in the individual chapters.

2 Kinematics

As the "geometry of motion," kinematics is concerned with the description of the motion of mechanical systems without taking forces or inertia into account. In many practical cases, robots can be viewed as systems of interconnected rigid bodies. Elastic deformations do play an important role in some cases. Typical examples are small bending or torsional deformations in light-weight robot links (see Bremer 2008; Siciliano and Khatib 2008, ch. 13.2.2). In the overwhelming majority of robotic applications, however, elastic deformations are small relative to the gross motion of the (deformable) body. In such cases, the overall motion of the body is equivalent to that of a rigid body with additional small elastic deformations relative to the rigid-body motion (Bremer 2008, ch. 5). This is why the description of rigid body motion is central to the kinematics, dynamics and control of robotic systems. In most cases, robots are explicitly designed to avoid significant link deformations. We will therefore limit the discussion to rigid body systems in the following. Treatments of elastic multibody dynamics may be found in (e.g., Shabana 2005; Bremer 2008; Gattringer 2011). Note that elastic deformations in components such as gears and bearings, which are the most relevant in the majority of applications, are not excluded. Such components are typically modeled by force elements describing the effect of elasticity (and damping) as a function of generalized coordinates and velocities. Such a model assumes that elasticity is relevant, but *internal vibrations* of the elastic structure itself are negligible.

This chapter derives the kinematics of a single rigid body (section 2.1), various useful coordinate transformations (section 2.2) and the kinematics of systems of interconnected rigid bodies (section 2.3). In section 2.4 different methods for improving the efficiency of kinematics calculations in computer programs are described.

2.1 Single Rigid Body Kinematics

This section develops equations governing the motion of a single rigid body, which are an important foundation for the kinematics and dynamics of multibody systems.

2.1.1 Conditions for Rigid Body Motion

A mathematical description of rigid body motion can be derived directly from the fact that the shape of a rigid body remains constant at all times, which implies a constant

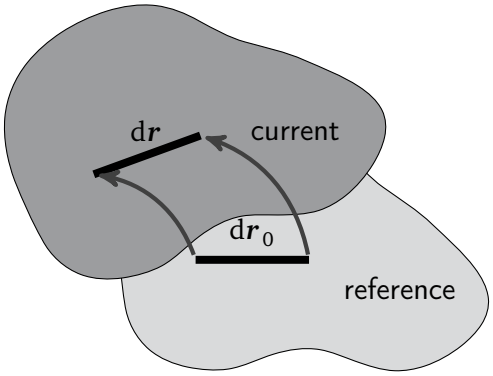


Figure 2.1: The current and the reference positions of a rigid body with a line segment in its current configuration ($d\boldsymbol{r}$) and in the reference configuration ($d\boldsymbol{r}_0$). The line segment is rotated and shifted, but the length is unchanged.

distance between any two body-fixed points $\boldsymbol{r}_i$ and $\boldsymbol{r}_k$:

$$\left\|\boldsymbol{r}_i - \boldsymbol{r}_k\right\| = \text{constant}, \quad \boldsymbol{r}_i, \boldsymbol{r}_k \in \mathbb{R}^3. \tag{2.1}$$

More specifically, the current length of an infinitesimal line segment $\mathrm{d}\boldsymbol{r}$ is equal to the length $\mathrm{d}\boldsymbol{r}_0$ in a reference position of the body (see figure 2.1) and therefore

$$\mathrm{d}\boldsymbol{r}^2 - \mathrm{d}\boldsymbol{r}_0^2 = 0. \tag{2.2}$$

Since the current configuration of the line segment $\mathrm{d}\boldsymbol{r}$ depends only on the point in the rigid body it emanates from, it is a function of $\boldsymbol{r}_0$. We can therefore express it as

$$\mathrm{d}\boldsymbol{r} = \underbrace{\frac{\partial \boldsymbol{r}(\boldsymbol{r}_0)}{\partial \boldsymbol{r}_0}}_{\boldsymbol{F}} \mathrm{d}\boldsymbol{r}_0, \tag{2.3}$$

where $\boldsymbol{F}$ is the deformation gradient tensor. With (2.3), we can re-formulate (2.2) as:

$$\mathrm{d}\boldsymbol{r}^2 - \mathrm{d}\boldsymbol{r}_0^2 = \mathrm{d}\boldsymbol{r}_0^T \underbrace{\left(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I}_3\right)}_{\boldsymbol{G}} \mathrm{d}\boldsymbol{r}_0. \tag{2.4}$$

The matrix $\boldsymbol{I}_3$ is the 3×3 identity matrix. A vanishing Green-Lagrangian strain tensor $\boldsymbol{G}$ therefore is a necessary and sufficient condition for rigid-body motion (Altenbach and Altenbach 1994, p. 56, Shabana 2005, ch. 4.4):

$$\boldsymbol{G} = \boldsymbol{0} \quad \Leftrightarrow \quad \boldsymbol{F}^T\boldsymbol{F} = \boldsymbol{I}_3. \tag{2.5}$$

This directly yields the useful relationship

$$\boldsymbol{F}^{-1} = \boldsymbol{F}^T. \tag{2.6}$$

With $\boldsymbol{F} = \begin{pmatrix} \boldsymbol{f}_1 & \boldsymbol{f}_2 & \boldsymbol{f}_3 \end{pmatrix}$, (2.5) can be expanded into six equations for the nine components of $\boldsymbol{F}$:

$$\boldsymbol{f}_i^T \boldsymbol{f}_k = 0 \quad \text{for} \quad i = \{x, y, z\},\ k = \{x, y, z\},\ i \neq k \tag{2.7}$$

$$\boldsymbol{f}_i^T \boldsymbol{f}_i = 1 \quad \text{for} \quad i = \{x, y, z\} \tag{2.8}$$

That is, the $\boldsymbol{f}_i$ form an orthonormal basis of $\mathbb{R}^3$ whose configuration may be uniquely described by three independent parameters.

In the reference configuration we have $\boldsymbol{r} = \boldsymbol{r}_0$, and therefore $\boldsymbol{F} = (\boldsymbol{e}_x \quad \boldsymbol{e}_y \quad \boldsymbol{e}_z)$, where the $\boldsymbol{e}_i$ are the basis of the inertial reference frame from which the current configuration $\boldsymbol{r}$ is measured.[1] Adopting an orthonormal, right-handed[2] frame as the inertial reference, this simplifies to $\boldsymbol{f}_i = \boldsymbol{e}_i$, $\boldsymbol{F} = \boldsymbol{I}_3$ in the reference configuration. By continuity this implies that the $\boldsymbol{f}_i$ also define a right-handed frame with unit determinant.[3] Mathematically, $\boldsymbol{F}$ can then be identified as a member of the *special orthogonal group* $SO(3)$ of matrices with orthonormal column vectors and a determinant equal to one.[4]

Since $\boldsymbol{F}$ is derived for the motion of an infinitesimal line segment without reference to an absolute position, it describes the *rotation* of the body relative to the inertial frame. An equivalent view is that $\boldsymbol{F}$ transforms a vector represented in the body-fixed frame given by $\boldsymbol{f}_i$ into a representation in the global frame.

When used as a *rotation matrix* describing such a transformation from a frame K to a second frame L, $\boldsymbol{F}$ will be denoted by $\boldsymbol{A}_{LK}$ in the following.[5] Since we will use several coordinate frames, the frame in which a vector is represented will be indicated by a preceding subscript in the following. That is, the vector $\boldsymbol{r}$ written in the K-frame ($\mathscr{F}_K$) and L-frame ($\mathscr{F}_L$) is denoted by ${}_K\boldsymbol{r}$ and ${}_L\boldsymbol{r}$. The representations can be converted into each other via

$${}_L\boldsymbol{r} = \boldsymbol{A}_{LK}\, {}_K\boldsymbol{r}\,. \tag{2.9}$$

In the following, we will denote the inertial reference frame by the index I ($\mathscr{F}_I$), and a

1. An "inertial" frame is a global, non-accelerated frame.

2. That is, $\boldsymbol{e}_z = \boldsymbol{e}_x \times \boldsymbol{e}_y$ with the usual definition of the cross product $\times$.

3. This follows from $\det\begin{pmatrix} \boldsymbol{f}_1 & \boldsymbol{f}_2 & \boldsymbol{f}_3 \end{pmatrix} = \boldsymbol{f}_1^T(\boldsymbol{f}_2 \times \boldsymbol{f}_3) = \boldsymbol{f}_1^T \boldsymbol{f}_1 = 1$. The first identity follows from the scalar triple product representation of the determinant and the second and third identities from the use of an orthonormal and right-handed frame.

4. The set of matrices with $\boldsymbol{F}^T\boldsymbol{F} = \boldsymbol{I}_3$ (elements of the orthogonal group $O(3)$) with $\det(\boldsymbol{F}) = -1$ are reflections and not relevant for representing rigid-body motion. For a presentation of mechanics based on concepts from differential geometry see (e.g., Holm 2008a, 2008b).

5. The naming of rotation matrices follows Shabana (2005) and Bremer (2008)

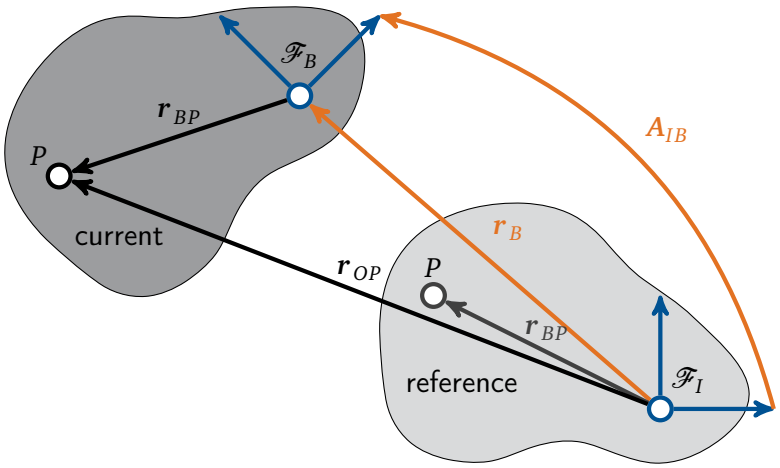


Figure 2.2: A point P of a rigid body in the current configuration (dark color) and the reference configurations (light color) of the body, together with the inertial frame $\mathscr{F}_I$ and body-fixed frame $\mathscr{F}_B$.

body-fixed frame by B ($\mathscr{F}_B$). We can then write (2.3) as

$$\mathrm{d}_I \boldsymbol{r} = \boldsymbol{A}_{IB}\, \mathrm{d}_B \boldsymbol{r}\,. \tag{2.10}$$

2.1.2 Rigid Body Positions

The current position of a point in the rigid body can be calculated by integrating (2.10):

$$_I\boldsymbol{r} = \boldsymbol{A}_{IB}\, {}_B\boldsymbol{r} + {}_I\boldsymbol{r}_T\,. \tag{2.11}$$

The "integration constant" $\boldsymbol{r}_T \in \mathbb{R}^3$ adds three more free parameters to the three in $\boldsymbol{A}_{IB}$, giving the rigid body six independent parameters, or DoFs. Mathematically, the *pose* (position and orientation) of a rigid body is a member of the *special Euclidean group* $SE(3) = \mathbb{R}^3 \times SO(3)$ (see, e.g., Holm 2008a). Geometrically, (2.11) implies that the position of the point $\boldsymbol{r}$ in $\mathscr{F}_I$ is obtained from the position in $\mathscr{F}_B$ by transforming the vector in the body-fixed frame ${}_B\boldsymbol{r}$ via $\boldsymbol{A}_{IB}$ about $\boldsymbol{r}_T$ (*rotation*) and adding $\boldsymbol{r}_T$ (*translation*).[6]

Eq. (2.11) is useful not only for rigid bodies, but also for the more general case of determining the absolute position of a point given in a moving frame, relative to the origin O_B of $\mathscr{F}_B$ (see section 2.3). To emphasize this, we can re-write it as:

$$_I\boldsymbol{r}_P = {}_I\boldsymbol{r}_B + \boldsymbol{A}_{IB}\, {}_B\boldsymbol{r}_{BP}\,. \tag{2.12}$$

6. This is also referred to as Chasles' theorem: rigid body motion consists of three translations and three rotations (Shabana 2005).

Here, $\boldsymbol{r}_B$ is the origin of $\mathscr{F}_B$ and $\boldsymbol{r}_{BP}$ the vector from O_B to P (see figure 2.2)

2.1.3 Velocities

The absolute velocity of a point given by the vector ${}_B\boldsymbol{r}_{BP}$ relative to the moving $\mathscr{F}_B$ is simply determined by differentiating with respect to time *in* $\mathscr{F}_I$:

$$\begin{aligned} {}_I\dot{\boldsymbol{r}}_P &= \tfrac{\mathrm{d}}{\mathrm{d}t}\left({}_I\boldsymbol{r}_B + \boldsymbol{A}_{IB}\,{}_B\boldsymbol{r}_{BP}\right) \\ &= {}_I\dot{\boldsymbol{r}}_B + \dot{\boldsymbol{A}}_{IB}\,{}_B\boldsymbol{r}_{BP} + \boldsymbol{A}_{IB}\,{}_B\mathring{\boldsymbol{r}}_{BP}\,. \end{aligned} \tag{2.13}$$

Here, the circle ($\mathring{\boldsymbol{x}}$) indicates the differentiation of the component representation of a vector in the current frame, while the dot ($\dot{\boldsymbol{x}}$) denotes the absolute differentiation of the vector. The two differ by the velocity induced by the rate of change of the basis vectors of $\mathscr{F}_B$ relative to $\mathscr{F}_I$. The absolute velocity in $\mathscr{F}_B$ is finally obtained from (2.13) by writing the result in $\mathscr{F}_B$:

$$\begin{aligned} {}_B\dot{\boldsymbol{r}}_P &= \boldsymbol{A}_{BI}\,{}_I\dot{\boldsymbol{r}}_P \\ &= {}_B\dot{\boldsymbol{r}}_B + \boldsymbol{A}_{BI}\dot{\boldsymbol{A}}_{IB}\,{}_B\boldsymbol{r}_{BP} + {}_B\mathring{\boldsymbol{r}}_{BP}\,. \end{aligned} \tag{2.14}$$

Eqs. (2.13) and (2.14) could be used to calculate the absolute velocity. This, however, would require calculating the derivative of the transformation matrix with respect to time, which would be tedious and inefficient as the parametrizations of $\boldsymbol{A}_{BI}$ given in section 2.2 show. Fortunately, there is a simpler formulation based on the *angular velocity*.

Angular Velocity

To determine the time derivative of the rotation matrix, it is useful to study the consequences of the rigidity conditions with respect to the time evolution of the column vectors $\boldsymbol{a}_i$ of $\boldsymbol{A}_{IB}$, which are the basis-vectors of $\mathscr{F}_B$. Differentiating (2.8) yields:

$$\tfrac{\mathrm{d}(\boldsymbol{a}_i^T\boldsymbol{a}_i)}{\mathrm{d}t} = 2\dot{\boldsymbol{a}}_i^T\boldsymbol{a}_i = 0, \tag{2.15}$$

Since the dot product vanishes, $\boldsymbol{a}_i$ and $\dot{\boldsymbol{a}}_i$ are orthogonal and we can parametrize $\dot{\boldsymbol{a}}_i$ by a vector $\boldsymbol{u}_i \in \mathbb{R}^3$:

$$\dot{\boldsymbol{a}}_i = \boldsymbol{u}_i \times \boldsymbol{a}_i = \widetilde{\boldsymbol{u}}_i\boldsymbol{a}_i. \tag{2.16}$$

The tilde operator $\widetilde{(\,)}$ maps a vector in $\mathbb{R}^3$ to a skew-symmetric matrix,[7] and is an equivalent way of denoting the cross product:

$$\widetilde{\begin{pmatrix} x \\ y \\ z \end{pmatrix}} := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}. \tag{2.17}$$

Differentiating the orthogonality relationship (2.7) and substituting (2.16) into the result we obtain:

$$\begin{aligned} 0 &= \tfrac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{a}_i^T \boldsymbol{a}_k) = \dot{\boldsymbol{a}}_i^T \boldsymbol{a}_k + \boldsymbol{a}_i^T \dot{\boldsymbol{a}}_k, \\ \Rightarrow 0 &= (\widetilde{\boldsymbol{u}}_i \boldsymbol{a}_i)^T \boldsymbol{a}_k + \boldsymbol{a}_i^T (\widetilde{\boldsymbol{u}}_k \boldsymbol{a}_k) = \boldsymbol{a}_i^T \left(\widetilde{\boldsymbol{u}}_i^T + \widetilde{\boldsymbol{u}}_k \right) \boldsymbol{a}_k. \end{aligned} \tag{2.18}$$

Since the tilde operator is skew-symmetric ($\widetilde{\boldsymbol{u}}_i^T = -\widetilde{\boldsymbol{u}}_i$), this implies that all $\boldsymbol{u}_i, \boldsymbol{u}_k$ are identical:

$$\boldsymbol{u}_i = \boldsymbol{u}_k = \boldsymbol{\omega} \quad i,k \in \{x,y,z\}. \tag{2.19}$$

The unique vector $\boldsymbol{\omega}$ describing the rate of change of the basis vectors of $\mathscr{F}_B$ relative to $\mathscr{F}_I$ is the *angular velocity* of the rigid body. From (2.16) we directly get:

$$\dot{\boldsymbol{a}}_i = {}_I\widetilde{\boldsymbol{\omega}}_{IB}\, \boldsymbol{a}_i, \tag{2.20}$$

$$\Rightarrow \quad \dot{\boldsymbol{A}}_{IB} = {}_I\widetilde{\boldsymbol{\omega}}_{IB}\, \boldsymbol{A}_{IB}, \tag{2.21}$$

$$\Leftrightarrow \quad {}_I\widetilde{\boldsymbol{\omega}}_{IB} = \dot{\boldsymbol{A}}_{IB} \boldsymbol{A}_{IB}^T. \tag{2.22}$$

The angular velocity in $\mathscr{F}_B$ is obtained via:

$${}_I\widetilde{\boldsymbol{\omega}}_{IB}\, {}_I\boldsymbol{r} = {}_I\widetilde{\boldsymbol{\omega}}_{IB} (\boldsymbol{A}_{IB}\, {}_B\boldsymbol{r}) \stackrel{!}{=} \boldsymbol{A}_{IB} ({}_B\widetilde{\boldsymbol{\omega}}_{IB}\, {}_B\boldsymbol{r}) \tag{2.23}$$

$$\Rightarrow {}_B\widetilde{\boldsymbol{\omega}}_{IB} = \boldsymbol{A}_{BI}\dot{\boldsymbol{A}}_{IB} = \boldsymbol{A}_{BI}\, {}_I\widetilde{\boldsymbol{\omega}}_{IB}\, \boldsymbol{A}_{IB}. \tag{2.24}$$

That is, $\widetilde{\boldsymbol{\omega}}$ satisfies the transformation rules of a second order tensor.

Linear Velocity

With the angular velocity $\boldsymbol{\omega}$, the linear velocity equation (2.13) may be re-written as:

$$\begin{aligned} {}_I\dot{\boldsymbol{r}}_P &= {}_I\dot{\boldsymbol{r}}_B + \dot{\boldsymbol{A}}_{IB}\, {}_B\boldsymbol{r}_{BP} + \boldsymbol{A}_{IB}\, {}_B\mathring{\boldsymbol{r}}_{BP} \\ &= {}_I\dot{\boldsymbol{r}}_B + {}_I\widetilde{\boldsymbol{\omega}}_{IB}\, \boldsymbol{A}_{IB}\, {}_B\boldsymbol{r}_{BP} + \boldsymbol{A}_{IB}\, {}_B\mathring{\boldsymbol{r}}_{BP}\,. \end{aligned} \tag{2.25}$$

7. Since $\widetilde{\boldsymbol{\omega}}$ is skew-symmetric, it is an element of $\mathfrak{so}(3)$, the tangent space to $O(3)$ at the identity matrix.

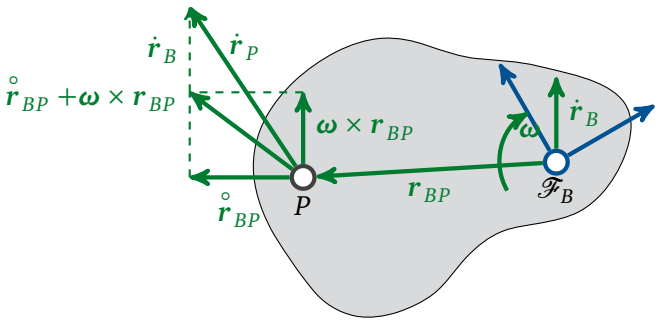


Figure 2.3: Illustration of the components contributing to the absolute velocity $\dot{\boldsymbol{r}}_P$ of a point in a rigid body. $\mathscr{F}_B$ is a body-fixed frame, $\boldsymbol{\omega}$ the angular velocity of the body and $\dot{\boldsymbol{r}}_B$ the absolute velocity of the origin of $\mathscr{F}_B$.

The absolute velocity in the body-fixed frame is then given by:

$$\begin{aligned} {}_B\dot{\boldsymbol{r}}_P &= \boldsymbol{A}_{BI}\, {}_I\dot{\boldsymbol{r}}_{BP} \\ &= {}_B\dot{\boldsymbol{r}}_B + \boldsymbol{A}_{BI}\dot{\boldsymbol{A}}_{IB}\, {}_B\boldsymbol{r}_{BP} + {}_B\dot{\boldsymbol{r}}_{BP} \\ &= {}_B\dot{\boldsymbol{r}}_B + {}_B\tilde{\boldsymbol{\omega}}_{IB}\, {}_B\boldsymbol{r}_{BP} + {}_B\mathring{\boldsymbol{r}}_{BP}\,. \end{aligned} \tag{2.26}$$

That is, the translational velocity is the sum of the velocity of $\mathscr{F}_B$, the speed induced by the rotation of $\mathscr{F}_B$ with $\boldsymbol{\omega}$ and the velocity ${}_B\mathring{\boldsymbol{r}}_{BP}$ relative to $\mathscr{F}_B$ (see figure 2.3). Note that ${}_B\dot{\boldsymbol{r}}_B$ still denotes the *absolute* rate of change of the vector $\boldsymbol{r}_B$ relative to the inertial frame and cannot be calculated by differentiating the component representation ${}_B\boldsymbol{r}_B$, since this would ignore the velocity induced by the moving basis vectors of the body-fixed frame.

2.1.4 Accelerations

As with velocities, the absolute acceleration with respect to an inertial frame is easily calculated by differentiation in the inertial frame:

$$\begin{aligned} {}_I\ddot{\boldsymbol{r}}_P &= \tfrac{\mathrm{d}}{\mathrm{d}t}\left({}_I\dot{\boldsymbol{r}}_P\right) \\ &= {}_I\ddot{\boldsymbol{r}}_B + {}_I\dot{\tilde{\boldsymbol{\omega}}}_{IB}\boldsymbol{A}_{IB}\, {}_B\boldsymbol{r}_{BP} + {}_I\tilde{\boldsymbol{\omega}}_{IB}\dot{\boldsymbol{A}}_{IB}\, {}_B\boldsymbol{r}_{BP} + {}_I\tilde{\boldsymbol{\omega}}_{IB}\boldsymbol{A}_{IB}\, {}_B\mathring{\boldsymbol{r}}_{BP} + \dot{\boldsymbol{A}}_{IB}\, {}_B\mathring{\boldsymbol{r}}_{BP} + \boldsymbol{A}_{IB}\, {}_B\mathring{\mathring{\boldsymbol{r}}}_{BP} \\ &= {}_I\ddot{\boldsymbol{r}}_B + \left({}_I\dot{\tilde{\boldsymbol{\omega}}}_{IB} + {}_I\tilde{\boldsymbol{\omega}}_{IB}\, {}_I\tilde{\boldsymbol{\omega}}_{IB}\right)\boldsymbol{A}_{IB}\, {}_B\boldsymbol{r}_{BP} + 2\, {}_I\tilde{\boldsymbol{\omega}}_{IB}\boldsymbol{A}_{IB}\, {}_B\mathring{\boldsymbol{r}}_{BP} + \boldsymbol{A}_{IB}\, {}_B\mathring{\mathring{\boldsymbol{r}}}_{BP}\,. \end{aligned} \tag{2.27}$$

The body-fixed representation is again obtained by back-transformation:

$$\begin{aligned}
{}_B\ddot{\boldsymbol{r}}_P &= \boldsymbol{A}_{BI}\,{}_I\ddot{\boldsymbol{r}}_P \\
&= {}_B\ddot{\boldsymbol{r}}_B + \boldsymbol{A}_{BI}\left({}_I\dot{\widetilde{\boldsymbol{\omega}}}_{IB} + {}_I\widetilde{\boldsymbol{\omega}}_{IB}\,{}_I\widetilde{\boldsymbol{\omega}}_{IB}\right)\boldsymbol{A}_{IB}\,{}_B\boldsymbol{r}_{BP} + 2\boldsymbol{A}_{BI}\,{}_I\widetilde{\boldsymbol{\omega}}_{IB}\boldsymbol{A}_{IB}\,{}_B\mathring{\boldsymbol{r}}_{BP} + {}_B\mathring{\mathring{\boldsymbol{r}}}_{BP} \\
&= {}_B\ddot{\boldsymbol{r}}_B + \left({}_B\dot{\widetilde{\boldsymbol{\omega}}}_{IB} + {}_B\widetilde{\boldsymbol{\omega}}_{IB}\,{}_B\widetilde{\boldsymbol{\omega}}_{IB}\right){}_B\boldsymbol{r}_{BP} + 2\,{}_B\widetilde{\boldsymbol{\omega}}_{IB}\,{}_B\mathring{\boldsymbol{r}}_{BP} + {}_B\mathring{\mathring{\boldsymbol{r}}}_{BP}\,.
\end{aligned} \tag{2.28}$$

2.2 Rotation Matrices

What is missing in the preceding section for a complete description of rigid body motion is a parametrization of the rotation matrix $\boldsymbol{A}_{IB}$, that is, a concrete representation of $\boldsymbol{A}_{IB}$ as a function of some free parameters representing the spacial rotation.

2.2.1 Axis-Angle Representation

Rotations can be parametrized in a number of ways. One possible parametrization may be derived directly from (2.21) by considering constant angular velocities.[8] With the reference configuration as initial condition, the Initial Value Problem (IVP) becomes:

$$\dot{\boldsymbol{A}}_{IB} = {}_I\widetilde{\boldsymbol{\omega}}_{IB}\boldsymbol{A}_{IB}, \tag{2.29}$$

$$\boldsymbol{A}_{IB}(t=0) = \boldsymbol{I}_3, \tag{2.30}$$

which has the closed-form solution:

$$\boldsymbol{A}_{IB} = \mathrm{e}^{\widetilde{\boldsymbol{\omega}}t} = \boldsymbol{I}_3 + \widetilde{\boldsymbol{\omega}}t + \frac{1}{2!}\widetilde{\boldsymbol{\omega}}^2t^2 + \frac{1}{3!}\widetilde{\boldsymbol{\omega}}^3t^3 + \frac{1}{4!}\widetilde{\boldsymbol{\omega}}^4t^4 + \cdots \tag{2.31}$$

Considering only the current and a final configuration at time T with a rotation angle of $\boldsymbol{\varphi} = \boldsymbol{\omega}T = \varphi\boldsymbol{u}$, the rotation matrix may be written as

$$\begin{aligned}
\boldsymbol{A}_{IB} &= \boldsymbol{I}_3 + \widetilde{\boldsymbol{u}}\varphi + \frac{1}{2!}\widetilde{\boldsymbol{u}}^2\varphi^2 + \frac{1}{3!}\widetilde{\boldsymbol{u}}^3\varphi^3 + \frac{1}{4!}\widetilde{\boldsymbol{u}}^4\varphi^4 + \cdots, \\
\varphi &= \sqrt{\boldsymbol{\varphi}^T\boldsymbol{\varphi}}, \\
\boldsymbol{u} &= \frac{\boldsymbol{\varphi}}{\boldsymbol{\varphi}^T\boldsymbol{\varphi}}.
\end{aligned} \tag{2.32}$$

Since the final rotation depends only on the magnitude and direction of rotation and not the intermediate configurations, this result is not restricted to constant angular

8. The derivation is inspired by that given by Bremer (2008, pp. 30). However, Bremer begins by parametrizing $\boldsymbol{A}_{IB}$ with $\boldsymbol{u}\varphi, \|\boldsymbol{u}\| = 1$ and studying the derivative of the squared length a vector $\boldsymbol{r}^T\boldsymbol{r}$ with respect to φ.

velocities. Using the identities $\widetilde{\boldsymbol{u}}^3 = -\widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{u}}^4 = -\widetilde{\boldsymbol{u}}\widetilde{\boldsymbol{u}}, \ldots$ (cf. appendix A.1) we obtain

$$\begin{aligned}
\boldsymbol{A}_{IB} &= \boldsymbol{I}_3 + \widetilde{\boldsymbol{u}}\left(\varphi - \frac{1}{3!}\varphi^3 + \cdots\right) + \widetilde{\boldsymbol{u}}^2\left(\frac{1}{2!}\varphi^2 - \frac{1}{4!}\varphi^4 + \cdots\right) \\
&= \boldsymbol{I}_3 + \widetilde{\boldsymbol{u}}\sin(\varphi) + \widetilde{\boldsymbol{u}}^2(1-\cos\varphi) \\
&= \begin{pmatrix} 1 + a_\varphi u_x^2 & -\mathrm{s}\,\varphi u_z + a_\varphi u_y u_x & \mathrm{s}\,\varphi u_y + a_\varphi u_z u_x \\ \mathrm{s}\,\varphi u_z + a_\varphi u_y u_x & 1 + a_\varphi u_y^2 & -\mathrm{s}\,\varphi u_x + a_\varphi u_z u_y \\ -\mathrm{s}\,\varphi u_y + a_\varphi u_z u_x & \mathrm{s}\,\varphi u_x + a_\varphi u_z u_y & 1 + a_\varphi u_z^2 \end{pmatrix}, \\
a_\varphi &:= (1 - \mathrm{c}\,\varphi).
\end{aligned} \tag{2.33}$$

Here, $\mathrm{s}\,x = \sin(x)$ and $\mathrm{c}\,x = \cos(x)$ are used for conciseness. The four parameters φ, u_x, u_y, u_z must satisfy the constraint $\|\boldsymbol{u}\| = 1$, leaving three free parameters as expected from (2.7). Eq. (2.33) is called the *axis-angle representation* of spacial rotations, since it produces a rotation by φ about the fixed rotation axis $\boldsymbol{u}$. Figure 2.4 gives a geometric illustration of (2.33).

Eq. (2.33) may be rearranged into an alternative representation, the so-called Rodriguez formula (e.g., Shabana 2005):

$$\boldsymbol{A}_{IB} = \boldsymbol{I}_3 + \widetilde{\boldsymbol{u}}\sin(\varphi) + 2\widetilde{\boldsymbol{u}}^2(\sin^2\frac{\varphi}{2}). \tag{2.34}$$

A minimal representation with three independent parameters and no constraints may be obtained from the axis-angle formulation by directly using $\boldsymbol{u} = \boldsymbol{\varphi}/\left\|\boldsymbol{\varphi}\right\|$ in (2.33):

$$\boldsymbol{A}_{IB} = \boldsymbol{I}_3 + \frac{1}{\varphi}\widetilde{\boldsymbol{\varphi}}\sin(\varphi) + \frac{1}{\varphi^2}\widetilde{\boldsymbol{\varphi}}^2(1-\cos\varphi), \qquad \varphi = \sqrt{\boldsymbol{\varphi}^T\boldsymbol{\varphi}}. \tag{2.35}$$

A more widely used geometrical derivation of the axis-angle parametrization may be found in (e.g., Siciliano et al. 2008; Gattringer 2011; Shabana 2005).

Angular Velocity

We assumed a constant angular velocity and axis of rotation for deriving the axis-angle representation. The angular velocity for a constant axis of rotation $\boldsymbol{u} =$ constant is obviously given by $\boldsymbol{u}\dot{\varphi}$. In the general case, however, a non-constant axis of rotation contributes to the angular velocity, which must then be calculated from (2.21), that is, $\boldsymbol{A}_{IB}^T\dot{\boldsymbol{A}}_{IB} = {}_B\widetilde{\boldsymbol{\omega}}_{IB}$, yielding:

$${}_B\boldsymbol{\omega}_{IB} = \boldsymbol{u}\dot{\varphi} + \left(\sin(\varphi)\boldsymbol{I}_3 - (1-\cos(\varphi))\widetilde{\boldsymbol{u}}\right)\dot{\boldsymbol{u}}. \tag{2.36}$$

A detailed derivation of this result is given in appendix A.2.

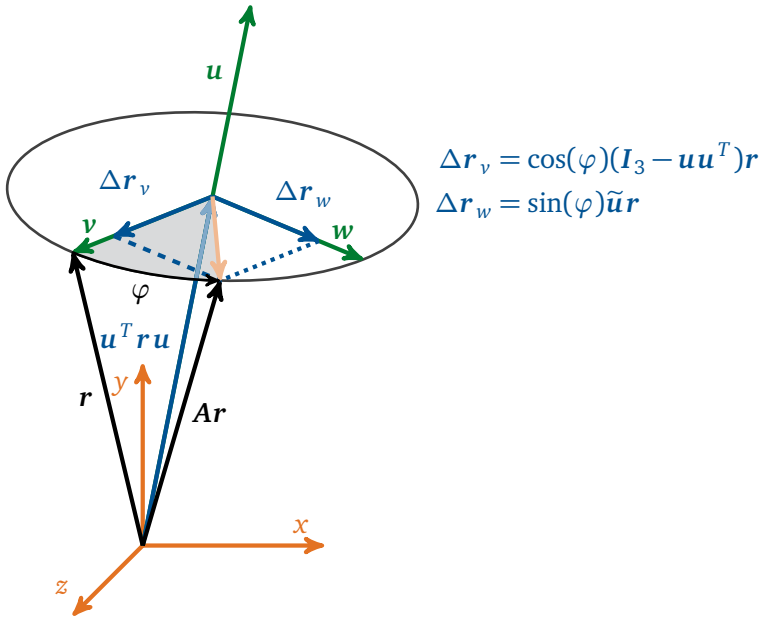


Figure 2.4: Geometric illustration of the axis-angle parametrization of spatial rotations. Rearranging the axis angle parametrization of $\boldsymbol{A}$ (2.33) using the identity (A.5) the rotated vector may be written as the sum of three components: $\boldsymbol{Ar} = (\boldsymbol{u})\boldsymbol{u}^T\boldsymbol{r} + \cos(\varphi)(\boldsymbol{I}_3 - \boldsymbol{u}\boldsymbol{u}^T)\boldsymbol{r} + \sin(\varphi)\tilde{\boldsymbol{u}}\boldsymbol{r}$. The first term contains the component in the direction of $\boldsymbol{u}$, which is not changed by the rotation. The second summand contains the component in direction of $\boldsymbol{v} = \boldsymbol{r} - \boldsymbol{u}\boldsymbol{u}^T\boldsymbol{r}$, which is a natural basis vector for an orthogonal frame in the plane formed by $\boldsymbol{r}$ and $\boldsymbol{u}$. The third term contains the contribution in the direction of $\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{r}$, which together with $\boldsymbol{u}$ and $\boldsymbol{v}$ forms a right-handed frame. The magnitude of the contributions in $\boldsymbol{x}$ and $\boldsymbol{y}$ direction are given by the dashed and dotted lines, whose length is directly obtained from φ via the geometric definition of the sine and cosine functions.

Inverse Calculation: Axis and Angle from Rotation Matrix

In case a rotation matrix is known, the corresponding axis and angle of rotation can be computed from (2.33). Following the procedure in (Gattringer 2011), the rotation angle is obtained by calculating the trace of $\boldsymbol{A}_{IB}$:

$$\begin{aligned} &\operatorname{tr}(\boldsymbol{A}_{IB}) = 1 + 2\cos(\varphi) \\ \Leftrightarrow\ &\cos(\varphi) = \frac{1}{2}\left(A_{IB,11} + A_{IB,22} + A_{IB,33} - 1\right). \end{aligned} \tag{2.37}$$

With $\cos(\varphi)$, the rotation axis may be calculated from $\boldsymbol{A}_{IB}$ (2.34) by observing that $\tilde{\boldsymbol{u}}$ is anti-symmetric, while $\tilde{\boldsymbol{u}}^2$ and $\boldsymbol{I}_3$ are symmetric, so the axis vector $\boldsymbol{u}$ is easily calculated by subtracting the corresponding non-diagonal elements of $\boldsymbol{A}_{IB}$, leaving only

the contribution due to $\widetilde{\boldsymbol{u}}\sin(\varphi)$:

$$\boldsymbol{u} = \frac{1}{2\sin(\varphi)} \begin{pmatrix} A_{IB,32} - A_{IB,23} \\ A_{IB,13} - A_{IB,31} \\ A_{IB,21} - A_{IB,12} \end{pmatrix}. \tag{2.38}$$

Apparently, the angle of rotation can always be calculated, while the axis is ill defined for all $\varphi = k\pi,\ \ k \in \mathbb{Z}$. For $\varphi = 0$ this is geometrically plausible, since any rotation axis gives the same rotation matrix $\boldsymbol{A}_{IB} = \boldsymbol{I}_3$. For $\varphi = \pm\pi$ the result is ambiguous, because the parameters $\varphi = \pi$ and $\boldsymbol{u}$ yield the same rotation matrix $\boldsymbol{A}_{IB} = \boldsymbol{I}_3 + \boldsymbol{u}^2$ as $\varphi = -\pi$ and $-\boldsymbol{u}$.

2.2.2 Elementary Rotations

An important special case are rotations about the coordinate axes $\boldsymbol{e}_i$: with the proper choice of reference frames, transformations between the body-fixed frames of segments connected by revolute joints can always be described using such *elementary rotations*.

Evaluating the axis-angle formulation (2.33) for a rotation about the x-axis $\boldsymbol{e}_x$ by an angle of α directly yields:

$$\begin{aligned} \boldsymbol{A}_{IB}(\alpha) = \boldsymbol{A}_x^T(\alpha) &= \boldsymbol{I}_3 + \widetilde{\boldsymbol{e}}_x \sin(\alpha) + \widetilde{\boldsymbol{e}}_x^2 (1 - \cos(\alpha)) \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \sin(\alpha) + \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} (1 - \cos(\alpha)) \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}. \end{aligned} \tag{2.39}$$

Here the "elementary rotation" $\boldsymbol{A}_x(\boldsymbol{\alpha})$ is defined for a positive angle and a transformation from $\mathscr{F}_I$ to $\mathscr{F}_B$, following the sign convention in (Gattringer 2011; Bremer 2008; Pfeiffer 1989).

The results for rotations by β about $\boldsymbol{e}_y$ and by γ about $\boldsymbol{e}_z$ are:

$$\boldsymbol{A}_{IB}(\beta) = \boldsymbol{A}_y^T(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}, \tag{2.40}$$

$$\boldsymbol{A}_{IB}(\gamma) = \boldsymbol{A}_z^T(\gamma) = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.41}$$

2.2.3 Successive Rotations

Other widely used descriptions of spatial rotations are constructed by three successive elementary rotations. Many choices for the rotation axes are possible, but successive rotations should be performed about different axes in order to maintain three rotary DoFs. This leaves twelve admissible choices out of the total of 27 ($3 \times 3 \times 3$).[9] Two of the most widely used sequences are xyz (Tait-Bryan or Cardan angles) and zxz (Eulerian angles).

Cardan Angles

For Cardan angles (xyz) the resulting rotation matrix is:

$$\begin{aligned} \boldsymbol{A}_{BI} &= \boldsymbol{A}_z(\gamma)\boldsymbol{A}_y(\beta)\boldsymbol{A}_x(\alpha) \\ &= \begin{pmatrix} \mathrm{c}\gamma\,\mathrm{c}\beta & \mathrm{s}\gamma\,\mathrm{c}\alpha + \mathrm{c}\gamma\,\mathrm{s}\beta\,\mathrm{s}\alpha & \mathrm{s}\gamma\,\mathrm{s}\alpha - \mathrm{c}\gamma\,\mathrm{s}\beta\,\mathrm{c}\alpha \\ -\mathrm{s}\gamma\,\mathrm{c}\beta & \mathrm{c}\gamma\,\mathrm{c}\alpha - \mathrm{s}\gamma\,\mathrm{s}\beta\,\mathrm{s}\alpha & \mathrm{c}\gamma\,\mathrm{s}\alpha + \mathrm{s}\gamma\,\mathrm{s}\beta\,\mathrm{c}\alpha \\ \mathrm{s}\beta & -\mathrm{c}\beta\,\mathrm{s}\alpha & \mathrm{c}\beta\,\mathrm{c}\alpha \end{pmatrix}. \end{aligned} \tag{2.42}$$

The Cardan angles α, β, γ can be calculated for a given rotation matrix by solving the trancendental equation (2.42). Elements $A_{BI,1,1}, A_{BI,2,1}$ directly yield γ:

$$\begin{aligned} \frac{\mathrm{s}\gamma}{\mathrm{c}\gamma} &= \frac{-A_{BI,2,1}}{A_{BI,1,1}}, \\ \Rightarrow \gamma &= \operatorname{atan2}\left(-A_{BI,2,1}, A_{BI,1,1}\right). \end{aligned} \tag{2.43}$$

Here the function $\operatorname{atan2}(y,x)$ is a variant of $\operatorname{atan}(y/x)$ that returns the correct angle between the positive x-axis and the point (x,y) for all values of x, y other than $x = y = 0$. The angle β follows from $A_{BI,3,1}$, $A_{BI,1,1}$ and $A_{BI,2,1}$:

$$\begin{aligned} \frac{A_{BI,3,1}}{\mathrm{c}\gamma A_{BI,1,1} - \mathrm{s}\gamma A_{BI,2,1}} &= \frac{\mathrm{s}\beta}{(\mathrm{c}\gamma)^2\,\mathrm{c}\beta + (\mathrm{s}\gamma)^2\,\mathrm{c}\beta} \\ \Rightarrow \beta &= \operatorname{atan2}\left(A_{BI,3,1}, \mathrm{c}\gamma A_{BI,1,1} - \mathrm{s}\gamma A_{BI,2,1}\right). \end{aligned} \tag{2.44}$$

9. For the first axis there are no restrictions (3 choices), the second must differ from the first (2 choices) and the third from the second (2 choices), resulting in $3 \times 2 \times 2 = 12$ parametrizations.

Finally, α is given by:

$$\begin{aligned} &\frac{A_{BI,3,2}}{A_{BI,3,3}} = \frac{-\mathrm{c}\,\beta\,\mathrm{s}\,\alpha}{\mathrm{c}\,\beta\,\mathrm{c}\,\alpha}, \\ \Rightarrow &\alpha = \mathrm{atan2}(-A_{BI,3,2}, A_{BI,3,3}). \end{aligned} \tag{2.45}$$

For successive rotations the angular velocity of the rotating body (or frame) can be calculated from the rotation matrix and its time derivative via (2.21), that is, $\boldsymbol{A}_{IB}^T \dot{\boldsymbol{A}}_{IB} = {}_B\widetilde{\boldsymbol{\omega}}_{IB}$. Alternatively the angular velocities induced by the elementary rotations may simply be added, since angular velocity is a vector quantity. However, the contributions $\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{\beta}}, \dot{\boldsymbol{\gamma}}$ must be transformed into a common reference frame first:

$$\begin{aligned} {}_B\boldsymbol{\omega}_{IB} &= {}_B\dot{\boldsymbol{\alpha}} + {}_B\dot{\boldsymbol{\beta}} + {}_B\dot{\boldsymbol{\gamma}} && (2.46)\\ &= \boldsymbol{A}_z(\gamma)\boldsymbol{A}_y(\beta)\boldsymbol{e}_x\dot{\alpha} + \boldsymbol{A}_z(\gamma)\boldsymbol{e}_y\dot{\beta} + \boldsymbol{e}_z\dot{\gamma} && (2.47)\\ &= \begin{pmatrix} \cos(\gamma)\cos(\beta) & \sin(\gamma) & 0 \\ -\sin(\gamma)\cos(\beta) & \cos(\gamma) & 0 \\ \sin(\beta) & 0 & 1 \end{pmatrix} \dot{\boldsymbol{\varphi}}, \qquad \boldsymbol{\varphi} := \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}. && (2.48) \end{aligned}$$

The inverse mapping is easily obtained by matrix inversion:

$$\dot{\boldsymbol{\varphi}} = \frac{1}{\cos(\beta)} \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma)\cos(\beta) & \cos(\gamma)\cos(\beta) & 0 \\ -\cos(\gamma)\sin(\beta) & \sin(\gamma)\sin(\beta) & \cos(\beta) \end{pmatrix} {}_B\boldsymbol{\omega}_{IB}\,. \tag{2.49}$$

The result clearly indicates a singular configuration at $\beta = \frac{\pi}{2} + k\pi$, $k \in \mathbb{Z}$: evaluating (2.49) requires a division by zero and evaluation of (2.45) and (2.43) evaluation of $\mathrm{atan2}(0,0)$. The geometrical explanation is that the third rotation about the z-axis is parallel or anti-parallel to the first about the x-axis, yielding one rotation by a total angle of $\alpha \pm \gamma$.

Eulerian Angles

The rotation matrix for Eulerian angles is also obtained from three elementary rotations: by ψ about $\boldsymbol{e}_z$, by ϑ about the new $\boldsymbol{e}_x$ and finally by ϕ about the new $\boldsymbol{e}_z$:

$$\begin{aligned} \boldsymbol{A}_{BI} &= \boldsymbol{A}_z(\phi)\boldsymbol{A}_x(\vartheta)\boldsymbol{A}_z(\psi) \\ &= \begin{pmatrix} \mathrm{c}\,\phi\,\mathrm{c}\,\psi - \mathrm{s}\,\phi\,\mathrm{c}\,\vartheta\,\mathrm{s}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\psi + \mathrm{s}\,\phi\,\mathrm{c}\,\vartheta\,\mathrm{c}\,\psi & \mathrm{s}\,\phi\,\mathrm{s}\,\vartheta \\ -\mathrm{s}\,\phi\,\mathrm{c}\,\psi - \mathrm{c}\,\phi\,\mathrm{c}\,\vartheta\,\mathrm{s}\,\psi & -\mathrm{s}\,\phi\,\mathrm{s}\,\psi + \mathrm{c}\,\phi\,\mathrm{c}\,\vartheta\,\mathrm{c}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\vartheta \\ \mathrm{s}\,\vartheta\,\mathrm{s}\,\psi & -\mathrm{s}\,\vartheta\,\mathrm{c}\,\psi & \mathrm{c}\,\vartheta \end{pmatrix}. \end{aligned} \tag{2.50}$$

The Eulerian angles can again be calculated from a given rotation matrix by solving (2.50), yielding:

$$\psi = \operatorname{atan2}(-A_{BI,3,1}, A_{BI,3,2}), \tag{2.51}$$
$$\vartheta = \operatorname{atan2}(A_{BI,3,1}\,\mathrm{s}\,\psi - A_{BI,3,2}\,\mathrm{c}\,\psi, A_{BI,3,3}), \tag{2.52}$$
$$\phi = \operatorname{atan2}(A_{BI,1,3}, A_{BI,2,3}). \tag{2.53}$$

Note that there are several expressions for the solution to the inverse problem, since the system of equations is overdetermined, for example, ϑ could also be calculated from $A_{BI,1,3}$ and $A_{BI,2,3}$ instead of $A_{BI,3,1}$ and $A_{BI,3,2}$.

The angular velocity is again obtained from the vector sum of the individual contributions and the inverse mapping by matrix inversion:

$$\begin{aligned}
{}_B\boldsymbol{\omega}_{IB} &= {}_B\dot{\boldsymbol{\psi}} + {}_B\dot{\boldsymbol{\vartheta}} + {}_B\dot{\boldsymbol{\phi}}\,, && (2.54)\\
&= \boldsymbol{A}_z(\phi)\boldsymbol{A}_x(\vartheta)\boldsymbol{e}_z\dot{\phi} + \boldsymbol{A}_z(\phi)\boldsymbol{e}_x\dot{\vartheta} + \boldsymbol{e}_z\dot{\psi}, && (2.55)\\
&= \begin{pmatrix} \sin(\phi)\sin(\vartheta) & \cos(\phi) & 0 \\ \cos(\phi)\sin(\vartheta) & -\sin(\phi) & 0 \\ \cos(\vartheta) & 0 & 1 \end{pmatrix}\dot{\boldsymbol{\varphi}}, \qquad \dot{\boldsymbol{\varphi}} := \begin{pmatrix} \dot{\psi} \\ \dot{\vartheta} \\ \dot{\phi} \end{pmatrix}, && (2.56)\\
\Rightarrow \quad \boldsymbol{\varphi} &= \frac{1}{\sin(\vartheta)} \begin{pmatrix} \sin(\phi) & \cos(\phi) & 0 \\ \cos(\phi)\sin(\vartheta) & -\sin(\phi)\sin(\vartheta) & 0 \\ -\sin(\phi)\cos(\vartheta) & -\cos(\vartheta)\cos(\phi) & \sin(\vartheta) \end{pmatrix} {}_B\boldsymbol{\omega}_{IB}. && (2.57)
\end{aligned}$$

For Eulerian angles the singular case obviously occurs for $\vartheta = k\pi$, $k \in \mathbb{Z}$, when the first and last rotation are about parallel or anti-parallel axes. This is again clear both from the inverse solution for $\dot{\boldsymbol{\varphi}}$ ((2.57): division by zero) and $\boldsymbol{\varphi}$ ((2.53) and (2.51): atan2(0, 0)).

2.2.4 Discussion

In this section we developed a number of different parametrizations of spacial rotations using either three or four parameters. The minimal representations using three parameters are simpler to use in most calculations, since for redundant representations additional constraints, such as the unit length of the axis of rotation, must be taken into account, for example, during the time integration of Equations of Motion (EoMs). On the other hand, minimal representations always have singular configurations. In many technological applications, however, the motion of the individual bodies is limited. In such cases, a proper sequence of elementary rotations can put the singularities outside the physical range of motion.

Besides the parametrizations shown in this section, there is a range of other, more

or less widely used parametrizations, such as Euler parameters[10] (Shabana 2005), Rodriguez parameters[11] (Shabana 2005) and Quaternions[12] (Siciliano et al. 2009). An extreme case of a redundant parametrization is the direct use of the rotation matrix (or basis vectors of the body fixed frame) together with the constraints (2.7) ensuring that the matrix remains orthonormal (see e.g., Betsch and Leyendecker 2006).

2.3 Multibody Kinematics

This section develops the kinematic equations of rigid multibody systems that are used in subsequent chapters.

2.3.1 Coordinate Systems, Constraints and Coordinate Sets

To simplify the following discussion, we will number the bodies in the Multibody System (MBS) from $1 \dots n_b$ and introduce one body-fixed frame $\mathscr{F}_i$ for each body i in the system and denote the inertial frame by $I = \mathscr{F}_0$. The pose of the i-th body is then given by the origin $\boldsymbol{r}_i = \boldsymbol{r}_{O_i}$ of $\mathscr{F}_i$ and the orientation $\boldsymbol{A}_{i0}$ of $\mathscr{F}_i$, which may be parametrized by $\boldsymbol{\varphi}_i \mathbb{R}^3$. The configuration of the whole MBS is then simply given by the *system coordinates*:[13]

$$\boldsymbol{z}^T = \begin{pmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \dots & \boldsymbol{r}_{n_b} & \boldsymbol{\varphi}_1 & \boldsymbol{\varphi}_2 & \dots & \boldsymbol{\varphi}_{n_b} \end{pmatrix} \in \mathbb{R}^{6n_b}. \tag{2.58}$$

For robotic systems, the individual bodies are generally connected by joints, limiting the physically feasible set of system coordinates. Figure 2.5 shows a simple example of a robotic manipulator with one rotary and one prismatic joint. This type of *holonomic* (or geometric) constraint only depends on position variables and may be written as

$$\boldsymbol{\Phi}(\boldsymbol{z}, t) = \boldsymbol{0} \in \mathbb{R}^m. \tag{2.59}$$

10. The Euler parameters $\boldsymbol{\vartheta}$ are a redundant parametrization related to axis-angle parameters via

$$\vartheta_0 = \cos(\frac{\varphi}{2}), \;\; \vartheta_1 = u_1 \sin(\frac{\varphi}{2}), \;\; \vartheta_2 = u_2 \sin(\frac{\varphi}{2}), \;\; \vartheta_3 = u_3 \sin(\frac{\varphi}{2}), \;\; \|\boldsymbol{\vartheta}\| = 1$$

11. The Rodriguez parameters $\boldsymbol{\gamma}$ are a minimal representation related to Euler parameters $\boldsymbol{\vartheta}$ via

$$\gamma_1 = \frac{\theta_1}{\theta_2}, \;\; \gamma_2 = \frac{\theta_2}{\theta_0}, \;\; \gamma_3 = \frac{\theta_3}{\theta_0}$$

12. Quaternions are a redundant parametrization $q_1, \boldsymbol{q}$ related to axis-angle parameters via

$$q_1 = \cos(\frac{\varphi}{2}), \;\; \boldsymbol{q} = \sin(\frac{\varphi}{2})\boldsymbol{u}$$

13. For planar systems the number of system coordinates becomes $\dim(\boldsymbol{z}) = 3n_b$

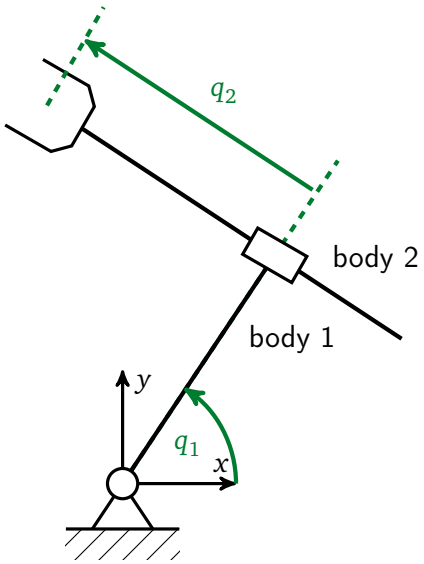


Figure 2.5: Example of a robotic manipulator as MBS with constraints. The two bodies and the environment are connected by one rotary and one prismatic joint. There are twelve system coordinates, position and orientation of body 1 and body 2. Due to the constraints (three position and two orientation constraints at joint 1, two position and three orientation constraints at joint 2) only the two DoFs q_1 and q_2 remain.

If time appears explicitly in the constraint equation, it is called *rheonomic*, otherwise *scleronomic*. Systems with constraints including velocities are called *nonholonomic* and may be written as:

$$\boldsymbol{\Phi}(\boldsymbol{z}, \dot{\boldsymbol{z}}, t) = \mathbf{0} \in \mathbb{R}^m. \tag{2.60}$$

Holonomic and scleronomic constraints are by far the most prevalent for robotic systems. A notable exception, however, are nonholonomic constraints introduced by wheels in mobile robots. Here the velocity of the vehicle is constrained to move in the direction the wheels are steering towards, while there is no such constraint for the position of the vehicle. In other words, there are additional restrictions for the feasible velocities beyond those imposed by geometric constraints.

When the system's configuration is described by the $6n_b$ variables $\boldsymbol{z}$, the constraint equation (2.59) must be explicitly taken into account to ensure feasible configurations (cf. section 3.4). In many cases it is possible to identify a set of *minimal coordinates*, that is, a minimal set of position parameters that (together with fixed parameters) is sufficient for describing the configuration of a system (i.e., the system coordinates $\boldsymbol{z}$). For minimal coordinates the constraints are automatically fulfilled: $\boldsymbol{\Phi}(\boldsymbol{z}(\boldsymbol{q})) \equiv \mathbf{0}$. The term *generalized coordinates* is also used for minimal coordinates, even though it should be understood to include non-minimal sets of coordinates.

2.3.2 Topology

Kinematic Trees

The topology of many robotic systems resembles a tree. For such *tree structured* systems every body has exactly one *predecessor* (or *parent*) and an arbitrary number of *successors* (or *children*). The only exception is the *root* of the tree, which does not have a predecessor. Obviously, the kinematics of the i-th body only depend on the motion of the predecessor and the relative motion between parent and child body, as described by the relative DoF $\boldsymbol{q}_i$. The topology of a tree is uniquely defined by the list of bodies with their corresponding parents. This list is easily converted into a more useful representation of the kinematic tree based on references to parent and child bodies for each body in the system. Pseudocode for building a graph representation for an implementation in a C++-like language is given in algorithm 1, figure 2.6 shows an example of a kinematic tree listing parent and child bodies and figure 2.7 illustrates the local description of a tree using references to neighboring bodies. Other approaches using matrix representations of the graph may be found in (Bremer 2008, pp. 50) and references therein.

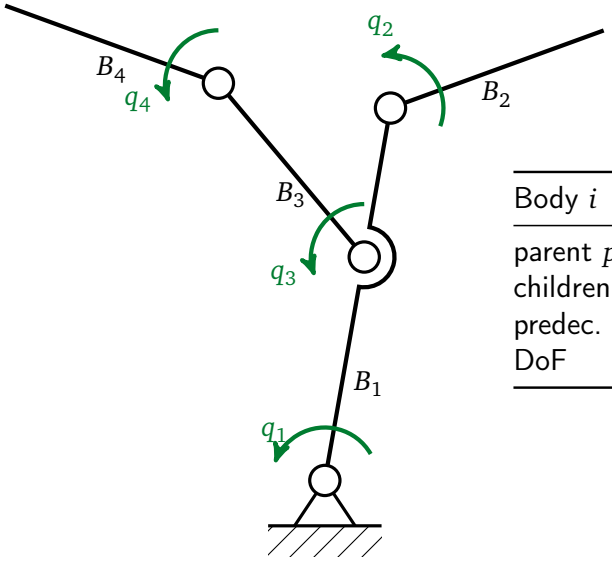
Algorithm 1: Building the kinematic tree from a list of bodies and parents. For this algorithm, each body should contain references (pointers) to parent and child nodes. The MBS object contains a reference to the root of the tree (cf. Yamane 2004; Buschmann et al. 2006; Buschmann 2010).

```
input  : list of bodies with parents as pair (body, parent)
output : Reference-based graph representation of kinematic tree
forall the pairs (body, parent) in list do
    b= get_node(body) ;              // get reference from body name
    if parent not empty then
        p= get_node(parent) ;       // get reference from parent name
        b.set_parent(p) ;                        // set parent of body
        p.add_child(b) ;              // add body as child of parent
    else
        root=b ;                 // no parent ⇒ this is the root node
```

A special case of a tree is a kinematic chain, where each body has one successor and one predecessor. Systems that cannot be described by a tree have *kinematic loops*, their structure is described by a general graph. Figure 2.8 illustrates these topological classes.

Systems with Loops

Because a tree structure enables many optimizations of kinematics and dynamics calculations, it is common to treat even general systems with loops as trees. This is



Body i	B_1	B_2	B_3	B_4
parent $p(i)$	$\emptyset$	B_1	B_1	B_3
children (N_{Ci})	$\{B_2, B_3\}$	$\emptyset$	B_4	$\emptyset$
predec. (P_i)	$\emptyset$	$\{B_1\}$	$\{B_1\}$	$\{B_3, B_1\}$
DoF	q_1	q_2	q_3	q_4

Figure 2.6: Example of a kinematic tree. The table on the right lists the parent/child relationships and DoFs, when B_1 is chosen as the root node. The set N_{Ci} contains all children of B_i.

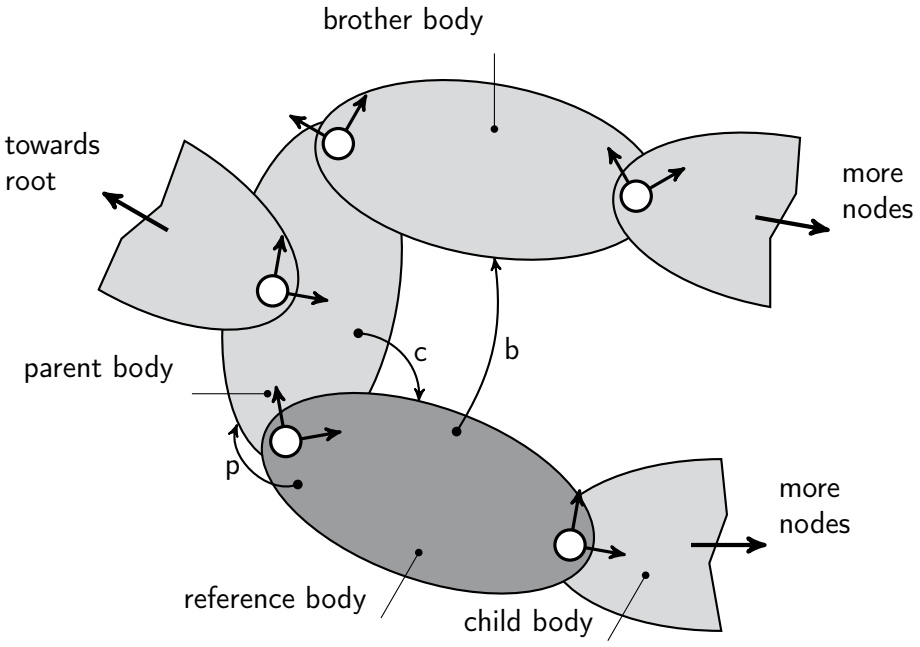


Figure 2.7: Local description of a kinematic tree using references to neighboring nodes (p: parent, c: child, b: brother; cf. Yamane (2004), Buschmann et al. (2006), and Buschmann (2010); figure redrawn from Buschmann 2010).

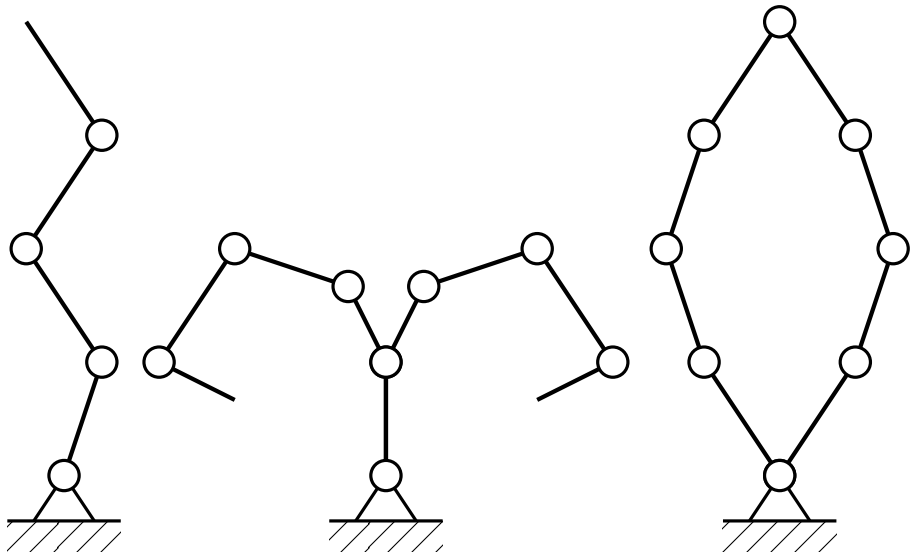


Figure 2.8: Topological classes: chain (left), tree (center) and graph (right).

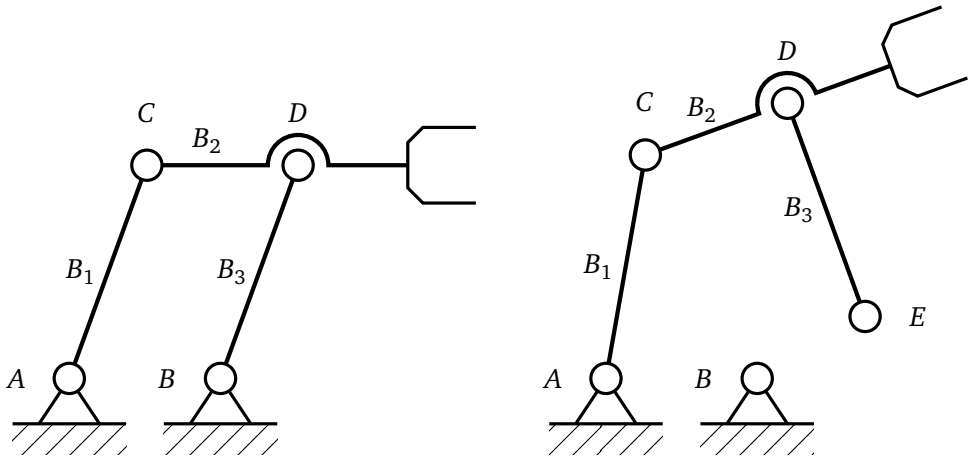


Figure 2.9: Illustration of how systems with loops can be treated as tree structured systems by cutting all loops open and adding algebraic constraints. The original system (left) consists of three bodies (B_1, B_2, B_3), forming the kinematic loop *A-C-D-B-A*. The system may be cut at B, generating an open kinematic chain *A-C-D-E* (the *spanning tree*). The removed joint at B is modeled by a holonomic constraint $\Phi(\boldsymbol{z}) = \left(\boldsymbol{r}_E^T - \boldsymbol{r}_B^T, \alpha_{EB}, \beta_{EB}\right) = \mathbf{0}$, where α_{EB}, β_{EB} are the relative out-of-plane angles between B_3 and the environment.

Symbol	Definition
$p(i)$ or p	parent of the i-th node
$c(i)$ or c	child of the i-th node
$b(i)$ or n	brother of the i-th node
P_i	list of all predecessors of the i-th node
N_B	set of all body nodes in the MBS
N_S	set of all subsystem nodes in the MBS
N_{Ci}	set of all children of the i-th node: $N_{Ci} = \{n \in N_B \mid \ p(n) = i\}$
N_{CSi}	set of all subsystems, that are children of i: $N_{CSi} = \{n \in \overline{N}_S \mid p(n) = i\}$
N_{Si}	set of all bodies in the i-th subsystem
C_{Si}	child nodes of nodes in N_{Si}, that are not themselves in N_{Si}: $C_{Si} = \{n \in N_B \setminus N_{Si} \mid \ p(n) \in N_{Si}\}$
N_{PSi}	parents of nodes in C_{Si}: $N_{PSi} = \{n \in N_{Si} \mid \ c(n) \in C_{Si}\}$
$\overline{N}_S$	set of all bodies that are part of a subsystem (union of all N_{Si})

Table 2.1: Notation for describing the topology of MBSs with and without loops (for the definition of parent/child/brother references see figure 2.7 and 2.11.). When it is clear from the context, the index in parentheses for $p(i), b(i)$ and $c(i)$ is omitted.

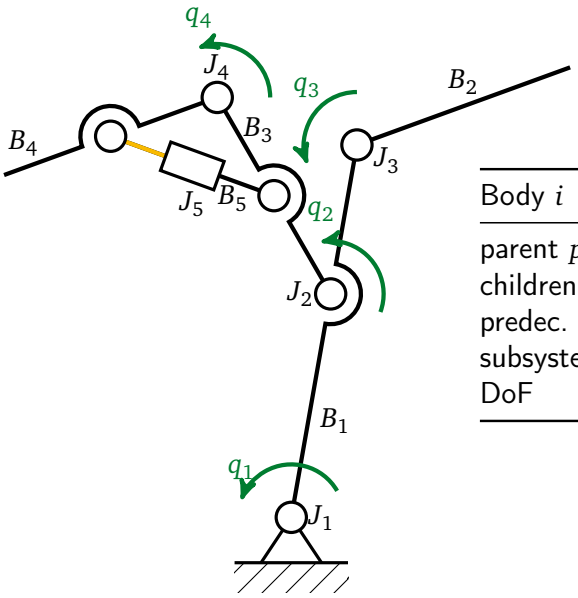
done by cutting the loops, thereby formally generating a tree structure, the *spanning tree*. Joints within the spanning tree are called *primary joints*. The joints that have been cut (*secondary joints*) are replaced by kinematic constraints $\boldsymbol{\Phi}(\boldsymbol{q}) = \mathbf{0}$ (cf. section 3.4). Figure 2.9 illustrates the procedure.

For robots, it is often possible to describe the configuration of all bodies in a loop as a function of the parent body's configuration and a set of minimal coordinates $\boldsymbol{q}_{Si}$ (Si denotes the i-th *subsystem* containing the loop, cf. section 3.4.2). In that case, the kinematics of all bodies within the subsystem depend on $\boldsymbol{q}_{Si}$ and the kinematics of the parent body (see figure 2.10 for an example and figure 2.11 for the local description of the kinematic structure). We can divide the loop into a *primary branch* and a *secondary branch*, and choose the relative DoFs $\boldsymbol{q}_i$ of the primary branch as subsystem DoFs $\boldsymbol{q}_{Si}$. The configuration of the bodies in the secondary branch of the loop are then computed using problem-specific equations.[14] This is often simplified by using results from the computations in the primary branch. The procedure is illustrated in figure 2.10.

Coordinate Frames and Minimal Coordinates

For robotic systems it is common to choose the body-fixed frames such that the z-axis of the i-th body is aligned with the axis of the joint connecting it to its parent (cf. Denavit-Hartenberg Convention in appendix A.3). For a kinematic tree with n_b bodies

14. The inverse kinematics for the loop could also be solved numerically, but this is not discussed here (also see section 3.4).



Body i	B_1	B_2	B_3	B_4	B_5
parent $p(i)$	$\emptyset$	B_1	B_1	B_3	B_3
children (N_{Ci})	$\{B_2, B_3\}$	$\emptyset$	B_4	$\emptyset$	$\emptyset$
predec. (P_i)	$\emptyset$	$\{B_1\}$	$\{B_1\}$	$\{B_3, B_1\}$	
subsystem	$\emptyset$	$\emptyset$	$\emptyset$	S_1	S_1
DoF	q_1	q_2	q_3	q_4	q_4

Figure 2.10: The system from figure 2.6 with an additional body that introduces a closed loop. The kinematics are described by the spanning tree, which is generated by cutting the loop at the yellow link. The relative DoFs in the *primary branch* are chosen as subsystem DoFs $\boldsymbol{q}_{Si}$. In the example, the primary branch in the loop consists of only B_4, while the secondary branch consists of B_5. The kinematics of the second branch (here: B_5) are computed as a function of $\boldsymbol{q}_{Si}$, possibly using results of the forward kinematics of the primary branch (here: B_4). There is only one subsystem consisting of the coupled bodies B_4 and B_5.

and n_q DoFs, a set of minimal coordinates $\boldsymbol{q}$ can be constructed from the variables $\boldsymbol{q}_i$ describing the relative motion between the i-th body and its predecessor:

$$\boldsymbol{q}^T = \left(\boldsymbol{q}_1^T, \boldsymbol{q}_2^T, \ldots, \boldsymbol{q}_{n_b}^T\right) \in \mathbb{R}^{n_q}. \tag{2.61}$$

To enable subsequent optimizations, the bodies in the tree should be numbered such that the index of a body closer to the root will always have a lower index. For systems with loops, the minimal coordinates may be constructed from the spanning tree, using only the primary branches. The motion of the bodies in the secondary branch depend on all relative DoFs in the primary branch (see figure 2.6 for trees and figure 2.10 for systems with loops).

Recursive Kinematics Calculations

An essential characteristic of a kinematic tree is the fact that the motion of the i-th body *only* depends on that of its parent and the relative DoFs $\boldsymbol{q}_i$. The kinematic quantities

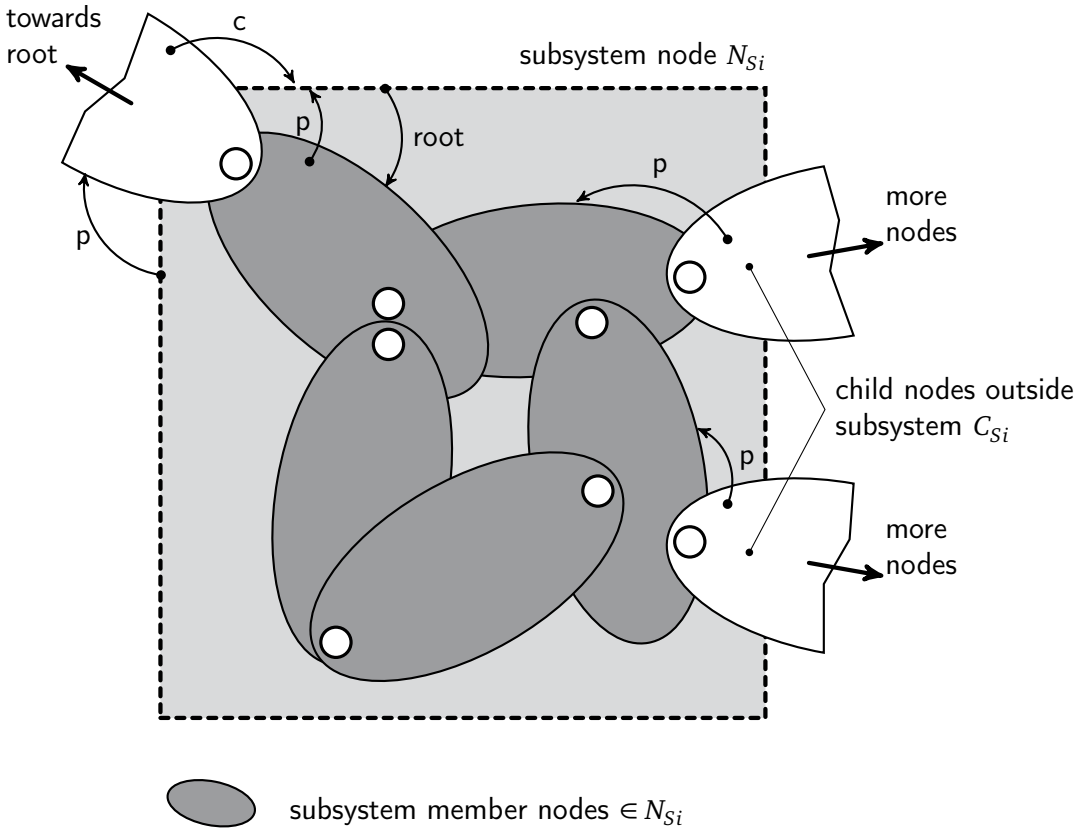


Figure 2.11: Local description of a system with loops using references to neighboring nodes and subsystems encapsulating loops (p: parent, c: child, b: brother, adapted from Schwienbacher (2013)). At the level of a full MBS, a subsystem is treated as a body, just as, e.g., bodies with one, two or three DoFs. At the same time, a subsystem is a small MBS, inheriting methods from full MBSs for calculating kinematics and dynamics. Programmatically, this is done in C++ via multiple inheritance.

of the system may therefore be calculated recursively starting at the root node, as illustrated in algorithm 2. This forward recursion requires recursive equations for the kinematic quantities. For the transform matrix this equation becomes:

$$\boldsymbol{A}_{0i} = \boldsymbol{A}_{0p}\boldsymbol{A}_{pi}, \quad i = 2 \dots n_b. \tag{2.62}$$

The relative rotation $\boldsymbol{A}_{pi}$ from parent to child is a function of the relative coordinates $\boldsymbol{q}_i$. In robotic systems it is usually composed of a constant transformation aligning the z-axis with the axis of rotation, followed by an additional elementary rotation about the z-axis in the case of rotary joints.

According to figure 2.12 and (2.12), the origin of the i-th body-fixed frame $\boldsymbol{r}_i$ is

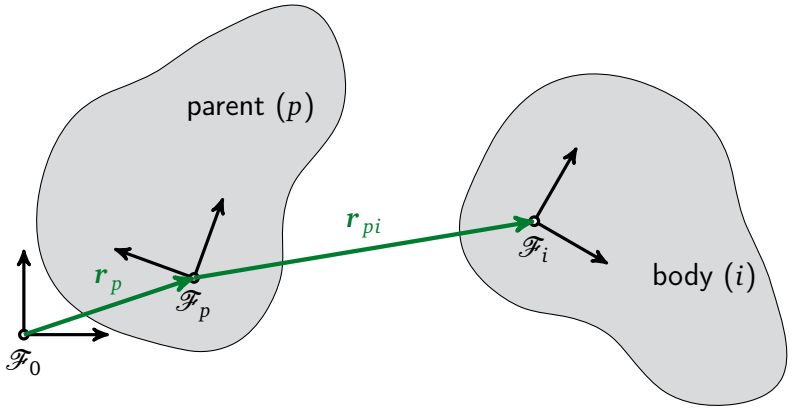


Figure 2.12: Coordinate systems and relative kinematics for body i in a tree structured MBS.

Algorithm 2: Calculate kinematic quantities by forward recursion.

input : Minimal coordinates $\boldsymbol{q}$ (and derivatives for velocities and accelerations)
output: Kinematic quantities for all bodies
for $i = 1$ **to** n_b **do**
  Calculate kinematics for i-th body from that of its parent body $p(i)$ and relative DoFs $\boldsymbol{q}_i$.

given by

$$ {}_i\boldsymbol{r}_i = \boldsymbol{A}_{ip}\left({}_p\boldsymbol{r}_p + {}_p\boldsymbol{r}_{pi}\right), \tag{2.63} $$

where $p = p(i)$ denotes the parent of the i-th body and $\boldsymbol{r}_{pi}$ is the vector from O_p to O_i. For the root body ($i = 1$) this equation is simply replaced by

$$ {}_0\boldsymbol{r}_0 = {}_0\boldsymbol{r}_{01}(\boldsymbol{q}_1), \tag{2.64} $$

where $\boldsymbol{r}_{01}(\boldsymbol{q}_1)$ is a function depending on the coupling with the environment. For a free floating reference, $\boldsymbol{r}_{01}$ is simply a subset of the DoFs $\boldsymbol{q}_1$ of the root body, if the body is connected via a rotary joint, $\boldsymbol{r}_{01}$ is constant.

The angular velocity of the i-th body is calculated from (2.62) and (2.22):

$$\begin{aligned} {}_i\widetilde{\boldsymbol{\omega}}_i &= \left(\boldsymbol{A}_{0p}\boldsymbol{A}_{pi}\right)^T \tfrac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{A}_{0p}\boldsymbol{A}_{pi}\right) \\ &= \boldsymbol{A}_{pi}^T\boldsymbol{A}_{0p}^T\dot{\boldsymbol{A}}_{0p}\boldsymbol{A}_{pi} + \boldsymbol{A}_{pi}^T\boldsymbol{A}_{0p}^T\boldsymbol{A}_{0p}\dot{\boldsymbol{A}}_{pi} \\ &= {}_i\widetilde{\boldsymbol{\omega}}_{0p} + {}_i\widetilde{\boldsymbol{\omega}}_{pi}, \\ \Rightarrow {}_i\boldsymbol{\omega}_i &= \boldsymbol{A}_{ip}\left({}_p\boldsymbol{\omega}_{0p} + {}_p\boldsymbol{\omega}_{pi}\right) \\ &= \boldsymbol{A}_{ip}\left({}_p\boldsymbol{\omega}_{0p} + {}_p\boldsymbol{J}_{J\omega i}\dot{\boldsymbol{q}}_i\right). \end{aligned} \tag{2.65}$$

Here the final equation defines the recursive calculation of angular velocities. The relative angular velocity ${}_i\boldsymbol{\omega}_{pi}$ and the corresponding *joint Jacobian*[15] $\boldsymbol{J}_{J\omega}$ is easily calculated: For free floating bodies, the required equations depend on the parametrization of the rotation matrix (see section 2.2), for prismatic joints ${}_i\boldsymbol{\omega}_{pi} = \boldsymbol{0}$, $\boldsymbol{J}_{J\omega} = \boldsymbol{0}$ and for rotary joints ${}_i\boldsymbol{\omega}_{pi} = {}_i\boldsymbol{u}\,\dot{q}_i$, $\boldsymbol{J}_{J\omega} = \boldsymbol{u}$, where $\boldsymbol{u}$ is the joint axis and q_i the corresponding DoF. For $\boldsymbol{u} = \boldsymbol{e}_z$ we simply have ${}_i\boldsymbol{\omega}_{pi}{}^T = (0 \quad 0 \quad \dot{q}_i)$, ${}_i\boldsymbol{J}_{J\omega i} = \boldsymbol{e}_z$.

The linear velocity of the i-th body is directly obtained from (2.63):

$$\begin{aligned} {}_i\dot{\boldsymbol{r}}_i &= \boldsymbol{A}_{ip}\left({}_p\dot{\boldsymbol{r}}_p + {}_p\widetilde{\boldsymbol{\omega}}_{0p\,p}\boldsymbol{r}_{pi} + {}_p\mathring{\boldsymbol{r}}_{pi}\right) \\ &= \boldsymbol{A}_{ip}\left({}_p\dot{\boldsymbol{r}}_p + {}_p\widetilde{\boldsymbol{\omega}}_{0p\,p}\boldsymbol{r}_{pi} + {}_p\boldsymbol{J}_{Jvi}\dot{\boldsymbol{q}}_i\right). \end{aligned} \tag{2.66}$$

Here $\boldsymbol{J}_{Jvi}$ is the translational joint Jacobian. The p-frame is chosen for $\boldsymbol{r}_{pi}$, since this often simplifies the description (for rotary joints: ${}_p\boldsymbol{r}_{pi}$ is constant; for prismatic joints: ${}_p\boldsymbol{r}_{pi} = \boldsymbol{u}q_i$, with $\boldsymbol{u}$ the direction of motion and q_i the displacement). The equation for the root body again depends on the connection to the environment.

The accelerations are obtained by differentiation (or from (2.28)):

$$\begin{aligned} {}_i\dot{\boldsymbol{\omega}}_i &= \boldsymbol{A}_{ip}\left({}_p\dot{\boldsymbol{\omega}}_{0p} + {}_p\widetilde{\boldsymbol{\omega}}_{pi\,p}\boldsymbol{\omega}_{0p} + {}_p\dot{\boldsymbol{\omega}}_{pi}\right) \\ &= \boldsymbol{A}_{ip}\left({}_p\dot{\boldsymbol{\omega}}_{0p} + {}_p\widetilde{\boldsymbol{\omega}}_{pi\,p}\boldsymbol{\omega}_{0p} + {}_p\dot{\boldsymbol{J}}_{J\omega i}\dot{\boldsymbol{q}}_i + {}_p\boldsymbol{J}_{J\omega i}\ddot{\boldsymbol{q}}_i\right), \end{aligned} \tag{2.67}$$

$$\begin{aligned} {}_i\ddot{\boldsymbol{r}}_i &= \boldsymbol{A}_{ip}\left({}_p\ddot{\boldsymbol{r}}_p + \left({}_p\dot{\widetilde{\boldsymbol{\omega}}}_{0p} + {}_p\widetilde{\boldsymbol{\omega}}_{0p\,p}\widetilde{\boldsymbol{\omega}}_{0p}\right){}_p\boldsymbol{r}_{pi} + 2\,{}_p\widetilde{\boldsymbol{\omega}}_{0p\,p}\mathring{\boldsymbol{r}}_{pi} + {}_p\mathring{\mathring{\boldsymbol{r}}}_{pi}\right) \\ &= \boldsymbol{A}_{ip}\left({}_p\ddot{\boldsymbol{r}}_p + \left({}_p\dot{\widetilde{\boldsymbol{\omega}}}_{0p} + {}_p\widetilde{\boldsymbol{\omega}}_{0p\,p}\widetilde{\boldsymbol{\omega}}_{0p}\right){}_p\boldsymbol{r}_{pi} + 2\,{}_p\widetilde{\boldsymbol{\omega}}_{0p\,p}\dot{\boldsymbol{J}}_{Jvi}\dot{\boldsymbol{q}}_i + {}_p\boldsymbol{J}_{Jvi}\ddot{\boldsymbol{q}}_i\right). \end{aligned} \tag{2.68}$$

Recursive equations for other kinematic quantities are easily derived from these basic equations. For example, the gradient of $\boldsymbol{\omega}$ with respect to $\dot{\boldsymbol{q}}$ (the rotational Jacobian)

15. Note: *joint Jacobians* denote Jacobians for the relative motion in joints, their dimension depends on the dimension of the joint DoFs $\boldsymbol{q}_i$.

is given by

$$\begin{aligned} {}_i\boldsymbol{J}_{R,i} &:= \frac{\partial\, {}_i\boldsymbol{\omega}_i}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{A}_{ip}\frac{\partial\, {}_p\boldsymbol{\omega}_{0p}}{\partial \dot{\boldsymbol{q}}} + \frac{\partial\, {}_i\boldsymbol{\omega}_{pi}}{\partial \dot{\boldsymbol{q}}} \\ &= \boldsymbol{A}_{ip}\, {}_p\boldsymbol{J}_{R,p} + {}_i\boldsymbol{J}_{R,\mathrm{rel},i}\,, \end{aligned} \tag{2.69}$$

where $\boldsymbol{J}_{R,\mathrm{rel},i}$ is obtained by expanding $\boldsymbol{J}_{J\omega i}$ with zero entries, depending on the location of $\boldsymbol{q}_i$ in $\boldsymbol{q}$. A comprehensive listing of formulas for recursively calculating kinematic quantities for tree structured systems is given in appendix A.4. The appendix also includes explicit equations for all relative kinematic quantities for rotational joints, prismatic joints and free floating bodies. Examples for more complex, nonlinear joint kinematics found in the robot Lola are given in (Buschmann et al. 2006; Buschmann 2010; Schwienbacher 2013).

Compact Representation Using 6D-Vectors

For rigid-body kinematics and dynamics it is useful to combine rotational and translational components of velocities and accelerations into 6D-vectors.[16] Note that the 6D-representation is used only for writing equations in a compact fashion. For computer implementations the multiplications with zeros introduced by this notation should be avoided, essentially leading to the original 3D-representation. The compact representation for the recursive kinematics calculation of velocities is (cf. (2.66), (2.65)):

$$\boldsymbol{v}_i = \boldsymbol{C}_{ip}\boldsymbol{v}_p + \boldsymbol{J}_{Ji}\dot{\boldsymbol{q}}_i. \tag{2.70}$$

with : (2.71)

$$\boldsymbol{v}_i = \begin{pmatrix} {}_i\boldsymbol{\omega}_i \\ {}_i\dot{\boldsymbol{r}}_i \end{pmatrix}, \tag{2.72}$$

$$\boldsymbol{C}_{ip} = \begin{pmatrix} \boldsymbol{A}_{ip} & \boldsymbol{0} \\ \boldsymbol{A}_{ip}\, {}_p\widetilde{\boldsymbol{r}}_{pi}^{\,T} & \boldsymbol{A}_{ip} \end{pmatrix}, \tag{2.73}$$

$$\boldsymbol{J}_{Ji} = \begin{pmatrix} \boldsymbol{J}_{J\omega i} \\ \boldsymbol{J}_{Jvi} \end{pmatrix}. \tag{2.74}$$

The equations for accelerations are (cf. (2.68), (2.67)):

$${}_i\boldsymbol{a}_i = {}_i\dot{\boldsymbol{v}}_i = \boldsymbol{C}_{ip}\, {}_p\boldsymbol{a}_p + \boldsymbol{J}_{Ji}\ddot{\boldsymbol{q}}_i + \boldsymbol{b}_i, \tag{2.75}$$

with : (2.76)

16. Such a notation is also used in the theory of screws (going back to, e.g., Ball (1876)). More recently, the closely related concept of "spacial vector algebra" has been popularized by Featherstone (e.g., Featherstone 1983, 2008). Note, however, that we make no use of screw theory or spacial vector algebra, but instead simply stack two 3D vectors to obtain a more compact notation, similar to (Brandl et al. 1987).

$$ {}_i\boldsymbol{a}_i = \begin{pmatrix} \dot{\boldsymbol{\omega}}_i \\ \ddot{\boldsymbol{r}}_i \end{pmatrix}, \tag{2.77} $$

$$ \boldsymbol{b}_i = \begin{pmatrix} \boldsymbol{A}_{ip}\, {}_p\widetilde{\boldsymbol{\omega}}_{pi}\, {}_p\widetilde{\boldsymbol{\omega}}_{0p} + \dot{\boldsymbol{J}}_{J\omega i}\dot{\boldsymbol{q}}_i, \\ \boldsymbol{A}_{ip}\left({}_p\widetilde{\boldsymbol{\omega}}_{0p}\, {}_p\widetilde{\boldsymbol{\omega}}_{0p}\, {}_p\boldsymbol{r}_{pi} + 2\, {}_p\widetilde{\boldsymbol{\omega}}_{0p}\, {}_p\mathring{\boldsymbol{r}}_{pi} \right) + \dot{\boldsymbol{J}}_{Jvi}\dot{\boldsymbol{q}}_i \end{pmatrix}. \tag{2.78} $$

Note that the rate of change of the joint Jacobian $\dot{\boldsymbol{J}}_J$ is often assumed to be zero (see e.g., Gattringer 2011; Bremer 2008; Hippmann 2008). This is in fact true for prismatic and revolute joints, the most important cases for robots, but not for more complex mechanisms. The nonlinear drive mechanisms used for the ankle and knee joints in the robot Lola are an example (see figure 2.13; kinematics equations may be found in Buschmann et al. (2006), Buschmann (2010), and Schwienbacher (2013)).

For the important cases of prismatic joints and hinge joints, the joint Jacobians have the form of simple Boolean matrices:

$$ \text{hinge:} \quad \boldsymbol{J}_J = \frac{\partial}{\partial \dot{\boldsymbol{q}}_i} \begin{pmatrix} \boldsymbol{e}_z \dot{\boldsymbol{q}}_i \\ \boldsymbol{0} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T, \tag{2.79} $$

$$ \text{prismatic:} \quad \boldsymbol{J}_J = \frac{\partial}{\partial \dot{\boldsymbol{q}}_i} \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{e}_z \dot{\boldsymbol{q}}_i \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T. \tag{2.80} $$

Similar to the *joint Jacobian*, we can define a *global Jacobian* for the 6D-motion of the i-th body according to (2.66) and (2.65):

$$ \begin{aligned} {}_i\boldsymbol{J}_i &= \frac{\partial}{\partial \dot{\boldsymbol{q}}} \begin{pmatrix} \boldsymbol{\omega}_i \\ \dot{\boldsymbol{r}}_i \end{pmatrix}_i = \begin{pmatrix} \boldsymbol{A}_{ip}\, {}_p\boldsymbol{J}_{R,p} + {}_i\boldsymbol{J}_{R,rel,i} \\ \boldsymbol{A}_{ip}\left({}_p\boldsymbol{J}_{T,p} + {}_p\widetilde{\boldsymbol{r}}^T_{pi}\, {}_p\boldsymbol{J}_{R,p} \right) + {}_i\boldsymbol{J}_{T,rel,i} \end{pmatrix}, \\ &= \boldsymbol{C}_{ip}\boldsymbol{J}_p + \bar{\boldsymbol{J}}_{Ji}. \end{aligned} \tag{2.81} $$

The Jacobian $\bar{\boldsymbol{J}}_{Ji} \in \mathbb{R}^{6\times n_q}$ consists of the columns of the joint Jacobian $\boldsymbol{J}_{Ji}$ (cf. (2.74), (2.65), (2.66)) at the corresponding position of $\boldsymbol{q}_i$ in $\boldsymbol{q}$ and zeros everywhere else. The relative Jacobians $\boldsymbol{J}_{R,rel,i}, \boldsymbol{J}_{T,rel,i}$ may be constructed from $\boldsymbol{J}_{J\omega i}, \boldsymbol{J}_{Jvi}$ in the same fashion. With this the *global Jacobian* $\boldsymbol{J}_i$ can be written as:

$$ \begin{aligned} \boldsymbol{J}_i &= \underbrace{\boldsymbol{C}_{ip(i)}\boldsymbol{C}_{p(i)p(p(i))}\ldots\boldsymbol{C}_{X1}}_{=:\boldsymbol{C}_{i1}}\bar{\boldsymbol{J}}_{J1} + \boldsymbol{C}_{i2}\bar{\boldsymbol{J}}_{J2} + \cdots + \boldsymbol{C}_{ip(i)}\bar{\boldsymbol{J}}_{p(i)} + \bar{\boldsymbol{J}}_{Ji} \\ &= \begin{pmatrix} \boldsymbol{C}_{i1}\boldsymbol{J}_{J1} & \boldsymbol{C}_{i2}\boldsymbol{J}_{J2} & \ldots & \boldsymbol{J}_{Ji} & \boldsymbol{0} & \ldots \end{pmatrix}, \end{aligned} \tag{2.82} $$

where X in $\boldsymbol{C}_{X1}$ denotes the child of body 1 on the branch towards body i.

2.4 Efficient Forward Kinematics Calculation

Forward kinematics are a basic ingredient for many algorithms in robot dynamics and control, making efficient implementations very useful. The recursive calculation scheme outlined above is efficient when a kinematic quantity should be calculated for all bodies in the MBS, since results for one body are reused for its successors. The basic recursive approach is due to Vukobratović and Potkonjak (1979). In this section, achievable efficiency improvements are illustrated using the example of calculating the rotational Jacobian. The special case when the Jacobian is needed only for the end-effector of a serial manipulator with prismatic and rotational joints is analyzed in detail by Orin and Schrader (1984). In such a case, methods that do not calculate the Jacobians for all bodies are more efficient, reaching linear complexity in the number of bodies and roughly halving the calculation time for a 7-DoF manipulator (Orin and Schrader 1984, table 2). In the following, we will only the case where the Jacobian (or another kinematic quantity) is needed for all bodies.[17]

There are a number of possible efficiency improvements to the basic recursive scheme. Here we will only discuss exploiting the tree structure to reduce the required number of additions and multiplications. Further reductions in the calculation times are possible through programming techniques exploiting the Single Instruction Multiple Data (SIMD) instruction sets available on modern Central Processing Units (CPUs)[18] and general methods for efficient numerical programming such as loop unrolling and reordering.[19]

The speedups achieved by exploiting the tree structure when calculating the rotational Jacobian (2.69) will be illustrated using three types of mechanisms. The first two are the synthetic Coil(n_b) and Dill(L) systems proposed as benchmark mechanisms by Featherstone (1999b). The third is a rigid-body model of the humanoid robot Lola (Buschmann et al. 2012b). The synthetic mechanisms can be generated for different integers n_b, L, enabling an experimental investigation of the scaling properties of the algorithms applied to them, while the model of Lola chosen here always has 30 DoFs (see figure 2.13).

The Coil(n_b) mechanism is an unbranched kinematic chain with n_b identical bodies connected by revolute joints. It is defined by the Denavit-Hartenberg parameters[20] $d_i = 0, a_i = 1/n_b, \alpha_i = \frac{5\pi}{n_b}$ for the i-th body (Featherstone 1999b).

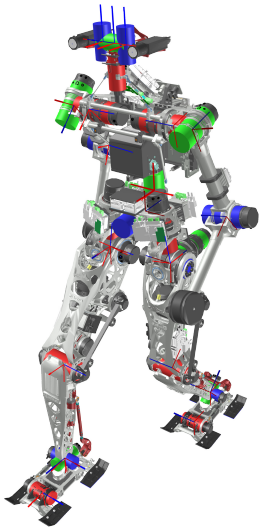
The Dill(L) mechanism is a self-similar kinematic tree consisting of $n_b = 2^L$ bodies connected by revolute joints. Dill(0) is a single cylindrical 0.1 m long body. Dill(L)

17. This is motivated by the fact that advanced motion generation and control methods often require Jacobians for all bodies in the system, e.g., for collision avoidance (see chapter 5).

18. Such as Streaming SIMD Extensions (SSE), Streaming SIMD Extensions 2 (SSE2), Advanced Vector Extensions (AVX), Advanced Vector Extensions 2 (AVX2) on Intel® CPUs (see Intel Corporation 2011).

19. For an overview of optimization strategies for numerical programs see (Goedecker and Hoisie 2001)

20. For a definition of the Denavit-Hartenberg parameters, see appendix A.3.



Total number of bodies	49
Number of drives	24
Number of drive-bodies	24
Active leg joints	2×7
Active arm joints	2×3
Active pelvis joints	2
Active head joints	2
Total height	$\approx$1.8 m
Total weight	$\approx$60 kg

Figure 2.13: Model of the humanoid robot Lola as an example of a system with moderate branching factor. Computer rendering of the robot model with joints additionally shown as cylinders (left). Red, green and blue lines show body-fixed x, y and z axes. Basic parameters of the model are listed in the table (right). An overview of the design is given in (Buschmann et al. 2012b), details in (Lohmeier 2010). A description of different MBS models of the machine may be found in (Buschmann 2010).

consists of n_b subtrees, all attached to a Dill(0) and scaled by L. The i-th subtree is a Dill$(i-1)$, shifted along and rotated about the x-axis by $0.1n_b$ and $\frac{i\pi}{3}$, respectively (Featherstone 1999b). Figure 2.14 shows examples of different Coil and Dill mechanisms.

The two synthetic mechanisms are extreme examples of unbranched and highly branched systems, while the humanoid robot is an example of an actual technical system with a moderate branching factor.

We have already observed that the kinematics of any body in the system only depends on the predecessor and the relative DoFs. For the recursive equation of the rotational Jacobian

$$ {}_i\boldsymbol{J}_{R,i} = \boldsymbol{A}_{ip}\, {}_p\boldsymbol{J}_{R,p} + {}_i\boldsymbol{J}_{R,\mathrm{rel},i} \tag{2.83} $$

this implies that the result ${}_i\boldsymbol{J}_{R,i}$ will only contain non-zero entries for columns corresponding to the DoFs of predecessors of the i-th body. Since the bodies are ordered such that the indices for the DoFs of successors are always larger than those of predecessors, the simplest optimization scheme is to evaluate the matrix-vector operations in (2.83) only for columns smaller than the index of the DoF occupied by the i-th body. For a serial mechanism with only 1-DoF joints, the evaluation of (2.83) then requires a total

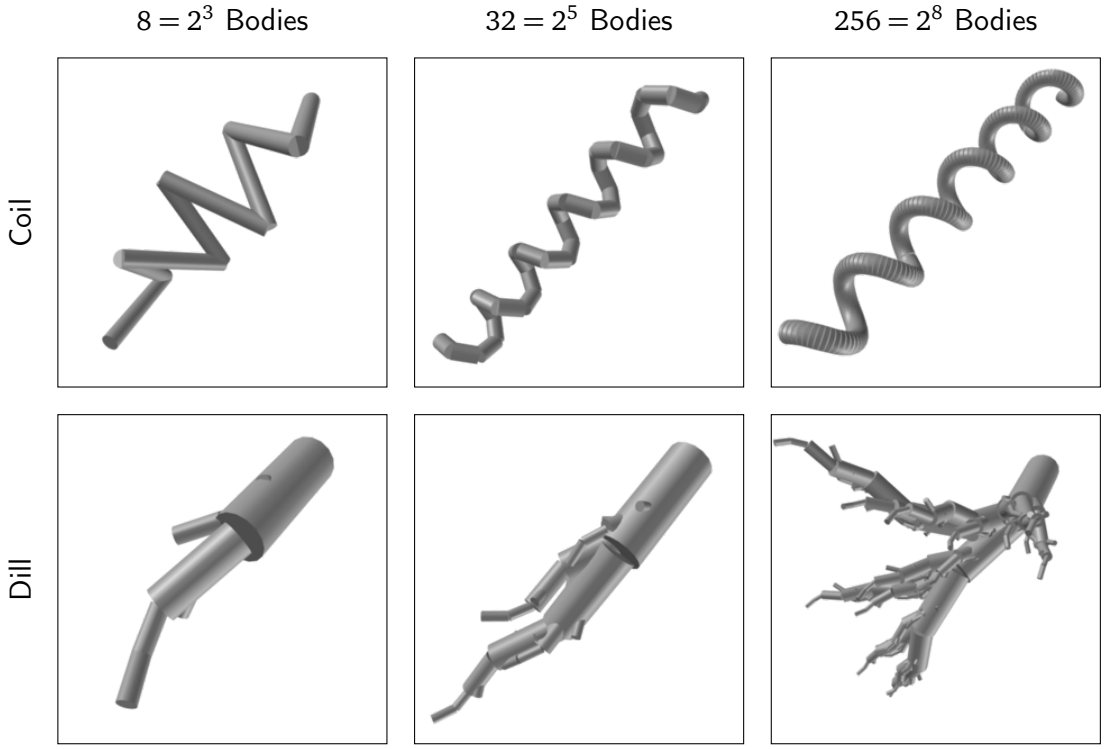


Figure 2.14: Examples of synthetic Coil (above) and Dill mechanisms (below).

of $9(i-1)$ multiplications and additions.[21] For all n_b bodies we then have

$$\sum_{i=1}^{n_b} 9(i-1) = \frac{9}{2} n_b (n_b - 1) \tag{2.84}$$

multiplications and additions. Without any optimizations, we have $9n_b$ additions and multiplications for every body,[22] leading to a total number of

$$\sum_{i=1}^{n_b} 9n_b = 9{n_b}^2 \tag{2.85}$$

additions and multiplications. For $n_b = 6$ the speedup is $\frac{12}{5}$ and decreases to 2 in the limit $n_b \to \infty$.

In the more general case of branched systems the optimization potential is even

21. From the matrix-matrix multiplication $\boldsymbol{A}_{ip}\,{}_p\boldsymbol{J}_{R,p}$. The addition of the relative Jacobian requires only a copy operation for simple joints, which is not accounted for here. In practical terms, however, memory access is very important, but difficult to include in a theoretical, operations-count oriented analysis (see chapter 3, figure 3.8).

22. Since we are evaluating $\boldsymbol{A}_{ip}\,{}_p\boldsymbol{J}_{R,p}$ for all n_b columns of ${}_p\boldsymbol{J}_{R,p}$

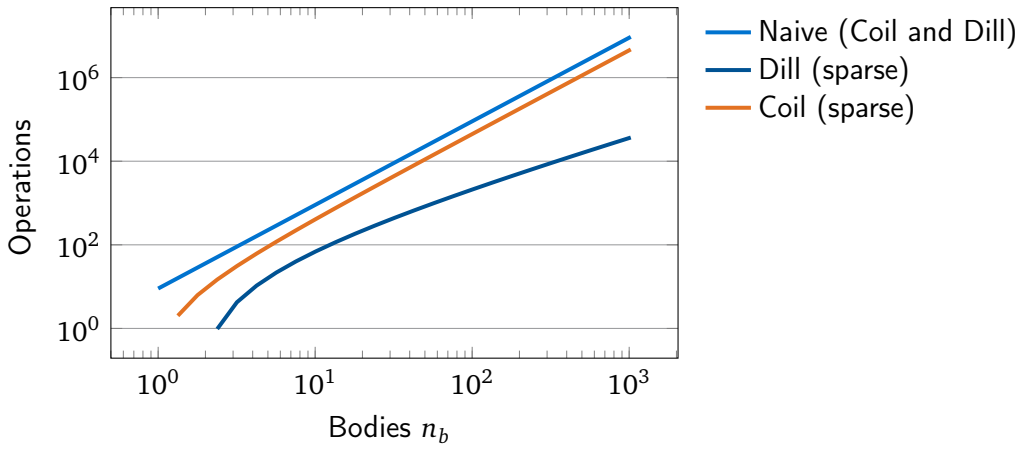


Figure 2.15: Theoretical operations for calculating the rotational Jacobians of Coil(n_b) and Dill(L) systems with and without exploiting the tree structure. The naive implementation gives the same operations count for Coil and Dill mechanisms, while using sparse linear algebra greatly reduces the necessary number of operations (cf. (2.84), (2.88)).

larger, since there are additional "holes" in the Jacobian because the motion of a body only depends on the DoFs corresponding to its predecessors. The rotational Jacobian of the third body of the mechanism in figure 2.6 may serve as an example. It has the following structure

$$ {}_3\boldsymbol{J}_{R,3} = \begin{pmatrix} \star & 0 & \star & 0 \\ \star & 0 & \star & 0 \\ \star & 0 & \star & 0 \end{pmatrix}, \tag{2.86} $$

which can be directly deduced from the predecessor list $P_3 = \{B_1\}$. The sparsity can be exploited by re-writing (2.83) in index notation and making use of the predecessor list of the i-the body P_i:

$$ {}_iJ_{R,i,[k,l]} = \sum_{m \in P_i} A_{ip,[k,m]}\, {}_pJ_{R,p,[m,l]} + {}_iJ_{R,\mathrm{rel},i,[k,l]}\,. \tag{2.87} $$

The efficiency improvements depend how sparse the Jacobian is, that is, how long the predecessor lists P_i are, which in turn depends on the topology of the mechanism. That is, the speedup is different for every system. The worst-case speedup is achieved for a serial mechanism, since this mechanism maximizes the length of P_i (speedup: $\frac{2n_b}{n_b-1}$, see above). The best-case speedup is obviously achieved for a mechanism with empty P_i, which corresponds to n_b bodies directly connected to the environment.

An analysis of the Dill(L) mechanism shows that its bodies have between 0 and L

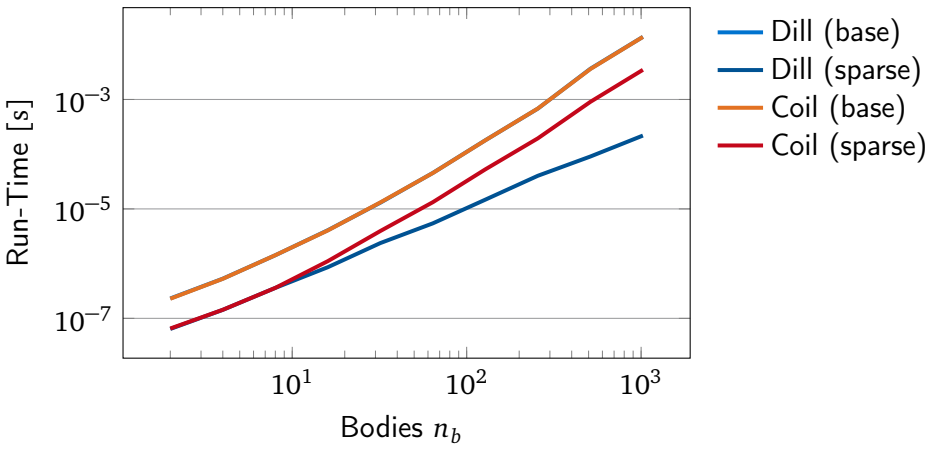


Figure 2.16: Measured run-times for calculating the rotational Jacobians of all bodies in the Coil(n_b) and Dill(L) systems with and without exploiting the tree structure. The basic implementation gives the same run-times for Coil and Dill mechanisms (see (2.85)), while using sparse linear algebra greatly reduces the computational cost, especially for the branched Dill mechanism, which translates into a reduced inclination of the corresponding graph (cf. (2.84), (2.88)).

predecessors. The number of bodies with i predecessors is $\binom{L}{i}$ because of the recursive definition of the mechanism. The number of multiplications and additions required for evaluating $\boldsymbol{A}_{ip}\,{}_p\boldsymbol{J}_{R,p}$ for a body with l predecessors is $9(l-1)$, since ${}_p\boldsymbol{J}_{R,p}$ has $l-1$ non-zero columns. The total number of operations for all Jacobians is then given by:

$$\begin{aligned} N_{\text{ops}} &= 9\sum_{i=1}^{L}\binom{L}{i}(i-1) \\ &= 9\left(1-2^L+2^{L-1}L\right) \\ &= 9\left(1-n_b+\frac{1}{2}n_b\log_2 n_b\right). \end{aligned} \tag{2.88}$$

The last equality uses the relationship between the number of bodies n_b and the recursion level of the mechanism L, $n_b = 2^L$. Apparently, the complexity is no longer $O({n_b}^2)$, but $O(n_b \log(n_b))$ for this branched mechanism. The theoretical operations count for optimized and naive implementations for Coil and Dill mechanisms is illustrated in figure 2.15.

The reduced computational cost is also evident in the computation times for the Dill and Coil mechanisms determined in numerical experiments (see figure 2.16). Note that the measured run-times do not exactly follow this simplified analysis, due to the overhead in sparse matrix multiplications and the complexities of modern CPUs and memory hierarchies (see also “Note on Run-Time Performance,” page 54). The trend

for large n_b is captured by the operations count, but there is a large difference for small n_b. Apparently, there are additional constant and linear terms not captured by the analysis (see page 54).

2.5 Chapter Summary

This chapter coveres the kinematics of rigid bodies and rigid multibody systems. Standard approaches to parametrize the configuration, as well as methods for computing positions, velocities, accelerations and Jacobian matrices are presented. Different kinematic topologies are discussed, as well as methods for exploiting the kinematic structure to reduce the cost of kinematics calculations. The equations and algorithms discussed in this chapter form the basis for dynamics and redundancy resolution methods presented in the following chapters.

3 Rigid Body Dynamics

This chapter covers methods from classical mechanics, which studies the motion of bodies caused by forces (forward dynamics) – and vice versa (inverse dynamics). The field is based on Newton's laws of motion for particles (Newton 1687; Eberhard and Schiehlen 2005) and Euler's laws of motion for the dynamics of rigid bodies (Euler 1776; Eberhard and Schiehlen 2005). Today Euler's laws of motion are known as the Newton-Euler equations, or, alternatively, the linear and angular momentum theorem. Fundamental contributions to the dynamics of constrained systems were presented D'Alembert (1743), who separated forces and motion into constrained and free components (impressed forces and constraint/lost forces). The mathematically consistent formulation in use today, based on constraint forces and the principle of virtual work was developed by Lagrange (1788). Alternative formulations for constrained systems were subsequently formulated on the velocity level by Jourdain (1909) and at the acceleration level by Gauss (1829; see below). Today, research is mainly targeted at developing and improving methods for efficiently formulating and solving the EoMs for systems of bodies. For a review of the history, current challenges, an overview of algorithms and a literature survey see (Schiehlen 1997; Eberhard and Schiehlen 2005; Bremer 1988, 1982). For methods for elastic multibody dynamics see (Ritz/modal approaches: Sorge (1993) and Bremer (2008); Finite Element Method (FEM)-based: Shabana (1990, 2005)).

Both simulation and control of robots require fast numerical codes for calculating forward and inverse dynamics of multibody systems. Together with the regular topology of most robots, this has led to the development of some of the most efficient dynamics algorithms. The 1980s saw a peak in this development, when the most well-known linear-complexity algorithms were developed (cf. section 3.3; Featherstone 1983; Brandl et al. 1986; Bae and Haug 1987a). The first algorithm following this approach, however, was published earlier by Vereshchagin (1974). Current research on efficient MBS algorithms is aimed at achieving sub-linear complexity by parallelizing the computation (cf. section 3.5; Featherstone 1999a; Yamane and Nakamura 2009). For a review of efficient algorithms for MBSs see (Featherstone and Orin 2000; Featherstone 2008).

The wide range of methods may be classified into purely numerical algorithms and methods using some symbolic processing prior to numerically solving the EoMs. The methods discussed in this chapter are examples of numerical algorithms (see sections 3.2, 3.3 and 3.4). Symbolic methods are used, for example, in the NEWEUL system,[1]

1. `http://www.itm.uni-stuttgart.de/research/neweul/neweulm2_en.php`, accessed 2013-11-12

MapleSim[2] and Modelica/Dymola.[3] A very large number of free and commercial software packages for multibody dynamics are available. Lists of programs are maintained, for example, by John McPhee[4] and Jeff Trinkle.[5] Both lists are extensive, but cannot include all relevant projects.

The remainder of this chapter is organized as follows: section 3.1 reviews basic principles of mechanics, section 3.2 develops methods for simulating multibody dynamics based on the EoM of the system and section 3.3 presents methods that scale linearly with the number of DoFs. The treatment of kinematic constraints is discussed in section 3.4, including a novel method with linear complexity that can handle kinematic loops in a minimal coordinate representation.

3.1 Principles of Mechanics

Classical mechanics can be built on several principles. If one of them is accepted as axiomatic, the remaining directly follow from it. Which one is chosen as a starting point is therefore a matter of taste and convenience. This section briefly reviews three principles from analytical mechanics that will be useful for the results in this chapter. In describing the different principles, we will study variations of the motion that are *compatible with the constraints*. The constraints are written as a set of equations,[6] which depend on the full set of system variables[7] $\boldsymbol{z}$ and time t

$$\boldsymbol{\Phi}(\boldsymbol{z}, t) = \mathbf{0} \tag{3.1}$$

and possibly also on their time derivatives $\dot{\boldsymbol{z}}$ (non-holonomic constraints)

$$\boldsymbol{\Phi}(\boldsymbol{z}, \dot{\boldsymbol{z}}, t) = \mathbf{0}. \tag{3.2}$$

We will make use of three types of variation of the motion of a mass particle dm, which must all be compatible with the constraints $\boldsymbol{\Phi} = \mathbf{0}$ (see Bremer 1993; Jourdain 1909):

- $\delta\boldsymbol{r}$: a variation of the position $\boldsymbol{r}$, for which *time is held constant* (Lagrange's variation).

2. `http://www.maplesoft.com/products/maplesim/`, accessed 2013-11-12
3. `http://www.modelica.org/` and `http://www.dymola.com`, accessed 2013-11-12
4. `http://real.uwaterloo.ca/~mbody/`, accessed 2013-11-12
5. `http://www.cs.rpi.edu/~trink/sim_packages.html`, accessed 2013-11-12, for systems with unilateral contacts
6. Inequality constraints are beyond the scope of this monograph, but the extension is straightforward, even if more sophisticated mathematical treatments are more recent (cf. footnote 13 on page 38).
7. That is, parameters that directly describe the configuration of every body without referring to the interconnections, i.e., six parameters for rigid bodies.

- $\delta\dot{\boldsymbol{r}}$: a variation of the velocity $\dot{\boldsymbol{r}}$, for which *time and* $\boldsymbol{r}$ *are held constant* (Jourdain's variation).
- $\delta\ddot{\boldsymbol{r}}$: a variation of the acceleration $\ddot{\boldsymbol{r}}$, for which *time,* $\boldsymbol{r}$ *and* $\dot{\boldsymbol{r}}$ *are held constant* (Gauss' variation).

Jourdain (1909) uses the symbols $\delta_1, \delta_2, \delta_3$ to distinguish variations at the position, velocity and acceleration level. To simplify notation, we use δ to denote all variations and assume that the variation applies only to accelerations for $\delta\ddot{\boldsymbol{r}}$, only velocities to $\delta\dot{\boldsymbol{r}}$ and only positions to $\delta\boldsymbol{r}$.

For the variation of the constraints we have:[8]

$$\delta\boldsymbol{\Phi} = \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{r}}\delta\boldsymbol{r} = \boldsymbol{0}. \tag{3.3}$$

Therefore, $\delta\boldsymbol{r}$ is orthogonal to the gradient of $\boldsymbol{\Phi}$ at $\boldsymbol{r}$, that is, tangential to the manifold $\boldsymbol{\Phi} = \boldsymbol{0}$. For Jourdain's and Gauss' variation we get equivalent results by studying variations of the time derivatives of $\boldsymbol{\Phi}$ and "canceling the dots:"[9]

$$\delta\dot{\boldsymbol{\Phi}} = \frac{\partial\dot{\boldsymbol{\Phi}}}{\partial\dot{\boldsymbol{r}}}\delta\dot{\boldsymbol{r}} = \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{r}}\delta\dot{\boldsymbol{r}} = \boldsymbol{0}, \tag{3.4}$$

$$\delta\ddot{\boldsymbol{\Phi}} = \frac{\partial\ddot{\boldsymbol{\Phi}}}{\partial\ddot{\boldsymbol{r}}}\delta\ddot{\boldsymbol{r}} = \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{r}}\delta\ddot{\boldsymbol{r}} = \boldsymbol{0}. \tag{3.5}$$

That is, all three variations can be identified as elements of the tangent plane of $\boldsymbol{\Phi}$ at $\boldsymbol{r}$.

Principles of d'Alembert and Jourdain

D'Alembert's principle in the form given by Lagrange[10] is given by:

$$\int_S \left(\ddot{\boldsymbol{r}}_m \,\mathrm{d}m - \mathrm{d}\boldsymbol{F}^i\right)^T \delta\boldsymbol{r}_m = \int_S \left(\mathrm{d}\boldsymbol{F}^c\right)^T \delta\boldsymbol{r}_m = \boldsymbol{0}. \tag{3.6}$$

Here, $\boldsymbol{r}_m$ is the vector to the mass element $\mathrm{d}m$, $\boldsymbol{F}^c$ denotes the *constraint forces* due to $\boldsymbol{\Phi}$ and $\boldsymbol{F}^i$ the impressed forces, that is, forces doing work. The integration is performed over the whole system S.

Jourdain's principle is similar, but formulated with variations at the velocity level

8. This may be obtained from the total differential $\mathrm{d}\boldsymbol{\Phi} = \frac{\partial\boldsymbol{\Phi}}{\partial r}\,\mathrm{d}\boldsymbol{r} + \frac{\partial\boldsymbol{\Phi}}{\partial t}\,\mathrm{d}t$ by setting $\mathrm{d}t = 0$ (time is held constant).

9. That is, using $\frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{r}} = \frac{\partial\dot{\boldsymbol{\Phi}}}{\partial\dot{\boldsymbol{r}}} = \frac{\partial\ddot{\boldsymbol{\Phi}}}{\partial\ddot{\boldsymbol{r}}} = \ldots$. Proof: $\dot{\boldsymbol{\Phi}} = \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{r}}\dot{\boldsymbol{r}} + \frac{\partial\boldsymbol{\Phi}}{\partial\dot{\boldsymbol{r}}}\ddot{\boldsymbol{r}} + \frac{\partial\boldsymbol{\Phi}}{\partial t} \Rightarrow \frac{\partial\dot{\boldsymbol{\Phi}}}{\partial\dot{\boldsymbol{r}}} = \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{r}}$, etc. (e.g., Hamel 1949; Bremer and Glocker 2000).

10. This principle is also called Lagrange's principle or the Lagrange-d'Alembert principle, since the form used today involving the principle of virtual work was developed by Lagrange (e.g., Hamel 1949, chapter 2).

instead of the position level (Jourdain 1909; Bremer 1993):

$$\int_S \left(\ddot{\boldsymbol{r}}_m \, \mathrm{d}m - \mathrm{d}\boldsymbol{F}^i\right)^T \delta \dot{\boldsymbol{r}}_m = \int_S (\mathrm{d}\boldsymbol{F}^c)^T \, \delta \dot{\boldsymbol{r}}_m = \boldsymbol{0}. \tag{3.7}$$

Since $\delta \boldsymbol{r}_m$ and $\delta \dot{\boldsymbol{r}}_m$ are simply elements of the tangent space to $\boldsymbol{\Phi}$ (see above and Bremer 1993), both Jourdain's and d'Alembert's principle have the same geometrical interpretation: constraint forces are orthogonal to the constraint manifold and therefore their virtual work/power vanishes.[11]

Gauss' Principle

Gauss (1829) introduced a further fundamental principle of mechanics. Gauss' Principle states that the accelerations of the mass particles in a system will minimize the distance to the acceleration that would occur in the absence of any constraints.[12] Since the effect of the constraints is minimized, the principle is also known as the *Principle of Least Constraint*. As noted in the original article, this principle is equivalent to the d'Alembert/Lagrange principle and does not lead to any new results. Formulating dynamics in terms of a constrained minimization problem can, however, be beneficial from a computational point of view and can lead to new (numerical) dynamics algorithms.[13] In modern mathematical notation Gauss' principle reads:

$$G = \frac{1}{2} \int_S (\ddot{\boldsymbol{r}}_m - \tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m})^2 \, \mathrm{d}m \to \min! \quad \text{with:} \quad \boldsymbol{\Phi}(\boldsymbol{z}, \dot{\boldsymbol{z}}, t) = \boldsymbol{0}. \tag{3.8}$$

Note that only the impressed forces $\boldsymbol{F}^i$ enter the function G, since $\frac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m}$ is the acceleration that dm would undergo *in the absence of any constraints*. The first order optimality

11. Jourdain himself provided a proof that the variations (and therefore the principles) are equivalent (see Jourdain 1909, eq. (4), (5) and (6)).

12. The exact wording is: "Die Bewegung eines Systems materieller, auf was immer für eine Art unter sich verknüpfter Punkte, deren Bewegungen zugleich an was immer für äußere Beschränkungen gebunden sind, geschieht in jedem Augenblick in möglichst größter Übereinstimmung mit der freien Bewegung, oder unter möglich kleinstem Zwange, in dem man als Maaß des Zwanges, den das ganze System in jedem Zeitteilchen erleidet, die Summe der Produkte aus dem Quadrate der Ablenkung jedes Punkts von seiner freien Bewegung in seine Maße betrachtet" (Gauss 1829; spelling changed in some points).

13. The footnote in (Gauss 1829, p. 234) is especially interesting in this respect. Gauss remarks that inequality constraints can be taken into account. That is, the original paper is directly connected to modern formulations for systems with unilateral constraints via optimization methods for convex problems (cf. e.g., Glocker 1998; Förg et al. 2006; Rockafellar 1976; Anitescu 2006). Gauss, however, does not treat impacts and obviously does not make use of modern mathematical constructs such as measure differential inclusions to treat them. The connection, however, is very strong. Anitescu (2006), for example, derives an optimization-based time stepping scheme at the velocity level. The same result, however, directly follows from Gauss' principle when applying a Euler discretization of the states together with the additional assumptions in (Anitescu 2006, figure 5.1).

condition for (3.8) is that the variation δG vanishes for any admissible variation of the accelerations, that is,

$$\delta G = \int_S (\ddot{\boldsymbol{r}}_m \,\mathrm{d}m - \boldsymbol{F}^i)^T \delta \ddot{\boldsymbol{r}}_m = 0, \tag{3.9}$$

which demonstrates the equivalence to Jourdain's and d'Alembert's principles (see above). Nevertheless, it can be beneficial to evaluate the integral for G before deriving optimality conditions, opening the path to new simulation algorithms. For a rigid body B this is done by substituting (2.28) for the acceleration of the mass element $\mathrm{d}m$:

$$G = \frac{1}{2}\int_B (\ddot{\boldsymbol{r}}_m - \tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m})^2 \,\mathrm{d}m \tag{3.10}$$

$$= \frac{1}{2}\int_B (\ddot{\boldsymbol{r}}_c + (\dot{\widetilde{\boldsymbol{\omega}}} + \widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}})\boldsymbol{r}_{cm} - \tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m})^2 \,\mathrm{d}m \tag{3.11}$$

$$= \frac{1}{2}m\ddot{\boldsymbol{r}}_c^2 + \frac{1}{2}\dot{\boldsymbol{\omega}}^T\boldsymbol{\Theta}_c\dot{\boldsymbol{\omega}} + \dot{\boldsymbol{\omega}}^T\left(\widetilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_c\boldsymbol{\omega}\right) - \ddot{\boldsymbol{r}}_c^T\boldsymbol{F}^i - \dot{\boldsymbol{\omega}}^T\boldsymbol{T}_c^i + \text{irrel. terms.} \tag{3.12}$$

$\boldsymbol{T}_c^i$	impressed moment about Center of Mass (CoM)
$\boldsymbol{\Theta}_c$	mass moment of inertia about CoM
$\boldsymbol{r}_m$	position of mass element $\mathrm{d}m$
$\boldsymbol{r}_c$	position of CoM
$\boldsymbol{r}_{cm}$	position of mass element relative to CoM

Here, "irrelevant terms" are all those that do not change the solution, that is, are independent of the linear accelerations of the CoM $\ddot{\boldsymbol{r}}_c$ and the angular acceleration $\dot{\boldsymbol{\omega}}$. A detailed derivation is given in appendix B. A more compact notation for the results is

$$G = \frac{1}{2}\left(\boldsymbol{a} + \boldsymbol{M}^{-1}\boldsymbol{h}^\star\right)^T \boldsymbol{M}\left(\boldsymbol{a} + \boldsymbol{M}^{-1}\boldsymbol{h}^\star\right), \tag{3.13}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{h}^\star$ is a force vector and $\boldsymbol{a}$ the vector of accelerations. When choosing the CoM as reference point for the body, these quantities are

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{\Theta}_c & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{pmatrix}, \tag{3.14}$$

$$\boldsymbol{h}^\star = \begin{pmatrix} \widetilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_c\boldsymbol{\omega} - \boldsymbol{T}_c^i \\ -\boldsymbol{F}^i \end{pmatrix}, \tag{3.15}$$

$$\boldsymbol{a} = \begin{pmatrix} \dot{\boldsymbol{\omega}} \\ \ddot{\boldsymbol{r}}_c \end{pmatrix}. \tag{3.16}$$

For an arbitrary reference point O they become (cf. appendix B)

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{\Theta}_o & m\widetilde{\boldsymbol{r}}_{oc} \\ m\widetilde{\boldsymbol{r}}_{oc}^T & m\boldsymbol{I}_3 \end{pmatrix}, \tag{3.17}$$

$$\boldsymbol{h}^\star = \begin{pmatrix} \widetilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_o\boldsymbol{\omega} - \boldsymbol{T}_o^i \\ m\widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}}\boldsymbol{r}_{oc} - \boldsymbol{F}^i \end{pmatrix}, \tag{3.18}$$

$$\boldsymbol{a} = \begin{pmatrix} \dot{\boldsymbol{\omega}} \\ \ddot{\boldsymbol{r}}_o \end{pmatrix}. \tag{3.19}$$

Note that the alternative formulation $G = \frac{1}{2}\left(\boldsymbol{M}\boldsymbol{a} + \boldsymbol{h}^\star\right)^T \boldsymbol{M}^{-1}\left(\boldsymbol{M}\boldsymbol{a} + \boldsymbol{h}^\star\right)$, which is often found in the literature, is equivalent. We prefer the definition used above, since it is closer the original proposal by Gauss (1829). Both formulations directly lead to the Newton-Euler equations of a rigid body via the necessary optimality conditions:

$$\boldsymbol{M}\boldsymbol{a} + \boldsymbol{h}^\star = \boldsymbol{0}. \tag{3.20}$$

3.2 Global EoM-based Dynamics

The straightforward approach towards dynamics simulation for MBSs is to calculate the EoM explicitly and to then solve it for either accelerations (forward dynamics) or forces (inverse dynamics). This section presents approaches for generating and solving the EoMs, as well as possibilities for reducing the run-time of the resulting computer programs.

3.2.1 Equations of Motion

Minimal Coordinate Formulation

The equations of motion for a MBS can be derived from any of the principles presented above. In the following, we use Gauss' Principle to derive the EoM in a minimal coordinate formulation, that is, in an Ordinary Differential Equation (ODE)-formulation without explicit algebraic (kinematic) constraints.

For a system of rigid bodies, the function G to be minimized is the sum of the contributions of the n_b bodies in the system, cf. (3.13):

$$G = \frac{1}{2}\sum_{i=1}^{n_b}\left(\boldsymbol{a}_i + \boldsymbol{M}_i^{-1}\boldsymbol{h}_i^\star\right)^T \boldsymbol{M}_i\left(\boldsymbol{a}_i + \boldsymbol{M}_i^{-1}\boldsymbol{h}_i^\star\right) \quad \rightarrow \quad \text{min!} \tag{3.21}$$

Assuming we have a set of minimal coordinates $\boldsymbol{q}$, the kinematic constraints are implicitly given by the forward kinematics for $\boldsymbol{a}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, cf. (2.75). Gauss' Principle can then be written as an unconstrained minimization problem in $\ddot{\boldsymbol{q}}$ by substituting the

constraints $\boldsymbol{a}_i = \boldsymbol{a}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ into G. The first order optimality conditions are obtained by differentiating with respect to $\ddot{\boldsymbol{q}}$:[14]

$$\left(\frac{\mathrm{d}G}{\mathrm{d}\ddot{\boldsymbol{q}}}\right)^T = \sum_{i=1}^{n_b} \left(\frac{\partial \boldsymbol{a}_i}{\partial \ddot{\boldsymbol{q}}}\right)^T \left(\boldsymbol{M}_i \boldsymbol{a}_i + \boldsymbol{h}_i - \boldsymbol{u}_i\right) \tag{3.22}$$

$$= \sum_{i=1}^{n_b} \boldsymbol{J}_i^T \left(\boldsymbol{M}_i \boldsymbol{a}_i + \boldsymbol{h}_i - \boldsymbol{u}_i\right) = \boldsymbol{0}. \tag{3.23}$$

Here, the vector $\boldsymbol{h}^\star = \boldsymbol{h} - \boldsymbol{u}$ was separated into forces $\boldsymbol{h}$ attributed to the physical MBS itself and applied actuator forces $\boldsymbol{u}$ influenced by a controller. This (or an equivalent) form of the EoM can be derived in a number of ways. Accordingly, different authors have given different derivations and names. Essentially, these variants are equivalent, but there are some differences in the formalisms related to them (e.g., Bremer 2008; Shabana 2005, p.117; Schwertassek and Roberson 1984; Kane and Levinson 1983; Kane 1961; for discussion see Bremer 1987; Piedboeuf 1993; Bremer and Pfeiffer 1988, p.413). In the following, the algorithms based on (3.23) will be called the Newton-Euler Method (NE).

Standard Form of the EoM

The most common approach to forward dynamics based on the global EoM is to solve for $\ddot{\boldsymbol{q}}$ and use a suitable time integration scheme to obtain the system trajectory $\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)$. A form of the EoM suitable for this task is obtained by substituting the accelerations $\boldsymbol{a}_i$ as a function of $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ into the optimality conditions given above. A non-recursive equation for the accelerations is obtained by calculating the absolute accelerations from (2.66) and (2.65), yielding:[15]

$$\begin{pmatrix} \dot{\boldsymbol{\omega}}_i \\ \ddot{\boldsymbol{r}}_i \end{pmatrix} = \begin{pmatrix} \left(\frac{\partial \boldsymbol{\omega}_i}{\partial \dot{\boldsymbol{q}}}\right)\ddot{\boldsymbol{q}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{\omega}_i}{\partial \dot{\boldsymbol{q}}}\right)\dot{\boldsymbol{q}} \\ \left(\frac{\partial \dot{\boldsymbol{r}}_i}{\partial \dot{\boldsymbol{q}}}\right)\ddot{\boldsymbol{q}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \dot{\boldsymbol{r}}_i}{\partial \dot{\boldsymbol{q}}}\right)\dot{\boldsymbol{q}} + \widetilde{\boldsymbol{\omega}}_i \dot{\boldsymbol{r}}_i \end{pmatrix}, \tag{3.24}$$

$$\Rightarrow \boldsymbol{a}_i = \boldsymbol{J}_i \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}} + \boldsymbol{c}_i, \qquad \boldsymbol{c}_i^T = \begin{pmatrix} \boldsymbol{0} & \left(\widetilde{\boldsymbol{\omega}}_i \dot{\boldsymbol{r}}_i\right)^T \end{pmatrix}. \tag{3.25}$$

Substituting this into (3.23) gives the standard form of the EoM (3.28):

$$\boldsymbol{0} = \sum_{i=1}^{n_b} \boldsymbol{J}_i^T \left(\boldsymbol{M}_i(\boldsymbol{J}_i \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}} + \boldsymbol{c}_i) + \boldsymbol{h}_i - \boldsymbol{u}_i\right) \tag{3.26}$$

14. Note that G is strictly convex, since the mass matrices $\boldsymbol{M}_i$ are positive definite. Solving the first order optimality conditions therefore always yields the uniquely defined global minimum.

15. In a computer implementation, it might be better not to use the time derivatives of the Jacobians, but to use $\frac{\partial \dot{\boldsymbol{r}}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \dot{\boldsymbol{r}}}{\partial \dot{\boldsymbol{q}}}\right)\dot{\boldsymbol{q}}$ instead. Note that while the previous equality holds, $\left(\frac{\partial \dot{\boldsymbol{r}}}{\partial \boldsymbol{q}}\right) \neq \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \dot{\boldsymbol{r}}}{\partial \dot{\boldsymbol{q}}}\right)$!

$$= \sum_{i=1}^{n_b} \boldsymbol{J}_i^T \boldsymbol{M}_i \boldsymbol{J}_i \ddot{\boldsymbol{q}} + \sum_{i=1}^{n_b} \boldsymbol{J}_i^T \left(\boldsymbol{M}_i \left(\dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}} + \boldsymbol{c}_i \right) + \boldsymbol{h}_i - \boldsymbol{u}_i \right), \tag{3.27}$$

$$\Rightarrow \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{u}. \tag{3.28}$$

3.2.2 Inverse Dynamics Simulation

Inverse dynamics, that is, calculating the generalized forces $\boldsymbol{u}$ for a given state of motion $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ is a simple algebraic problem.[16] In many cases, the simplest approach is to use the explicit form of the EoM (3.25), if it is available, and to calculate actuator forces by evaluating $\boldsymbol{u} = \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h}$. If efficiency is important, however, this is not the best choice, since calculating $\boldsymbol{M}$ is generally an $O(n_b^2)$ operation (but see section 3.2.3, "Efficiently Calculating $\boldsymbol{M}$ and $\boldsymbol{h}$"). A more efficient approach is to use (3.23) directly: after calculating $\boldsymbol{a}_i$ recursively for given $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$, the corresponding generalized actuator forces $\boldsymbol{u} = \sum_{i=1}^{n_b} \boldsymbol{J}_i^T \boldsymbol{u}_i$ are given by (3.23). The most efficient known inverse dynamics algorithms avoid calculating the Jacobians, since calculating all Jacobians is generally an $O(n_b^2)$ operation (cf. section 2.4). This can be avoided by using a recursive approach which has $O(n_b)$ complexity, as described in section 3.3.1.

3.2.3 Forward Dynamics Simulation

All terms for $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are easily calculated by forward recursion (cf. section 2.3). The simplest method to simulate the forward dynamics is: (1) calculate $\boldsymbol{M}$ and $\boldsymbol{h}$ using (3.28), (2) solve for $\ddot{\boldsymbol{q}}$ using, for example, a Cholesky Factorization (e.g., Quarteroni et al. 2007) and (3) integrate $\ddot{\boldsymbol{q}}$ using a suitable ODE integration scheme. There are a number of possible optimizations to this basic forward scheme. They may be classified into (1) speedups in calculating $\boldsymbol{M}$ and $\boldsymbol{h}$ and (2) efficiency improvements in solving for $\ddot{\boldsymbol{q}}$. Additionally, (3) efficient programming is essential for speeding up run-times.

Efficiently Calculating M and h

An analysis of the EoMs (3.28) and (3.23) shows, that the most costly operation is calculating the global Jacobians and using them to map forces to generalized coordinates and calculate the mass matrix. Calculating the Jacobians can be made much more efficient by exploiting the sparsity introduced by branches in the kinematic tree (cf. section 2.4). Using sparse linear algebra for evaluating the products $\boldsymbol{J}_i^T \boldsymbol{h}_i$ and $\boldsymbol{J}_i^T \boldsymbol{M}_i \boldsymbol{J}_i$ can greatly reduce the computational cost. Similar to the Jacobians (cf. (2.87)), $\boldsymbol{M}$ can be determined more efficiently by only calculating non-zero elements. The matrix $\boldsymbol{M}$ is

16. This is true for most robotic systems. More complex cases, for example, overactuated systems with no unique solution for the inverse dynamics problem, are not considered here.

the sum of the contributions by all bodies:

$$M_{ij} = \sum_{n=1}^{n_b} \Delta M_{n,ij}, \qquad \Delta M_{n,ij} = \sum_{k=1}^{6} \sum_{l=1}^{6} J_{n,k,i} M_{n,k,l} J_{n,l,j}. \tag{3.29}$$

Programmatically, the sum is calculated in a triple loop over all bodies, all rows and columns of the matrix. Here, we can exploit the fact that $\Delta M_{n,ij}$ is non-zero only if i and j are indices of a DoF of one of the parents of n (i.e., $i \in P_n \wedge j \in P_n$). Further optimizations are possible in the calculation of $\Delta M_{n,ij}$ itself by decomposing the matrices $\boldsymbol{M}_i$ and $\boldsymbol{J}_i$ into rotational and translational components, according to their definitions (3.17) and (2.74). Then, the cross product can be used instead of the tilde operator for terms involving $\widetilde{\boldsymbol{r}}_{ic}$ and multiplications with the 3×3 matrix $m\boldsymbol{I}_3$ are replaced with scalar multiplications.[17]

Also, $\boldsymbol{h}$ may be calculated recursively based on an inverse dynamics routine without calculating the Jacobians (see section 3.3.1; Luh et al. 1980a). The basic procedure, described by Walker and Orin (1982), is as follows: (1) Calculate the inverse dynamics for the mechanism, setting $\ddot{\boldsymbol{q}} = \boldsymbol{0}$. Since the impressed forces $\boldsymbol{u}$ are equal to the *bias force* $\boldsymbol{h}$ according to (3.28), this procedure is $O(n_b)$, if an $O(n_b)$ inverse dynamics algorithm is used (cf. section 3.3.1). A slightly less efficient, but simpler variant is to use the inverse dynamics algorithm based on (3.23) (cf. section 3.2.2). (2) Calculate the mass matrix recursively without using the Jacobians. This may be achieved either by calculating the inverse dynamics for $\ddot{\boldsymbol{q}} = \boldsymbol{e}_i$, with $\dot{\boldsymbol{q}} = \boldsymbol{0}$ and without gravity. According to (3.28) the resulting $\boldsymbol{u}$ is equal to the i-th column of $\boldsymbol{M}$. An even more efficient method is based on the comparison of (3.28) for unit accelerations $\ddot{\boldsymbol{q}} = \boldsymbol{e}_i$ with the NE equations for the subtree of the MBS rooted at the joint corresponding to $\boldsymbol{q}_i$.[18] Obviously, the corresponding mass matrix entries are related to the effective inertias of the subtree with respect to the joint currently studied via the joint Jacobians. The effective inertias for the subtree may be calculated recursively via the Huygens-Steiner (or parallel axis) theorem (e.g., Ginsberg 1998, p.176). This algorithm was proposed by Walker and Orin (1982) as *method 3* and has subsequently been called the Composite Rigid Body Algorithm (CRBA) (term introduced in Featherstone 1983; see original paper Walker and Orin 1982 or Siciliano and Khatib 2008, section 2.5.3 for a detailed description of the algorithm).

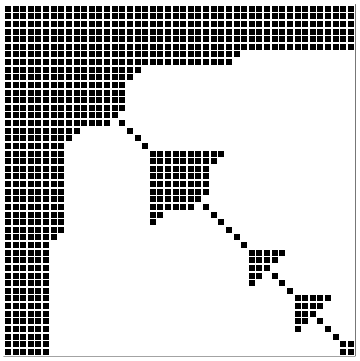
Exploiting Sparsity in Solving for $\ddot{\boldsymbol{q}}$

The computational cost of forward dynamics methods based on the explicit equation of motion[19] (3.28) have a *worst-case* computational complexity of $O(n_q^3)$, since this is the complexity of solving a linear equation using a direct method such as the

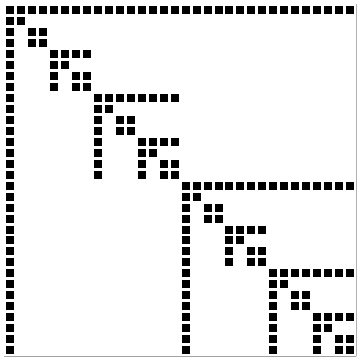
17. The implementation of many of these optimization for the results described here was done by Markus Schwienbacher (see also Schwienbacher 2013).

18. That is, the tree is cut at the joint for $\boldsymbol{q}_i$ and only this body and its successors are considered.

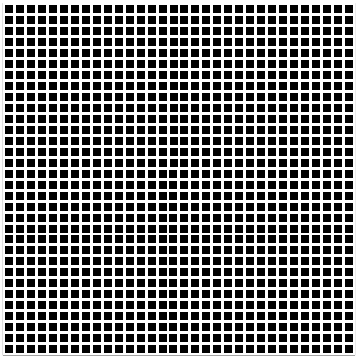
19. Also called *mass matrix methods* (Featherstone 2008).



Humanoid Lola with elastic gears (46 DoFs).



Dill(5) mechanism with $2^5 = 32$ DoFs.



Coil(32) mechanism with $2^5 = 32$ DoFs.

Figure 3.1: Examples of the sparsity patterns in mass matrices, non-zero elements marked by ■. The only tree-structured system with no systematic sparsity is a kinematic chain, for example, the Coil(n_b) shown on the right (Coil and Dill mechanisms are shown in figure 2.14).

Cholesky Factorization. If the sparsity introduced by branches is exploited, however, solving the linear equation has a lower complexity than $O(n_q^3)$. How the sparsity arises mathematically is evident from the equation for the mass matrix (3.29) and the sparsity of $\boldsymbol{J}_i$ resulting from the connectivity of the kinematic tree (see section 2.4; for an example see (2.86) and figure 2.6). Figure 3.1 shows the structure of the mass matrices of a biped robot model (Lola, see figure 2.13), a Dill mechanism and a Coil mechanism (see section 2.4 for a description of Dill and Coil).

The number of non-zero mass matrix elements can be determined from the equation for the mass matrix (3.29). The j-th column in the Jacobian J_i for the i-th body is non-zero, if the body for the j-th DoF is a predecessor of the i-the body (see section 2.4, p. 32). Therefore, the number of non-zero elements in the j-th column above the diagonal is equal to the number of predecessors of the corresponding body. For a matrix factorization procedure, only the number of non-zero elements N_{nz} in the upper triangular matrix are relevant, which depends on the number of predecessors $N_{p(i)}$ of the i-the body:

$$N_{nz} = n_b + \sum_{i=1}^{n_b} N_{p(i)}. \tag{3.30}$$

For the Dill mechanism this sum may be re-written using binomial coefficients and the Dill level (cf. section 2.4, p. 32):

$$N_{nz} = n_b + \sum_{i=k}^{L} \binom{L}{k} k \tag{3.31}$$

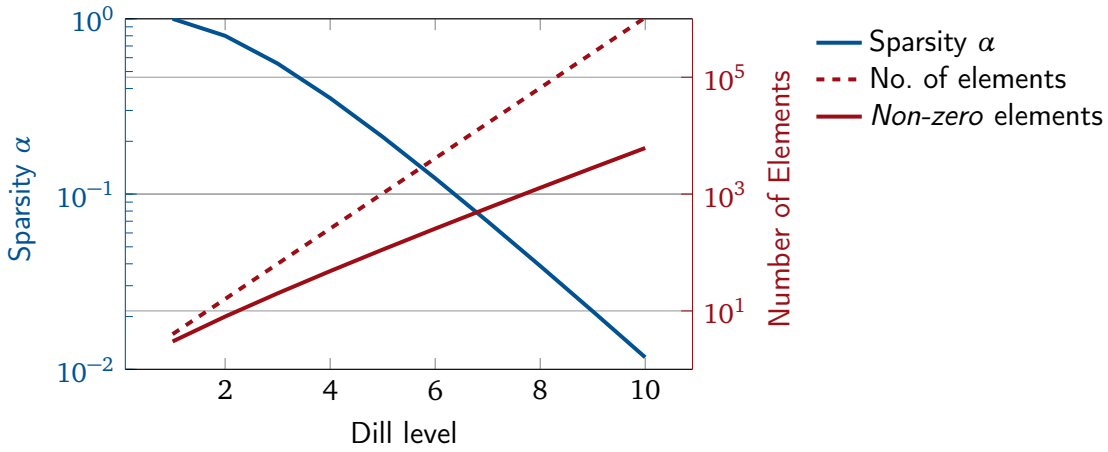


Figure 3.2: Total number of elements, non-zero elements and sparsity of the mass matrix of Dill mechanisms (i.e., the sparsity value, cf. (3.33); for examples of the mechanism, see figure 2.14). The log scale shows the exponential increase in sparsity for large systems.

$$= n_b + \frac{L n_b}{2}. \tag{3.32}$$

The resulting sparsity factor α, that is, the ratio of non-zero elements to all elements of the upper triangular matrix, decreases exponentially with increasing L:

$$\alpha = \frac{N_{nz}}{n_b(n_b+1)/2} = \frac{2+L}{1+2^L}. \tag{3.33}$$

Figure 3.2 illustrates the resulting sparsity pattern for the Dill mechanism.

The standard Cholesky Factorization of a sparse matrix does *not* preserve the sparsity pattern, therefore it cannot take advantage of the sparsity and accordingly also has the complexity $O(n_q^3)$. The sparsity may be exploited, however, by a Reversed Sparse Cholesky (RSC) decomposition. The algorithm is known from standard sparse matrix algebra, but was described and analyzed in detail for branched MBS by Featherstone (2005).[20] The resulting complexity depends on the topology of the tree, not only the overall sparsity α. The *worst case* complexity is $O(n_b d^2)$, where d is the depth[21] of the tree (Featherstone 2008, p. 116).

Figure 3.3 illustrates the run-time improvements achievable for Coil-chains and figure 3.4 those for Dill-trees. For chains, the CRBA[22] algorithm is significantly faster

20. The implementation used for the work presented here is described by Schwienbacher (2013, previously unpublished), who independently discovered the applicability of RSC to branched systems after Featherstone (2005).

21. The depth is the largest number of predecessors for any body, e.g., the level L for a Dill(L) system.

22. Maximilian Grill implemented the CRBA algorithm as part of a semester project (Grill 2013).

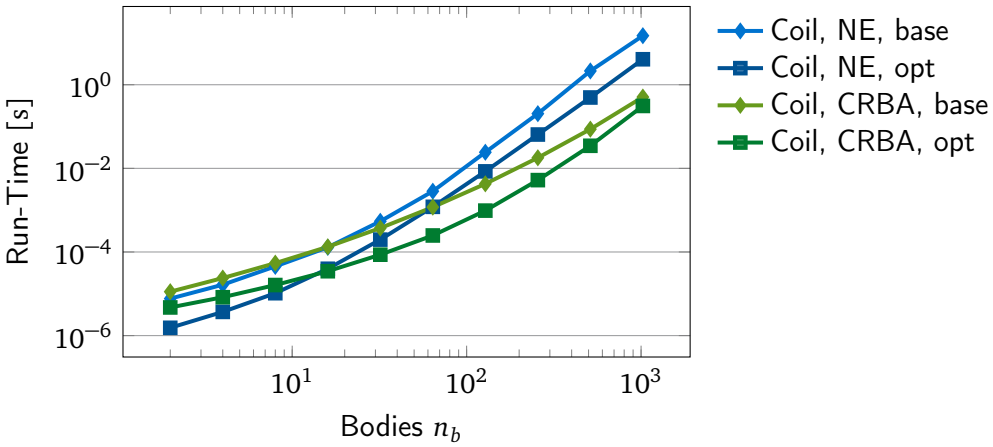


Figure 3.3: Run-times for calculating $\ddot{q}$ for Coil mechanisms using the NE and CRBA methods. Trends are similar for basic (base) and optimized (opt) code, but optimization reduces absolute run-time significantly.

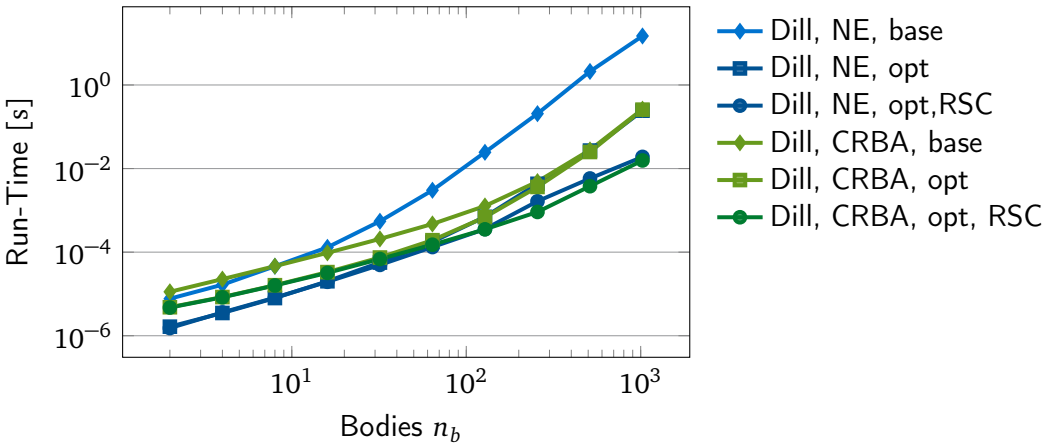


Figure 3.4: Run-times for calculating $\ddot{q}$ for Dill mechanisms using the NE and CRBA methods. Both methods shown in basic (base), optimized (opt) and optimized versions with RSC decomposition.

than NE, especially for very large systems. Note that pure chains with more than ten bodies are very uncommon. For a small number of DoFs, however, the quality of the implementation (optimizations, used programming language, etc.) will be more important than the choice of the algorithm (cf. figure 3.3). The performance of the naive implementations of CRBA and NE without RSC are identical for Coil and Dill, since sparsity is not exploited (i.e., CRBA is superior for large systems). For optimized[23] versions of NE and CRBA-based algorithms, however, the performance for the Dill mechanism is comparable. For very small systems NE is faster, for fairly large systems CRBA is faster.[24] It is important to note that the actual run-time varies depending on the computer that is used (CPU, compiler, etc.; cf. run-time comparisons in section 3.3.2).

3.3 Recursive Linear-Complexity Dynamics for Trees

As detailed above, most methods based on the global EoM (3.28) have a computational complexity of $O(n_q^3)$, which can be reduced to linear complexity in the number of DoFs ($O(n_q d^2)$, see above) if sparsity of the mass matrix is exploited. Over the past decades, a number of methods have been developed with a linear complexity in the number of bodies ($O(n_b)$). Many derivations and variants of these algorithms have been given. All require a tree structure to function and exploit two basic recursive properties introduced by it: (1) The kinematic state of any body is related to that of its predecessor via the relative DoF. Since the state of the first body is fixed or directly related to $\boldsymbol{q}$, this leads to a *forward recursion* for kinematic quantities; (2) Constraint forces acting on a body depend on the impressed forces (e.g., actuator torques) and the constraint forces acting on its neighboring bodies. Since leaf bodies[25] only have

23. If not explicitly stated otherwise, all calculations were performed on a computer with Intel® Core™ i5 CPU 660 (4 cores at 3.33 GHz), 7.7 GB Random Access Memory (RAM) in 64 bit mode, running Linux 3.12.6, programs compiled with GCC 4.8.2 (the GNU Compiler Collection, `http://gcc.gnu.org/`).

Most optimizations were implemented by Markus Schwienbacher (partially published in Schwienbacher 2013). The optimizations include exploiting the sparsity patterns, as described above, manual loop-unrolling and reordering of operations, as well as the usage of SSE-intrinsics (Intel Corporation 2011). Using SSE provides average run-time improvements of approximately 20-50%. The "base" or "naive" implementation is a straightforward application of the equations of the methods described in this chapter without further optimization. All programs were written in the C++ programming language using an in-house linear algebra library. The library uses operator overloading for basic linear algebra, some simple optimizations for the most frequent operations encountered in kinematics calculations and a pool-allocator for efficient memory management (allocating and freeing memory accounts for a significant portion of the run-time for small linear algebra operations encountered in rigid body kinematics and dynamics).

24. Note, however, that for very large systems details of the CPU cache hierarchy and aspects of the memory management and memory usage of the programs become more and more important relative to the number of floating point operations, see "Note of Run-Time Performance," page 54.

25. That is, bodies with no children.

constraint forces from the connection to their parent, this leads to a *backward recursion* for forces. The linear complexity methods avoid calculating the explicit form of the EoM (3.28) and instead directly operate on a form of dynamics equation prior to projection into the constraint free directions.

The first known presentation of an O(n_b) method is given by Vereshchagin (1974). The method is derived based on Gauss' principle and dynamic programming (Bellman 1954). The derivation is limited to serial chains, makes some simplifying assumptions regarding inertia matrices and the choice of coordinate frames is not optimal from a computational point of view. Nevertheless, this concisely written paper preceded the publication of the essentially equivalent Articulated Body Algorithm (ABA) by nine years and the paper by Walker and Orin (1982), previously widely believed to be the most efficient method, by eight years. A further variation of the O(n_b) algorithm was presented by Bae and Haug (1987a). The most efficient general formulation currently known was developed by Brandl et al. (1986). Basically, the O(n_b) algorithms constitute a symbolic Gaussian elimination process for the EoM (cf. Saha 1997; Mohan and Saha 2007). A different approach towards a dynamics algorithm with linear complexity is taken by Baraff (1996), based on a non-minimal coordinate set. The author notes that minimal coordinate methods are generally faster, but more complex to implement. A further drawback of this approach is the need for constraint stabilization, which can increase the cost for numerical time integration (Baraff 1996).

Improvements and specialization of the O(n_b) method are still an active research area. In a recent publication, a method for treating systems with very small mass and inertia parameters based on singular perturbation theory was presented by Arnold (2013). For applications of the recursive approach to elastic MBSs see (e.g., Sorge 1993; Shabana 1991; Bremer 2008; Bremer and Pfeiffer 1988).

3.3.1 Linear-Complexity Inverse Dynamics

Luh et al. (1980b) proposed an algorithm with linear complexity in the number of bodies for calculating the inverse dynamics of kinematic trees. The algorithm has subsequently been called Recursive Newton Euler Algorithm (RNEA), a term introduced by Featherstone (see Featherstone and Orin 2000).

With reference to (3.23) and (3.28), the task of inverse dynamics is to calculate the impressed forces $\boldsymbol{u}$ (usually actuator torques). Here, we give a derivation based on the projective form of the EoM (3.23) and the recursive kinematics equations, which are repeated here for convenience:

$$\mathbf{0} = \sum_{i=1}^{n_b} \boldsymbol{J}_i^T \left(\boldsymbol{M}_i \boldsymbol{a}_i + \boldsymbol{h}_i - \boldsymbol{u}_i \right), \tag{3.34}$$

$$\boldsymbol{J}_i = \begin{pmatrix} \boldsymbol{C}_{i1}\boldsymbol{J}_{J1} & \boldsymbol{C}_{i2}\boldsymbol{J}_{J2} & \dots & \boldsymbol{J}_{Ji} & \mathbf{0} & \dots \end{pmatrix}. \tag{3.35}$$

The original paper by Luh et al. (1980b) studies rigid bodies in $\mathbb{R}^3$, connected by translational and rotational joints and derives the recursive equations based on the Newton-Euler equations and the free body diagram of the system. We choose the derivation given in the following for consistency with the other material presented in this chapter, and because it appears more straightforward to extend it to more complex joint and body types. Combining both equations given above, we can write the i-th line of the EoM as:

$$\boldsymbol{J}_{Ji}^T\Big[\left(\boldsymbol{M}_i\boldsymbol{a}_i+\boldsymbol{h}_i-\boldsymbol{u}_i\right)+\sum_{k\in N_{Ci}}\boldsymbol{C}_{ki}^T\left(\boldsymbol{M}_k\boldsymbol{a}_k+\boldsymbol{h}_k-\boldsymbol{u}_k\right)\Big]=\boldsymbol{0}, \tag{3.36}$$

where N_{Ci} denotes the set of child bodies of i (cf. table 2.1). In this equation, the only unknowns are the impressed forces for the body i ($\boldsymbol{u}_i$) and its children ($\boldsymbol{u}_k$). All other terms may be calculated by forward recursion with $\mathrm{O}(n_b)$ operations (cf. section 2.4). The key observation is that for leaf nodes, $N_{Ci}=\emptyset$ by definition.[26] Since the columns of $\boldsymbol{J}_{Ji}$ are the unconstrained directions for the i-th joint, $\boldsymbol{u}_i=\boldsymbol{J}_{Ji}\bar{\boldsymbol{u}}_i$, where the dimension of $\bar{\boldsymbol{u}}_i$ corresponds to the DoFs of i. With this, (3.36) for a leaf node becomes:

$$\boldsymbol{0}=\boldsymbol{J}_{Ji}^T\left(\boldsymbol{M}_i\boldsymbol{a}_i+\boldsymbol{h}_i-\boldsymbol{J}_{Ji}\bar{\boldsymbol{u}}_i\right), \tag{3.37}$$

$$\Rightarrow \boldsymbol{u}_i=\boldsymbol{J}_{Ji}\left(\boldsymbol{J}_{Ji}^T\boldsymbol{J}_{Ji}\right)^{-1}\boldsymbol{J}_{Ji}^T\left(\boldsymbol{M}_i\boldsymbol{a}_i+\boldsymbol{h}_i\right). \tag{3.38}$$

The matrix $\boldsymbol{J}_{Ji}^T\boldsymbol{J}_{Ji}$ is always invertible, constant for standard joint types and scalar in the case of one-DoF joints. Substituting (3.38) into the equation (3.36) for the parent node yields:

$$\boldsymbol{J}_{Ji}^T\left(\boldsymbol{M}_i\boldsymbol{a}_i+\bar{\boldsymbol{h}}_i-\boldsymbol{u}_i\right)=\boldsymbol{0}, \tag{3.39}$$

$$\text{with:}\quad \bar{\boldsymbol{h}}_i=\boldsymbol{h}_i-\sum_{k\in N_{Ci}}\boldsymbol{C}_{ki}^T\underbrace{\Big[\boldsymbol{I}-\boldsymbol{J}_{Jk}\left(\boldsymbol{J}_{Jk}^T\boldsymbol{J}_{Jk}\right)^{-1}\boldsymbol{J}_{Jk}^T\Big]}_{\boldsymbol{N}_{Jk}}\left(\boldsymbol{M}_k\boldsymbol{a}_k+\boldsymbol{h}_k\right). \tag{3.40}$$

The matrix $\boldsymbol{N}_{Jk}$ is the nullspace projection matrix for the joint Jacobian $\boldsymbol{J}_{Jk}$, which projects any vector into the nullspace of $\boldsymbol{J}_{Ji}$ (see also section 5.1.2). This indicates that forces in the directions *blocked* by joint k are propagated from the child nodes back to the parents. Solving the above equation for $\boldsymbol{u}_i$ yields

$$\boldsymbol{u}_i=\boldsymbol{J}_{Ji}\left(\boldsymbol{J}_{Ji}^T\boldsymbol{J}_{Ji}\right)^{-1}\boldsymbol{J}_{Ji}^T\left(\boldsymbol{M}_i\boldsymbol{a}_i+\bar{\boldsymbol{h}}_i\right). \tag{3.41}$$

The same procedure may be applied recursively back to the root body (proof by induction). The algorithm is summarized in algorithm 3. It is also used as a component in forward dynamics algorithms (cf. section 3.2.3).

26. This corresponds to the "backward recursion property" of forces in trees (see above).

Algorithm 3: $O(n_b)$ inverse dynamics. Instead of using a loop, the forward and backward recursions can also be defined using recursive function calls and references from each body to parent and child bodies (cf. section 2.3.2).

input : Minimal coordinates, velocities and accelerations $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$
output: Generalized actuator forces $\boldsymbol{u}$

for $i = 1$ **to** n_b `// Forward recursion`
do
  calculate kinematics for i-th body $(\boldsymbol{r}_i, \dot{\boldsymbol{r}}_i, \ddot{\boldsymbol{r}}_i, \boldsymbol{A}_{0i}, \boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \boldsymbol{J}_{Ji}, \dot{\boldsymbol{J}}_{Ji})$
  calculate $\boldsymbol{h}_i$

for $i = n_b$ **to** 1 `// Backward recursion`
do
  calculate effective force $\bar{\boldsymbol{h}}_i$ (3.40)
  calculate $\boldsymbol{u}_i, \bar{\boldsymbol{u}}_i$ (3.41)
concatenate $\bar{\boldsymbol{u}}_i$ to obtain $\boldsymbol{u}$

3.3.2 Linear-Complexity Forward Dynamics

A number of different linear-complexity forward dynamics algorithms for MBSs with tree topology have been proposed (see above). In this section, we give a derivation roughly following the first known publication of the method by Vereshchagin (1974), but in a notation consistent with this monograph and extended to general tree-structured systems.

A Derivation Using Gauss' Principle
According to Gauss' principle, the minimal accelerations must fulfill (cf. (3.21), (2.75), coordinate frame indices omitted):

$$G = \sum_{i=1}^{n_b} G_i(\boldsymbol{a}_i) \quad \rightarrow \quad \text{min!}, \tag{3.42}$$

$$G_i(\boldsymbol{a}_i) = \frac{1}{2}\left(\boldsymbol{a}_i + \boldsymbol{M}_i^{-1}\boldsymbol{h}_i^{\star}\right)^T \boldsymbol{M}_i \left(\boldsymbol{a}_i + \boldsymbol{M}_i^{-1}\boldsymbol{h}_i^{\star}\right), \tag{3.43}$$

$$\boldsymbol{a}_i = \boldsymbol{C}_{ip}\boldsymbol{a}_{p(i)} + \boldsymbol{J}_{Ji}\ddot{\boldsymbol{q}}_i + \boldsymbol{b}_i. \tag{3.44}$$

Vereshchagin's key observation was that equations (3.42) to (3.44) have the structure of a *dynamic programming problem of an N-stage process*: The accelerations $\boldsymbol{a}_i$ may be seen as "state variables," the minimal accelerations $\ddot{\boldsymbol{q}}_i$ as the control input at stage i and the recursive kinematics equation (3.44) as the state transfer equation. That is, the joint index i is seen as the discrete time parameter (not the actual physical time) which is possible because of the direction introduced into the system graph due to the tree structure.

An efficient solution procedure for such problems is based on the *Principle of Optimality*, which states that: "An optimal policy has the property that whatever the initial state and the initial decision may be, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" (Bellman 1954, p. 47). Using this for applying Gauss' Principle to a kinematic *chain*, we get:

$$G_k^\star(\boldsymbol{a}_{k-1}) = \min_{\ddot{\boldsymbol{q}}_k}\Big[G_k(\boldsymbol{a}_{k-1},\ddot{\boldsymbol{q}}_k) + G_{k+1}^\star(\boldsymbol{a}_k(\boldsymbol{a}_{k-1},\ddot{\boldsymbol{q}}_k))\Big], \qquad k = n_b,\ldots,1. \tag{3.45}$$

The solution strategy is to determine the minimal accelerations at the last stage as a function of the predecessor accelerations. With this, the optimal choice for the predecessor minimal accelerations follow from (3.45). This procedure is then repeated recursively until the root body is encountered. In dynamic programming, there is no branching of the time parameter. This corresponds to the unbranched mechanisms studied by Vereshchagin (1974). Here, we extend the approach to branched systems, where "all remaining decisions" at the k-th stage are not encapsulated in G_{k+1}, but in all contributions by children of k:

$$G_k^\star(\boldsymbol{a}_{p(k)}) = \min_{\ddot{\boldsymbol{q}}_k}\Big[G_k(\boldsymbol{a}_{p(k)},\ddot{\boldsymbol{q}}_k) + \sum_{n\in N_{Ck}} G_n^\star(\boldsymbol{a}_k(\boldsymbol{a}_{p(k)},\ddot{\boldsymbol{q}}_k))\Big], \qquad k = n_b,\ldots,1. \tag{3.46}$$

Note that the backward-recursion following from the Principle of Optimality does not have to follow the bodies from n_b to 1 for branched systems, but can use any sequence ensuring that children are treated before parents.

For a terminal body i the condition for a minimum of G_i is:

$$\frac{\partial G_i}{\partial \ddot{\boldsymbol{q}}_i} = \left(\frac{\partial \boldsymbol{a}_i}{\partial \ddot{\boldsymbol{q}}_i}\right)^T \left(\boldsymbol{M}_i\boldsymbol{a}_i + \boldsymbol{h}_i^\star\right), \tag{3.47}$$

$$= \boldsymbol{J}_{Ji}^T\Big[\boldsymbol{M}_i\left(\boldsymbol{C}_{ip}\boldsymbol{a}_{p(i)} + \boldsymbol{J}_{Ji}\ddot{\boldsymbol{q}}_i + \boldsymbol{b}_i\right) + \boldsymbol{h}_i^\star\Big] = \boldsymbol{0}. \tag{3.48}$$

Solving for $\ddot{\boldsymbol{q}}_i$ gives:

$$\begin{aligned}
&\Rightarrow \ddot{\boldsymbol{q}}_i = \boldsymbol{P}_i\boldsymbol{a}_{p(i)} + \boldsymbol{p}_i,\\
&\text{with:}\\
&\quad \boldsymbol{P}_i = -\left(\boldsymbol{J}_{Ji}^T\boldsymbol{M}_i\boldsymbol{J}_{Ji}\right)^{-1}\boldsymbol{J}_{Ji}^T\boldsymbol{M}_i\boldsymbol{C}_{ip},\\
&\quad \boldsymbol{p}_i = -\left(\boldsymbol{J}_{Ji}^T\boldsymbol{M}_i\boldsymbol{J}_{Ji}\right)^{-1}\boldsymbol{J}_{Ji}^T\left(\boldsymbol{M}_i\boldsymbol{b}_i + \boldsymbol{h}_i^\star\right).
\end{aligned} \tag{3.49}$$

According to (3.46), we must minimize the sum $G_k + \sum_{n\in N_{Ck}} G_n^\star$ in the next step:

$$\boldsymbol{0} = \frac{\partial}{\partial \ddot{\boldsymbol{q}}_k}\Big(G_k + \sum_{n\in N_{Ck}} G_n^\star\Big) \tag{3.50}$$

$$
\begin{aligned}
&= \left(\frac{\partial \boldsymbol{a}_k}{\partial \ddot{\boldsymbol{q}}_k}\right)^T \left[\boldsymbol{M}_k \boldsymbol{a}_k + \boldsymbol{h}_k^\star\right] \\
&+ \left(\frac{\partial \boldsymbol{a}_k}{\partial \ddot{\boldsymbol{q}}_k}\right)^T \sum_{n \in N_{Ck}} \left(\frac{\partial \boldsymbol{a}_n}{\partial \boldsymbol{a}_k}\right)^T \left[\boldsymbol{M}_n \left(\boldsymbol{C}_{nk} \boldsymbol{a}_k + \boldsymbol{J}_{Jn} \left(\boldsymbol{P}_n \boldsymbol{a}_k + \boldsymbol{p}_n\right) + \boldsymbol{b}_n\right) + \boldsymbol{h}_n^\star\right]
\end{aligned} \tag{3.51}
$$

$$
\begin{aligned}
=& \boldsymbol{J}_{Jk}^T \left[\boldsymbol{M}_k \boldsymbol{a}_k + \boldsymbol{h}_k^\star\right] \\
&+ \boldsymbol{J}_{Jk}^T \sum_{n \in N_{Ck}} \boldsymbol{C}_{nk}^T \Big[\underbrace{\boldsymbol{M}_n \left(\boldsymbol{C}_{nk} + \boldsymbol{J}_{Jn} \boldsymbol{P}_n\right)}_{\boldsymbol{K}_n} \boldsymbol{a}_k + \underbrace{\boldsymbol{M}_n \left(\boldsymbol{J}_{Jn} \boldsymbol{p}_n + \boldsymbol{b}_n\right) + \boldsymbol{h}_n^\star}_{\boldsymbol{k}_n} \Big]
\end{aligned} \tag{3.52}
$$

Regrouping the final equation leads to an identical structure to that found for the terminal body:

$$
\boldsymbol{0} = \boldsymbol{J}_{Jk}^T \left[\overline{\boldsymbol{M}}_k \boldsymbol{a}_k + \overline{\boldsymbol{h}^\star}_k\right],
$$

with:

$$
\begin{aligned}
\overline{\boldsymbol{M}}_k &= \boldsymbol{M}_k + \sum_{n \in N_{Ck}} \boldsymbol{C}_{nk}^T \boldsymbol{K}_n, \\
\overline{\boldsymbol{h}^\star}_k &= \boldsymbol{h}_k^\star + \sum_{n \in N_{Ck}} \boldsymbol{C}_{nk}^T \boldsymbol{k}_n.
\end{aligned} \tag{3.53}
$$

Accordingly, the solution is given by (3.49) and (3.53), if $\boldsymbol{M}, \boldsymbol{h}^\star$ are replaced by $\overline{\boldsymbol{M}}, \overline{\boldsymbol{h}^\star}$:

$$
\ddot{\boldsymbol{q}}_i = \boldsymbol{P}_i \boldsymbol{a}_{p(i)} + \boldsymbol{p}_i,
$$

with:

$$
\begin{aligned}
\boldsymbol{P}_i &= -\left(\boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{J}_{Ji}\right)^{-1} \boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{C}_{ip}, \\
\boldsymbol{p}_i &= -\left(\boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{J}_{Ji}\right)^{-1} \boldsymbol{J}_{Ji}^T \left(\overline{\boldsymbol{M}}_i \boldsymbol{b}_i + \overline{\boldsymbol{h}_i^\star}\right), \\
\boldsymbol{K}_i &= \overline{\boldsymbol{M}}_i \left(\boldsymbol{C}_{ip} + \boldsymbol{J}_{Ji} \boldsymbol{P}_i\right), \\
\boldsymbol{k}_i &= \overline{\boldsymbol{M}}_i \left(\boldsymbol{J}_{Ji} \boldsymbol{p}_i + \boldsymbol{b}_i\right) + \overline{\boldsymbol{h}_i^\star}, \\
\overline{\boldsymbol{M}}_i &= \boldsymbol{M}_i + \sum_{n \in N_{Ck}} \boldsymbol{C}_{ni}^T \overline{\boldsymbol{K}}_i, \\
\overline{\boldsymbol{h}^\star}_i &= \boldsymbol{h}_i^\star + \sum_{n \in N_{Ci}} \boldsymbol{C}_{ni}^T \overline{\boldsymbol{k}}_i.
\end{aligned} \tag{3.54}
$$

The resulting computational scheme is outlined in algorithm 4. For completeness, alternative derivations of the $O(n_b)$-method using the same notation are listed in the appendix (free body approach: appendix C.1; projective EoM approach: appendix C.2).

Algorithm 4: $O(n_b)$ forward dynamics for trees. Instead of using a loop, the forward and backward recursions can also be defined using recursive function calls and references from each body to parent and child bodies (cf. section 2.3.2). The vector $\boldsymbol{h}_i^\star$ includes impressed forces and moments $\boldsymbol{F}_i^i, \boldsymbol{T}_o^i$ due to gravity, actuation, etc., which are either provided externally (by a control algorithm) or computed from force laws as a function of $\boldsymbol{q}, \dot{\boldsymbol{q}}$. Note that force laws are often most naturally described in generalized coordinates, for example, for joint friction or joint drives. In such cases, equivalent impressed forces must be calculated and applied to all bodies which contributing to the relevant DoFs. In the example of joint actuators, the forces must be applied both to the driven body and its predecessor.

input : Minimal coordinates, minimal velocities and impressed forces
output: Minimal accelerations

for $i = 1$ **to** n_b `// First forward recursion`
do
  calculate kinematics for i-th body $(\boldsymbol{r}_i, \dot{\boldsymbol{r}}_i, \boldsymbol{A}_{0i}, \boldsymbol{\omega}_i, \boldsymbol{J}_{Ji}, \dot{\boldsymbol{J}}_{Ji}, \boldsymbol{C}_{ip}, \boldsymbol{b}_i)$

for $i = n_b$ **to** i `// Backward recursion`
do
  calculate $\boldsymbol{h}_i^\star$

$$\overline{\boldsymbol{h}^\star}_i \leftarrow \boldsymbol{h}_i^\star + \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{k}_k$$

$$\overline{\boldsymbol{M}}_i \leftarrow \boldsymbol{M}_i + \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{K}_k$$

$$\boldsymbol{p}_i \leftarrow -\left(\boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{J}_{Ji}\right)^{-1} \boldsymbol{J}_{Ji}^T \left(\overline{\boldsymbol{M}}_i \boldsymbol{b}_i + \overline{\boldsymbol{h}_i^\star}\right)$$

$$\boldsymbol{P}_i \leftarrow -\left(\boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{J}_{Ji}\right)^{-1} \boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{C}_{ip}$$

$$\boldsymbol{k}_i \leftarrow \overline{\boldsymbol{M}}_i \left(\boldsymbol{J}_{Ji} \boldsymbol{p}_i + \boldsymbol{b}_i\right) + \overline{\boldsymbol{h}_i^\star}$$

$$\boldsymbol{K}_i \leftarrow \overline{\boldsymbol{M}}_i \left(\boldsymbol{C}_{ip} + \boldsymbol{J}_{Ji} \boldsymbol{P}_i\right)$$

for $i = 1$ **to** n_b `// Second forward recursion`
do

$$\ddot{\boldsymbol{q}}_i \leftarrow \boldsymbol{P}_i \boldsymbol{a}_{p(i)} + \boldsymbol{p}_i$$

$$\boldsymbol{a}_i = \boldsymbol{C}_{ip} \boldsymbol{a}_{p(i)} + \boldsymbol{J}_{Ji} \ddot{\boldsymbol{q}}_i + \boldsymbol{b}_i$$

Run-Time Measurements

Calculation times for the Dill and Coil systems are given in figure 3.5 and figure 3.6. As expected, the $O(n_b)$ methods are vastly superior for very large n_b, especially for unbranched systems. For unbranched robots, the number of bodies will typically be

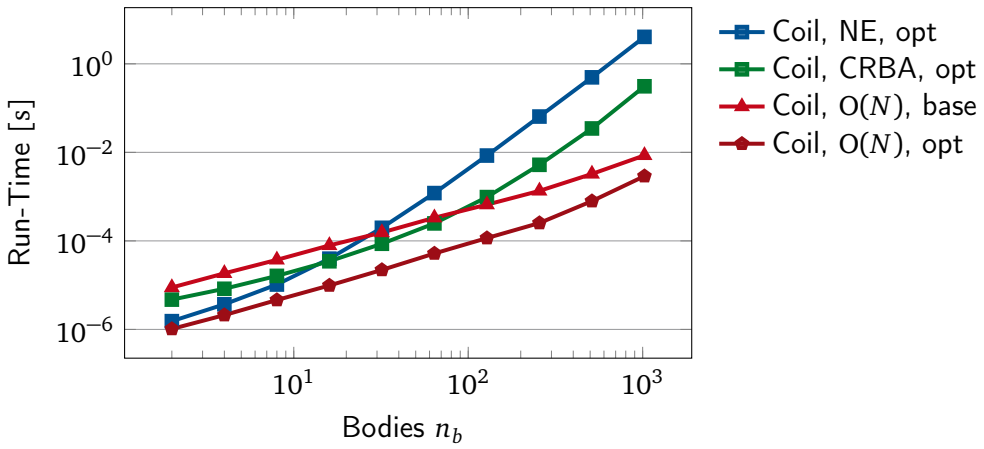


Figure 3.5: Run-times for calculating $\ddot{q}$ for Coil mechanisms using the optimized NE and CRBA methods (best versions from figure 3.3), as well as naive and optimized implementations of the O(n_b) method.

closer to 10 than 100 and more than 100 bodies are very unusual even for branched robots. The run-time measurements show that optimized NE and CRBA-based software is more efficient than a naive implementation of the O(n_b) method for fairly large systems (up to 10...100 bodies for chains and up to 100...400 bodies in branched systems, cf. figure 3.5, figure 3.6; see footnote 23 on page 47).

The results are in line with those by Featherstone, who found that for 30-DoFs systems "The O(n) algorithm beats the O(n^3) algorithm by a factor of 2.6 on the unbranched chain, but is only about 15% faster on the humanoid. This is mainly due to the O(n^3) algorithm running approximately 2.2 times faster on the humanoid [with 30 DoFs] than on the unbranched chain" (Featherstone 2005). We obtained similar results for naive implementations with RSC decomposition, where the CRBA algorithm is 2.41 times slower than the O(n_b) method for a Coil(32) mechanism and 1.31 times slower for a Dill(5) mechanism. Optimized versions of CRBA and NE are almost always faster than the naive O(n_b) algorithm, but optimized O(n_b) are always superior (see table 3.1 for numerical values).

Note on Run-Time Performance

It should be noted that Featherstone's cost analysis is based on counting the number of additions and multiplications, while the results given here are experimentally determined run-time measurements. Counting operations might appear to be a more rigorous method of determining the computational complexity of different algorithms and certainly does show the *asymptotic behavior of the number of arithmetic operations* for large n_b (cf. run-time for Jacobian calculations, figures 2.15 and figure 2.16). However, as noted by Baraff (1996), "It used to be that one could attempt to discuss the running times of algorithms based on the number of multiplications and additions;

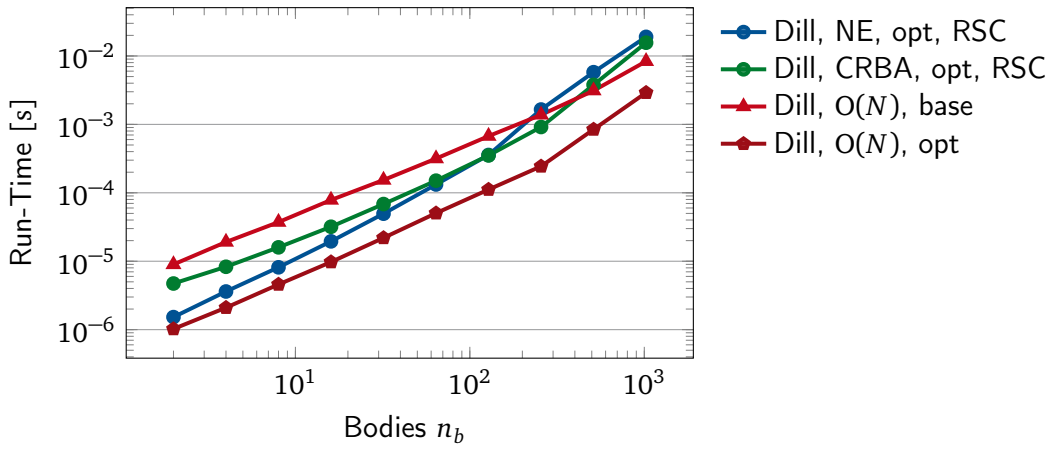


Figure 3.6: Run-times for calculating $\ddot{q}$ for Dill mechanisms using the optimized NE and CRBA methods (best versions from figure 3.4), as well as naive and optimized implementations of the O(n_b) method.

today, when a memory access may be as costly as a multiplication, such analysis no longer holds true." This remark is even more relevant today, since cache hierarchies in CPUs have become more complex, main memory access has become even more expensive in terms of CPU clock cycles and special instruction sets such as SSE are required to use the full potential of modern processors. In conclusion, the degree of code optimization is far more important for almost all moderately sized systems than the question if NE, CRBA or O(n_b) algorithms are used.

A close analysis of figure 3.5 and figure 3.6 shows that the scaling of the O(n_b) algorithm is in fact not exactly linear, but piecewise linear with a clear increase of the inclination at approximately 256 and 512 bodies.[27] That this is not an artifact of our implementation can be seen from figure 3.7, where run-times of the optimized algorithm are shown for two different CPUs, as well as for the commercial MBS code SIMPACK.[28]

The largest change in inclination occurs for 512 bodies on CPU A and for 256 bodies on CPU B. The kink is more pronounced in the curves showing run-times of our research code, but also clearly visible in the timings obtained from SIMPACK simulations.[29] This

27. This is more evident in a linearly scaled plot, see figure 3.7.

28. SIMPACK is a commercial MBS simulation program which implements the O(n_b) algorithm described by Brandl et al. (1986, 1987). For information regarding the program, see Rulka and Eichberger (1993) and `http://www.simpack.com`. The computational kernel is written in FORTRAN.

29. The simulations were performed using input files generated by Python scripts (`http://www.python.org`), which were written by Maximilian Grill as part of his semester project. A fixed step-size integrator was chosen and all output was deactivated, leading to a fixed number of 30000 solver steps per model. The average "wall clock time" reported by the solver was taken as run-time.

The fact that the kink in our code is more pronounced is probably because no attempt was made to eliminate unnecessary data structures in the research code. Instead, the classes contain various methods

	Run-time [s]	Run-time rel. to base $O(n_b)$	Run-time rel. to opt $O(n_b)$
Coil(32), $O(n_b)$, base	1.55×10^{-4}	(1.00)	7.00
Coil(32), CRBA, base	3.73×10^{-4}	2.41	16.90
Coil(32), NE, base	5.42×10^{-4}	3.50	24.53
Coil(32), $O(n_b)$, opt	2.21×10^{-5}	0.14	(1.00)
Coil(32), CRBA, opt	8.62×10^{-5}	0.56	3.90
Coil(32), NE, opt	1.98×10^{-4}	1.28	8.95
Dill(5), $O(n_b)$, base	1.54×10^{-4}	(1.00)	7.01
Dill(5), CRBA, base, sparse	2.01×10^{-4}	1.31	9.17
Dill(5), NE, base, sparse	5.41×10^{-4}	3.52	24.64
Dill(5), $O(n_b)$, opt	2.20×10^{-5}	0.14	(1.00)
Dill(5), CRBA, opt, sparse	6.85×10^{-5}	0.45	3.12
Dill(5), NE, opt, sparse	4.95×10^{-5}	0.32	2.26

Table 3.1: Run-times for branched and unbranched mechanisms with 32 bodies (Coil(32) and Dill(5)), using naive implementations of CRBA, NE and $O(n_b)$ methods, together with RSC decomposition for the Dill examples, compared to optimized implementations (cf. figure 3.3, figure 3.4, figure 3.5 and figure 3.6).

is due to the memory architecture used in most current computers: multiple levels of cache with different speeds and sizes are used to reduce the latency of read and write access to main memory. For very small systems, all data will fit in the fastest cache. With increasing n_b, the storage requirements increase, requiring access first to higher level caches and finally to main memory. Figure 3.8 shows the large difference in memory bandwidth for level one cache (L1), level two cache (L2), level three cache (L3) and main memory for the two CPUs. Since CPU A (green and blue lines) has 4096 kB of cache at the highest level CPU B (orange and red lines) only 2048 kB, the drop in bandwidth for the latter one appears at half the number of bodies (figure 3.7). The reduced speed of main memory increases the change of run-time per body for large systems by almost an order of magnitude. Counting operations is therefore not very meaningful, since for very large systems the actual performance is dominated by memory speed even for such a numerically intensive application as multibody simulation.

and data structures required for different algorithms. This leads to a larger memory footprint and a less optimal memory layout (i.e., distribution of data used for a specific algorithm in memory). Also, manual optimizations of our code were carried out for medium sized systems ($n_b < 100$).

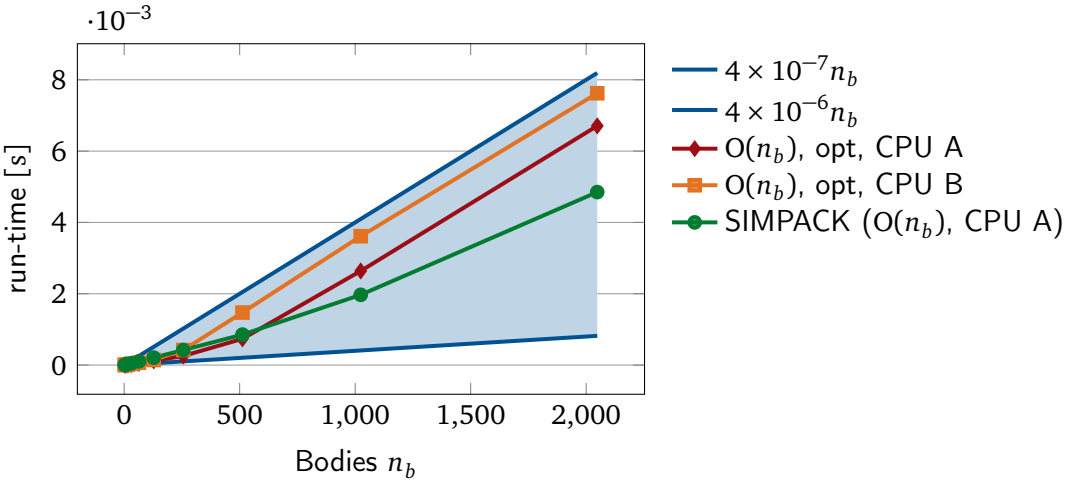


Figure 3.7: Run-times for kinematic chains (Coil mechanism) for research code run on two different CPUs (red: 4096 kB cache, orange: 2048 kB cache). The green line shows run-times using SIMPACK for the same models. Due to different implementations the memory requirements and therefore the position of the kinks are different, but also present. As a reference for comparison, the blue lines show perfectly linear behavior. The run-time per body increases by approximately one order of magnitude for large n_b, even though the algorithm is O(n_b). This is due to the speed difference between cache and main memory (see figure 3.8).

3.4 Kinematic Constraints

Kinematic constraints are present in any non-trivial system. If a minimal parametrization of the configuration is known, they can be eliminated by projection into the constraint-free directions, see (3.2), (3.1), (3.26). In case no minimal parametrization is known, a kinematically independent subset $\boldsymbol{q}_I$ of the redundant parametrization $\boldsymbol{q}^T = (\boldsymbol{q}_I \quad \boldsymbol{q}_D)$ can often be found. Together with the kinematic constraints, the EoM can then be projected into the constraint free directions after numerically determining the dependent parameters $\boldsymbol{q}_D$ (for the basic technique, see Shabana 2005; Bayazitoglu and Chace 1973, p. 416; for an application to a Hardware in the Loop (HiL) simulation, where $\boldsymbol{q}_D$ is determined using a velocity-level Inverse Kinematics (IK) algorithm, see Rulka and Pankiewicz 2005). Systematic methods for analyzing the kinematics of complex mechanisms and finding analytical solutions for the global kinematics as a function of a set of minimal coordinates were presented by Kecskemethy and Hiller (1993). The problem is related to IK in robotics (Hiller and Woernle 1988; Hiller et al. 1986), but extended and applied to the problem of automatically setting up the explicit EoM (3.26) by Kecskemethy and Hiller (1993). A less general variant of the method is

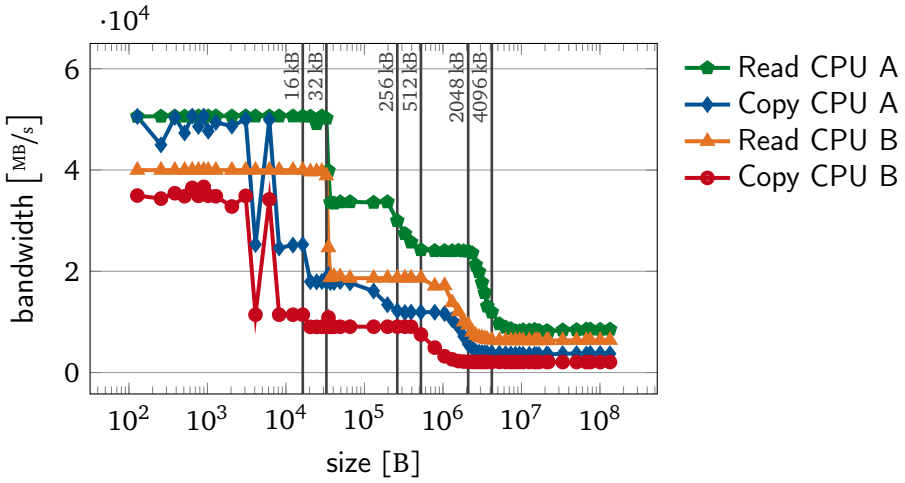


Figure 3.8: Memory bandwidth for sequentially reading and copying data using 128-bit transfers for two different CPUs.[c] The distinct drops in bandwidth are due to the memory architecture: CPU A has 32 kB L1 cache, 512 kB L2 cache, 4096 kB L3 cache; CPU B has 32 kB L1 cache and 2048 kB L2 cache; both CPUs have a 4 kB page size). The largest drop in read bandwidth occurs when the highest level cache is exhausted, since main memory access is much slower than all cached reads. For copying, the drops occur at half the cache sizes.[d] CPU A is the one used for other run-time measurements in this chapter, see footnote 23 on page 47.

c. Results obtained with the synthetic "Bandwidth" benchmarking program (available at `http://zsmith.co/bandwidth.html`)

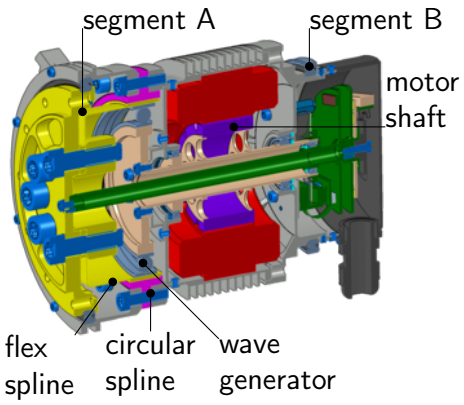
d. The oscillations visible for small data sizes are due to the intricacies of the memory architecture, which are beyond the scope of this monograph.

implemented in the Modelica simulation system.[30] While these and related techniques are important for general MBSs, complex interconnected kinematic loops are less common in robotics.

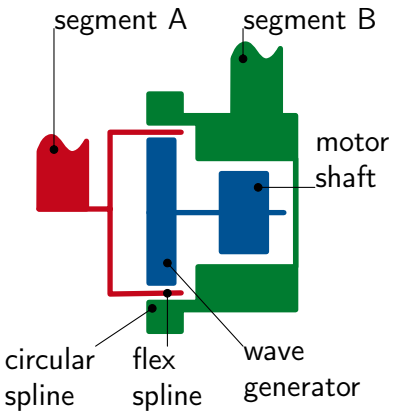
For tree-structured systems, efficient recursive forward dynamics procedures are available (see above), but their basic form cannot treat kinematic loops. Even though the $O(n_b)$ methods were developed with robots in mind, many robotic systems have kinematic loops. For serial robots with rigid gears, every drive introduces a kinematic loop (see figure 3.9). Modeling the gears as elastic eliminates the loops, but increases the number of DoFs and leads to a stiffer EoM, due to the high gear stiffness and low motor shaft inertia. The classical solution is based on a Lagrangian multiplier approach.

The following section reviews methods based on Lagrangian multipliers and section 3.4.2 describes a novel recursive linear-time algorithm for systems with kinematic

30. See `https://www.modelica.org/` for information about Modelica; for a description of the treatment of loops, see `https://build.openmodelica.org/Documentation/Modelica.Mechanics.MultiBody.Joints.Assemblies.html`; sites accessed 2013-11-12.



CAD drawing of joint drive with permanent magnet synchronous motor and Harmonic Drive gear.



Schematic diagram of joint module (left).

Figure 3.9: Computer Aided Design (CAD) drawing (top left) and schematic diagram (top right) of a typical robot joint with electric motor and reduction gear.[c] If segment B is closest to the root node, it is the parent of both segment A and the motor shaft. If all components are assumed to be rigid, the relative motion between segment A and the motor shaft is fully determined by the kinematic loop formed by wave generator, flex spline and circular spline.[d] Depending on the sizing of the gear relative to the loads during operation, deformations can be non-negligible. In that case, at least the deformation in the flex spline[e] must be taken into account. This removes the kinematic loop, leading to a tree structure.

c. Joint designed for Lola by Lohmeier (2010).

d. For details of the operational principle of Harmonic Drive gears, refer to the company website `http://www.harmonicdrive.com`

e. The flex spline is the pot-shaped component connecting segment A and segment B to the motor shaft.

loops.

3.4.1 Lagrangian Multiplier-based Methods

Basic Approach

For a system with constraints in a non-minimal parametrization, Gauss' principle of least constraint (see section 3.1) can be written as:

$$\begin{aligned} G &= \frac{1}{2}\left(\ddot{\boldsymbol{q}} + \boldsymbol{M}^{-1}\boldsymbol{h}^{\star}\right)\boldsymbol{M}\left(\ddot{\boldsymbol{q}} + \boldsymbol{M}^{-1}\boldsymbol{h}^{\star}\right) \rightarrow \min! \\ \boldsymbol{\Phi}(\boldsymbol{q}) &= \boldsymbol{0}. \end{aligned} \tag{3.55}$$

Here, the free motion is obtained from the EoM without the constraints.[31] A standard quadratic optimization problem is obtained by differentiating the constraints until accelerations appear[32]

$$\begin{aligned} G &= \frac{1}{2}\left(\ddot{\boldsymbol{q}} + \boldsymbol{M}^{-1}\boldsymbol{h}^{\star}\right)\boldsymbol{M}\left(\ddot{\boldsymbol{q}} + \boldsymbol{M}^{-1}\boldsymbol{h}^{\star}\right) \rightarrow \min!, \\ \ddot{\boldsymbol{\Phi}}(\boldsymbol{q}) &= \left(\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}\right)\ddot{\boldsymbol{q}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}\right)\dot{\boldsymbol{q}} = \boldsymbol{0}. \end{aligned} \tag{3.56}$$

Following the method of Lagrangian multipliers, the solution is given by the stationary point of the Lagrangian $L = G - \boldsymbol{\lambda}^T \ddot{\boldsymbol{\Phi}}$:[33]

$$\begin{pmatrix} \frac{\partial L}{\partial \ddot{\boldsymbol{q}}} \\ \frac{\partial L}{\partial \boldsymbol{\lambda}} \end{pmatrix} = \boldsymbol{0}, \tag{3.57}$$

$$\Rightarrow \begin{pmatrix} \boldsymbol{M} & -\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}^T \\ -\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{h}^{\star} \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}\right)\dot{\boldsymbol{q}} \end{pmatrix}. \tag{3.58}$$

There are a number of methods for solving (3.58). Standard procedures can be found in textbooks on advanced engineering dynamics, multibody dynamics and robot dynamics (see, e.g., Ginsberg 1998; Shabana 2005; Featherstone 2008). The most straightforward approach is to solve (3.58) directly, preferably using a sparsity exploiting decomposition. Another approach is to solve for $\boldsymbol{\lambda}$, substitute the solution and then solve for $\ddot{\boldsymbol{q}}$.

31. Here we have assumed holonomic constraints. However, non-holonomic constraints can be treated without any difficulty.

32. This is equivalent to a reduction of the corresponding Differential Algebraic Equation (DAE) from index 3 to index 1. The index is the number of differentiations required in order to obtain an equivalent ode in all unknowns (here: in $\boldsymbol{q}, \boldsymbol{\lambda}$, cf. (3.58)).

33. A minus sign is used in front of the Lagrangian multiplier $\boldsymbol{\lambda}$ in order to have the usual direct connection to constraint forces.

Constraint Stabilization

While the EoM (3.58) with constraints at the acceleration level is mathematically exact, it is numerically unstable and leads to unbounded numerical drift since it contains an "open-loop" integration of $\ddot{\mathbf{\Phi}}$. A simple solution to obtain a numerically stable system is to close the loop artificially by introducing a PD-type feedback to stabilize the system around $\mathbf{\Phi} = \mathbf{0}$ (Baumgarte 1972):

$$\ddot{\mathbf{\Phi}} + 2\alpha\dot{\mathbf{\Phi}} + \beta^2\mathbf{\Phi} = \mathbf{0}. \tag{3.59}$$

Typically, the feedback parameters are chosen such that:

$$\ddot{\mathbf{\Phi}} + 2\alpha\dot{\mathbf{\Phi}} + \alpha^2\mathbf{\Phi} = \mathbf{0}. \tag{3.60}$$

Substituting (3.60) into (3.58) yields the stabilized form of the EoM:

$$\begin{pmatrix} \boldsymbol{M} & -\frac{\partial \mathbf{\Phi}}{\partial \boldsymbol{q}}^T \\ -\frac{\partial \mathbf{\Phi}}{\partial \boldsymbol{q}}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{h}^\star \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathbf{\Phi}}{\partial \boldsymbol{q}}\right)\dot{\boldsymbol{q}} + 2\alpha\dot{\mathbf{\Phi}} + \alpha^2\mathbf{\Phi} \end{pmatrix}. \tag{3.61}$$

The method has the appeal of being easy to understand and to implement: a spring-damper system introduces fast dynamics that keep the system within a neighborhood of the constraint manifold $\mathbf{\Phi} = \mathbf{0}$. A criticism of this method is the need for tuning the stabilization parameter(s) (α, β) and that the numerical solution based on (3.61) no longer satisfies the "exact" constraints $\mathbf{\Phi} = \mathbf{0}$. However, the "exact" constraint $\mathbf{\Phi} = \mathbf{0}$ is itself an idealization of real constraints, often imposed by joints with imperfect geometries and finite stiffness. In practical applications, this error can therefore usually be neglected (cf., e.g., Bremer 2008). A further point of criticism is the fact that large stabilization parameters can lead to stiff ODEs, requiring small integrator time steps and leading to an inefficient solution procedure. Constraint violation due to numerical drift and the effect of Baumgarte's stabilization method are illustrated in figure 3.10 and figure 3.11.

An alternative approach to stabilization is to re-project the solution obtained by integration onto the manifold $\mathbf{\Phi} = \mathbf{0}$ (see e.g., Ascher et al. (1995, 1994) and Schiehlen (1997); see also Schoeder et al. (2013) for a discussion of the non-smooth case). According to Arnold, "...the practical use of Baumgarte's approach is restricted to small scale models .... In off-line simulation, it is today state-of-the-art to avoid the drift-of effect by projection techniques" (Arnold 2009, p. 31). Since robot models are usually "small" in terms of multibody systems, the first point is less relevant in this context. However, the fact that there is always a finite loop-closure error and reducing it increases the numerical cost remains (see Arnold 2009).

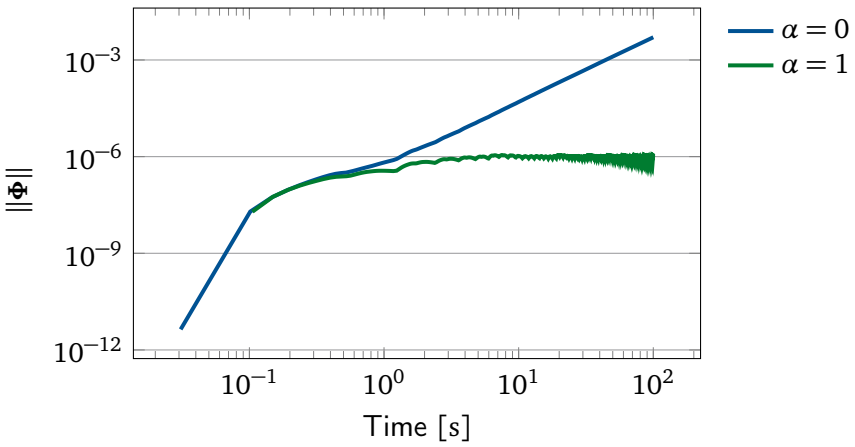


Figure 3.10: Illustration of the influence of the stabilization parameter α for simulating a constrained system based on (3.61). The example system is a simple planar pendulum with unit mass and rod length and a gravitational acceleration of $9.81\,\mathrm{m/s^2}$. The system was integrated for 100 s using Matlab's ode45 with absolute and relative tolerances of 1×10^{-6} starting in a horizontal position. The plot shows the constraint violation for $\alpha = 0$ and $\alpha = 1$. Without stabilization, numerical drift leads to a quadratic increase of the constraint violation error (inclination two in log-log graph), while $\alpha > 0$ leads to a bounded error with a magnitude dependent on α (as well as system and integrator parameters).

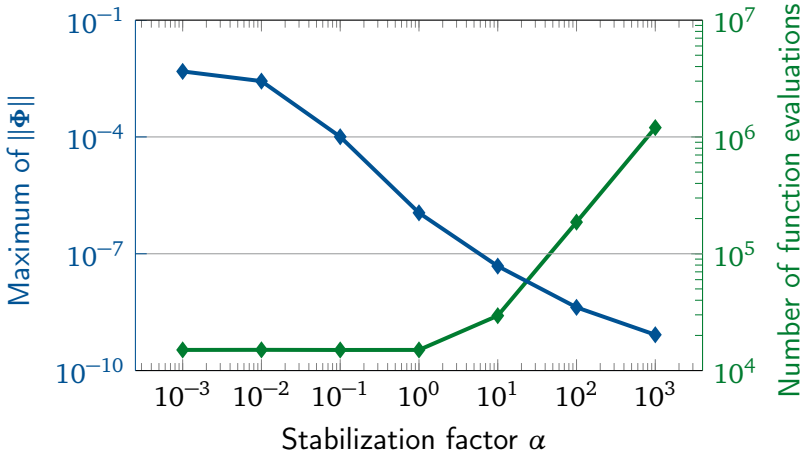


Figure 3.11: Maximum constraint violation and required number of function calls depending on the stabilization parameter α for a fixed integrator error bound (same system as in figure 3.10). In this example, the constraint violation error is approximately equal to the error bounds for numerical integration. The required number of function evaluations is almost constant for a small value of α. For the large values of α required for very high accuracy, however, the resulting stiffness of the ODE strongly increases the required number of time steps.

3.4.2 Minimal Coordinate Linear-Complexity Forward Dynamics with Loops

Introduction and Related Work

As discussed in the previous section, arbitrary kinematic constraints can be treated by the Lagrangian multiplier method. Lagrangian multipliers can also be used to extend the recursive $O(n_b)$ algorithms discussed in section 3.3.2 to general systems with loops. The special case of serial manipulators with endpoint constraints was studied by Lathrop (1986). To treat arbitrary constraints, a spanning tree must be defined by cutting all kinematic loops and introducing corresponding kinematic constraints and Lagrangian multipliers for the eliminated joints. The Lagrangian multipliers can be calculated within the recursive procedure at the point where the last body in a loop (or connected system of loops) is encountered. This operation is cubic in the number of constraint forces, which can increase the cost beyond that of a non-recursive method in the worst case (see Brandl et al. 1987; Bae and Haug 1987b; Sorge 1993).

The Lagrangian multipliers can also be computed in a global approach. Rather than computing $\boldsymbol{\lambda}$ during the recursive procedure, the method proposed by Gattringer (2011) and Gattringer and Bremer (2005) is based on the constrained EoM (3.58), (3.61). Instead of explicitly solving the equation for $\boldsymbol{\lambda}$, the multipliers can be solved by running the $O(n_b)$ algorithm $m + 2$ times for m constraints. Gattringer (2011) also uses the approach for recursively solving a system with a single impact without inversion of the mass matrix. In his PhD-thesis, Förg also applied the $O(n_b)$ method to constrained systems (Förg 2007). The approach assumes that the multipliers are calculated iteratively in an outer loop. This simplifies the $O(n_b)$ algorithm, since $\boldsymbol{\lambda}$ is treated as a known impressed force, but it requires multiple runs of the algorithm, increasing the numerical cost.

Lagrangian multiplier-based methods have the merit of being able to treat arbitrary kinematic loops. The disadvantage is the use of a non-minimal coordinate representation, increasing the number of equations and requiring the use of either constraint stabilization methods (see section 3.4.1) or DAE solvers.

In the case of robotic systems, geared drives are a common cause for kinematic loops (see figure 3.9). For the simplest case of a tree with only revolute joints and one motor and gear per joint, the DoFs are doubled and, additionally, n_b Lagrangian multipliers are introduced.[34] This motivates the search for methods to efficiently calculate the generalized accelerations in $O(n_b)$ time, while still using a minimal parametrization of the system to decrease numerical cost, eliminate the need for parameter tuning and avoid any constraint violation.

Schwienbacher (2013) developed a method for recursively calculating the forward dynamics in linear time using minimal coordinates for systems with kinematic loops. The method uses the original graph representation of the system and automatically

34. The number of multipliers depends on details of the formulation.

identifies bodies that are part of at least one kinematic loop. All bodies within a loop (or multiple connected loops) are added to a subsystem. The $O(n_b)$ algorithm can then be run on the system consisting of individual bodies and subsystems.

The concept of subsystems was previously proposed as a tool for simplifying the modeling of complex MBS (Bremer 1987; Bremer and Pfeiffer 1988). An application of the subsystem concept to the recursive solution of forward dynamics was presented by Bremer (2008), Bremer and Pfeiffer (1988), Gattringer (2011), and Gattringer and Bremer (2005). The subsystems are manually defined during the modeling procedure. The process enables the treatment of loops, but subsystems and the EoM of each subsystem must be defined manually. The procedure uses an extended state vector for each subsystem with > 6 DoFs to allow an explicit calculation of the full subsystem state without referring to the state of the parent, which incurs additional overhead.[35] The method presented here is different, because it is based on an automatic analysis of the unmodified graph of the MBS and only introduces subsystems where this is required by the presence of loops. The MBS still explicitly contains the individual bodies, which are used, for example, for various kinematics calculations. Essentially, the "subsystems" used here introduce a hierarchical definition of a MBS as a set of abstract "bodies", which in turn can themselves be small MBSs (cf. figure 2.11). Also, the minimal coordinates of the subsystem only include the relative DoFs with respect to the predecessor, which reduces the size of the vectors and matrices used within the algorithm.

The automatic generation of subsystems is based on first automatically building a graph of the system and assigning DoFs by traversing the graph. Bodies within a loop or group of connected loops will use the same indices into the vector of generalized coordinates for describing relative kinematics. This is in turn used to move all of these bodies into a subsystem according to figure 2.11 (for details of the graph restructuring algorithm, refer to Schwienbacher 2013).

Recursive $O(n_b)$ Algorithm with Subsystems

The recursive procedure for systems with loops follows the basic approach for tree structured systems (see section 3.3.2). If there are kinematic loops, however, we may encounter a group of m kinematically coupled bodies during the backward recursion. In that case, all minimal accelerations for these bodies are coupled and cannot be resolved with the standard recursive procedure. In such a case, the coupled bodies are combined to a subsystem, whose relative motion with respect to the common predecessor is described by the subsystem DoFs $\boldsymbol{q}_{Si}$ (cf. section 2.3.2, figure 2.11).

In order to apply the procedure used for tree structures in section 3.3.2, we can re-write (3.42) by grouping bodies into those that are part of one of the subsystems

35. The overhead will depend on the amount of symbolic pre-processing and the exploitation of sparsity patterns for specific subsystems. A certain amount of overhead is present due to the use of redundant "describing velocities" $\dot{\boldsymbol{y}}_i$ for the i-th body, with $\dot{\boldsymbol{y}}_i \in \mathbb{R}^{6+f_i}$ (f_i: no. of DoFs of the i-th body relative to its predecessor). The advantage is that one set of equations can be used for all subsystems.

and the remaining ones (see node sets defined in table 2.1):

$$G = \sum_{i \in n_b} G_i(\boldsymbol{a}_i) = \sum_{i \in N_B \setminus \overline{N}_S} G_i(\boldsymbol{a}_i) + \sum_{i \in N_S} G_{Si}, \tag{3.62}$$

$$G_{Si} = \sum_{k \in N_{Si}} G_k(\boldsymbol{a}_k). \tag{3.63}$$

When minimizing this function using the dynamic programming principle, we encounter two different cases: (1) the next contribution to G is from an individual body and (2) the next contribution is from a subsystem. The first case has already been treated in section 3.3.2. We can therefore assume that we are encountering a subsystem during the backward recursion process. In this case, (3.46) becomes:

$$G^\star_{Si}(\boldsymbol{a}_{p(si)}) = \min_{\ddot{\boldsymbol{q}}_{Si}} \left[G_{Si}(\boldsymbol{a}_{p(si)}, \ddot{\boldsymbol{q}}_{Si}) + \sum_{k \in N_{CSi}} G^\star_k(\boldsymbol{a}_k(\boldsymbol{a}_{p(sk)}, \ddot{\boldsymbol{q}}_{Si})) \right]. \tag{3.64}$$

The index $p(si)$ denotes the predecessor of the subsystem the i-th body belongs to, that is, $\boldsymbol{a}_{p(si)}$ is the predecessor's acceleration (or, equivalently, that of the subsystem node, cf. figure 2.11 and figure 2.7). Following the derivation in section 3.3.2, we first study the case of a terminal subsystem, that is, $N_{CSi} = \emptyset$. In that case:

$$\begin{aligned} G^\star_{Si}(\boldsymbol{a}_{p(si)}) &= \min_{\ddot{\boldsymbol{q}}_{Si}} \left[G_{Si}(\boldsymbol{a}_{p(si)}, \ddot{\boldsymbol{q}}_{Si}) \right], \\ \Rightarrow \boldsymbol{0} &= \sum_{k \in N_{Si}} \frac{\partial G_k(\boldsymbol{a}_k)}{\partial \ddot{\boldsymbol{q}}_{Si}} = \sum_{k \in N_{Si}} \left(\frac{\partial \boldsymbol{a}_k}{\partial \ddot{\boldsymbol{q}}_{Si}} \right)^T \left(\boldsymbol{M}_k \boldsymbol{a}_k + \boldsymbol{h}^\star_k \right). \end{aligned} \tag{3.65}$$

This is obviously similar to the projective form of the EoM (3.26) used for the entire MBS. The difference lies in the fact that here we have an additional dependency on the acceleration of the subsystem's parent body. The acceleration of each body in the subsystem can be written as a function of the parent's acceleration and the minimal subsystem accelerations $\ddot{\boldsymbol{q}}_{Si}$ using the known subsystem kinematics:[36]

$$\boldsymbol{a}_k = \widehat{\boldsymbol{C}}_k \boldsymbol{a}_{p(si)} + \widehat{\boldsymbol{J}}_k \ddot{\boldsymbol{q}}_{Si} + \widehat{\boldsymbol{b}}_k. \tag{3.66}$$

The newly introduced quantities are obtained by recursively applying the relative kinematics equations (2.75) and (2.81) until we arrive at the subsystem's root node:

$$\widehat{\boldsymbol{C}}_k = \boldsymbol{C}_{kp(k)} \widehat{\boldsymbol{C}}_{p(k)} \qquad \text{for subsystem root:} \quad \widehat{\boldsymbol{C}}_k = \boldsymbol{I}_6 \tag{3.67}$$

$$\widehat{\boldsymbol{J}}_k = \frac{\partial \boldsymbol{a}_k}{\partial \ddot{\boldsymbol{q}}_{si}} = \boldsymbol{C}_{kp(k)} \widehat{\boldsymbol{J}}_{p(k)} + \widehat{\boldsymbol{J}}_{Jk} \qquad \text{for subsystem root:} \quad \widehat{\boldsymbol{J}}_k = \boldsymbol{0} \tag{3.68}$$

36. This is straightforward for many robotic systems. In cases without a closed-form solution, directly resorting to Lagrangian-multiplier methods might be more efficient.

$$\widehat{\boldsymbol{b}}_k = \boldsymbol{C}_{kp(k)}\widehat{\boldsymbol{b}}_{p(k)} + \boldsymbol{b}_k \qquad \text{for subsystem root:} \qquad \widehat{\boldsymbol{b}}_k = \boldsymbol{0} \tag{3.69}$$

Here (and in what follows), the hat $\widehat{(\cdot)}$ marks variables used for describing a body *within its subsystem*. Equation (3.68) is analogous to (2.81) for the entire MBS: $\widehat{\boldsymbol{J}}_k$ is a block matrix consisting of zero entries and the joint Jacobian $\boldsymbol{J}_{Jk}$, similar to $\overline{\boldsymbol{J}}_k$ for the entire MBS (cf. (2.82)), and $\widehat{\boldsymbol{J}}_k$ is similar to the corresponding global Jacobian $\boldsymbol{J}_k$ for the entire MBS. That is, quantities related to the subsystem are defined in analogy to those for the whole MBS, but retain an additional dependency on the predecessor body. After substituting the kinematic relationships, (3.65) becomes:

$$\sum_{k\in N_{Si}} \widehat{\boldsymbol{J}}_k^T \left[\boldsymbol{M}_k \left(\widehat{\boldsymbol{C}}_k \boldsymbol{a}_{p(si)} + \widehat{\boldsymbol{J}}_k + \widehat{\boldsymbol{b}}_k \right) + \boldsymbol{h}_k^\star \right] = \boldsymbol{0} \tag{3.70}$$

$$\Leftrightarrow \boldsymbol{M}_{Si}\ddot{\boldsymbol{q}}_{Si} + \boldsymbol{B}_{Si}\boldsymbol{a}_{p(si)} + \boldsymbol{h}_{Si}^\star = \boldsymbol{0}. \tag{3.71}$$

$$\text{With:} \quad \begin{cases} \boldsymbol{M}_{Si} := \sum\limits_{k\in N_{Si}} \widehat{\boldsymbol{J}}_k^T \boldsymbol{M}_k \widehat{\boldsymbol{J}}_k, \\ \boldsymbol{B}_{Si} := \sum\limits_{k\in N_{Si}} \widehat{\boldsymbol{J}}_k^T \boldsymbol{M}_k \widehat{\boldsymbol{C}}_k, \\ \boldsymbol{h}_{Si}^\star := \sum\limits_{k\in N_{Si}} \widehat{\boldsymbol{J}}_k^T \left(\boldsymbol{M}_k \widehat{\boldsymbol{b}}_k + \boldsymbol{h}_k^\star \right). \end{cases} \tag{3.72}$$

Solving (3.71) for $\ddot{\boldsymbol{q}}_{Si}$ gives:

$$\ddot{\boldsymbol{q}}_{Si} = \underbrace{-\boldsymbol{M}_{Si}^{-1}\boldsymbol{B}_{Si}}_{=:\boldsymbol{P}_{Si}} \boldsymbol{a}_{p(si)}, \underbrace{-\boldsymbol{M}_{Si}^{-1}\boldsymbol{h}_{Si}^\star}_{=:\boldsymbol{p}_{Si}} \tag{3.73}$$

$$\Rightarrow \ddot{\boldsymbol{q}}_{Si} = \boldsymbol{P}_{Si}\boldsymbol{a}_{p(si)} + \boldsymbol{p}_{Si}. \tag{3.74}$$

Similar to the O(n_b)-method for systems without loops, we can substitute $\ddot{\boldsymbol{q}}_{Si}$ into the corresponding equation for the parent node $i = p(si)$ of the terminal subsystem si. Child nodes of i that are *not* part of a subsystem can be eliminated using equations from the standard O(n_b)-method.[37] As for the case without loops, the next step consists of minimizing the sum of the contributions for the terminal subsystem and its parent body k, taking the solution (3.74) for the terminal node into account. Since all children are assumed to be members of a subsystem,[38] we can split the sum in (3.50) into sums over the individual subsystems:

$$\boldsymbol{0} = \frac{\partial}{\partial \ddot{\boldsymbol{q}}_k} \left(G_k + \sum_{l\in N_{CSk}} \sum_{n\in l} G_n^\star \right) \tag{3.75}$$

37. In that case, $\boldsymbol{M}_k, \boldsymbol{h}_k^\star$ is replaced by $\overline{\boldsymbol{M}}_k, \overline{\boldsymbol{h}}_k^\star$, see section 3.3.2.
38. This is possible, since single bodies have already been eliminated by the standard O(n_b) method.

$$
\begin{aligned}
&= \left(\frac{\partial \boldsymbol{a}_k}{\partial \ddot{\boldsymbol{q}}_k}\right)^T \Bigg\{ \boldsymbol{M}_k \boldsymbol{a}_k + \boldsymbol{h}_k^\star \\
&+ \sum_{l \in N_{CSk}} \sum_{n \in l} \left(\frac{\partial \boldsymbol{a}_n}{\partial \boldsymbol{a}_k}\right)^T \left[\boldsymbol{M}_n \left(\widehat{\boldsymbol{C}}_{nk} \boldsymbol{a}_k + \widehat{\boldsymbol{J}}_n \left(\boldsymbol{P}_{Sn} \boldsymbol{a}_k + \boldsymbol{p}_{Sn} \right) + \widehat{\boldsymbol{b}}_n \right) + \boldsymbol{h}_n^\star \right] \Bigg\}
\end{aligned} \tag{3.76}
$$

$$
\begin{aligned}
= \boldsymbol{J}_k^T \Bigg(& \boldsymbol{M}_k \boldsymbol{a}_k + \boldsymbol{h}_k^\star + \sum_{l \in N_{CSk}} \Bigg[\underbrace{\sum_{n \in l} \widehat{\boldsymbol{C}}_{nk}^T \boldsymbol{M}_n \widehat{\boldsymbol{C}}_{nk}}_{=: \overline{\boldsymbol{M}}_{Sl}} + \sum_{n \in l} \widehat{\boldsymbol{C}}_{nk}^T \boldsymbol{M}_n \widehat{\boldsymbol{J}}_n \boldsymbol{P}_{Sn} \Bigg] \boldsymbol{a}_k \\
& + \sum_{l \in N_{CSk}} \left[\sum_{n \in l} \widehat{\boldsymbol{C}}_{nk}^T \boldsymbol{M}_n \widehat{\boldsymbol{J}}_n \boldsymbol{p}_{Sn} + \sum_{n \in l} \widehat{\boldsymbol{C}}_{nk}^T \boldsymbol{M}_n \widehat{\boldsymbol{b}}_n + \boldsymbol{h}_n^\star \right] \Bigg)
\end{aligned} \tag{3.77}
$$

$$
\begin{aligned}
\overset{(3.71)}{=} \boldsymbol{J}_k^T \Bigg(& \boldsymbol{M}_k \boldsymbol{a}_k + \boldsymbol{h}_k^\star + \sum_{l \in N_{CSk}} \underbrace{\left[\overline{\boldsymbol{M}}_{Sl} + \boldsymbol{B}_{Sl}^T \boldsymbol{P}_{Sl} \right]}_{\boldsymbol{K}_{Sl}} \boldsymbol{a}_k \\
& + \sum_{l \in N_{CSk}} \underbrace{\left[\boldsymbol{B}_{Sl}^T \boldsymbol{p}_l + \sum_{n \in l} \widehat{\boldsymbol{C}}_{nk}^T (\boldsymbol{M}_n \widehat{\boldsymbol{b}}_n + \boldsymbol{h}_n^\star) \right]}_{\boldsymbol{k}_{Sl}} \Bigg)
\end{aligned} \tag{3.78}
$$

$$
= \boldsymbol{J}_k^T \Bigg[\underbrace{\left(\boldsymbol{M}_k + \sum_{l \in N_{CSk}} \boldsymbol{K}_{Sl} \right)}_{=: \overline{\boldsymbol{M}}_{Sk}} \boldsymbol{a}_i + \underbrace{\left(\boldsymbol{h}_k^\star + \sum_{l \in N_{CSk}} \boldsymbol{k}_{Sl} \right)}_{=: \overline{\boldsymbol{h}}_{Sk}^\star} \Bigg] = \boldsymbol{0}. \tag{3.79}
$$

The last equation again has the same structure encountered for a single terminal body or subsystem (cf. (3.47)), allowing us (via induction) to continue with the recursive elimination of nodes until we encounter the root node. For the final algorithm, all inertia and force variables in the previous derivation must be replaced by the "effective" quantities including the contributions from bodies farther down the tree.

Comparing the result for systems with and without loops (3.53) suggests defining $\boldsymbol{C}_{Sl} =: \boldsymbol{I}_6$, conceptually treating the entire subsystem as a complex "meta-body" within the MBS, whose reference frame moves with the parent node. The full procedure for systems containing loops is summarized in algorithm 5, with detailed equations shown in algorithm 6, 7 and 8. An alternative derivation of the method based on a minimal example is given by Schwienbacher (2013).

Run-Time Measurements

We have analyzed the run-time performance of the algorithm using modified Coil and Dill mechanisms (cf. section 2.4): an additional body representing motor and gear components was added for every body in the Coil or Dill system. Figure 3.12 shows run-times for the driven Coil mechanism together with the unmodified Coil mechanism (i.e., without kinematic loops) with the same number of bodies. Since the overhead

Algorithm 5: $O(n_b)$ forward dynamics with loops using subsystems. Instead of using a loop, the forward and backward recursions can also be defined using recursive function calls and references from each body to parent and child bodies (cf. section 2.3.2).
The if clauses to choose the correct equations for subsystems or simple bodies are not explicitly necessary in a computer implementation. In our research code, the switch is implemented in C++ using virtual functions, ensuring that the correct function is called for every body-type. This mechanism is also used to provide optimized implementation for every kind of body (e.g., 6-DoF, 1-DoF rotational, optimized 1-DoF drive bodies, etc.).

input : Minimal coordinates, minimal velocities and impressed forces
output: Minimal accelerations

```
// Position and velocity dependent kinematics (algorithm 6)
ForwardRecursion1()
// Propagate forces and inertias (algorithm 7)
BackwardRecursion()
// Minimal accelerations and body accelerations (algorithm 8)
ForwardRecursion2()
```

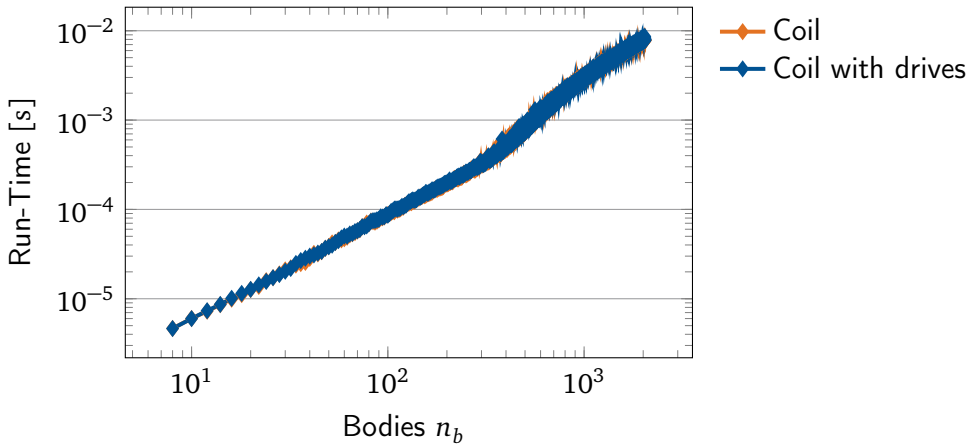


Figure 3.12: Run-times for calculating $\ddot{q}$ for a modified Coil mechanism with drive bodies. All calculations were performed with optimized versions of the $O(n_b)$ algorithm. Results for an open chain (Coil) with the same number of bodies shown for comparison. Since the overhead for resolving loops is minimal, run-times are practically identical. The oscillations in run-time for large systems are due to memory access issues (cf. see "Note on Run-Time Performance" on page 54, figure 3.7 and figure 3.8).

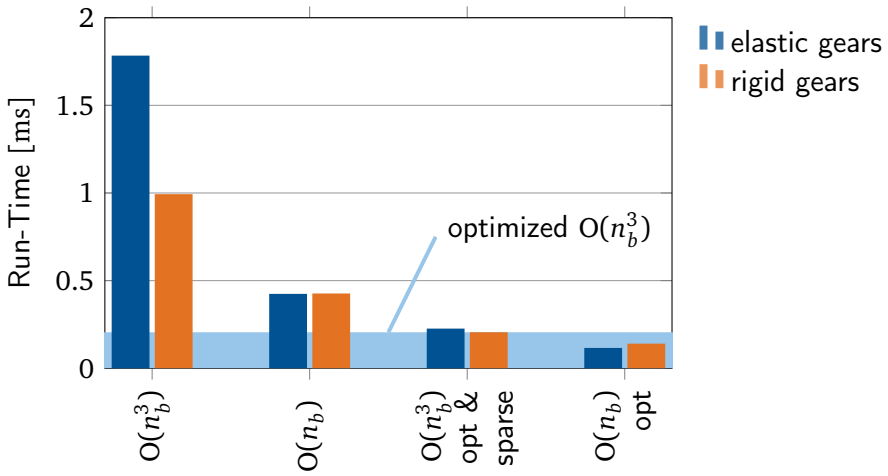


Figure 3.13: Run-times for different multibody models of the robot Lola (cf. figure 2.13). As with the synthetic Coil and Dill mechanisms, optimized O(n_b) implementations are the fastest, but optimized O(n_b^3) implementations are about twice as fast as naive O(n_b) implementations.

for resolving loops is minimal, run-times are practically identical for both systems, which demonstrates the efficiency of the proposed algorithm. The drawbacks of more widespread methods based on Lagrangian multipliers are the additional overhead for calculating the multipliers, a higher cost for time integration due to non-minimal coordinate representation, additional numerical error due to numerical loop closure and in some cases, depending on system parameters, smaller integrator time steps due to a higher stiffness introduced by constraint stabilization methods.

Figure 3.13 shows run-times for two different models of the bipedal robot Lola, which has 25 driven joints: (1) a model with rigid gears and (2) a model with elastic gears. The optimized O(n_b) implementation is the most efficient for both models. The reduction in run-time is very high for the model with elastic gears, due to the higher number of DoFs. Optimized implementations exploiting branch-induced sparsity, however, are not much slower for both models.

3.5 Outlook – Parallel Algorithms

Any serial forward dynamics algorithm obviously has at least O(n_b) complexity. Several algorithms have been proposed that reduce the overall run-time by using multiple processors in parallel. This section briefly reviews possibilities of speedups through parallel processing. Fijany and Bejczy (1991) proposed a parallel algorithm based on the O(n_b^3) method. Run on $n_b(n_b+1)/2$ processors it has O(n_b) complexity, but for reasonably small n_b, O(log(n_b))-terms dominate. A straightforward parallel implemen-

Algorithm 6: First forward recursion for O(n_b) forward dynamics with loops (see algorithm 5).

input : Minimal coordinates, minimal velocities
output: Velocity and position dependent kinematics

for $i = 1$ **to** n_b `// Forward recursion`
do
  if *i is a subsystem* `// Check if node is a subsystem or a simple body`
  then
    `// Initialize variables`
    $\widehat{\boldsymbol{b}}_i \leftarrow \boldsymbol{0}$
    $\widehat{\boldsymbol{C}}_i \leftarrow \boldsymbol{I}_6$
    $\widehat{\boldsymbol{J}}_{Ji} \leftarrow \boldsymbol{0}$
    forall the $k \in N_{Si}$ `// Iteration over subsystem member bodies`
    do
      calculate kinematics for body k $(\boldsymbol{r}_k, \dot{\boldsymbol{r}}_k, \boldsymbol{A}_{0k}, \boldsymbol{\omega}_k, \boldsymbol{J}_{Jk}, \dot{\boldsymbol{J}}_{Jk}, \boldsymbol{C}_{kp}, \boldsymbol{b}_k)$
      `// Calculate quantities w.r.t. the subsystem`
      $\widehat{\boldsymbol{b}}_k \leftarrow \boldsymbol{C}_{kp(k)} \widehat{\boldsymbol{b}}_{p(k)} + \boldsymbol{b}_k$
      $\widehat{\boldsymbol{C}}_k \leftarrow \boldsymbol{C}_{kp(k)} \widehat{\boldsymbol{C}}_{p(k)}$
      $\widehat{\boldsymbol{J}}_k \leftarrow \boldsymbol{C}_{kp(k)} \widehat{\boldsymbol{J}}_{p(k)} + \widehat{\boldsymbol{J}}_{Jk}$
  else
    `// Single body`
    calculate kinematics for body i $(\boldsymbol{r}_i, \dot{\boldsymbol{r}}_i, \boldsymbol{A}_{0i}, \boldsymbol{\omega}_i, \boldsymbol{J}_{Ji}, \dot{\boldsymbol{J}}_{Ji}, \boldsymbol{C}_{ip}, \boldsymbol{b}_i)$
    calculate $\boldsymbol{h}_i^\star$

Algorithm 7: Backward recursion for $O(n_b)$ forward dynamics with loops (see algorithm 5).

input : Kinematics from algorithm 7 and impressed forces
output: Intermediate variables for backward recursion algorithm 8

for $i = n_b$ **to** 1 `// Backward recursion`
do

if *i is a subsystem* **then**

forall the $k \in N_{Si}$ `// iteration over subsystem member bodies`
do

calculate $\boldsymbol{h}_k^\star$
`// Effect of child nodes that are not in the subsystem`

$$\widehat{\boldsymbol{h}}_k^\star \leftarrow \boldsymbol{h}_k^\star + \sum_{j \in (C_{Si} \cap N_{Ck})} \boldsymbol{C}_{jp}^T \boldsymbol{k}_j$$

$$\widehat{\boldsymbol{M}}_k \leftarrow \boldsymbol{M}_k + \sum_{j \in (C_{Si} \cap N_{Ck})} \boldsymbol{C}_{jp}^T \boldsymbol{K}_j$$

`// Calculate subsystem quantities`

$$\boldsymbol{M}_{Si} \leftarrow \sum_{k \in N_{Si}} \widehat{\boldsymbol{J}}_k^T \widehat{\boldsymbol{M}}_k \widehat{\boldsymbol{J}}_k$$

$$\overline{\boldsymbol{M}}_{Si} \leftarrow \sum_{k \in N_{Si}} \widehat{\boldsymbol{C}}_{ki}^T \widehat{\boldsymbol{M}}_k \widehat{\boldsymbol{C}}_{ki}$$

$$\boldsymbol{B}_{Si} \leftarrow \sum_{k \in N_{Si}} \widehat{\boldsymbol{J}}_k^T \widehat{\boldsymbol{M}}_k \widehat{\boldsymbol{C}}_k$$

$$\boldsymbol{h}_{Si}^\star \leftarrow \sum_{k \in N_{Si}} \widehat{\boldsymbol{J}}_k^T \left(\widehat{\boldsymbol{M}}_k \widehat{\boldsymbol{b}}_k + \widehat{\boldsymbol{h}}_k^\star \right)$$

$$\boldsymbol{P}_{Si} \leftarrow \boldsymbol{M}_{Si}^{-1} \boldsymbol{B}_{Si}$$

$$\boldsymbol{p}_{Si} \leftarrow -\boldsymbol{M}_{Si}^{-1} \boldsymbol{h}_{Si}^\star$$

$$\boldsymbol{K}_{Si} \leftarrow \overline{\boldsymbol{M}}_{Si} + \boldsymbol{B}_{Si}^T \boldsymbol{P}_{Si}$$

$$\boldsymbol{k}_{Si} \leftarrow \boldsymbol{B}_{Si}^T \boldsymbol{p}_i + \sum_{k \in N_{Si}} \widehat{\boldsymbol{C}}_{ki}^T (\boldsymbol{M}_k \widehat{\boldsymbol{b}}_k + \boldsymbol{h}_k^\star)$$

else

`// single body`
calculate $\boldsymbol{h}_i^\star$

$$\overline{\boldsymbol{h}^\star}_i \leftarrow \boldsymbol{h}_i^\star + \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{k}_k$$

$$\overline{\boldsymbol{M}}_i \leftarrow \boldsymbol{M}_i + \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{K}_k$$

$$\boldsymbol{p}_i \leftarrow -\left(\boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{J}_{Ji} \right)^{-1} \boldsymbol{J}_{Ji}^T \left(\overline{\boldsymbol{M}}_i \boldsymbol{b}_i + \overline{\boldsymbol{h}^\star}_i \right)$$

$$\boldsymbol{P}_i \leftarrow -\left(\boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{J}_{Ji} \right)^{-1} \boldsymbol{J}_{Ji}^T \overline{\boldsymbol{M}}_i \boldsymbol{C}_{pi}$$

$$\boldsymbol{k}_i \leftarrow \overline{\boldsymbol{M}}_i \left(\boldsymbol{J}_{Ji} \boldsymbol{p}_i + \boldsymbol{b}_i \right) + \overline{\boldsymbol{h}^\star}_i$$

$$\boldsymbol{K}_i \leftarrow \overline{\boldsymbol{M}}_i \left(\boldsymbol{C}_{ip} + \boldsymbol{J}_{Ji} \right) \boldsymbol{P}_i$$

Algorithm 8: Second forward recursion for $O(n_b)$ forward dynamics with loops (see algorithm 5).

input : Output from algorithm 6 and algorithm 7
output: Minimal accelerations $\ddot{\boldsymbol{q}}$, linear and angular accelerations for all bodies

for $i = 1$ **to** n_b `// Forward recursion`
do
  `// Calculate minimal accelerations`
  $\ddot{\boldsymbol{q}}_i \leftarrow \boldsymbol{P}_i \boldsymbol{a}_{p(i)} + \boldsymbol{p}_i$

  if *i is a subsystem* **then**
    $\boldsymbol{a}_i = \boldsymbol{a}_{p(i)}$ `// Subsystem node moves with parent`
    forall the $k \in N_{PSi}$ `// Nodes with children outside the subsystem`
    do
      $\boldsymbol{a}_k = \widehat{\boldsymbol{C}}_{ki} \boldsymbol{a}_i + \widehat{\boldsymbol{J}}_k \ddot{\boldsymbol{q}}_i + \widehat{\boldsymbol{b}}_k$
  else
    `// Single body`
    $\boldsymbol{a}_i = \boldsymbol{C}_{ip} \boldsymbol{a}_{p(i)} + \boldsymbol{J}_{Ji} \ddot{\boldsymbol{q}}_i + \boldsymbol{b}_i$

tation of recursive $O(n_b)$ schemes is to process each branch (or groups of branches) in parallel on a separate processor. Such a parallel implementation of forward and backward recursion steps was proposed by Bae et al. (1988), along with parallel matrix factorization methods.

A different class of algorithms for kinematic trees works by recursively disassembling and assembling sub-trees in order to enable parallel processing. The Divide and Conqueror Algorithm (DCA) proposed by Featherstone (1999a) achieves $O(\log(n_b))$ complexity when run on n_b processors. An extension to (almost) arbitrary topologies based on loop-cutting and Baumgarte-stabilization (Baumgarte 1972) is described by Featherstone (1999b). The asymptotic complexity is preserved for many systems, but worst-case complexity can be $O(n_b^3)$. Featherstone (1999b) also analyzes the computational cost of different parallel algorithms and the ABA by counting the number of additions and multiplications. If the parallel algorithm is run on n_b processors, the operations count is lower than for the $O(n_b)$ ABA for as few as 14 bodies. Operations counts for the DCA and ABA algorithms are shown in figure 3.14. It is important to note that for current processors, a similar speedup of the run-time is unlikely, since the locking and communication costs induced by such a massive parallelization are very large compared to the number of operations between synchronizations.

Yamane and Nakamura (2009) analyzed and compared the serial ABA-Algorithm with two parallel $O(\log(n_b))$ methods, Assembly Dissasembly Algorithm (ADA) by Yamane and Nakamura (2009), a method similar to Featherstone's DCA, and the

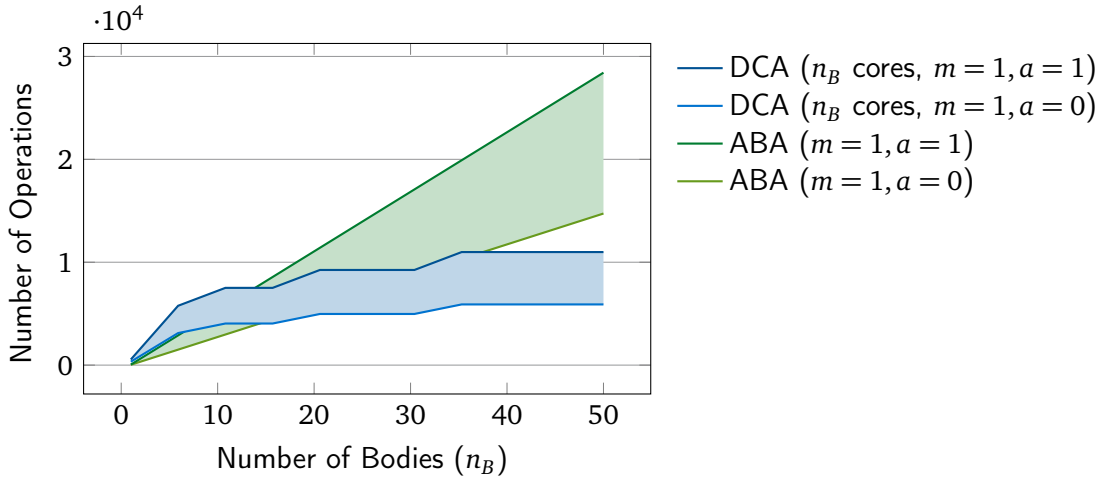


Figure 3.14: Number of operations for the ABA and DCA (on n_b processors). The areas show upper and lower limits of the number of operations, when multiplications (m) are given a weight of one and additions (a) a weight between zero and one (operations count equations from Featherstone 1999b).

Constraint Force Algorithm (CFA) proposed by Fijany et al. (1995). Differences in the parallel algorithms are mainly due to the formulation of the EoM for the subtree (or "articulated body") and the constraints.

Critchley et al. presented an alternative DCA, which is more efficient for a smaller number of processors and provides approximately 40% reduction of run-time for a 128-body chain on a four processor system.[39] (Critchley et al. 2007, figure 8)

In his PhD-thesis, Clauberg describes work on parallelizing MBS code. Speedups of a factor of 2.3…2.4 are achieved on an eight processor machine (Clauberg 2013, figure 4.3 and figure 4.4). The implementation is based on work by Förg (2007), which uses a redundant coordinate formulation in combination with bilateral and unilateral constraints. The mass matrix is always block diagonal, so its decomposition can be calculated efficiently. Lagrangian multipliers are calculated with a fixed-point iteration scheme, requiring the inversion of a matrix with the dimension of the Lagrangian multipliers (for details see Förg 2007).

For very large systems, parallelization can potentially provide significant speedups. For kinematic trees, special O(log(n_b)) algorithms have been developed. For typical system sizes in robotics, however, the gains are smaller than those of code-optimizations (see above). It is likely, however, that the importance of parallel algorithms will only become greater with the wider availability of "Many-Core" processors and more efficient, hardware-supported communication and locking mechanisms.

39. System specification: 3.2 GHz quad-processor Intel® Xeon® shared memory system running Linux. Programs compiled with GCC, POSIX threads and semaphores used for parallelization (Critchley et al. 2007).

3.6 Chapter Summary and Discussion

This chapter presents principles and methods from classical mechanics that form the basis of computational algorithms for forward and inverse dynamics of multibody systems. We presented both global EoM-based methods and recursive approaches in a uniform mathematical notation.

It is widely recognized that linear-time algorithms are the most efficient for tree structured systems. They provide a dramatic improvement of run-times over naive implementations of the NE method for very large kinematic chains. Efficient implementations of the NE method exploiting sparsity induced by the topology of the mechanism, however, are quite competitive for reasonably sized and/or branched systems.

While the classical linear-time algorithm uses Lagrangian multipliers and redundant coordinates to enable the simulation of systems with kinematic loops, a new formulation of a subsystem-based approach described in this chapter enables the simulation of mechanisms with loops using minimal coordinates.

The detailed run-time analyses in this chapter support the basic trends in run-time performance expected from counting the number of operations required per DoF. The actual run-times, however, vary strongly and depend on details of the implementation, the computer and memory architecture.

The algorithms outlined above can be used to provide the core of an efficient simulation program for robotic systems, as well as dynamics and kinematics routines required as building blocks for model-based controllers.

Realistic simulations must incorporate models for actuators (including friction, motor dynamics, possibly gear elasticity) and contact dynamics. Effectively, this adds additional ODEs for the actuator dynamics and algebraic force laws. For possible models for motor dynamics and gear friction see (Buschmann et al. 2006; Buschmann 2010; Baur et al. 2014) and references therein. In many cases, robot-environment interaction can be modeled as visco-elastic. For unilateral, visco-elastic contacts the resulting force law is non-smooth and requires the detection of contacts and the integration of a first-order ODE. A possible implementation for biped robots with visco-elastic soles walking on a rigid polygonal mesh is given in (Buschmann 2010).

Unilateral contacts between rigid bodies require special treatment. They are not considered here, since the contact elasticity is non-negligible in most robotic applications. For a treatment of non-smooth mechanical systems the reader is referred to Glocker (2006, 2000), Acary and Brogliato (2008), and Leine and Nijmeijer (2004) and references therein.

4 Biped Walking Control

This chapter reviews widely-used control strategies for biped walking robots. It is mostly based on a previous paper co-authored by me (Buschmann et al. 2014). This chapter, however, focuses on walking control for biped robots, as well as the posture and balance control of humans. It also adds references to other parts of the monograph and a discussion motivating the following chapter.

For a comprehensive review of the neural and robotic control of walking of two, four and six-legged systems with cross-references between the fields, see (Buschmann et al. 2014).

Section 4.1 briefly reviews posture and balance control of humans to provide the context for the following sections covering walking robots. Section 4.2 reviews typical hardware designs, since they determine the dynamics, the sensing capabilities and applicable forces. Section 4.3 covers basic concepts in the mechanics of legged locomotion, while the remaining sections review widely-used walking control approaches, which we have classified into model-based control of underactuated robots (section 4.4), model-based control of fully-actuated robots (section 4.5), biologically-based control (section 4.6) and gait cycle-centered control (section 4.7).

There are some parallels in the design and control of legged robots and animals, since they share the same basic principle of locomotion and are governed by the same physical laws. However, there are significant differences, mainly due to three reasons. (1) The physical properties of robots and animals differ with respect to sensing, actuation and mass distribution, mainly as a consequence of the currently available technology. (2) Since any robot must function as a whole, the focus in robot walking control is mostly on global coordination, postural control and balance, since these are the predominant problems for bipedal robots. In neuroscience, on the other hand, the complexities of the organisms and available experimental techniques often lead to a focus on low-level aspects of neural control. (3) Because there is no thorough understanding of the neural control of posture and balance in the Central Nervous System (CNS) (see section 4.1), robotic controllers are mostly based on first principles and control theory. Nevertheless, a brief review on the neural control of walking is given in the following, since it has and continues to influence research on robotic walking controllers (and vice versa).

4.1 Neural Control of Posture and Balance

Posture and balance control in walking animals is a complex neural task. Posture control concerns the generation of motor activity to produce and maintain an adequate

position and orientation of the animal relative to the vector of gravity and the foot locations. The actions of the skeleton-muscle system undertaken to prevent the body from falling out of any position taken during (quasi-) static motion and the actions taken to prevent falling during dynamic locomotion are called balance control (e.g., Winter 1995; Orlovsky et al. 1999; Deliagina et al. 2012). Legged animals differ in the control of posture and balance depending on the number of legs. For a review of the control of multilegged animals, see (Buschmann et al. 2014).

There are two central systems that play an important role in the balance and posture control of humans (for review see Orlovsky et al. 1999; Deliagina et al. 2012). The first is located in the spinal cord at the level of the legs. It receives input from limb mechanoreceptors about force, load and position.[1] It counteracts disturbances by generating corrective motor responses. The second is located in the motor cortex and the brain stem. It receives signals from limb mechanoreceptors, vestibular and visual signals (figure 4.1). These three sensory qualities contribute to balance control in a redundant fashion. This second system mediates its influence on the muscle-skeleton system by acting on those local networks in the spinal cord that generate motor activity controlling posture and movements. Which of the two systems contributes how much to the control of posture and balance is not clear yet (Buschmann et al. 2014).

The details of neural control of human balance are not well-understood, but basic postural reflexes and some fundamentals of the dynamics of locomotion are known from biomechanical studies. In stance, humans respond to external disturbances with a limited set of motor primitives (Horak and Nashner 1986). The dominant reaction is the *ankle-strategy*, when anterior and posterior disturbances are effectively counteracted by reflex activation of ankle plantarflexors and dorsiflexors to maintain balance. An alternative response is the *hip-strategy*, that is, flexion or extension of the hip. There also are mixed reactions containing ankle and hip activation, suggesting that these strategies are really *muscle synergies* underlying the neural control of balance (Torres-Oviedo and Ting 2007; Horak and Nashner 1986). Larger disturbances are compensated by a *stepping-strategy*, that is, by modifying the Base of Support (BoS), or even by hopping or stumbling motions (Horak 1987). It has been suggested that ankle-, hip- and stepping-strategy constitute a hierarchy of possible responses (Horak 1987), but more recent work indicates that all strategies are activated in parallel and that the stepping strategy is aborted if deemed unnecessary (Maki and McIlroy 1997). It has also been shown that prior knowledge of and repeated exposure to perturbations significantly

1. Vertebrates have different sensory organs that provide force, position and velocity information to the CNS (see Buschmann et al. 2014, and citations therein). The golgi-tendon organs (GTOs) are responsible for force information and are located in the tendons of all skeletal muscles. The Muscle Spindles (MSs) report information concerning muscle stretch and velocity. There are two types of MSs: static and dynamic, whose sensitivity is actively and separately modified by γ-motoneurons. Type Ia afferent fibers innervate both types of MSs, reporting velocity and position information to the CNS, while type IIa fibers only innervate static MSs and therefore mostly report length changes. The relative contribution of both types of MSs depends both on the specific muscle and the activation by γ-motoneurons.

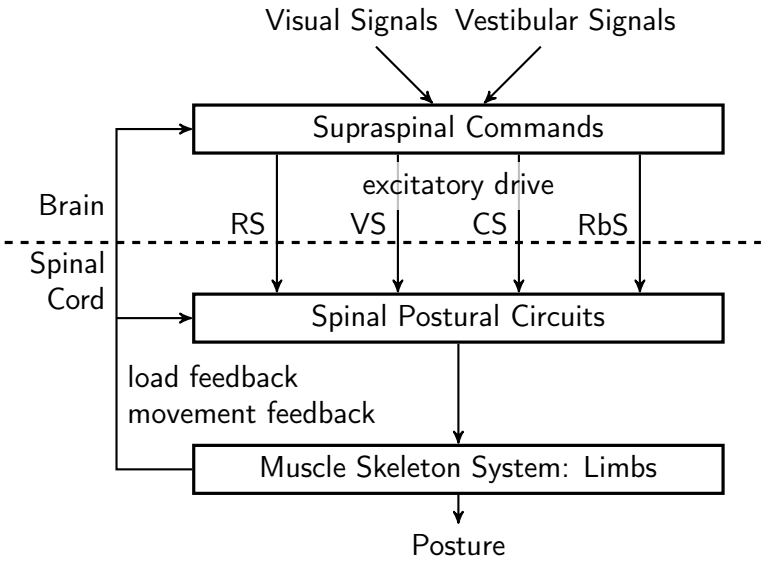


Figure 4.1: Main components of the postural system in quadrupeds. Two closed loop systems contribute to postural control. Spinal networks generate postural reflexes and their action is added to the influence of supraspinal commands, which originate from sensory information from visual and vestibular sensors and which are transmitted to the spinal networks via reticulaspinal (RS), vestibulospinal (VS), corticospinal (CS) and rubrospinal (RbS) tracts (modified from Deliagina et al. 2012).

changes the response (Marigold and Patla 2002; Horak and Nashner 1986). These three basic control strategies are also applied in many biped walking control algorithms (see section 4.5.2).

There have been some attempts to model and reproduce the overall behavior of human balance control during stance using methods from control theory. Fujisawa et al. (2005) analyzed the hip and ankle strategies in humans by studying responses to small disturbances during stance.[2] Using system identification techniques and modeling the human as a double pendulum, linear transfer functions from upper and lower body inclination to ankle and hip moment were determined. The study found proportional and derivative contributions in the response, but distinguishing the contributions of neural control and the muscle-skeleton system is not possible. A different model of stance control based on sensor fusion, disturbance estimation and linear control theory is discussed in (Tahboub and Mergner 2007; Mergner et al. 2009).

In biomechanical studies, balance in humans is mostly studied based on simple models such as the Inverted Pendulum Model (IPM) or Linear Inverted Pendulum Model (LIPM) and the concept of balance is usually not clearly connected to the mathematical theory of the stability of dynamical systems. In this field, balance is

2. Knee bending was prevented by wooden splints.

classically equated with keeping the CoM over the BoS (Winter 1995). Hof et al. (2005) proposed an extended *stability measure* based on an IPM, requiring that the eXtrapolated center of Mass (XcoM) remain within the BoS (cf. section 4.5.2) and proposed hypothetical walking control laws based on this concept (Hof 2008). Note that this analysis is based on a transformation of the pendulum equations previously proposed in the context of biped robots (see section 4.5). Aftab et al. (2012) proposed another hypothetical control law for stepping based on an optimal predictive controller for the LIPM and found similar responses to perturbations during stance also found in humans.

It should be noted that current state-of-the-art robots are not only capable of maintaining balance during stance, but also during dynamic locomotion and even in somewhat uneven terrain (see following sections). Neural control of posture and balance in humans is not well understood in such cases, partly because they are difficult to analyze experimentally with humans.

4.2 Hardware Design

Robot Structure

A wide range of different walking robots have been built, with quite different mechanical designs. The leg mechanisms are often designed as rigid links connected by rotary joints. The rigidity of the links is especially important for model-based control approaches: Measuring link deformations is difficult even with additional sensors such as strain gauges on structural components. Also, structural elasticity can lead to undesirable vibrations and link deformations can cause early or late ground contact, disturbing the gait. Some robots, however, intentionally include elasticity in the drives or in the telescopic legs of running robots (see below).

For fully-actuated bipeds, the most common design features six actuated joints per leg and feet enabling surface contact (Hirose and Ogawa 2006; Pfeiffer 2006; Nelson et al. 2012; Pratt et al. 2009; Ogura et al. 2006). Planar bipeds and some three dimensional systems use point feet and as few as two DoFs (Chevallereau et al. 2003; Miura and Shimoyama 1984).

Actuation

Differences in robot hardware in comparison to biology are to a large extent due to available actuator technologies (cf. Buschmann et al. 2014; Pons 2005). For many years, the prevalent choice of actuators for bipedal robots were geared brushed DC motors (Hirai et al. 1998; Nishiwaki et al. 2000; Pfeiffer et al. 2002). More recently, permanent magnet synchronous motors have been used for highly loaded joints (Lohmeier et al. 2009; Hirose and Ogawa 2006; Tsagarakis et al. 2011). Hydraulics (Nelson et al. 2012; G. Cheng et al. 2006; Raibert et al. 2008) and pneumatics (Vanderborght et al. 2005a, 2005b) are also used, but less often.

One motivation for using pneumatic actuators is their inherent compliance, which is also apparent in biological muscle-tendon systems where it has a significant impact on the dynamics of walking. An early attempt towards designing a human-like actuation system is described by Yamaguchi and Takanishi 1997. The humanoid Wabian has antagonistically driven joints incorporating nonlinear spring mechanisms, which enables the robot to control both the effective torque and stiffness of the joint, similar to animals and humans.

In a different approach to actuation involving compliance, a position controlled, geared electrical motor is connected to the output side by mechanical springs, a concept known as *series elastic actuation* (Pratt and Williamson 1995). The goal is to provide high-fidelity force control by measuring and controlling the spring deflection, which is proportional to the applied force, but also adds an unactuated DoF to the system and lowers the position control bandwidth. Robots using these devices include the 2D-biped Spring Flamingo (Pratt and Pratt 1998b) and the 3D-bipeds M2[3] and its more recent redesign M2V2 (Pratt and Krupp 2008). Similar to Robinson et al. (1999) and Pratt and Williamson (1995), more recent studies also cite high fidelity torque control as an advantage of series elastic actuation (Remy et al. 2012; Hutter et al. 2012). Several humanoid robots with compliant, antagonistic and bi-articular actuation were developed at the University of Tokyo[4](e.g., Shirai et al. 2011).

Elasticity is widely used in the actuation systems of one-legged and multi-legged hopping robots, but less so in bipeds. Hyon and Mita (2002) and others use mechanical springs, M. H. Raibert (1986) uses pneumatic cylinders and Brown and Zeglin (1998) bow-shaped legs. For an overview of the hardware design (and control principles) of one-legged hopping machines, see (Sayyad et al. 2007). Most notably, M. H. Raibert (1986) built a one-legged hopping machine to conduct experiments on balancing in three dimensions during continuous hopping. The leg is a pneumatic cylinder that is connected to the body by a gimbal joint. The lateral DoFs of the leg are actuated by hydraulic actuators. For details on the hardware design, refer to (M. H. Raibert 1986; Raibert et al. 1984).

Sensing

In contrast to biological walking systems, walking machines generally have a much lower number of sensors which are, on the other hand, often more precise than their biological counterparts. Most robots are equipped with motor encoders for position sensing, a smaller percentage also has torque sensing in the joints and/or link side encoders. Due to the importance of contact forces for balance, Force/Torque Sensors (FTSs) in the feet are a standard component in biped robots. FTSs are sometimes combined with or substituted by contact switches for detecting ground contact. Most robots also have Inertial Measurement Units (IMUs) in the upper body or pelvis which are used for balance control (Lohmeier et al. 2009; Tajima et al. 2009; Hirose and

3. `http://www.ai.mit.edu/projects/leglab/robots/m2/m2.html`, accessed 2013/18/10
4. `http://www.jsk.t.u-tokyo.ac.jp/research.html`, accessed 2013/18/10

Ogawa 2006; Kaneko et al. 2004; Nelson et al. 2012).

4.3 Basic Dynamics of Legged Locomotion

Machine locomotion can be categorized either as statically or as dynamically stable (cf. Berns et al. 1999; Bekey 2005; Full et al. 2002). The first formulation of a condition for static stability of a walking system is given by McGhee and Frank (1968), where the authors define static stability for walking on a horizontal plane as a configuration where the projected CoM lies within the current BoS. This criterion, however, is only suitable in the idealized quasi-static case, when inertial forces are negligible. In cases with significant accelerations, dynamic stability has to be investigated. See M. Raibert (1983) and Full et al. (2002) for a discussion of dynamic stability. Note that stability is not to be confused with feasibility, which is related to the set of possible contact forces compatible with the unilateral foot ground contact. See also section 4.5 and (Wieber 2002).

Walking machines are hybrid systems, since the foot-ground contact is unilateral: The contact states of the feet take on discrete values such as opened, closed, slipping or sticking, while at the same time the pose of the robot varies continuously. This hybrid nature is also found in the dynamics of walking systems: Trajectories of joint angles or joint torques are continuous functions of time, but landing positions of feet and stride lengths have discrete values for each step.

As with any controlled system, the resulting motion of the robot will depend both on the dynamics of the uncontrolled system, composed of the robot and its environment, and the walking controller. In the large body of work on walking control, the emphasis put on the design of either (passive) mechanics or control varies greatly, as does the emphasis put on either the discrete (step-to-step) or continuous aspects of walking. In figure 4.2 we give a rough overview of which aspects are emphasized in the predominant approaches to walking control described below. Figure 4.3 and 4.4 illustrate two different approaches towards stable bipedal walking described in more detail below: stabilization through contact force control (figure 4.4) and (almost) passively stable walking, augmented by some control and energy supply (figure 4.3).

Strictly speaking, every legged robot is underactuated, since the foot-ground contact always has finite stiffness and there is no input corresponding to the elastic deformation of the contact. Nevertheless, robots with one actuator per joint will be termed *fully actuated* in this monograph, if the remaining underactuation is only due to the deformation in the ground contact. This is to distinguish them from robots with point feet, with unactuated joints or additional elastic DoFs.

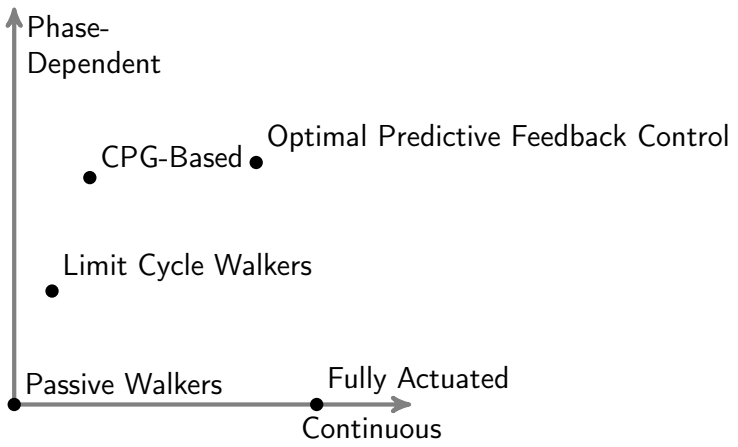


Figure 4.2: Schematic diagram of different walking control methods. The approaches are categorized according to the emphasis put on continuous feedback control and phase dependent or step-to-step control. In our view, an ideal walking controller would be situated in the central area of the diagram with equal emphasis on the discrete and continuous aspects of both dynamics and control. A possible implementation could be achieved by optimal control methods with properly chosen cost functions (figure from Buschmann et al. 2014).

4.4 Model-Based Control of Underactuated Robots

In this section, we give an overview of model-based control methods for underactuated hopping machines and biped robots.

Raibert's experiments with one-legged hopping machines (cf. section 4.2) and the general control strategy developed for them is a milestone in legged robot research. The control system is model-based and the control task is decomposed into the three parts *forward running velocity*, *body attitude* and *hopping height* (M. Raibert 1983). The employed control principle is sometimes referred to as *Raibert's three-part control* (Semini 2010). By introducing the *virtual leg* concept originally introduced by Sutherland and Ullner (1984), where multiple legs are considered to behave as one *virtual leg*, the *three part control* principle can be used to control walking machines with more than one leg (M. H. Raibert 1986).

An interesting theoretical framework for walking control that emphasizes the problem of underactuation was developed for the five-link planar robot Rabbit (Chevallereau et al. 2003; Westervelt et al. 2003). This robot uses four electric motors for actuation. The actuated joints, as well as the passive joints connecting the robot to the central boom enforcing the planar motion, are equipped with position sensors and the feet with contact force sensors. The robot has point feet, leaving one degree of underactuation at the foot-ground contact.

The controller enforces the same number of position-dependent *virtual constraints*

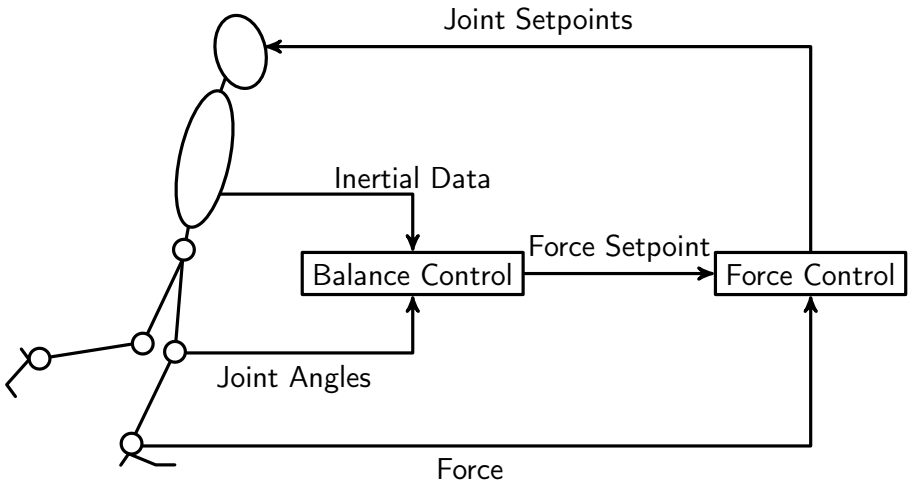


Figure 4.3: Structure of model-based biped balance control system based on force and inertial measurements following the approach found in (e.g., Buschmann et al. 2009; Takenaka et al. 2009c). The inertial data is used to calculate modified contact force setpoints, which are tracked via a low-level joint position control loop. Note that this is a simplified drawing emphasizing a basic principle. An actual implementation must at least add load distribution during double support, as well as account for swing-stance and stance-swing transitions (figure modified from Buschmann et al. 2014).

as can be directly controlled via the actuators, for example, four in this case. The constraints are not given as a function of time, but parameterized by some monotonic state variable, for example, the stance leg inclination. Since the chosen outputs (or virtual constraints) can easily be tracked by a position controller, stability is determined by the remaining internal dynamics of the system (the zero dynamics) that are not visible in the outputs. Since the robot is a hybrid dynamical system, the internal dynamics are called *hybrid zero dynamics*. A stable control system is designed by calculating virtual constraints using offline optimization methods that can enforce stable zero dynamics via inequality constraints in the optimization problem. The essence of the approach is to control only as many outputs as can be directly controlled and to choose these in such a way as to assure stability of the uncontrolled DoFs.

This approach has more recently been applied to the planar five-link biped Mabel (Sreenath et al. 2011). The basic structure is similar to Rabbit's, but forces are applied by compliant, antagonistic actuators, increasing the degree of underactuation and adding a mechanical energy storage and shock absorption mechanisms. Very recently, Park et al. (2013) expanded the system to include a component based on a Finite State Machine (FSM) for reacting to large changes in ground height. According to the detected system state (step-down, step-up, tripping, ...), modified virtual constraints are controlled. Additionally, active force control is added to stabilize the system and attenuate shocks when stepping off high platforms.

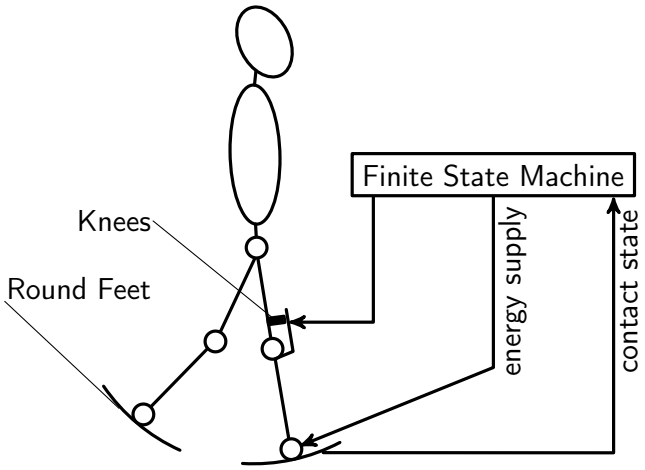


Figure 4.4: Semi-passive limit cycle walker with typical design elements. Round feet with a (passive or active) knee-locking mechanism enable a passive rocking motion during stance. Limited actuation is included to compensate for energy losses and control is performed by a simple finite state machine. The degree of actuation and control may vary from fully passive (McGeer 1990) to some actuation (Collins and Ruina 2005) or even almost full actuation with step position control (see Hobbelen and Wisse 2009; figure modified from Buschmann et al. 2014).

While a number of robots with elastic actuation mechanisms have been developed (see section 4.2), the question of how to utilize the compliance in walking machines remains an open problem. In the controller proposed by Vanderborght et al. (2005a) for the pneumatically actuated robot Lucy (cf. section 4.2), joint position references are tracked at a lower control level, compensating joint elasticity.

Robots with *series elastic actuators* are also underactuated (see section 4.2). However, since a low-level controller tracks the desired joint torque, the upper control levels can view the system as a torque controlled robot. Pratt et al. (2001) proposed using *Virtual Model Control* for such robots. In this approach, virtual forces acting on the robot are generated using intuitive, heuristic rules and then mapped to actuator forces via the corresponding Jacobians.

4.5 Model-Based Control of Fully-Actuated Robots

Many successful approaches to walking controller design for fully-actuated robots are based on physical models and classical control system design. This section reviews model-based control of fully-actuated bipeds.

From a control systems perspective, the major differences between the control of a (rigid) manipulator and walking control are the hybrid nature of the walking system, the underactuation and the inequality constraints imposed by the unilateral ground contact.

While questions such as low-level control of the joints can be efficiently handled with well-known methods from manipulator control, the additional tasks of (1) satisfying all constraints and (2) stabilizing the unactuated DoFs, while (3) exhibiting the desired walking behavior remain challenging.

Due to the highly nonlinear dynamics and the inequality constraints, a straightforward solution of the control problem, using, for example, optimal control methods, is currently impossible due to the computational costs. Nevertheless, a number of promising results from simulation studies using detailed models and optimization have been reported: Azevedo used nonlinear Model Predictive Control (MPC)[5] for a planar bipedal robot (Azevedo et al. 2002), Schultz presented optimal trajectories for a 3D model of a human (Schultz and Mombaur 2010) and Tassa reported a fast but non yet real-time MPC method for generating and stabilizing various motions of a humanoid robot (cf. Tassa et al. 2012 and section 5.6.1). Therefore, most control systems use a hierarchical design with higher levels responsible for overall behavior, intermediate levels for the discrete gait sequences and lower levels for task-space and joint-space motion (see Buschmann et al. 2009; Nishiwaki et al. 2002 and figure 4.5), thereby expressing organizational similarities to walking animals (see Buschmann et al. 2014). The control problem is often separated into planning ideal reference trajectories and modifying the references using feedback control. However, more recently there has been a trend towards incorporating sensor feedback into planning to improve long term stability, blurring the distinction between planning and feedback control (Tajima et al. 2009; Nishiwaki and Kagami 2006).

4.5.1 Planning Reference Trajectories

Since real-time planning using complex multibody robot models is currently computationally intractable, there are two dominating approaches to planning reference trajectories: (1) off-line planning using comprehensive models and (2) online planning using simplified models. Due to the importance of real-time gait generation for adapting to changes in the environment or the desired behavior, most systems today use real-time methods. For examples of offline planning methods see (Denk and Schmidt 2003; Buschmann et al. 2005; Bessonnet et al. 2004; Cho and Oh 2008). Most work on online planning is based on simplified lumped mass models, usually the LIPM (Kajita and Tani 1995). This model describes the linear dynamics of the CoM in the lateral and sagittal planes when the CoM height remains constant (see Buschmann 2010, for a discussion). Occasionally, additional terms for approximating the influence of leg motion and torso rotation are added (Pratt et al. 2006; Takenaka 2004b; Park and Kim 1998; Buschmann et al. 2007).

The simplified robot model is given as an ODE relating the CoM to the contact moment

5. That is, solving an optimal control problem, using the result for the next time step and then repeating the procedure at every time step.

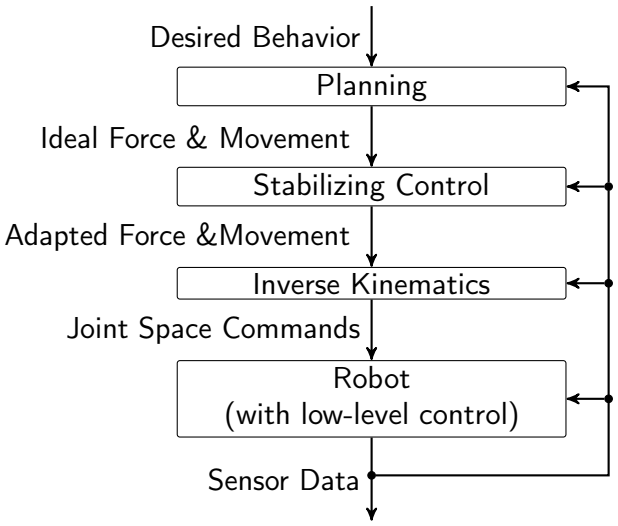


Figure 4.5: Example of a hierarchical control architecture with real-time planning and balance control for fully-actuated robots, as it is used in many state-of-the-art walking machines (e.g., Buschmann et al. 2009). Each level can have multiple sub-levels, for example, step-cycle and trajectory planning within the "Planning" level. Planning and control are typically formulated in terms of foot and CoG positions and contact forces. Joint space variables for low-level control are then obtained from inverse kinematics. Most early and some current implementations use offline planning, but online methods now dominate since they facilitate more reactive behavior. (figure from Buschmann et al. 2014).

at the foot. The planning system must determine contact force and CoM trajectories that satisfy the ODE and for which the forces satisfy the inequality constraints imposed by the unilateral ground contact. In biped walking control the Zero Moment Point (ZMP)[6] is often used instead of the contact moment, since it leads to a simple criterion for feasible contact forces when walking on flat ground: the ZMP must remain within the convex hull of the BoS (Vukobratović and Borovac 2004). This feasibility criterion is often mistakenly used as a *stability criterion*, even though satisfying it does not assure that the robot will not fall.[7] General feasibility criteria are discussed in (Hirukawa et al.

6. This is the point where the resulting contact force acts.

7. The relationship between ZMP and stability is controversial (Vukobratović et al. 2006; Sardain and Bessonnet 2004). We will only try to briefly describe the controversy. A central issue is the fact that Vukobratović distinguishes between the case when the ZMP is strictly within the BoS and the case when it is on the edge (Vukobratović et al. 2006), a fine point that is often overlooked. Vukobratović only refers to the first case as stable, since the ZMP may be moved towards the edge to stabilize the system. If the resulting contact force lies on the edge of the BoS, this point is no longer called the ZMP. This distinction separates the Center of Pressure (CoP) from the ZMP according to Vukobratović. Nevertheless, it is easy to design cases when the ZMP is at the center of the BoS and no feasible control can prevent the robot from falling. This is possible, since the state of the system is not uniquely defined by the

2006; Takao et al. 2003). Most planners first calculate an admissible force trajectory that can then be used to determine a CoM trajectory by solving the ODE. To assure a stable solution to the planning problem, this approach requires either a deviation from the ideal force trajectory by actively modifying the reference (Takenaka et al. 2009a; Buschmann et al. 2007), by only approximately satisfying it (Kajita et al. 2003b; Diedam et al. 2008), or by violating the desired boundary values for the CoM-trajectory (Sugihara and Nakamura 2005).

Proposed planning algorithms using this approach include analytical solutions (Harada et al. 2004; Morisawa et al. 2006; Löffler et al. 2004), numerical solutions by finite difference approximation (Kagami et al. 2002), a kind of shooting method (Takenaka 2004b; Takenaka et al. 2009a), methods based on model predictive control (Kajita et al. 2003b) and a method based on spline collocation (Buschmann et al. 2007). Diedam et al. (2008) proposed using linear MPC for solving the planning problem. The implementation is based on quadratic programming, enabling a simple extension of CoM-planning to footstep adaptation.

4.5.2 Stabilizing Control

General Walking Control

A large and growing number of control methods for stabilizing fully-actuated bipeds has been proposed and successfully implemented. The task of balance control (also called stabilizing control) is to assure a given reference motion while simultaneously preventing the robot from falling. Most methods build on the basic stabilizing mechanisms of (1) modifying the contact forces, (2) horizontally accelerating the CoM and (3) footstep control. These mechanisms are roughly equivalent with the (1) ankle-strategy, (2) hip-strategy and (3) stepping-strategy observed in animals (see section 4.1). Figure 4.3 illustrates the basic idea of balance control through contact force modification.

If the robot becomes unstable, it will diverge from the ideal, stable reference motion. A deviation of individual joint angles from the reference gait pattern is not critical, but the robot must avoid falling. Since a computationally tractable, practical and correct stability criterion for walking systems remains to be found (cf. Wieber 2002), the divergence of the total linear and angular momentum of the system may be taken as a practical measure of instability. The total linear and angular momentum of a system can only be changed by external forces acting on the system boundary. Modifying contact forces therefore is the only possible way of stabilizing the system. In fact, the hip-strategy changes contact forces by accelerating the CoM (or vice versa) and the stepping-strategy modifies the whole range of feasible forces by changing the BoS.

ZMP. The simplest examples are statically stable configurations for which a very high CoM velocity is chosen (Buschmann 2010). Regardless of the relationship of the ZMP to stability, the concept has been instrumental in the development of biped walking controllers.

The dominant method of stabilization via contact force control consists of modifying the contact moment in order to stabilize the upper body orientation or CoM position (Kajita et al. 2005; Löffler et al. 2004; Takenaka 2004a; Takenaka et al. 2009c; Buschmann et al. 2009). Usually, the contact moment is measured by the FTS in the foot and controlled via an underlying joint position controller. The landing impact may be reduced by an additional vertical impedance or force control (Löffler et al. 2004; Kajita et al. 2005; Nishiwaki and Kagami 2009; Buschmann et al. 2009), or an indirect force control via reduced joint position control gains (Tajima et al. 2009; Nishiwaki 2001; Hashimoto et al. 2009).

Accelerating the upper body or CoM for balance control is also widespread (Tajima et al. 2009; Nishiwaki and Kagami 2006). Takenaka et al. propose activating this strategy when the maximum admissible contact moment is reached (Takenaka et al. 2009c).

An interesting approach addressing the issue of underactuation was proposed by Chevallereau et al. (2008). Instead of tracking trajectories, only a reference path is tracked. This adds the path parameter as an additional control input, making planar bipeds fully actuated. This method is related to ideas from path-following control of robot manipulators (Pfeiffer and Johanni 1986).

Modifying the next foothold leads to a strong deviation from the originally intended motion, but at the same time is the most powerful method of stabilization. For the stepping-strategy the range for modifying the forces is limited by the maximum step length and stepping speed, while contact force control and upper body acceleration are limited by the size of the feet and the friction coefficient between the feet and the ground.

The stepping-strategy is also used by the one-, two- and four-legged hopping machines developed by M. H. Raibert (1986) and Hodgins and Raibert (1991). In this work the control of speed and balance is achieved by modifying the stepping position relative to a neutral point defined by the foothold corresponding to a periodic hopping motion (M. H. Raibert 1986; Raibert and Brown 1984). This strategy can be used very effectively for the hopping machines due to the very low leg mass and inertia.

Takenaka et al. (2009c) propose modifying the step positions in case modifications of the upper body trajectory induced by the hip-strategy cannot be compensated during the next step without changing the step length. Urata et al. (2011) propose calculating footstep positions using singular preview control and online parameter optimization. The strategy requires fast stepping, which is enabled by high-speed actuation and high structural stiffness (Urata et al. 2010). Tajima et al. (2009) propose a controller for a running biped based on frequently replanning the CoM trajectories, as proposed by Nishiwaki and Kagami (2006). The system uses the initial velocity of a periodic gait pattern as final velocity for calculating the CoM trajectory of the current step. The resulting final CoM position for the current step and the initial distance of the CoM from the stance leg for the periodic gait are then used to determine the next stance leg position.

Pratt et al. (2006) coined the term *Capture Point* for positions where the robot can step in order to come to a complete stop. The set of all Capture Points is called the *Capture Region*. To facilitate the calculation of Capture Points, the robot is described by the LIPM, augmented by a fly wheel. Based on this concept, strategies for push-recovery and walking control are formulated. Nelson et al. (2012) propose positioning the foot relative to the Capture Point according both to the desired gait and the balancing requirements in the walking control of the Petman robot.

For the LIPM, the *Capture Point* is equivalent to the XcoM introduced by Hof et al. (2005, also see section 4.1). It is also equivalent to the *divergent component* introduced even earlier in a patent by Honda Motor Co. Ltd. (Takenaka 2004b). The essence of Honda's method is a diagonalization of a first-order representation of the LIPM ODE, which leads to two decoupled systems, of which one is stable (the *convergent component*) and one is unstable (the *divergent component*). The divergent component is then used to assure convergence of the CoM-trajectory to a desired periodic gait.[8] The concept has since been extended to running gaits with non-constant CoM-heights (Takenaka et al. 2009d).

Walking in Uneven Terrain

Walking in unknown and uneven terrain is especially challenging for biped robots and has been addressed both by control and by hardware design. Kang et al. (2010) developed a special foot mechanism for the robot Wabian-IIR with distance sensors that can mechanically adapt to uneven terrain. Similar mechanisms were previously developed at the same institute for biped walking chairs (Yamaguchi et al. 1995, 1994). Impedance or force control have been proposed for reducing the landing impact during walking (Buschmann et al. 2009; Nishiwaki and Kagami 2010; Lim et al. 2001). Other researchers have proposed implicit force control for reducing landing impacts, implemented by reducing joint position control gains during the expected impact phase (Tajima et al. 2009; Nishiwaki 2001; Hashimoto et al. 2009). Very recently, the robot ATLAS developed by Boston Dynamics for the DARPA Robotics Challenge[9] based on Petman, was shown to walk over loose rocks held in a wooden frame. To our knowledge, the only publication is a video[10] released on the internet and details on the controller design are unknown.

4.6 Biologically-Based Pattern Generation

Background

Most robot control systems are model-based and adopt the basic architecture of planning ideal reference trajectories and modifying them based on sensor feedback (cf. figure 4.5).

8. Note that coming to a complete stop is a special case of a periodic orbit.
9. `http://www.theroboticschallenge.org`
10. `http://youtu.be/SD60kylclb8`, accessed 2013/18/10.

In contrast to this, many biologically inspired concepts use Central Pattern Generator (CPG) models for creating walking trajectories. The goal of these concepts is to replicate the generation of rhythmic patterns for walking generation as observed in animals and humans in the walking machine control systems. As a CPG model, a neural oscillator proposed by Matsuoka (1985) is often used (e.g., Endo et al. 2008; Fukuoka and Kimura 2009). Matsuoka's CPG is a model of two mutually inhibiting neurons described by coupled differential equations. This model represents the half-center oscillator concept which is the most important element of neural networks for creating alternating left-right locomotion activity in vertebrates. One feature of CPGs is their ability to synchronize their intrinsic oscillations with input signals,[11] thus allowing adaptability to changing environments. This integration of internal and reflex loops is seen as one of the key features of biological walking control (see Buschmann et al. 2014).

However, CPG models often require costly parameter tuning to realize even basic walking in bipeds. Many optimization methods have been proposed for creating a set of suitable parameters, such as a gradient method with reinforcement learning in (Matsubara et al. 2006; Sugimoto and Morimoto 2011).

One important approach in CPG-based control is the concept of resetting the phase of a neural oscillator as soon as ground contact of the swing leg is detected. It has been shown in theory (e.g., Aoi and Tsuchiya 2006) and in experiments (e.g., Nakanishi et al. 2004) that this phase reset can improve the stability of a CPG-controlled walking machine. Fukuoka and Kimura (2003) propose a CPG-based control system for a quadruped robot in which the walking pattern for each leg is generated by defining desired joint positions depending on the current state of the leg. The criteria for state transition initiation depend on CPG model variables and position feedback.

Application to Biped Robots

Sugihara (2009) proposes a CPG based on a controlled LIPM. While the pendulum is inherently unstable when generating the pendulum position by integrating the equation of motion based on a given ZMP trajectory, a stable limit cycle can be generated by calculating the input ZMP using a nonlinear feedback law.

Geng et al. (2006) proposed a biologically inspired control method for the small biped robot RunBot with four active joints, based on previous work with the similar WalkBot robot. Walking is based solely on reflexive control implemented in a network of non-spiking neurons consisting of one extensor and one flexor neuron per joint (see Buschmann et al. 2014; Büschges et al. 2011). The motoneuron output is directly converted to an armature voltage for the driving DC motors. The mechanical design of WalkBot has typical features of passive walkers such as mechanical end-stops in the knees and round feet. Sensory feedback is provided by potentiometers in the joints and contact sensors in the feet. The authors have also used online learning based on policy searching to tune controller parameters (Geng et al. 2006).

11. This is known as *entrainment*. A famous example is the adaptation of the stepping rate of decerebrate cats walking on a motor-driven treadmill (Pearson 2008).

Applicability and Potential
For high-performance walking robots the dominant control methods are based on physical models and classical control theory. We can hypothesize that biologically-based pattern generation approaches have been less successful so far because they have mostly focused on low-level aspects, such as the structure of the neural oscillators instead of the performance of the system as a whole. However, it is known from neurobiology that the relevant aspect of the pattern generators is not necessarily the periodic output they are capable of generating, since this will always be modified by task-dependent intra-joint-, intra-leg- and inter-leg- sensor feedback and coupling influences of other CPGs as well as by higher brain areas. The more important aspect is the structure of possible feedback paths they define, for example, positive force feedback during stance or position feedback during swing. The biological background is reviewed in (Buschmann et al. 2014). An example of biologically inspired walking control within a model-based framework is given in (Buschmann et al. 2012a).

4.7 Gait Cycle-Centered Control

4.7.1 Limit Cycle Walkers

It is known from biomechanics that the physical properties of the musculosceletal system are important for human and animal walking. In fact, simple mechanical models of walking and running have been developed that show similar force patterns in simulations to those observed in experiments with humans. The simplest models are the inverted pendulum model and the SLIP model (Blickhan 1989). Interestingly, it has been observed that such very simple spring-mass models are stable without active control, a property often called *self stability* or *open-loop stability*.

This has motivated the design of a number of mechanisms and robots that emphasize the *natural dynamics* of walking over active control. At the extreme end are passive dynamic walkers with no active control pioneered by McGeer (1990), which go back to simple mechanical toys that can walk down an inclined slope by rocking from one foot to the other. The gait is fully determined by the kinematics and mass distribution of the mechanism and the incline of the slope. In a sense, the walking control is embedded into the mechanical design of these machines. The passively swinging legs are stabilized by mechanical stops during stance, the landing impact acts as a physical resetting mechanism and the inclined ground provides energy. While mostly explored in the context of bipedal robots, the underlying principles of passive dynamic walking have also been applied to multi-legged mechanisms (Sugimoto et al. 2011).

More recent work has focused on exploiting passive dynamics while adding some actuation and control (cf. figure 4.4). Collins and Ruina (2005) presented a mechanism incorporating actuators and springs connected in series to the ankle joints. This mechanism provides energy via a push-off motion after initial contact of the contralateral leg.

The actuator is controlled by a simple FSM. Sensing is provided by ground contact and foot extension switches.

In general, limit cycle walkers emphasize step-to-step control, not continuous stabilization or tracking control (Hobbelen et al. 2008). An example of a more recent limit cycle walker is the robot Flame. It combines some classical design elements seen in passive dynamic walkers, such as hyperextension stops in the knees, but still is capable of level ground walking thanks to the addition of joint actuation (Hobbelen et al. 2008). The robot uses lateral foot placement for stabilization. The heuristic linear control law is based on the CoM position and velocity in combination with foot contact switches for step-phase control (Hobbelen and Wisse 2009).

4.7.2 Step-Phase Control

In the majority of robot walking control systems there is no real step-phase control in the sense of a feedback control system, since timing and duration of stance and swing phases are predetermined by a planning module. Note that this is not the case for CPG-based controllers, which often include step-phase control via a phase-resetting mechanism for the neural oscillators (see section 4.6). This section focuses on step-phase control for non-CPG-based systems.

Event-based phase switching mechanisms that incorporate sensory feedback into step-phase control have been developed for a variety of walking systems such as a hopping robot (Sato 2007), quadruped robots (Fukuoka and Kimura 2003; Hawker and Buehler 2000), planar biped robots (Pratt and Pratt 1998a, 1998b; Pratt et al. 2001; Sreenath et al. 2011) and 3D biped robots (Furusho and Sano 1990; Pratt and Pratt 1999; Morisawa et al. 2011; Buschmann et al. 2012a). Sato (2007) presents a switching controller for a hopping robot where the same controller is used during the flight and the ground contact phases. Phase change events cause a switching of desired variables and gains.

The control framework for biped robots developed by Pratt et al. incorporates a state machine for each leg. The rules for state transitions are formulated intuitively based on the position of the body relative to the feet and ground contact force information (Pratt and Pratt 1998a, 1999, 1998b; Pratt et al. 2001). In simulation, the contact forces acting on the feet are used to realize a phase reset mechanism for the transition to the double support phase. The transition to the single support phase is initiated once the ground contact force on the designated swing leg falls below a given threshold (Pratt and Pratt 1998b, 1999). In experiments with a planar robot, these contact force conditions are replaced with geometric conditions (Pratt and Pratt 1998a). The phase reset mechanism is also used by Furusho and Sano (1990), who developed a walking control system for a 3D robot and realized stable walking at a speed of 0.18 m/s. The condition for the transition to the single support phase is that the CoM passes above the toe. A phase reset mechanism is also used in the walking control of the planar robot

MABLE (Sreenath et al. 2011) and in a simulation of the robot HRP-2 (Morisawa et al. 2011).

Currently, many realizations of event-based walking control systems incorporate intuitive rules for phase transition, especially for the transition to the single support phase (Furusho and Sano 1990; Pratt and Pratt 1998a, 1998b; Pratt et al. 2001; Sreenath et al. 2011). Most of the event-based control strategies mentioned above are tested in simulation (Pratt et al. 2001; Pratt and Pratt 1998a, 1999) or experiments with planar robots with few degrees-of-freedom (Pratt and Pratt 1998a, 1998b; Sreenath et al. 2011) and not in experiments with 3D robots. The author has proposed step-phase control laws for biped walking based on stability considerations, which has enabled the robot Lola to walk over unexpected and unmodelled obstacles (Buschmann et al. 2012a).

4.8 Chapter Summary and Discussion

This chapter reviews common approaches to hardware and control design for biped robots, as well as current knowledge of human balance and postural control. A large number of different approaches have been taken in this field and no review can do full justice to all of them. Nevertheless, some distinct approaches can be distinguished. Currently, robots with stiff structures and hierarchical controllers with an inner high-gain control loop and outer loops handling contact force/impedance control and balance clearly show the best performance.

Theoretically, MPC methods should be the tool of choice for walking control since they provide a systematic approach towards optimally exploiting the known dynamics in feedback controllers for complex systems. With increasing computational power, these approaches will become more widespread. At the same time, the direct implementation of walking control using real-time MPC for realistic robot models is still many years away due to the prohibitive computational cost. More importantly, no control method can tell us how to choose the inputs, the outputs and the control objectives.

This is where neurobiological findings can help us identify which physical quantities are controlled during a certain task in animals (positions, velocities, forces, etc.) and which coordinates are chosen for the control (joint velocities, foot velocities, etc.), which can help to identify promising control approaches for walking machines.

While state-of-the-art bipedal robots are capable of quite fast and robust locomotion, the gap between the capabilities of humans and machines still is considerable. Much research is still at the level of generating and stabilizing a basic walking pattern and successful approaches use strong simplifications and impose severe restrictions on possible motions.

Promising approaches towards improving the capabilities of bipedal robots are therefore to be found in basic research on improved methods for generating, modeling and controlling bipedal walking. One approach currently being pursued is to move

from time-based trajectories that are modified by feedback for stabilization to a fundamentally feedback-based coordination of the gait cycle, as it is found in animals and humans (see Buschmann et al. 2012a, 2014). The second promising line of research is to improve the capability of generating motions in real-time that efficiently make use of all DoFs. This second line of research is addressed in the following chapter.

5 Optimal Redundancy Resolution and Collision Avoidance

Algorithms for generating and controlling the motion of complex robots must take many constraints into account while trying to satisfy multiple goals. This chapter reviews some classical methods for robot motion generation that are suitable for real-time use and presents methods and application examples for humanoid robots and redundant agricultural manipulators. Classical methods for choosing locally optimal motions are reviewed in section 5.1. An efficient approach to modeling complex robot geometries for the purpose of collision detection and avoidance is presented in section 5.2. Application examples of local methods for nullspace optimization and collision avoidance are described in section 5.3, an approach to task-space collision avoidance for humanoid robots in section 5.4. An extension of the classical local methods to hybrid task descriptions composed of force and position variables is described in section 5.5. A predictive approach to redundancy resolution with an application to redundant manipulators is presented in section 5.6. Section 5.7 concludes the chapter.

5.1 Local Inverse Kinematics

Often motion generation can be simplified by not planning a trajectory of generalized coordinates $\boldsymbol{q}(t) \in \mathbb{R}^n$ (typically joint angles), but rather a set of suitably defined task-space trajectories $\boldsymbol{w}(t) \in \mathbb{R}^m$. For robot manipulators, $\boldsymbol{w}$ typically consists of the position and orientation of the end-effector, but can be an arbitrary function $\boldsymbol{w}(\boldsymbol{q})$ of the generalized coordinates. In the case of bipedal robots, $\boldsymbol{w}$ might contain the CoM as well as the foot positions and orientations.[1] The following review of local IK methods is in part based on (Buschmann et al. 2012c, section 1).

One approach to controlling a robot along a task-space trajectory $\boldsymbol{w}_\mathrm{d}$ is to track an equivalent joint-space trajectory $\boldsymbol{q}_\mathrm{d}$ satisfying

$$\boldsymbol{w}(\boldsymbol{q}_\mathrm{d}) = \boldsymbol{w}_\mathrm{d}. \tag{5.1}$$

Here (and in what follows) the index d indicates a desired quantity. Solving the IK problem (5.1) is more complex than direct kinematics calculations (see section 2.3), since there generally is no function mapping $\boldsymbol{w}$ to $\boldsymbol{q}$, even in the non-redundant

1. See (Buschmann et al. 2009; Buschmann 2010) for examples of task-space definitions for bipedal robots.

case where $\dim(\boldsymbol{w}) = \dim(\boldsymbol{q})$ (or: $m = n$). In general, there are four characteristic configurations (see figure 5.1): (1) there is exactly one solution, (2) there are multiple solutions, (3) there is no solution and (4) there are an infinite number of solutions.

The IK problem (5.1) can be solved using several approaches. In simple cases geometrical reasoning or analytical manipulations might lead to a solution. Much effort has been put into the analysis of serial manipulators with four, five and six DoFs. In special cases, such as when three consecutive axes of rotation of a six-axis manipulator intersect, or when three consecutive rotational joints are parallel, there are closed-form solutions (see Pieper 1968; for a recent review see Siciliano and Khatib 2008, chapter 2). Tsai and Morgan (1985) reduced the IK problem for general six-axis manipulators to the solution of a system of eight second order equations in eight unknowns. In special cases the solutions of this system can again be computed in closed form. In the general case all solutions can be calculated using continuation methods (a numerical technique, see Tsai and Morgan 1985).

The advantage of analytical approaches is the ability to obtain all solutions. They are, however, restricted to special cases. General choices of $\boldsymbol{w}$ and arbitrary branched systems can only be handled numerically. The direct solution of (5.1) (position-level IK) is described in the next section, more widespread methods at the velocity and acceleration levels are discussed in the following ones.

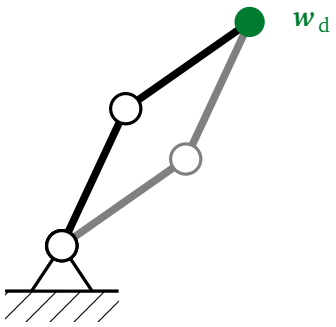
5.1.1 Position-Level Inverse Kinematics

The simplest numerical IK method consists of iteratively solving the nonlinear equation $\boldsymbol{w}_{\mathrm{d}} = \boldsymbol{w}(\boldsymbol{q})$ starting from an initial guess $\boldsymbol{q}_0$. In the non-redundant case the classical Newton-Raphson method ($\dim(\boldsymbol{q}) = \dim(\boldsymbol{w})$) is well suited. In this approach, a linear approximation of (5.1)

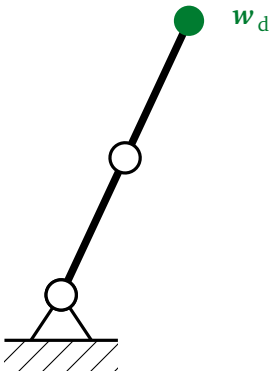
$$\boldsymbol{w}(\boldsymbol{q}_0 + \Delta\boldsymbol{q}) \approx \boldsymbol{w}(\boldsymbol{q}_0) + \tfrac{\partial \boldsymbol{w}}{\partial \boldsymbol{q}}|_{\boldsymbol{q}_0} \Delta\boldsymbol{q}, \tag{5.2}$$

is repeatedly solved for $\Delta\boldsymbol{q}$, until some error bound is deemed small enough. With the Jacobian $\boldsymbol{J}_w = \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{q}}$, applying the Newton-Raphson method yields the basic algorithm 9 (see Pieper 1968, for an early use of this method). This approach inherits characteristic properties of Newton's method: (1) the solution depends on the initial guess, (2) only one solution is generated, (3) rapid convergence for good initial guess, (4) rapid divergence for bad initial guess and (5) no solution for singular $\boldsymbol{J}_w$.

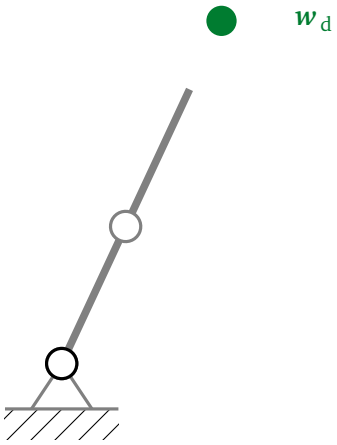
Better convergence properties are achievable by standard globalization strategies such as damping methods or steepest descent methods (for details see, for example, Deuflhard (2011, chapter 3)). A straightforward generalization for the redundant case is to use the pseudoinverse instead of the inverse of the Jacobian, which is closely related to velocity-level approaches for redundant systems (cf. section 5.1.2). The IK problem can also be formulated as a nonlinear optimization problem (for a good overview see Bonnans et al. 1997). The advantage is the straightforward inclusion



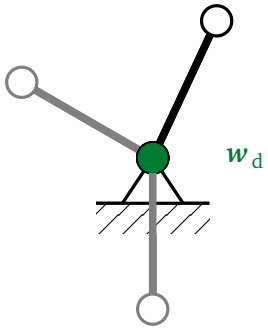
Case 1: Two solutions. w_d in the interior of $\mathcal{W}$.



Case 2: Exactly one solution. w_d on the boundary of $\mathcal{W}$.



Case 3: No solution. w_d outside of $\mathcal{W}$.



Case 4: An infinite number of solutions (the second body exactly covers the first). w_d in the interior of $\mathcal{W}$.

Figure 5.1: Illustration of some configurations encountered during in the solution of IK problems. The desired end-effector position $\boldsymbol{w}_d^T = (x \quad y)$ is given (green dot) the joint angles $\boldsymbol{q}^T = (q_1 \quad q_2)$ are unknown. Depending on the position of $\boldsymbol{w}_d$ in the domain $\mathcal{W}$ of reachable task-space positions $\boldsymbol{w}$ and the parameters of the system there may be no solutions, one solution, multiple solutions or an infinite number of solutions. Here $\mathcal{W}$ the disc centered on the first joint with a radius given by the stretched length of both bodies.

Algorithm 9: Position-level IK based on the Newton-Raphson method.

input : Initial guess $\boldsymbol{q}_0$, desired output $\boldsymbol{w}_\mathrm{d}$, error bound ε
output: Approximate solution $\boldsymbol{q}$ to $\boldsymbol{w}_\mathrm{d} = \boldsymbol{w}(\boldsymbol{q})$

repeat
  $\Delta\boldsymbol{w} \leftarrow \boldsymbol{w}_\mathrm{d} - \boldsymbol{w}(\boldsymbol{q}_k)$
  Solve $\boldsymbol{J}_w \Delta\boldsymbol{q} = \Delta\boldsymbol{w}$
  $\boldsymbol{q}_k \leftarrow \boldsymbol{q}_{k-1} + \Delta\boldsymbol{q}$
  $k \leftarrow k+1$
until $\|\Delta\boldsymbol{w}\| < \varepsilon$

of additional objectives and constraints, the disadvantage the larger numerical cost. An early example is the method published by Goldenberg et al. (1985) based on coordinate partitioning and numerical optimization. More recent optimization-based work including a comparison of different descent algorithms[2] is reported by Ayusawa and Nakamura (2012).

5.1.2 Velocity-Level Inverse Kinematics

The direct kinematics function $\boldsymbol{w}(\boldsymbol{q})$ is nonlinear, such that (5.1) has no closed form solution in the general case. A standard method of obtaining an approximate solution is to differentiate (5.1) once with respect to time, which leads to an implicit first-order ODE in the unknown $\boldsymbol{q}$:

$$\dot{\boldsymbol{w}} = \boldsymbol{J}_w \dot{\boldsymbol{q}} = \dot{\boldsymbol{w}}_\mathrm{d}. \tag{5.3}$$

For non-redundant robots ($\dim(\boldsymbol{q}) = \dim(\boldsymbol{w})$) in non-singular configurations, we can directly solve for $\dot{\boldsymbol{q}}$ to obtain an explicit ODE:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_w^{-1} \dot{\boldsymbol{w}}_\mathrm{d}. \tag{5.4}$$

If a solution $\boldsymbol{w}(\boldsymbol{q}(t_0)) = \boldsymbol{w}_\mathrm{d}(t_0)$ is known at a time t_0, calculating $\boldsymbol{q}(t)$ from a given $\boldsymbol{w}_\mathrm{d}(t)$ amounts to solving an IVP for (5.4). Similar to integrating differentiated kinematic constraints in a mechanical system, directly integrating (5.4) leads to unbounded drift (cf. section 3.4.1). As for constrained mechanical systems, drift can be limited by a Baumgarte-like control term (Baumgarte 1972), replacing $\boldsymbol{w}_\mathrm{d}$ in (5.3), (5.4) by

$$\dot{\boldsymbol{w}}_{\mathrm{d,eff}} = \dot{\boldsymbol{w}}_\mathrm{d} + \boldsymbol{K}(\boldsymbol{w}_\mathrm{d} - \boldsymbol{w}(\boldsymbol{q})). \tag{5.5}$$

This Closed Loop Inverse Kinematics (CLIK) algorithm is illustrated in figure 5.2.

Using the measured instead of the current desired generalized coordinates $\boldsymbol{q}$ in (5.5)

2. Steepest descent, conjugate gradient Levenberg-Marquardt and Quasi-Newton

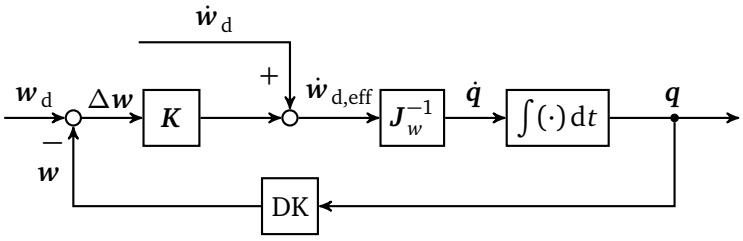


Figure 5.2: Velocity-level inverse kinematics using the inverse Jacobian and drift compensation (CLIK, closed loop inverse kinematics).

leads to a task-space feedback control law, where the actual task-space tracking error is controlled instead of the numerical drift. In this scheme, the calculated reference velocity $\dot{\boldsymbol{q}}$ can also be directly input to a joint velocity controller.

For position controlled robots, $\boldsymbol{q}(t)$ is calculated by numerically integrating $\dot{\boldsymbol{q}}(t)$, usually using a simple forward Euler scheme. A significant difference between standard IVPs and the differential IK problem is the fact that the integration error $\Delta\boldsymbol{w} = \boldsymbol{w}_d - \boldsymbol{w}(\boldsymbol{q})$ can be computed directly. This has led to the development of problem-specific step-size control methods (e.g., Gupta and Kazerounian 1985). An investigation of different integration and error control algorithms that achieve better accuracy than Euler's algorithm at the same cost may be found in (Featherstone 1994). For large step-sizes or very high accuracy requirements, advanced algorithms should be used, but the basic algorithms are sufficient for most real-time applications. For a stability proof of CLIK algorithms in the discrete domain, see (Falco and Natale 2011).

For redundant systems with $\dim(\boldsymbol{q}) > \dim(\boldsymbol{w})$, one of the infinite number of possible solutions to (5.3) may be chosen by adding a cost function to be minimized at every time step. The use of a quadratic penalty on joint velocities was proposed by Whitney (1969) as Resolved Motion Rate Control (RMC):

$$\begin{aligned} \Phi &= \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} && \rightarrow \min! \\ \boldsymbol{g} &= \dot{\boldsymbol{w}}_{d,eff} - \boldsymbol{J}_w \dot{\boldsymbol{q}} && = \boldsymbol{0}. \end{aligned} \tag{5.6}$$

The positive definite, and usually diagonal, matrix $\boldsymbol{W}$ defines the norm for penalizing $\dot{\boldsymbol{q}}$ and usually reflect the motion range and/or maximum speeds of different joints. An extension to more general cost functions was proposed by Liégeois (1977). This *Automatic Supervisory Control* method minimizes the weighted sum of the original cost $\frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}}$ and the change ΔH of an additional cost function $H(\boldsymbol{q})$ during the next time step Δt:

$$\begin{aligned} \Phi &= \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} + \alpha \tfrac{\partial H}{\partial q} \dot{\boldsymbol{q}} \rightarrow \min! \\ \boldsymbol{g} &= \dot{\boldsymbol{w}}_{d,eff} - \boldsymbol{J}_w \dot{\boldsymbol{q}} = \boldsymbol{0} \end{aligned} \tag{5.7}$$

The first equation uses the linear approximation $\Delta H = \frac{\partial H}{\partial q}\dot{\boldsymbol{q}}\Delta t$ which is added to Φ with an appropriately chosen weighting factor α.

A straightforward solution to this Quadratic Program (QP) is obtained with the method of Lagrange multipliers. The Lagrangian

$$L = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W}\dot{\boldsymbol{q}} + \alpha\frac{\partial H}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{\lambda}^T(\dot{\boldsymbol{w}}_{\text{d,eff}} - \boldsymbol{J}_w\dot{\boldsymbol{q}}) \tag{5.8}$$

yields the optimality conditions

$$\frac{\partial L}{\partial \dot{\boldsymbol{q}}}^T = \boldsymbol{0} = \boldsymbol{W}\dot{\boldsymbol{q}} + \alpha\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T - \boldsymbol{J}_w^T\boldsymbol{\lambda}, \tag{5.9}$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}}^T = \boldsymbol{0} = \dot{\boldsymbol{w}}_{\text{d,eff}} - \boldsymbol{J}_w\dot{\boldsymbol{q}}. \tag{5.10}$$

Since $\boldsymbol{W}$ is positive definite, (5.9) can always be solved for $\dot{\boldsymbol{q}}$:

$$\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\boldsymbol{J}_w^T\boldsymbol{\lambda} + \boldsymbol{W}^{-1}\alpha\frac{\partial H}{\partial q}^T, \tag{5.11}$$

$$\overset{(5.10)}{\Rightarrow} \dot{\boldsymbol{w}}_{\text{d,eff}} = \boldsymbol{J}_w\boldsymbol{W}^{-1}\boldsymbol{J}_w^T\boldsymbol{\lambda} + \boldsymbol{J}_w\boldsymbol{W}^{-1}\alpha\frac{\partial H}{\partial q}^T, \tag{5.12}$$

$$\Rightarrow \boldsymbol{\lambda} = \left(\boldsymbol{J}_w\boldsymbol{W}^{-1}\boldsymbol{J}_w^T\right)^{-1}\dot{\boldsymbol{w}}_{\text{d,eff}} - \left(\boldsymbol{J}_w\boldsymbol{W}^{-1}\boldsymbol{J}_w^T\right)^{-1}\boldsymbol{J}_w\boldsymbol{W}^{-1}\alpha\frac{\partial H}{\partial q}^T. \tag{5.13}$$

Substituting $\boldsymbol{\lambda}$ into (5.11) yields the closed-form solution:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{w,W}^{\#}\dot{\boldsymbol{w}}_{\text{d,eff}} - \alpha\boldsymbol{N}_{w,W}\boldsymbol{W}^{-1}\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T, \tag{5.14}$$

$$\boldsymbol{J}_{w,W}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{J}_w^T\left(\boldsymbol{J}_w\boldsymbol{W}^{-1}\boldsymbol{J}_w^T\right)^{-1}, \tag{5.15}$$

$$\boldsymbol{N}_{w,W} = (\boldsymbol{I} - \boldsymbol{J}_{w,W}^{\#}\boldsymbol{J}_w). \tag{5.16}$$

Here, $\boldsymbol{J}_{w,W}^{\#}$ is the ($\boldsymbol{W}$-) weighted (Moore-Penrose) pseudoinverse and $\boldsymbol{N}_{w,W}$ the corresponding *nullspace projection matrix*, which maps any vector $\boldsymbol{x} \in \mathbb{R}^{n_q}$ into $\boldsymbol{J}_w$'s nullspace, that is:

$$\boldsymbol{J}_w(\boldsymbol{N}_{w,W}\boldsymbol{x}) = \boldsymbol{J}_w\boldsymbol{x} - \underbrace{\boldsymbol{J}_w\boldsymbol{J}_{w,W}^{\#}}_{\boldsymbol{I}}\boldsymbol{J}_w\boldsymbol{x} = \boldsymbol{0} \quad \forall \quad \boldsymbol{x} \in \mathbb{R}^{n_q}. \tag{5.17}$$

A numerically more efficient method is the direct solution of the original optimization problem, without explicit calculation of the pseudoinverse,[3] see algorithm 10.

3. This is due to Klein and Huang (1983).

Algorithm 10: Velocity-level IK for redundant robots without explicit calculation of the pseudoinverse.

input : Task $\boldsymbol{w}_{\mathrm{d}}$, nullspace vector $\boldsymbol{z}$
$\boldsymbol{B} \leftarrow \boldsymbol{J}_w \boldsymbol{W}^{-1} \boldsymbol{J}_w^T$
$\boldsymbol{p} \leftarrow \dot{\boldsymbol{w}}_{\mathrm{d}} - \boldsymbol{J}_W \boldsymbol{z}$
Solve $\boldsymbol{B}\boldsymbol{\lambda} = \boldsymbol{p}$ for $\boldsymbol{\lambda}$
$\dot{\boldsymbol{q}} \leftarrow \boldsymbol{W}^{-1}\left(\boldsymbol{J}_W^T \boldsymbol{\lambda} - \boldsymbol{z}\right)$

5.1.3 Acceleration-Level Inverse Kinematics

Differentiating the velocity-level forward kinematics function (5.3) once more yields

$$\boldsymbol{J}_w \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_w \dot{\boldsymbol{q}} = \ddot{\boldsymbol{w}}_{\mathrm{d}}, \tag{5.18}$$

which can be solved for the generalized accelerations $\ddot{\boldsymbol{q}}$ and integrated to obtain $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$. Luh et al. (1980b) proposed a *Resolved Acceleration Rate Control* law for a six-axis manipulator which is similar to (5.5), but defined at the acceleration level:

$$\ddot{\boldsymbol{w}}_{\mathrm{d,eff}} = \ddot{\boldsymbol{w}}_d + \boldsymbol{K}_D(\dot{\boldsymbol{w}}_d - \dot{\boldsymbol{w}}(\boldsymbol{q})) + \boldsymbol{K}_P(\boldsymbol{w}_d - \boldsymbol{w}(\boldsymbol{q})). \tag{5.19}$$

Luh et al. (1980b) closes the feedback loop via an inverse model control, that is, the required input forces are computed from an inverse model of the robot and then applied to it. An alternative for position-controlled robots is to integrate $\ddot{\boldsymbol{q}}$ and send $\boldsymbol{q}$ to a joint position controller.

Hollerbach and Suh (1985) proposed resolving kinematic redundancy by locally minimizing joint torques. Since both the equations of motion and (5.18) are linear in $\ddot{\boldsymbol{q}}$, this corresponds to solving a QP. The linearity in $\ddot{\boldsymbol{q}}$ is also exploited in the operational space control framework proposed by Khatib (1983). This approach has been generalized to hybrid position/force control based on the operational space formulation of the EoM[4] (Khatib 1987).

A generalized algorithm with collision avoidance and joint limits is applied to a humanoid by Dariush et al. (2010). The methods from first order redundancy resolution (see (5.1.2)) are modified in this work for second order kinematics.

5.1.4 A Note on Singularities

IK methods that rely on solving a linear equation involving $\boldsymbol{J}_w$ fail when the Jacobian becomes singular. Geometrically, the rank deficiency of $\boldsymbol{J}_w$ corresponds to a loss of

4. The "operational space formulation" uses an EoM in task space $\boldsymbol{M}_w \ddot{\boldsymbol{w}} + \boldsymbol{h}_w = \boldsymbol{u}_w$, calculated from the original EoM and the acceleration-level differential kinematics (5.18).

DoFs. task-space velocities are related to the Jacobian via

$$\dot{\boldsymbol{w}} = \frac{\partial \boldsymbol{w}(\boldsymbol{q})}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} = \boldsymbol{J}_w \dot{\boldsymbol{q}}, \tag{5.20}$$

that is, $\dot{\boldsymbol{w}}$ is a linear combination of $\boldsymbol{J}_w$'s columns. Rank deficiency therefore corresponds to a reduction of the dimension of the space of achievable task-space velocities.[5] Singular configurations can appear at the outer edge of the reachable workspace (e.g., for fully stretched manipulators) or inside the workspace (often when multiple axes become parallel).

A classical approach to handle singular (or nearly singular) configurations at the velocity-level is due to Nakamura and Hanafusa (1984, 1986). It is based on a least squares approach to solving (5.3), where an additional term penalizing velocities is added:

$$\Phi = \frac{1}{2}(\dot{\boldsymbol{w}}_{\mathrm{d,eff}} - \boldsymbol{J}_w \dot{\boldsymbol{q}})^2 + k\dot{\boldsymbol{q}}^2 \rightarrow \min!, \tag{5.21}$$

$$\Rightarrow \dot{\boldsymbol{q}} = \left(\boldsymbol{J}_w^T \boldsymbol{J}_w + k\boldsymbol{I}\right)^{-1} \boldsymbol{J}_w^T \dot{\boldsymbol{w}}_{\mathrm{d,eff}}. \tag{5.22}$$

For a weighting factor $k > 0$ the inverse exists. The main drawback is an inexact solution outside of singularities. This, however, may be counteracted by an adaptive, configuration-dependent choice of k.

A further classical method is related to minimizing the task-space tracking error $\boldsymbol{e}_w = \boldsymbol{w}_{\mathrm{d}} - \boldsymbol{w}(\boldsymbol{q})$. The steepest descent method for minimizing $\frac{1}{2}\boldsymbol{e}_w^2$ gives the update rule:

$$\boldsymbol{q}^{k+1} = \boldsymbol{q}^k - \frac{\partial \boldsymbol{e}_w}{\partial \boldsymbol{q}}^T = \boldsymbol{q}^k + \boldsymbol{J}_w^T(\boldsymbol{w}_{\mathrm{d}} - \boldsymbol{w}(\boldsymbol{q})). \tag{5.23}$$

By analogy, the corresponding *transposed Jacobian* CLIK algorithm is defined by:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_w^T \dot{\boldsymbol{w}}_{\mathrm{d,eff}}. \tag{5.24}$$

This IK scheme does not suffer from singularities and is very cheap computationally, but also significantly less accurate since task space velocities are only tracked due to the feedback law: in almost all cases

$$\dot{\boldsymbol{w}} = \boldsymbol{J}_w \boldsymbol{J}_w^T \dot{\boldsymbol{w}}_{\mathrm{d,eff}} \neq \dot{\boldsymbol{w}}_{\mathrm{d}}, \tag{5.25}$$

since $\boldsymbol{J}_w \boldsymbol{J}_w^T \neq \boldsymbol{I}$. The method is obviously stable in the sense of Lyapunov in the static case (e.g., Lyapunov 1992; Khalil 2000).[6] For $\dot{\boldsymbol{w}}_{\mathrm{d}} \neq \boldsymbol{0}$ stability is not guaranteed, but

5. The same argument applies to Newton's method at the position-level or acceleration-level approaches, when "task-space velocity" is replaced by "task-space position increment" or "task-space acceleration" and $\dot{\boldsymbol{q}}$ is replaced by $\Delta\boldsymbol{q}$ (during the Newton iteration) or $\ddot{\boldsymbol{q}}$.

6. This is easily proven using the Lyapunov function candidate $V(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{e}_w^T \boldsymbol{K} \boldsymbol{e}_w$, where $\boldsymbol{K}$ is the

maintained in most applications.

5.1.5 Hierarchical Methods and Inequality Constraints

The idea proposed by Liégeois (1977) of actively exploiting kinematic redundancy has been applied to tracking multiple tasks with a given priority (Nakamura et al. 1987). Effectively, the tracking error of a less important task is tracked in the nullspace of a more important task by appropriate application of pseudoinverses. The method has been extended to an arbitrary number of priority levels (Baerlocher and Boulic 1998; Siciliano 1990). More recently, extensions to incorporate inequality constraints have been proposed (F. Cheng et al. 1994; Kanoun 2011; Escande et al. 2010). When lower-priority tasks are not feasible within the nullspace of high-priority tasks, the projected matrices become rank-deficient, a phenomenon known as an *algorithmic singularity*. A possible solution is to use truncated singular value decomposition to calculate inverses, or to use smooth regularization methods similar to the damped pseudoinverse (cf. (5.22); Chiaverini 1997; Kanoun 2011). Inequality constraints typically arise due to physical limits of the robot such as joint ranges, maximum joint speeds and accelerations or collision avoidance (e.g., Stasse et al. 2008). A number of groups have worked on this approach extensively, adding more capabilities and application examples and developing software frameworks. Notable examples are the "iTaSC" software (see Decre et al. 2013; De Schutter et al. 2007; Decre et al. 2009) and the "Stack of Tasks" (see Mansard et al. 2009b, 2009a).

5.2 Efficient Collision Geometries and Distance Calculations

Since robots usually have a large workspace and wide range of motion, they can easily collide with the environment or with themselves. This problem is especially acute for branched systems with many DoFs such as humanoid robots. For such systems, most choices of generalized coordinates are unfeasible, since they lead to self-collision or violate joint limits. This section reviews some approaches to collision detection and collision avoidance, with a focus on humanoid robots. It also presents an efficient system for real-time detection and avoidance of potential collisions, which has been successfully applied to an agricultural manipulator and a humanoid robot. This section is based on joint work with Markus Schwienbacher (Schwienbacher et al. 2011).

closed-loop feedback matrix. Alternatively, this can be viewed as a property that is inherited from the stability of the steepest descent method for convex objective functions.

5.2.1 Background and Related Work

Distance calculations are not only essential for real-time control of complex robots, but also central to other fields such as computer graphics, CAD or dynamics simulation. Much research has therefore gone into developing efficient algorithms for this task (for review see, e.g., Lin and Gottschalk 1998; Jiménez et al. 2001). In motion planning and computer graphics polygonal models are dominant, since they enable the approximation of arbitrarily complex geometries and hardware-accelerated rendering is enabled by Graphics Processing Units (GPUs). The major drawback is the high computational cost of distance queries between complex objects. Also, the edges of the polyhedra lead to kinks in the distance function between objects, with is not ideal for optimization-based methods and real-time control (see below).

The importance of proper collision avoidance for complex robots has motivated several research efforts on (real-time) self-collision avoidance for humanoid robots (e.g., Kuffner et al. 2002; Okada et al. 2005; Sugiura et al. 2007; Toussaint et al. 2007; Gienger et al. 2008a; Stasse et al. 2008; Kanehiro et al. 2009). To enable real-time control, the robot geometry is always simplified and the number of possible collision pairs is often (manually) pruned.

To ensure real-time performance, early work was more focused on collision detection than avoidance. Kuffner et al. (2002) developed a collision detection system for the humanoid H7 (Nishiwaki et al. 2002) which uses the V-clip algorithm (Mirtich 1998) to compute the distance between conservative convex polyhedral approximations of the robot's segments. If a potential collision is detected in the planned motion, a pre-determined safe stopping trajectory is executed.

Okada et al. (2005) experimentally evaluated the performance of several libraries in a real-time self-collision detection program for the robot HRP-2. The authors used as many collision pairs as possible with detailed geometric models of the robot. The authors suggest the use of Axis-Aligned Bounding Box (ABB)-based methods which are faster than conventional Oriented Bounding Box (OBB)-based methods. The method was tested on HRP2 in an experiment where the commanded arm motion would lead to a self-collision.

The work on upper-body control of the robot Asimo at the Honda Research Institute Europe is also notable (Sugiura et al. 2007; Toussaint et al. 2007; Gienger et al. 2008a). The authors use simplified robot geometry models composed of Swept Sphere Volumes (SSVs) (e.g., Larsen et al. 1999), enabling fast distance calculations between robot links. The collision model is used with gradient-based approaches for online collision avoidance during reaching and grasping. The same collision avoidance scheme is used for off-line optimization of a sequence of attractors to generate collision free trajectories of the arms for reaching objects (Toussaint et al. 2007; Gienger et al. 2008a).

Many approaches involving collision avoidance rely on off-line computations, due to the computational complexity of performing many distance queries over the full duration of a trajectory. An example of such an approach is the work on whole-body

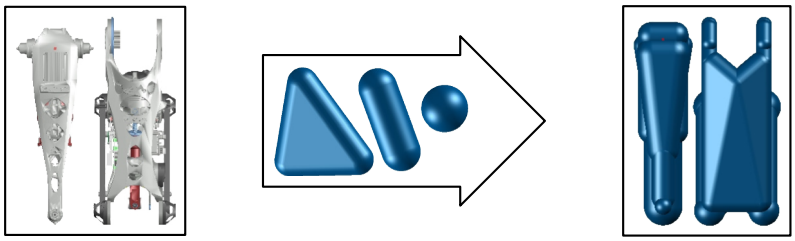


Figure 5.3: The complex CAD-geometry is manually approximated by SSVs. The figure shows the lower and upper leg segments on the left and the corresponding collision geometries on the right (figure redrawn from Schwienbacher et al. 2011).

motion generation for humanoids presented by Kanehiro et al. (2009).

Many collision avoidance schemes use the gradients of distance functions to incrementally modify the robot's motion, either in velocity-level CLIK schemes, or during trajectory optimization.[7] This makes C^1-continuity of the distance function desirable, since discontinuous gradients pose problems for standard optimization algorithms and lead to non-smooth velocity references, which is not ideal for the widely used high-gain joint controllers. Escande et al. proposed using strictly convex bounding volumes constructed of patches of spheres and tori (Sphere Torus Patch Bounding Volumes, STP-BV), which ensures C^1 continuity of the distance function with other *convex objects* (Escande et al. 2007, 2014, 2010). It must be noted, however, that smooth gradients are only possible between strictly convex shapes. Approximating every shape by a strictly convex hull can, in some cases, introduce unacceptable modeling errors.

A further approach based on "Sphere Swept Convex Hulls" (SSCH) was developed by Taubig et al. (2011): a polygonal convex bounding volume is enlarged by a buffer radius and used for collision avoidance in a torque-level control system.

5.2.2 Geometry Models Based on Swept Sphere Volumes

In the following, a collision avoidance and detection scheme based on SSVs is presented.[8] The system includes three shapes, which are defined by the volume generated when a sphere is moved along a *generating geometric primitive*: (1) the Point Swept Sphere (PSS), (2) the Line Swept Sphere (LSS) and (3) the Triangle Swept Sphere (TSS). CAD geometries are manually approximated using a graphical user interface. An example is given in figure 5.3, for details see (Schwienbacher 2013). The TSS

7. This is in contrast to classical sampling-based *path planners*, that only need to know if a pose is collision free, not which modification would improve the motion (for an overview of path planing algorithms see, e.g., LaValle 2006)

8. A prototypical implementation of the library this work is based was developed by Träger (2010). The current system was developed and optimized by Markus Schwienbacher (Schwienbacher et al. 2011; Schwienbacher 2013).

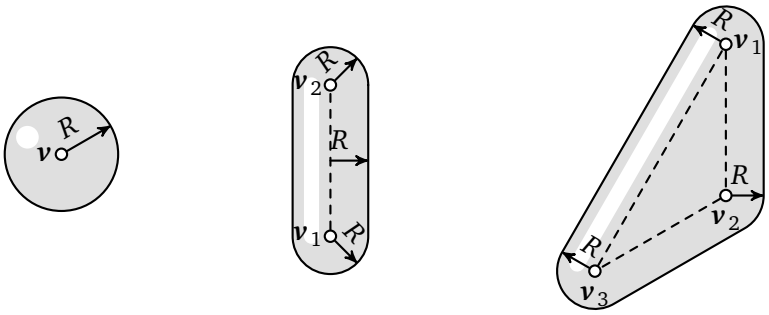


Figure 5.4: Basic SSVs Point-Swept-Sphere (PSS), Line-Swept-Sphere (LSS) and Triangle-Swept-Sphere (TSS). The generating geometric primitives point, line and triangle are defined by the vertices $\boldsymbol{v}_i$, the SSV additionally by the radius R.

developed for this system enables more accurate modeling of general 3D shapes than the rectangle-SSV proposed by Larsen et al. (2000).

With reference to figure 5.4, the *generating primitives* are defined by the vertices $\boldsymbol{v}_i$ and any point $\boldsymbol{r} \in \mathbb{R}^3$ on them is given by:

$$\text{PSS} \quad \boldsymbol{r} = \boldsymbol{v}, \tag{5.26}$$

$$\begin{aligned} \text{LSS} \quad & \boldsymbol{r} = \boldsymbol{v}_1 + (\boldsymbol{v}_2 - \boldsymbol{v}_1)s,\ s \in \Omega_L, \\ & \Omega_L = \{s \in \mathbb{R} | \, 0 \le s \le 1\}, \end{aligned} \tag{5.27}$$

$$\begin{aligned} \text{TSS} \quad & \boldsymbol{r} = \boldsymbol{v}_1 + \begin{pmatrix} (\boldsymbol{v}_2 - \boldsymbol{v}_1) & (\boldsymbol{v}_3 - \boldsymbol{v}_1) \end{pmatrix} \boldsymbol{s},\ \boldsymbol{s} \in \Omega_T, \\ & \Omega_T = \{\boldsymbol{s} \in \mathbb{R}^2 | \, (0 \le s_1 \le 1) \wedge (0 \le s_2 \le 1) \wedge (s_1 + s_2 \le 1)\}. \end{aligned} \tag{5.28}$$

Geometrically relevant parts of the robot $\mathscr{R}$ are modeled by n_S segments $\mathscr{S}_i$, each consisting of n_{SVi} of SSV volumes $\mathscr{V}_{ik}$. Similarly, the environment $\mathscr{E}$ is modeled by n_O segments (i.e., obstacles) $\mathscr{O}_l$, each built from n_{OVl} SSV volumes $\mathscr{V}_{lm}$:

$$\mathscr{R} = \{\mathscr{S}_i | \, 0 \le i < n_S\}, \tag{5.29}$$

$$\mathscr{S}_i = \{\mathscr{V}_{ik} | \, 0 \le k < n_{SVi}\}, \tag{5.30}$$

$$\mathscr{E} = \{O_l | \, 0 \le l < n_O\}, \tag{5.31}$$

$$\mathscr{O}_l = \{\mathscr{V}_{lm} | \, 0 \le m < n_{OVl}\}. \tag{5.32}$$

The collision environment $\mathscr{C}$ is defined as a set of n_C collision pairs $\mathscr{P}_k$, which in turn consist of two different segments $\mathscr{S}_F, \mathscr{S}_T$ (from/to):

$$\mathscr{C} = \{\mathscr{P}_l | \, 0 \le l < n_C\}, \tag{5.33}$$

$$\mathscr{P}_l = \{(\mathscr{S}_F, \mathscr{S}_T) | \, \mathscr{S}_F \in \mathscr{R}, \mathscr{S}_T \in \{\mathscr{R} \setminus \mathscr{S}_F \cup \mathscr{E}\}\}. \tag{5.34}$$

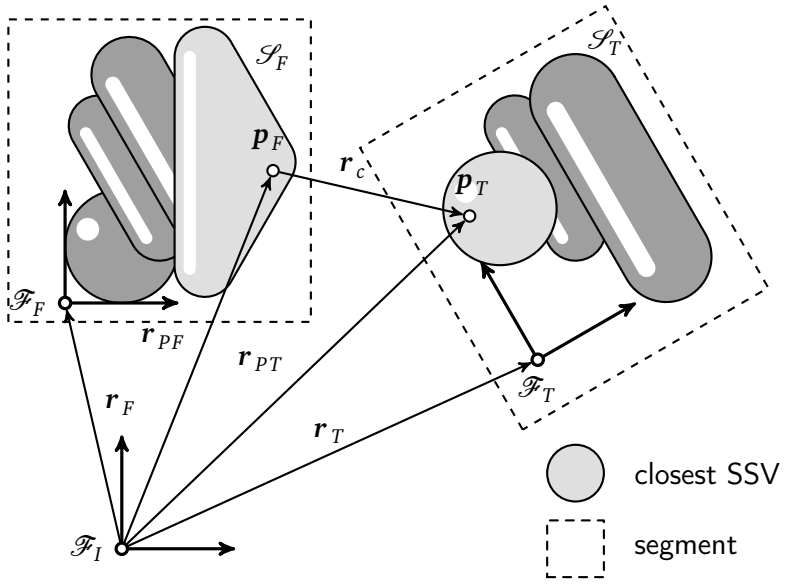


Figure 5.5: A collision pair with the definition of the shortest distance vector $\boldsymbol{r}_c$ from segment F to segment T (redrawn from Schwienbacher et al. 2011).

Figure 5.6 shows the simplified collision model for the bipedal robot Lola. For collision detection, the minimum distance for each collision pair $\mathscr{P}_k$ must be calculated. For collision avoidance, we additionally require the determination of the locations on the segments that have this minimum distance. The situation is illustrated in figure 5.5.

Computing the minimum distance between two segments in turn requires calculating the distance between all SSV primitives the segments are composed of. The minimum distance between two SSVs is obviously given by the minimimum between the generating primitives, minus the radii of the corresponding spheres.

For complex geometries, the procedure can be sped-up by adding bounding boxes to the segments (for details see Schwienbacher 2013). Alternatively, a hierarchy of bounding SSVs can be constructed. We have used this procedure for speeding up collision detection between a walking robot and obstacles in the environment (see section 5.4, figure 5.13).

5.2.3 Distance Calculation

Finding the minimum distance between two segments $\mathscr{S}_i$ is equivalent to minimizing the square of the distance:

$$\frac{1}{2}(\mathscr{S}_1 - \mathscr{S}_2)^2 \rightarrow \min! \tag{5.35}$$

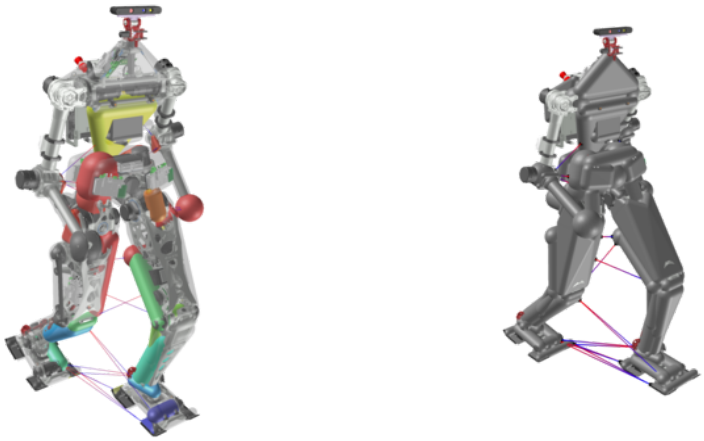


Figure 5.6: Visualization of the collision geometry of the robot Lola during walking, together with the underlying CAD-geometry. In the left image, the closest SSVs in a segment is drawn opaquely, with the color indicating the minimum distance to other segments. Objects with sufficient distance to others are transparent. In the right image, all SSVs are rendered in a uniform, opaque gray.

Since the segments $\mathscr{S}_i$ are composed of PSSs, LSSs and TSSs, the problem is solved by calculating the minimum between all SSV primitives $\mathscr{O}_k$ in both segments. The basic task therefore is to minimize the distance between two SSVs:

$$\frac{1}{2}(\mathscr{O}_1 - \mathscr{O}_2)^2 \rightarrow \min! \tag{5.36}$$

In the following, the basic procedure is outlined. A detailed description of an efficient numerical implementation is given in (Schwienbacher 2013, section 6.4).

PSS to PSS

The PSS is defined by its center $\boldsymbol{v}_i$ and its radius R_i (see figure 5.4). The squared distance function d between two (generating volumes of) PSSs (index 1, 2) is obviously given by the constant

$$d_{PP} = \frac{1}{2}(\boldsymbol{v}_1 - \boldsymbol{v}_2)^2 \tag{5.37}$$

and the corresponding points on the generating volumes are the vertices $\boldsymbol{v}_i$ themselves.

PSS to LSS

An equivalent optimization problem for determining the minimum distance between a PSS ($\boldsymbol{v}_3$) and an LSS ($\boldsymbol{v}_1, \boldsymbol{v}_2$) is

$$\begin{aligned} d_{PL} &= \frac{1}{2}(\boldsymbol{v}_1 + (\boldsymbol{v}_2 - \boldsymbol{v}_1)s - \boldsymbol{v}_3)^2 \\ &= \frac{1}{2}(\underbrace{(\boldsymbol{v}_1 - \boldsymbol{v}_3)}_{\boldsymbol{w}} + \underbrace{(\boldsymbol{v}_2 - \boldsymbol{v}_1)}_{\boldsymbol{d}}s)^2 \rightarrow \text{min!}, \quad s \in \Omega_L. \end{aligned} \tag{5.38}$$

Since the problem is strictly convex, we can determine the solution by calculating the minima in the unconstrained case and along the edges of Ω_L. The global minimum is then obtained by choosing among these. If the solution to the unconstrained problem is in Ω_L, it is the global solution. Otherwise, the candidate solutions along the edges must be compared. The unconstrained minimum is obtained from:

$$\frac{\partial d_{PL}}{\partial s} = (\boldsymbol{d}s_1 + \boldsymbol{w})^T\boldsymbol{d} = 0, \tag{5.39}$$

$$\Rightarrow s_1 = \frac{-\boldsymbol{w}^T\boldsymbol{d}}{\boldsymbol{d}^2}. \tag{5.40}$$

The function values at the edges of Ω_L ($s = 0, 1$) are given by:

$$d_{PL}(0) = \frac{1}{2}(\boldsymbol{w}), \tag{5.41}$$

$$d_{PL}(1) = \frac{1}{2}(\boldsymbol{w} + \boldsymbol{d})^2. \tag{5.42}$$

Apparently, the global solution depends only on the value of s_1 (5.40):

$$s^\star = \begin{cases} s_1 & \text{for} \quad s_1 \in [0, 1], \\ 0 & \text{for} \quad s_1 < 0 \\ 1 & \text{for} \quad s_1 > 1. \end{cases} \tag{5.43}$$

The corresponding point on the LSS's line segment is obtained by substituting $s^\star$ into (5.27).

LSS to LSS

The minimum-distance problem for two LSSs L_1 (vertices $\boldsymbol{v}_{1,1}, \boldsymbol{v}_{1,2}$) and L_2 (vertices $\boldsymbol{v}_{2,1}, \boldsymbol{v}_{2,2}$) is equivalent to:

$$\begin{aligned} d_{LL} &= \frac{1}{2}\Big[\boldsymbol{v}_{1,1} + (\boldsymbol{v}_{1,2} - \boldsymbol{v}_{1,1})s_1 - \big(\boldsymbol{v}_{2,1} + (\boldsymbol{v}_{2,2} - \boldsymbol{v}_{2,1})s_2\big)\Big]^2 \\ &= \frac{1}{2}\Bigg[\underbrace{(\boldsymbol{v}_{1,1} - \boldsymbol{v}_{2,1})}_{\boldsymbol{w}} + \underbrace{\big(\overbrace{\boldsymbol{v}_{1,2} - \boldsymbol{v}_{1,1}}^{\boldsymbol{d}_1} \quad \overbrace{\boldsymbol{v}_{2,1} - \boldsymbol{v}_{2,2}}^{\boldsymbol{d}_2}\big)}_{\boldsymbol{D}} \boldsymbol{s}\Bigg]^2 \to \min!, \quad \boldsymbol{s} \in \Omega_L \times \Omega_L. \end{aligned} \tag{5.44}$$

The unconstrained minimum at $\boldsymbol{s}_1$ is calculated from:

$$\frac{\partial d_{LL}}{\partial \boldsymbol{s}} = \boldsymbol{w}^T\boldsymbol{D} + \boldsymbol{s}_1^T\boldsymbol{D}^T\boldsymbol{D} = \boldsymbol{0}, \tag{5.45}$$

$$\Rightarrow \boldsymbol{s}_1 = -(\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D}^T\boldsymbol{r} \tag{5.46}$$

The global minimum $d_{LL}^\star$ can be determined by comparing the unconstrained minimum distances with the minimum distances obtained from all possible point-line calculation for the vertex/line-segment pairs in L_1, L_2

$$d_{LL}^\star = \min\{d_{LL}(\boldsymbol{s}_1), d_{PL}(\boldsymbol{v}_{1,1}, L_2), d_{PL}(\boldsymbol{v}_{1,2}, L_2), d_{PL}(\boldsymbol{v}_{2,1}, L_1), d_{PL}(\boldsymbol{v}_{2,2}, L_1)\}. \tag{5.47}$$

The procedure failes when $\boldsymbol{D}^T\boldsymbol{D}$ is singular, which is the case for parallel line segments. In that case, the unconstrained minimum is the same for all points along the lines, including the vertices. The global solution can then be computed by omitting the first term in (5.47):

$$d_{LL}^\star = \min\{d_{PL}(\boldsymbol{v}_{1,1}, L_2), d_{PL}(\boldsymbol{v}_{1,2}, L_2), d_{PL}(\boldsymbol{v}_{2,1}, L_1), d_{PL}(\boldsymbol{v}_{2,2}, L_1)\} \tag{5.48}$$

If the parallel lines overlap, there is a continuous region along the lines with equal distance. In that case it can be more convenient to return the middle of this minimizing region, instead of one of the vertices. Some edges of the admissible region $\Omega_L \times \Omega_L$ can be omitted from the distance query after calculating the unconstrained minimum and analyzing its location relative to $\Omega_L \times \Omega_L$. For details of such an efficient implementation see (Schwienbacher 2013, section 6.4).

Queries Involving TSSs

The procedure for distance queries involving TSSs is similar to that outlined above. First, the unconstrained minimum is calculated by solving the associated quadratic problem. Then, the minima along the edges of the admissible region are calculated. Similar to the LSS-LSS query, this is solved by calling minimum-distance routines associated with simpler geometric primitives. The minimum distance is then obtained by choosing among the candidate solutions. Due to the large number of configurations that must be tested, distance queries involving TSSs are quite complex. For two TSSs, there are a

Distance function	absolute run-time (t_{abs}) [µs]	stdev(t_{abs}) [ns]	relative run-time (t_{rel}) [–]	stdev(t_{rel}) [–]
PSS-PSS	0.0538	0.422	1.00	0.000
PSS-LSS	0.0555	0.527	1.03	0.016
LSS-LSS	0.0773	0.483	1.44	0.012
PSS-TSS	0.0909	0.738	1.69	0.023
LSS-TSS	0.3351	0.568	6.23	0.054
TSS-TSS	0.9007	0.919	16.74	0.140

Table 5.1: Run-time comparison for SSV distance computations on Lola's on-board computer[a] for a set of test cases that include all possible geometrical configurations of SSV objects (from Schwienbacher et al. 2011).

a. Intel Core2 Duo at 2.33 GHz, running the 32-bit real-time OS QNX 6.4.1. More recent CPUs and 64-bit code perform significantly better.

total of 15 different configurations that must be considered: 9 edge/edge queries and 6 vertex/plane queries.[9] For details, the reader is again referred to (Schwienbacher 2013, section 6.4).

Run-time measurements from Lola's real-time computer are shown in table 5.1. Since queries involving the complex TSSs are about one-order of magnitude more expensive than simple PSSs, modeling accuracy and computational efficiency must be carefully weighed when constructing the collision model.

5.3 Local Nullspace Optimization for Walking and Manipulation

In this section, two application examples for real-time redundancy resolution are presented: a redundant manipulator for harvesting single crops and a bipedal walking robot. In both projects a velocity-level CLIK method based on Liégeois' approach is used (cf. section 5.1.2). The original implementation for the bipedal robot Lola is described in (Buschmann et al. 2009). It was subsequently extended to include self-collision avoidance in the nullspace of the task description, see (Schwienbacher et al. 2011; Schwienbacher 2013). The implementation for the agricultural robot was performed by Baur et al. (2012).[10]

9. The vertex/edge queries are implicitly included in edge/edge queries.

10. Markus Schwienbacher helped port the collision avoidance software to the agricultural robot.

Collision Avoidance

Collision avoidance in nullspace can be implemented using the potential-field approach described in section 5.1.2. The total collision potential is given by the sum over the contributions from all collision pairs:

$$H_c = \sum_{i \in \mathscr{C}} H_{c,i}(d_i). \tag{5.49}$$

The scalar d_i denotes the distance of the i-th collision pair. In principle, any potential $H_{c,i}$ that increases with the proximity of robot segments is admissible. From a practical point of view, it is desirable to have a zero potential field for safe distances and a rapidly increasing potential for smaller distances. Typical choices are quadratic and cubic functions, such as (e.g., Schwienbacher et al. 2011; Sugiura et al. 2007):

$$H_{c2}(d) = \begin{cases} s_0(d-d_a)^2 & \text{for} \quad d < d_a, \\ 0, & \text{otherwise.} \end{cases} \tag{5.50}$$

$$H_{c3}(d) = \begin{cases} s_0(d-d_a)^3 & \text{for} \quad d < d_a, \\ 0, & \text{otherwise.} \end{cases} \tag{5.51}$$

Here, $d_a > 0$ is an activation threshold and $s_0 > 0$ a gain factor. Better performance for a wider range of velocities is obtained for a piecewise function constructed from a quadratic term for low distances and a cubic term for larger distances (cf. Schwienbacher 2013):

$$H_{c23} = \begin{cases} H_2(d) & \text{for} \quad d < t d_a, \\ H_3(d) & \text{for} \quad t d_a \le d \le 0, \\ 0 & \text{for} \quad d > d_a. \end{cases} \tag{5.52}$$

The functions $H_2(d), H_3(d)$ are general polynomials of 2nd and 3rd degree:

$$H_2(d) = \sum_{i=0}^{2} a_i d^i, \tag{5.53}$$

$$H_3(d) = \sum_{i=0}^{3} b_i d^i. \tag{5.54}$$

The seven unknown parameters a_i, b_i are calculated from C^2-continuity conditions at the switching point $d = t d_a$, the given slope s_0 for zero distance and the choice of a zero potential, zero gradient and zero curvature at the activating distance:

$$H_2'(0) = s_0, \tag{5.55}$$

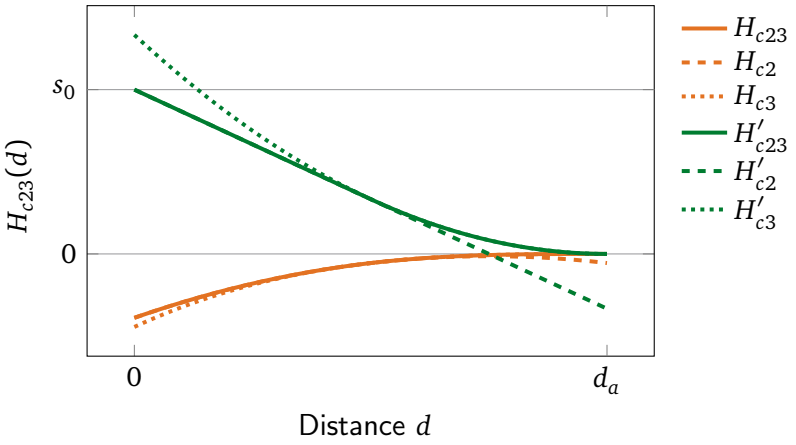


Figure 5.7: Piecewise definition of the repulsive collision avoidance potential and the corresponding gradient (cf. (5.58), (5.59)).

$$H_2(td_a) - H_3(td_a) = H'_2(td_a) - H'_3(td_a) \quad = H''_2(td_a) - H''_3(td_a) \quad = 0, \tag{5.56}$$
$$H_3(d_a) = H'_3(d_a) \quad = H''_3(d_a) \quad = 0. \tag{5.57}$$

The solution is obtained by straightforward calculations:

$$H_{c23} = \begin{cases} \frac{-s_0 d^2}{(d_a(t+1)} + s_0 d - \frac{s_0 d_a (t^2+t+1)}{3(t+1)} & \text{for} \quad d < td_a, \\ \frac{s_0 (d_a - d)^3}{3d_a^2 (t^2-1)} & \text{for} \quad td_a \le d \le 0, \\ 0 & \text{for} \quad d > d_a. \end{cases} \tag{5.58}$$

The corresponding gradient is given by:

$$H'_{c23} = \begin{cases} \frac{-2s_0}{d_a(t+1)} d + s_0 & \text{for} \quad d < td_a, \\ \frac{-s_0 (d_a - d)^2}{d_a^2 (t^2-1)} & \text{for} \quad td_a \le d \le 0, \\ 0 & \text{for} \quad d > d_a. \end{cases} \tag{5.59}$$

The function is illustrated in figure 5.7.

Collision avoidance in nullspace according to (5.7) requires computing the gradient $\frac{\partial H_{c23}}{\partial \boldsymbol{q}}$:

$$\frac{\partial H_{c23}}{\partial \boldsymbol{q}} = \frac{\partial H_{c23}}{\partial d} \frac{\partial d}{\partial \boldsymbol{q}}. \tag{5.60}$$

With reference to figure 5.5, the distance is given by $d = \sqrt{\boldsymbol{r}_c^2}$, with the gradient

$$\frac{\partial d}{\partial \boldsymbol{q}} = \frac{1}{d} \boldsymbol{r}_c^T \frac{\partial (\boldsymbol{p}_T - \boldsymbol{p}_F)}{\partial \boldsymbol{q}} = \frac{1}{d} \left({}_T\boldsymbol{r}_c^T \, {}_T\boldsymbol{J}_{T,T} - {}_F\boldsymbol{r}_c^T \, {}_F\boldsymbol{J}_{T,F} \right). \tag{5.61}$$

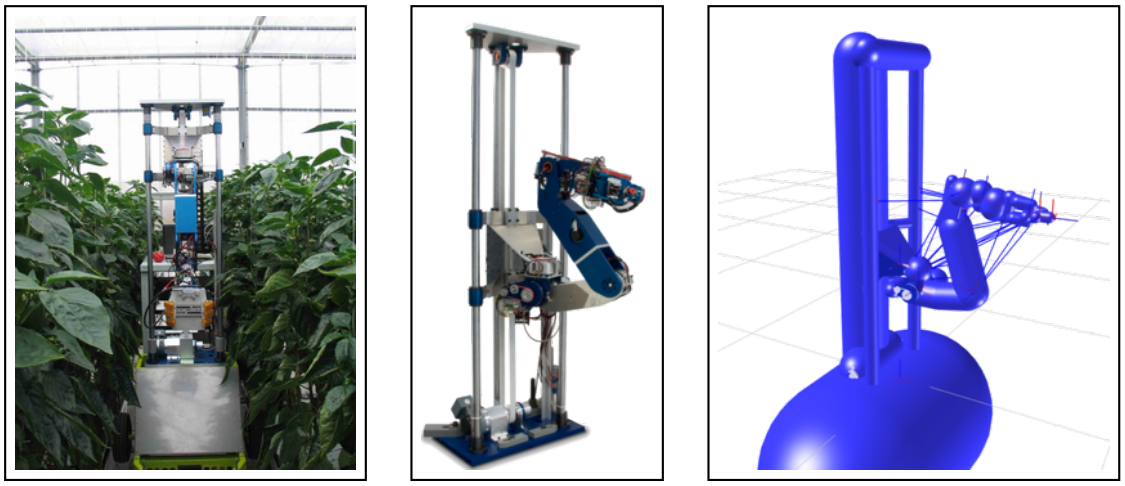


Figure 5.8: Photographs of the CROPS manipulator in a greenhouse for bell peppers (left, with gripper) and in the lab (center, no gripper). The collision model is shown shown on the right (from Baur et al. 2013). Experiments in the greenhouse, greenhouse platform design and system integration realized by Wageningen University and Research Center (WUR), gripper designed and built by FESTO.

The matrices $\boldsymbol{J}_{T,T}, \boldsymbol{J}_{T,F}$ are the translational Jacobians for the closest points on the corresponding robot segments, which are calculated via the methods described in section 2.3.2. It is worth noting that the rotational and translational Jacobians of all bodies are readily available as intermediate results, since the task-space Jacobians are computed by the recursive approach.

This collision-avoidance scheme must be combined with a joint-limit avoidance potential, since the collision geometry generally does not account for collisions of neighboring segments. This is easily achieved by potentials similar to the one described above, when d is taken as the distance to a joint limit.

The same approach was also implemented for a redundant, nine-DoF agricultural manipulator, including SSV-based collision avoidance and joint-limit avoidance in nullspace in combination with joint velocity minimization (see Baur et al. 2012). The manipulator and collision model are shown in figure 5.8, the kinematic structure in figure 5.9. The effect of collision avoidance is illustrated in figure 5.10.

Angular Momentum Minimization

A straightforward extension to this scheme, which is relevant for walking robots, is the integration of angular-momentum into the IK-scheme by either tracking references or minimizing the angular momentum in the nullspace. The relevance of angular momentum for biped walking has been recognized previously (see, e.g., Goswami and Kallem 2004; Orin and Goswami 2008). A well-known method in humanoid robotics involving angular and linear momentum is the “resolved momentum control” proposed

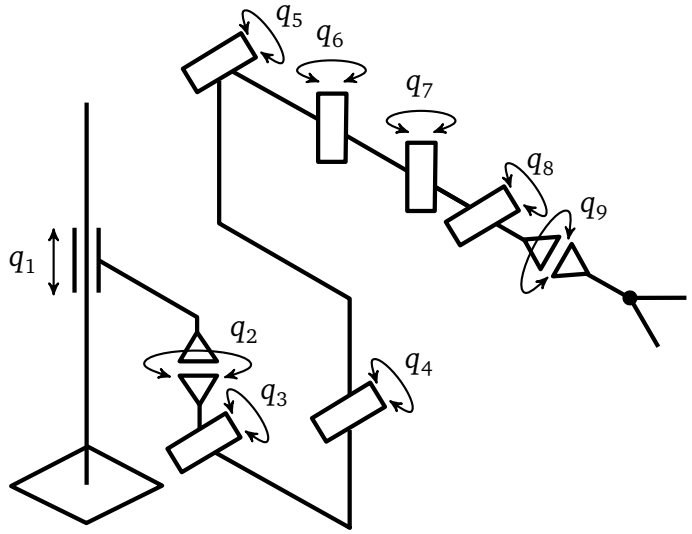


Figure 5.9: Kinematics of the modular 9-DoF agricultural manipulator (adapted from Schuetz et al. 2014; Baur et al. 2012).

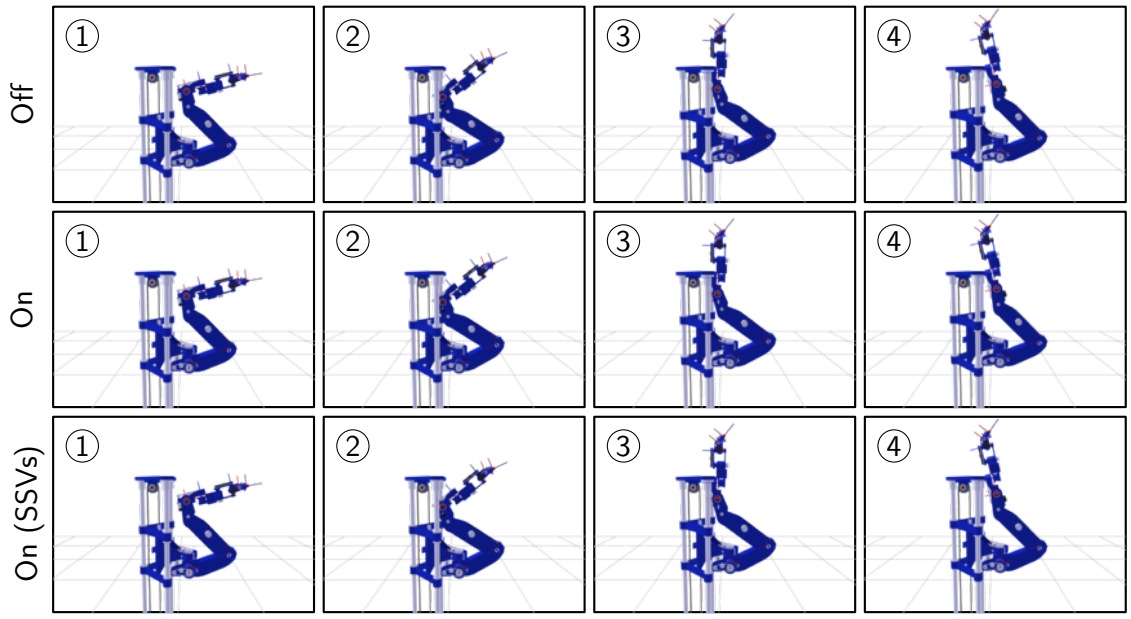


Figure 5.10: Example of the effect of nullspace collision avoidance for a point-to-point trajectory of the agricultural manipulator. The top row shows an animation without collision avoidance, which leads to self-collision for snapshot 3 and 4. Active collision avoidance (middle row: CAD geometry, bottom row: SSV collision geometry) prevents this (adapted from Baur et al. 2013).

by Kajita et al. (2003a). This method allows tracking a prescribed angular and linear momentum trajectories. The cited approach, however, does not directly implement this in the general framework of velocity-level IK, but instead directly makes use of the specific structure of a humanoid robot.

Within the general scheme described in section 5.1.2, exact tracking of linear and angular momentum is trivial. In fact, linear momentum (or, equivalently the CoM), have been used in the task-space coordinates of bipedal robots for many years (Buschmann et al. 2005, 2007; Löffler et al. 2002). The difficulty in tracking angular momentum lies in the prescription of adequate reference trajectories (which is not the subject of Kajita et al. 2003a). This difficulty is caused by the fact that the exact limits of feasible angular momentum depends on the details of the MBS model and is not easily predictable. That is, planning the angular momentum within the feasible limits requires taking the full kinematic complexity into account. This, however, would run contrary to the intention of the hierarchical approach aimed at reducing the planning complexity to enable real-time planning. A further complication is due to the fact that angular momentum is non-integrable, which makes the compensation of numerical drift less straight-forward than for linear momentum/CoM trajectories.

This motivated the development of the angular momentum minimization approach (see also Schwienbacher et al. 2012). Since the angular momentum for the whole system depends linearly on the generalized velocities ($\boldsymbol{L} = \boldsymbol{J}_L \dot{\boldsymbol{q}}$), it is easily integrated into the primary nullspace objective function $\frac{1}{2}\dot{\boldsymbol{q}}\boldsymbol{W}\dot{\boldsymbol{q}}$. In biped walking, minimizing the vertical angular momentum $L_z = \boldsymbol{J}_{Lz}\dot{\boldsymbol{q}}$ is especially important to avoid slipping at high velocities. This can be taken into account by the cost function

$$\frac{\beta}{2}(\gamma \dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}} + L_z^2) = \frac{\beta}{2}\dot{\boldsymbol{q}}^T \underbrace{(\gamma \boldsymbol{I} + \boldsymbol{J}_{Lz}^T \boldsymbol{J}_{Lz})}_{\boldsymbol{W}} \dot{\boldsymbol{q}}. \tag{5.62}$$

Including the squared joint velocities with a weight of γ avoids a singular optimization problem. The scalar β is used to scale (5.62) relative to other nullspace potential functions and to choose the step size for the descent along the direction of the projected gradient. The drawback of this method is a non-constant and non-diagonal $\boldsymbol{W}$ matrix, which makes solving the IK more expensive.[11] The method enables a significant reduction of the resulting angular momentum and also of the vertical contact moments (see figure 5.11). For small step-lengths an effective reduction to zero is possible. For larger step length the mass distribution of the robot makes this physically impossible, but L_z^2 is still reduced significantly.

Figure 5.12 shows snapshots from a walking experiment with the robot Lola with active angular momentum minimization and collision avoidance.

11. An efficient inversion is possible using the Sherman-Morris-Woodbury identity.

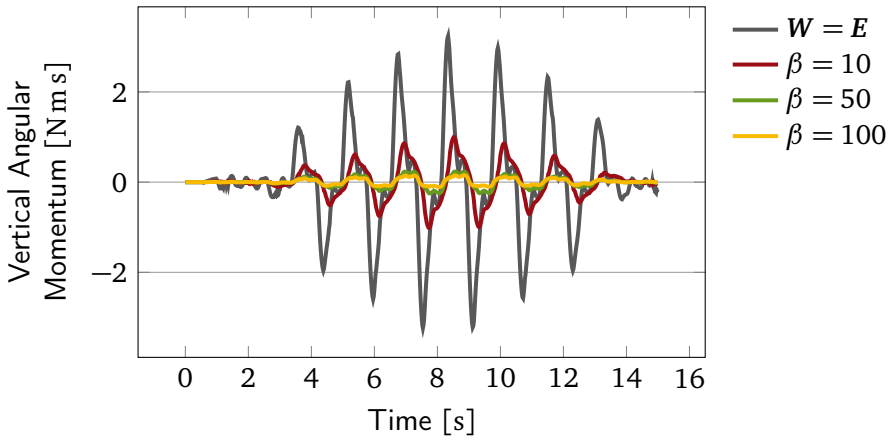


Figure 5.11: Vertical angular momentum for a reference walking pattern with a unit nullspace weighting matrix (no angular momentum minimization) compared to solutions with nullspace angular momentum minimization based on (5.62). For angular momentum minimization, $\gamma = 0.008$ and β is given in the key. In both cases the robot is walking with a maximum step length of 60 cm and a step duration of 0.8 s (results from MBS simulation).

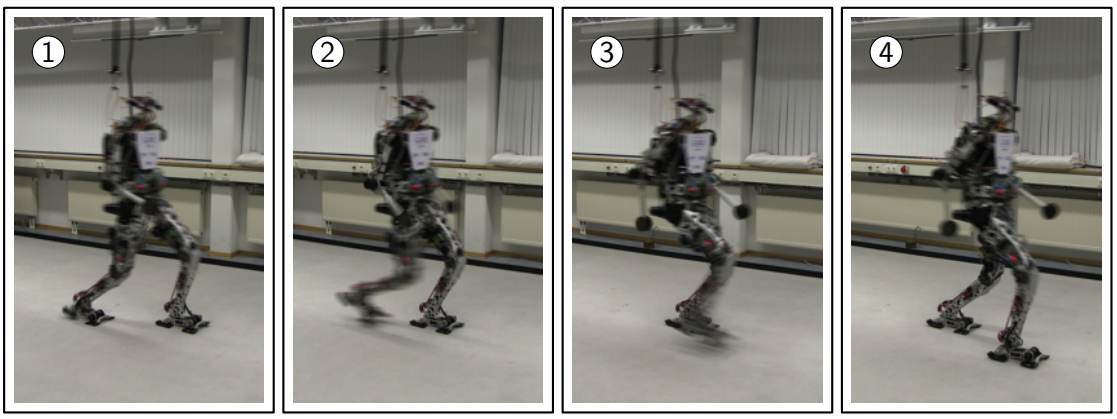


Figure 5.12: Snapshots from an experiment of the robot Lola walking at 3.5 km/h. A velocity-level CLIK-algorithm with nullspace optimization is used. Optimization criteria are collision avoidance, angular momentum minimization, joint limit avoidance and a preference posture. The arm and pelvis motion is not set by task-space references, leading to frequent collisions without active collision avoidance. Arm swinging counteracting the leg motion is the result of nullspace optimization (see text).

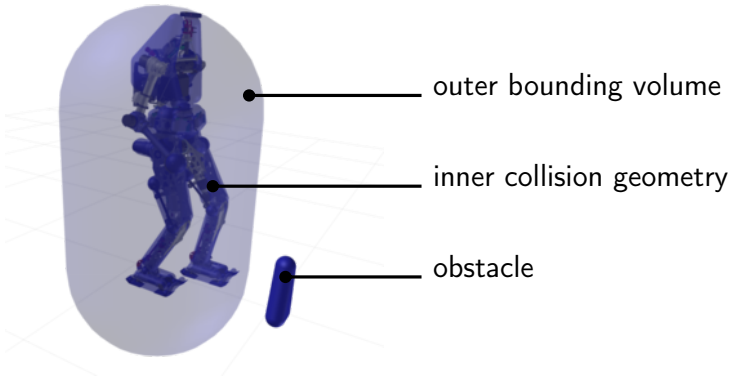


Figure 5.13: Collision geometry for the robot Lola during an obstacle avoidance experiment. Calculation times are reduced by using a simple bounding volume hierarchy with one LSS bounding the entire robot. Distance computations between the obstacle and the complex robot SSV-geometry are only necessary when the obstacle is within the bounding LSS.

5.4 Collision Avoidance in Task-Space

5.4.1 Introduction and Related Work

As described in the preceding sections, collision avoidance is important for any complex, branched robotic system. Bipedal robots must also avoid obstacles during locomotion in cluttered environments. This is a very challenging task due to the complex 3D geometry and large number of DoFs. One of the primary advantages of legged robots over tracked or wheeled vehicles is their potential for locomotion in rough terrain, including stepping over or onto large and complex-shaped obstacles. Solving the problem of walking in cluttered, dynamically changing environments by real-time planning and control is therefore essential.

The objective of this section is to present a method suitable for generating collision-free and stable trajectories in 3D for bipeds walking among arbitrary obstacles while exploiting all the system's DoFs. The emphasis is on locally modifying the walking pattern. It is assumed that the general direction of motion is given by the user or a higher-level path planner. The method is the outcome of joint work with Arne-Christoph Hildebrandt and others and was previously published in (Hildebrandt et al. 2014).

The self-collision avoidance algorithm discussed in section 5.3 can be extended to include collisions with the environment. Since it operates in the nullspace of the primary task, however, the algorithm requires the existence of a collision-free solution to the primary task, which cannot be guaranteed by simple real-time task-space planners.

This section presents a real-time approach to collision avoidance based on modifying the *task-space* trajectories. The method generates locally optimized trajectories online

during locomotion in order to avoid complex 3D obstacles. The proposed method combines a heuristic trajectory and footstep re-planning algorithm based on the obstacle's bounding box with a local potential-field method for obstacle avoidance in task-space. The system was integrated into Lola's walking control system and self-collision avoidance framework (cf. section 5.3).

There is a large body of work on navigation of mobile robots in known and unknown environments, including specializations for legged robots. Most approaches to navigating humanoids in cluttered environments are hierarchical, with higher layers planning discrete footstep sequences based on global mapping data (Seara et al. 2002; Chestnutt et al. 2003, 2010; Baudouin et al. 2011). Footstep locations are planned and rated based on heuristics of which locations are feasible and desirable, approximations of which obstacles can be traversed and (mostly) fixed action sets. We previously presented work on autonomous locomotion of bipeds in unknown environments (Buschmann et al. 2010).

There is a smaller body of work dedicated to the narrower task of stepping over obstacles. Guan et al. investigated the feasibility of stepping over obstacles for bipedal robots and have proposed a quasi-static trajectory planner for this task (Guan et al. 2004, 2005, 2006). Dynamically stepping over obstacles has also been studied (Stasse et al. 2009; Verrelst et al. 2006b, 2006a; Arbulú et al. 2010). From a planning perspective, the main difference to quasi-static planning is the necessity of taking the feasibility of generated contact forces into account. Fundamentally, this can be enforced by using adequate planning methods for the overall CoM motion (e.g., Buschmann et al. 2007), or the method for exactly tracking force references (see section 5.5). Many systems track the torso position instead of the CoM. This, in turn, leads to strong perturbations for fast swing leg trajectories, which are not such a significant issue for Lola's walking control system thanks to exact CoM tracking and angular momentum minimization.

Arbulú et al. (2010) modeled obstacles and the lower part of the swing leg as boxes. Based on this representation, the algorithm checks collisions of the robot with the obstacle for several key configurations via distance calculations between the boxes before the step is executed. Based on the key frames, smooth swing foot trajectories are interpolated using clamped splines. The robot's motion is finally generated using the preview control and resolved momentum control algorithms for planning (Kajita et al. 2003b, 2003a), and verified for feasibility by calculating the inverse dynamics of the multibody model.

A similar approach is proposed by Verrelst et al. (2006a, 2006b) and Stasse et al. (2009). During a step planning process the authors determine the required step length and the appropriate waist height for a collision-free double support phase. Based on this, foot trajectories are calculated. One important feature in comparison to Arbulú et al. (2010) is the horizontal online adaptation of the swing foot trajectory for collision avoidance. Unlike Arbulú et al. (2010), the collision detection used by Verrelst et al. relies on a 2D line segment model of the robot's legs and the obstacle.

Although the approaches mentioned above allowed humanoids to step over large

obstacles, they also have several disadvantages: (1) only collisions between the obstacle and the legs are considered, (2) very simplified geometric models are used, (3) potential self-collisions and the complex 3D parts of the robot and the environment are not taken into account. (4) Most importantly, the methods are limited to foot movements in a plane and do not allow for more general movements which make use of all of the robot's DoFs. Finally, the methods are probably not all suitable for real-time use.[12]

Many promising approaches that address several of the problems mentioned above have been proposed for generating reaching and grasping motions. El Khoury et al. (2013) proposed a quite general strategy based on generating an initial path using Rapidly-Exploring Random Trees (RRTs) (see LaValle 1998) and calculating a final solution using a multiple shooting optimal control approach. Although general enough for complex walking tasks, the approach is not suitable for real-time use in the controller of a complex humanoid during locomotion, due to the computational cost.

The work by Michael Gienger et al. is especially relevant, since it enables real-time motion planning for complex, branched systems with collision and obstacle avoidance. In this system, geometries are approximated using SSVs and self-collision is implemented using a projected gradient approach at the velocity level. The group has presented different hierarchical real-time planning methods. The basic layer consists of velocity-level redundancy resolution, whose references are generated by task-space attractors. The low number of parameters for the attractors enable online optimization for reaching and grasping motions (Toussaint et al. 2007; Gienger et al. 2008b). Behnisch et al. (2010) combined the approach with task-space randomized path planning using the RRT algorithm. A subsequent paper introduces a gradient-projection approach to obstacle avoidance. In this approach, modifications are projected into *task-space* (Behnisch et al. 2011; Behnisch 2012). From a kinematic point of view these approaches are suitable for generating stepping trajectories in cluttered environments. However, due to their origin in grasping, dynamic constraints and exact timing are not considered. We therefore cannot directly apply these methods to biped walking control. The idea of task-space gradient projection, however, is suitable for biped locomotion and was adopted for the system presented in this section.

The rest of the section is organized as follows: section 5.4.2 presents the proposed method, section 5.4.3 results from walking experiments and section 5.4.4 concludes the section with an outlook an future directions for research.

5.4.2 Proposed Method

As mentioned above, the framework for obstacle avoidance consists of two separate algorithms: (1) The Step Sequence and Heuristic Trajectory Adaption (SSTA) that modifies foot placement and reference trajectories during the planning phase to ap-

12. This is conjecture, since most papers do not mention computation times and the exact integration into the real-time control systems is not discussed.

proximately avoid obstacles and (2) the Reactive 3D Collision Avoidance (RCA), which takes the 3D geometries and kinematics into account. Figure 5.14 shows the integration of both algorithms into the overall planning process. For further information about the walking controller, see (Buschmann et al. 2007, 2009, 2011).

The robot and environment geometry are represented by SSV objects and distances are calculated by the framework described in section 5.2. Figures 5.15 and 5.18 show two walking experiments with obstacles, together with the corresponding collision geometries. As discussed in section 5.2.1, page 105, the distance functions between SSVs are not strictly convex, leading to discontinuities in the Jacobians (see Escande et al. 2007). Discontinuous references are undesirable and discontinuous gradients can pose a significant problem for optimization-based approaches. In this application, however, the exact value of the gradient is not relevant. Also, for a local method, convexifying the volumes by adding additional curvature, for example, in the form of Sphere Torus Bounding Volumess (STP-BVs), will simply smooth the transition from the discrete values observed at the discontinuity for the SSV geometry. In the interest of simplicity and run-time efficiency, we therefore simply temporally filter the Jacobians to obtain a smooth map for gradient projection. This approach allows us to use our highly optimized SSV library, enabling a control cycle of 2 ms including all planning, collision avoidance, redundancy resolution, force and position control tasks.[13]

Step Sequence and Heuristic Trajectory Adaption

The first part of the proposed system, the SSTA, is integrated into the hierarchical motion generation process. Based on simple, high level commands such as walking direction and velocity, the step sequence planner determines required parameters such as step lengths, step heights and CoM height. From these parameters, the walking pattern generator calculates ideal reference trajectories in task-space $\boldsymbol{w}_0(t) \in \mathbb{R}^m$. The basic motion pattern is re-computed once per step.

The SSTA is an extension of the step-sequence-planner presented in (Buschmann et al. 2008; Buschmann 2010). It modifies the stance foot positions and swing foot trajectories to obtain an initial solution according to the size of the obstacle's bounding box, which is calculated from the SSV geometry. In the first processing step, the situation is categorized according to the size and location of the obstacle:

1. The robot can set one foot next to the obstacle,
2. The robot has to step over the obstacle with both feet or
3. The obstacle is too large.

13. The real-time system for these experiments had an Intel® Xeon™ Quad CPU@2.3 GHz running the 32-bit Operating System (OS) QNX 6.50. The cycle time includes sampling sensor data and setting new motor commands, that is, there is no additional delay.

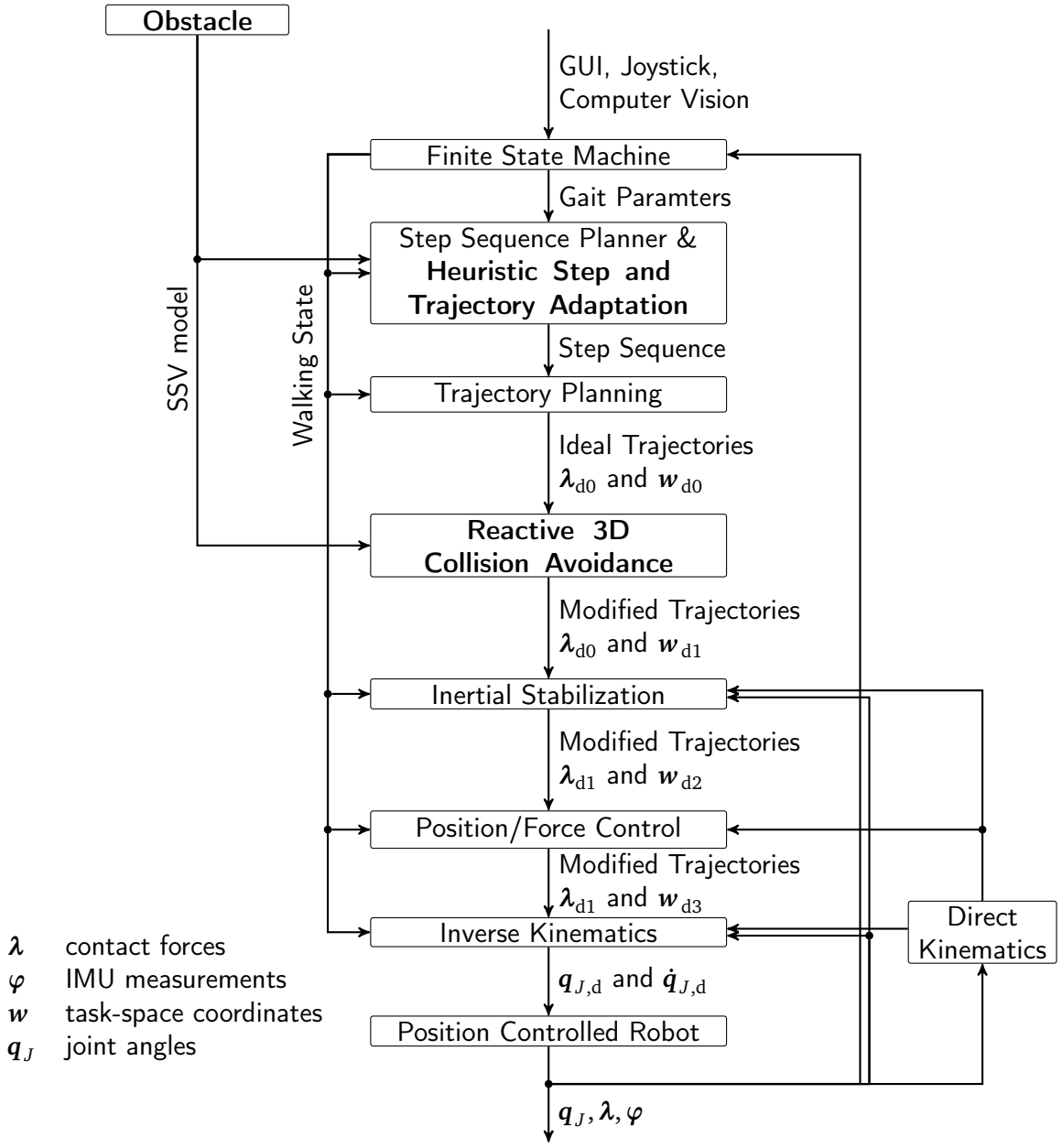


Figure 5.14: Integration of obstacle avoidance modules into Lola's real-time walking control system (additions shown in bold; figure adapted from Hildebrandt et al. (2014)). Reference values are indicated by an index dn, where n identifies modified values after different algorithms have been applied.

Based on this classification, the footstep locations are calculated using basic geometric reasoning.[14] The algorithm determines footstep parameters that minimize the required step lengths while maintaining pre-defined security margins from the obstacle's bounding box.

The task-space of the robot includes the position and spacial orientation of the swing leg, which is planned kinematically and represented by piecewise-defined 5th order polynomials (see Buschmann 2010). In the second processing step of the SSTA, the control points of the ideal swing foot trajectories are modified according to the obstacle's bounding box. The modification, again, depends on a geometric classification of the situation:

1. The robot must step over the obstacle (see first experiment with beam-shaped obstacle, figure 5.15). In that case, the height of the control points for the vertical position of the swing foot z_F are set to the height of the obstacle's bounding box.

2. The robot can swing the foot around the obstacle sideways (see second experiment with triangular obstacle, figure 5.18). In that case, the position of the control points for the lateral swing foot position y_F are shifted sideways, approximately by the overlap of swing foot and obstacle.

Providing approximate foot trajectories during planning has two major benefits. First, larger obstacles can be handled, since there are limits to the possible modifications of the reactive layer, especially for fast movements. Second, we use a three mass model for planning the CoM trajectory to improve the planning accuracy (see Buschmann et al. 2007). While local, reactive swing leg modifications act as a disturbance for the balance controller, the dynamic effect of swing leg trajectories set during the planning stage are taken into account. Note that the collocation method proposed by us (Buschmann et al. 2007) would also enable modifications of the vertical CoM trajectory, which might be valuable for very large obstacles. This has, however, not been necessary so far. Since we track the CoM, not the torso position, stretched leg configurations are not a kinematic singularity and CoM positions can be controlled via arm joint angles. That is, the leg segments can be repelled by the potential field, which is then compensated by the motion of other body parts (e.g., the arms). Note also that the heuristic trajectory modifications (generally) do not lead to collision-free motions. Their purpose is to reduce the modification that the reactive layer must perform and to increase the quality of the dynamic CoM trajectory planning.

Reactive 3D Collision Avoidance

The second part of the proposed system, the RCA, is integrated into the main feedback control loop responsible for stabilizing control and redundancy resolution (see fig-

14. Obviously, a further alternative would be to walk around the obstacle. Since we assume that the global planning has already been performed, however, this option is not considered here.

ure 5.14; Buschmann et al. (2011)). In the IK module, generalized velocities $\dot{\boldsymbol{q}}$ are obtained from task-space velocities $\dot{\boldsymbol{w}}$ via[15]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}_{\boldsymbol{w},\boldsymbol{W}}\dot{\boldsymbol{w}} - \alpha \boldsymbol{W}^{-1}\boldsymbol{N}_{\boldsymbol{w},\boldsymbol{W}}\left(\frac{\partial H_{\text{null}}}{\partial \boldsymbol{q}}\right)^{T}. \tag{5.63}$$

Since collision avoidance is handled by the nullspace potential H_{null} and its gradient is projected into the nullspace of $\boldsymbol{J}_{\boldsymbol{w}}$ via $\boldsymbol{N}_{\boldsymbol{w},\boldsymbol{W}}$, some collisions cannot be avoided by this method. Behnisch et al. (2011) proposes an additional task-space modification $\boldsymbol{w}_{\text{col}}$ that changes the original task-space trajectories $\boldsymbol{w}_{d0}$:

$$\boldsymbol{w}_{d1} = \boldsymbol{w}_{d0} + \boldsymbol{w}_{\text{col}} \tag{5.64}$$

To implement collision avoidance in task-space, we construct a potential function $H_{w,\text{col}}$ based on the same distance potential used for nullspace collision avoidance (5.58). The locally optimal choice for the task-space modification minimizes the value of the potential function at the next time step. For the velocity-level CLIK algorithm, the generalized coordinates at the next time step are given by

$$\boldsymbol{q}^{k+1} = \boldsymbol{q}^{k} + \Delta t \dot{\boldsymbol{q}}^{k} \tag{5.65}$$

and the first-order change of the potential function during the next time step by

$$\Delta H_{w,\text{col}} = \frac{\partial H_{w,\text{col}}}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}^{k+1}}{\partial \dot{\boldsymbol{w}}}\dot{\boldsymbol{w}}_{\text{col}}\Delta t = \left(\frac{\partial H_{w,\text{col}}}{\partial \boldsymbol{q}}\boldsymbol{J}^{\#}_{\boldsymbol{w},\boldsymbol{W}}\right)\dot{\boldsymbol{w}}_{\text{col}}\Delta t. \tag{5.66}$$

The direction for which $\dot{\boldsymbol{w}}_{\text{col}}$ locally minimizes the potential function (for a given length of $\dot{\boldsymbol{w}}_{\text{col}}$) obviously is minus the bracketed term in (5.66):

$$\dot{\boldsymbol{w}}_{\text{col}} = -\left(\boldsymbol{J}^{\#}_{\boldsymbol{w},\boldsymbol{W}}\right)^{T}\left(\frac{\partial H_{w,\text{col}}}{\partial \boldsymbol{q}}\right)^{T}. \tag{5.67}$$

This is the task-space modification proposed in (Behnisch et al. 2011; Behnisch 2012). The same result is obtained by solving the local optimization problem

$$\begin{aligned} \Phi &= \frac{1}{2}\dot{\boldsymbol{w}}^{2}_{\text{col}} + \Delta H \rightarrow \min! \\ \dot{\boldsymbol{q}} &= \boldsymbol{J}^{\#}_{\boldsymbol{w},\boldsymbol{W}}\dot{\boldsymbol{w}}_{\text{col}} \end{aligned} \tag{5.68}$$

with the optimality conditions

$$\frac{\partial \Phi}{\partial \dot{\boldsymbol{w}}_{\text{col}}} = \dot{\boldsymbol{w}}_{\text{col}} + \frac{\partial H}{\partial \boldsymbol{q}}\boldsymbol{J}^{\#}_{\boldsymbol{w},\boldsymbol{W}} = \boldsymbol{0}, \tag{5.69}$$

$$\Rightarrow \dot{\boldsymbol{w}}_{\text{col}} = -\left(\boldsymbol{J}^{\#}_{\boldsymbol{w},\boldsymbol{W}}\right)^{T}\left(\frac{\partial H_{w,\text{col}}}{\partial \boldsymbol{q}}\right)^{T}. \tag{5.70}$$

15. Note that this must be understood symbolically. The real-time system uses algorithm 10, page 101 and does not explicitly calculate pseudoinverses and nullspace projection matrices.

The time step Δt was omitted, since only the direction is relevant. Apparently this trajectory modification law minimizes the sum of the collision potential and squared task-space velocity.

We can derive a number of different task-space modification laws by changing the local optimization problem (5.68). One obvious choice is to replace $\dot{\boldsymbol{w}}^2$ by $\dot{\boldsymbol{q}}^2$ in order to penalize joint velocities instead of task-space velocities:

$$\begin{aligned} \Phi &= \frac{1}{2}\dot{\boldsymbol{q}}^2 + \Delta H \rightarrow \min! \\ \dot{\boldsymbol{q}} &= \boldsymbol{J}^{\#}_{w,W}\dot{\boldsymbol{w}}_{\text{col}}. \end{aligned} \tag{5.71}$$

The solution is obtained as (omitting the weighting matrix $\boldsymbol{W}$ for clarity):

$$\frac{\partial \Phi}{\partial \dot{\boldsymbol{w}}_{\text{col}}} = \frac{\partial}{\partial \dot{\boldsymbol{w}}_{\text{col}}}\left(\dot{\boldsymbol{w}}^T_{\text{col}}\left(\boldsymbol{J}^{\#}_{w}\right)^T \boldsymbol{J}^{\#}_{w}\dot{\boldsymbol{w}}_{\text{col}}\right) + \frac{\partial H}{\partial \boldsymbol{q}}\boldsymbol{J}^{\#}_{w}\dot{\boldsymbol{w}}_{\text{col}} \tag{5.72}$$

$$= \dot{\boldsymbol{w}}^T_{\text{col}}\left(\boldsymbol{J}_w\boldsymbol{J}^T_w\right)^{-1}\boldsymbol{J}_w\boldsymbol{J}^T_w\left(\boldsymbol{J}_w\boldsymbol{J}^T_w\right)^{-1}\boldsymbol{J}_w + \frac{\partial H}{\partial \boldsymbol{q}}\boldsymbol{J}^T_w\left(\boldsymbol{J}_w\boldsymbol{J}^T_w\right)^{-1} = \boldsymbol{0} \tag{5.73}$$

$$\Rightarrow \dot{\boldsymbol{w}}_{\text{col}} = \boldsymbol{J}_w\frac{\partial H}{\partial \boldsymbol{q}}. \tag{5.74}$$

The obvious merit of this version is the decreased numerical cost in calculating $\dot{\boldsymbol{w}}_{\text{col}}$, the disadvantage the smaller decrease in the potential ΔH for a given task-space modification.

For biped walking, we cannot use any of the above approaches, since we must assure proper tracking of the dynamically planned CoM trajectory and the position and orientation of feet during ground contact. Also, the timing and location of ground contact must be maintained. The exact trajectory of the foot and leg during swing, however, are not essential. So simply modifying the whole task-space is not acceptable.

We therefore limit the modification to the subspace of our task-space occupied by the Cartesian swing foot coordinates $\boldsymbol{w}_F \in \mathbb{R}^6$. The subset $\boldsymbol{w}_F$ and the remaining coordinates $\boldsymbol{w}_R \in \mathbb{R}^{m-6}$ can be defined by a binary selection matrix $\boldsymbol{S}_F$ and its compliment $\bar{\boldsymbol{S}}_F$:

$$\boldsymbol{w}_F = \boldsymbol{S}_F\boldsymbol{w}, \tag{5.75}$$

$$\boldsymbol{w}_R = \bar{\boldsymbol{S}}_F\boldsymbol{w}, \tag{5.76}$$

$$\boldsymbol{w} = \boldsymbol{S}^T_F\boldsymbol{w}_F + \bar{\boldsymbol{S}}^T_F\boldsymbol{w}_R. \tag{5.77}$$

The modification law directly corresponding to that proposed by Behnisch et al. (2011) is obtained by applying the chain rule

$$\begin{aligned} \Delta H_{w,\text{col}} &= \frac{\partial H_{w,\text{col}}}{\partial \boldsymbol{q}}\frac{\partial}{\partial \dot{\boldsymbol{w}}_{F,\text{col}}}\left[\boldsymbol{J}^{\#}_{\boldsymbol{w},W}\left(\boldsymbol{S}^T_F(\dot{\boldsymbol{w}}_{F,0} + \dot{\boldsymbol{w}}_{F,\text{col}}) + \bar{\boldsymbol{S}}^T_F\dot{\boldsymbol{w}}_{R,0}\right)\right]\dot{\boldsymbol{w}}_{F,\text{col}}\Delta t, \\ \Rightarrow \quad \dot{\boldsymbol{w}}_{F,\text{col}} &= -\boldsymbol{S}_F\left(\boldsymbol{J}^{\#}_{\boldsymbol{w},W}\right)^T\left(\frac{\partial H_{w,\text{col}}}{\partial \boldsymbol{q}}\right)^T. \end{aligned} \tag{5.78}$$

This is a viable approach, but it has the drawback that we must first calculate the whole task-space modification before selecting the swing-foot components. An alternative is to use a modified local optimization problem in which we minimize the modification $\dot{\boldsymbol{w}}_{F,\text{col}}$ and the potential function under the condition that the corresponding $\dot{\boldsymbol{q}}$ is obtained from the least squares solution of $\dot{\boldsymbol{w}}_{F,\text{col}} = \boldsymbol{S}_F \boldsymbol{J}_w \dot{\boldsymbol{w}}$.

$$\begin{aligned} \Phi &= \frac{1}{2}\dot{\boldsymbol{w}}_{F,\text{col}}^2 + \Delta H \rightarrow \min! \\ \dot{\boldsymbol{q}} &= \left(\boldsymbol{S}_F \boldsymbol{J}_{w,W}\right)^{\#} \dot{\boldsymbol{w}}_{\text{col}}, \end{aligned} \tag{5.79}$$

leading to the task-space modification

$$\dot{\boldsymbol{w}}_{F,\text{col}} = -\left[\left(\boldsymbol{S}_F \boldsymbol{J}_{\boldsymbol{w},W}\right)^{\#}\right]^T \left(\frac{\partial H_{w,\text{col}}}{\partial \boldsymbol{q}}\right)^T \tag{5.80}$$

$$\dot{\boldsymbol{w}}_{d1} = \dot{\boldsymbol{w}}_{d0} + \boldsymbol{S}_F^T \dot{\boldsymbol{w}}_{F,\text{col}}. \tag{5.81}$$

The modified task-space velocity $\dot{\boldsymbol{w}}_{d1}$ is sent to the lower feedback control layers (see figure 5.14). Note that the pseudoinverse in (5.80) does not have to be calculated explicitly, since the algorithm 10 can be applied for calculating $\dot{\boldsymbol{w}}_{F,\text{col}}$ directly.

As with nullspace collision avoidance, an additional nullspace potential for avoiding joint limits is required (see section 5.3).

Ideal Reference Trajectory Attractor and Re-planning

As explained above, it is crucial that the exact timing and location of foot contacts are maintained. This is equivalent to convergence of the modified trajectory $\boldsymbol{w}_{d1}$ to the original trajectory $\boldsymbol{w}_{d0}$ at the end of the step. We propose a two-layer approach to solve this problem. First, an additional potential function is introduced, which penalizes the divergence from the original trajectory

$$H_{w,\text{att}} = \frac{s_{\text{att}}}{2}(\boldsymbol{w}_{d1} - \boldsymbol{w}_{d0})^2. \tag{5.82}$$

The scalar s_{att} is a gain determining the relative weighing of this quadratic attractor. As with most potential-field approaches, H_{att} reduces the effect of the collision avoidance and joint limit avoidance potentials. Choosing the relative weights of parameters is quite important for the correct operation and the performance of the algorithms. The tuning is done manually, but the same set of parameters is used for all experiments.

While the attractor can reduce the distance to the original trajectory, the difference will never be reduced to zero. We therefore deactivate the task-space potentials before the end of the step and plan back to the original trajectory using a smooth, overshoot-free function. Planning C^2-smooth, overshoot free trajectories for arbitrary initial values back to zero is a surprisingly difficult problem. Here, we use the same approach used for planning fast stopping trajectories for event-triggered phase transitions in bipedal walking (see Ewald et al. 2012; Buschmann et al. 2012a). In summary, the

total task-space potential H_w is denoted by

$$H_w = H_{w,\text{coll}} + H_{w,\text{limit}} + H_{w,\text{att}}. \tag{5.83}$$

5.4.3 Experimental Evaluation

The proposed method has been analyzed both in closed loop dynamics simulations using our in-house simulation system (see Buschmann et al. 2006; Buschmann 2010) and in experiments with Lola. Both, simulations and experiments showed the efficiency and flexibility of the method in trials with obstacles of various shapes and sizes. In this section, two experiments are discussed. A step duration of $T_S = 1$ s and a main control cycle of Δt =2 ms was used. Obstacle geometries and positions were manually determined prior to the experiments. All calculations were performed in real-time within one control cycle. Both experiments used the identical controllers and parameter sets.

First Experiment – Rectangular Obstacle
In the first experiment a 16.1 cm high, 8 cm deep and 35 cm wide rectangular obstacle is used. It is modeled as an LSS with a radius of 10 cm and a center 7 cm beneath the floor. The scene is shown in figure 5.15. The obstacle is placed in a position where the normal step sequence would make both the robot's feet collide with it. The SSTA algorithm modifies the footstep locations to avoid the obstacle, positioning them as closely in front and behind the obstacle as possible. The right swing leg trajectory is modified according to the height of the obstacle (16.1 cm), allowing the robot to step over it. The left foot is swung around the obstacle (modification 3.8 cm), since this is the less severe modification in this situation. A top view of the modified footstep locations and foot trajectories is shown on the left of figure 5.17. The normal, regular walking pattern resumes after the obstacle, before the robot comes to a stop at approximately 4 m. The lateral and vertical components of the swing foot trajectories are shown in figure 5.16. Note that all six foot coordinates are modified, but only two are shown here for conciseness.

Second Experiment – Triangular Obstacle
In the second experiment a 30 cm high and 9 cm deep triangular obstacle is used that is 13 cm wide at the base. It is modeled by one TSS. The scene is shown in figure 5.18.

As with the first experiment, both the robot's feet would collide with the obstacle, if no evasive action would be taken. Since the obstacle is very high but not too wide, the SSTA algorithm choose to swing both feet around the obstacle, by approximately 4.1 cm for the left and 10 cm for the right foot. A top view of the modified footstep locations and foot trajectories is shown on the right of figure 5.17. The normal, regular walking pattern resumes after passing the obstacle. The lateral and vertical components of the swing foot trajectories are shown in figure 5.19, with other components omitted for conciseness.

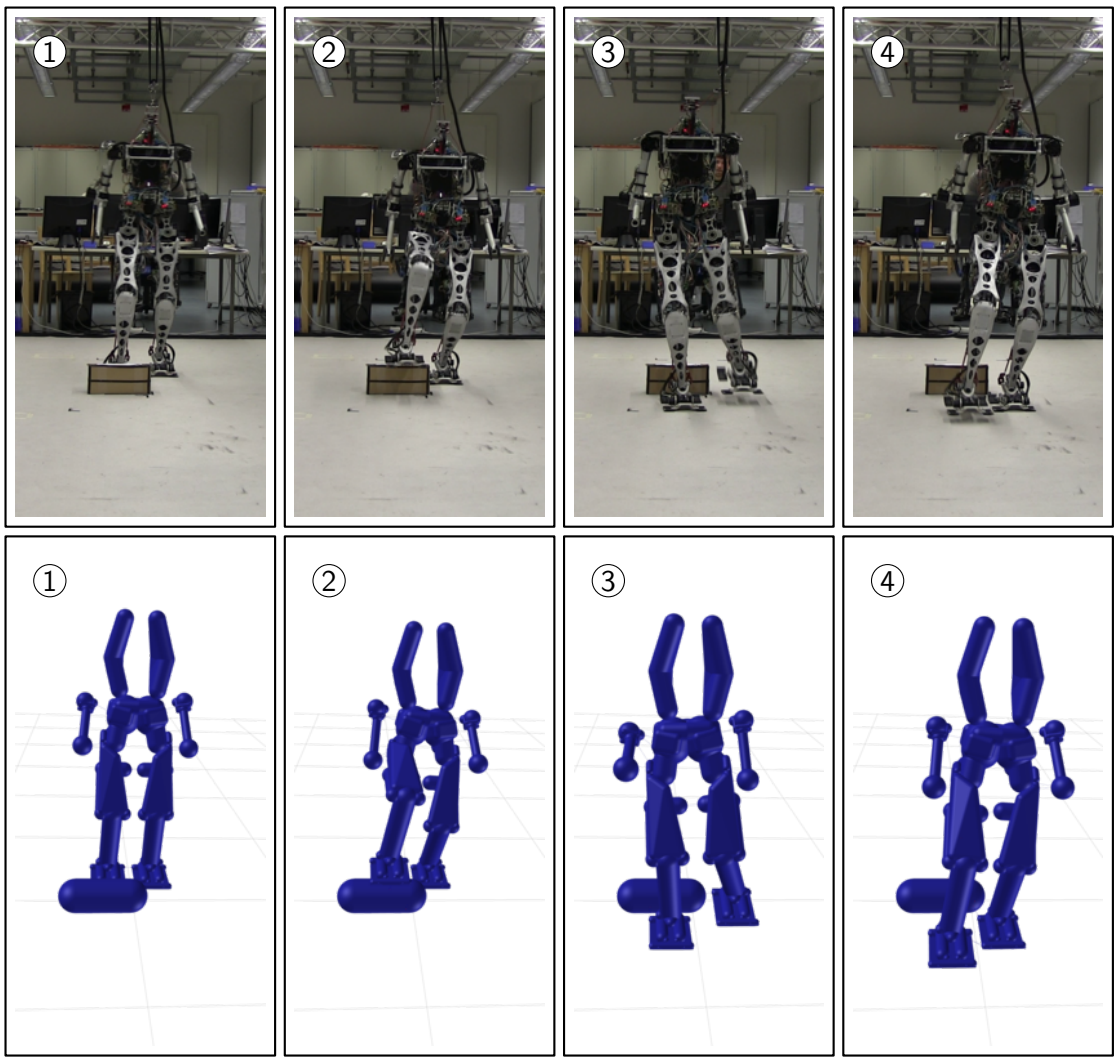


Figure 5.15: Photographs of Lola during the first obstacle avoidance experiment (top row) and corresponding collision model (bottom row; approximate pose, obtained from OpenGL rendering after the experiment). The obstacle is a 16.1 cm high, 8 cm deep and 35 cm wide block.

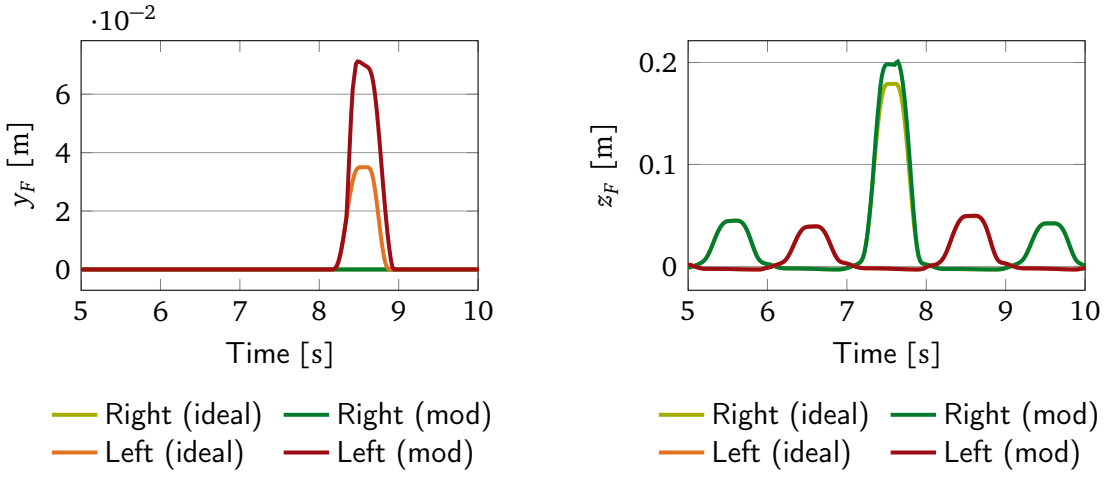


Figure 5.16: Foot trajectories for the first obstacle avoidance experiment (see figure 5.15). Both initial, heuristically modified (ideal) and final (mod) foot trajectories in the lateral (y_F) and vertical (z_F) directions are shown. Note that the sagittal position, as well as the spatial orientation of the foot are also modified, but not shown here (redrawn from Hildebrandt et al. 2014).

5.4.4 Summary

We developed a new method which enables a bipedal robot to walk over and around obstacles in a cluttered environment while avoiding collisions. This method extends the framework designed for self-collision avoidance presented in the previous section, enabling the robot to dynamically integrate obstacles into its collision model. Based on this, a local potential field method is combined with a heuristic modification of swing leg trajectories based on the height and width of an obstacle. This enables the robot to flexibly overcome obstacles of arbitrary shapes using locally optimized 3D trajectories. The efficiency of the proposed method is demonstrated in experiments with the robot Lola.

5.5 Inverse Kineto-Dynamics

This section describes an extension to the IK algorithms described in the previous sections to hybrid task descriptions consisting of both task-space trajectories and generalized forces. The approach was reported in a paper previously published by the author, upon which this section is strongly based (Buschmann et al. 2012c). The method presented here is motivated by the problem of biped walking control for which a natural task description consists of contact forces and task-space trajectories. We present both a general method based on the full EoM of the MBS, and a more efficient approach tailored to the problem of biped walking control. We analyze the method and

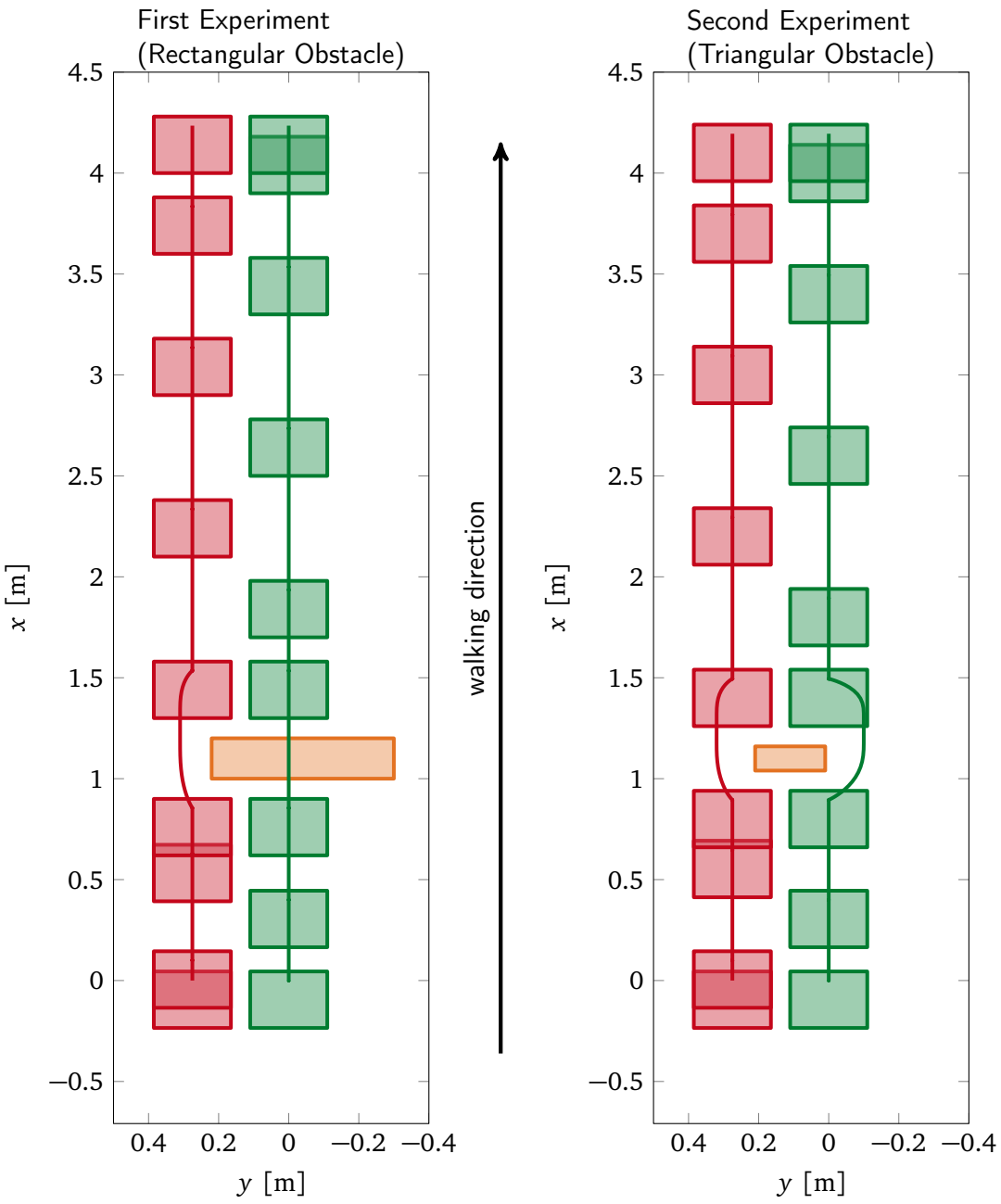


Figure 5.17: Modified step sequences for two obstacle avoidance experiments. Left stance foot positions shown in red, right stance foot positions in green and corresponding heuristically adapted TCP trajectories as red and green lines. Obstacle bounding box shown in orange (note that footprints are distorted; redrawn from Hildebrandt et al. (2014)).

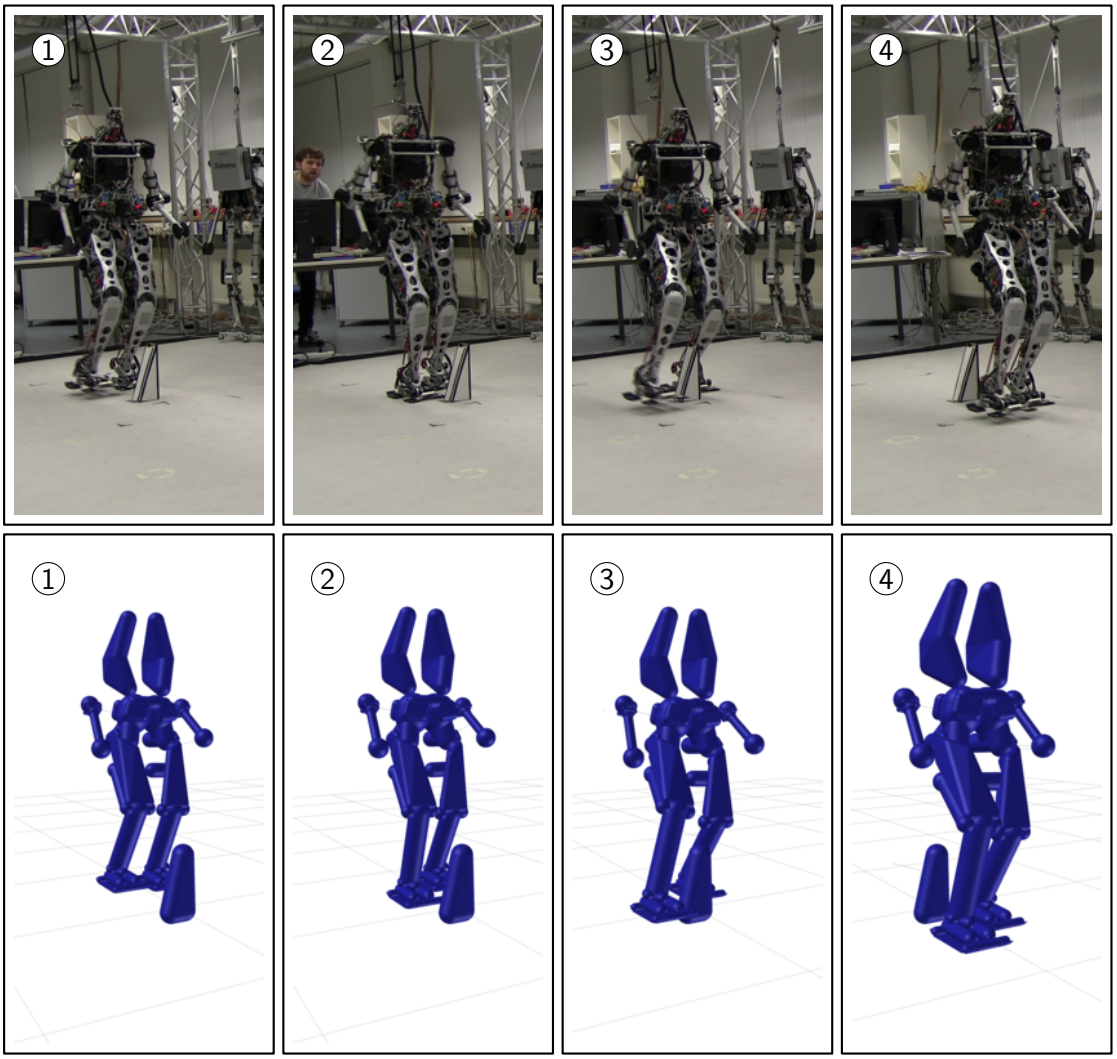


Figure 5.18: Photographs of Lola during the second obstacle avoidance experiment (top row) and corresponding collision model (bottom row; approximate pose, obtained from OpenGL rendering after the experiment). The obstacle is a triangle that is 30 cm high, 9 cm deep and 13 cm wide at the base.

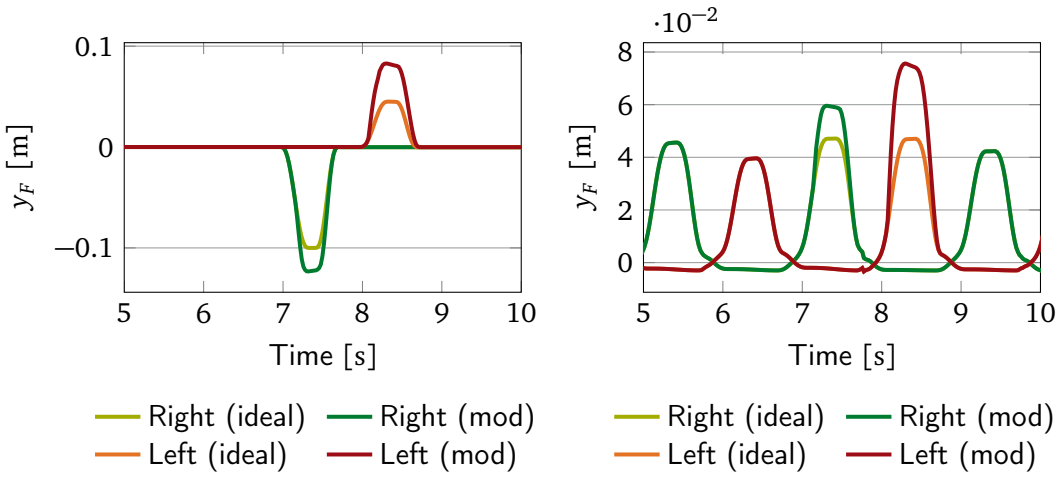


Figure 5.19: Foot trajectories for the second obstacle avoidance experiment (see figure 5.18). Both initial, heuristically modified (ideal) and final (mod) foot trajectories in the lateral (y_F) and vertical (z_F) directions are shown. Note that the sagittal position, as well as the spatial orientation of the foot are also modified, but not shown here (redrawn from Hildebrandt et al. 2014).

compare it to the dynamics error compensation method proposed by Takenaka et al. (2009b, figure 4), which can reduce the planning error resulting from simplified robot models. The controller is evaluated in walking experiments and compared to a different walking controller using an inverse kinematics algorithm. The method enables exact tracking of desired contact forces in real-time, since they are directly included in the task-description.

5.5.1 Introduction and Related Work

Generating and controlling stepping in humanoid walking robots is a complex task: The unilateral ground contact limits the set of feasible contact forces, the dynamics are non-linear and the system has a large number of DoFs (cf. chapter 4). The computational complexity of full MBS models has led to the development of planning methods based on simplified point-mass models (see section 4.5.1). While these methods enable real-time planning, they do not generate dynamically consistent force and motion references that respect the dynamic constraints. That is, if the inverse dynamics for a comprehensive model are solved along the trajectory generated with the simplified model, the calculated forces will differ from the targets used during trajectory generation. We have proposed a method based on spline collocation for the three point mass model which reduces the modeling error, but cannot fully remove it (see Buschmann et al. 2007).

Even though the widely used lumped-mass models are vast simplifications, they have

proven to be well-suited for walking pattern generation up to moderate walking speeds. At higher speeds, however, the influence of modeling errors becomes more severe. The difference between the desired contact forces or, equivalently, CoP trajectories for which the planning problem is solved and the values calculated by solving the inverse dynamics for a comprehensive model becomes unacceptably large. For this reason, approximate methods for reducing the effects of the modeling error have been proposed. All methods exploit the fact that the simplified linear model of the CoM dynamics also approximates the modeling error. The simplified model can therefore be reused to reduce the planning error without having to solve a planning problem for the full MBS model. Kajita et al. (2003b) used the preview control method for reducing the modeling error by feeding the difference between desired ZMP and that determined by inverse dynamics into the preview control method. Takenaka et al. proposed feeding the moment error into an inverted pendulum model and adding the output to the original CoM trajectory (Takenaka 2005; Takenaka et al. 2009b). These methods are capable of reducing the modeling error significantly. However, perfect tracking of the reference force is not possible and the size of the contact force tracking error depends on the specific robot motion and is not easily controllable.

This section describes a method that enables exact tracking of desired contact forces and task-space trajectories by directly including them into the task specification and using a full MBS model of the robot for generating joint-space trajectories.

The method tries to solve the same problem other researchers have addressed by incorporating angular momentum into walking pattern generation methods (see also: "Angular Momentum Minimization" in section 5.3, page 114). The "resolved momentum control" approach is notable, but suffers from the problem of prescribing adequate angular momentum trajectories (see discussion on page 114). Also, the method is used for offline generation of reference trajectories according to Kajita et al. (2005). Goswami and Kallem (2004) introduced a new criterion for stable states of the robot based on rate of change of angular momentum about the robot's CoM. Three different strategies are proposed to obtain a vanishing rate of change of angular momentum about the CoM. The angular momentum minimization method described in section 5.3 enables local minimization of the difference to a zero (or non-zero) angular momentum reference. Due to complex kinematic limits, however, the reference can usually not be perfectly tracked.

The method introduced in this section, by contrast, uses the correlation between ground reaction forces and the rate of change of angular momentum about the CoM $\dot{\boldsymbol{L}}_c$. With this, the full MBS model is taken into account, the contact forces which must strictly satisfy the conditions of unilateral ground contact are accurately tracked and a zero-reference for $\boldsymbol{L}_C$ or $\dot{\boldsymbol{L}}_C$ is avoided, which is usually unfeasible and leads to unnaturally looking movements.

5.5.2 The Concept of Inverse Kineto-Dynamics

As outlined above, motion generation can be simplified by choosing task-space coordinates that naturally matches the control task. Especially in the case of interaction control, a natural task description might contain both position and force variables. This has led to the development of hybrid position/force control, which was originally proposed for torque-controlled manipulators (Craig and Raibert 1979; Khatib 1987). The wider availability of position controlled devices has led to the development of hybrid position/force control methods based on an inner position or velocity control loop (Siciliano and Khatib 2008, chapter 7). We have previously proposed a method for hybrid position/force control of biped robots based on an inner velocity or position control loop (Buschmann et al. 2009).

While hybrid position/force control addresses the situation of a task description based on both position and force variables, it does not solve the problem of generating a reference trajectory for the generalized coordinates that satisfy a hybrid task description. This task is similar to differential IK algorithms, but includes both kinematic and kinetic quantities. We will therefore call this approach Inverse Kineto-Dynamics (IKD).

Vukobratović et al. (1990, chapter 1) described the "prescribed synergy method" for generating biped walking patterns. The method is related to the general IKD method presented in section 5.5.2 and motivated by similar observations, but is different in the following aspects. First, the "prescribed synergy method" works in joint space, calculating some joint angles from given contact forces and assuming that the other joint angles are given. This makes it more difficult to take geometrical boundary conditions such as early ground impact into account. Also, only contact forces are taken into account, not the general case of any generalized force studied in section 5.5.2. Additionally, the important aspect of stabilizing the system discussed in section 5.5.3 is not addressed. Finally, Vukobratović proposes numerically solving the Boundary Value Problem (BVP) using periodicity conditions, which makes this approach an offline method due to the high computational cost.

Inverse Kineto-Dynamics for Biped Walking

A dominant approach to walking pattern generation for biped robots consists of first determining feasible contact forces or (equivalently) CoP trajectories and then calculating the horizontal components of the CoM trajectory by solving the planning problem for a simplified model (see section 4.5.1 and 5.5.1). In this section we describe a method for directly generating joint trajectories from given contact moment and task-space trajectories.

The global dynamics of an arbitrarily complex MBS can be described by the angular and linear momentum theorems evaluated for the entire system:

$$\dot{\boldsymbol{p}} = m\ddot{\boldsymbol{r}}_C \quad = m\boldsymbol{g} + \boldsymbol{F} \tag{5.84}$$

$$\dot{\boldsymbol{L}}_C = \sum_{i=1}^{n_b} \dot{\boldsymbol{L}}_{C,i} = \boldsymbol{T} + \boldsymbol{r}_{CF} \times \boldsymbol{F} \tag{5.85}$$

Here, m denotes the total mass, $\boldsymbol{p}$ the linear momentum, $\boldsymbol{L}_{C,i}$ the angular momentum of the i-th body about the system's CoM, $\boldsymbol{g}$ the gravitational acceleration, n_b the number of bodies in the MBS, $\boldsymbol{F}$ the contact force, $\boldsymbol{T}$ the contact moment, $\ddot{\boldsymbol{r}}_C$ the acceleration of the CoM and $\boldsymbol{r}_{CF}$ the vector from the CoM to the point where the contact force $\boldsymbol{F}$ acts. In the preceding equations, all quantities are evaluated in an inertial coordinate frame. Note that the terms for a moving reference point in the angular momentum theorem (5.85) cancel, since the robot's CoM is used as a reference point.

The absolve rate of change of angular momentum can be rewritten as:

$$\dot{\boldsymbol{L}}_C = \boldsymbol{J}_L \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_L \dot{\boldsymbol{q}}. \tag{5.86}$$

The matrix $\boldsymbol{J}_L = \frac{\partial \boldsymbol{L}_C}{\partial \dot{\boldsymbol{q}}}$ is the Jacobian of the system's angular momentum. Substituting (5.86) and (5.84) into (5.85) and choosing the foot reference point as the origin, that is, $\boldsymbol{r}_F = \boldsymbol{0} \Rightarrow \boldsymbol{r}_{CF} = -\boldsymbol{r}_C$, we obtain:

$$\boldsymbol{J}_L \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_L \dot{\boldsymbol{q}} - m\boldsymbol{r}_C \times \boldsymbol{g} + m\boldsymbol{r}_C \times \ddot{\boldsymbol{r}}_C - \boldsymbol{T} = \boldsymbol{0}. \tag{5.87}$$

Taking the x and y components[16] of (5.87), factoring out the unknown horizontal components of the CoM-acceleration $\ddot{x}_C$ and $\ddot{y}_C$, assuming that $\boldsymbol{g}$ acts in the negative $\boldsymbol{z}$-direction and rearranging, we obtain:

$$\begin{aligned} &\begin{pmatrix} \boldsymbol{J}_{L,x} \\ \boldsymbol{J}_{L,y} \end{pmatrix} \ddot{\boldsymbol{q}} + \begin{bmatrix} 0 & -mz_C \\ mz_C & 0 \end{bmatrix} \begin{pmatrix} \ddot{x}_C \\ \ddot{y}_C \end{pmatrix} \\ &+ \begin{pmatrix} \dot{\boldsymbol{J}}_{L,x} \\ \dot{\boldsymbol{J}}_{L,y} \end{pmatrix} \dot{\boldsymbol{q}} + \begin{pmatrix} my_C(g + \ddot{z}_C) - T_x \\ -mx_C(\ddot{z}_C + g) - T_y \end{pmatrix} = \boldsymbol{0}. \end{aligned} \tag{5.88}$$

Here, $\boldsymbol{J}_{L,x}$ and $\boldsymbol{J}_{L,y}$ are the first and second row of $\boldsymbol{J}_L$, respectively. This can be written more compactly as

$$\boldsymbol{A}\ddot{\boldsymbol{q}} + \boldsymbol{B}\ddot{\boldsymbol{w}}_u + \boldsymbol{c} = \boldsymbol{0}, \tag{5.89}$$

where the unknown CoM accelerations are denoted by $\ddot{\boldsymbol{w}}_u^T = \begin{pmatrix} \ddot{x}_C & \ddot{y}_C \end{pmatrix}$.

In order to determine the generalized accelerations, we split the full set of task-space trajectories into known ($\boldsymbol{w}_k$) and unknown components ($\boldsymbol{w}_u$) using appropriate matrices:

$$\boldsymbol{w} = \boldsymbol{T}_{wk}\boldsymbol{w}_k + \boldsymbol{T}_{wu}\boldsymbol{w}_u. \tag{5.90}$$

16. The x-axis as assumed to point forward, the z-axis upward and the y-axis to the left.

Note that $\boldsymbol{T}_{wk}$ and $\boldsymbol{T}_{wu}$ are not restricted to binary selection matrices. Combining (5.18), (5.89) and (5.90) yields:

$$\begin{aligned} & \boldsymbol{R}\ddot{\boldsymbol{q}} + \boldsymbol{r} = \boldsymbol{0}. \\ \text{With:} \quad & \boldsymbol{R} = \boldsymbol{J}_w + \boldsymbol{T}_{wu}\boldsymbol{B}^{-1}\boldsymbol{A}, \\ & \boldsymbol{r} = \dot{\boldsymbol{J}}_w\dot{\boldsymbol{q}} - \boldsymbol{T}_{wk}\ddot{\boldsymbol{w}}_k + \boldsymbol{T}_{wu}\boldsymbol{B}^{-1}\boldsymbol{c}. \end{aligned} \tag{5.91}$$

In the non-redundant case, $\ddot{\boldsymbol{q}}$ can be determined directly by solving (5.91). In the redundant case, the typical methods of nullspace optimization can be applied to minimize accelerations, avoid joint angle limits and collisions. Note that terms in $\boldsymbol{R}$ originating from the angular momentum Jacobian and from the task-space Jacobian are automatically scaled to similar values by premultiplication with $\boldsymbol{B}^{-1}$.

The General Case

The same basic approach applied to walking pattern generation above can be generalized to arbitrary tasks with a hybrid task description consisting of both position and force variables. The method is not restricted to humanoids and may be applied to any system governed by the MBS equations

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h} = \boldsymbol{W}_\lambda\boldsymbol{\lambda}, \tag{5.92}$$

where $\boldsymbol{\lambda}$ denotes the impressed actuator and contact forces and $\boldsymbol{W}_\lambda$ the corresponding (transposed) Jacobians used for projecting $\boldsymbol{\lambda}$ into the space of generalized coordinates. In the case of a fully actuated manipulator, $\boldsymbol{\lambda}$ is equal to the joint torques. For humanoid robots, $\boldsymbol{\lambda}$ contains both actuator torques and the total contact wrench.[17]

We assume that the task is specified using sub-vectors of $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ and that $\boldsymbol{\lambda}$ is partitioned into known (index k) and unknown (index u) components:

$$\boldsymbol{\lambda} = \boldsymbol{T}_{\lambda k}\boldsymbol{\lambda}_k + \boldsymbol{T}_{\lambda u}\boldsymbol{\lambda}_u. \tag{5.93}$$

As with the partitioning of the task variables $\boldsymbol{w}$ in (5.90), the $\boldsymbol{T}_{\lambda j}$ are suitably defined matrices for assembling the known and unknown sub-vectors of $\boldsymbol{\lambda}$.

Substituting (5.93) into (5.92) and (5.90) into (5.18) yields:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h} = \boldsymbol{W}_\lambda\left(\boldsymbol{T}_{\lambda k}\boldsymbol{\lambda}_k + \boldsymbol{T}_{\lambda u}\boldsymbol{\lambda}_u\right), \tag{5.94}$$

$$\boldsymbol{J}_w\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_w\dot{\boldsymbol{q}} = \left(\boldsymbol{T}_{wk}\ddot{\boldsymbol{w}}_k + \boldsymbol{T}_{wu}\ddot{\boldsymbol{w}}_u\right). \tag{5.95}$$

Solving (5.94) for $\ddot{\boldsymbol{q}}$, substituting the result into (5.95) and rearranging yields a linear

17. In the overactuated case, or when both legs of a biped are in contact, a force distribution must be chosen to obtain a unique solution.

equation in the unknown variables $\boldsymbol{\lambda}_u, \ddot{\boldsymbol{w}}_u$:

$$\begin{aligned} \boldsymbol{G}\begin{pmatrix}\boldsymbol{\lambda}_u \\ \ddot{\boldsymbol{w}}_u\end{pmatrix} + \boldsymbol{f} &= \boldsymbol{0}, \\ \boldsymbol{G} &= \left(\boldsymbol{J}_w \boldsymbol{M}^{-1} \boldsymbol{W}_\lambda \boldsymbol{T}_{\lambda u}\ ,\ \ -\boldsymbol{T}_{wu}\right), \\ \boldsymbol{f} &= \boldsymbol{J}_w \boldsymbol{M}^{-1} \boldsymbol{W}_\lambda \boldsymbol{T}_{\lambda k} \boldsymbol{\lambda}_k - \boldsymbol{J}_w \boldsymbol{M}^{-1} \boldsymbol{h} + \dot{\boldsymbol{J}}_w \dot{\boldsymbol{q}} - \boldsymbol{T}_{wk} \ddot{\boldsymbol{w}}_k. \end{aligned} \tag{5.96}$$

The joint accelerations are determined by solving the above equation and $\boldsymbol{q}(t)$ is then calculated by numerical integration. In the case of redundant systems, well-known redundancy resolution methods may be applied (see previous sections). Appropriate weighting matrices should be chosen to take the different physical meaning of $\boldsymbol{\lambda}_u$ and $\ddot{\boldsymbol{w}}_u$ into account.

The approach proposed above is equivalent to determining the trajectory resulting from the impressed forces $\boldsymbol{\lambda}_k$ by simulating the forward dynamics of the MBS (5.92) with rheonomic constraints according to the kinematic part of the task (see also section 3.4). Therefore, the proposed approach consists of designing a constrained mechanical system, simulating it online and then forcing the physical system to track the simulated one using an adequate controller.

5.5.3 Walking Pattern Generation Based on Inverse Kineto-Dynamics

This section presents an application of the ideas outlined above to the walking pattern generation problem. The method was implemented in the walking control system used for the robot Lola. Figure 5.20 shows an overview of the walking control system. Without activated IKD, the block "Trajectory Planning" outputs a full set of task-space and contact force trajectories, including a CoM reference calculated with the collocation method (Buschmann et al. 2007). With the IKD method, only a subset $\boldsymbol{w}_k$ of the original task-space without the horizontal CoM coordinates is used (see section 5.5.2).

Figure 5.21 shows CoM trajectories in the lateral plane calculated using the IKD method. The contact moments were taken from the solution for the simplified model using the collocation method (Buschmann et al. 2007) and are exactly tracked by the generated joint trajectories. The original CoM planning method is therefore still required for generating a suitable contact moment reference. The results in figure 5.21 show that for a good choice of the relative leg mass $\frac{m_l}{m}$ in the three point model (for details see Buschmann et al. 2007; Buschmann 2010) the solution is close to the reference trajectory obtained from the collocation method. For poor choices of $\frac{m_l}{m}$ the trajectory quickly diverges. Even for an optimal choice the CoM will eventually diverge, making the approach unusable in its simplest implementation.

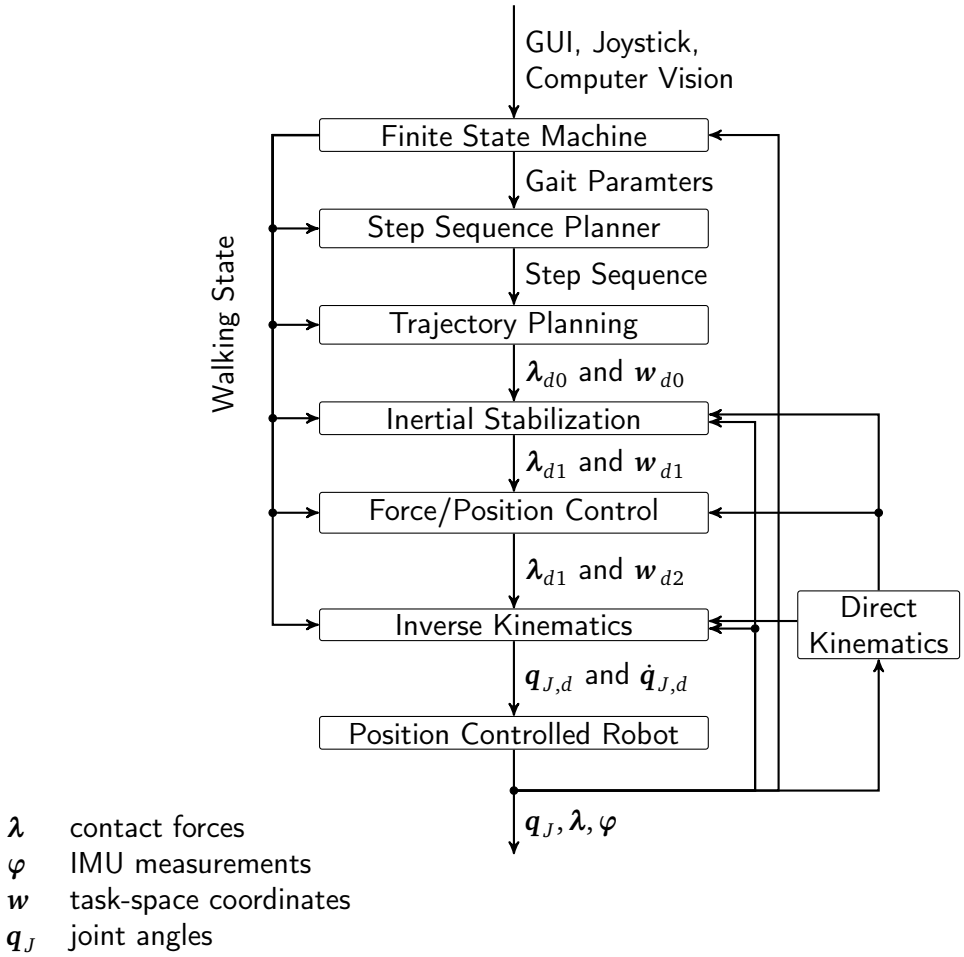


Figure 5.20: Overview of Lola's basic walking control system. Reference values are indicated by an index dn, where n identifies modified values. The "inertial stabilization" modifies contact force trajectories to stabilize the robot, as well as foot trajectories to assure proper landing of the swing foot. task-space trajectories are modified by the "position/force" control algorithm, in order to track a subset of contract forces and moments. The modified task space trajectories are then mapped to joint space by the inverse kinematics algorithm. For details, refer to (Buschmann et al. 2009; Buschmann 2010; Buschmann et al. 2011).

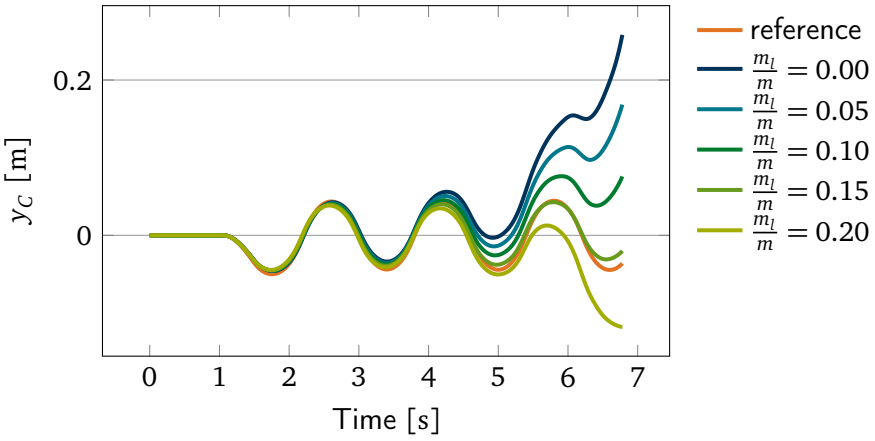


Figure 5.21: CoM trajectories calculated with the IKD method without stabilization. The plot shows results for contact forces corresponding to the solutions for the three point mass model using the collocation method (Buschmann et al. 2007) with different ratios of leg mass m_l to total mass m. The reference trajectory is the solution for the simplified model.

Stabilization via Planning

A closer analysis shows that the divergence of the CoM can be approximately modeled using the simplified robot model, a fact used by the "error compensation" methods mentioned above (Takenaka et al. 2009a; Kajita et al. 2003b). This motivates a stabilization method based on the planning algorithms for the simplified model.

The collocation method generates a CoM trajectory by solving a BVP starting at the current position and velocity of the CoM. In the original publication (Buschmann et al. 2007) we chose the position and velocity given by a periodic reference step as further boundary conditions. Here, the "divergent component" of the CoM proposed by Takenaka et al. (2009a) is used instead. In order to be able to satisfy more than two boundary conditions, the method modifies the contact force trajectory using shape functions. These shape functions are designed to only allow contact force modifications during periods when the reference is far from the maximum and minimum admissible force, for example, during double support.

We can therefore stabilize the CoM planning system simply by taking the output of the IKD method as initial value for the planning system based on the simplified model.[18] This is equivalent to a predictive control of the final CoM position and velocity. Figure 5.22 shows a resulting trajectory in the lateral plane obtained by this method together with the corresponding CoP trajectory and the maximum and minimum

18. The stabilizing effect follows directly from the fact that the "divergent component" at the end of the step is equal to that of the next periodic (and therefore stable) reference. Obviously stability is conditioned on a sufficiently small difference between the three mass model and the MBS model.

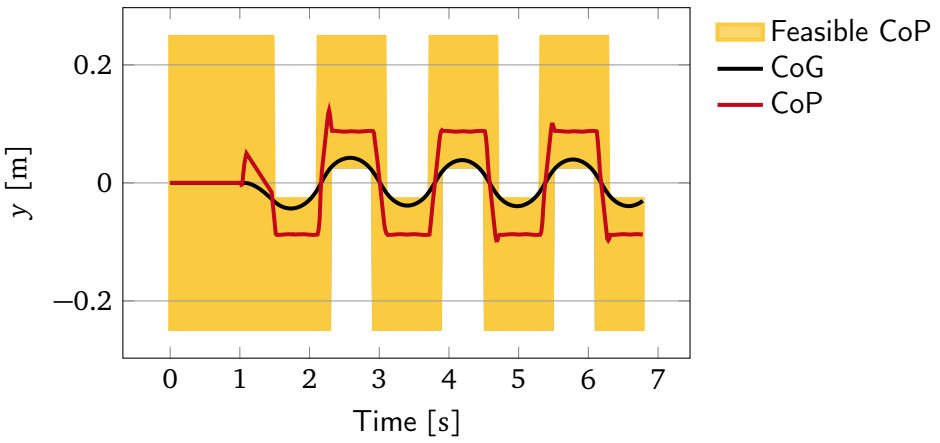


Figure 5.22: Lateral CoM position with stabilization via feedback to the planning system.

admissible CoP values. The CoP stays within the permissible bounds and is tracked exactly by the generated joint trajectories.

Stabilization via planning enables exact tracking of the modified reference forces and allows us to choose the time when the forces should be modified. The ability to choose when to modify contact forces is a major strength of the method — we can use the large contact area during double support and are not forced to move the CoP close to the edge of the foot during single support. In addition, stabilization via planning explicitly takes long term stability into account, since the contact moment trajectory is modified in such a way as to ensure that the divergent component of the CoM becomes equal to that of the ideal and stable reference generated by the collocation method.

The drawback is that feedback from the IKD output to the planner is only provided at the step frequency of the current walking pattern and that numerical drift is not compensated. For slow steps this can lead to strong divergence of the CoM from the periodic reference, which in turn requires large modifications of the contact moment for stabilization. Also, the predictive stabilization is based on the reduced order model, which introduces a modeling error in the control loop.

Stabilization via the Contact Moment

An alternative for stabilization is to locally modify the desired contact moment by a small amount $\Delta \boldsymbol{T}_{\text{cont}}$ in order to avoid large differences between the CoM generated by the IKD and the CoM output by the collocation method. This can be achieved by a PD control law, augmented by a saturation function sat$(\cdot)$ for avoiding unfeasible

contact forces:

$$\begin{aligned}\Delta \boldsymbol{T}_{\text{cont}} = \text{sat}\big(\boldsymbol{B}[&K_{\text{p,CoG}}(\boldsymbol{w}_{u,\text{plan}} - \boldsymbol{w}_{u,\text{IKD}}) \\ &+ K_{\text{d,CoG}}(\dot{\boldsymbol{w}}_{u,\text{plan}} - \dot{\boldsymbol{w}}_{u,\text{IKD}})]\big).\end{aligned} \tag{5.97}$$

Small values are chosen for the gain matrices $\boldsymbol{K}_{\text{p,CoG}}, \boldsymbol{K}_{\text{d,CoG}}$ in order to avoid large modifications of the contact moments. The matrix $\boldsymbol{B}$ scales the control output, see (5.91). For large gains, it is possible to closely track the reference CoM trajectory, essentially turning the IKD into an acceleration level IK algorithm. The difference is that we prescribe the contact force and can guarantee that it will remain within the feasible set.[19]

Figure 5.23 shows the ideal contact moment T_x and the resulting contact moments calculated by inverse dynamics when the system is stabilized by modifying the contact moments. The original reference moment is perfectly tracked without feedback ($k = 0$), but the CoM eventually becomes unstable. For a low but non-zero gain of $k = 10^{-3}$ the system is stabilized, but the resulting moment shows small oscillations about the ideal reference. As k is increased, the deviations from the ideal reference increase and the resulting CoM trajectory approaches the one calculated by the collocation method.

Hybrid Stabilization

The differences between the ideal reference moment and the resulting contact moment for the calculated trajectory can be further minimized by combining stabilization via the contact moment and the trajectory planner. Figure 5.24 shows the ideal contact moment T_x and the trajectories obtained for different values of k. The system is stable for all values of k. For $k = 0$, the reference is perfectly tracked and the planner modifies the reference using a trapezoidal shape function during double support. The modifications are clearly visible during the double support phases, which are approximately indicated by yellow rectangles. Larger values of k reduce the modifications during double support at the cost of larger tracking errors. For the walking experiments shown here, we used $k = 2 \times 10^{-4}$, which leads to very small tracking errors and equally small modifications during double support.

Summing up, the hybrid method combines two components to ensure a stable CoM reference: (1) fast feedback by locally modifying the contact moment and (2) slow feedback via the CoM-planner. The first component compensates small errors

19. Stability of the IKD with PD control is not guaranteed. This is due to the nonlinear coupling between $\boldsymbol{w}_k$ and $\boldsymbol{w}_u$ in $\dot{\boldsymbol{L}}_C$ — it should even be possible to construct mass distributions and task-spaces where the control law is destabilizing. However, if we make the reasonable assumption that (5.88) is dominated by the CoM-dynamics ($|\dot{\boldsymbol{L}}_C|$ is sufficiently small), (5.97) is reduced to a PD-type control of the inverted pendulum in the limiting case. For perfect position tracking ($z_C = z_{C,d}$) and $\dot{\boldsymbol{L}}_C = \boldsymbol{0}$, the error dynamics are reduced to $\Delta\ddot{\boldsymbol{w}}_u + \boldsymbol{K}_{\text{d,CoG}}\Delta\dot{\boldsymbol{w}}_u + \left(\boldsymbol{K}_{\text{p,CoG}} - \frac{(g+\ddot{z})}{z}\right)\Delta\boldsymbol{w}_u = \boldsymbol{0}$. In this simplified case we can expect stability for $\boldsymbol{K}_{\text{d,CoG}} > 0, \boldsymbol{K}_{\text{p,CoG}} > \frac{(g+\ddot{z})}{z}$. This is confirmed by the numerical results (see figure 5.23).

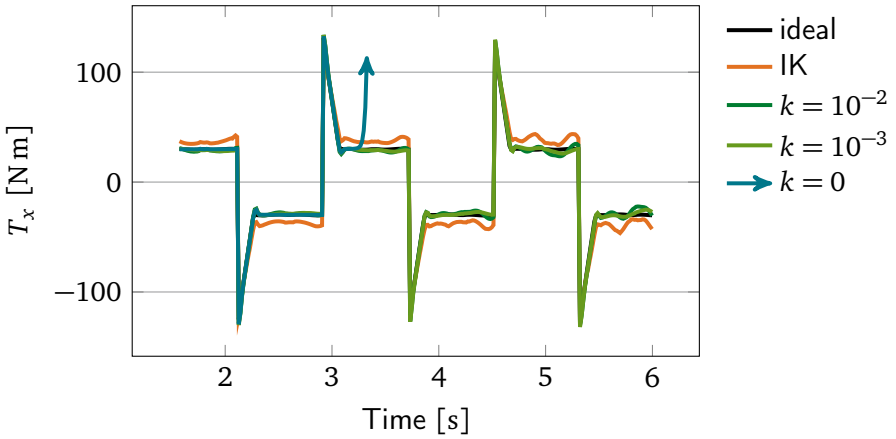


Figure 5.23: Total resulting contact moment T_x summed up at the stance leg (calculated by inverse dynamics), when the CoM is stabilized via local feedback to the reference contact moment. The parameter k represents the overall feedback gain. It is multiplied with the drift compensation gains (see section 5.5.3) to obtain $K_{\mathrm{p,cog}}, K_{\mathrm{d,cog}}$. The effective gains are: $K_{\mathrm{p,cog}} = 26.7$, $K_{\mathrm{d,cog}} = 8.8$ for $k = 10^{-3}$ and $K_{\mathrm{p,cog}} = 266.7$, $K_{\mathrm{d,cog}} = 27.8$ for $k = 10^{-2}$. The configuration with $k = 0$ is unstable. Note that instability occurs at a different time than in figure 5.21, since the walking sequence is different. The divergence of T_x for $k = 0$ is due to joint limit avoidance triggered by a diverging CoM.

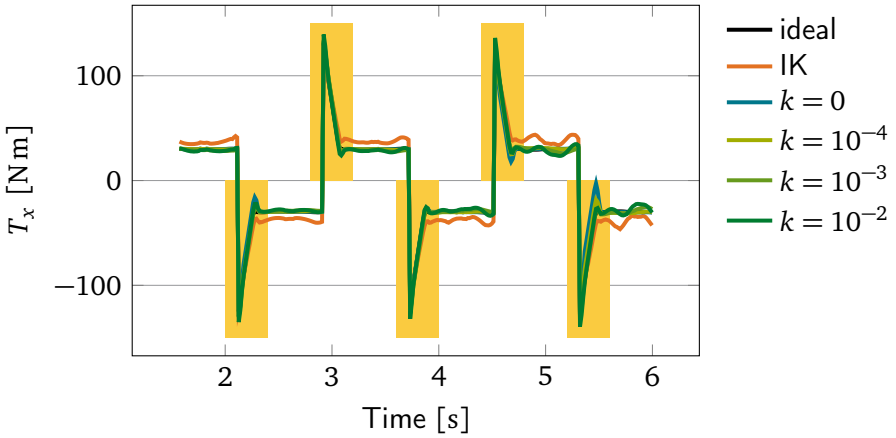


Figure 5.24: The total resulting contact moment T_x summed up at the stance leg (calculated by inverse dynamics), when the CoM is stabilized via local feedback and predictive control (cf. figure 5.23 for only local feedback; parametrization same as Figure 5.23). Double support periods with CoP modifications via planning approximately indicated by yellow rectangles.

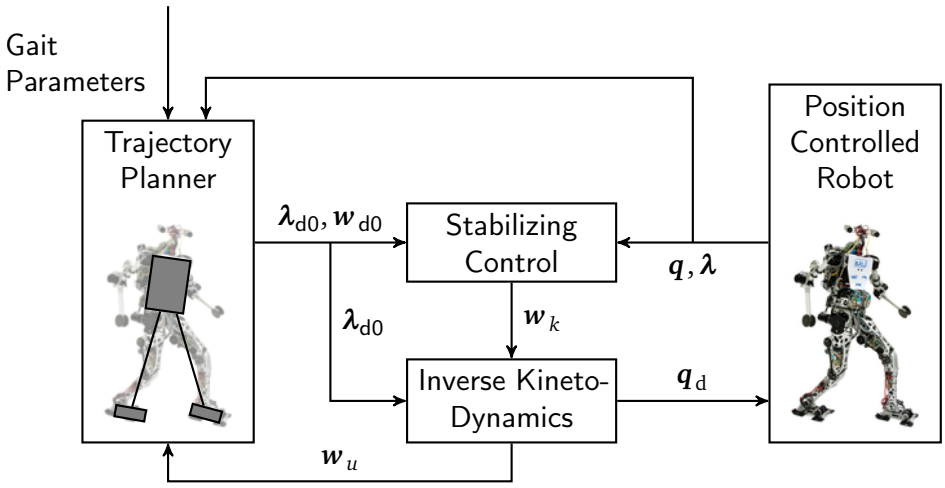


Figure 5.25: Integration of Inverse Kineto-Dynamics into the walking control system.

such as drift due to numerical integration. The second component ensures long term stability by predicting the CoM trajectory for the next step and modifying the contact force reference accordingly. While stability can be ensured by both basic stabilization methods, the hybrid approach minimizes the difference between ideal and modified force reference and moves most of the modification into the double support phase.

Comparison With Dynamics Error Compensation

We have compared the proposed approach with a combination of our collocation-based planner with additional dynamics error compensation. We have implemented the feed-forward compensation method described by Takenaka et al. (2009b, figure 4). The method works by first calculating the contact moment resulting from the current CoM-trajectory by inverse dynamics for a full MBS model. The difference from the original reference is used as input for an inverted pendulum model. The output is then added to the CoM trajectory. To prevent the pendulum from diverging a controller is added that stabilizes the pendulum by modifying the contact moment. Figure 5.26 illustrates the method. We use a real PD controller with the continuous-time transfer function

$$K_R \frac{T_R s + 1}{T_d s + 1} \tag{5.98}$$

to stabilize the pendulum. Here, s denotes the Laplace variable. We chose the control parameters $T_d = 0.02$, $T_R = 0.2$, $K_R = 3200$. The results shown here were obtained with a discrete time implementation of (5.98) with a sampling period of 1.5 ms.

Figure 5.27 shows the contact moments T_x and T_y calculated by inverse dynamics for three different planning methods: (1) the three point mass model, (2) the three

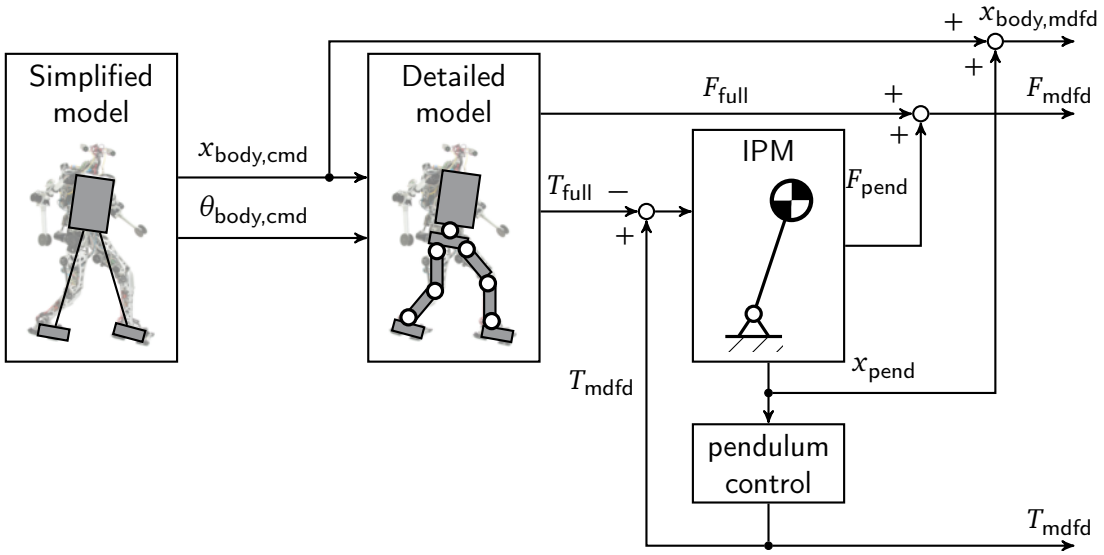


Figure 5.26: Dynamics error compensation method proposed by Takenaka et al. Real-time trajectory planning is based on simplified dynamics such as the three point mass model. Takenaka also adds a reaction wheel (angle Θ) to model torso rotation dynamics. The error between intended contact moment and that obtained by inverse dynamics from a full model (T_{full}) is approximately compensated by a feedback system based on an inverted pendulum model (IPM). The modified quantities used as reference for walking control are indicated by the index mdfd. Figure modified from (Takenaka et al. 2009b, figure 4).

point mass model with error compensation and (3) IKD with hybrid stabilization. The overall deviation is the smallest for the proposed IKD-method. For T_x, a trapezoidal modification during the double support phase (approximately 6.1 s…6.3 s) can be seen. This illustrates an advantage of the IKD-based trajectory generation. The dynamics error compensation approach is local, so we cannot choose when to perform the modification of the contact forces that are necessary for stabilizing the CoM. The IKD-based approach, on the other hand, allows us to perform most of the modification at a time during the gait cycle when the CoP is far from the edges of the support polygon. Accordingly, the difference during the single support phase can be reduced significantly.

Also, we can guarantee that the resulting CoP calculated by inverse dynamics will strictly stay within the support polygon. Stabilization via local feedback modifies the contact force, but we can saturate it according to the current support polygon. Since the modified contact moment is used as input to the IKD, the resulting contact moment is exactly equal to the input. We can also saturate the output from the dynamics error compensation. However, since the compensation works with an approximate model, the resulting contact moments cannot be guaranteed to stay strictly within a prescribed range. Finally, since we can exactly track a chosen contact moment, we know the

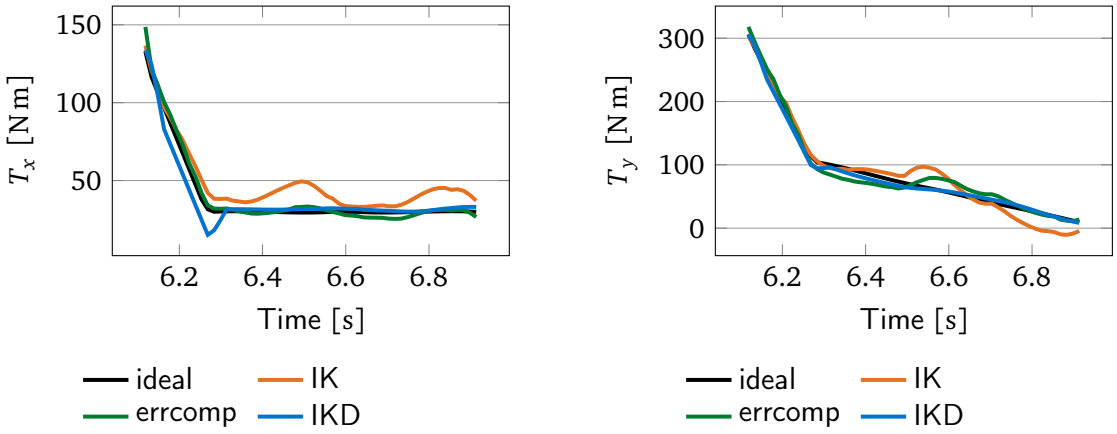


Figure 5.27: Comparison of the ideal contact moments and the moment from inverse dynamics using the collocation method with the three point mass model (orange), using additional model error compensation (green) and using the IKD method with hybrid stabilization (blue, $k = 10^{-4}$).

moment and the resulting CoP without calculating the inverse dynamics.

Implementation Issues

In the practical implementation a drift compensation law similar to (5.19) must be added to the IKD equation (5.91) to stabilize tracking of the task-space trajectories. Drift compensation is only applied to the planned task-space components $\boldsymbol{w}_k$, however, since applying it to $\boldsymbol{w}_u$ would prevent perfect tracking of the desired forces $\boldsymbol{\lambda}_k$. For the results shown here the drift compensation gains were chosen as $K_P = 40/\Delta t, K_d = 2\sqrt{K_p}0.85$ (i.e., a slightly underdamped configuration). For a sampling period of $\Delta t = 1.5\,\text{ms}$ the numerical values are $K_p = 2.7 \times 10^4$ and $K_d = 2.8 \times 10^2$.

All kinematic and kinetic quantities are calculated using the recursive analytic approach described in section 2.3.2. To reduce the computational cost, we use the gradient $\frac{\partial \dot{\boldsymbol{w}}}{\partial \boldsymbol{q}}$ instead of the time derivative of the Jacobian $\dot{\boldsymbol{J}}_w$. This is admissible, since $\dot{\boldsymbol{J}}_w \dot{\boldsymbol{q}} = \frac{\partial \dot{\boldsymbol{w}}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}$. Similarly, $\dot{\boldsymbol{J}}_L$ is replaced by $\frac{\partial \dot{\boldsymbol{L}}_C}{\partial \boldsymbol{q}}$.

On Lola's onboard PC with an Intel® Core2Duo™ Mobile CPU running at 2.33 GHz, all calculations for the IKD-based walking controller require an average of 823 µs. A walking controller based on velocity-level CLIK requires only 548 µs. However, the IKD-based program still is sufficiently fast for a control cycle of 1.5 ms from the sampling of new sensor data to the application of new motor commands.

5.5.4 Experimental Results

We have evaluated the method in walking experiments with the robot Lola (see figure 2.13). In the first set of experiments we used a walking controller based on online planning using the collocation method and balancing control based on hybrid position/force control (Buschmann et al. 2009). In the second set of experiments we replace the IK with redundancy resolution (Schwienbacher et al. 2011) with the IKD with hybrid stabilization and $k = 2 \times 10^{-4}$.

In all experiments, Lola walked with a step cycle of 0.8 s, starting with a step length of 40 cm and accelerating to 55 cm steps. The step length and walking speed was limited since angular momentum minimization via nullspace optimization was not yet implemented for the IKD-based controller.

For the moderate walking speed of approximately 2.5 $^{\text{km}}/_{\text{h}}$, both the original IK-based control and the proposed IKD-based control showed very similar performance. Figure 5.28 shows the upper body inclination error in the sagittal and lateral planes. Both IK and IKD work in a reference frame that rotates with the upper body and the resulting joint angles are tracked with high gain controllers. Accordingly, the upper body error $\Delta\varphi_x$ is roughly proportional to the CoM-tracking error in an inertial reference frame. The reference CoP, planned CoM and CoM from IKD for the same experiment are shown in figure 5.29. On the position level, the CoM-trajectories are quite similar, but the amplitude for the planned CoM is larger.

For walking speeds above 3 $^{\text{km}}/_{\text{h}}$ nullspace optimization methods (see section 5.3) significantly improve walking performance and stability. It is to be expected that the advantages of the IKD method will become more apparent during fast walking with additional nullspace optimization.

5.6 Predictive Inverse Kinematics

The algorithms discussed in the previous sections of this chapter take a local approach to solving the IK for redundant manipulators. Optimization-based approaches were used in the previous sections, but the optimization was only applied to the current time step. This section presents a predictive approach to redundancy resolution that optimizes the motion for a given task-space reference $\boldsymbol{w}_{\text{d}}$ over a finite horizon $t_b \dots t_e$. A direct formulation as a constrained optimal control problem in the unknown generalized velocities $\dot{\boldsymbol{q}}$ is

$$\begin{aligned} \Phi &= \int_{t_b}^{t_e} \varphi\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, t\right) \mathrm{d}t \rightarrow \min! \\ \boldsymbol{J}_w \dot{\boldsymbol{q}} &- \dot{\boldsymbol{w}}_{\text{d}} = \boldsymbol{0}. \end{aligned} \tag{5.99}$$

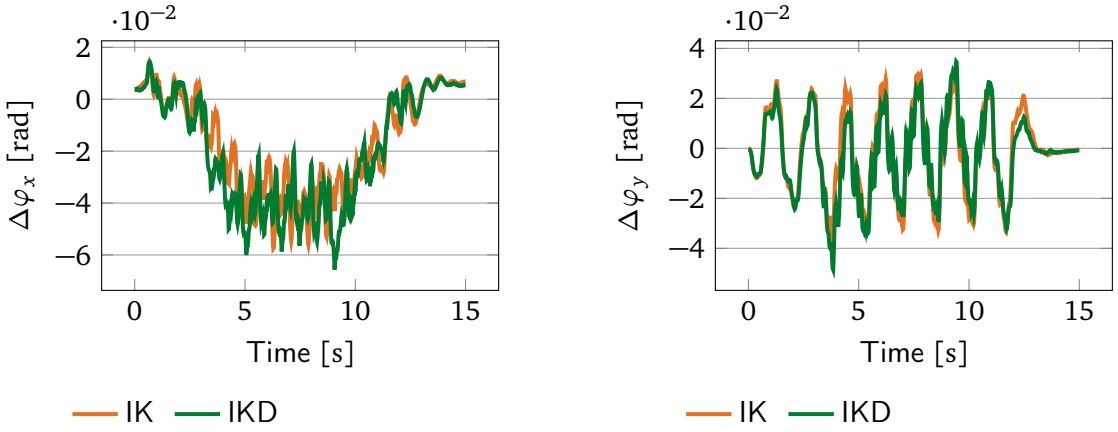


Figure 5.28: Measured upper body inclination $\Delta\varphi_x$ in sagittal (walking) direction (left) and $\Delta\varphi_y$ in lateral direction (right) with IKD (hybrid stabilization, $k = 2 \times 10^{-4}$) compared to the collocation method with inverse kinematics in a walking experiment with Lola.

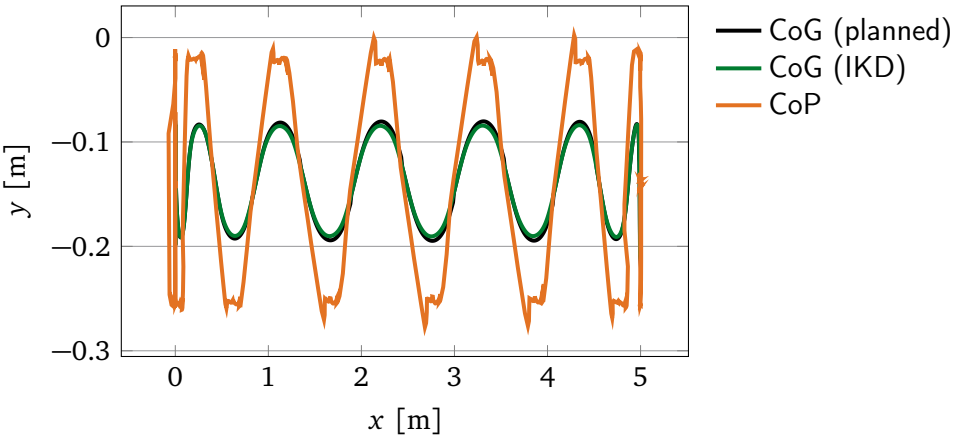


Figure 5.29: Planned CoM, CoM from IKD (hybrid stabilization, $k = 2 \times 10^{-4}$) and desired CoP for a walking experiment with Lola.

The equation assures that the task-space trajectories are followed and Φ describes the cost of a solution as the integral over a suitably chosen function $\varphi(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$. Typical choices for φ are the squared joint velocities and collision avoidance potentials.

This section describes an efficient method for solving this problem based on Pontryagin's Minimum Principle (PMP) and the (conjugate) gradient method. The results in this section are the outcome of joint work with Christoph Schütz, who implemented the algorithms. The results were previously published in a joint paper (Schuetz et al. 2014). The next section briefly discusses the background and related work and is followed by a presentation of the proposed method. A number of different minimal examples, as well as an application to the 9-DoF agricultural robot are also discussed.

5.6.1 Introduction and Related Work

The preceding sections reviewed classical methods for redundancy resolution, which often operate at the velocity level (see section 5.1), as well as an extension to hybrid task descriptions composed of force and position variables (section 5.5). These approaches have the major advantage of being very efficient computationally, enabling the on-line generation of reactive behaviors for robots with complex kinematics and 3D geometry (see section 5.3). Since the motion is only optimized individually for every time step at the level of joint velocities, however, the steepest descent within the nullspace resulting from the current configuration is taken. The effect this local change will have on the *future* configuration cannot be taken into account. This is the main drawback of the local approach, since a locally slightly better choice can lead to significantly greater costs in the future. At the same time, many of the future constraints are known at the current time, even for reactive systems. The constraints can still be satisfied in many cases, but at the cost of severe peaks in the joint velocities that are not necessary if a longer horizon is taken into account when controlling the given task (see section 5.6.2).

There are two major approaches to generating optimal trajectories: (1) Pontryagin's approach, which leads to a set of ODEs and an open-loop optimal trajectory and (2) dynamic programming (and variants), which lead to optimal feedback laws (see, e.g., Pontryagin 1962; Tolle 1971; Bellman 1954). A large number of different approaches and numerical algorithms have been proposed for solving the optimal control problem.

One possible solution is to directly use the PMP, which leads to necessary optimality conditions in the form of a BVP that can be solved using a number of different methods (e.g., collocation, finite differences, etc.; for some standard numerical methods, see, e.g., Quarteroni et al. 2007). In cases without terminal constraints, the BVP is decoupled and can easily be solved using the gradient method (see below; review, e.g., Polak 1973).

The direct application of dynamic programming is not possible for systems with more than a few DoFs since both the computational cost and memory requirements increase rapidly with the number of states (this is the "curse of dimensionality," see, e.g., Bellman and Lee 1984). However, a number of methods based on the principle of dynamic

programming have been developed that use local approximations of the value function[20] and the dynamic equations. The best-known method may be Differential Dynamic Programming (DDP), which uses a quadratic approximation of the value function and dynamics to derive a set of ODEs to be solved for iteratively improving the solution (Mayne 1966; Jacobson 1968c, 1968a; Jacobson and Mayne 1970). The method has quadratic convergence and maintains the forward and backward integrations characteristic for dynamic programming methods (cf. section 3.3), but does not require the solution of Bellman's partial differential equation. Other methods based on linear and quadratic approximations were developed by Jacobson (1968b).

Todorov and Li (2005) proposed a variant of DDP which they term iterative Linear Quadratic Regulator (iLQR) that uses a linear approximation of the dynamics and a simplified quadratic approximation of the value function.[21] The authors report significantly better performance than DDP for some problems (Li and Todorov 2004). A recent article describes an implementation that almost achieves real-time capability for moderately complex systems through the use of multiple processors for calculating finite difference approximations (Tassa et al. 2012).

One major advantage of DDP and iLQR over the gradient method is the faster rate of convergence. The gradient method can be significantly improved, however, by computing better directions of descent via the conjugate gradient method. An extension of this method to optimal control was given by Lasdon et al. (1967). The paper reports significantly better performance of conjugate gradient methods compared to the simple gradient method. Second order algorithms are reported to converge more rapidly than the conjugate gradient method at the cost of more complex calculations for every iteration. One reason for reducing the order in iLQR compared to DDP is that it is applied in an MPC or receding horizon approach. That is, the optimal trajectory is computed over a finite horizon, but the resulting control is only applied for the next time step. The procedure is then repeated frequently, using the current state as initial condition (for review, see Diehl et al. 2009). The current estimate should therefore always be close to the optimal trajectory, making a simplified approximation of the dynamics and value function sufficient. The same reasoning let us adopt the (conjugate) gradient method. Since we intend to use it in a receding horizon approach, we have chosen it over dynamic programming approaches, since we only require the optimal control, not the feedback law. Feedback is provided via the repeated calculation of the optimal open-loop trajectory. The gradient method, the linear method proposed by Jacobson (1968b) and iLQR (Todorov and Li 2005) are related, but distinct. All three methods have one ODE in common which is integrated backward in time, corresponding to the co-state or, equivalently, the gradient of the value function (for a discussion of the equivalence of value function gradient and co-state, see Vinter 1986).

20. The "value function" is a term widely used for the cost-to-go, i.e., the remainder of the integral cost starting at the current time step, and any terminal costs.

21. Essentially, the Hessian of the value function is not taken into account.

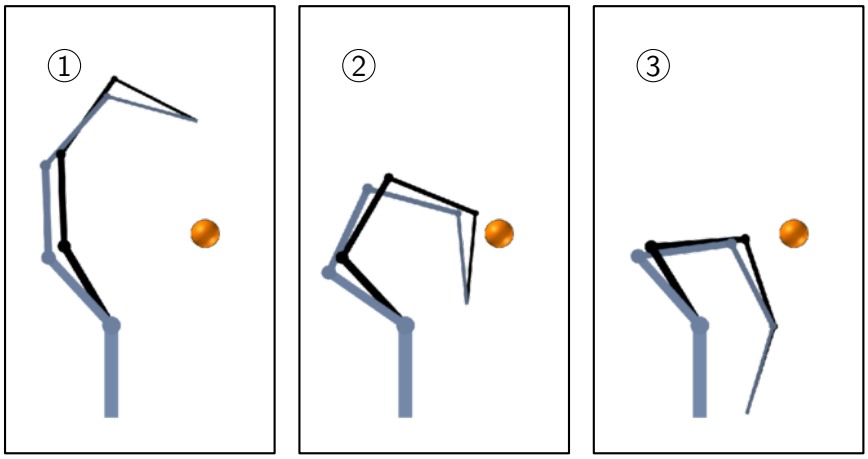


Figure 5.30: Collision avoidance in the nullspace of the robot. *Black robot:* Local Optimization. *Blue robot:* Global Optimization (adapted from Schuetz et al. 2014).

5.6.2 Motivational Example

A motivational example for the predictive approach to IK is given in figure 5.30. The planar 4-DoF robot performs a predefined Tool Center Point (TCP) movement on a straight line from top to bottom. The reference trajectory of the TCP does not collide with the obstacle (orange sphere). To demonstrate the effect of a longer control horizon, the sum of squared joint velocities (pseudoenergy) is chosen as a cost function for local redundancy optimization and the integral of the pseudoenergy as cost functional for global optimization. In both cases, collision avoidance is implemented using the penalty approach.

Without active collision avoidance, the robot would collide with the obstacle. The local nullspace collision avoidance scheme can prevent collisions, but it is only activated after the obstacle is close to the robot (see section 5.3). By this time, however, the local optimization has led to an unfavorable configuration relative to the obstacle, requiring very high joint velocities to avoid a collision. The globally optimal solution (blue robot) uses slightly higher velocities during the initial period to obtain a better configuration relative to the obstacle when passing it, significantly reducing peak joint speeds. This behavior can be seen in the the pseudoenergy shown in figure 5.31. The predictive approach with a planning horizon of 1 s performs slightly worse than the global solution, but much better than the local approach.

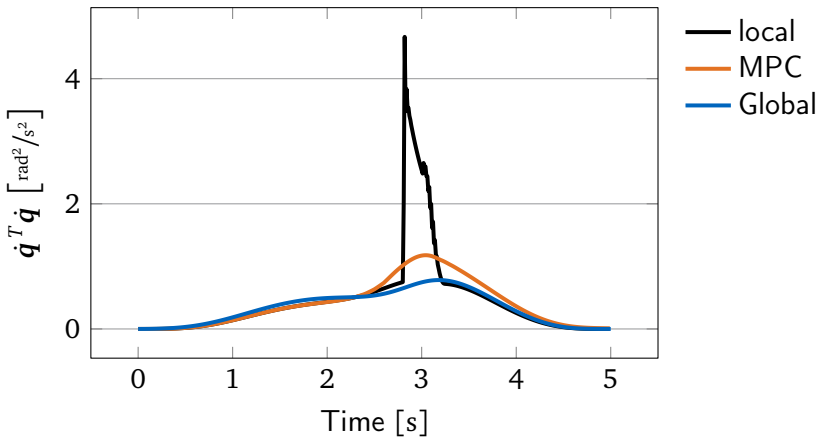


Figure 5.31: Pseudoenergy of the global, local and predictive (MPC) redundancy optimization methods. The relative weights for the optimization are: 1.0 velocity minimization, 1.0 for collision avoidance and 0.6 for joint limit avoidance (prediction horizon 1 s; redrawn from Schuetz et al. 2014).

5.6.3 Predictive Approach to Redundancy Resolution

Formulation of the Optimal Control Problem

Nakamura proposed a formulation of globally optimal redundancy resolution based on the general solution

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_w^{\#}\dot{\boldsymbol{w}}_{\mathrm{d}} + \boldsymbol{N}_w\boldsymbol{u} \tag{5.100}$$

of the underdetermined linear constraints imposed by the desired task space trajectory (Nakamura and Hanafusa 1987; Nakamura 1991). The newly introduced variable $\boldsymbol{u}$ parameterizes the nullspace velocity and is chosen as input for the optimal control problem:

$$\Phi(\boldsymbol{u},\boldsymbol{q}) = \int_{t_b}^{t_e} \varphi\left(\boldsymbol{q},\boldsymbol{u},t\right)\mathrm{d}t \rightarrow \min! \tag{5.101}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{f}\left(\boldsymbol{q},\boldsymbol{u},t\right) = \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}_{\mathrm{d}} + \boldsymbol{N}_w\boldsymbol{u} \tag{5.102}$$

$$\boldsymbol{w}(t_b) = \boldsymbol{w}_{d,b} \tag{5.103}$$

$$t \in \left[t_b, t_e\right]$$

The optimality conditions following from the PMP are

$$H(\boldsymbol{q},\boldsymbol{u},\boldsymbol{\lambda},t) = \varphi\left(\boldsymbol{q},\boldsymbol{u},t\right) + \boldsymbol{\lambda}^T\boldsymbol{f}\left(\boldsymbol{q},\boldsymbol{u},t\right) \tag{5.104}$$

$$\left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \dot{\boldsymbol{q}} \tag{5.105}$$

$$-\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T = \dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \varphi}{\partial \boldsymbol{q}}\right)^T - \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda} \tag{5.106}$$

$$\boldsymbol{w}(\boldsymbol{q}(t_b)) = \boldsymbol{w}_{d,b} \tag{5.107}$$

$$\boldsymbol{\lambda}(t_e) = \boldsymbol{0} \tag{5.108}$$

$$\boldsymbol{N}\boldsymbol{\lambda}(0) = \boldsymbol{0} \tag{5.109}$$

$$\boldsymbol{u}^\star = \underset{\boldsymbol{u}}{\operatorname{argmin}} H(\boldsymbol{q}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{u}, t), \tag{5.110}$$

where starred versions of the variables denote the optimal solution. The boundary conditions on $\boldsymbol{w}$ (5.107) are self-explanatory and those on $\boldsymbol{\lambda}$ (5.108) follow from the free final state and the fact that the problem has Lagrange form. The transversality condition (5.109) ensures that the adjoint variables lie within the normal cone to the tangent cone of the constraint set defined by $\boldsymbol{w}(\boldsymbol{q}(t_b)) = \boldsymbol{w}_{d,b}$. Requiring that the prescribed task $\boldsymbol{w}_{d,b}$ be satisfied at the beginning enables the calculation of a globally optimal solution, but also leads to a coupled two-point BVP due to the transversality condition.

If we intend to solve the problem continuously in the spirit of MPC as proposed above, we must ensure that $\boldsymbol{q}(t)$ is continuous, that is, $\boldsymbol{q}(t_b) = \boldsymbol{q}_b$. This modified optimal control problem leads to optimality criteria with *decoupled boundary conditions* for $\boldsymbol{\lambda}$ and $\boldsymbol{q}$:

$$H(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = \varphi\left(\boldsymbol{q}, \boldsymbol{u}, t\right) + \boldsymbol{\lambda}^T \boldsymbol{f}\left(\boldsymbol{q}, \boldsymbol{u}, t\right) \tag{5.111}$$

$$\left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \dot{\boldsymbol{q}} \tag{5.112}$$

$$-\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T = \dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \varphi}{\partial \boldsymbol{q}}\right)^T - \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda} \tag{5.113}$$

$$\boldsymbol{q}(t_b) = \boldsymbol{q}_b \tag{5.114}$$

$$\boldsymbol{\lambda}(t_e) = \boldsymbol{0} \tag{5.115}$$

$$\boldsymbol{u}^\star = \underset{\boldsymbol{u}}{\operatorname{argmin}} H(\boldsymbol{q}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{u}, t) \tag{5.116}$$

Conditions (5.112)-(5.115) allow us to calculate $\boldsymbol{\lambda}(t), \boldsymbol{q}(t)$ for any given $\boldsymbol{u}(t)$ by simply integrating (5.112) forward in time and (5.113) backward in time. We can therefore think of (5.101) as an unconstrained problem that only depends on the input $\boldsymbol{u}$:

$$\widehat{\Phi}(\boldsymbol{u}) = \int_{t_b}^{t_e} \varphi\left(\boldsymbol{q}(\boldsymbol{u}), \boldsymbol{u}, t\right) \mathrm{d}t \to \min! \tag{5.117}$$

Such problems are easily solved using the gradient method,[22] which is characterized by an iterative calculation of the unknowns according to the basic rule

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \alpha \frac{\partial \widehat{\Phi}}{\partial \boldsymbol{u}}^T, \tag{5.118}$$

where α is calculated from some step-size rule. The gradient of the reduced optimal control problem (5.117) happens to be the gradient of the Hamiltonian, leading to the update rule

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \alpha \frac{\partial H}{\partial \boldsymbol{u}}^T. \tag{5.119}$$

A derivation of this result is given by Gerdts (2011, chapter 8). The text also includes a comparison of the function space gradient method used here and the gradient method applied to a discretized optimal control problem. While the two methods are closely related, the function space approach often converges much more rapidly (Gerdts 2011, p. 410). A review of the mathematical and historical background of the method is given by Polak (1973).

Efficient Numerical Solution

While simple, the gradient method is also known for its relatively slow convergence. We therefore use the conjugate gradient method as described by Lasdon et al. (1967) to solve the optimal control problem (5.111)-(5.116), since it enables a computationally efficient solution of the nonlinear problem in real-time. Applying the basic algorithm leads to sequential forward integration of the system ODE (5.112) and backward integration of the adjoint variable ODE (5.113) to update the system for a new input $\boldsymbol{u}$. The computational scheme is shown in algorithm 11. Direct kinematics of the robot are calculated by the recursive approach described in section 2.4. The step size α can either be chosen as a sufficiently small constant or computed for every iteration using a step-size control algorithm, for example, the backtracking method (see, e.g., Bonnans et al. 1997). We could confirm the superior convergence of the conjugate gradient method compared to the steepest descent method in our applications (see above and Lasdon et al. 1967). Results for the motivational example given above are shown in figure 5.32.

Predictive Approach

Depending on the number of DoFs and the length of the trajectory, the global approach may not be solvable in real time. We therefore propose an approach which follows the idea of MPC. We solve the optimal control problem for the nullspace vector $\boldsymbol{u}$ at every time step t_k with a prediction horizon of $[t_k, t_k + T_{\text{horizon}}]$. The basic scheme is shown in algorithm 12.

The straightforward formulation of predictive inverse kinematics as an optimal

22. If $\boldsymbol{u}$ is constrained ($\boldsymbol{u} \in \mathscr{U}$), the projected gradient method can be used.

Algorithm 11: Conjugate Gradient Method for optimal control.

input : Initial guess for control: $\boldsymbol{u}^0$, interval $[t_b, t_e]$
output: Optimal control: $\boldsymbol{u}^\star$
$j \leftarrow 0$
$\boldsymbol{q}^0(t) = \int_{t_b}^{t} \dot{\boldsymbol{q}}(\boldsymbol{u}^0, \tau)\mathrm{d}\tau$ // $\boldsymbol{q}$ for $\boldsymbol{u}^0$ by forward integration
$\Phi^0 \leftarrow \int_{t_b}^{t_e} \varphi(\boldsymbol{q}^0, \boldsymbol{u}^0, \tau)\,\mathrm{d}\tau$ // Cost for initial trajectory
repeat
  $\boldsymbol{\lambda}^j(t) = \int_{t}^{t_e} \dot{\boldsymbol{\lambda}}(\boldsymbol{q}^j, \dot{\boldsymbol{q}}^j, \boldsymbol{u}^j, \tau)\mathrm{d}\tau$ // $\boldsymbol{\lambda}$ for $\boldsymbol{u}^j$ by backward integration
  $\boldsymbol{g}^j \leftarrow \frac{\partial H}{\partial \boldsymbol{u}}$ // gradient for $\boldsymbol{u}^j$
  if $j \neq 0$ **then**
    $$\beta^j \leftarrow \frac{\left\langle \boldsymbol{g}^j, \boldsymbol{g}^j \right\rangle}{\left\langle \boldsymbol{g}^{(j-1)}, \boldsymbol{g}^{(j-1)} \right\rangle}$$
    where $$\left\langle \boldsymbol{g}^j, \boldsymbol{g}^j \right\rangle = \int_{t_b}^{t_e} (\boldsymbol{g}^j(\tau))^T \boldsymbol{g}(\tau)^j \,\mathrm{d}\tau$$ // dot product
    $$\boldsymbol{s}^j \leftarrow -\boldsymbol{g}^j + \beta^j \boldsymbol{s}^{(j-1)}$$
  else
    $\boldsymbol{s}^j \leftarrow -\boldsymbol{g}^j$
  $\boldsymbol{u}^{(j+1)} \leftarrow \boldsymbol{u}^j + \alpha^j \boldsymbol{s}^j$ // improved control input
  $j \leftarrow j + 1$ // next iteration
  $\boldsymbol{q}^j(t) = \int_{t_b}^{t} \dot{\boldsymbol{q}}(\boldsymbol{u}^j, \tau)\mathrm{d}\tau$ // $\boldsymbol{q}$ for $\boldsymbol{u}^j$ by forward integration
  $\Phi^j \leftarrow \int_{t_b}^{t_e} \varphi(\boldsymbol{q}^j, \boldsymbol{u}^j, \tau)\,\mathrm{d}\tau$ // Cost for updated trajectory
until $\left(j > j_{max}\right) \vee \left(\frac{\Phi^j - \Phi^{j-1}}{\Phi^{j-1}} < \epsilon\right) \vee \left(\Phi^j > \Phi^{j-1}\right)$
return $\boldsymbol{u}^j$

Algorithm 12: Model Predictive Control

$t_{\text{start}} = t_b$, $t_e = t_n$, $\Delta t_{\text{mpc}} = t_{k+1} - t_k$
for $k = 0$ *to* n // for every control cycle
do
  $t_b \leftarrow t_k + \Delta t_{\text{mpc}}$ // lower bound for prediction interval
  $t_e \leftarrow t_k + T_{\text{horizon}}$ // upper bound for prediction interval
  $\boldsymbol{u}^0 \leftarrow \boldsymbol{u}^{*,k-1}$ // last result as initial guess
  solve optimal control problem for $\boldsymbol{u}^*$ // e.g., using algorithm 11
  apply $\boldsymbol{u}^*(t_{\text{start}})$ // Use result for next time step

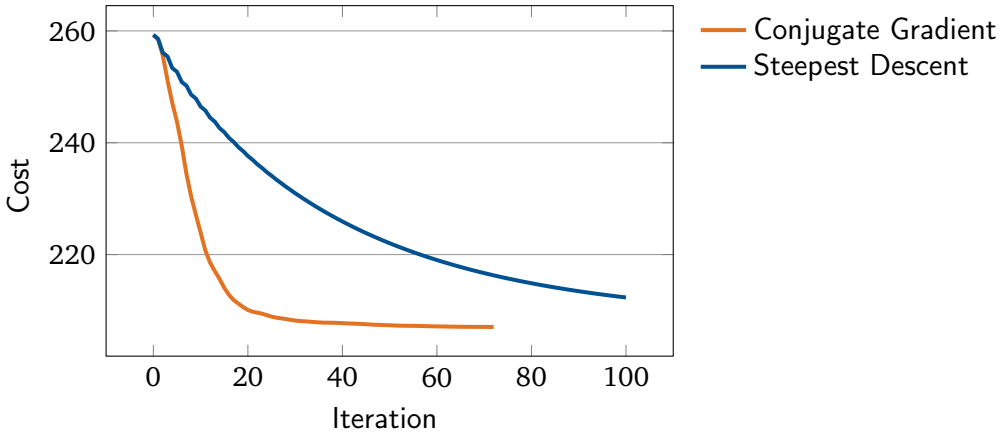


Figure 5.32: Comparison between the conjugate gradient and the steepest descent method for the planar 4-DoF manipulator (cf. figure 5.30; redrawn from Schuetz et al. 2014).

control problem (5.102) imposes constraints for the joint positions $\boldsymbol{q}_k$ at the start of the period, but not for the joint velocities $\dot{\boldsymbol{q}}_k$. Since the constraints and the time interval change at every time step, the resulting joint velocities at the first time step can change discontinuously from one sampling period to the next. This leads to a non-smooth joint trajectory for the predictive solution, as shown in figure 5.33.

One possible solution to this problem would be to formulate the task-space constraints at the acceleration level, which would shift the discontinuity to the joint accelerations. This, however, would require the computation of additional gradients (e.g., of $\boldsymbol{J}, \boldsymbol{J}^{\#}$), unacceptably increasing the computational cost. A less expensive approach is to use an augmented state space consisting of $\boldsymbol{q}$ and $\boldsymbol{u}$ and to use the derivative of $\boldsymbol{u}$ as the new input vector. The modified cost functional and augmented system dynamics are

$$\Phi = \int_{t_b}^{t_e} \varphi\left(\boldsymbol{q}, \hat{\boldsymbol{u}}, t\right) \mathrm{d}t \rightarrow \min!, \tag{5.120}$$

$$\boldsymbol{f}_{\mathrm{acc}}\left(\boldsymbol{q}, \hat{\boldsymbol{u}}, t\right) = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}_{\mathrm{d}} + \left(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J}\right)\boldsymbol{u} \\ \hat{\boldsymbol{u}} \end{bmatrix}. \tag{5.121}$$

The modified optimality conditions are:

$$\boldsymbol{\lambda}_{\mathrm{acc}}^T = \begin{bmatrix} \boldsymbol{\lambda}_1^T & \boldsymbol{\lambda}_2^T \end{bmatrix}, \tag{5.122}$$

$$H(\boldsymbol{q}, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}_{\mathrm{acc}}, t) = \varphi\left(\boldsymbol{q}, \hat{\boldsymbol{u}}, t\right) + \boldsymbol{\lambda}_1^T \boldsymbol{f}_1\left(\boldsymbol{q}, \hat{\boldsymbol{u}}, t\right) + \boldsymbol{\lambda}_2^T \boldsymbol{f}_2(\hat{\boldsymbol{u}}), \tag{5.123}$$

$$\left(\frac{\partial H}{\partial \boldsymbol{\lambda}_{\mathrm{acc}}}\right)^T = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{u}} \end{pmatrix}, \tag{5.124}$$

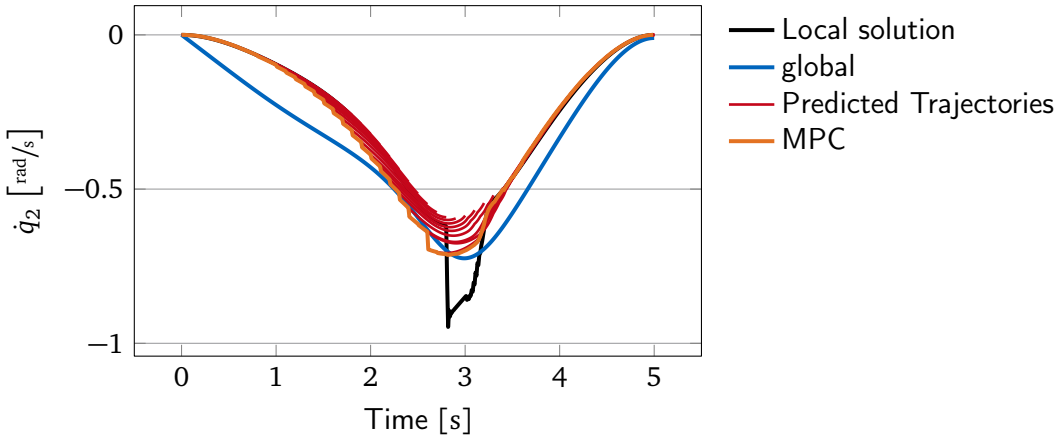


Figure 5.33: Solutions for the motivational example for different velocity-level approaches (cf. figure 5.30). The curves show the nonsmooth trajectory (second joint) for velocity-level MPC, the individual solutions of the optimal control problem (prediction horizon 1 s) for different time steps, the local solution and the global solution (cf. figure 5.30; for weights see figure 5.31; redrawn from Schuetz et al. 2014).

$$-\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T = \dot{\boldsymbol{\lambda}}_1 = -\left(\frac{\partial \varphi}{\partial \boldsymbol{q}}\right)^T - \left(\frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda}_1, \tag{5.125}$$

$$-\left(\frac{\partial H}{\partial \boldsymbol{u}}\right)^T = \dot{\boldsymbol{\lambda}}_2 = -\left(\frac{\partial \varphi}{\partial \boldsymbol{u}}\right)^T - \left(\frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\lambda}_1, \tag{5.126}$$

$$\boldsymbol{q}(t_b) = \boldsymbol{q}_b, \quad \boldsymbol{u}(t_b) = \boldsymbol{u}_b, \tag{5.127}$$

$$\boldsymbol{\lambda}_{\text{acc}}(t_e) = \boldsymbol{0}, \tag{5.128}$$

$$\hat{\boldsymbol{u}}^\star = \underset{\hat{\boldsymbol{u}}}{\operatorname{argmin}} H(\boldsymbol{q}^\star, \boldsymbol{\lambda}^\star_{\text{acc}}, \hat{\boldsymbol{u}}, t). \tag{5.129}$$

The augmented state space allows the introduction of the additional constraint $\boldsymbol{u}(t_b) = \boldsymbol{u}_b$, which assures continuous velocities, since the task-space component is continuous if $\dot{\boldsymbol{w}}_\text{d}$ is. The numerical solution is analogous to the optimization on velocity level (see above). Required gradients are listed appendix A.5. The smooth trajectories computed using this approach are shown in figure 5.34 for the motivational example.

5.6.4 Application to 9-DoF Manipulator

The approach has been evaluated for the 9-DoF agricultural manipulator (see figure 5.8; Baur et al. 2012). A similar setup to that described in section 5.3 is evaluated. In the general case, the cost functional is defined as the sum of n_c terms:

$$\Phi(\boldsymbol{q}, \boldsymbol{u}, t) = \int_{t_b}^{t_e} \sum_{i=1}^{n_c} \varphi_i(\boldsymbol{q}, \boldsymbol{u}, t) \mathrm{d}t \tag{5.130}$$

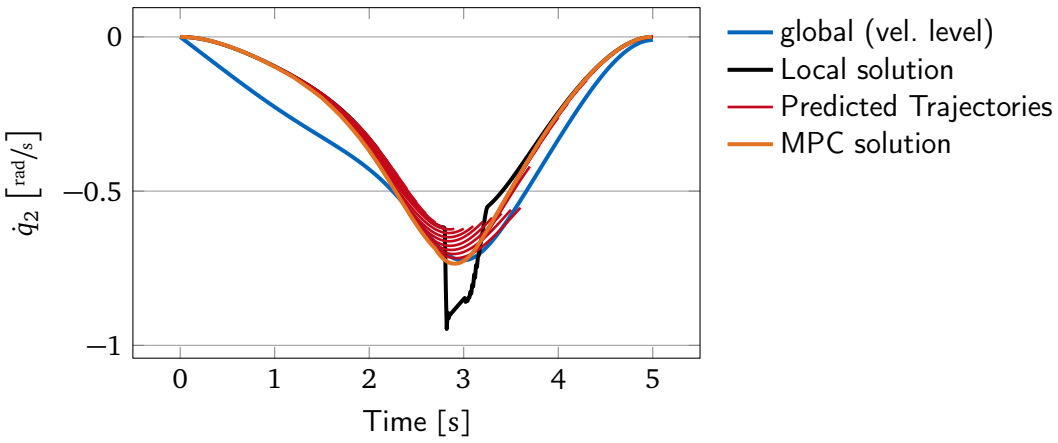


Figure 5.34: Smooth trajectory (second joint) for MPC solution with augmented state space (prediction horizon 1 s), individual trajectories of the MPC-solution and local solution. Global solution of velocity-level approach shown for comparisson (Schuetz et al. 2014, redrawn from).

For the example given below, three terms were used: (1) joint velocity minimization φ_v, (2) joint limit avoidance φ_{jl} and (3) collision avoidance φ_{ca}. The term φ_v is defined by

$$\varphi_v = \dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}}. \tag{5.131}$$

For φ_{jl}, the piece-wise function given in (5.58) is used and a simple cubic repulsive potential is used for φ_{ca}.

Results for the local solution, as well as the predictive approach with different lengths of the moving horizon T_{horizon} are shown in figure 5.35. The smaller horizon could be realized in the MATLAB/SIMULINK-based real-time control unit at a sampling rate of 10 ms and one MPC iteration per time step, while a larger horizon of 2 s currently requires up to seven sampling periods to solve. A small horizon of 0.15 s already slightly reduces velocity peaks, but a longer horizon leads to significant better results.

5.6.5 Conclusion

This section presents a novel approach to real-time predictive inverse kinematics for redundant manipulators. The optimal control problem is solved based on Pontryagin's Minimum Principle and the conjugate gradient method. A similar approach to global optimization was previously proposed by Nakamura and Hanafusa (1987). The contribution of the method proposed here is the receding horizon approach, which leads to a decoupled BVP, which enables an efficient solution in real-time. The potential of the approach is demonstrated both with a minimal 4-DoF example and a redundant 9-DoF agricultural manipulator. Currently, the implementation does not yet enable long

prediction horizons on the 9-DoF manipulator, but this should become possible with more efficient programming, the use of multiple CPUs and a larger sampling period for the predicted trajectory and more sophisticated time integration methods.

5.7 Chapter Summary

This chapter reviews classical approaches to solving the inverse kinematics problem for redundant and non-redundant robots. The framework first proposed by Liégeois (1977) is used to develop a system for reducing joint velocities, avoiding joint limits and potential self-collisions. An application of the approach to angular momentum minimization for bipedal robots shows the flexibility of the method.

While the nullspace approach is very powerful, there are situations in which the primary task-space trajectory must be modified to avoid collisions. A real-time task-space modification method is developed, which combines approximate heuristic re-planning with local potential field-based obstacle avoidance. The method is used in experiments with the biped Lola, allowing it to avoid different obstacles by modifying footstep locations and task-space trajectories in real-time.

An extension of differential inverse kinematics to force references (Inverse Kineto-Dynamics) enables real-time tracking of desired generalized forces, taking the full multibody model into account. The method is successfully applied to biped walking control.

Local potential field-based redundancy resolution methods are very powerful and computationally efficient techniques for real-time and reactive motion generation. Since they are local, however, changing configurations and obstacles can sometimes lead to large peaks in joint velocities. The methods are also theoretically prone to local minima. The predictive approach to inverse kinematics presented in this chapter solves some of these drawbacks. The integral cost is significantly reduced and local peaks can often be avoided by predictively modifying the robot's motion. The second drawback of local approaches, local minima, have not been an issue for the applications discussed in this chapter. For general manipulation tasks, however, local minima can quickly become a significant problem. Possible solutions are the combination with sampling based methods, which, however, are much more demanding computationally.

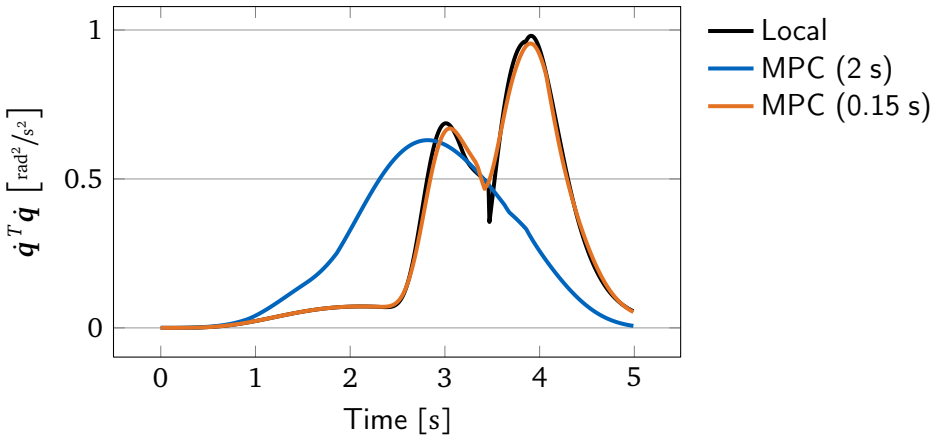


Figure 5.35: Pseudoenergy of revolute joints of the 9-DoF manipulator in a self-collision scenario. Results shown for local IK and MPC-approach with a moving horizon of 0.15 s and 2 s. The relative weights for the optimization are: 1.0 velocity minimization, 2.0 for collision avoidance and 0.6 for joint limit avoidance for local version; 1.0 velocity minimization, 5000.0 for collision avoidance and 80 for joint limit avoidance for MPC version (redrawn from Schuetz et al. 2014).

6 Conclusion and Directions for Future Work

This monograph summarizes recent research on efficient algorithms for redundant robots the author is involved in. Most of the work has been motivated by the problems posed by designing and controlling bipedal walking machines. The results, however, are applicable to a much wider range of systems. One of the core challenges is to design and implement efficient algorithms that are applicable to complex, redundant systems in dynamic environments.

The first chapters treat basic aspects of multibody kinematics and dynamics and present efficient algorithms suitable for robotic systems. A special emphasis is put on tree structures and structure exploiting methods. Detailed comparisons of standard and structure-exploiting versions of both EoM-oriented and recursive dynamics algorithms are presented. The extreme improvements provided by linear-time recursive algorithms over naive EoM-based implementations are confirmed by run-time measurements. Efficient, structure-exploiting implementations, however, often give comparable results for realistic system topologies. To apply the efficient recursive approach to general multibody systems, closed kinematic loops are usually cut and then handled by additional algebraic constraints. The chapter on dynamics introduces a new method for efficiently calculating the forward dynamics of systems with closed loops in minimal coordinates, without the introduction of algebraic constraints.

A detailed review of biped walking control is given, spanning different approaches to hardware and controller design, as well as current knowledge on human control of posture and balance. One area in which current systems are still lacking is generating complex movements of systems with many DoFs in 3D environments.

Since holistic optimal-control approaches are not suitable for real-time use due to the computational cost, a layered approach must be taken. Mapping a coarse high-level description to joint level motion is a core building block of such a system and addressed in detail in the chapter on optimal redundancy resolution and collision avoidance. Next to a review of standard methods, an extension of inverse kinematics problem to hybrid tasks involving generalized force trajectories is presented (“Inverse Kineto-Dynamics”). This extension is motivated by and well suited for generating dynamically consistent walking trajectories, but is not limited to this application. Other potential applications can be found in many robotic tasks, especially such involving contacts.

The tasks described above require efficient collision modeling and distance computation methods. An approach to this problem is presented, together with applications to a biped robot and a redundant manipulator. Collision avoidance in the nullspace of a given task, as well as collision avoidance by task-space trajectory modifications

are discussed. The methods are based on a local potential-field approach and work very well both for bipedal walking tasks and reaching tasks for redundant agricultural manipulators. A well-known drawback of this approach are local minima. This has not been a problem in the presented applications, but certainly is a potential source of system failure for manipulation in very cluttered environments. A second drawback is more relevant to the presented applications: the motion locally optimized for the current time step is generally not optimal for the entire trajectory. A potential solution is to solve the optimization problem over a receding horizon. An implementation of this idea based on the conjugate gradient method is presented, which shows very promising results both for a minimal planar model and a spatial 9-DoF manipulator.

The presented algorithms can serve as building blocks enabling efficient modeling, simulation and real-time control of redundant and high-dimensional systems. They have been proven in many experiments with the walking robot Lola and the redundant CROPS manipulator. For the agricultural robot, the algorithms are successfully used by many partners working with the manipulator.

While the presented methods have all proven to work well, many open questions remain. An additional line of research necessary for realizing the vision of robots performing human-like tasks reactively in complex, 3D environments is aimed at improving the basic mechanisms for generating and stabilizing the motion using feedback. For the task of biped walking, this is the objective of ongoing research in the robotics group at Institute of Applied Mechanics (Buschmann et al. 2012a; Ewald et al. 2012; Ewald and Buschmann 2013; Wittmann et al. 2014).

As mentioned above, the real-time capability is one of the major challenges in the control of complex robots. An open question is how to use some off-line computation for reducing run-time costs without excessive memory requirements or reducing the space of potential movements. Also, the technological road-map for processors shows that any algorithm aiming at fully utilizing the available computing power will have to be parallelized. This is a major challenge in robotics, where many standard algorithms are currently inherently serial.

The section on predictive redundancy resolution shows one possible path towards using optimal control methods in real-time for complex systems by reducing the search space. A detailed comparison of the proposed conjugate gradient method with other linear and quadratic approaches from dynamic programming (e.g., DDP, iLQR) are the subject of ongoing work. An open question is how far this approach can be extended by including dynamic models without loosing real-time capability and how such a module should be integrated into a layered control system. This is related to the constant question of how to adequately model the system for a given module in the controller, so as to include sufficient detail, but still retain the real-time capability essential for robust and flexible performance.

A further open question is the integration of continuous and state-dependent components, including continuous task switching without excessive cost (cf. Mansard and Khatib 2008). This is an important aspect for integrating the work described in this

monograph with the second, ongoing line of research mentioned above.

Finally, learning from motor control research in neuroscience remains an ongoing challenge. While classically designed controllers currently offer superior performance for most robotic tasks, the interaction with neuroscience can help us reason about possible approaches to generating a desired behavior (cf. Buschmann et al. 2014, 2012a).

Appendix A

Kinematics

This appendix summarizes some useful equations for kinematics calculations of kinematic trees.

A.1 Useful Identities

Cross Product and Spin Tensor

$$\widetilde{\boldsymbol{x}}\boldsymbol{y} = \boldsymbol{x} \times \boldsymbol{y} \tag{A.1}$$

$$\widetilde{\boldsymbol{x}}\boldsymbol{y} = \widetilde{\boldsymbol{y}}^T\boldsymbol{x} = -\widetilde{\boldsymbol{y}}\boldsymbol{x} \tag{A.2}$$

$${}_I\widetilde{\boldsymbol{x}} = \boldsymbol{A}_{IB}\,{}_B\widetilde{\boldsymbol{x}}\,\boldsymbol{A}_{BI} \tag{A.3}$$

$$\widetilde{\boldsymbol{x}}\boldsymbol{x} = \boldsymbol{x} \times \boldsymbol{x} = \boldsymbol{0} \tag{A.4}$$

$$\widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{y}} = \boldsymbol{y}\boldsymbol{x}^T - \boldsymbol{x}^T\boldsymbol{y}\boldsymbol{I}_3 \tag{A.5}$$

$$\widetilde{\widetilde{\boldsymbol{x}}\boldsymbol{y}} = \boldsymbol{y}\boldsymbol{x}^T - \boldsymbol{x}\boldsymbol{y}^T = \widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{y}}\widetilde{\boldsymbol{x}} \tag{A.6}$$

$$\widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{x}} = -\widetilde{\boldsymbol{x}}\|\boldsymbol{x}\|^2 \tag{A.7}$$

$$\widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{y}}\widetilde{\boldsymbol{y}}\boldsymbol{x} = \widetilde{\boldsymbol{y}}\widetilde{\boldsymbol{x}}^T\widetilde{\boldsymbol{x}}\boldsymbol{y} \tag{A.8}$$

A.2 Angular Velocity for Axis-Angle Representation

The following is a detailed derivation of the angular velocity for the axis-angle representation of spatial rotations:

$${}_B\widetilde{\boldsymbol{\omega}}_{IB} \overset{1}{=} \boldsymbol{A}_{IB}^T\dot{\boldsymbol{A}}_{IB} \tag{A.9}$$

$$\overset{2}{=} \Big(\boldsymbol{I}_3 - \widetilde{\boldsymbol{u}}\,\mathrm{s}\,\varphi + \widetilde{\boldsymbol{u}}^2(1-\mathrm{c}\,\varphi)\Big)\Big(\dot{\widetilde{\boldsymbol{u}}}\,\mathrm{s}\,\varphi + \widetilde{\boldsymbol{u}}\,\mathrm{c}\,\varphi\dot{\varphi} + (\dot{\widetilde{\boldsymbol{u}}}\widetilde{\boldsymbol{u}} + \widetilde{\boldsymbol{u}}\dot{\widetilde{\boldsymbol{u}}})(1-\mathrm{c}\,\varphi) + \widetilde{\boldsymbol{u}}^2\,\mathrm{s}\,\varphi\dot{\varphi}\Big) \tag{A.10}$$

$$\overset{3}{=} \dot{\widetilde{\boldsymbol{u}}}\,\mathrm{s}\,\varphi + \widetilde{\boldsymbol{u}}\,\mathrm{c}\,\varphi\dot{\varphi} + (\dot{\widetilde{\boldsymbol{u}}}\widetilde{\boldsymbol{u}} + \widetilde{\boldsymbol{u}}\dot{\widetilde{\boldsymbol{u}}})(1-\mathrm{c}\,\varphi) + \widetilde{\boldsymbol{u}}^2\,\mathrm{s}\,\varphi\dot{\varphi} \tag{A.11}$$

$$- \widetilde{\boldsymbol{u}}\dot{\widetilde{\boldsymbol{u}}}\,\mathrm{s}^2\,\varphi - \widetilde{\boldsymbol{u}}^2\,\mathrm{s}\,\varphi\,\mathrm{c}\,\varphi\dot{\varphi} - (\widetilde{\boldsymbol{u}}\dot{\widetilde{\boldsymbol{u}}}\widetilde{\boldsymbol{u}} + \widetilde{\boldsymbol{u}}^2\dot{\widetilde{\boldsymbol{u}}})(\mathrm{s}\,\varphi - \mathrm{s}\,\varphi\,\mathrm{c}\,\varphi) - \widetilde{\boldsymbol{u}}^3\,\mathrm{s}^2\,\varphi\dot{\varphi} \tag{A.12}$$

$$+ \widetilde{\boldsymbol{u}}^2\dot{\widetilde{\boldsymbol{u}}}(\mathrm{s}\,\varphi - \mathrm{s}\,\varphi\,\mathrm{c}\,\varphi) + \widetilde{\boldsymbol{u}}^3(\mathrm{c}\,\varphi - \mathrm{c}^2\,\varphi)\dot{\varphi} + (\widetilde{\boldsymbol{u}}^2\dot{\widetilde{\boldsymbol{u}}}\widetilde{\boldsymbol{u}} + \widetilde{\boldsymbol{u}}^3\dot{\widetilde{\boldsymbol{u}}})(1-\mathrm{c}\,\varphi)^2 \tag{A.13}$$

$$+ \widetilde{\boldsymbol{u}}^4(\mathrm{s}\,\varphi - \mathrm{s}\,\varphi\,\mathrm{c}\,\varphi)\dot{\varphi} \tag{A.14}$$

$$\overset{4}{=} \widetilde{\boldsymbol{u}}\dot{\varphi} + \dot{\widetilde{\boldsymbol{u}}}\,\mathrm{s}\,\varphi - (1-\mathrm{c}\,\varphi)(\widetilde{\boldsymbol{u}}\dot{\widetilde{\boldsymbol{u}}} - \dot{\widetilde{\boldsymbol{u}}}\widetilde{\boldsymbol{u}}) - \widetilde{\boldsymbol{u}}\dot{\widetilde{\boldsymbol{u}}}\widetilde{\boldsymbol{u}}\,\mathrm{s}\,\varphi(1-\mathrm{c}\,\varphi) + \widetilde{\boldsymbol{u}}^2\dot{\widetilde{\boldsymbol{u}}}\widetilde{\boldsymbol{u}}(1-\mathrm{c}\,\varphi)^2 \tag{A.15}$$

$$\overset{5}{=}\widetilde{\boldsymbol{u}}\dot{\varphi}+\dot{\widetilde{\boldsymbol{u}}}\,\mathrm{s}\,\varphi-(1-\mathrm{c}\,\varphi)\widetilde{\widetilde{\boldsymbol{u}}\dot{\boldsymbol{u}}}. \tag{A.16}$$

The first equality follows from (2.22), the second one from (2.33) and the third one simply by multiplication. Equality four uses (A.7), as well as trigonometric simplifications, and the last one (A.6). For all transformations the fact that $\|\boldsymbol{u}\|=1$ is exploited. Eliminating the spin tensor from the final result gives the vector representation of $\boldsymbol{\omega}$:

$${}_B\boldsymbol{\omega}_{IB}=\boldsymbol{u}\dot{\varphi}+\dot{\boldsymbol{u}}\sin\varphi-(1-\cos\varphi)\widetilde{\boldsymbol{u}}\dot{\boldsymbol{u}} \tag{A.17}$$

A.3 The Denavit-Hartenberg Convention

The Denavit-Hartenberg convention (DH convention) is a widely used parametrization of the relative location of two rigid bodies connected by a rotary or prismatic joint. It describes both the location of the body-fixed frame and the relative location of the bodies to each other. It is repeated here for convenience. Note that different variants of the DH convention can be found in the literature (for review see Lipkin 2005). In the following, the *proximal variant* presented in (Craig 1989) is used. The DH convention is usually applied to serial kinematic chains and will be described for this case in the following. However, it can easily be adapted to tree structures by replacing the indices $i-1$ with $p(i)$ in the following equations and associating the parameters with $p(i)$ not with the parent but the i-th body itself.

The DH parameters are defined as:

- a_i: distance between the z_i axis and z_{i+1} axis along the x_i axis.
- α_i: twist angle between z_i axis and z_{i+1} axis about the x_i axis.
- d_i: distance between x_{i-1} axis and x_i axis along the z_i axis.
- θ_i: rotation angle about x_{i-1} axis and $\boldsymbol{x}_i$ axis about z_i axis.

For a given mechanism, the body-fixed frames are then chosen according to the following recipe:

1. Assign numbers 1 to N to the bodies. The environment is denoted by 0.
2. Label the axes of motion from 1 to N.
   The following statements refer to the axes i and $i+1$
3. The origin of $\mathscr{F}_i$ is defined by:
   - The intersection of the i-th axis of motion with the orthogonal connection to the $i+1$-st axis, or
   - The intersection of the i-th and $i+1$-st axis
4. The z_i axis is identical to the i-th axis of motion with the positive axis direction defined by the positive direction of the corresponding DoF.

5. The x_i axis is chosen as follows:
   - Along the orthogonal connection of z_i and z_{i+1} with x_i pointing from $\mathscr{F}_i$ towards $\mathscr{F}_{i+1}$.
   - If the axes intersect, x_i is chosen orthogonally to z_i and z_{i+1}
6. The y_i axis is chosen such that x_i, y_i, z_i form a right-handed coordinate frame.
7. Choosing $\mathscr{F}_0$ and $\mathscr{F}_N$:
   - $\mathscr{F}_0$: Aligns with $\mathscr{F}_1$ for $q_1 = 0$ (θ_1 or d_1, depending on the type of joint).
   - $\mathscr{F}_N$: Origin and x_N axis are chosen in such a way, that as many DH parameters as possible vanish.

In cases in which there is more than one possible choice of parameters, the general rule is to set as many as possible to zero.

The Denavit-Hartenberg Transformation

The position of a point P relative to frame $\mathscr{F}_i$ is given by the vector ${}_i\boldsymbol{r}_{iP}$. The same physical point may also be described relative another frame $\mathscr{F}_k$ by the vector ${}_k\boldsymbol{r}_{kP}$. A conversion between both representation is achieved via the vector chain $\mathscr{F}_k, \mathscr{F}_i, P$ and a coordinate transformation:

$$ {}_k\boldsymbol{r}_{kP} = {}_k\boldsymbol{r}_{ki} + \boldsymbol{A}_{ki}\,{}_i\boldsymbol{r}_{iP}\,. \tag{A.18} $$

The same equation may be written more compactly using the *homogeneous transformation* $\boldsymbol{T}_{ki}$:

$$ \underbrace{\begin{pmatrix} {}_k\boldsymbol{r}_{kP} \\ 1 \end{pmatrix}}_{{}_k\boldsymbol{z}_p} = \underbrace{\begin{pmatrix} \boldsymbol{A}_{ki} & {}_k\boldsymbol{r}_{ki} \\ \boldsymbol{0} & 1 \end{pmatrix}}_{\boldsymbol{T}_{ki}} \underbrace{\begin{pmatrix} {}_i\boldsymbol{r}_{iP} \\ 1 \end{pmatrix}}_{{}_i\boldsymbol{z}_p}. \tag{A.19} $$

The coordinate vector $\boldsymbol{r}_p$ extended by 1 ($\boldsymbol{z}_p$) is called the vector of *homogeneous coordinates*.

The homogeneous transformation $\boldsymbol{T}_{i-1,i}$ from $\mathscr{F}_i$ to $\mathscr{F}_{i-1}$ is uniquely defined by the DH parameters. The full *DH transformation* is obtained by four consecutive *elementary homogeneous transformations* (translations or rotations along/about the coordinate axes):

1. Rotation by $-\theta_i$about z_i: $\boldsymbol{T}_1 = \begin{pmatrix} \boldsymbol{A}_z(-\theta_i) & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{pmatrix}$
2. Translation by $-d_i$ along z_i: $\boldsymbol{T}_2 = \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{e}_z d_i \\ \boldsymbol{0} & 1 \end{pmatrix}$
3. Translation by $-a_{i-1}$ along $\boldsymbol{x}_{i-1}$: $\boldsymbol{T}_3 = \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{e}_x a_{i-1} \\ \boldsymbol{0} & 1 \end{pmatrix}$
4. Rotation by $-\alpha_{i-1}$ along $\boldsymbol{x}_{i-1}$: $\boldsymbol{T}_4 = \begin{pmatrix} \boldsymbol{A}_x(-\alpha_{i-1}) & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{pmatrix}$

The full DH transformation is then given by:

$$\begin{aligned}\boldsymbol{T}_{i-1,i} &= \boldsymbol{T}_4\boldsymbol{T}_3\boldsymbol{T}_2\boldsymbol{T}_1 \\ &= \begin{pmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}.\end{aligned} \tag{A.20}$$

A.4 Recursive Kinematics Calculations

This section lists equations for recursively calculating kinematic quantities of tree structured MBSs. Note: gradients with respect to the generalized coordinates $\boldsymbol{q}$ are written in index notation using the Einstein summation convention. In that case, vector and matrix indices are written in square brackets, that is, $M_{[i,j]}$ denotes the i-th row and j-th column of $\boldsymbol{M}$ and $a_{[i]} = M_{[i,k]}b_{[k]}$ is equivalent to $\boldsymbol{a} = \boldsymbol{M}\boldsymbol{b}$.

A.4.1 Recursive Equations

Positions, Velocities and Accelerations

These equations are valid for all bodies except the for the root, i.e. $i > 1$. For the root body, the equations depend on the coupling with the environment and are a function of the relative DoFs $\boldsymbol{q}_1$ (see section 2.3.2). The index p refers to the parent of the i-th body, i.e. $p = p(i)$.

$$\boldsymbol{A}_{0i} = \boldsymbol{A}_{0p}\boldsymbol{A}_{pi} \tag{A.21}$$

$${}_i\boldsymbol{r}_i = \boldsymbol{A}_{ip}\left({}_p\boldsymbol{r}_p + {}_p\boldsymbol{r}_{pi}\right) \tag{A.22}$$

$${}_i\boldsymbol{\omega}_{0i} = \boldsymbol{A}_{ip}\,{}_p\boldsymbol{\omega}_{0p} + {}_i\boldsymbol{\omega}_{pi} \tag{A.23}$$

$${}_i\dot{\boldsymbol{r}}_i = \boldsymbol{A}_{ip}\left({}_p\dot{\boldsymbol{r}}_p + {}_p\widetilde{\boldsymbol{\omega}}_{0p}\,{}_p\boldsymbol{r}_{pi} + {}_p\mathring{\boldsymbol{r}}_{pi}\right) \tag{A.24}$$

$${}_i\dot{\boldsymbol{\omega}}_{0i} = \boldsymbol{A}_{ip}\left({}_p\dot{\boldsymbol{\omega}}_{0p} + {}_p\widetilde{\boldsymbol{\omega}}_{pi}\,{}_p\widetilde{\boldsymbol{\omega}}_{0p}\right) + {}_i\dot{\boldsymbol{\omega}}_{pi} \tag{A.25}$$

$${}_i\ddot{\boldsymbol{r}}_i = \boldsymbol{A}_{ip}\left({}_p\ddot{\boldsymbol{r}}_p + \left({}_0\dot{\widetilde{\boldsymbol{\omega}}}_{0p} + {}_0\widetilde{\boldsymbol{\omega}}_{0p}\,{}_0\widetilde{\boldsymbol{\omega}}_{0p}\right)\boldsymbol{A}_{0p}\,{}_p\boldsymbol{r}_{pi} + 2\,{}_0\widetilde{\boldsymbol{\omega}}_{0p}\boldsymbol{A}_{0p}\,{}_p\mathring{\boldsymbol{r}}_{pi} + \boldsymbol{A}_{0p}\,{}_p\mathring{\mathring{\boldsymbol{r}}}_{pi}\right) \tag{A.26}$$

Jacobian Matrices

$${}_i\boldsymbol{J}_{R,i} := \frac{\partial\,{}_i\boldsymbol{\omega}_i}{\partial\dot{\boldsymbol{q}}} = \boldsymbol{A}_{ip}\,{}_p\boldsymbol{J}_{R,p} + \underbrace{\frac{\partial\,{}_i\boldsymbol{\omega}_{pi}}{\partial\dot{\boldsymbol{q}}}}_{=:{}_i\boldsymbol{J}_{R,rel,i}} \tag{A.27}$$

$${}_i\boldsymbol{J}_{T,i} := \frac{\partial\,{}_i\dot{\boldsymbol{r}}_i}{\partial\dot{\boldsymbol{q}}} = \boldsymbol{A}_{ip}\,{}_p\boldsymbol{J}_{T,p} + \boldsymbol{A}_{ip}\,{}_p\widetilde{\boldsymbol{r}}_{pi}^{\,T}\,{}_p\boldsymbol{J}_{T,p} + \underbrace{\frac{\partial\,{}_i\mathring{\boldsymbol{r}}_{pi}}{\partial\dot{\boldsymbol{q}}}}_{=:{}_i\boldsymbol{J}_{T,rel,i}} \tag{A.28}$$

$${}_iJ_{Rq,i,[k,l]} := \frac{\partial\,{}_i\omega_{i,[k]}}{\partial q_{[l]}} = A_{ip,[k,m]}\,{}_pJ_{Rq,p,[m,l]}$$

$$+ \underbrace{\frac{\partial A_{ip,[k,m]}}{\partial q_{[l]}} {}_p\boldsymbol{\omega}_{p,[m]} + \frac{\partial {}_i\omega_{pi,[k]}}{\partial q_{[l]}}}_{=:{}_iJ_{Rq,rel,i,[k,l]}} \tag{A.29}$$

$${}_iJ_{Tq,i,[k,l]} := \frac{\partial {}_i\dot{r}_{i,[k]}}{\partial q_{[l]}} = A_{ip,[k,m]}\, {}_pJ_{Tq,p,[m,l]} + \left(\boldsymbol{A}_{ip}\, {}_p\tilde{\boldsymbol{r}}_{pi}^{\,T}\, {}_p\boldsymbol{J}_{Rq,p}\right)_{[k,l]}$$

$$+ \underbrace{\frac{\partial A_{ip,[k,m]}}{\partial q_{[l]}} {}_p\dot{r}_{p,[m]} + \frac{\partial {}_i\mathring{r}_{pi,[k]}}{\partial q_{[l]}}}_{=:{}_iJ_{Tq,rel,i,[k,l]}} \tag{A.30}$$

A.4.2 Relative Quantities

In the following, relative quantities are given for rotary and prismatic joints described by the Denavit-Hartenberg convention (see appendix A.3). However, the procedure is not limited to these simple joints and can easily be extended to more complex coupling mechanisms (for example, equations for nonlinear joint kinematics of Lola are given in Buschmann 2010).

Note that it might be desirable to deviate from the DH convention in order to be able to choose the orientation of the body-fixed frames more freely. In that case, the relative location and orientation may be described by a fixed rotation matrix $\boldsymbol{H}$ and offset vector $\boldsymbol{r}_{pi,0}$, such that for a rotary joint

$${}_p\boldsymbol{r}_{pi} = {}_p\boldsymbol{r}_{pi,0} \tag{A.31}$$

$$\boldsymbol{A}_{ip} = \boldsymbol{A}_z(q_i)\boldsymbol{H} \tag{A.32}$$

and for a prismatic joint

$${}_p\boldsymbol{r}_{pi} = {}_p\boldsymbol{r}_{pi,0} + \boldsymbol{H}\boldsymbol{e}_z q_i \tag{A.33}$$

$$\boldsymbol{A}_{ip} = \boldsymbol{H}. \tag{A.34}$$

The additional flexibility in choosing the frames, however, comes at the cost of slightly more floating point operations.

In the following, the equations are given based on the DH-Transformation (A.20).

Rotary Joints

For rotary joints, the parameter d_i is fixed and $\theta_i = q_i$ is the joint DoF. Relative rotation and translation are directly given by (A.20):

$${}_p\boldsymbol{r}_{pi} = \boldsymbol{A}_x^T(\alpha)(\boldsymbol{e}_z d + \boldsymbol{e}_x a) = \begin{pmatrix} a \\ -\sin(\alpha)d \\ \cos(\alpha)d \end{pmatrix} = \text{constant} \tag{A.35}$$

$$\boldsymbol{A}_{ip} = \boldsymbol{A}_z(q_i)\boldsymbol{A}_x(\alpha) \tag{A.36}$$

The remaining quantities are obtained by differentiation:

$${}_p\dot{\boldsymbol{r}}_{pi} = {}_p\boldsymbol{\omega}_p \times {}_p\boldsymbol{r}_{pi} \tag{A.37}$$

$${}_i\boldsymbol{\omega}_{pi} = \boldsymbol{e}_z\dot{q}_i \tag{A.38}$$

$$ {}_iJ_{R,rel,i,[k,l]} = \begin{cases} 1 & \text{for} \quad l = i \wedge k = 3 \\ 0 & \text{else} \end{cases} \tag{A.39} $$

$$ {}_i\boldsymbol{J}_{T,rel,i} = \boldsymbol{0} \tag{A.40} $$

$$ {}_iJ_{Rq,rel,i,[k,l]} = \frac{\partial A_{ip,[k,m]}}{\partial q_{[l]}} {}_p\omega_{p,[m]} \qquad \text{(Note: 0 for } l \neq i) \tag{A.41} $$

$$ {}_iJ_{Tq,rel,i,[k,l]} = \frac{\partial A_{ip,[k,m]}}{\partial q_{[l]}} \left({}_p\dot{\boldsymbol{r}}_i \right)_{[m]} \tag{A.42} $$

Prismatic Joints

For prismatic joints $d_i = q_i$ is the DoF and θ_i is a fixed parameter. Rotation and translation are again given by (A.20):

$$ {}_p\boldsymbol{r}_{pi} = \boldsymbol{A}_x^T(\alpha)(\boldsymbol{e}_z d + \boldsymbol{e}_x a) = \underbrace{\begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix}}_{=:{}_p\boldsymbol{r}_{pi,0}=\text{constant!}} + \underbrace{\begin{pmatrix} 0 \\ -\sin(\alpha) \\ +\cos(\alpha) \end{pmatrix} q_i}_{=:{}_p\Delta\boldsymbol{r}_{pi}(q_i)} \tag{A.43} $$

$$ \boldsymbol{A}_{pi} = \boldsymbol{A}_z(\theta)\boldsymbol{A}_x(\alpha) = \text{constant!} \tag{A.44} $$

The remaining quantities are again obtained by differentiation:

$$ {}_p\dot{\boldsymbol{r}}_{pi} = \boldsymbol{A}_x^T(\alpha)\boldsymbol{e}_z\dot{q}_i + {}_p\boldsymbol{\omega}_p \times {}_p\boldsymbol{r}_{pi} \tag{A.45} $$

$$ {}_i\boldsymbol{\omega}_{pi} = \boldsymbol{0} \tag{A.46} $$

$$ {}_i\boldsymbol{J}_{R,rel,i} = \boldsymbol{0} \tag{A.47} $$

$$ {}_iJ_{T,rel,i,[k,l]} = \begin{cases} (\boldsymbol{A}_{ip}\boldsymbol{A}_x^T(\alpha)\boldsymbol{e}_z)_{[k]} & \text{for} \quad \text{l=i} \\ 0 & \text{else} \end{cases} \tag{A.48} $$

$$ {}_iJ_{Rq,rel,i,[k,l]} = \boldsymbol{0} \tag{A.49} $$

$$ {}_iJ_{Tq,rel,i,[k,l]} = \begin{cases} \left(\boldsymbol{A}_{ip} \left({}_p\boldsymbol{\omega}_p \times (\boldsymbol{A}_x^T(\alpha)\boldsymbol{e}_z) \right) \right)_{[k]} & \text{for} \quad \text{l=i} \\ 0 & \text{else} \end{cases} \tag{A.50} $$

Free Bodies (6-DoF Joints)

For freely floating bodies, the position is simply given by the absolute position relative to the inertial frame

$$ {}_I\boldsymbol{r}_{Ii} = \begin{pmatrix} q_{xi}, \\ q_{yi} \\ q_{zi} \end{pmatrix}, \tag{A.51} $$

$$ {}_i\boldsymbol{r}_{Ii} = \boldsymbol{A}_{iI}\,{}_I\boldsymbol{r}_{Ii}, \tag{A.52} $$

where q_{xi}, q_{yi}, q_{zi} are corresponding entries of the generalized coordinate vector $\boldsymbol{q}$. For the orientation, any of the parametrizations of spatial rotations can be chosen (see section 2.2).

Jacobians, velocities and accelerations are trivially obtained by differentiation.

A.5 Gradients for Predictive Inverse Kinematics

A.5.1 Gradients for Optimization on Velocity Level

$$\frac{\partial f}{\partial \boldsymbol{q}} = \frac{\partial \dot{\boldsymbol{q}}}{\partial \boldsymbol{q}} = \boldsymbol{J}_q^{\#}\dot{\boldsymbol{w}} - \boldsymbol{J}_q^{\#}\boldsymbol{J}_u - \boldsymbol{J}^{\#}\boldsymbol{J}_q\boldsymbol{u} \tag{A.53}$$

$$\frac{\partial f}{\partial \boldsymbol{u}} = \boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J} \tag{A.54}$$

$$\frac{\partial H}{\partial \boldsymbol{u}} = \frac{\partial \varphi}{\partial \boldsymbol{u}} + \boldsymbol{\lambda}^T \frac{\partial f}{\partial \boldsymbol{u}} \tag{A.55}$$

$$\frac{\partial \varphi_v}{\partial \boldsymbol{q}} = \frac{\partial}{\partial \boldsymbol{q}}\left(\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}}\right) = 2\left(\frac{\partial f}{\partial \boldsymbol{q}}\right)^T \dot{\boldsymbol{q}} \tag{A.56}$$

$$\frac{\partial \varphi_v}{\partial \boldsymbol{u}} = 2\left(\frac{\partial f}{\partial \boldsymbol{u}}\right)^T \dot{\boldsymbol{q}} \tag{A.57}$$

$$\frac{\partial \varphi_l}{\partial \boldsymbol{q}} = \begin{cases} \sum_{i=1}^{n} 2\frac{(q_i - \bar{q}_{M,i})}{(q_{M,i} - \bar{q}_{M,i})^2} & \text{if } q_i > \bar{q}_{M,i} \\ \sum_{i=1}^{n} 2\frac{(q_i - \bar{q}_{m,i})}{(q_{m,i} - \bar{q}_{m,i})^2} & \text{if } q_i < \bar{q}_{m,i} \\ 0 & \text{else} \end{cases} \tag{A.58}$$

$$\boldsymbol{J}_{q_i}^{\#} = \frac{\partial \boldsymbol{J}^{\#}}{\partial q_i} = \partial_{q_i}\boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} + \boldsymbol{J}^T\left[-\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1}\left((\partial_{q_i}\boldsymbol{J})\boldsymbol{J}^T + \boldsymbol{J}(\partial_{q_i}\boldsymbol{J}^T)\right)\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1}\right] \tag{A.59}$$

A.5.2 Gradients for Optimization on Acceleration Level

$$\frac{\partial f_1}{\partial \boldsymbol{q}} = \frac{\partial f}{\partial \boldsymbol{q}}, \quad \frac{\partial f_1}{\partial \boldsymbol{u}} = \boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J} \tag{A.60}$$

$$\frac{\partial f_2}{\partial \boldsymbol{q}} = \frac{\partial f_2}{\partial \boldsymbol{u}} = \boldsymbol{0} \tag{A.61}$$

$$\frac{\partial \varphi_{na}}{\partial \hat{\boldsymbol{u}}} = 2\hat{\boldsymbol{u}}, \quad \frac{\partial f_2}{\partial \hat{\boldsymbol{u}}} = \boldsymbol{I} \tag{A.62}$$

$$\frac{\partial H}{\partial \hat{\boldsymbol{u}}} = \frac{\partial \varphi}{\partial \hat{\boldsymbol{u}}} + \boldsymbol{\lambda}_2^T \tag{A.63}$$

Appendix B

Gauss' Principle for a Rigid Body

In this section we derive Gauss' principle for rigid bodies. We start with (3.8), which is reproduced here for convenience:

$$G = \frac{1}{2}\int_S (\ddot{\boldsymbol{r}}_m - \tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m})^2\,\mathrm{d}m \to \min! \quad \text{with} \quad \boldsymbol{\Phi}(\boldsymbol{z}) = \boldsymbol{0}. \tag{B.1}$$

Substituting (2.28) for $\ddot{\boldsymbol{r}}_m$ yields:

$$G = \frac{1}{2}\int_S \left(\ddot{\boldsymbol{r}}_m - \tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m}\right)^2 \tag{B.2}$$

$$= \frac{1}{2}\int_S \left(\ddot{\boldsymbol{r}}_c + (\dot{\tilde{\boldsymbol{\omega}}} + \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})\boldsymbol{r}_{cm} - \tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m}\right)^2 \tag{B.3}$$

$$\begin{aligned} = &\underbrace{\frac{1}{2}\int_S \ddot{\boldsymbol{r}}_c^2\,\mathrm{d}m}_{(1)} \underbrace{- \int_S \ddot{\boldsymbol{r}}_c^T\,\mathrm{d}\boldsymbol{F}^i}_{(2)} + \underbrace{\frac{1}{2}\int_S \left(\tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m}\right)^2 \mathrm{d}m}_{(3)} \underbrace{- \int_S \boldsymbol{r}_{cm}^T(\dot{\tilde{\boldsymbol{\omega}}}^T + \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})\left(\tfrac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m}\right)\mathrm{d}m}_{(4)} \\ &+ \underbrace{\int_S \ddot{\boldsymbol{r}}_c^T(\dot{\tilde{\boldsymbol{\omega}}}^T + \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})\boldsymbol{r}_{cm}\,\mathrm{d}m}_{(5)} + \underbrace{\frac{1}{2}\int_S \boldsymbol{r}_{cm}^T(\dot{\tilde{\boldsymbol{\omega}}}^T + \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})(\dot{\tilde{\boldsymbol{\omega}}}^T + \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})\boldsymbol{r}_{cm}\,\mathrm{d}m}_{(6)}. \end{aligned} \tag{B.4}$$

In the general case, we must know the force-field $\mathrm{d}\boldsymbol{F}^i(\boldsymbol{r}_m)$, if we want to evaluate the integral. For rigid bodies, however, any force-field that yields the same overall wrench is equivalent. Assuming the impressed wrench $(\boldsymbol{F}^i, \boldsymbol{T}_c^i)$ at the CoM is known, an equivalent force field $\mathrm{d}\boldsymbol{F}^i(\boldsymbol{r}_m)$ can be calculated using the ansatz $\mathrm{d}\boldsymbol{F}^i(\boldsymbol{r}_m) = (\boldsymbol{a} + \tilde{\boldsymbol{r}}_m^T\boldsymbol{b})\,\mathrm{d}m$, where $\boldsymbol{a}, \boldsymbol{b}$ are unknown constants. The conditions for determining the unknowns are

$$\boldsymbol{F}^i = \int_s (\boldsymbol{a} + \tilde{\boldsymbol{r}}_m^T\boldsymbol{b})\,\mathrm{d}m = m\boldsymbol{a} \quad \textit{and} \tag{B.5}$$

$$\boldsymbol{T}_c^i = \int_s \tilde{\boldsymbol{r}}_m(\boldsymbol{a} + \tilde{\boldsymbol{r}}_m^T\boldsymbol{b})\,\mathrm{d}m = \boldsymbol{\Theta}_c\boldsymbol{b}. \tag{B.6}$$

The last equations make use of the definition of the CoM $m\boldsymbol{r}_c = \int_s \boldsymbol{r}_m \,\mathrm{d}m$ and the mass moment of inertia $\boldsymbol{\Theta}_c = \int_s \widetilde{\boldsymbol{r}}_{cm}\widetilde{\boldsymbol{r}}_{cm}^T \,\mathrm{d}m$. Solving for $\boldsymbol{a}, \boldsymbol{b}$ yields:

$$\boldsymbol{a} = m^{-1}\boldsymbol{F}^i, \tag{B.7}$$

$$\boldsymbol{b} = \boldsymbol{\Theta}_c^{-1}\boldsymbol{T}_c^i, \tag{B.8}$$

$$\Rightarrow \mathrm{d}\boldsymbol{F}^i = \left(m^{-1}\boldsymbol{F}^i + \widetilde{\boldsymbol{r}}_{cm}^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}_c^i\right)\mathrm{d}m. \tag{B.9}$$

Evaluating the integrals yields (equations labeled according to (B.4), using identities for the cross product from appendix A.1):

$$\frac{1}{2}\int_S \ddot{\boldsymbol{r}}_c^2 \,\mathrm{d}m = \frac{1}{2}m\ddot{\boldsymbol{r}}_c^2, \tag{1}$$

$$\int_S \ddot{\boldsymbol{r}}_c^T \,\mathrm{d}\boldsymbol{F}^i = \ddot{\boldsymbol{r}}_c^T\boldsymbol{F}^i, \tag{2}$$

$$\begin{aligned}\frac{1}{2}\int_S \left(\frac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m}\right)^2 \mathrm{d}m =& \frac{1}{2}\int_S m^{-2}(\boldsymbol{F}^i)^2 + m^{-1}(\boldsymbol{F}^i)^T\widetilde{\boldsymbol{r}}_{cm}^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}^i \,\mathrm{d}m \\ &+ \frac{1}{2}\int_S (\boldsymbol{T}^i)^T\boldsymbol{\Theta}_c^{-1}\widetilde{\boldsymbol{r}}_{cm}\boldsymbol{F}^i m^{-1}\boldsymbol{T}^i + (\boldsymbol{T}^i)^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{r}_{cm}\widetilde{\boldsymbol{r}}_{cm}^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}^i \,\mathrm{d}m \\ &= \frac{1}{2m}(\boldsymbol{F}^i)^2 + \frac{1}{2}(\boldsymbol{T}^i)^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}^i,\end{aligned} \tag{3}$$

$$\begin{aligned}\int_S \boldsymbol{r}_{cm}^T(\dot{\widetilde{\boldsymbol{\omega}}}^T + \widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}})\left(\frac{\mathrm{d}\boldsymbol{F}^i}{\mathrm{d}m}\right)\mathrm{d}m =& \underbrace{\int_S \boldsymbol{r}_{cm}^T(\dot{\widetilde{\boldsymbol{\omega}}}^T + \widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}})m^{-1}\boldsymbol{F}^i}_{=0} \\ &+ \int_S \boldsymbol{r}_{cm}^T(\dot{\widetilde{\boldsymbol{\omega}}}^T + \widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}})\widetilde{\boldsymbol{r}}_{cm}^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}_c^i \\ =& \int_S \boldsymbol{r}_{cm}^T\dot{\widetilde{\boldsymbol{\omega}}}^T\widetilde{\boldsymbol{r}}_{cm}^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}_c^i + \int_S \boldsymbol{r}_{cm}^T\widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{r}}_{cm}^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}_c^i \\ =& \dot{\boldsymbol{\omega}}^T\boldsymbol{T}^i - \boldsymbol{\omega}^T\boldsymbol{\Theta}_c\widetilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}^i,\end{aligned} \tag{4}$$

$$\int_S \ddot{\boldsymbol{r}}_c^T(\dot{\widetilde{\boldsymbol{\omega}}}^T + \widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}})\boldsymbol{r}_{cm} \,\mathrm{d}m = 0, \tag{5}$$

$$
\begin{aligned}
&\frac{1}{2}\int_S \boldsymbol{r}_{cm}^T(\dot{\tilde{\boldsymbol{\omega}}}^T+\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})(\dot{\tilde{\boldsymbol{\omega}}}^T+\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})\boldsymbol{r}_{cm}\,\mathrm{d}m\\
=&\frac{1}{2}\int_S \boldsymbol{r}_{cm}^T\dot{\tilde{\boldsymbol{\omega}}}^T\dot{\tilde{\boldsymbol{\omega}}}\boldsymbol{r}_{cm}\,\mathrm{d}m+\frac{1}{2}\int_S \boldsymbol{r}_{cm}^T\dot{\tilde{\boldsymbol{\omega}}}^T\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\,\mathrm{d}m\\
&+\frac{1}{2}\int_S \boldsymbol{r}_{cm}^T\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\dot{\tilde{\boldsymbol{\omega}}}\,\mathrm{d}m+\frac{1}{2}\int_S \boldsymbol{r}_{cm}^T\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\boldsymbol{r}_{cm}\,\mathrm{d}m\\
=&\frac{1}{2}\dot{\boldsymbol{\omega}}^T\int_S \tilde{\boldsymbol{r}}_{cm}\tilde{\boldsymbol{r}}_{cm}^T\,\mathrm{d}m\dot{\boldsymbol{\omega}}+\frac{1}{2}\dot{\boldsymbol{\omega}}^T\tilde{\boldsymbol{\omega}}\int_S \tilde{\boldsymbol{r}}_{cm}\tilde{\boldsymbol{r}}_{cm}^T\,\mathrm{d}m\boldsymbol{\omega}\\
&-\frac{1}{2}\boldsymbol{\omega}^T\int_S \tilde{\boldsymbol{r}}_{cm}\tilde{\boldsymbol{r}}_{cm}^T\,\mathrm{d}m\tilde{\boldsymbol{\omega}}\dot{\tilde{\boldsymbol{\omega}}}+\frac{1}{2}\boldsymbol{\omega}^2\boldsymbol{\omega}^T\int_S \tilde{\boldsymbol{r}}_{cm}\tilde{\boldsymbol{r}}_{cm}^T\,\mathrm{d}m\boldsymbol{\omega}\\
=&\frac{1}{2}\dot{\boldsymbol{\omega}}^T\boldsymbol{\Theta}_c\dot{\boldsymbol{\omega}}+\dot{\boldsymbol{\omega}}^T\left(\tilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_c\boldsymbol{\omega}\right)+\frac{1}{2}\boldsymbol{\omega}^2\boldsymbol{\omega}^T\boldsymbol{\Theta}_c\boldsymbol{\omega}.
\end{aligned}
\tag{6}
$$

With this, we have:

$$
\begin{aligned}
G=&\frac{1}{2}m\ddot{\boldsymbol{r}}_c^2+\frac{1}{2}\dot{\boldsymbol{\omega}}^T\boldsymbol{\Theta}_c\dot{\boldsymbol{\omega}}+\dot{\boldsymbol{\omega}}^T\left(\tilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_c\boldsymbol{\omega}\right)-\ddot{\boldsymbol{r}}_c^T\boldsymbol{F}^i-\dot{\boldsymbol{\omega}}^T\boldsymbol{T}^i\\
&+\underbrace{\left(\boldsymbol{\omega}^T\boldsymbol{\Theta}_c\tilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}^i+\frac{1}{2}\boldsymbol{\omega}^2\boldsymbol{\omega}^T\boldsymbol{\Theta}_c\boldsymbol{\omega}+\frac{1}{2m}(\boldsymbol{F}^i)^2+\frac{1}{2}(\boldsymbol{T}^i)^T\boldsymbol{\Theta}_c^{-1}\boldsymbol{T}^i\right)}_{\text{irrelevant}}.
\end{aligned}
\tag{B.10}
$$

Since only terms involving the accelerations $\dot{\boldsymbol{\omega}}, \ddot{\boldsymbol{r}}_c$ influence the minimum, the bracketed terms are irrelevant and may be omitted.

A more compact notation, which is equivalent except for irrelevant terms, uses the mass matrix $\boldsymbol{M}$ of the rigid body and the vector of impressed and inertial forces $\boldsymbol{h}^\star$:

$$
G=\frac{1}{2}\left(\boldsymbol{a}+\boldsymbol{M}^{-1}\boldsymbol{h}^\star\right)^T\boldsymbol{M}\left(\boldsymbol{a}+\boldsymbol{M}^{-1}\boldsymbol{h}^\star\right),
\tag{B.11}
$$

$$
\boldsymbol{M}=\begin{pmatrix}\boldsymbol{\Theta}_c & \boldsymbol{0}\\ \boldsymbol{0} & m\boldsymbol{I}\end{pmatrix},
\tag{B.12}
$$

$$
\boldsymbol{h}^\star=\begin{pmatrix}\tilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_c\boldsymbol{\omega}-\boldsymbol{T}_c^i\\ -\boldsymbol{F}^i\end{pmatrix},
\tag{B.13}
$$

$$
\boldsymbol{a}=\begin{pmatrix}\dot{\boldsymbol{\omega}}\\ \ddot{\boldsymbol{r}}_c\end{pmatrix}.
\tag{B.14}
$$

The equation for G for a reference point $o \neq c$ is easily calculated by substituting $\ddot{\boldsymbol{r}}_c = \ddot{\boldsymbol{r}}_o + (\dot{\tilde{\boldsymbol{\omega}}} + \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}})\boldsymbol{r}_{oc}$ into the above equation, yielding:

$$
G=\frac{1}{2}\left(\boldsymbol{a}+\boldsymbol{M}^{-1}\boldsymbol{h}^\star\right)^T\boldsymbol{M}\left(\boldsymbol{a}+\boldsymbol{M}^{-1}\boldsymbol{h}^\star\right),
\tag{B.15}
$$

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{\Theta}_o & m\widetilde{\boldsymbol{r}}_{oc} \\ m\widetilde{\boldsymbol{r}}_{oc}^{T} & m\boldsymbol{I} \end{pmatrix}, \tag{B.16}$$

$$\boldsymbol{h}^{\star} = \begin{pmatrix} \widetilde{\boldsymbol{\omega}}\boldsymbol{\Theta}_o\boldsymbol{\omega} - \boldsymbol{T}_o^i \\ m\widetilde{\boldsymbol{\omega}}\widetilde{\boldsymbol{\omega}}\boldsymbol{r}_{oc} - \boldsymbol{F}^i \end{pmatrix}, \tag{B.17}$$

$$\boldsymbol{a} = \begin{pmatrix} \dot{\boldsymbol{\omega}} \\ \ddot{\boldsymbol{r}}_o \end{pmatrix}. \tag{B.18}$$

Appendix C

Alternative Derivations of $\mathrm{O}(n_b)$ Method

C.1 Derivation of $\mathrm{O}(n_b)$ Method Using Constraint Forces

This section contains an alternative derivation of the $\mathrm{O}(n_b)$ method based on the free-body method, d'Alembert's Principle and the constraint forces acting at the joints. The Newton-Euler equations for the i-th body may be written as (cf. (3.20)):

$$\boldsymbol{M}_i \boldsymbol{a}_i + \boldsymbol{h}_i = \boldsymbol{f}_{Ji} - \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{f}_{Jk}. \tag{C.1}$$

Here, $\boldsymbol{f}_{Ji}$ is the total force (constraint force and impressed force) acting on the i-th body at the connection to its parent. Substituting the joint kinematics equation (2.75) into the EoM (C.1) and pre-multiplying with the transposed joint Jacobian yields:

$$\boldsymbol{J}_{Ji}^T \boldsymbol{M}_i \boldsymbol{J}_{Ji} \ddot{\boldsymbol{q}}_i + \boldsymbol{J}_{Ji}^T \boldsymbol{M}_i \boldsymbol{C}_{ip} \boldsymbol{a}_p + \boldsymbol{J}_{Ji}^T \left(\boldsymbol{M}_i \boldsymbol{b}_i + \boldsymbol{h}_i + \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{f}_{Jk} - \boldsymbol{f}_{Ji} \right) = \boldsymbol{0}. \tag{C.2}$$

According to d'Alembert's principle the constraint forces are canceled by the multiplication with the joint Jacobian and we may substitute the (known) impressed forces $\boldsymbol{u}_i$ for the sum of impressed and constraint forces $\boldsymbol{f}_{Ji}$, which are included in $\boldsymbol{h}_i^\star$. For leaf nodes we have $N_{Ci} = \emptyset$ and can therefore directly solve (C.2) for $\ddot{\boldsymbol{q}}_i$:

$$\ddot{\boldsymbol{q}}_i = \underbrace{-\left(\boldsymbol{J}_{Ji}^T \boldsymbol{M}_i \boldsymbol{J}_{Ji} \right)^{-1} \boldsymbol{J}_{Ji}^T \boldsymbol{M}_i \boldsymbol{C}_{ip}}_{\boldsymbol{P}_i} \boldsymbol{a}_p \underbrace{-\left(\boldsymbol{J}_{Ji}^T \boldsymbol{M}_i \boldsymbol{J}_{Ji} \right)^{-1} \boldsymbol{J}_{Ji}^T \left(\boldsymbol{M}_i \boldsymbol{b}_i + \boldsymbol{h}_i^\star \right)}_{\boldsymbol{p}_i} \tag{C.3}$$

$$= \boldsymbol{P}_i \boldsymbol{a}_p + \boldsymbol{p}_i. \tag{C.4}$$

Substituting the result for $\ddot{\boldsymbol{q}}_i$ into the EoM (C.1) gives us the joint force $\boldsymbol{f}_{Ji}$ as a function of the (unknown) predecessor acceleration $\boldsymbol{a}_{p(i)}$:

$$\boldsymbol{f}_{Ji} = \underbrace{\boldsymbol{M}_i \left(\boldsymbol{C}_{ip} + \boldsymbol{J}_{Ji} \boldsymbol{P}_i \right)}_{\boldsymbol{K}_i} \boldsymbol{a}_p + \underbrace{\boldsymbol{M}_i \left(\boldsymbol{J}_{Ji} \boldsymbol{p}_i + \boldsymbol{b}_i \right) + \boldsymbol{h}_i^\star}_{\boldsymbol{k}_i} \tag{C.5}$$

$$= \boldsymbol{K}_i \boldsymbol{a}_p + \boldsymbol{k}_i. \tag{C.6}$$

Substituting the joint forces $\boldsymbol{f}_{Ji}$ of the leaf nodes into the EoM of the i-th predecessor body we obtain:

$$\boldsymbol{M}_i\boldsymbol{a}_i + \boldsymbol{h}_i^\star + \sum_{k\in N_{Ci}} \boldsymbol{C}_{ki}^T \left(\boldsymbol{K}_k \underbrace{\boldsymbol{a}_{p(k)}}_{\overset{!}{=}\boldsymbol{a}_i} + \boldsymbol{k}_k \right) = \boldsymbol{0}, \tag{C.7}$$

$$\Leftrightarrow \underbrace{\left(\boldsymbol{M}_i + \sum_{k\in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{K}_k \right)}_{\overline{\boldsymbol{M}}_i} \boldsymbol{a}_i + \underbrace{\left(\boldsymbol{h}_i^\star + \sum_{k\in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{k}_k \right)}_{\overline{\boldsymbol{h}}_i^\star} = \boldsymbol{0}. \tag{C.8}$$

Since this equation has the same structure as (C.1) for a leaf node with $N_{Ci} = \emptyset$, we can recursively apply the above equations until we arrive at the root node. Since we apply the procedure to the modified EoM (C.7), we must use the modified quantities $\overline{\boldsymbol{M}}_i, \overline{\boldsymbol{h}^\star}_i$. At the root node we can directly solve (C.4) for $\ddot{\boldsymbol{q}}_1$, since there is no unknown predecessor acceleration.

C.2 Derivation of $\mathrm{O}(n_b)$ Method Using Projective EoM

This section contains an alternative derivation of the $\mathrm{O}(n_b)$ method based on the projective form or the EoM and the recursive kinematics equations. The i-th line of the EoM is (cf. (3.36)):

$$\boldsymbol{J}_{Ji}^T \left[\left(\boldsymbol{M}_i\boldsymbol{a}_i + \boldsymbol{h}_i^\star \right) + \sum_{k\in N_{Ci}} \boldsymbol{C}_{ki}^T \left(\boldsymbol{M}_k\boldsymbol{a}_k + \boldsymbol{h}_k^\star \right) \right] = \boldsymbol{0}. \tag{C.9}$$

For any terminal body in the tree, the sum over the child nodes N_{Ci} vanishes. We can therefore directly solve (3.36) for the unknown $\ddot{\boldsymbol{q}}_i$ after substituting the equation for $\boldsymbol{a}_i$ (2.75):

$$\boldsymbol{J}_{Ji}^T \left(\boldsymbol{M}_i \left(\boldsymbol{C}_{ip}\boldsymbol{a}_{p(i)} + \boldsymbol{J}_{Ji}\ddot{\boldsymbol{q}}_i + \boldsymbol{b}_i \right) + \boldsymbol{h}_i^\star \right) = \boldsymbol{0}, \tag{C.10}$$

$$\Rightarrow \quad \ddot{\boldsymbol{q}}_i = \underbrace{-\left(\boldsymbol{J}_{Ji}^T\boldsymbol{M}_i\boldsymbol{J}_{Ji} \right)^{-1} \boldsymbol{J}_{Ji}^T\boldsymbol{M}_i\boldsymbol{C}_{ip}}_{=:\boldsymbol{P}_i} \boldsymbol{a}_{p(i)} \underbrace{-\left(\boldsymbol{J}_{Ji}^T\boldsymbol{M}_i\boldsymbol{J}_{Ji} \right)^{-1} \boldsymbol{J}_{Ji}^T \left(\boldsymbol{M}_i\boldsymbol{b}_i + \boldsymbol{h}_i^\star \right)}_{=:\boldsymbol{p}_i}, \tag{C.11}$$

$$\Rightarrow \ddot{\boldsymbol{q}}_i = \boldsymbol{P}_i\boldsymbol{a}_{p(i)} + \boldsymbol{p}_i. \tag{C.12}$$

From this we can write $\boldsymbol{a}_i$ as a function of the parent acceleration $\boldsymbol{a}_{p(i)}$, which we can substitute into (3.36) for bodies which have only leaf nodes as children:

$$\begin{aligned} &\boldsymbol{J}_{Ji}^T \Big[\left(\boldsymbol{M}_i\boldsymbol{a}_i + \boldsymbol{h}_i^\star \right) \Big] \\ &+ \sum_{k\in N_{Ci}} \boldsymbol{C}_{ki}^T \left(\boldsymbol{M}_k \left(\boldsymbol{C}_{kp(k)}\boldsymbol{a}_{p(k)} + \boldsymbol{J}_{Jk}\ddot{\boldsymbol{q}}_k + \boldsymbol{b}_k \right) + \boldsymbol{h}_k^\star \right) \Bigg] = \boldsymbol{0} \end{aligned} \tag{C.13}$$

With $\boldsymbol{a}_{p(k)} = \boldsymbol{a}_i$, rearranging yields:

$$\boldsymbol{J}_{Ji}^T \left\{ \underbrace{\left(\boldsymbol{M}_i + \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{K}_k \right)}_{=:\overline{\boldsymbol{M}}_i} \boldsymbol{a}_i + \underbrace{\left(\boldsymbol{h}_i^\star + \sum_{k \in N_{Ci}} \boldsymbol{C}_{ki}^T \boldsymbol{k}_k \right)}_{=:\overline{\boldsymbol{h}^\star}_i} \right\} = \boldsymbol{0}, \tag{C.14}$$

$$\boldsymbol{K}_k =: \boldsymbol{M}_k \left(\boldsymbol{C}_{kp(k)} + \boldsymbol{J}_{Jk} \boldsymbol{P}_k \right), \tag{C.15}$$

$$\boldsymbol{k}_k =: \boldsymbol{M}_k \left(\boldsymbol{J}_{Jk} \boldsymbol{P}_k + \boldsymbol{b}_k \right) + \boldsymbol{h}_k^\star. \tag{C.16}$$

The key observation for the recursive calculation of $\ddot{\boldsymbol{q}}$ is the fact that the previous equations have the same structure as (3.36) for a leaf node. We can therefore recursively apply the previous steps, if we use the modified mass matrix $\overline{\boldsymbol{M}}_i$ and force vector $\overline{\boldsymbol{h}^\star}_i$ (cf. algorithm 4).

Bibliography

Acary, V., and B. Brogliato. 2008. *Numerical Methods for Nonsmooth Dynamical Systems.* Edited by F. Pfeiffer and P. Wriggers. Lecture Notes in Applied and Computational Mechanics. Springer. doi:10.1007/978-3-540-75392-6.

Aftab, Z., T. Robert, and P.-B. Wieber. 2012. "Predicting multiple step placements for human balance recovery tasks." *Journal of Biomechanics* 45 (16): 2804–2809. doi:10.1016/j.jbiomech.2012.08.038.

Altenbach, J., and H. Altenbach. 1994. *Einführung in die Kontinuumsmechanik.* Teubner-Studienbücher. Teubner B.G. GmbH. ISBN: 9783519030966.

Anitescu, M. 2006. "Optimization-based simulation of nonsmooth rigid multibody dynamics." *Mathematical Programming* 105 (1): 113–143. doi:10.1007/s10107-005-0590-7.

Aoi, Shinya, and K. Tsuchiya. 2006. "Stability analysis of a simple walking model driven by an oscillator with a phase reset using sensory feedback." *Robotics, IEEE Transactions on* 22 (2): 391–397. doi:10.1109/TRO.2006.870671.

Arbulú, M., A. Kheddar, and E. Yoshida. 2010. "An approach of generic solution for humanoid stepping over motion." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids).* doi:10.1109/ICHR.2010.5686345.

Arnold, M. 2009. "Numerical methods for simulation in applied dynamics." In *Simulation Techniques for Applied Dynamics,* edited by M. Arnold and W. Schiehlen, 507:191–246. CISM International Centre for Mechanical Sciences. Springer Vienna. ISBN: 978-3-211-89547-4. doi:10.1007/978-3-211-89548-1_5.

———. 2013. "A recursive multibody formalism for systems with small mass and inertia terms." *Mechanical Sciences* 4 (1): 221–231. doi:10.5194/ms-4-221-2013.

Ascher, U. M., H. Chin, L. R. Petzold, and S. Reich. 1995. "Stabilization of Constrained Mechanical Systems with DAEs and Invariant Manifolds." *Mechanics of Structures and Machines* 23 (2): 135–157. doi:10.1080/08905459508905232.

Ascher, U. M., H. Chin, and S. Reich. 1994. "Stabilization of DAEs and invariant manifolds." *Numerische Mathematik* 67 (2): 131–149. doi:10.1007/s002110050020.

Ayusawa, K., and Y. Nakamura. 2012. "Fast inverse kinematics algorithm for large DOF system with decomposed gradient computation based on recursive formulation of equilibrium." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2012.6385780.

Azevedo, C., P. Poignet, and B. Espiau. 2002. "Moving horizon control for biped robots without reference trajectory." In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 3:2762–2767. doi:10.1109/ROBOT.2002.1013650.

Bae, D.-S., and E. J. Haug. 1987a. "A Recursive Formulation for Constrained Mechanical System Dynamics: Part I. Open Loop Systems." *Mechanics of Structures and Machines* 15 (3): 359–382. doi:10.1080/08905458708905124.

———. 1987b. "A Recursive Formulation for Constrained Mechanical System Dynamics: Part II. Closed Loop Systems." *Mechanics of Structures and Machines* 15 (4): 481–506. doi:10.1080/08905458708905130.

Bae, D.-S., J. G. Kuhl, and E. J. Haug. 1988. "A Recursive Formulation for Constrained Mechanical System Dynamics: Part III. Parallel Processor Implementation." *Mechanics of Structures and Machines* 16 (2): 249–269. doi:10.1080/08905458808960263.

Baerlocher, P., and R. Boulic. 1998. "Task-priority formulations for the kinematic control of highly redundant articulated structures." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.1998.724639.

Ball, R.S. 1876. "The theory of screws: A study in the dynamics of a rigid body." *Mathematische Annalen* 9 (4): 541–553. doi:10.1007/BF01442479.

Baraff, D. 1996. "Linear-time dynamics using Lagrange multipliers." In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. SIGGRAPH '96. New York, NY, USA: ACM. doi:10.1145/237170.237226.

Baudouin, L., N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida. 2011. "Real-time replanning using 3D environment for humanoid robot." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids),* 584–589. doi:10.1109/Humanoids.2011.6100844.

Baumgarte, J. 1972. "Stabilization of constraints and integrals of motion in dynamical systems." *Computer Methods in Applied Mechanics and Engineering* 1 (1): 1–16. doi:10.1016/0045-7825(72)90018-7.

Baur, J., S. Dendorfer, J. Pfaff, C. Schütz, T. Buschmann, and H. Ulbrich. 2014. "Experimental Friction Identification in Robot Drives." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. (to appear).

Baur, J., J. Pfaff, C. Schuetz, and H. Ulbrich. 2013. "Dynamic modeling and realization of an agricultural manipulator." In *Proceedings of XV International Symposium on Dynamic Problems of Mechanics, DINAME.*

Baur, J., J. Pfaff, H. Ulbrich, and T. Villgrattner. 2012. "Design and development of a redundant modular multipurpose agricultural manipulator." In *Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on,* 823–830. doi:10.1109/AIM.2012.6265928.

Bayazitoglu, Y. O., and M. A. Chace. 1973. "Methods for Automated Dynamic Analysis of Discrete Mechanical Systems." *J. Appl. Mech.* 40:809–811. doi:10.1115/1.3423095.

Behnisch, M. 2012. "Task Level Motion Planning – Integrating Local Robot Control and Global Sampling." PhD diss., Universität Bielefeld. http://pub.uni-bielefeld.de/publication/2532359.

Behnisch, M., R. Haschke, and M. Gienger. 2010. "Task space motion planning using reactive control." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2010.5651285.

Behnisch, M., R. Haschke, H. Ritter, and M. Gienger. 2011. "Deformable trees - exploiting local obstacle avoidance." In *2011 11th IEEE-RAS International Conference on Humanoid Robots.* doi:10.1109/Humanoids.2011.6100876.

Bekey, G. A. 2005. *Autonomous robots: from biological inspiration to implementation and control.* The MIT Press.

Bellman, R. 1954. "Some Problems in the Theory of Dynamic Programming." *Econometrica* 22 (1): 37–48. ISSN: 00129682.

Bellman, R., and E. Lee. 1984. "History and development of dynamic programming." *Control Systems Magazine, IEEE.* doi:10.1109/MCS.1984.1104824.

Berns, K., W. Ilg, M. Deck, J. Albiez, and R. Dillmann. 1999. "Mechanical Construction and Computer Architecture of the Four-Legged Walking Machine BISAM." *IEEE/ASME Trans. Mechatron.* 4 (1): 32–38.

Bessonnet, G., S. Chessé, and P. Sardain. 2004. "Optimal Gait Synthesis of a Seven-Link Planar Biped." *The Intl Journal of Robotics Research* 23 (10-11): 1059–1073. doi:10.1177/0278364904047393.

Betsch, P., and S. Leyendecker. 2006. "The discrete null space method for the energy consistent integration of constrained mechanical systems. Part II: multibody dynamics." *International Journal for Numerical Methods in Engineering* 67 (4): 499–552. doi:10.1002/nme.1639.

Blickhan, R. 1989. “The Spring Mass Model for Running And Hopping.” *J. of Biomechanics* 22:1217–1227.

Bonnans, J.F., J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. 1997. *Numerical Optimization: Theoretical and Practical Aspects.* Springer.

Brandl, H., R. Johanni, and M. Otter. 1986. “A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix.” In *IFAC/IFIP/IMACS Symposium on Theory of Robots,* 95–100.

———. 1987. “An algorithm for the simulation of multibody systems with kinematic loops.” In *Theory of Robots. Selected Papers from the IFAC/IFIP/IMACS Symposium, Vienna/Austria.* Pergamon Press.

Bremer, H. 1982. “Kinetik Starr-elastischer Mehrkörpersysteme.” (German). Habilitation diss., Technische Universität München.

———. 1987. “Subsystem computation of large mechanical systems.” In *Proceedings of the 7th World Congress on Theory of Machines and Mechanisms,* 1:413–416.

———. 1988. *Dynamik und Regelung mechanischer Systeme.* Wiesbaden: B.G. Teubner Verlag.

———. 1993. “Das Jourdainsche Prinzip.” *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 73 (3): 184–187. doi:10.1002/zamm.19930730317.

———. 2008. *Elastic Multibody Dynamics.* 35:29–58. Intelligent Systems, Control, And Automation: Science And Engineering. Springer Netherlands. doi:10.1007/978-1-4020-8680-9.

Bremer, H., and Ch. Glocker. 2000. “Contact in Multibody Systems.” *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 80 (S1): 33–36. doi:10.1002/zamm.20000801309.

Bremer, H., and F. Pfeiffer. 1988. *Elastische Mehrkörpersysteme.* (German). Wiesbaden: B.G. Teubner Verlag.

Brown, B., and G. Zeglin. 1998. “The bow leg hopping robot.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA).* doi:10.1109/ROBOT.1998.677072.

Büschges, A., H. Scholz, and A. El Manira. 2011. “New Moves in Motor Control.” *Current Biology* 21 (13): R513–R524. doi:10.1016/j.cub.2011.05.029.

Buschmann, T. 2010. “Simulation and Control of Biped Walking Robots.” PhD diss., Technische Universität München. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20101201-997204-1-6.

Buschmann, T., A. Ewald, A. von Twickel, and A. Büschges. 2014. “Controlling Legs for Locomotion — Insights from Robotics and Neurobiology.” (to appear), *Journal of Bioinspiration and Biomimetics.*

Buschmann, T., A. Ewald, H. Ulbrich, and A. Buschges. 2012a. “Event-based walking control – From neurobiology to biped robots.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2012.6385783.

Buschmann, T., V. Favot, S. Lohmeier, M. Schwienbacher, and H. Ulbrich. 2011. “Experiments in fast biped walking.” In *Proc IEEE Intl Conf on Mechatronics (ICM)*. doi:10.1109/ICMECH.2011.5971235.

Buschmann, T., S. Lohmeier, M. Bachmayer, H. Ulbrich, and F. Pfeiffer. 2007. “A Collocation Method for Real-Time Walking Pattern Generation.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2007.4813841.

Buschmann, T., S. Lohmeier, M. Schwienbacher, V. Favot, H. Ulbrich, F. Von Hundelshausen, G. Rohe, and H. -J Wuensche. 2010. “Walking in unknown environments – A step towards more autonomy.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids),* 237–244. doi:10.1109/ICHR.2010.5686338.

Buschmann, T., S. Lohmeier, and H. Ulbrich. 2009. “Biped walking control based on hybrid position/force control.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2009.5354643.

Buschmann, T., S. Lohmeier, H. Ulbrich, and F. Pfeiffer. 2005. “Optimization based gait pattern generation for a biped robot.” In *Humanoid Robots, 2005 5th IEEE-RAS Intl Conf on,* 98–103. doi:10.1109/ICHR.2005.1573552.

———. 2006. “Dynamics simulation for a biped robot: modeling and experimental verification.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2006.1642105.

———. 2008. “Regelung des humanoiden Laufroboters LOLA,” in *Tagungsband ROBOTIK 2008.* (German).

Buschmann, T., M. Schwienbacher, V. Favot, A. Ewald, and H. Ulbrich. 2012b. “The Biped Walking Robot Lola – Hardware Design and Walking Control –.” *Journal of the Robotics Society of Japan* 30 (4): 363–366.

Buschmann, T., R. Wittmann, M. Schwienbacher, and H. Ulbrich. 2012c. “A method for real-time Kineto-Dynamic trajectory generation.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/HUMANOIDS.2012.6651519.

Cheng, F.T., T.H. Chen, and Y.Y. Sun. 1994. "Resolving manipulator redundancy under inequality constraints." *Robotics and Automation, IEEE Transactions on* 10 (1): 65–71. doi:10.1109/70.285587.

Cheng, G., S.-H. Hyon, J. Morimoto, A. Ude, G. Colvin, W. Scroggin, and S.C. Jacobsen. 2006. "CB: A Humanoid Research Platform for Exploring NeuroScience." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2006.321382.

Chestnutt, J., J. Kuffner, K. Nishiwaki, and S. Kagami. 2003. "Planning Biped Navigation Strategies in Complex Environments." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*.

Chestnutt, J., Y. Takaoka, M. Doi, K. Suga, and S. Kagami. 2010. "Safe adjustment regions for legged locomotion paths." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2010.5686829.

Chevallereau, C., G. Abba, Y. Aoustin, F. Plestan, E.R. Westervelt, C. Canudas-De-Wit, and J.W. Grizzle. 2003. "RABBIT: a testbed for advanced control theory." *Control Systems, IEEE* 23 (5): 57–79. doi:10.1109/MCS.2003.1234651.

Chevallereau, C., D. Djoudi, and J. Grizzle. 2008. "Stable Bipedal Walking With Foot Rotation Through Direct Regulation of the Zero Moment Point." *IEEE Trans. Robot. Automat.* 24:390–401. doi:10.1109/TRO.2007.913563.

Chiaverini, S. 1997. "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators." *Robotics and Automation, IEEE Transactions on* 13 (3): 398–410. doi:10.1109/70.585902.

Cho, K.-B., and J.-H. Oh. 2008. "Running pattern generation of humanoid biped with a fixed point and its realization." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2008.4755968.

Clauberg, J. P. 2013. "Methoden zur parallelen Berechnung von nicht-glatten dynamischen Systemen." (German). PhD diss., Technische Universität München. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20130122-1111045-0-3.

Collins, S.H., and A. Ruina. 2005. "A Bipedal Walking Robot with Efficient and Human-Like Gait." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*, 1983–1988. doi:10.1109/ROBOT.2005.1570404.

Craig, J.J. 1989. *Introduction to Robotics, Mechanics & Control.* Addison-Wesley Publishing Company, Inc.

Craig, J.J., and M.H. Raibert. 1979. "A systematic method of hybrid position/force control of a manipulator." In *Computer Software and Applications Conference, 1979. Proceedings. COMPSAC 79. The IEEE Computer Society's Third International,* 446–451. doi:10.1109/CMPSAC.1979.762539.

Critchley, J. H., K. Anderson, and A. Binani. 2007. "An Efficient Multibody Divide And Conquer Algorithm." In *Proc of the ASME Intl Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE)*.

D'Alembert, J. 1743. *Traité de Dynamique.* Paris.

Dariush, B., G. Bin Hammam, and D. Orin. 2010. "Constrained resolved acceleration control for humanoids." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2010.5653673.

De Schutter, J., T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx. 2007. "Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty." *The International Journal of Robotics Research* 26 (5): 433–455. doi:10.1177/0278364907078091O7.

Decre, W., H. Bruyninckx, and J. De Schutter. 2013. "Extending the iTaSC Constraint-based Robot Task Specification Framework to Time-Independent Trajectories and User-Configurable Task Horizons." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ICRA.2013.6630835.

Decre, W., R. Smits, H. Bruyninckx, and J. De Schutter. 2009. "Extending iTaSC to support inequality constraints and non-instantaneous task specification." In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 964–971. doi:10.1109/ROBOT.2009.5152477.

Deliagina, T.G., P.V. Zelenin, and G.N. Orlovsky. 2012. "Physiological and circuit mechanisms of postural control." *Current Opinion in Neurobiology* 22 (4): 646–652. doi:http://dx.doi.org/10.1016/j.conb.2012.03.002.

Denk, J., and G. Schmidt. 2003. "Synthesis of walking primitive databases for biped robots in 3D-environments." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2003.1241778.

Deuflhard, P. 2011. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms.* Springer.

Diedam, H., D. Dimitrov, P. -B Wieber, K. Mombaur, and M. Diehl. 2008. "Online walking gait generation with adaptive foot positioning through Linear Model Predictive control." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2008.4651055.

Diehl, M., H.-J. Ferreau, and N. Haverbeke. 2009. “Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation.” In *Nonlinear Model Predictive Control SE - 32,* edited by L. Magni, D.M. Raimondo, and F. Allgöwer, 384:391–417. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg. doi:10.1007/978-3-642-01094-1_32.

Eberhard, P., and W. Schiehlen. 2005. “Computational Dynamics of Multibody Systems: History, Formalisms, and Applications.” *Journal of Computational and Nonlinear Dynamics* 1 (1): 3–12. doi:10.1115/1.1961875.

El Khoury, A., F. Lamiraux, and M. Taix. 2013. “Optimal motion planning for humanoid robots.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. IEEE. doi:10.1109/ICRA.2013.6631013.

Endo, G., J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng. 2008. “Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot.” *The International Journal of Robotics Research* 27:213–228.

Escande, A., N. Mansard, and P. -B Wieber. 2010. “Fast resolution of hierarchized inverse kinematics with inequality constraints.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2010.5509953.

Escande, A., S. Miossec, M. Benallegue, and A. Kheddar. 2014. “A Strictly Convex Hull for Computing Proximity Distances With Continuous Gradients.” *Robotics, IEEE Transactions on* PP (99): 1–13. doi:10.1109/TRO.2013.2296332.

Escande, A., S. Miossec, and A. Kheddar. 2007. “Continuous gradient proximity distance for humanoids free-collision optimized-postures.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2007.4813867.

Euler, L. 1776. “Nova methods motum corporum rigidarum determinandi.” *Novi Commentarii Academiae Scientiarum Petropolitanae* 20:208–238.

Ewald, A., and T. Buschmann. 2013. “Online Trajectory Replanning for Biped Robots Based on Foot Step Position Modification.” In *Advances in Nano, Biomechanics, Robotics, and Energy Research,* 1:155–163. Seoul, Korea.

Ewald, A., J. Mayet, T. Buschmann, and H. Ulbrich. 2012. “Generating Smooth Trajectories Free from Overshoot for Humanoid Robot Walking Pattern Replanning,” in *Autonomous Mobile Systems (AMS)*.

Falco, P., and C. Natale. 2011. “On the Stability of Closed-Loop Inverse Kinematics Algorithms for Redundant Robots.” *Robotics, IEEE Transactions on* 27 (4): 780–784. doi:10.1109/TRO.2011.2135210.

Featherstone, R. 1983. “The Calculation of Robot Dynamics Using Articulated-Body Inertias.” *The International Journal of Robotics Research* 2 (1): 13–30. doi:10.1177/027836498300200102.

———. 1994. “Accurate trajectory transformations for redundant and nonredundant robots.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.1994.351189.

———. 1999a. “A Divide-and-Conquer Articulated-Body Algorithm for Parallel O(log(n)) Calculation of Rigid-Body Dynamics. Part 1: Basic Algorithm.” *The International Journal of Robotics Research* 18 (9): 867–875. doi:10.1177/02783649922066619.

———. 1999b. “A Divide-and-Conquer Articulated-Body Algorithm for Parallel O(log(n)) Calculation of Rigid-Body Dynamics. Part 2: Trees, Loops, and Accuracy.” *The International Journal of Robotics Research* 18 (9): 876–892. doi:10.1177/02783649922066628.

———. 2005. “Efficient Factorization of the Joint-Space Inertia Matrix for Branched Kinematic Trees.” *The Intl Journal of Robotics Research* 24 (6): 487–500. doi:10.1177/0278364905054928.

———. 2008. *Rigid Body Dynamics Algorithms.* Kluwer international series in engineering and computer science: Robotics. Springer. ISBN: 9780387743158.

Featherstone, R., and D. Orin. 2000. “Robot dynamics: equations and algorithms.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2000.844153.

Fijany, A., and A. K. Bejczy. 1991. “Parallel algorithms and architecture for computation of manipulator forward dynamics.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 1156–1162 vol.2. doi:10.1109/ROBOT.1991.131765.

Fijany, A., I. Sharf, and G. M. T. D’Eleuterio. 1995. “Parallel O(log N) algorithms for computation of manipulator forward dynamics.” *Robotics and Automation, IEEE Transactions on* 11 (3): 389–400. doi:10.1109/70.388780.

Förg, M. 2007. *Mehrkörpersysteme mit mengenwertigen Kraftgesetzten – Theorie und Numerik.* Fortschrittberichte VDI, Reihe 20 411. (German). Düsseldorf: VDI-Verlag. ISBN: 978-3-18-341120-7.

Förg, M., T. Geier, L. Neumann, and H. Ulbrich. 2006. “R-Factor Strategies for the Augmented Lagrangian Approach in Multi-Body Contact Mechanics.” In *III European Conf on Computational Mechanics, Solids, Structures and Coupled Problems in Engineering.*

Fujisawa, N., T. Masuda, H. Inaoka, Y. Fukuoka, A. Ishida, and H. Minamitani. 2005. "Human standing posture control system depending on adopted strategies." *Medical and Biological Engineering and Computing* 43 (1): 107–114. doi:10.1007/BF02345130.

Fukuoka, Y., and H. Kimura. 2003. "Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Based on Biological Concepts." *The International Journal of Robotics Research* 22 (3-4): 187–202.

———. 2009. "Dynamic locomotion of a biomorphic quadruped Tekken robot using various gaits: walk, trot, free-gait and bound." *Journal of Applied Physiology* 6 (1): 1–9.

Full, R. J., T. Kubow, J. Schmitt, P. Holmes, and D. Koditschek. 2002. "Quantifying Dynamic Stability and Maneuverability in Legged Locomotion." *Integrative and Comparative Biology* 42 (1): 149–157.

Furusho, J., and A. Sano. 1990. "Sensor-based Control of a Nine-link Biped." *The Intl Journal of Robotics Research* 9 (2): 83–98.

Gattringer, H. 2011. *Starr-elastische Robotersysteme – Theorie und Anwendungen.* (German). Springer Berlin / Heidelberg. doi:10.1007/978-3-642-22828-5.

Gattringer, H., and H. Bremer. 2005. "Ein O(n) Verfahren mit Kontaktbedingungen zur Modellierung einer zweibeinigen Laufmaschine." (German), *PAMM* 5 (1): 199–200. doi:10.1002/pamm.200510077.

Gauss, C. F. 1829. "Über ein neues allgemeines Grundgesetz der Mechanik." (German), *Journal für die reine und angewandte Mathematik* 4:232–235.

Geng, T., B. Porr, and F. Wörgötter. 2006. "Fast Biped Walking with a Sensor-driven Neuronal Controller and Real-time Online Learning." *The International Journal of Robotics Research* 25 (3): 243–259. doi:10.1177/0278364906063822.

Gerdts, M. 2011. *Optimal Control of ODEs and DAEs.* De Gruyter.

Gienger, M., M. Toussaint, and C. Goerick. 2008a. "Task maps in humanoid robot manipulation." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2008.4651027.

Gienger, M., M. Toussaint, N. Jetchev, A. Bendig, and C. Goerick. 2008b. "Optimization of fluent approach and grasp motions." *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2008.4755940.

Ginsberg, J. H. 1998. *Advanced Engineering Dynamics.* Cambridge University Press.

Glocker, Ch. 1998. “The Principles of d’Alembert, Jourdain, and Gauss in Nonsmooth Dynamics Part I: Scleronomic Multibody Systems.” *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 78 (1): 21–37. doi:10.1002/(SICI)1521-4001(199801)78:1<21::AID-ZAMM21>3.0.CO;2-W.

———. 2000. “Set-Valued Force Laws in Rigid Body Dynamics.” Habilitation diss., Technische Universität München.

———. 2006. “An Introduction to Impacts.” *Nonsmooth Mechanics of Solids. CISM Courses and Lectures* 485:45–102.

Goedecker, S., and A. Hoisie. 2001. *Performance Optimization of Numerically Intensive Codes.* Society for Industrial / Applied Mathematics. doi:10.1137/1.9780898718218.

Goldenberg, A.A., B. Benhabib, and R.G. Fenton. 1985. “A complete generalized solution to the inverse kinematics of robots.” *Robotics and Automation, IEEE Journal of* 1 (1): 14–20. doi:10.1109/JRA.1985.1086995.

Goswami, A., and V. Kallem. 2004. “Rate of change of angular momentum and balance maintenance of biped robots.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2004.1308858.

Grill, M. 2013. “Comparison of Forward Dynamics Algorithms for Rigid Multibody Systems.” Term paper (Semesterarbeit), Lehrstuhl für Angewandte Mechanik, Technische Universität München.

Guan, Y., E.S. Neo, K. Yokoi, and K. Tanie. 2006. “Stepping over obstacles with humanoid robots.” *IEEE Transactions on Robotics* 22 (5): 958–973. doi:10.1109/TRO.2006.878962.

Guan, Y., K Yokoi, and K. Tanie. 2004. “Feasibility of humanoid robots stepping over obstacles.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2004.1389341.

Guan, Y., K. Yokoi, and K. Tanie. 2005. “Feasibility: Can Humanoid Robots Overcome Given Obstacles?” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2005.1570255.

Gupta, K., and K. Kazerounian. 1985. “Improved numerical solutions of inverse kinematics of robots.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.1985.1087237.

Hamel, G. 1949. *Theoretische Mechanik.* (German). Springer, Berlin/Heidelberg/New York.

Harada, K., S. Kajita, K. Kaneko, and H. Hirukawa. 2004. "An Analytical Method on Real-time Gait Planning for a Humanoid Robot." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*.

Hashimoto, K., A. Hayashi, T. Sawato, Y. Yoshimura, T. Asano, K. Hattori, Y. Sugahara, Hun-ok Lim, and A. Takanishi. 2009. "Terrain-adaptive control to reduce landing impact force for human-carrying biped robot." In *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on,* 174–179. doi:10.1109/AIM.2009.5230020.

Hawker, G., and M. Buehler. 2000. "Quadruped trotting with passive knees: design, control, and experiments." In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 3:3046–3051.

Hildebrandt, A.-C., R. Wittmann, D. Wahrmann, A. Ewald, and T. Buschmann. 2014. "Real-Time 3D Collision Avoidance for Biped Robots." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. (submitted).

Hiller, M., A. Kecskemethy, and C. Woernle. 1986. "A loop-based kinematical analysis of complex mechanisms." In *Proc of the ASME Design Engineering Technical Conference,* 184.

Hiller, M., and C. Woernle. 1988. "The characteristic pair of joints – an effective approach for the inverse kinematic problem of robots." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.1988.12166.

Hippmann, G. 2008. *Introduction to SIMPACK's MBS-Formalism.* Technical report. SIMPACK AG.

Hirai, K., M. Hirose, Y. Haikawa, and T. Takenaka. 1998. "The development of Honda humanoid robot." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.1998.677288.

Hirose, M., and K. Ogawa. 2006. "Honda humanoid robots development." *Phil. Trans. R. Soc. A* 365:11–19.

Hirukawa, H., S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa. 2006. "A universal stability criterion of the foot contact of legged robots - adios ZMP." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2006.1641995.

Hobbelen, D. G. E., and M. Wisse. 2009. "Active Lateral Foot Placement For 3D Stabilization Of A Limit Cycle Walker Prototype." *International Journal of Humanoid Robotics* 06 (01): 93–116. doi:10.1142/S0219843609001632.

Hobbelen, D., T. de Boer, and M. Wisse. 2008. "System overview of bipedal robots Flame and TUlip: Tailor-made for Limit Cycle Walking." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2008.4650728.

Hodgins, J.K., and M.H. Raibert. 1991. "Adjusting step length for rough terrain locomotion." *Robotics and Automation, IEEE Transactions on* 7 (3): 289–298. doi:10.1109/70.88138.

Hof, A.L., M.G.J. Gazendam, and Sinke W.E. 2005. "The condition for dynamic stability." *Journal of Biomechanics* 38:1–8. doi:10.1016/j.jbiomech.2004.03.025.

Hof, At L. 2008. "The 'extrapolated center of mass' concept suggests a simple control of balance in walking." *Human Movement Science* 27 (1): 112–125. doi:10.1016/j.humov.2007.08.003.

Hollerbach, J., and Ki Suh. 1985. "Redundancy resolution of manipulators through torque optimization." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.1985.1087285.

Holm, D. D. 2008a. *Geometric Mechanics, Part I: Dynamics and Symmetry.* Imperial College Press, London.

———. 2008b. *Geometric MeRolling, Part II: Rotating, Translating and Rolling.* Imperial College Press, London.

Horak, Fay B. 1987. "Clinical Measurement of Postural Control in Adults." *Physical Therapy* 67 (12): 1881–1885.

Horak, F.B., and L.M. Nashner. 1986. "Central programming of postural movements: adaptation to altered support-surface configurations." *Journal of Neurophysiology* 55 (6): 1369–1381.

Hutter, M., C. Gehring, M. Bloesch, M.A. Hoepflinger, C.D. Remy, and R. Siegwart. 2012. "StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion." In *Proc Intl Conf Climbing and Walking Robots (CLAWAR),* 1–8.

Hyon, S.H., and T. Mita. 2002. "Development of a biologically inspired hopping robot — "Kenken"." In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 4:3984–3991.

Intel Corporation. 2011. *Intel®64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture.*

Jacobson, D. H. 1968a. "Differential dynamic programming methods for solving bang-bang control problems." *Automatic Control, IEEE Transactions on* 13 (6): 661–675. doi:10.1109/TAC.1968.1099026.

Jacobson, D. H. 1968b. “New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach.” *Journal of Optimization Theory and Applications* 2 (6): 411–440. doi:10.1007/BF00925746.

———. 1968c. “Second-order and Second-variation Methods for Determining Optimal Control: A Comparative Study using Differential Dynamic Programming.” *International Journal of Control* 7 (2): 175–196. doi:10.1080/00207176808905594.

Jacobson, D. H., and D. Q. Mayne. 1970. *Differential Dynamic Programming.* New York, New York, USA: American Elsevier Publishing Company, Inc.

Jiménez, P., F. Thomas, and C. Torras. 2001. “3D collision detection: a survey.” *Computers & Graphics* 25:269–285.

Jourdain, P.E.B. 1909. “Note on an analogue of Gauss’ principle of least constraints.” *Quarterly Journal of Pure and Applied Mathematics* 40:153–197.

Kagami, S., T. Kitagawa, K. Nishiwaki, T. Sugihara, M. Inaba, and H. Inoue. 2002. “A Fast Dynamically Equilibrated Walking Trajectory Generation Method of Humanoid Robot.” *Autonomous Robots* 12:71–82. doi:10.1023/A:1013210909840.

Kajita, S., F. Kanehiro, F. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. 2003a. “Resolved Momentum Control: Humanoid Motion Planning based on the Linear and Angular Momentum.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*.

Kajita, S., F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. 2003b. “Biped walking pattern generation by using preview control of zero-moment point.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* vol. 2, 1620–1626 vol.2.

Kajita, S., T. Nagasaki, K. Kaneko, K. Yokoi, and K. Tanie. 2005. “A Running Controller of Humanoid Biped HRP-2LR.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 616–622.

Kajita, S., and K. Tani. 1995. “Experimental study of biped dynamic walking in the linear inverted pendulum mode.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.1995.525693.

Kane, T. R. 1961. “Dynamics of Nonholonomic Systems.” *Journal of Applied Mechanics* 28 (4): 574–578. doi:10.1115/1.3641786.

Kane, T.R., and D.A. Levinson. 1983. “Multibody Dynamics.” *Transactions of the ASME Journal of Applied Mechanics* 50:1071–1078.

Kanehiro, F., W. Suleiman, K. Miura, M. Morisawa, and E. Yoshida. 2009. "Feasible pattern generation method for humanoid robots." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids),* 542–548. doi:10.1109/ICHR.2009.5379520.

Kaneko, K., F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. 2004. "Humanoid robot HRP-2." In *Proc IEEE Intl Conf Robotics and Automation (ICRA).* doi:10.1109/ROBOT.2004.1307969.

Kang, H.J., K. Hashimoto, H. Kondo, K. Hattori, K. Nishikawa, Y. Hama, Hun-ok Lim, A. Takanishi, K. Suga, and K. Kato. 2010. "Realization of biped walking on uneven terrain by new foot mechanism capable of detecting ground surface." In *Proc IEEE Intl Conf Robotics and Automation (ICRA).* doi:10.1109/ROBOT.2010.5509348.

Kanoun, O. 2011. "Real-time prioritized kinematic control under inequality constraints for redundant manipulators." In *Proc Robotics Science and Systems (RSS).* Los Angeles, CA, USA.

Kecskemethy, A., and M. Hiller. 1993. "Modellierung der Dynamik komplexer Mehrkörpersysteme mit Hilfe von kinematischen Übertragungselementen." (German), *Archive of Applied Mechanics* 63 (6): 386–401. doi:10.1007/BF00805739.

Khalil, H. K. 2000. *Nonlinear Systems.* 3rd ed. Pearson Education.

Khatib, O. 1983. "Dynamic Control of Manipulators in Operational Space." In *Proc. of the Sixth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators,* 1128–1131. New Delhi, India.

———. 1987. "A unified approach for motion and force control of robot manipulators: The operational space formulation." *Robotics and Automation, IEEE Journal of* 3 (1): 43–53. doi:10.1109/JRA.1987.1087068.

Klein, C.A., and C.-H. Huang. 1983. "Review of pseudoinverse control for use with kinematically redundant manipulators." *Systems, Man and Cybernetics, IEEE Transactions on* SMC-13 (2): 245–250. doi:10.1109/TSMC.1983.6313123.

Kuffner, J., K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. 2002. "Self-collision detection and prevention for humanoid robots." In *Proc IEEE Intl Conf Robotics and Automation (ICRA).* doi:10.1109/ROBOT.2002.1013569.

Lagrange, J.-L. 1788. *Méchanique Analytique.* L'Académie Royal des Sciences, Paris.

Larsen, E., S. Gottschalk, M. C. Lin, and D. Manocha. 1999. *Fast Proximity Queries with Swept Sphere Volumes (TR99-018).* Technical report. Department of Computer Science, UNC Chapel Hill.

Larsen, E., S. Gottschalk, M.C. Lin, and D. Manocha. 2000. "Fast distance queries with rectangular swept sphere volumes." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2000.845311.

Lasdon, L., S. Mitter, and A. Waren. 1967. "The conjugate gradient method for optimal control problems." *IEEE Transactions on Automatic Control* 12 (2): 132–138. doi:10.1109/TAC.1967.1098538.

Lathrop, R.H. 1986. "Constrained (closed-loop) robot simulation by local constraint propagation." In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 3:689–694. doi:10.1109/ROBOT.1986.1087601.

LaValle, S.M. 1998. *Rapidly-Exploring Random Trees – A New Tool for Path Planning.* Technical report. Iowa State University.

———. 2006. *Planning Algorithms.* Cambridge University Press.

Leine, R. I., and H. Nijmeijer. 2004. *Dynamics and Bifurcations of Non-smooth Mechanical Systems.* Springer.

Li, W., and E. Todorov. 2004. "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems." In *ICINCO,* 222–229.

Liégeois, A. 1977. "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms." *Systems, Man and Cybernetics, IEEE Transactions on* 7 (12): 868–871. doi:10.1109/TSMC.1977.4309644.

Lim, H.O., S. A. Setiawan, and A. Takanishi. 2001. "Balance and Impedance Control for Biped Humanoid Robot Locomotion." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*.

Lin, M. C., and S. Gottschalk. 1998. "Collision detection between geometric models: a survey." In *In Proc. of IMA Conf on Mathematics of Surfaces,* 37–56.

Lipkin, H. 2005. "A note on Denavit-Hartenberg notation in robotics." In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol 7, Pts A and B,* 921–926.

Löffler, K., M. Gienger, and F. Pfeiffer. 2002. "Model Based Control of a Biped Robot." In *Proc. Int. Workshop on Advanced Motion Control (AMC),* 443–448. Maribor, Slovenia.

Löffler, K., M. Gienger, F. Pfeiffer, and H. Ulbrich. 2004. "Sensors and control concept of a biped robot." *Industrial Electronics, IEEE Transactions on* 51 (5): 972–980. doi:10.1109/TIE.2004.834948.

Lohmeier, S. 2010. "Design and Realization of a Performance Enhanced Humanoid Robot." PhD diss., Technische Universität München. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20101126-980754-1-4.

Lohmeier, S., T. Buschmann, and H. Ulbrich. 2009. "System Design and Control of Anthropomorphic Walking Robot LOLA." *IEEE/ASME Trans. Mechatron.* 14 (6): 658–666. doi:10.1109/TMECH.2009.2032079.

Luh, J.S., M. W. Walker, and R. C. Paul. 1980a. "On-Line Computational Scheme for Mechanical Manipulators." *J. Dyn. Sys., Meas., Control.* 102 (2): 69–76. doi:10.1115/1.3149599.

Luh, J.S., M.W. Walker, and R.C. Paul. 1980b. "Resolved-acceleration control of mechanical manipulators." *Automatic Control, IEEE Transactions on* 25 (3): 468–474. doi:10.1109/TAC.1980.1102367.

Lyapunov, A. M. 1992. *The General Problem of the Stability of Motion.* Taylor & Francis.

Maki, B. E., and W. E McIlroy. 1997. "The Role of Limb Movements in Maintaining Upright Stance: The "Change-in-Support" Strategy." *Physical Therapy* 77 (5): 488–507.

Mansard, N., and O. Khatib. 2008. "Continuous control law from unilateral constraints." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2008.4543723.

Mansard, N., O. Khatib, and A. Kheddar. 2009a. "A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks." *IEEE Transactions on Robotics* 25 (3): 670–685. doi:10.1109/TRO.2009.2020345.

Mansard, N., O. Stasse, P. Evrard, and A. Kheddar. 2009b. "A versatile Generalized Inverted Kinematics implementation for collaborative working humanoid robots: The Stack Of Tasks." In *Advanced Robotics, 2009. ICAR 2009. International Conference on,* 1–6. Munich.

Marigold, Daniel S., and Aftab E. Patla. 2002. "Strategies for Dynamic Stability During Locomotion on a Slippery Surface: Effects of Prior Experience and Knowledge." *Journal of Neurophysiology* 88 (1): 339–353. doi:10.1152/jn.00691.2001.

Matsubara, T., J. Morimoto, J. Nakanishi, M. Sato, and K. Doya. 2006. "Learning CPG-based biped locomotion with a policy gradient method." *Robotics and Autonomous Systems* 54:911–920.

Matsuoka, K. 1985. "Sustained oscillations generated by mutually inhibiting neurons with adaptation." *Biological Cybernetics* 52 (52): 367–376.

Mayne, D. 1966. “A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems.” *International Journal of Control* 3 (1): 85–95. doi:10.1080/00207176608921369.

McGeer, Tad. 1990. “Passive Dynamic Walking.” *The Intl Journal of Robotics Research* 9 (2): 62–82. doi:10.1177/027836499000900206.

McGhee, R.B., and A.A. Frank. 1968. “On the stability properties of quadruped creeping gaits.” *Mathematical Biosciences* 3 (1–2): 331–351.

Mergner, T., G. Schweigart, and L. Fennell. 2009. “Vestibular humanoid postural control.” *Journal of Physiology-Paris* 103 (3–5): 178–194. doi:10.1016/j.jphysparis.2009.08.002.

Mirtich, B. 1998. “V-Clip: Fast and robust polyhedral collision detection.” *ACM Transactions on Graphics (TOG)* 17 (3): 177–208.

Miura, H., and I. Shimoyama. 1984. “Dynamic Walk of a Biped.” *The Intl Journal of Robotics Research* 3:60–74. doi:10.1177/027836498400300206.

Mohan, Ashish, and S. Saha. 2007. “A recursive, numerically stable, and efficient simulation algorithm for serial robots.” *Multibody System Dynamics* 17 (4): 291–319. doi:10.1007/s11044-007-9044-8.

Morisawa, M., K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, S. Nakaoka, and H. Hirukawa. 2006. “A Biped Pattern Generation Allowing Immediate Modification of Foot Placement in Real-time.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids),* 581–586. doi:10.1109/ICHR.2006.321332.

Morisawa, M., F. Kanehiro, K. Kaneko, S. Kajita, and K. Yokoi. 2011. “Reactive Biped Walking Control for a Collision of a Swinging Foot on Uneven Terrain.” In *Proc. IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids),* 768–773.

Nakamura, Y. 1991. *Advanced Robotics: Redundancy and Optimization.* Addison-Wesley.

Nakamura, Y., and H. Hanafusa. 1984. “Singularity Low-Sensitive Motion Resolution of Articulated Robot Arms.” (Japanese), *Transactions of the Society of Instrument and Control Engineers* 20:453–459.

———. 1986. “Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control.” *Journal of Dynamic Systems, Measurement, and Control* 108:0022–0434. doi:10.1115/1.3143764.

———. 1987. “Optimal Redundancy Control of Robot Manipulators.” *The Intl Journal of Robotics Research* 6 (1): 32–42. doi:10.1177/027836498700600103.

Nakamura, Yoshihiko, Hideo Hanafusa, and Tsuneo Yoshikawa. 1987. "Task-Priority Based Redundancy Control of Robot Manipulators." *The Intl Journal of Robotics Research* 6 (2): 3–15. doi:10.1177/027836498700600201.

Nakanishi, J., J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. 2004. "An Empirical Exploration of Phase Resetting for Robust Biped Locomotion with Dynamical Movement Primitives." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. Sendai, Japan.

Nelson, G., A. Saunders, B. Swilling, J. Bondaryk, D. Billings, C. Lee, R. Playter, and M.H. Raibert. 2012. "PETMAN: A Humanoid Robot for Testing Chemical Protective Clothing." *Journal of the Robotics Society of Japan* 30:372–377.

Newton, I. 1687. *Philosophiae Naturalis Principia Mathematica.* Royal Society.

Nishiwaki, K. 2001. "Design of walking system and online control of a humanoid robot." (Japanese). PhD diss., University of Tokyo.

Nishiwaki, K., and S. Kagami. 2006. "High frequency walking pattern generation based on preview control of ZMP." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2006.1642104.

———. 2009. "Frequent walking pattern generation that uses estimated actual posture for robust walking control." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2009.5379519.

———. 2010. "Strategies for adjusting the ZMP reference trajectory for maintaining balance in humanoid walking." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2010.5510002.

Nishiwaki, K., S. Kagami, J. Kuffner, M. Inaba, and H. Inoue. 2002. "Humanoid "JSK-H7": Research Platform for Autonomous Behavior and Whole Body Motion." In *Proc Intl Workshop Humanoid and human friendly Robotics (IARP),* 2–9. Tsukuba, Japan.

Nishiwaki, K., T. Sugihara, S. Kagami, F. Kanehiro, M. Inaba, and H. Inoue. 2000. "Design and development of research platform for perception-action integration in humanoid robot: H6." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2000.895195.

Ogura, Y., H. Aikawa, K. Shimomura, A. Morishima, Hun-ok Lim, and A. Takanishi. 2006. "Development of a new humanoid robot WABIAN-2." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2006.1641164.

Okada, K., M. Inaba, and H. Inoue. 2005. "Real-time and Precise Self Collision Detection System for Humanoid Robots." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2005.1570256.

Orin, D.E., and A. Goswami. 2008. "Centroidal Momentum Matrix of a humanoid robot: Structure and properties." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS),* 653–659. doi:10.1109/IROS.2008.4650772.

Orin, D.E., and W.W. Schrader. 1984. "Efficient Computation of the Jacobian for Robot Manipulators." *The Intl Journal of Robotics Research* 3 (4): 66–75. doi:10.1177/027836498400300404.

Orlovsky, G.N., T.G. Deliagina, and S. Grillner. 1999. *Neuronal Control of Locomotion.* Oxford University Press.

Park, H.-W., A. Ramezani, and J.W. Grizzle. 2013. "A Finite-state Machine for Accommodating Unexpected Large Ground Height Variations in Bipedal Robot Walking." *IEEE Transactions on Robotics.*

Park, Jong H., and Kyoung D. Kim. 1998. "Biped Robot Walking Using Gravity-Compensated Inverted Pendulum Mode and Computed Torque Control." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*.

Pearson, K.G. 2008. "Role of sensory feedback in the control of stance duration in walking cats." *Brain Research Reviews* 57 (1): 222–227.

Pfeiffer, F. 1989. *Einführung in die Dynamik.* (German). B.G. Teubner Verlag.

———. 2006. "The TUM walking machines." *Phil. Trans. R. Soc. A* 365:109–131. doi:10.1098/rsta.2006.1922.

Pfeiffer, F., and R. Johanni. 1986. "A Concept for Manipulator Trajectory Planning." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/JRA.1987.1087090.

Pfeiffer, F., K. Löffler, and M. Gienger. 2002. "The Concept of Jogging Johnnie." In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 3129–3135. Washington, USA.

Piedboeuf, J.-C. 1993. "Kane's equations or Jourdain's principle?" In *Circuits and Systems, 1993., Proceedings of the 36th Midwest Symposium on.* doi:10.1109/MWSCAS.1993.343389.

Pieper, D.L. 1968. "The Kinematics Of Manipulators Under Computer Control." PhD diss., Stanford University California Dept Of Computer Science.

Polak, E. 1973. "An Historical Survey of Computational Methods in Optimal Control." *SIAM Review* 15 (2): pages. ISSN: 00361445.

Pons, J.L. 2005. *Emerging Actuator Technologies: A Micromechatronic Approach.* John Wiley & Sons, Ltd.

Pontryagin, L. S. 1962. *The mathematical theory of optimal processes.* Interscience / John Wiley & Sons Inc.

Pratt, G.A., and M.M. Williamson. 1995. “Series elastic actuators.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.1995.525827.

Pratt, J., J. Carff, S. Drakunov, and A. Goswami. 2006. “Capture Point: A Step toward Humanoid Push Recovery.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/ICHR.2006.321385.

Pratt, J., C.M. Chew, A. Torres, P. Dilworth, and G.A. Pratt. 2001. “Virtual Model Control: An Intuitive Approach for Bipedal Locomotion.” *The Intl Journal of Robotics Research* 20:129–143.

Pratt, J., and B. Krupp. 2008. “Design of a bipedal walking robot.” In *Unmanned Systems Technology X*. doi:10.1117/12.777973.

Pratt, J., B. Krupp, V. Ragusila, J. Rebula, T. Koolen, N. van Nieuwenhuizen, C. Shake, et al. 2009. “The Yobotics-IHMC Lower Body Humanoid Robot.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2009.5354430.

Pratt, J., and G.A. Pratt. 1998a. “Exploiting Natural Dynamics in the Control of a Planar Bipedal Walking Robot.” In *Proceedings of the Thirty-Sixth Annual Allerton Conference on Communication, Control, and Computing*.

———. 1998b. “Intuitive control of a planar bipedal walking robot.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.1998.680611.

———. 1999. “Exploiting Natural Dynamics in the Control of a 3D Bipedal Walking Simulation.” In *Proceedings of the International Conference on Climbing and Walking Robots*.

Quarteroni, A., R. Sacco, and F. Saleri. 2007. *Numerical Mathematics*. Vol. 37. Texts in Applied Mathematics Series. Springer. ISBN: 978-3-540-49809-4.

Raibert, M. H. 1986. *Legged Robots that Balance (Artificial Intelligence)*. Cambridge: MIT Press.

Raibert, M.H. 1983. *Dynamically stable legged locomotion: progress report: October 1982 – October 1983*.

Raibert, M.H., K. Blankespoor, G. Nelson, and R. Playter. 2008. “BigDog, the Rough-Terrain Quadruped Robot.” In *Proc. of the 17th World Congress The Intl Federation of Automatic Control*.

Raibert, M.H., and H.B. Jr. Brown. 1984. “Experiments in Balance With a 2D One-Legged Hopping Machine.” *Journal of Dynamic Systems, Measurement, and Control* 106:75–81.

Raibert, M.H., H.B. Brown, and M. Chepponis. 1984. “Experiments in Balance with a 3D One-Legged Hopping Machine.” *The Intl Journal of Robotics Research* 3 (2): 75–92. doi:10.1177/027836498400300207.

Remy, C.D., M. Hutter, M. Hoepflinger, M. Bloesch, C. Gehring, and R. Siegwart. 2012. “Quadrupedal Robots with Stiff and Compliant Actuation.” *at – Automatisierungstechnik* 60:682–691.

Robinson, D., J. Pratt, D. Paluska, and G.A. Pratt. 1999. “Series Elastic Actuator Development for a Biomimetic Walking Robot.” In *Proc. of the ASME Intl Conf on Advanced Intelligent Mechatronics,* 561–568. Atlanta, Georgia, USA.

Rockafellar, R. T. 1976. “Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming.” *Mathematics Of Operations Research* 1 (2): 97–116. doi:10.1287/moor.1.2.97.

Rulka, W., and A. Eichberger. 1993. “SIMPACK An Analysis and Design Tool for Mechanical Systems.” *Vehicle System Dynamics* 22 (sup1): 122–126. doi:10.1080/00423119308969483.

Rulka, W., and E. Pankiewicz. 2005. “MBS Approach to Generate Equations of Motions for HiL-Simulations in Vehicle Dynamics.” *Multibody System Dynamics* 14 (3-4): 367–386. doi:10.1007/s11044-005-1144-8.

Saha, S.K. 1997. “A decomposition of the manipulator inertia matrix.” *Robotics and Automation, IEEE Transactions on* 13 (2): 301–304. doi:10.1109/70.563652.

Sardain, P., and G. Bessonnet. 2004. “Forces acting on a biped robot. Center of pressure-zero moment point.” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 34 (5): 630–637. doi:10.1109/TSMCA.2004.832811.

Sato, A. 2007. “Simple switching control for hybrid dynamics of a planar hopping robot.” In *Ninth IASTED International Conference on Control and Applications.*

Sayyad, A., B. Seth, and P. Seshu. 2007. “Single-legged hopping robotics research — A review.” *Robotica* 25 (5): 587–613.

Schiehlen, W. 1997. “Multibody System Dynamics: Roots and Perspectives.” *Multibody System Dynamics* 1 (2): 149–188. doi:10.1023/A:1009745432698.

Schoeder, S., H. Ulbrich, and T. Schindler. 2013. “Discussion of the Gear–Gupta–Leimkuhler method for impacting mechanical systems.” *Multibody System Dynamics:*1–19. doi:10.1007/s11044-013-9370-y.

Schuetz, C., T. Buschmann, J. Baur, J. Pfaff, and H. Ulbrich. 2014. “Predictive Online Inverse Kinematics for Redundant Manipulators.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA).* (to appear).

Schultz, G., and K. Mombaur. 2010. "Modeling and Optimal Control of Human-Like Running." *Mechatronics, IEEE/ASME Transactions on* 15 (5): 783–792. doi:10.1109/TMECH.2009.2035112.

Schwertassek, R., and R.E. Roberson. 1984. "A state-space dynamical representation for multibody mechanical systems part I: Systems with tree configuration." *Acta Mechanica* 50 (3-4): 141–161. doi:10.1007/BF01170956.

Schwienbacher, M. 2013. "Efficient Algorithms for Biped Robots – Simulation, Collision Avoidance and Angular Momentum Tracking." (submitted). PhD diss., Technische Universität München.

Schwienbacher, M., T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich. 2011. "Self-collision avoidance and angular momentum compensation for a biped humanoid robot." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ICRA.2011.5980350.

Schwienbacher, M., T. Buschmann, and H. Ulbrich. 2012. "Vertical Angular Momemtum Minimization For Biped Robots With Kinematically Redundant Joints." In *23rd Intl Conf on Theoretical and Applied Mechanics (ICTAM)*.

Seara, J.F., K.H. Strobl, and G. Schmidt. 2002. "Information management for gaze control in vision guided biped walking." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IRDS.2002.1041357.

Semini, C. 2010. "HyQ — Design and Development of a Hydraulically Actuated Quadruped Robot." PHD thesis, University of Genoa.

Shabana, A. A. 1990. "On the use of the finite element method and classical approximation techniques in the non-linear dynamics of multibody systems." *International Journal of Non-Linear Mechanics* 25 (23): 153–162. doi:10.1016/0020-7462(90)90047-D.

———. 1991. "Constrained motion of deformable bodies." *International Journal for Numerical Methods in Engineering* 32 (8): 1813–1831. doi:10.1002/nme.1620320817.

———. 2005. *Dynamics of Multibody Systems.* Cambridge University Press. ISBN: 9780521850117.

Shirai, T., J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. 2011. "Whole body adapting behavior with muscle level stiffness control of tendon-driven multijoint robot." In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on,* 2229–2234. doi:10.1109/ROBIO.2011.6181623.

Siciliano, B. 1990. "Kinematic Control of Redundant Robot Manipulators: A Tutorial." *Journal of lntelligent and Robotic Systems* 3:201–212. doi:10.1007/BF00126069.

Siciliano, B., and O. Khatib, eds. 2008. *Springer Handbook of Robotics.* Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-540-30301-5.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo. 2008. *Robotics: Modelling, Planning and Control.* Advanced Textbooks in Control and Signal Processing Series. Springer. ISBN: 9781849966344.

———. 2009. *Robotics.* Edited by J. Grimble and A. M. Johnson. Springer Verlag London Ltd. doi:10.1007/978-1-84628-642-1.

Sorge, K. 1993. "Mehrkörpersysteme mit starr-elastischen Subsystemen." PhD diss., Technische Universität München.

Sreenath, K., H.-W. Park, I. Poulakakis, and J. Grizzle. 2011. "A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on MABEL." *The Intl Journal of Robotics Research* 30 (9): 1170–1193.

Stasse, O., A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar. 2008. "Real-time (self)-collision avoidance task on a HRP-2 humanoid robot." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2008.4543698.

Stasse, O., B. Verrelst, B. Vanderborght, and K. Yokoi. 2009. "Strategies for Humanoid Robots to Dynamically Walk Over Large Obstacles." *IEEE Transactions on Robotics* 25 (4): 960–967. doi:10.1109/TRO.2009.2020354.

Sugihara, T. 2009. "Dynamics morphing from regulator to oscillator on bipedal control." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2009.5354582.

Sugihara, T., and Y. Nakamura. 2005. "A Fast Online Gait Planning with Boundary Condition Relaxation for Humanoid Robots." In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on,* 305–310. doi:10.1109/ROBOT.2005.1570136.

Sugimoto, N., and J. Morimoto. 2011. "Phase-dependent Trajectory Optimization for CPG-based Biped Walking Using Path Integral Reinforcement Learning." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids),* 255–260.

Sugimoto, Y., H. Yoshioka, and K. Osuka. 2011. "Development of Super-multi-legged Passive Dynamic Walking robot "Jenkka-III"." In *SICE Annual Conference (SICE), 2011 Proceedings of,* 576–579.

Sugiura, H., M. Gienger, H. Janssen, and C. Goerick. 2007. "Real-Time Collision Avoidance with Whole Body Motion Control for Humanoid Robots." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*.

Sutherland, I.E., and M.K. Ullner. 1984. "Footprints in the Asphalt." *The Intl Journal of Robotics Research* 3 (2): 29–36.

Tahboub, K., and T. Mergner. 2007. "Biological and engineering approaches to human postural control." *Integr. Comput.-Aided Eng.* 14 (1): 15–31. ISSN: 1069-2509.

Tajima, R., D. Honda, and K. Suga. 2009. "Fast running experiments involving a humanoid robot." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2009.5152404.

Takao, S., Y. Yokokohji, and T. Yoshikawa. 2003. "FSW (feasible solution of wrench) for multi-legged robots." In *Proc IEEE Intl Conf Robotics and Automation (ICRA)*. doi:10.1109/ROBOT.2003.1242182.

Takenaka, T. 2004a. *Controller of Legged Mobile Robot.* European Patent Application no. EP1475198A1.

———. 2004b. *Gait generation system of legged mobile robot.* european patent application no. EP1671754A2.

———. 2005. *Gait Generating Device For Moving Robot.* european patent application no. EP1738879A1.

Takenaka, T., T. Matsumoto, and T. Yoshiike. 2009a. "Real time motion generation and control for biped robot –1st report: Walking gait pattern generation–." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2009.5354662.

———. 2009b. "Real time motion generation and control for biped robot –3rd report: Dynamics error compensation–." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2009.5354542.

Takenaka, T., T. Matsumoto, T. Yoshiike, T. Hasegawa, S. Shirokura, H. Kaneko, and A. Orita. 2009c. "Real time motion generation and control for biped robot –4th report: Integrated balance control–." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2009.5354522.

Takenaka, T., T. Matsumoto, T. Yoshiike, and S. Shirokura. 2009d. "Real time motion generation and control for biped robot – 2nd report: Running gait pattern generation –." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*, 1092–1099. doi:10.1109/IROS.2009.5354654.

Tassa, Y., T. Erez, and E. Todorov. 2012. "Synthesis and stabilization of complex behaviors through online trajectory optimization." In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2012.6386025.

Taubig, H., B. Bauml, and U. Frese. 2011. “Real-time swept volume and distance computation for self collision detection.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS),* 1585–1592. doi:10.1109/IROS.2011.6094611.

Todorov, E., and Weiwei Li. 2005. “A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems.” In *American Control Conference, 2005. Proceedings of the 2005.* doi:10.1109/ACC.2005.1469949.

Tolle, H. 1971. *Optimierungsverfahren für Variationsaufgaben mit gewöhnlichen Differentialgleichungen als Nebenbedingungen.* Ingenieurwissenschaftliche Bibliothek. (German). Springer. ISBN: 9780387051628.

Torres-Oviedo, G., and L. H. Ting. 2007. “Muscle Synergies Characterizing Human Postural Responses.” *Journal of Neurophysiology* 98 (4): 2144–2156. doi:10.1152/jn.01360.2006.

Toussaint, M., M. Gienger, and C. Goerick. 2007. “Optimization of sequential attractor-based movement for compact behaviour generation.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*.

Träger, M. 2010. “Echtzeitfähige Kollisionserkennung mit Swept-Sphere-Volumes in einer Robotikanwendung.” (German). Term paper (Semesterarbeit), Lehrstuhl für Angewandte Mechanik, Technische Universität München.

Tsagarakis, N.G., Zhibin Li, J. Saglia, and D.G. Caldwell. 2011. “The design of the lower body of the compliant humanoid robot.” In *Robotics and Automation (ICRA), 2011 IEEE International Conference on,* 2035–2040. doi:10.1109/ICRA.2011.5980130.

Tsai, L.W., and A. P. Morgan. 1985. “Solving the Kinematics of the Most General Six- and Five-Degree-of-Freedom Manipulators by Continuation Methods.” *J. of Mech., Trans, and Automation.* 107(2):189–200. doi:10.1115/1.3258708..

Urata, J., Y. Nakanishi, K. Okada, and M. Inaba. 2010. “Design of High Torque and High Speed Leg Module for High Power Humanoid.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. doi:10.1109/IROS.2010.5649683.

Urata, J., K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba. 2011. “Online decision of foot placement using singular LQ preview regulation.” In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids)*. doi:10.1109/Humanoids.2011.6100894.

Vanderborght, B., B. Verrelst, R. van Ham, J. Vermeulen, and D. Lefeber. 2005a. “Dynamic Control of a Bipedal Walking Robot actuated with Pneumatic Artificial Muscles.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 1–6.

Vanderborght, B., B. Verrelst, R. Van Ham, M. Van Damme, and D. Lefeber. 2005b. "A pneumatic biped: experimental walking results and compliance adaptation experiments." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids),* 44–49.

Vereshchagin, A. F. 1974. "Computer Simulation of the Dynamics of Complicated Mechanisms of Robot Manipulators." *Engineering Cybernetics* 6:65–70.

Verrelst, B., O. Stasse, K. Yokoi, and B. Vanderborght. 2006a. "Dynamically Stepping Over Obstacles by the Humanoid Robot HRP-2." In *Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids).* Proc IEEE-RAS Intl Conf on Humanoid Robotics (Humanoids). ISBN: 1-4244-0199-2. doi:10.1109/ICHR.2006.321372.

Verrelst, B., K. Yokoi, O. Stasse, H. Arisumi, and B. Vanderborght. 2006b. "Mobility of Humanoid Robots: Stepping over Large Obstacles Dynamically." In *Proc IEEE Intl Conf Mechatronics and Automation.* doi:10.1109/ICMA.2006.257774.

Vinter, R.B. 1986. "Is the costate variable the state derivative of the value function?" In *Decision and Control, 1986 25th IEEE Conference on,* 25:1988–1989. doi:10.1109/CDC.1986.267362.

Vukobratović, M., and B. Borovac. 2004. "Zero-Moment Point - Thirty Five Years of its Life." *Intl Journal of Humanoid Robotics (IJHR)* 1 (1): 157–173. doi:10.1142/S0219843604000083.

Vukobratović, M., B. Borovac, and V. Potkonjak. 2006. "Zmp: A Review Of Some Basic Misunderstandings." *Intl Journal of Humanoid Robotics (IJHR)* 3:153–175. doi:10.1142/S0219843606000710.

Vukobratović, M., B. Borovac, D. Surla, and D. Stokic. 1990. *Biped Locomotion: Dynamics, Stability, Control and Applications.* Berlin, Heidelberg, New York: Springer.

Vukobratović, M., and V. Potkonjak. 1979. "Contribution of the forming of computer methods for automatic modelling of spatial mechanisms motions." *Mechanism and Machine Theory* 14 (3): 179–188. doi:10.1016/0094-114X(79)90051-X.

Walker, M. W., and D. E. Orin. 1982. "Efficient Dynamic Computer Simulation of Robotic Mechanisms." *Journal of Dynamic Systems, Measurement, and Control* 104:205–211.

Westervelt, E.R., J.W. Grizzle, and D.E. Koditschek. 2003. "Hybrid zero dynamics of planar biped walkers." *Automatic Control, IEEE Transactions on* 48 (1): 42–56. doi:10.1109/TAC.2002.806653.

Whitney, D. E. 1969. “Resolved Motion Rate Control of Manipulators and Human Prostheses.” *IEEE Transactions on Man Machine Systems* 10 (2): 47–53. doi:10.1109/TMMS.1969.299896.

Wieber, P. B. 2002. “On the stability of walking systems.” In *Proceedings of the Intl Workshop on Humanoid and Human Friendly Robotics*.

Winter, D. A. 1995. “Human balance and posture control during standing and walking.” *Gait & Posture* 3 (4): 193–214. doi:10.1016/0966-6362(96)82849-9.

Wittmann, R., A. Hildebrandt, A. Ewald, and T. Buschmann. 2014. “An Estimation Model for Footstep Modifications of Biped Robots.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS)*. (submitted).

Yamaguchi, J., and A. Takanishi. 1997. “Development of a biped walking robot having antagonistic driven joints using nonlinear spring mechanism.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* vol. 1, 185–192 vol.1. doi:10.1109/ROBOT.1997.620036.

Yamaguchi, J., A. Takanishi, and I. Kato. 1994. “Development of a Biped Walking Robot Adapting to a Horizontally Uneven Surface.” In *Proc IEEE/RSJ Intl Conf Intelligent Robots and Systems (IROS),* 1156–1163. Munich, Germany.

———. 1995. “Experimental Development of a Foot Mechanism with Shock Absorbing Material for Acquisition of Landing Surface Position Information and Stabilization of Dynamic Biped Walking.” In *Proc IEEE Intl Conf Robotics and Automation (ICRA),* 2892–2899. Nagoya, Japan.

Yamane, K. 2004. *Simulating and Generating Motions of Human Figures.* Springer Tracts in Advanced Robotics. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-20317-9.

Yamane, K., and Y. Nakamura. 2009. “Comparative Study on Serial and Parallel Forward Dynamics Algorithms for Kinematic Chains.” *The Intl Journal of Robotics Research* 28:622–629. doi:10.1177/0278364909102350.

Acronyms

ABA	Articulated Body Algorithm.
ABB	Axis-Aligned Bounding Box.
ADA	Assembly Dissasembly Algorithm.
AVX	Advanced Vector Extensions.
AVX2	Advanced Vector Extensions 2.
BoS	Base of Support.
BVP	Boundary Value Problem.
CAD	Computer Aided Design.
CFA	Constraint Force Algorithm.
CLIK	Closed Loop Inverse Kinematics.
CNS	Central Nervous System.
CoM	Center of Mass.
CoP	Center of Pressure.
CPG	Central Pattern Generator.
CPU	Central Processing Unit.
CRBA	Composite Rigid Body Algorithm.
DAE	Differential Algebraic Equation.
DCA	Divide and Conqueror Algorithm.
DDP	Differential Dynamic Programming.
DK	Direct Kinematics.
DoF	Degree of Freedom.
EoM	Equation of Motion.
FEM	Finite Element Method.
FSM	Finite State Machine.
FTS	Force/Torque Sensor.
GCC	GNU Compiler Collection.
GPU	Graphics Processing Unit.
HiL	Hardware in the Loop.
IK	Inverse Kinematics.

IKD	Inverse Kineto-Dynamics.
iLQR	iterative Linear Quadratic Regulator.
IMU	Inertial Measurement Unit.
IPM	Inverted Pendulum Model.
IVP	Initial Value Problem.
LIPM	Linear Inverted Pendulum Model.
LSS	Line Swept Sphere.
MBS	Multibody System.
MPC	Model Predictive Control.
MS	Muscle Spindles.
NE	Newton-Euler Method.
OBB	Oriented Bounding Box.
ODE	Ordinary Differential Equation.
OS	Operating System.
PMP	Pontryagin's Minimum Principle.
PSS	Point Swept Sphere.
QP	Quadratic Program.
RAM	Random Access Memory.
RCA	Reactive 3D Collision Avoidance.
RMC	Resolved Motion Rate Control.
RNEA	Recursive Newton Euler Algorithm.
RRT	Rapidly-Exploring Random Tree.
RSC	Reversed Sparse Cholesky.
SIMD	Single Instruction Multiple Data.
SSE	Streaming SIMD Extensions.
SSE2	Streaming SIMD Extensions 2.
SSTA	Step Sequence and Heuristic Trajectory Adaption.
SSV	Swept Sphere Volumes.
STP-BV	Sphere Torus Bounding Volumes.
TCP	Tool Center Point.
TSS	Triangle Swept Sphere.
XcoM	eXtrapolated center of Mass.
ZMP	Zero Moment Point.

Symbols

$\boldsymbol{A}_{ik}$	transformation matrix performing the mapping from a vector described in the frame $\mathscr{F}_k$ to a description in the frame $\mathscr{F}_i$, $\in \mathbb{R}^{3\times3}$.
$\boldsymbol{a}$	spacial acceleration $\in \mathbb{R}^6$.
$\boldsymbol{A}_x$	elementary rotation about x-axis, $\in \mathbb{R}^{3\times3}$.
$\boldsymbol{A}_y$	elementary rotation about y-axis, $\in \mathbb{R}^{3\times3}$.
$\boldsymbol{A}_z$	elementary rotation about z-axis, $\in \mathbb{R}^{3\times3}$.
$\boldsymbol{b}$	velocity-dependent terms in spatial acceleration $\in \mathbb{R}^6$.
$\widehat{\boldsymbol{b}}$	velocity-dependent terms in spatial acceleration, relative to subsystem.
$\mathrm{c}\,x$	abbreviation for $\cos(x)$.
$\mathscr{C}$	Collision pair for SSV distance computation.
$\widehat{\boldsymbol{C}}_{ip}$	spacial transform matrix relative to the subsystem, $\in \mathbb{R}^{6\times6}$.
$\boldsymbol{C}_{ip}$	spacial transform matrix from p to $i \in \mathbb{R}^{6\times6}$.
$\delta\boldsymbol{r}$	virtual displacement (Lagrange's variation).
$\delta\dot{\boldsymbol{r}}$	virtual velocity (Jourdain's variation).
$\delta\ddot{\boldsymbol{r}}$	virtual acceleration (Gauss' variation).
$\mathscr{E}$	SSV model of environment.
$\boldsymbol{e}_i$	i-th unit vector, $i = \{x, y, z\}$ or $i = \{1, 2, 3, \dots\}$.
$\boldsymbol{F}$	Strain Tensor.
$\boldsymbol{F}^c$	constraint force.
$\boldsymbol{F}^i$	impressed force.
$\boldsymbol{f}_i$	i-th column vector of the strain tensor.
$\boldsymbol{G}$	Green-Lagrange strain tensor.
$\boldsymbol{G}$	Gibbs-Appell function / "acceleration energy".
H	nullspace potential function (scalar).
$\boldsymbol{h}$	force vector without actuator forces, $\in \mathbb{R}^6$ for rigid body, $\in \mathbb{R}^{n_q}$ for MBS.
H	Hamiltonian (scalar).
$\boldsymbol{h}^\star$	force vector, $\in \mathbb{R}^6$ for rigid body, $\in \mathbb{R}^{n_q}$ for MBS.

i	imaginary number $\sqrt{-1}$.
$\boldsymbol{I}_n$	Identity matrix, $\in \mathbb{R}^{n\times n}$.
$\boldsymbol{J}$	global Jacobian $\in \mathbb{R}^{6\times n_q}$.
$\widehat{\boldsymbol{J}}$	Jacobian relative to the subsystem.
$\boldsymbol{J}_J$	joint Jacobian $\in \mathbb{R}^{6\times n_q}$.
$\boldsymbol{J}_{J\omega}$	rotational joint Jacobian $\in \mathbb{R}^{3\times n_q}$.
$\boldsymbol{J}_{Jv}$	translational joint Jacobian $\in \mathbb{R}^{3\times n_q}$.
$\boldsymbol{J}_R$	rotational Jacobian $\in \mathbb{R}^{3\times n_q}$.
$\boldsymbol{J}_{R,\mathrm{rel}}$	relative rotational Jacobian $\in \mathbb{R}^{3\times n_q}$.
$\boldsymbol{J}_T$	translational Jacobian $\in \mathbb{R}^{3\times n_q}$.
$\boldsymbol{J}_{T,\mathrm{rel}}$	relative translational Jacobian $\in \mathbb{R}^{3\times n_q}$.
$\boldsymbol{J}_w$	task coordinate Jacobian, $\in \mathbb{R}^{m\times n_q}$.
$\boldsymbol{J}_{w,W}^{\#}$	pseudo inverse for $\boldsymbol{J}_w$, with weighting matrix $\boldsymbol{W}$, $\in \mathbb{R}^{n_q\times m}$.
$\boldsymbol{K}$	backward propagation matrix for recursive forward dynamics.
$\boldsymbol{k}$	backward propagation vector for recursive forward dynamics.
L	Lagrangian (scalar).
$\boldsymbol{\lambda}$	Lagrangian multiplier or adjoint variable.
$\boldsymbol{L}_o$	angular momentum w.r.t. point O, $\in \mathbb{R}^3$.
$\boldsymbol{M}$	mass matrix, $\in \mathbb{R}^{n_q\times n_q}$ for MBS, $\in \mathbb{R}^{6\times 6}$ for rigid body.
m	mass (scalar).
n_b	number of bodies in MBS.
n_q	Dimension of $\boldsymbol{q}$.
$\boldsymbol{N}_{w,W}$	nullspace projection matrix for $\boldsymbol{J}_w$, with weighing matrix $\boldsymbol{W}$, $\in \mathbb{R}^{n_q\times n_q}$.
O_B	origin of frame B.
O_{B_i}	origin of B_i (the i-th body-fixed frame).
$\mathscr{O}$	SSV segment (for $\mathscr{E}$).
$\boldsymbol{\omega}$	angular velocity.
$\dot{\boldsymbol{\omega}}$	angular acceleration.
$\boldsymbol{p}$	linear momentum, $\in \mathbb{R}^3$.
Φ	cost function (scalar).
$\boldsymbol{\Phi}(\boldsymbol{z}, t)$	holonomic constraints for MBS.
$\boldsymbol{\Phi}(\boldsymbol{z}, \dot{\boldsymbol{z}}, t)$	nonholonomic constraints for MBS.
$\boldsymbol{P}$	forward propagation matrix for recursive forward dynamics.

$\boldsymbol{p}$	forward propagation vector for recursive forward dynamics.
$\boldsymbol{q}$	minimal coordinates / generalized coordinates, $\in \mathbb{R}^{n_q}$.
$\mathbb{R}^3$	Euclidean space of dimension 3.
$\mathscr{R}$	SSV model of robot.
$\boldsymbol{r}_P$	vector to point P, $\in \mathbb{R}^3$.
$\dot{\boldsymbol{r}}_P$	absolute rate of change of $\boldsymbol{r}_P$.
$\ddot{\boldsymbol{r}}_P$	absolute rate of change of $\dot{\boldsymbol{r}}_P$.
$\mathring{\boldsymbol{r}}_P$	time derivative of components of $\boldsymbol{r}_P$.
$\mathring{\mathring{\boldsymbol{r}}}_P$	time derivative of components of $\dot{\boldsymbol{r}}_P$.
$\mathrm{s}\,x$	abbreviation for $\sin(x)$.
$\mathscr{S}$	SSV segment (for $\mathscr{R}$).
$\boldsymbol{T}^c$	constraint moment.
$\boldsymbol{\Theta}_P$	mass moment of inertia w.r.t. point P, $\in \mathbb{R}^{3\times3}$.
$\boldsymbol{T}^i$	impressed moment.
$\mathrm{tr}(\boldsymbol{M})$	trace of matrix $\boldsymbol{M}$.
$\boldsymbol{u}$	actuator forces or control input.
$\boldsymbol{v}$	spacial velocity $\in \mathbb{R}^6$.
$\mathscr{V}$	SSV element.
$\boldsymbol{w}$	task space coordinates, $\in \mathbb{R}^m$.
$\boldsymbol{z}$	system coordinates, $\in \mathbb{R}^{6n_b}$.

Indices and Accents

$(\cdot)_d$	desired quantity.
$\dot{(\,)}$	derivative with respect to time (relative to inertial frame).
$\overset{\circ}{(\,)}$	derivative with respect to time (relative to current frame).
$(\cdot)^{\#}$	(Moore Penrose) pseudo inverse.
$\widetilde{\boldsymbol{x}} \in \mathbb{R}^{3\times 3}$	spin tensor.

Index