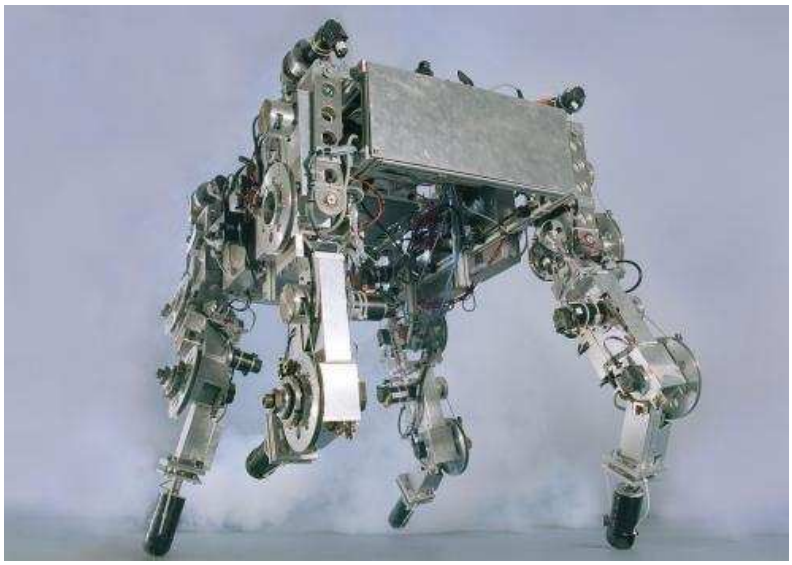
TRITA-MMK 2003:19
ISSN 1400-1179
ISRN KTH/MMK–03/19–SE

Doctoral thesis

Legged locomotion: Balance, control and tools — from equation to action

Christian Ridderström



Stockholm
2003

Department of Machine Design
Royal Institute of Technology
S-100 44 Stockholm, Sweden

Akademisk avhandling som med tillstånd från Kungliga Tekniska Högskolan i Stockholm framläggs till offentlig granskning för avläggande av teknologie doktorsexamen, den *12:e maj 10:00* i *Kollegiesalen*, vid Institutionen för Maskinkonstruktion, Kungliga Tekniska Högskolan, Stockholm.

Stockholm 2003, Universitetsservice US AB

Till familj och vänner

Abstract

This thesis is about control and balance stability of legged locomotion. It also presents a combination of tools that makes it easier to design controllers for large and complicated robot systems. The thesis is divided into four parts.

The first part studies and analyzes how walking machines are controlled, examining the literature of over twenty machines briefly, and six machines in detail. The goal is to understand how the controllers work on a level below task and path planning, but above actuator control. Analysis and comparison is done in terms of: i) generation of trunk motion; ii) maintaining balance; iii) generation of leg sequence and support patterns; and iv) reflexes.

The next part describes WARP1, a four-legged walking robot platform that has been built with the long term goal of walking in rough terrain. First its modular structure (mechanics, electronics and control) is described, followed by some experiments demonstrating basic performance. Finally the mathematical modeling of the robot's rigid body model is described. This model is derived symbolically and is general, i.e. not restricted to WARP1. It is easily modified in case of a different number of legs or joints.

During the work with WARP1, tools for model derivation, control design and control implementation have been combined, interfaced and augmented in order to better support design and analysis. These tools and methods are described in the third part. The tools used to be difficult to combine, especially for a large and complicated system with many signals and parameters such as WARP1. Now, models derived symbolically in one tool are easy to use in another tool for control design, simulation and finally implementation, as well as for visualization and evaluation — thus going from equation to action.

In the last part we go back to "equation" where these tools aid the study of balance stability when compliance is considered. It is shown that a legged robot in a "statically balanced" stance may actually be unstable. Furthermore, a criterion is derived that shows when a radially symmetric "statically balanced" stance on a compliant surface is stable. Similar analyses are performed for two controllers of legged robots, where it is the controller that cause the compliance.

Keywords

legged locomotion, control, balance, legged machines, legged robots, walking robots, walking machines, compliance, platform stability, symbolic modeling

Preface

There is a really long list of people to whom I wish to express my gratitude — the serious reader may skip this section. In short, I would like to thank the entire DAMEK group, as well as some people no longer working there.

I've had two supervisors, no three actually... my main supervisors have been Professor Jan Wikander and Doctor Tom Wadden, but I'm also grateful for being able to talk things over with Bengt Eriksson in the beginning. We did a project involving a hydraulic leg and managed to spray oil on Tom's jacket. Tom was also the project leader for WARP1 for a while and he put the fear of red markings in us PhD. students. When you got something you'd written back from Tom, you were sure to get it back slaughtered. He definitively had an impact on my English.

Speaking of the WARP1 project, I want to thank all the other PhD. students in it for a great time, especially Freyr who finished last summer and Johan who's still "playing" with me in the lab on Tuesdays. Johan and I are almost too creative when working together — we can get so many ideas that we don't know which to choose. Then there is also Henrik, who I wrote my first article with.

The research engineers, Micke, Johan and Andreas also deserve a special thank you, as well as "our" Japanese visitors Yoshi and Taka. Peter and Payam should not be forgotten either — the department's SYS.ADM. — both very fun guys.

The social aspect of a PhD. student's life is very important, and Martin Grimheden earns a special thanks for his *fredagsrus*[1] which I believe were introduced by Micke. Most of the other PhD. students at the department have also been social. Ola got us to play basketball once a week for a season... in retrospect its amazing that so many of us actually were there playing at 08.00 in the morning.

Thank you guys!

Finally I want to thank the developers of LyX [1] for a great program, and also everybody on the User's List for their help in answering questions. I've used LyX to write this thesis and all my articles, and not regretted it once!

Acknowledgments

This work was supported by the Foundation for Strategic Research through its Centre of Autonomous Systems at the Royal Institute of Technology. Scientific and technical contributions are acknowledged in each part respectively.

[1] A Swedish word local to DAMEK, the etymology involves weekday, running and beer.

Notation

See chapter 6 for details on coordinate systems, reference frames, triads, points, vectors, dyads and general vector algebra notation. Table 6.2 on page 152 lists points and bodies in the WARP1 rigid body model (rbm). Vectors fixed in bodies of the WARP1 rbm are listed in table 6.3 on page 152.

Bodies, points and matrices are denoted using captial letters, e.g. A. Vectors are written in bold and the column matrix with components of a vector $\mathbf{r}$ is written as r. Similarly, dyads are written in bold $\mathbf{A}$ and the components of a dyad $\mathbf{A}$, i.e. its matrix representation, is written as A. Note that a left superscript indicates reference body (or triad) for vectors and matrices (see 144).

$\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3,\ b$ triad fixed relative to B, $b = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]^T$

$\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3,\ n$ triad fixed relative to N, $n = [\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3]^T$

$N, B, T_l, S_l, H_l, K_l, P_l$ points in WARP1 rbm (table 6.2).

$\mathbf{r}^{BH_l}, \mathbf{r}^{HT_l}, \mathbf{r}^{HK_l}, \mathbf{r}^{KS_l}, \mathbf{r}^{KP_l}$ body fixed vectors of the WARP1 rbm (table 6.3).

β	duty factor
l or $_l$	leg l, index indicating leg l
L	total number legs (L=4 for WARP1)
t	time
T	period or cycle time
$x, \dot{x}$	state and state derivative, i.e. $\dot{x} = \frac{\partial x}{\partial t}$
f, F	force
m	mass
g	constant of gravity
$q, \dot{q}, \ddot{q}$	generalized coordinates, velocities and accelerations
$w, \dot{w}$	generalized speeds and corresponding derivative
I_x, I_y, I_z	inertia parameters of a body
λ,$\lambda_{i,j}$	eigenvalue, $\lambda_{i,j} = \pm c$ means that $\lambda_i = c$ and $\lambda_j = -c$

CM	centre of mass of robot (or trunk)
P_{CM}	projection of CM onto a horizontal plane
P_{CP}	centre of pressure
A_{sup}	support area or region
$\phi_i^{\underline{\sigma_i}}$	instance of a controller from a control basis
J_l	Jacobian for leg leg
τ^l	joint torques for leg l
P_1, P_2	foot points for leg 1 and 2
C_1, C_2	ground contact point for leg 1 and 2
ϕ_l	relative phase of leg l
k	stiffness coefficient
d	damping coefficient
$K^l, {}^B K^l$	stiffness coefficients for leg l
$D^l, {}^B D^l$	damping coefficients for leg l
k_1, k_2, k_3	stiffness coefficients see (11.2) on page 221.
k_B, k_α, k_L	stiffness coefficients (table 11.1)
d_B, d_α, d_L	stiffness coefficients (table 11.1)
α	rotation angle planar rigid body (part IV)
r	radius (in part IV)
h	height (in part IV)
γ	asymmetry parameter (part IV)
R	asymmetry parameter, $R = \gamma r$ (part IV)
a, a_2, a_3, a_4, a_L	bifurcation parameters (part IV)

Contents

List of Figures

List of Figures

List of Tables

List of Tables

1. Introduction

This thesis is about legged locomotion in several ways, but it is also about working with complex robot systems. Of course, walking robots are usually complicated... The thesis begins with a comparative overview of control methods for legged locomotion, and there are no simple robots in it. The four-legged robot WARP1 is described next in the thesis, and there the complexity has been tackled by modularity in terms of mechanics, electronics as well as control structure and mathematical modeling. Being able to research and test control methods for WARP1 requires advanced tools and methods and these are also described in the thesis. With these tools, complicated symbolic models and expressions are used for numeric simulations, as well as in the implemented robot controller, hence going from equation to action.

Finally the complex issue of static stability of a legged robot when compliance is included in the model is approached analytically. The complexity is managed by working with, relatively speaking, small symbolic models, combined with numerical analysis and simulations, where tools and methods from the control design are used.

Outline of thesis

The main contents of this thesis is divided into four parts, where I have tried to make it possible to read the parts independently.

Part I is about walking machines in general. Chapter 2 first gives a brief background of walking research, followed by a more detailed introduction to basic concepts within this field. Then chapter 3 contains detailed descriptions of different controllers for legged machines and finally chapter 4 contains an analysis and summary of these controllers, but also a discussion of legged controllers in general.

Part II describes the four-legged walking robot platform WARP1, where chapter 5 describes the platform in terms of its modular structure. Chapter 6 then discusses

the mathematical modeling of the robot, especially its rigid body model. Basic performance of the robot, such as strength and speed, is demonstrated through experiments. The mathematical modeling is general and not restricted to WARP1. This is also true for the software tools and methods developed to deal with the scale and complexity of WARP1.

Part III describes these tools and methods in chapter 7, and chapter 8 contains an example demonstrating some of the tools use on a "simpler" robot. Today, in addition to tools such as CAD/CAM and tools for model derivation, control design and implementation there are also tools for exporting models to control design environments, as well as from control design to implementation (rapid prototyping tools). It is however, still difficult to combine these tools, especially when working with large systems, i.e. systems with a lot of signals and parameters. We have therefore combined, interfaced and augmented some of these tools into a method that bridges the gaps between automatic model derivation and control implementation. Analytically derived functions (Maple) are used for control design, simulation, visualization, evaluation (MATLAB) and implementation (Real-Time Workshop/xPC Target).

Part IV studies balance of walking machines when the system model includes compliance. In chapter 9 it is shown that a planar symmetric legged robot in a "statically balanced" stance on a soft surface may actually be unstable. Chapter 10 then extends these results for radially symmetric "statically balanced" stances. And in chapter 11, a similar analysis is also performed for legged robots where the compliance comes from the controller. The part ends with chapter 12, where these results are summarized and discussed.

Chapter 12 also contains a general discussion related to all the parts.

1.1. Thesis contributions

The main contributions of this thesis are as follows:

- A small number of questions/aspects are suggested that are useful in order to compare different controllers for walking robots, and to help understand how they work. (Theoretical understanding)
- A detailed description of how to derive the rigid body model for a general class of walking robots. (Practical)

- A method where tools are combined to make it feasible working with large and complicated systems with lots of signals and parameters. (Practical, control design, analysis)
- A method that covers going from equation to action automatically. (Practical, design)
- Criteria are derived for asymptotic stability of statically balanced stances when the model includes compliance. It is shown that a statically balanced robot on a compliant surface can actually fall over. (Theoretical understanding)

1. Introduction

Part I.

A study of legged locomotion control

Introduction to part I

This part is based on a literature study by Ridderström [156] of methods used to control walking machines. The question of *how* to study the machines is analyzed in chapter 4, with the result that the following main questions are chosen in order to understand how the control of walking machines "really" work:

- What determines a walking machine's balance?
- What determines a walking machine's motion, as seen from the controller's perspective?
- What determines a walking machine's support sequence? (foothold selection and sequence) What causes leg phase transitions?
- What, if any, "reflexes" are used?

Another purpose of this study is to give an overview of methods used to control walking machines, as well as to provide suitable reading for new students of legged robots. This study is part of the effort at the Centre for Autonomous Systems [18] to create legged robots capable of both statically and dynamically balanced locomotion. As part of that effort, different control strategies and types of actuators are investigated, while the number of legs has been fixed to four for the research platform WARP1.

Previous work within the centre includes surveys of the design of mechanics (Hardarson [62]), computer control architectures (Pettersson [140]) and also the control of dynamically balanced locomotion (Eriksson [39]).

This study is broader than Eriksson's survey [39], in the sense that it also includes statically balanced walking and tries to find common principles and ideas used for the control of walking. It also studies a few walking controllers in detail, focusing on the level above control of individual actuators, but below the level of long range[1] path- and task planning. Studies of sensors, sensor filtering and mechanics are not included.

Outline The outline of this part is as follows. Chapter 2 first gives a brief background of walking research, followed by a more detailed introduction to basic concepts within this field. Then chapter 3 contains detailed descriptions of different controllers for legged machines and finally chapter 4 contains an analysis (sections 4.1-4.2) as well as a summary and discussion (section 4.3).

[1] i.e. more than a few trunk lengths.

1.

Acknowledgements This work was supported by the Swedish Foundation for Strategic Research [176] through the Centre for Autonomous Systems [18] at the Royal Institute of Technology [100] in Stockholm.

2. Introduction to legged locomotion

Animals have used legs for a long time and legged machines have been around for at least a hundred years. Todd [180] gives a nice introduction to early history of walking machines, basic principles and some walking systems. Song and Waldron's book [174] gives a good overview of statically balanced walking, while Raibert's book [149] describes the design and control of his hopping robots. These references mostly deal with machines having four or more legs, while Furusho and Sano [44] give a review of two-legged robots, including a table of related research.

Why use legged locomotion instead of wheels? Some reasons are given below:

- A US Army investigation [186] reports[1] that about half the earth's surface is inaccessible to wheeled or tracked vehicles, whereas this terrain is mostly exploited by legged animals.
- Legged locomotion should be mechanically[2] superior to wheeled or tracked locomotion over a variety of soil conditions according to Bekker [8, pp.491-498] and certainly superior for crossing obstacles according to Waldron et al. [196].
- The path of the legged machine can be (partially) decoupled from the sequence of footholds, allowing a higher degree of mobility. This can be especially useful in narrow surroundings or terrain with discrete footholds [149].
- Legs can have less of a destructive impact on the terrain than wheels or tracks. This is important within for instance forestry and agriculture.

From the assumption that legged locomotion is useful in rough terrain, applications such as exploration[3] or locomotion in dangerous environments like disaster areas follow naturally.

[1] According to Waldron et al. [196], we have not been able to acquire this report.

[2] With respect to the foot – ground interaction.

[3] The walking robot Dante has actually been used to explore the volcano Mount Erebus [200].

Now that the motivation for using legged locomotion has been given, let us look at some of the problems. Designing a legged robot is far from trivial. Creating a machine that is powerful enough, but still light enough is very difficult. This is however not considered in this report, instead it focuses on the control of walking machines. Some of the problems are very similar, or the same, as those encountered when controlling traditional industrial robot manipulators.

- The robot kinematics and dynamics are nonlinear, difficult to accurately model[4] and simple models are generally not adequate [98]. Furthermore, the dynamics depend on which legs are on the ground, and might therefore be considered as switching. Robot parameters (centre of mass position, amount of payload etc) are not known exactly and might also vary [131].
- The environment is unknown and dynamic. The surface, for example, might be elastic, sticky, soft or stiff [57].

Since a legged machine has a free base, other problems are more specific.

- Contact forces, in general, only allow pushing the feet into the surface, not pulling. This directly limits the total downwards acceleration that can be "applied" to a walking machine.
- The system might be unstable without control, like Raibert's hopping robots [149]. Simply locking all joint angles might not be enough to achieve stability.
- The goal of keeping balance is difficult to decompose into actuator commands.
- A legged system has a lot of degrees of freedom. Waldron et al. [196] argue that, in order to allow a completely decoupled motion[5] over irregular terrain, at least three degrees of freedom per leg are required. This results in 12 actuators for a four-legged robot, compared to six for a traditional industrial manipulator.
- It is difficult to estimate states of the system, such as the translational and rotational position/velocity of the trunk as reported by Pugh et al. [146].

From an implementation point of view, there are also arguments for centralized v.s. decentralized solutions.

- Due to limitations in computer performance or cabling, it might be desirable or necessary to use decentralized/distributed solutions.

[4] As an example, parts of the machine might have to be elastic, like the legs of the Adaptive Suspension Vehicle due to weight constraints.

[5] I.e. the trunk can move independently of the feet.

- Time delays in the control loop amplify stability problems, which might be a reason for using centralized solutions.

Because of the large number of actuators, problems with coordinating the actuators arise. For instance, by considering more than one foot rigidly connected to the ground, we will have closed kinematic chains. Undesired constraint forces (and/or slipping) can occur if these chains are not considered properly.

Finally, we would like to point out that control is not always difficult (or even necessary). In fact, there are machines that can walk *passively* by starting the system in suitable initial conditions, typically on an inclined slope. See for instance the work by McGeer [115], Dankowicz [33], Berkemeier [10] and others.

2.1. Basic walking

This section will describe some basic walking concepts and definitions, in order to aid a novice to this field. Let us first classify[6] legged locomotion into walking, running and hopping. There are several definitions of these terms in the literature. One way to differentiate between walking and running is to use a dimensionless measure such as the Froude number[7]. Another is to say that running is legged locomotion with flight phases. However, we will use the terms walking and running rather interchangeably, but use hopping to mean locomotion with only (almost) instantaneous support phases, i.e. really a bouncing motion such as used by Raibert's hopping robots [149].

Next some basic terminology will be described (section 2.2), followed by definitions of gaits (section 2.3). Then static (section 2.4) and dynamic balance (section 2.5) is discussed, followed finally by a discussion of center of pressure (section 2.5.1) and zero moment point (section 2.5.2).

2.2. Terminology

The terminology within the field of walking robots has borrowed frequently from the fields of biology and biomechanics. The word *body* is often used in the literature to describe the major part of a robot. However, in this report the term *trunk*[8] [9]

[6] This study ignores locomotion methods such as jumping, leaping, vertical clinging and brachiation.

[7] The *Froude number* is $\frac{u^2}{gh}$ where u is the locomotion speed, g is the acceleration constant and h is the distance from the ground to the hip joint. See Alexander [4] for an example of how it is used.

[8] The word *trunk* means the human or animal body apart from the head and appendages [122].

[9] The trunk is not always one rigid body, e.g. the robots TITAN VI [70] and BISAM [11].

will be used instead, since from a rigid mechanics point of view, a robot typically consists of several bodies. The *legs* are attached at the *hips* of the trunk and the legs can sometimes be described as *articulated*. An articulated[10] leg can kinematically be described as links connected by individual revolute joints. A *pantograph* mechanism is often used as a *gravitationally decoupled actuator* [72] (GDA) in *gravity decoupled walking robots* [128], where the actuators are used either to propel the machine, or to support its weight. There are of course other types of legs and actuator mechanisms, like linear joints or direct mechanical linkages between different sets of legs [200]. The *feet* are attached to the legs and are used to walk on the *ground*[11].

Depending on the number of feet different terms are used such as *monopod* (one foot), *biped* (two feet), *quadruped* (four feet), *hexapod* (six feet) and *octapod* (eight feet). To describe directions and locations, the following terminology is used within this report.

- The *ground reference frame*, N, (or *ground frame* for short, figure 2.1) is the inertial[12] frame fixed with respect to the ground. Usually, the first two axes, $\mathbf{n}_1$ and $\mathbf{n}_2$, lie within the *horizontal plane*, while the third axis, $\mathbf{n}_3$, is directed upwards, i.e. opposite to the field of gravity.
- The *trunk reference frame*, B, (or *trunk frame* for short, figure 2.1) is the frame fixed with respect to the trunk, usually[13] at the trunk's centre of mass (CM). Assuming a standard orientation of the trunk, the first axis, $\mathbf{b}_1$, is directed forward, the second axis, $\mathbf{b}_2$, directed to the left and the third axis, $\mathbf{b}_3$, directed upwards.
- The term *anterior* means situated before, or toward the front and the term *posterior* means situated behind.
- The term *lateral* means *of or related to the side*, and hence *ipsilateral* means situated on the same side, whereas *contralateral* means situated on the opposite side. A lateral direction means sideways, i.e. $\mathbf{b}_2$ in figure 2.1.
- The term *sagittal plane* (or *median plane*) denotes the plane that divides a bilateral animal (or machine in this case) into equal left and right halves, i.e. the plane is normal to $\mathbf{b}_2$.
- The *longitudinal axis* is the axis going from the posterior to the anterior, i.e. the forward axis of the machine ($\mathbf{b}_1$ in figure 2.1).

[10] The term articulated is used in other ways too, but this is how the term is used within this report.
[11] The contact between a foot and the ground is often modelled as a point contact.
[12] This is strictly speaking not an inertial frame since the earth is rotating.
[13] Other options are the machine's centre of mass or the trunk's geometric centre.

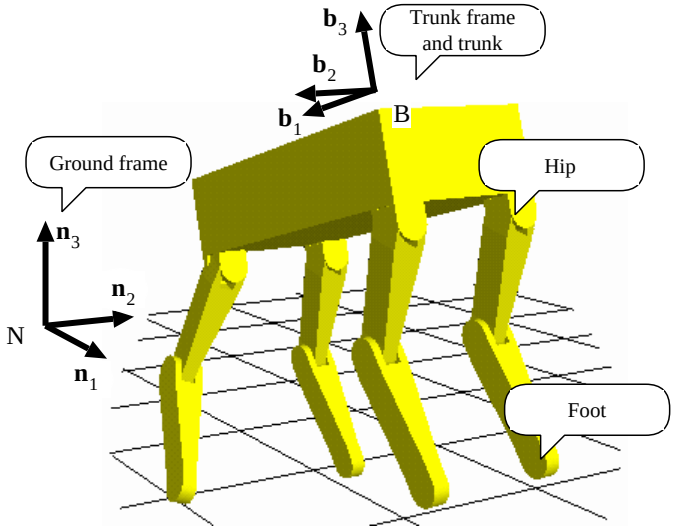


Figure 2.1: Illustration of the ground and trunk reference frames.

- *Cursorial* means “adapted to running” [122] and will in this report be used to denote a leg configuration similar to standing with straight legs, thus minimizing the hip torques.
- The term *attitude* is used to describe the roll and pitch angles of the trunk, while *orientation* is used for all three angles of the trunk.
- *Posture* is used in several ways in the literature. One use, as defined in a dictionary [122], means

  > the position or bearing of the body whether characteristic or assumed for a special purpose <erect posture>.

  However, if not otherwise specified, it will in this report denote

  > the attitude and height of the trunk.

2.3. Definition of gait and gaits

A *gait* is, according to Hildebrand [65]

> A manner of moving the legs in walking or running.

Another definition (used by Alexander [5] in his studies of vertebrate locomotion at different speeds) defines gait as

> ... a pattern of locomotion characteristic to a limited range of speeds described by quantities of which one or more change discontinuously at transition to other gaits.

As an example, quadrupedal mammals typically change between the gaits walk, trot and gallop when they increase their speed. We will use the latter definition in this study. It is not very specific, but a more mathematically precise example of a gait definition is given later (p. 36).

The leg cycle During walking and running, the individual legs typically move cyclically and, in order to facilitate analysis and/or control, the motion of a leg is often partitioned into the following two phases[14]:

- During the *support* phase, the leg is used to support and propel the robot. The terms *power stroke* and *stance* are also used for this phase in the literature.
- During the *transfer* phase, the leg is moved from one foothold to the next. The terms *return stroke* or *swing* are also used for this phase in the literature.

Hildebrand, McGhee, Frank and others introduced a parameterization based on this partitioning to describe the locomotion. The definitions vary somewhat between authors and are defined below as they are used in this paper. They are mostly useful for periodic locomotion patterns, such as when all legs perform the same motion but with some phase shift. The parameters will vary as speed change, but even with a constant speed there is a natural variation in these parameters for animals (Hildebrand [65]). Alexander [4] claims that for sustained gaits, the variation is "nearly always" quite small.

- The *posterior extreme position* (PEP) is the transition[15] from the support phase to the transfer phase.
- The *anterior extreme position* (AEP) is the transition from the transfer phase to the support phase.

[14] The motion of a leg can of course be partitioned in other ways, for instance into the four phases: footfall, support, foot lift-off and transfer.

[15] Intending either the position at the time of phase change or the actual event.

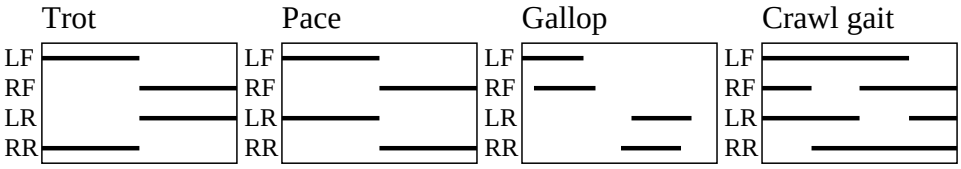


Figure 2.2: Gait diagram of the gaits: trot ($\beta = 0.5$); pace ($\beta = 0.5$); rotary gallop ($\beta = \frac{5}{16}$); and crawl ($\beta = 0.75$). LF, RF, LR and RR stands for left front, right front, left rear and right rear respectively.

- A *gait diagram*[16] is used to illustrate the phases of the different legs as a function of time. Figure 2.2 illustrates this, where the solid lines indicate the support phase.
- A *support pattern at a time* t is the two-dimensional point set created by the convex hull of the projection of the supporting parts of the feet onto a horizontal plane. (Modified from Song and Waldron [174])

  The *support area* (denoted A_{sup} in this study), is the interior and boundary of the support pattern. Sometimes the support pattern will (slightly incorrect) be referred to as the *support polygon*. This usage stems from the idea that a contact between a foot and the supporting surface is modeled as a point contact.

  A *conservative support polygon* is a support polygon where any supporting leg can fail without causing the machine to fall ([128]).
- A *step* is the advance of one leg, *step cycle* the cyclic motion of one leg and *step length* the distance[17] between two consecutive footholds of one leg in a ground frame.

  Hirose [72] defines a step as the interval from one footfall until the following footfall (not necessarily of the same leg).
- A *stride* consists of as many steps as there are legs, i.e. typically each leg completes a cycle of motion and the *stride length* of a gait is the distance the trunk translates during one stride. *Stride duration* is the duration of one stride and the locomotion velocity of periodic locomotion is simply stride length divided by stride duration.

[16] The gait diagram was first used by Hildebrand [65] according to Song and Waldron [174], but we have not been able to find that term in that reference. However, it was most likely Hildebrand that introduced the diagram.

[17] It is assumed that there is no slipping.

- A *duty factor* (typically denoted β) describes the percentage of a step cycle (in time) that a leg is in the support phase.
- A *relative phase of leg* l (typically denoted ϕ_l) describes the leg's phase with respect to a reference leg.

McGhee [116], Kugushev and Jaroshevskij [95] and others also use the two-phase partioning to mathematically define *gait.* Gait is defined as a sequence of binary vectors, $q^1, q^2, \ldots, q^n$, where q^i_l indicates the phase (transfer or support) of leg l. To include time into the description of locomotion, the i^{th} component of the *duration vector* describes the duration of state q^i. The gait is thus defined by the sequence in which the legs change phase, i.e. the *leg sequence.* In this study, the term *support sequence* is used to mean the leg sequence as well as the support pattern.

We will now describe a few more periodic gaits, in addition to trot, pace and gallop (figure 2.2).

- In the *wave gait*[18], the footfalls begin on one side at the rear and proceed like a wave towards the front. For each leg, the laterally paired leg is exactly half a stride cycle out of phase (Song and Waldron [174]).
- The *crawl gait*[19] is the wave gait for a quadruped, but a quadruped can start walking with any leg (figure 2.3).
- The *crab-walking* gait is a walking motion, with the direction of locomotion different from, or equal to, the longitudinal axis of the trunk. The angle between the longitudinal axis and the direction of motion is the crab angle, α, where $\alpha = 0$ corresponds to walking straight forward (Hirose [72]).
- The *turning gait* is a steady circular walking movement, where a point on the robot rotates around a (fixed) turning centre. A turning centre located at an infinite distance corresponds to crab-walking, while a turning centre close to the trunk's centre of mass corresponds to turning on the spot (Hirose et al. [67]).
- The *creeping gait* is a gait where at most one leg is in the transfer phase (Tomović [181] according to McGhee and Frank [117]).
- The *tripod* and *tetrapod gaits* (figure 2.4) are commonly used by hexapods. In the tripod gait two sets of three legs each are moved repeatedly. One could

[18] Wilson [201] (according to Hirose [72]) reports this to be the standard gait of insects. Wave gaits are optimal in the sense that a static stability margin, see the next section, is maximized.

[19] Muybridge [127](according to McGhee and Frank [117, p.332]), reports this to be the typical gait used by quadrupeds at slow gaits. McGhee and Frank [117] showed this to be an optimal statically stable gait for quadrupeds.

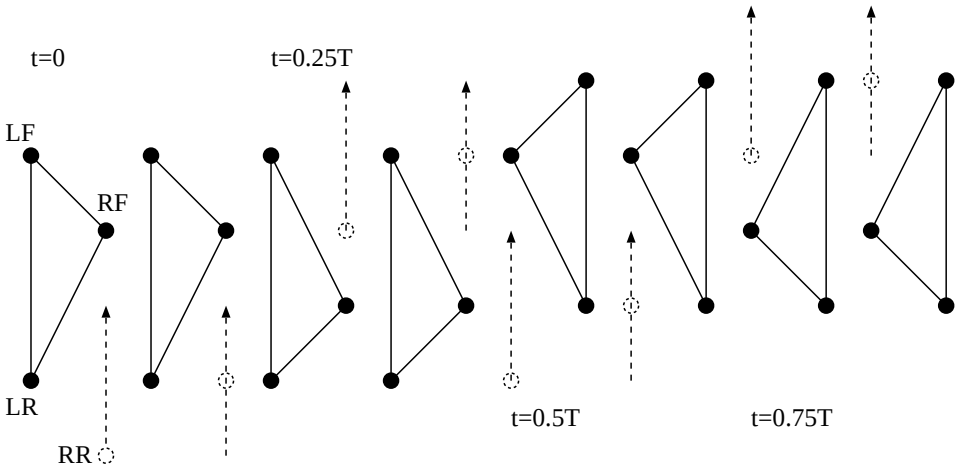


Figure 2.3: Illustration of support sequence for crawl gait with cycle time T, where LF, RF, LR and RR stands for left front, right front, left rear and right rear respectively.

consider it an analogy to the trot for a hexapod. The tetrapod gait is also used by animals and machines with six or more legs. Typically insects use the tetrapod gait at slow speeds and the tripod gait at higher speeds.

In really rough terrain, cyclic gaits are not suitable and the *free gait* (Kugushev and Jaroshevskij [95]; McGhee and Iswandhi [118]) is used instead. The support sequence is rarely periodic and a recent paper by Chen et al. [19] contains a nice overview of how free gaits can be generated.

- The *follow-the-leader* gait, is more of a strategy for leg placement than a gait. Posterior feet are placed closed to, or a the same position, as the anterior foothold. This way, all but the front legs use footholds that have already been used (Song and Waldron [174]).

- *Discontinuous gaits* are used in very irregular terrain and are named so because the trunk motion is very discontinuous, because only one leg is moved at a time. (Gonzales de Santos and Jimenez [34]).

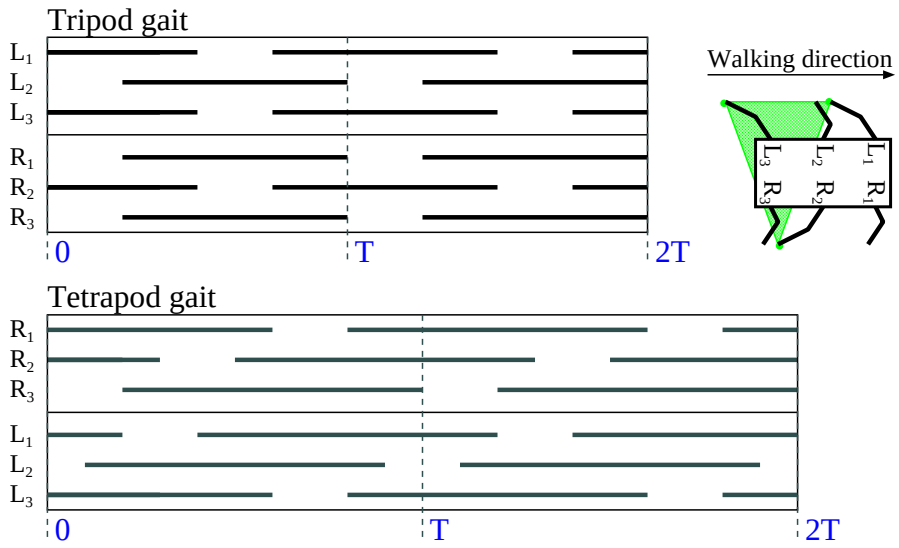


Figure 2.4: Gait diagrams of tripod gait and tetrapod gait.

2.4. Static balance

The early work (and recent work too) on stability analysis was based on the position of the robot's centre of mass (robot CM)[20]. In this study, P_{CM}, denotes the two-dimensional point obtained by projecting the CM onto a horizontal plane. The first[21] definition by McGhee and Frank [117] deals with locomotion over a horizontal plane, using an *ideal legged locomotion machine*, i.e. the trunk is modeled as a rigid body, the legs massless and able to supply an unlimited force (but no torque) into the contact surface at the feet's contact points.

> An ideal legged locomotion machine is *statically stable at time* t if all legs in contact with the support plane at the given time remain in contact with that plane when all legs of the machine are fixed at their locations at time t and the translational and rotational velocities of the resulting rigid body are simultaneously reduced to zero.

McGhee and Frank then showed that their definition is equivalent to the condition $P_{CM} \in A_{sup}$. From this condition, they define the *static stability margin at a time*

[20] The trunk's centre of mass is often used instead of the entire robot's centre of mass.
[21] I.e. the earliest defintion found by the authors of this paper.

t, as the shortest distance from P_{CM} to the support polygon's boundary.

Song and Waldron [174] define[22] the *gait longitudinal stability margin* (over a stride of a periodic gait) to the be minimum of the distances from the P_{CM} to the front and rear boundaries of the support polygon. From this they define the static stability of a gait:

> A gait is *statically stable* if the gait longitudinal stability margin is positive, otherwise it is *statically unstable* (Song and Waldron [174]).

However, in this report a gait will be said to be statically stable if the static stability margin is positive at all times during the locomotion, i.e. the condition [76]

$$P_{CM}(t) \in A_{sup}(t) \, \forall t$$

is satisfied. Furthermore, the terms *static balance* and *statically balanced gait* will be used instead of static stability. We wish to emphasize the use of the condition as a strategy to maintain balance (or a constraint on the motions) in order to avoid falling over[23].

With a static balance requirement, at least four legs are required for locomotion if an ideal legged locomotion machine is assumed. For a quadruped, this greatly reduces the maximum speed (compared to a trot gait for instance). Consider the following reasoning, similar to one by Waldron et al. [196]. First assume that moving the legs vertically (including any footfall bouncing) takes no time. Then assume a creeping gait, a constant trunk velocity V_{trunk} and a maximum velocity of the foot with respect to the trunk, $\hat{V}_{leg}$. Let d denote the distance that the trunk translates during one step. This is also the distance that the leg must be transferred with respect to the trunk during the transfer phase. Then we have

$$d = \beta T V_{trunk}$$

for the support phase of the leg and

$$d = (1-\beta) T \hat{V}_{leg}$$

for the transfer phase, where T is the step time. This gives us the following relationship:

$$V_{trunk} \leq \frac{1-\beta}{\beta} \hat{V}_{leg}$$

[22] They only consider tipping over a lateral axes.

[23] Other criteria are also used to avoid falling over; Hirose et al. [74] for instance compare different energy based criteria. However, they are beyond the scope of this survey.

For a quadruped, static balance puts a limit on the duty factor, $\beta \geq 0.75$, resulting in

$$V_{trunk} \leq \frac{\hat{V}_{leg}}{3}$$

Compare this to a trot gait with $\beta = 0.5$ where the trunk velocity would be limited as:

$$V_{trunk} \leq \hat{V}_{leg}$$

This is of course one reason not to use static balance. Another reason, given by Raibert [149], is that mobility would improve, partly due to reduced foothold restrictions. Yet another drawback is that static balance is only valid as a criterion to avoid falling over for a system that is not in motion. As an example, consider what would happen if a robot that walks very fast suddenly stops: It would tip over due to the inertial forces, even though $P_{CM} \in A_{sup}$ up until the robot has tipped over. However, falling over a bit is not necessarily bad at all times. Hirose and Yoneda [76] suggest the concept of a *safe walk*, to be a walk where, if all joints are suddenly frozen, the system still ends up in a (statically) stable equilibrium. This concept does not imply a statically balanced gait, since falling is allowed as long as the system ends in a safe configuration. Note also that a statically balanced gait does not imply a safe walk, as illustrated by the tipping example above.

2.5. Dynamic balance

When a system does not use static balance, it should maintain a *dynamic balance*, where the compensation of tipping motions takes place over time. Dynamic balance is also referred to as *active balance* or *dynamic stability* in the literature [149]. In general the term *dynamic stability* seems very loosely interpreted within the "walking community" and a *dynamic gait* is often any gait that is not statically balanced at all times, i.e. $\exists t$ during the motion such that $P_{CM}(t) \notin A_{sup}(t)$.

Hirose [76] points out that a statically balanced gait can be used arbitrarily slow and defines a *dynamic walk* by writing:

> Under dynamic walk, the robot will begin to fall and will be unable to walk as planned when the walking speed is reduced to a level such that the dynamic effect of walking can no longer be expected.

This emphasizes the importance of the dynamic effects. For a statically balanced gait, we could express this mathematically as follows:

> For a statically balanced gait with joint motions $q(t)$, the joint motions $q(\epsilon t)$, $0 < \epsilon < 1$ should simply result in a slower version of the same gait.

Consider the trot gait for instance, where slowing down the joint motions would result in a completely different type of walking.

Another kind of criterion for a walking system to be *dynamically stable at a time* t is suggested by Karčnik and Kralj [24], where the system is said to be dynamically stable if it can stop in a statically stable configuration without changing the support polygon.

Vukobratović et al. [193] suggest defining a *stable locomotion system* by dividing the stability into three types[24]: *orientation and height stability*, *(trunk) path stability* and *stationary gait stability*. They emphasize that any definition of stability depends on a concept or a class of disturbances. Examples of disturbances used by Vukobratović et al. are external force disturbances and parameter variations in a finite time period. They argue that since legged locomotion is naturally cyclic[25], these disturbances can be considered as variations in the initial conditions for the next cycle.

Vukobratović's definitions were given for a biped on a horizontal smooth surface, but have here been modified to include a more general case.

- The orientation and height is considered stable if there exists a closed region R, which encloses the undisturbed trajectory of the three orientation angles and height, such that if disturbed by a disturbance $d \in \mathfrak{D}$, the trajectory returns to the region R as time goes to infinity. $\mathfrak{D}$ is a class of disturbances.

To define the path stability, some kind of nominal trajectory of the CM must exist.

- The path of the trunk is considered stable if the *average velocity vector* returns toward its original direction and magnitude after a disturbance $d \in \mathfrak{D}$. The average velocity vector is

$$\mathbf{v}_{av} := \frac{1}{T}\int_0^T \mathbf{v}\partial t$$

  where T is the period of a complete cycle.

[24] Vukobratović et al. used the terms posture stability and body path stability, but these were changed for consistency.

[25] By which Vukobratović et al. mean that specific characteristics (defined from case to case) in general tend to repeat.

A *stationary gait* is characterized by the following factors (defined/calculated over a stride) being constant: average forward velocity, stride length, relative leg phases, duty factor and stride duration.

- A stationary gait is considered stable, if the characteristic factors of the undisturbed system (represented as a point) lie within a volume, and if after a disturbance, the characteristic factors returns and remains within that volume.

2.5.1. Center of pressure and dynamic stability margin

The concept of static stability margin as a stability index, can be directly extended to include dynamic effects by using the *centre of pressure* instead of P_{CM}.

> The centre of pressure, P_{CP}, is defined as the point on the supporting surface given by the intersection of the supporting surface and a projected line from the system's center of mass along the direction of the resultant force on the system (Lin and Song [108]).

The *dynamic stability margin* can then be defined as the minimum distance between P_{CP} and the boundary of the support pattern. Alternatively, it can be defined as follows [108]:

$$S_d = \min_i \frac{M_i}{W_g}$$

where $W_g = m_{system}g$ is the weight of the robot and M_i, the resultant moment about the i:th border is calculated as

$$M_i = \mathbf{e}_i \cdot (\mathbf{F}_e \times \mathbf{r}^{GP_i} + \mathbf{M}_e)$$

where $\mathbf{F}_e$ and $\mathbf{M}_e$ are the resultant force and moment. The unit vector $\mathbf{e}_i$ points clockwise along the i:th border and $\mathbf{r}^{GP_i}$ is a vector from the systems centre of mass to any point on the i:th border. When M_i is negative, this corresponds to a moment that would tip the robot.

Note that there are other ways to define the center of pressure. By assuming that L contact points lie in a horizontal support plane, the vector from the origin to the center of pressure can be defined as follows

$$\mathbf{r}^{OP_{CP}} = \frac{\sum_{i=1}^{L} \left(\mathbf{F}^i_{gnd} \cdot \mathbf{n}_3 \right) \mathbf{r}^{OP_i}}{\sum_{i=1}^{L} \mathbf{F}^i_{gnd} \cdot \mathbf{n}_3}$$

where $\mathbf{r}^{OP_i}$ is a vector from the origin to the i:th contact point, $\mathbf{F}^i_{gnd}$ is the force applied to the machine at the i:th contact point. This is actually an alternative definition of the *Zero Moment Point* (ZMP).

2.5.2. Zero Moment Point

The ZMP was introduced by Vukobratović and Stepanenko [194, 195], where they suggest using the ZMP as a tool to plan motions. Following that lead, Shih et al. [168] (20 years later) use the ZMP as one of several criteria to verify that their biped's planned trajectories are physically realizable during the single support phase[26]. They define the ZMP as follows below, assuming a horizontal support plane.

> The ZMP, i.e. the vector from the origin, $\mathbf{r}^{OZ}$, and the corresponding moment, $\mathbf{M}_{ZMP}$, are defined through the following equations:
>
> $$\begin{aligned} \mathbf{r}^{OZ} \times \sum_{i=1}^{M} \mathbf{F}_e^i + \mathbf{M}_{ZMP} &= \sum_{i=1}^{M} \left(\mathbf{r}^{OG_i} \times \mathbf{F}_e^i + \mathbf{M}_e^i \right) \\ \mathbf{M}_{ZMP} \cdot \mathbf{n}_1 &= 0 \\ \mathbf{M}_{ZMP} \cdot \mathbf{n}_2 &= 0 \\ \mathbf{r}^{OZ} \cdot \mathbf{n}_3 &= 0 \end{aligned}$$
>
> where $\mathbf{r}^{OG_i}$ is the vector from the origin to the i:th rigid body's centre of mass. $\mathbf{F}_e^i$ is the (translational) inertial and gravitational force from the i:th rigid body motion, $\mathbf{F}_e^i = -m^i g \mathbf{n}_z - m^i \frac{{}^N\partial^2 \mathbf{r}^{OG_i}}{\partial t^2}$. $\mathbf{M}_e^i$ is the rotational inertial force from the i:th rigid body motion, $\mathbf{M}_e^i = -\frac{{}^N\partial J^i \cdot {}^N\omega^i}{\partial t}$, where J^i is the inertia dyad for the i:th rigid body and ${}^N\omega^i$ is the angular velocity of the i:th rigid body with respect to the inertial coordinate system N. The solution can be written explicitly in for instance the inertial coordinates (Shih et al. [168]).

Shih et al. [168] use the criterion that the ZMP must belong to the support area at all times, for the planned motion to be "stable". A problem with this definition is that it assumes that all contact points lie in a horizontal plane, which in general is unlikely when walking on irregular terrain. Takanishi and Lim [178] solve this by introducing different *virtual surfaces*. In their control of the biped WL-12, they plan compensating trunk motions[27] to ensure that the ZMP will be within the virtual surface, similar to the method used by TITAN IV and TITAN VI, described later in this paper.

[26] During the *single-support phase* , a biped is supported by one leg only.

[27] Vukobratović and Stepanenko [194] basically suggested this idea way back in 1972. They used biological data to fix the leg motions, and then used an algorithm to calculate the compensating trunk motions in order to specify the motion of the ZMP.

2.5.3. ZMP and stability

There are sometimes references in the literature indicating that keeping the ZMP within the support area will guarantee a stable gait. This is indirectly discussed by Vukobratović and Stepanenko [194]. They assume that all joints will track the planned trajectories perfectly and can then calculate the magnitude of disturbances that can be tolerated by the system (assuming a simplified model). In principle, this corresponds to the stability of a four-legged chair that is tilted. If it is given enough energy, it will fall over, if not it will fall back. The same reasoning approximately applies to the walking system, except that all joints are assumed to track their desired trajectories perfectly.

2.6. Miscellaneous joint and leg controller types

There are a lot of different control methods (position control, force control, impedance control etc) used as subparts within the controllers for walking machines. A few of them are listed below with references to where to look for more detailed information.

- *Position control* will be used to denote any method to track a reference position or trajectory. It will sometimes also include tracking not only of the position, but also of a velocity reference. Very often, simple P- or PI-control is used for position control.
- *Impedance control*, loosely put, means not only controlling the position (of a foot for instance), but also its dynamic behaviour. See for instance Tzafestas et al. [184] for an example of impedance control, or Hogan's three articles on impedance control [78, parts I, II and III].
- *Artificial Neural Networks* (ANN) includes a large variety of control methods. For a good book on the subject, see Haykin [64].
  Cerebellar modeled articulation controller (CMAC) is one example of an ANN that Lin and Song [109] use for hybrid position/force control of a quadruped. Kun and Miller have also used it for their UNH Biped [98, 99].

There are other methods such as stiffness control, damping control, combined stiffness and damping control etc, that Lin and Song [109] compare to their CMAC. Even more methods, such as fuzzy control etc are used in walking controllers, but are beyond the scope of this survey.

3. Controller examples

The next sections will describe a few examples of legged machines and describe how they are controlled. These examples were chosen after a brief survey of a lot of legged robots, so as to try and cover a broad spectrum of control principles. However, there is no guarantee that this was achieved, and important principles such as those used by hopping robots are not described in detail. Nor are there any examples of robots controlled by neural oscillators or ANN's for instance, only an example of a simulated robot (section 3.4). One practical criteria for selecting these groups were that there should be a reasonable level of information available, which is not the case for all robots (consider the Honda Humanoid robot for instance).

For each example, we have tried to include some information about the robot (e.g. physical properties), since we believe this to be relevant to the control. However, the main purpose of each example is to explain the principles of the controller and give a reasonable level of detail.

The first example (section 3.1) describes the controllers of three robots (TITAN III, TITAN IV and TITAN VI from Tokyo Institute of Technology), since their control architectures are very similar. The controllers are mainly deliberative, but vary from using pure position control (TITAN III) to partial force control (TITAN VI).

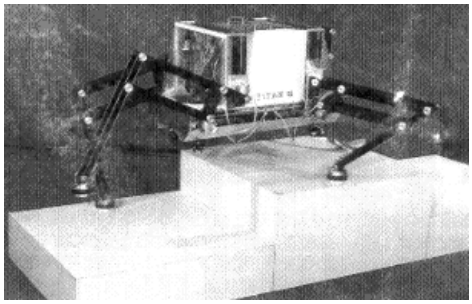
The second example (section 3.2) describes the controllers of three different robots (The ASV from Ohio State University, and Ralphy and SAP from Laboratoire de Robotique de Paris) that use very similar control principles. These controllers are also mainly deliberative, but the motion could be considered driven by the desired acceleration of the trunk.

The third example (section 3.3) describes an example of a hybrid DEDS controller of the robot Thing from University of Massachusetts. This is an example of a reactive controller.

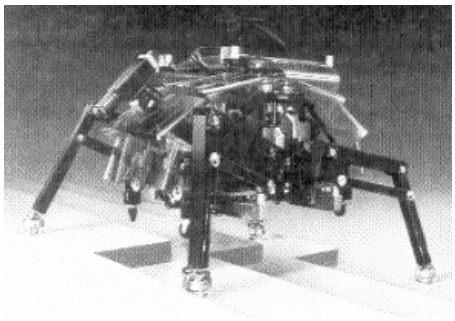
The final example (section 3.4) describes the biologically inspired control of a

simulated stick insect. This work was done at the University of Bielefeld. Although no specific robot was used with this controller, the ideas behind this controller have been used by others.

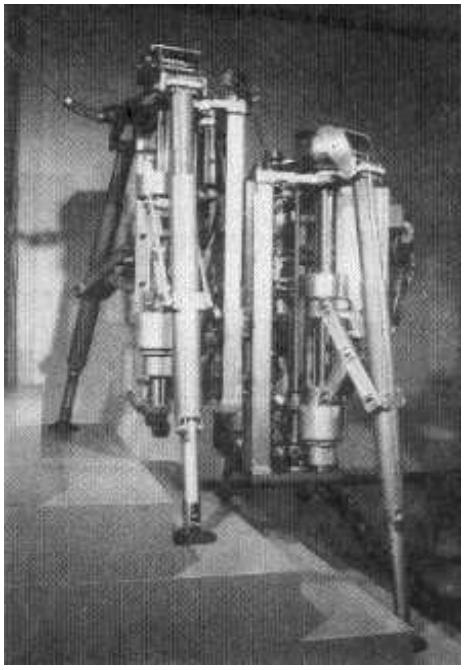
Note that the terminology has sometimes been changed with respect to the original references in order to achieve a more consistent description.



a) Titan III



b) Titan VI



c) Titan VI

Figure 3.1: The robots TITAN III (a), TITAN IV (b) and TITAN VI (c) [77].

3.1. Deliberative controllers I

This section will describe three examples of hierarchical, deliberative controllers and a posture control algorithm for rough terrain. These have been developed at the Tokyo Institute of Technology, in the Hirose and Yoneda Lab and used with the quadruped robots TITAN III [68], TITAN IV [71] and TITAN VI [69] [70].

The robots (figure 3.1) are actuated by DC motors and have about 1.2 meter long legs that are based on GDA-principles. They do however differ in mass and kinematics; TITAN III weighs 80 kg, TITAN IV weighs 160 kg and TITAN VI weighs 195 kg. TITAN VI also has a linear joint in the trunk, allowing it to better ascend steps. Furthermore, its feet have elastic padding for better performance on irregular surfaces.

Since the more recent controllers were based on the oldest controller, they have a common structure that is described below. Therefore, only the details that separate the controllers will be described in the following sections. The oldest controller

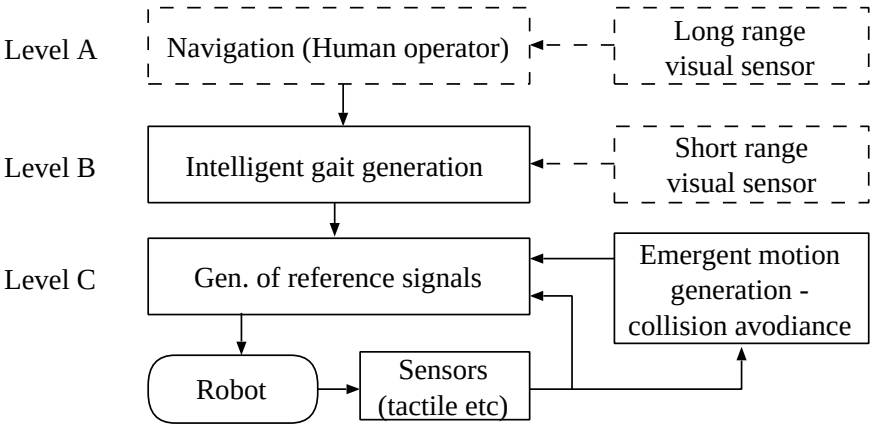


Figure 3.2: Overview of deliberative control structure of TITAN robots.

(section 3.1.1) only achieved statically balanced walking with TITAN III, whereas the other controllers (section 3.1.2 and 3.1.4) achieved dynamically balanced walking with TITAN IV and TITAN VI. The latter robot has also been used with the postural control algorithm (The Sky-Hook suspension, section 3.1.3).

The common structure

The main idea in these controllers is to combine feed-forward with feedback. Reference trajectories and/or gait parameters (feed-forward) are generated and tracked, and another (feedback) part copes with unexpected events (using reflexes) and terrain roughness. However, Hirose et al. found that simple reference tracking alone could achieve statically balanced walking [66] and even dynamically balanced walking [71].

Figure 3.2 illustrates a common structure of the controllers, where the dashed blocks represent a vision system, that was originally assumed to be available by Hirose et al. [66]. It was supposed to provide the controller with information about terrain type and height etc. Later, Yoneda et al. [209], stated that there (in 1994) were no such vision systems available and emphasized the need for a feedback part for rough terrain (such as the Sky-Hook suspension algorithm, section 3.1.3).

The control structure is hierarchical with three levels:

- Level A performs global path planning, giving directional commands resulting in a global path command, but the robot is only required to follow the path *approximately.*

- Level B is an "intelligent gait control system". This level performs two major planning tasks intermittently:
  - The global path is modified based on a local terrain map to avoid and pass obstacles, producing a local path (reference) for the trunk.
  - The gait is planned by determining parameters such as which leg(s) to swing, footfall position(s) and the trunk's translation and rotation during a step[1]. Planning is done for the next step during the current step, and assumes that the current plan will be accurately executed.
- Level C generates/tracks reference signals and generates emergency motions (i.e. reflexes). It is implemented as a sampled system and executes continuously.
  - The emergency motions block handles reflexes for events such as a foot striking an obstacle, by assuming command of the system. All (other) motions are suspended until the situation has been resolved (i.e., until the foot has been lifted over the obstacle).
  - The planning is based on accurate execution of the current step. Therefore, an "irregularity absorbing gait" is used to ensure a correct footfall (position), i.e. the other levels of the controller waits if necessary.

The next section will describe the controller for static walking in more detail.

3.1.1. Statically balancing controller

This section describes a controller that executes the statically balanced "standard crab-walk gait". Figure 3.3 illustrates the architecture[2] and contains more of the details in level B and C than figure 3.2. The gait will first be described briefly and then how the controller works.

The gait

The "standard crab-walk gait" is a combination of a free gait and a crab gait. A step is here defined [72] as the time interval between two consecutive footfalls. The algorithm that generates the support sequence selects the next transferring leg and foothold target in each step. If possible, it selects the support sequence of the crawl

[1] The exact definition of a step depends on the type of gait implemented, see the following subsections.

[2] See the reference [66] for details.

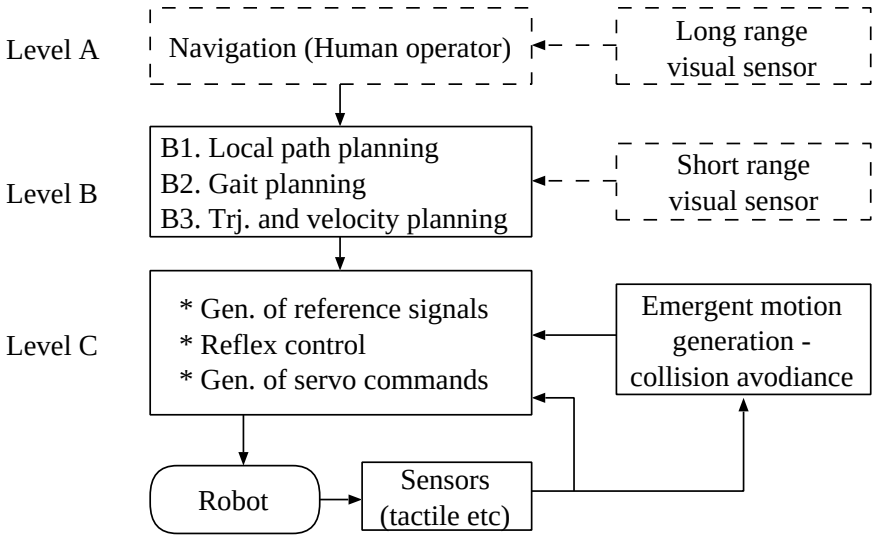


Figure 3.3: Controller for a statically balanced gait, see section 3.1.1 for details.

gait. See the references [72] and [66] for details about the algorithm. This algorithm was later extended by Hirose and Kunieda [73] to remove the requirements of prismatic leg workspaces and horizontal attitude.

The robot's orientation is fixed in the "standard crab-walk gait", but Hirose et al. have added turning through the use of the "standard circular gait" [67].

How it works

Level B is executed once during a step to plan for the next step. After generating the local path by modifying the global path ($B1$), the gait is first planned ($B2$) and then used to determine the parameters for the reference trajectories ($B3$). Level C generates and tracks the reference signals.

Horizontal gait planning The horizontal gait planning algorithm uses the feet's initial position, crab-walk angle, duty factor and information about leg workspaces to:

1. Determine which leg that will be transferred next, by checking if the standard leg-sequence of the crab gait can be used. Otherwise, the leg with the longest possible transfer distance is used

2. Determine the CM shifting distance during initial four-legged support phase, under a static balance constraint.
3. Determine the CM shifting distance during the three-legged support phase, under a static balance constraint.
4. Determine the next foothold of the transfer leg. Care is here taken to select a foothold that does not cause dead-lock later on. Map information is also used to exclude candidate footholds.

Vertical gait planning The vertical motion is planned as follows:

1. The (supporting) front feet are used to estimate the terrain height for the intersection point defined by the intersection of the line connecting the front feet and the (planned) horizontal CM trajectory.
2. The desired height at the next step switching point is then linearly interpolated from the current height and the necessary height over the intersection point.

Trajectory and velocity planning The planned trunk velocity is first reduced if necessary. Then the horizontal velocity of the transferring leg is determined based on the time it takes to raise the foot (assumes maximum vertical velocity). The time that should be spent in the up, transfer, down and support phases are also calculated. More detail about planning transferring leg trajectories can be found in reference [210].

Level C Level C is a sampled ($f = 50$ Hz [66]) controller, that tracks foot reference positions using P-controllers. The horizontal references are calculated by integrating the (velocity) parameters from level B and different parameters are used for each phase. To eliminate drift over several steps, the measured foot positions are used as initial values for the integration at the beginning of each step.

The vertical reference for the transfer foot is calculated similarly, but the reference height for the j:th supporting foot at sample time $n\Delta t$, z_j^d, is calculated according to:

$$\begin{aligned} z_j^d &= z_j^m(n\Delta t) + z_j^*(n\Delta t) - \mathbf{z}_j^m(n\Delta t) + \\ &\quad C_1(-x_j^m(n\Delta t)\theta_p + y_j^m(n\Delta t)\theta_r) + \\ &\quad C_2\Delta z \end{aligned}$$

where $\mathbf{z}_j^m(n\Delta t)$ is the average measured supporting leg height, and $z_j^*(n\Delta t)$ is the integrated desired vertical body velocity (initialized with $\mathbf{z}_j^m(0)$). Thus $z_j^*(t)$ gives the desired body height with respect to the average height. The terms on the first row

adds a vertical component that is common for all feet, i.e. the desired trunk height. The second row corrects for the trunk's roll, θ_r and pitch, θ_p, where the measured horizontal foot position comes from x_j^m and y_j^m. Finally, the third row is only added when a supporting foot accidentally loses contact with the ground.

Emergency actions The emergency actions are performed as follows (details in the reference [66]):

- If a transfer leg hits an obstacle, the robot stops all motions and performs a retract-and-elevate reflex. Normal motions will resume if the leg can pass the obstacle while moving forward slowly (and upward if a proximity detector indicates an obstacle). Otherwise, the estimated obstacle position is mapped as prohibited and a search for a new foothold (in level B) is done before normal operation resumes.
- If a transfer leg does not achieve footfall at a desired instant, the trunk motion stops immediately and the transfer leg descends vertically. If a foothold still cannot be found, this position is mapped as prohibited and a search for a new foothold is done before normal operation resumes.
- If a foothold at footfall is found to be unstable[3], the corresponding area is mapped as prohibited and a search for a new foothold is done before normal operation resumes.

The next section will describe a modified version of this controller that allows dynamically balanced walking.

3.1.2. Dynamically balancing controller — The expanded trot gait

To achieve dynamically balanced walking, Hirose et al. introduced the "expanded trot gait" and a modified controller [71]. The gait will first be described briefly (details in the reference [208]) and then how the controller works.

The gait

The "expanded trot gait" is designed to combine the advantages of the statically and dynamically balanced walking. Here a walking cycle consists of two *waves*, where a wave is the time interval from the lifting of a forward foot until the placing of the

[3] E.g. the contact sensors on the foot indicate that it is close to an edge.

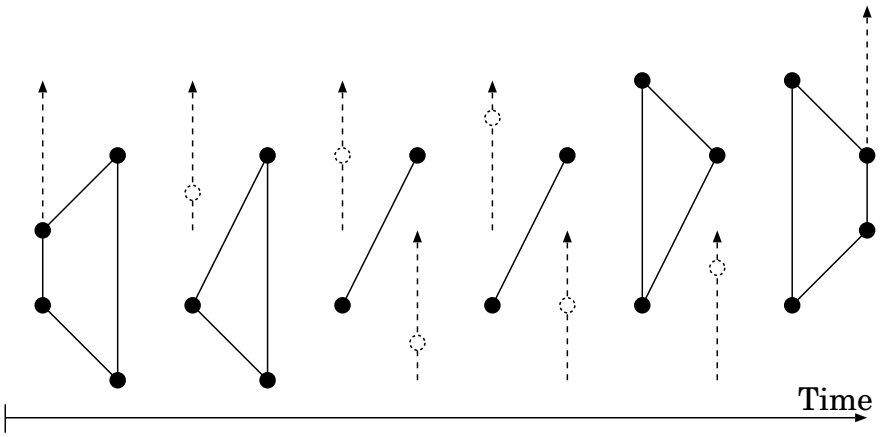


Figure 3.4: Diagram illustrating a "wave" in the expanded trot gait.

diagonal (rear) foot. There is always (at least instantaneously) a four-legged support phase when a wave ends and the next begins, as illustrated in figure 1.

Hirose et al. emphasize [71] that this is a safe walk, i.e. in case the robot stumbles, the feet can immediately be set down and the four-legged phase resumes[4].

How it works

Figure 3.5 illustrates the controller for the expanded trot gait. There are three major changes:

- The local path generation (B_1) produces a desired velocity in addition to desired body position and re-planning is done after a specific distance.
- The gait planning (B_2) is executed once for each wave, planning one wave ahead of time. It is now based on the expanded trot gait and the static stability criterion is not used, instead the body motion is now planned (C_1) to maintain dynamic stability using a ZMP criterion.

Horizontal gait planning The current segment of the commanded local path is approximated as a straight line, corresponding to a wave, and the horizontal motion is then planned:

1. Select the next leg to be transferred.

[4]There is of course no guarantee that it will not tip over, although it has all feet on the ground.

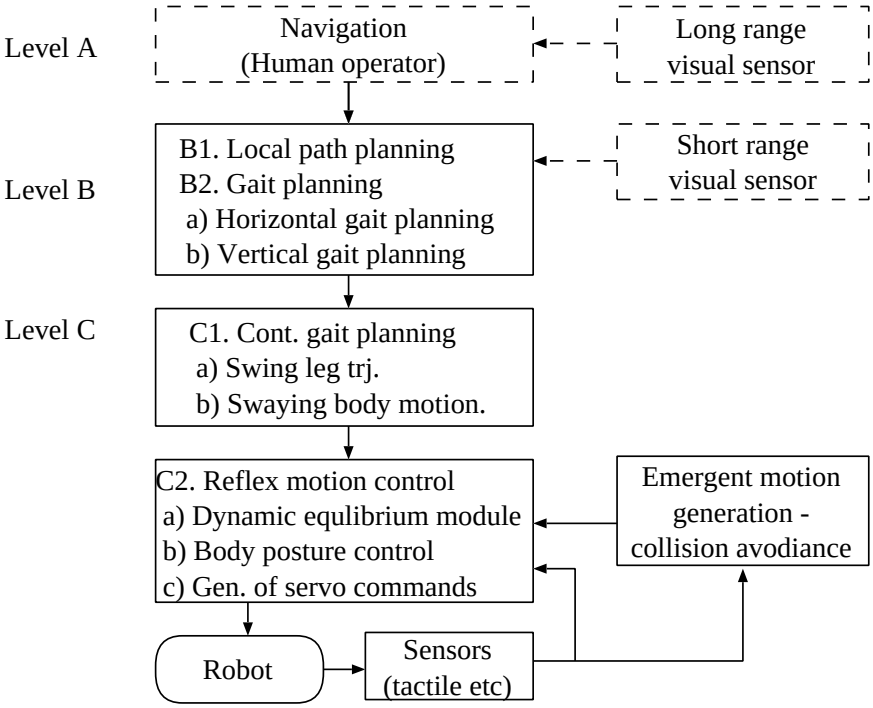


Figure 3.5: Controller for the expanded trot gait, details in text.

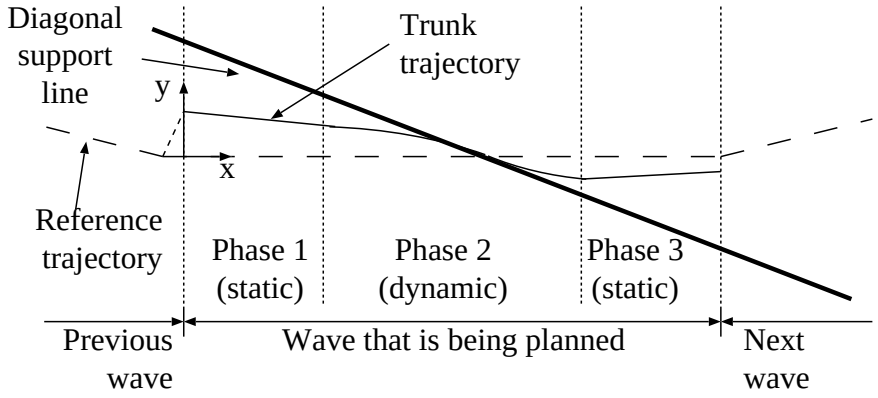


Figure 3.6: Planned trunk trajectory, details in text.

2. Do a map search for the next foothold. The start position of the search is determined from the following data:
   - The planned trunk CM position at the time of the next wave switch
   - Leg workspaces
   - An estimate of the duty factor based on the commanded local path
3. Determine trunk position at the next wave switch using the previously determined foothold.
4. Determine the velocity and (precise) duty factor based on the local path.

Vertical gait planning The vertical motion planning is done as described in the previous section.

Level C — continuous gait planning Level C_1 plans a continuous leg transfer trajectory and a trunk trajectory that maintains the ZMP within the support pattern (or on the diagonal support line during the two-legged support phase) [208]. Figure 3.6 illustrates a planned trunk trajectory, where each wave is divided into two statically balanced phases and one dynamically balanced phase. The following parameters are required for planning:

- The wave start and end times.
- The duty factor and duration of phases.

- Orientation, position and velocity of the trunk CM at the start of the wave.
- The desired heading in the first phase of the following wave.
- The desired CM velocity and footholds at the end of the wave.

The acceleration along the commanded straight line is constant during a wave and the orthogonal acceleration can change continuously to control the ZMP. Dynamic effects of leg motions are neglected.

The velocity in the y-direction (see figure 3.6) is chosen to be constant during the statically balanced phases and a slight velocity discontinuity is allowed during wave switching (i.e. when in a four-legged stance). To calculate the trajectory, y-acceleration is planned so as to maintain the ZMP on the support line. The resulting trajectory is calculated in local coordinates (x,y), transformed to a ground frame and finally to a trunk frame.

A side-effect of neglecting the dynamic effects of leg motions was that this caused unwanted trunk oscillations [207]. The algorithm in the next section, is useful to suppress these oscillations, but also to suppress ground disturbances, i.e. an inaccurate terrain map.

3.1.3. The Sky-hook Suspension

This section describes how the Sky-hook Suspension algorithm provides active suspension to suppress ground disturbances. It implements a virtual spring/damper system that "suspends" the trunk in an ground reference frame, by modulating the vertical leg forces.

The goal is to minimize trunk oscillations in a dynamically balanced walk, using a combined feedback and feed-forward algorithm:

1. Plan a suitable desired (feed-forward) leg force, assuming no disturbances.
2. Modify the desired force (feedback) to maintain the desired trunk orientation and height.

Each leg is force controlled in the vertical direction during the support phase and position controlled during the transfer phase. Switching between the two control types is activated by foot force sensors (and leg height thresholds to avoid chattering).

The desired vertical force, $F_{ff,l}$ for leg l, is calculated directly from the desired trunk torque and vertical force in the two- and three-legged stances. For a four-legged stance, the feed-forward force is planned based on a linear interpolation of the planned leg forces of the two- or three-legged stances just before and after the four-legged stance.

When in a two-legged stance, only the torque perpendicular to the support line is distributed since the legs are assumed to only apply forces. Yoneda et al. [209] argue that it works because there are also three- or four-legged stances (in the case of pure trotting, they argue that it works because of the alternating orientation of the supporting line).

A gyroscope is used to estimate the orientation, but to estimate the height, a weighted average is used,

$$z_G = \frac{-\sum_{l=1}^{4} F_{ff,l} z_{Gl}}{\sum_{l=1}^{4} F_{ff,l}},$$

where z_{Gl} is the height of each leg. The purpose of the weights is to eliminate discontinuities in the estimated height, which would otherwise cause discontinuous desired forces.

The next section describes a second controller for dynamic walking that is based on similar principles, but with a gait that is simpler than the expanded trot gait.

3.1.4. Dynamically balancing controller — The intermittent trot gait

Yoneda et al. [207] discovered that the transferring legs in their previous controller (section 3.1.2) caused the trunk to oscillate at high speeds, due to neglected inertial effects. To reduce these effects (still modeling the legs as massless), they introduced the *intermittent trot gait*, where the diagonal legs are swung in phase. The gait handles a range of duty factors ($0.5 \leq \beta < 1$) and allows omnidirectional translation and rotation around a vertical axis. However, it assumes walking on a horizontal surface and always attempts to maintain a horizontal attitude. This gait was tested on the TITAN VI and will be described next, followed by how the planning algorithm works.

The gait

Each cycle in the intermittent trot gait consists of two steps, as illustrated by the time intervals $[T_0, T_1)$ and $[T_1, T_2)$ in figure 3.7a. There is always a two-legged stance phase, (T_0, T_s), in each step and if $\beta > 0.5$, there is also a four-legged stance phase $[T_s, T_1]$, where T_s is defined by $T_s - T_0 = 2(1 - \beta)(T_1 - T_0)$.

As seen from the diagram, a pair of diagonal legs are always swung simultaneously. To simplify planning, each pair is therefore considered to constitute a virtual leg with a position and orientation. The position is illustrated (figure 3.7b) by the base of the arrow and the orientation by the direction of the arrow. Note that the

orientation corresponds to the direction of the support line plus an offset related to the "normal" leg configuration.

Since the distances between the corresponding feet and the virtual leg are assumed equal, it is straight forward to convert from virtual leg position and orientation to the real foot positions.

How it works

There are two major parts in the planning algorithm: planning of leg motions and planning of trunk motion. Prior to each step (i.e. at lift-off), the next virtual foothold is first planned. The trunk motion is then planned to maintain the attitude based on a ZMP criterion together with other imposed criteria that causes the trunk to "follow" the virtual legs.

Planning of virtual footholds The use of virtual legs allows a simple planning algorithm for the next virtual foothold based on the desired trunk linear and angular velocity. To plan the next virtual foothold, the desired velocities are multiplied with the transfer time to calculate a step length that is used to translate the virtual leg's foothold at the time of lift-off. The transfer time is derived from the duty cycle and step cycle time, given as parameters from a higher level. When standing still, a standard formation is planned.

Planning of trunk motion The trunk orientation is planned by integrating the desired angular velocity. Trunk motion orthogonal to the current support line (solid in figure 3.7c) during the two-legged stance phase is derived from a ZMP criterion. Otherwise, it is calculated as the integration of constant velocities for the different phases $\left[T_0, \frac{T_0+T_s}{2}\right)$, $\left[\frac{T_0+T_s}{2}, T_s\right)$, $\left[T_s, \frac{T_s+T_1}{2}\right)$ and $\left[\frac{T_s+T_1}{2}, T_1\right)$.

During the first two phases (during the two-legged stance) the motion perpendicular to the support line, is given by

$$\begin{aligned}\Phi_{ZMP}(t; T_0, y_0, \dot{y}_0) \quad &= y_0 \cosh\left(\sqrt{\tfrac{g}{H}}(t - T_0)\right) + \\ &\quad \dot{y}_0 \sqrt{\tfrac{H}{g}} \sinh\left(\sqrt{\tfrac{g}{H}}(t - T_0)\right)\end{aligned}$$

where g is the coefficient of gravity, H is the height of the trunk CM, y_0 and $\dot{y}_0$ are the distance and velocity orthogonal to the support line at $t = T_0$. This motion is derived from the condition that the ZMP remains on the support line (based on a simple linearized inverted pendulum model). The condition is necessary to maintain a constant attitude and there is no freedom in the choice of this motion. However,

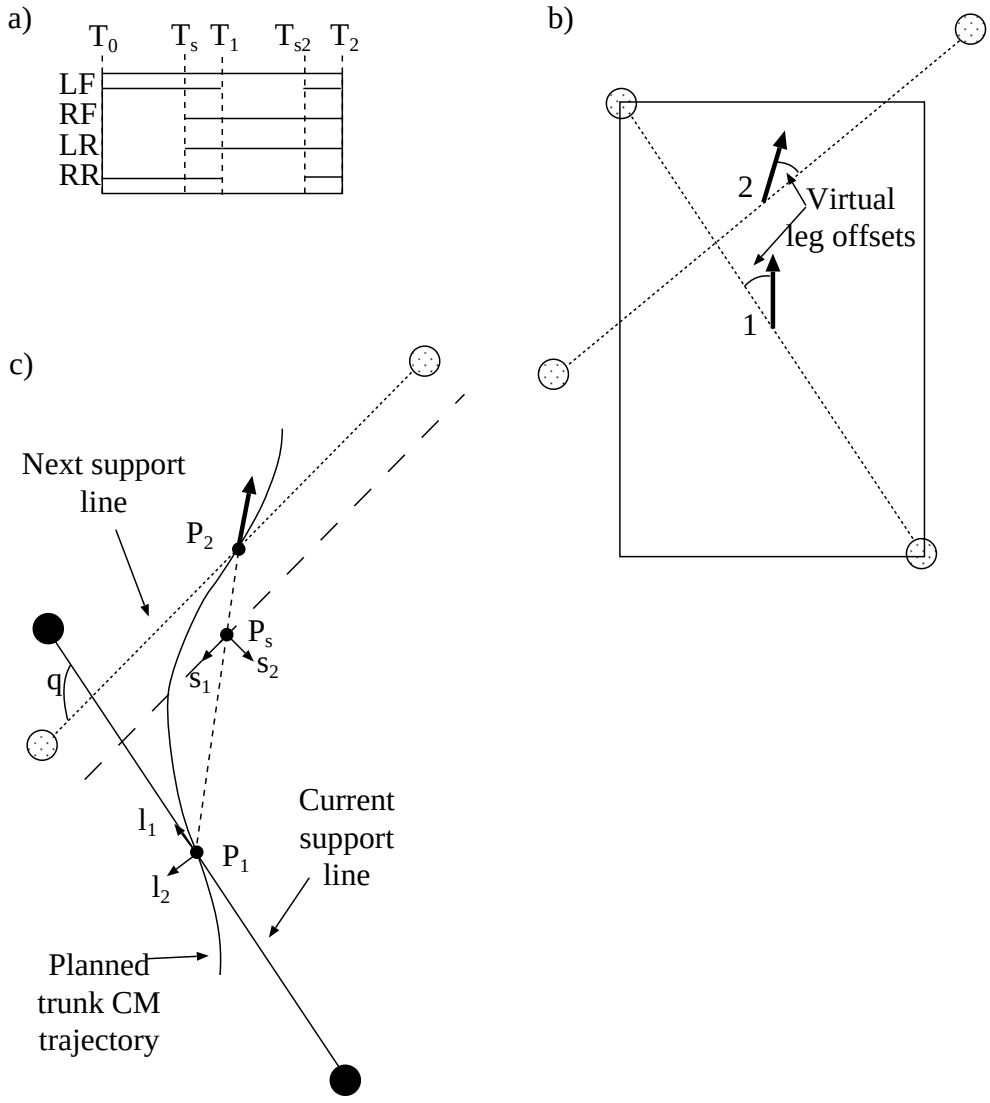


Figure 3.7: Illustrations of: a) the intermittent trot gait, b) virtual legs and c) reference frames. Details in text.

the motion parallel to the support line can be chosen more freely, but the velocities for the different phases above have to be chosen[5] so as to cause a convergence that keeps the trunk from "drifting" away from the feet. Furthermore, since there will be another two-legged support phase in the next step, this is taken into account by using a ZMP criterion together with a symmetry condition to derive the motion parallel to the current support line. To handle duty factors greater than 0.5, the virtual support line (dashed in figure 3.7c) is introduced by defining it as being parallel to next support line (dotted in figure 3.7c) and with a virtual foot at the point

$$P_s = \frac{(T_s - T_0)P_2 + (T_1 - T_s)P_1}{T_1 - T_0},$$

where P_1 and P_2 are the positions of the current and the next "supporting" virtual leg, respectively. This line is used, instead of the next support line, with the ZMP criterion to plan a suitable motion parallel to the current support line. In addition to the ZMP criterion, there are also criteria with the purpose of achieving small velocity variations and a trunk CM velocity at the time $t = T_1$ that is parallel to the line between the trunk CM position at times $t = T_s$ and $t = T_1$.

Finally, it should be noted out that there are some typographic errors in the reference [207] with the details of this algorithm. The reader should therefore be careful when reading it.

3.1.5. Summary and discussion

To sum up how these controllers work, let us begin with TITAN III.

TITAN III The trunk motion is planned based on the desired path, terrain data and a static balance criterion. This is then used to determine and plan the spatial trajectories of the feet. Finally, inverse kinematics (trivial here) gives the motion of the joints. However, this alone would not work well in irregular terrain, so balance is maintained by modifying the leg reference trajectories to achieve a horizontal attitude. The reason why the trunk should be kept horizontal, is of course the use of GDA:s. The support sequence is determined for each step using an algorithmic method, based on the standard crawl gait.

TITAN IV Actually, the control principles are very similar to that of TITAN III. The trunk motion is here planned to meet a ZMP criterion, i.e. the ZMP is kept within the support pattern at all times. Further, the algorithm that determines the support sequence is now based on the expanded trot gait.

[5] For simplicity, they are chosen to be constant during each phase.

TITAN VI The ideas for considering dynamic aspects have been further developed, based on the intermittent trot gait. By using this new gait, the algorithm to generate the support sequence is also much simpler. In addition, the Sky-Hook suspension algorithm explicitly considers the problem of maintaining a desired attitude. As a part of this, force tracking was introduced in the vertical direction during the legs' support phases.

A common aspect of these controllers is the fact that the trunk's motion (horizontally at least) is "kinematically driven", i.e. the controllers generate the desired trunk motion by moving the feet in a planned manner.

Performance

All of the machines have actually walked. The performance varies from 0.18 m/s (TITAN III) and 0.4 m/s (TITAN IV) up to 1 m/s (TITAN VI) on flat surfaces. Besides achieving the highest speed, TITAN VI could also walk (0.125 m/s) on irregular terrain[6].

The next section will now describe controller examples, where we consider the motion to be "force driven".

[6]The terrain was unknown and varied about 0.1 meter vertically.

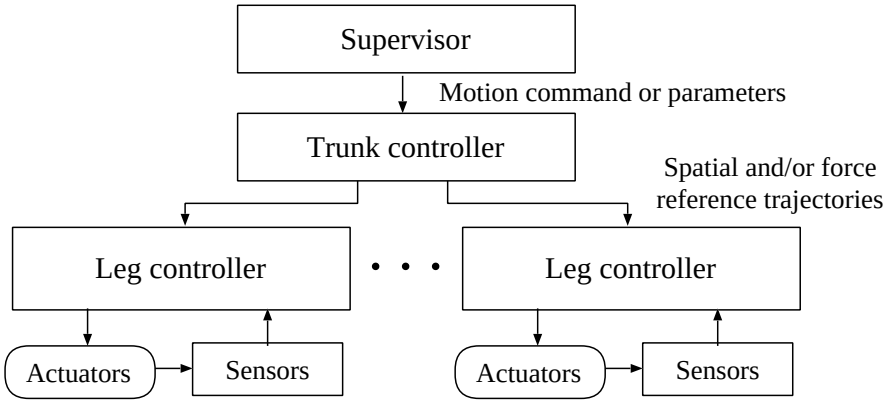


Figure 3.8: Overview of control structure.

3.2. Deliberative controllers II

This section describes two types of hierarchical, distributed and deliberative controllers. The first controller has been used with the Adaptive Suspension Vehicle (ASV). The ASV is a hexapod that was developed at the Ohio State University (1982–1990) as a "proof-of-concept" that a walking machine can be built at a useful scale.

The second controller type has been used[7] (in slightly different versions) with the robots RALPHY, SAP and BIPMAN. RALPHY and SAP have been developed at the Laboratoire de Robotique de Paris (LRP) and BIPMAN at Laboratoire de Genie de Biomecanique et Biophysique (LGMBP).

The common principle behind these controllers is described next, followed first by some information about the ASV (section 3.2.1) and then how the controller works (section 3.2.2). Some background on RALPHY, SAP and BIPMAN (section 3.2.3) is given, but since the basic principles are similar to that of the ASV, only the differences are really considered (section 3.2.4). Instead, the focus lies with some work at LRP in the generation of leg reference trajectories.

The common structure Figure 3.8 illustrates three levels in this common control structure. The supervisor provides a desired trunk motion, that is used by the trunk controller to generate commands to the individual leg controllers, one for each leg.

[7] We have only found simulation results.

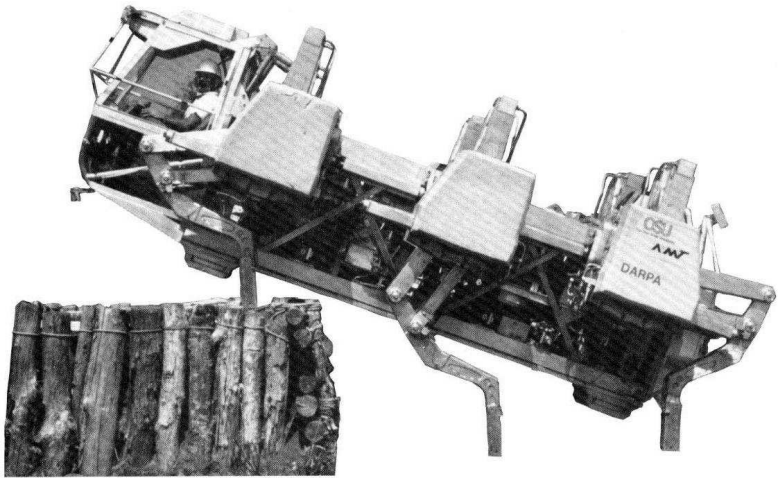


Figure 3.9: The Adaptive Suspension Vehicle weighs about 2700 kg, with dimensions $5.8 \times 2.2. \times 3.1$ metres (L $\times$ W $\times$ H) [139].

The main idea lies within the trunk controller, where a desired trunk acceleration is computed (based on the error in desired trunk velocity or trajectory). This acceleration is then used to compute a desired trunk wrench, i.e. a net force and torque on the trunk. Since this wrench must be applied by the supporting[8] legs, a force distribution algorithm is used to calculate desired forces from the individual legs. In essence, this is a kind of force or acceleration control on the trunk. Note however, that this requires some kind of force tracking capabilities in the leg controllers.

3.2.1. The Adaptive Suspension Vehicle

The ASV shown in figure 3.9 is a machine with impressive performance and will be described in some detail. Normal operation requires 37 kW (22 kW for mechanical work) and it is powered by an on-board gasoline engine.

Everything, including a human operator, is on board. The computer system includes two special purpose processors, one to sample and process terrain elevation data from a laser scanner (at 2 Hz), the other to compute force distributions (in 1 ms). To estimate position and orientation, there is also a specially developed inertial measurement system based on a vertical gyroscope, angular-rate gyroscopes and accelerometers.

[8] Inertial effects of transferring legs are ignored.

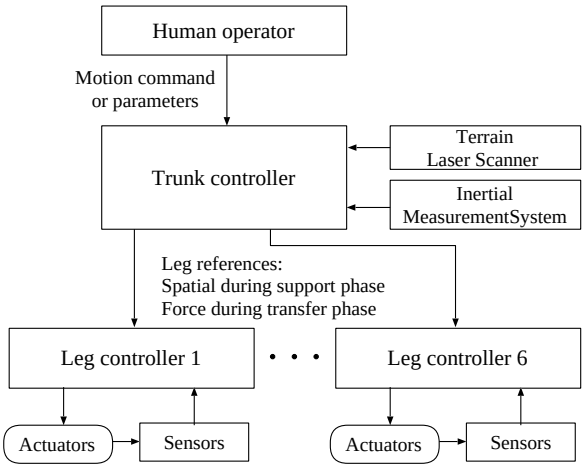


Figure 3.10: Overview of ASV control hierarchy.

The six legs each have three degrees of freedom in a planar pantograph configuration, where the actuators work independently in the sagittal plane. Velocity controlled hydraulic actuators are used[9], that for small signals allow velocity control up to 50 Hz and a response time less than 10 ms. Joint positions and velocities are measured using encoders, while differential pressure sensors are used to measure ground forces. To save weight, the legs have a compliant structure, which results in a low trunk resonance frequency (0.5 Hz for some axis).

Waldron et al. [196] describe design considerations, while Pugh et al. [146] give a comprehensive technical description of the vehicle and its systems, including its control. The feet and other parts of the ASV are also described by Song and Waldron [174].

3.2.2. The ASV controller

The ASV control hierarchy is illustrated in figure 3.10.

- The supervising is done by a human operator that also selects *operating mode*.
- The trunk controller plans motion and controls balance, based on the commands from the operator and operating mode.

[9] Each joint has its own variable displacement (flow) pump. A swash-plate in each pump regulates the flow and the plate is controlled by a rotary hydralic actuator, that in turn is controlled by a two-stage servovalve.

- The leg controllers use position control for transferring legs and force control for supporting legs.

The operating modes will be described next, followed by a brief description of how planning is done. Then the algorithm to control the trunk's motion is described and finally the leg controllers.

Operating modes

Each operating mode uses a slightly different algorithm to generate and coordinate the trunk and the leg motions. All but the obstacle crossing mode have been fully implemented and field tested. The operating modes are: utility, precision footing, close maneuvering, terrain following and follow-the-leader. They will be described in some detail, since some of them might be useful to implement in other walking machines.

1. The *utility mode* is used for start-up and shut-down (the ASV lies on the ground) allowing the operator to manually place and calibrate the legs.
2. The *precision footing mode* is used to manually control one leg at a time (or the trunk) without changing the support pattern. When controlling the trunk, the operator gives the desired velocities for all the trunk's degrees of freedom. The trunk control described in section 3.2.2 is used in this and the modes below. Switching between modes always passes through this mode.
3. The *close maneuvering mode* uses the tripod gait to walk over relatively smooth terrain with only small obstacles. The operator gives the desired horizontal and yaw velocities, while the roll and pitch are automatically controlled.

   Leg motions are generated by the planning software to minimize the transfer time, where the ground is assumed to be a plane through the supporting feet, or the terrain map is used. To avoid the automatic deceleration that occurs when the planner fails, (see p. 66) the desired velocities are limited by the following procedure:

   a) Calculate the maximum time the current tripod support pattern can be used, based on the given velocity command, leg workspaces and force constraints.
   b) Calculate the minimum leg transfer time, based on leg lift heights and terrain profile.
   c) Compare the times to determine the highest attainable speed.

4. The *terrain following mode* uses a free gait to walk over relatively smooth terrain with a moderate number of small[10] obstacles. The operator gives the

[10] Small relative to the vehicle.

desired horizontal and yaw velocities, while the roll and pitch are automatically controlled.

Leg lift height is calculated automatically from the terrain map and the legs are used in a sequence based on a free gait algorithm:

a) Calculate the temporal workspace limit for each supporting leg, i.e. the time before the leg reaches a workspace limit.
b) The next leg to be raised, is the one with the smallest limit that can be lifted without violating stability.
c) Raised legs are kept in a ready position until a foothold is found.

Footholds are automatically selected from the terrain map, to be as close as possible to a nominal position (based on the vehicle's velocity). Note that infrequent deadlocks are not an important planning problem, since the human operator can handle these manually.

5. The *follow the leader mode* uses the follow-the-leader-gait, for severe terrain with relatively few footholds. The operator gives the desired velocities for all the trunk's degrees of freedom and selects footholds for the front legs. Then the middle and rear legs automatically use footholds close to the front legs' footholds. The sequencing of the legs are fixed, but independent for the left and the right side, where the times of lift-off and footfall depend on the relative position of the footholds with respect to the trunk.
6. The *obstacle crossing mode* is used to cross vertical-step obstacles.

Planning

The exact use of the planning algorithms depend on the operating mode. Below, the more general principles for planning the spatial reference trajectories for the trunk and transferring legs are described; remember that the operating mode also affects the support sequence. During planning, the terrain elevation data is used to check if footholds are viable by estimating their slopes.

Trunk trajectory planning The planning software (and the balance control described in the next section) is used with all the modes except the utility mode. It determines how closely the commanded velocity can be tracked without violating the condition that the machine remains stable, here that the force distribution algorithm can find a feasible solution. Another aspect of the planning is that there is always a current plan, that consists of two parts. The first part aims to achieve the commanded velocity, whereas the second part contains a plan to decelerate and stop in a statically stable configuration. This means that if the planner fails to find a new trajectory, the second part can always be used to stop safely.

If the commanded velocity requires too large accelerations, e.g. they require too large leg forces, the planner iteratively changes the command to try and find an acceleration that does not violate the constraints. The planner neglects the effects of transferring legs.

Leg trajectory planning and generation Parameters for the transfer motions are first planned and used to generate position references. The effect of trunk motions is neglected and the trajectories are planned as follows:

1. The planner receives a motion command with constraints such as foothold position, velocity at footfall and time-window for the event.
2. The leg trajectory is partioned into segments and for each segment:

    a) The boundary conditions (endpoint position, velocity and time) are calculated assuming constant acceleration.
    b) A 5th order polynomial is fitted to the segment (results in smooth accelerations).

After planning, the leg trajectory is repeatedly generated as follows:

- The current 5th order polynomial is evaluated each 50 ms.
- The leg trajectory is modified with respect to kinematic limitations and leg collisions.
- A simplified 2nd order polynomial valid for 50 ms, is sent to the leg servo controller.
- The leg position controller evaluates the 2nd order polynomial every 10 ms to get the desired position.

Balance control

In addition to planning, the trunk controller (figure 3.11) tracks the desired trunk motion (sample time 60 ms). To do this, the trunk[11] servo needs accurate and fast motion estimates from the Inertial Measurement System. The motion error (in position, orientation and corresponding velocities) is multiplied with gains to compute a desired acceleration command. To this command a feed-forward term is added to generate a desired trunk acceleration and a simple inverse dynamic model then gives a desired trunk wrench. Finally, the force distribution algorithm calculates the desired forces for the supporting legs.

[11]The term “Body servo” has here been changed to “Trunk servo” for consistency in terminology.

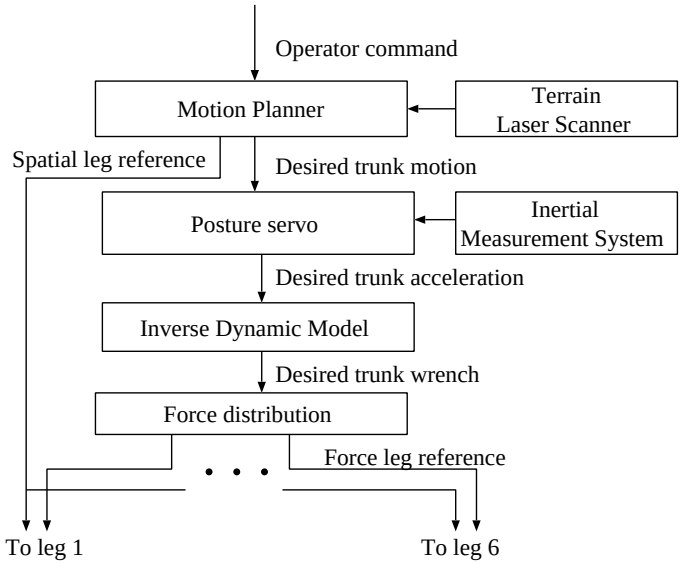


Figure 3.11: ASV trunk controller. Only one of the reference types are used at a time by each leg controller.

The implementation is reported to be stable with respect to the number of supporting legs and leg placement. It is also reported to work extremely well in soft soils (slippage) because of the force tracking.

Force distribution The task of the force distribution algorithm is to select a combination of leg forces that gives the desired trunk wrench. A criterion on the solution is that there should be no opposing leg forces and the implementation is divided into two steps for speed.

1. First preliminary estimates of the vertical forces are calculated and used as weights when determining the horizontal forces (using an approximation of friction coefficients), that achieves the desired horizontal forces and the yaw torque.
2. Then the vertical forces are determined to achieve the desired vertical force, roll torque and pitch torque.

If a maximum force constraint is reached, the corresponding force is fixed to its limit and the remaining forces are recalculated. In case too many limits are reached, a least square solution on the errors in the desired forces is used instead.

Figure 3.12: The photo (from [12]) shows the hybrid wheeled and legged robot SAP with dimensions $0.5 \times 0.3. \times 0.6$ metres (L $\times$ W $\times$ H) . The wheels are passive and the system is used to test leg reference trajectories and control.

Leg controllers

The leg controllers operate in either position or force control mode. In both modes, temperature variations, wear and leakage make the control difficult. This is compensated for by adding feed-forward terms that are estimated on-line. Note that the position control only uses a position reference and that the force control exploits the leg's compliance.

3.2.3. RALPHY, SAP and BIPMAN

This section gives a background on the robots RALPHY, SAP and BIPMAN and the next section will describe briefly how their controllers differ from the ASV controller. However, the focus is on work at LRP on the generation of leg reference trajectories.

The work on the control structure described in the next section began at the Laboratoire de Robotique de Paris (LRP) where first RALPHY and later SAP (figure 3.12) were built. RALPHY [191] [190] [125] is a quadruped, whereas SAP [61] is a hybrid legged and wheeled structure, built to study generation of leg reference trajectories and control. Lately, they have started working on an approach they call Controlled Limit Cycles [124] where they attempt to control the energy within the system to achieve and control fast leg motions. This is however beyond the scope of this report and not discussed further.

Around 1995, some researchers moved to the Laboratoire de Genie de Biome-

canique et Biophysique (LGMBP) and continued in a slightly different direction with the biped BIPMAN. The basic control structure [54] is very similar to that of RALPHY and SAP. However, Guihard and Gorce [60] [59] argue that BIPMAN's local joint impedance controllers are computationally more efficient than RALPHY's and SAP's leg impedance controllers [183] [184].

Another difference is that the work at LGMPB focuses more on higher level control, such as adaptive criteria under external perturbations [188] [40], learning [53] and postural control [55] [52]. The higher level control is essentially based on adapting the constraints on the force distribution algorithm with the Real-Time Criteria and Constraints Adaption (RTCA) architecture [188].

The control of BIPMAN is not further covered in this survey, due to space and time constraints, although we consider their work on the higher level of control to be quite interesting. However, we have so far only found simulation results on BIPMAN. This is also true for RALPHY, for which we have found no report that it has actually walked. SAP on the other hand has been used in real experiments.

The legs of RALPHY and SAP are about 0.4 metre long and weighs about 1 kg with two[12] revolute joints, a hip flexion/extension and a knee flexion extension joint.

The robots use pneumatic actuators that are controlled using an electropneumatic four-way servo-valve (torque motor+two penumatic stages) that is controlled by an electrical current. The valve controls the flow and regulates the chamber pressures, i.e. the joint torque, from an air pressure supply (10 bar).

3.2.4. The control

The control ideas originate from manipulation (Gorce et al. [56]). Originally, the goal was to track a spatial reference trajectory, assuming position control. Later this changed to impedance control that (with constant parameters) avoids the problems of switching controllers [184].

Figure 3.13 illustrates the control structure that is similar to that of the ASV. The supervisor here determines gait parameters (duty factor etc) and the trunk level is very similar to that of the ASV. One difference is however, that the leg references are both spatial references and force references. Another difference is in the implementation of the force control, see section 3.2.4.

Note that the discussion of the control structure in this section assumes its application to RALPHY (the application to BIPMAN is similar [60], except it has arms).

[12] Villard et al. [190] reported plans to add electric hip abduction/adduction actuators, with the motivation that this would be necessary for turning. We have not found any further information, instead it seems that SAP was built. However, note that it is not always necessary with three joints per leg in order to turn as illustrated by for instance the SCOUT robots [17].

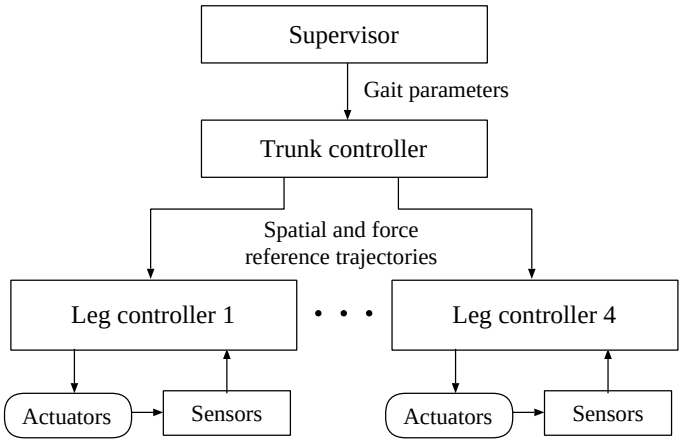


Figure 3.13: RALPHY Control structure.

However, the SAP implementation [61] needs to take special care of the fact that it has two passive wheels and we consider this irrelevant for walking systems.

The supervisor

Villard et al. [191] state that "according to the desired gait, we have identified the possible range of values of (β, ϕ) by means of several studies in animal locomotion", where β is the duty factor and ϕ is the relative phase of the legs (assuming a symmetric gait). Other parameters that come from this supervisory level are the footholds, stride length and frequency. However, we have found no work from LRP on this level except the use of fixed gait patterns (for walking straight ahead in simulations).

The trunk controller

The trunk control is very similar to that of the ASV and with the exception of the generation of leg trajectories, we have found little work on this level.

Leg trajectories

The ideas for planning leg reference trajectories at LRP have changed over time. From studying cursorial animals [190] Villard et al. concluded that

- "the length and the height of a stride increases with the speed"

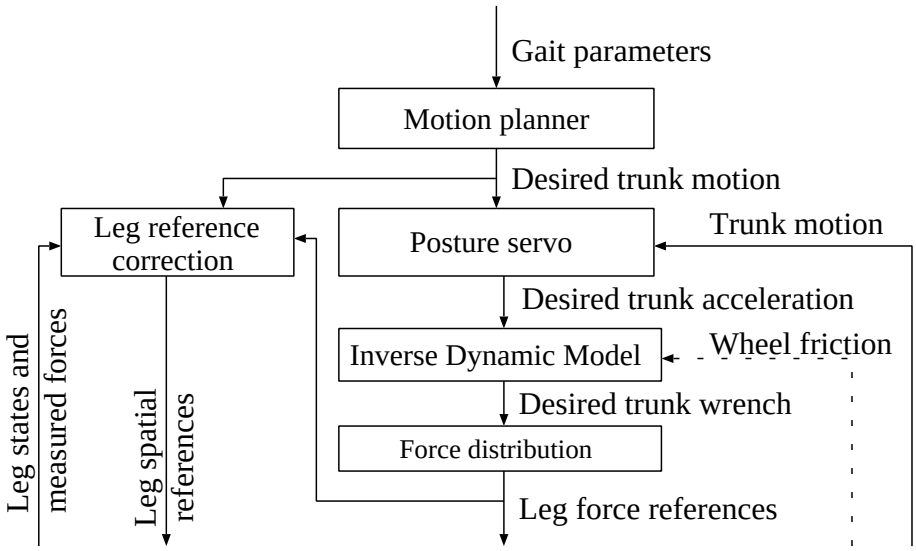


Figure 3.14: LRP trunk controller. Note the difference to the ASV (figure 3.11), the leg trajectories are modified here to achieve the desired force.

- "the legs are kept straight during" ... support

Based on these conclusions, and the goal to minimize footfall velocities, they suggested a prototype trajectory consisting of a cycloid (transfer phase) and an arc (support phase). However, Guihard et al. [61] tested this with SAP and found that it did not work well, since the robot barely moved as the feet slipped (only leg position control was used here). They therefore "rotated" the trajectory in the sagittal plane to "push" the feet more into the ground, but they still encountered problems. Eventually, they decided to introduce force tracking [184]. During the transfer phase they use the following reference trajectory (trunk frame[13]):

$$\begin{aligned} x_d(t) &= x_{d0} + r \cdot (\omega(t - t_k) - \sin(\omega(t - t_k))) \\ y_d(t) &= y_{d0} + r \cdot (1 - \cos(\omega(t - t_k))) \end{aligned}$$

[13] We think the frame is assumed to be horizontal and moving with a constant horizontal velocity.

where t_k is the time of footfall[14]. The spatial reference trajectory

$$\begin{aligned} x_d(t) &= x_{d0} + \frac{\Delta x}{\Delta t}(-t + t_k) \\ y_d(t) &= y_{d0} \end{aligned}$$

is used during the support phase, where Δx is the step length, Δt is the support duration and x_{d0} and y_{d0} reference positions related to ground parameters. The force reference trajectory is chosen as

$$\begin{aligned} F_{dx}(t) &= vt \cdot F_{dy0} + \\ &\quad + vtC\cos\left(\frac{\pi f t}{\beta}\right) - vtq\cos\left(\frac{3\pi f t}{\beta}\right) \\ F_{dy}(t) &= F_{dy0} + C\cos\left(\frac{\pi f t}{\beta}\right) - q\cos\left(\frac{3\pi f t}{\beta}\right) \end{aligned}$$

where C "quantifies the fraction of the weight supported by each leg" and v is the mean horizontal speed of the foot. β is the duty factor ($\beta = 0.75$) and $q = 0.2$. We recognize this as an approximation Alexander [3, eq. 6, eq. 19] made of foot motions, plus a constant force offset. The idea behind this is using a truncated Fourier series to approximate forces as measured from human and animals [7].

In the implementation [184], the ground (including the foot) is assumed compliant and an adaption stage is added to the coordination. This new part estimates the ground stiffness and modifies the spatial reference trajectory to achieve the desired force. Altering the impedance parameters could also achieve force tracking, but could cause instability. Too see how this is done, note that the steady-state force, F_{ef}, between the the ground and the leg can be written as:

$$F_{ef} = F_e(t \to \infty) = \frac{K_e}{K_r + K_e}(F_d + K_r(x_e - x_d))$$

where F_e is the ground force, F_d the desired force, K_e is the ground stiffness, K_r is the combined leg and leg control stiffness as implemented by the impedance controller. The foot reference position is x_d and the position of the ground is x_e (Note that x_e and x_d are column matrices here, not the horizontal position). The vertical parameter y_{d0} will now be calculated as:

$$y_{d0} = \hat{y}_e - \frac{F_d}{\hat{K}_e}$$

[14] It might be that t_k is different for the transfer and the support phase in these formulas, i.e. in one formula t_k is the beginning of the support phase, and in the other it is the beginning of the transfer phase.

where y_e is the ground position and $\hat{y}_e$ and $\hat{K}_e$ are the respective estimates. The ground parameter K_e and x_e needs to be estimated by the adaption law (see [184, eq 46] for details):

$$\begin{aligned} \hat{K}_e &= \gamma_{e1} y \cdot (\hat{F}_e - F_e) \\ \hat{y}_e &= \frac{\hat{F}_e - F_e}{\hat{K}_e}(-\gamma_{e2} - \gamma_{e1} y \hat{y}_e) \end{aligned}$$

where the prediction of the measured force is $\hat{F}_e = \hat{K}_e(\hat{y}_e - y)$ and γ_{e1} and γ_{e2} are positive constants that are estimation gains. We have so far only seen simulated results from this scheme.

Force distribution The force distribution algorithm uses a different method than that described in section 3.2.2. Here, the solution is found by solving a constrained linear optimization problem[15]. Below is an example given of how they formulate the problem for four supporting legs, in order to solve for time $t = t_{k+1}$:

$$\text{Minimize} \qquad \left|\left({}^{B}F_y^{H_{\text{left rear}}} + {}^{B}F_y^{H_{\text{left fore}}}\right) - \left({}^{B}F_y^{H_{\text{right rear}}} + {}^{B}F_y^{H_{\text{right fore}}}\right)\right|$$

under the constraints

$$\begin{aligned} \sum_{k \in \{\text{supporting legs}\}} {}^{B}F_y^{H_k} &= {}^{B}F_{c\ y}^{O}, \\ {}^{B}F_y^{H_k} &\geq \min_1, \forall k \\ {}^{B}F_y^{H_k} &\leq \max_1, \forall k \\ \left|{}^{B}F_y^{H_k}(t_{k+1}) - {}^{B}F_y^{H_k}(t_k)\right| &\leq \max_2, \forall k \end{aligned}$$

where ${}^{B}F_y^{H_k}$ is leg k's vertical force component in a trunk coordinate system. The equality constraint means that the forces must add up and the first two inequalities demand a bounded value and the last inequality enforces continuity over time.

Leg level

The leg controllers implement an adaptive impedance controller, adaptive both in the sense that it estimates leg parameters on-line and also in the sense that the level

[15] One of the main ideas within BIPMAN's control architecture is to vary the constraints to achieve different responses to disturbances.

above estimates the ground stiffness (in order to modify the reference trajectory to achieve the desired force). Here we will only mention that they report (from simulation) a 200 Hz bandwidth of the actuators servo-valve, and 70 Hz in the torque tracking. This is higher than that reported for a spring in series with an electric actuator [144]. However, the latter actuators have been used for a few years now, whereas we have only found simulation results on the previous. Details of this leg controller are beyond the scope of this survey, but see the article by Tzafestas et al. [184] for more information.

3.2.5. Summary and discussion

We consider the trunk motion to be "force driven" in these controllers, since the desired trunk motion is used to calculate a desired trunk acceleration. This is then used to calculate a desired force on the trunk, that is distributed among the supporting legs. Each leg is then controlled to achieve its desired force. The ASV uses direct leg force control, whereas RALPHY and SAP use impedance control. This scheme will, in addition to causing motion, also maintain the desired attitude. In the ASV, balance is also, in a sense, achieved by the existence of an emergency stop plan that halts the vehicle when for instance, no solution to the force distribution problem is found. The ASV either relies on the human operator or algorithms to plan the support sequence. In RALPHY and SAP, the support sequence is preprogrammed and fixed.

Performance

The ASV can walk with a maximum speed of 3.6 m/s (tripod gait) and ascend slopes up to 35%. RALPHY on the other hand seems to only have walked in simulation (about 0.4 m/s). The same more or less holds for SAP and BIPMAN, except that some experiments have been performed with SAP.

The examples in this section and those in the previous were very deliberative. In the next section a more reactive, behaviour based, controller will be described.

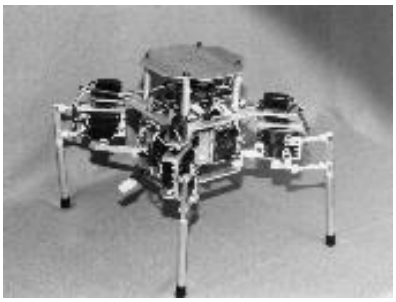


Figure 3.15: Photo of the quadruped robot Thing. The robot is 0.23 metre high and weighs approximately 2 kg [101].

3.3. A hybrid DEDS controller

This section will describe a controller based on a *hybrid discrete event dynamic system* (hybrid DEDS, section 3.3.1) for statically balanced walking. It has been created in the Laboratory for Perceptual Robotics[16] and although their research is focused on manipulation in combination with vision, MacDonald [110] built the quadruped *Thing* (figure 3.15) in 1994. Huber and Grupen then proceeded to work on machine learning, letting the robot safely explore how to turn [80] and walk[17] [81] .

The robot is 0.23 metre high, weighs approximately two kg and is actuated by 12 position controlled hobby servos. It can navigate around obstacles, using an IR-sensor in the front and walk over irregular terrain [111] [112]. An interesting point is that the elements of Thing's walking controller were originally used for manipulation and that Thing is actually capable of rotating an earth globe while lying on its back.

DEDS control has so far not been very common in robotics, but for instance Košecká and Bajcsy [166] have used it for navigation. Ramadge and Wonham [150] explain some basic theory, while Sobh et al. [172] give an indexed list of discrete event systems. The next section briefly describes the DEDS architecture, see Huber and Grupen [79] for details, while section 3.3.3 describes some of their work on machine learning.

[16] The laboratory is at the University of Massachusetts.
[17] Learning to walk has so far only been done in simulation.

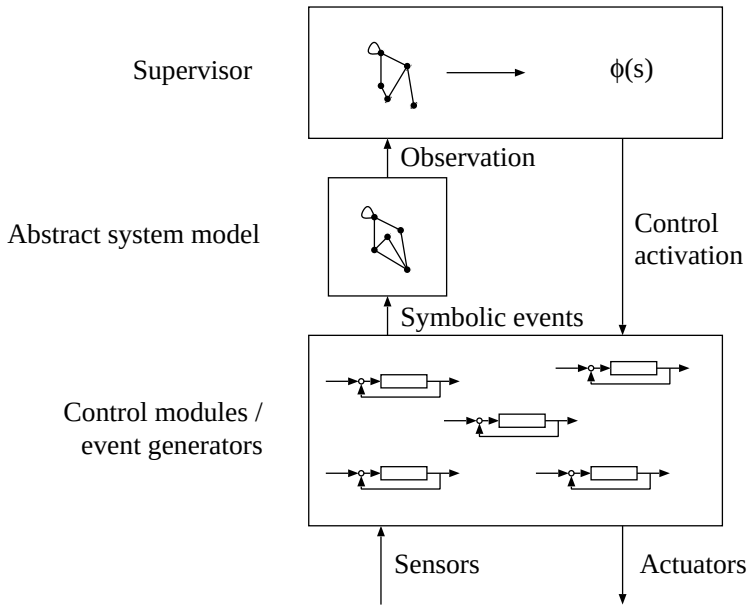


Figure 3.16: A hybrid discrete dynamic event system.

3.3.1. The control architecture

The control architecture is illustrated in figure 3.16 and consists of two parts, a supervisor and a set of continuous controllers. The supervisor observes the state of the abstract system model and the feedback map on the supervisor returns a *composition policy* that activates a subset of the continuous controllers, i.e. reactive behaviors.

The abstract system is modeled as a DEDS, i.e as a system that evolves with the occurrences of discrete events, which in this case are the activation and convergence of the continuous controllers. The model is realized as a finite state machine (FSM) over a predicate state space, where each component of a predicate vector indicates whether any one of a set of continuous controllers has converged, see section 3.3.3 for an example. Using DEDS methods, the state machine can automatically be generated, based on descriptions of the continuous controllers, i.e. which predicates that can be affected by a specific continuous controller. From this abstract model, a supervisor can then be derived automatically that observes the state based on the occurrence of events.

This is a hybrid DEDS, since the continuous controllers are used as an intermediate layer between the supervisor and the world. The controllers help to suppress

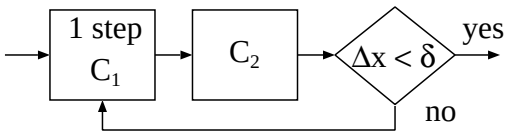


Figure 3.17: Approximation algorithm of the $C_1 \lhd C_2$ constraint.

model uncertainties and reduce complexity, by dividing the control problem into a continuous and a discrete part. However, it is important to use controllers with well defined characteristics, since the abstract model is based on these. Furthermore, since several controllers will be activated in parallel, the combined controllers must also have well defined characteristics.

By only allowing concurrent activation of *orthogonal* controllers, i.e. controllers that don't affect each other, their combined characteristics will be well defined. It is, however, enough that the controllers are orthogonal with respect to the "subject to" constraint, written as $\phi_i \lhd \phi_j$, meaning that controller ϕ_i is only allowed to have an effect on the *nullspace* of controller ϕ_j. In practice, this can be approximated using the algorithm shown in figure 3.17. The constrained controller is executed for one time step and the unconstrained controller is then allowed to converge. This is repeated until the change in state is less than a threshold[18]. The set of controllers that were used for walking will be described in the next section.

3.3.2. The control basis

A *control basis* is a set of *elemental controllers*, $\{\phi_0, \phi_1 \ldots, \phi_n\}$, that can be linearly combined (activated concurrently) in a *composition policy* to span a task space. For walking, the following elemental controllers are used:

- ϕ_0 — position controller
- ϕ_1 — contact controller
- ϕ_2 — posture controller.

These elemental controllers work in continuous time using feedback and can be looked upon as reactive behaviors. They are also generic, in the sense that they can be bound to different *input* and *output resources*, as illustrated in figure 3.18. As a consequence, the goal and action of an instantiated controller will depend on the

[18] We have not seen any proof that this will actually converge.

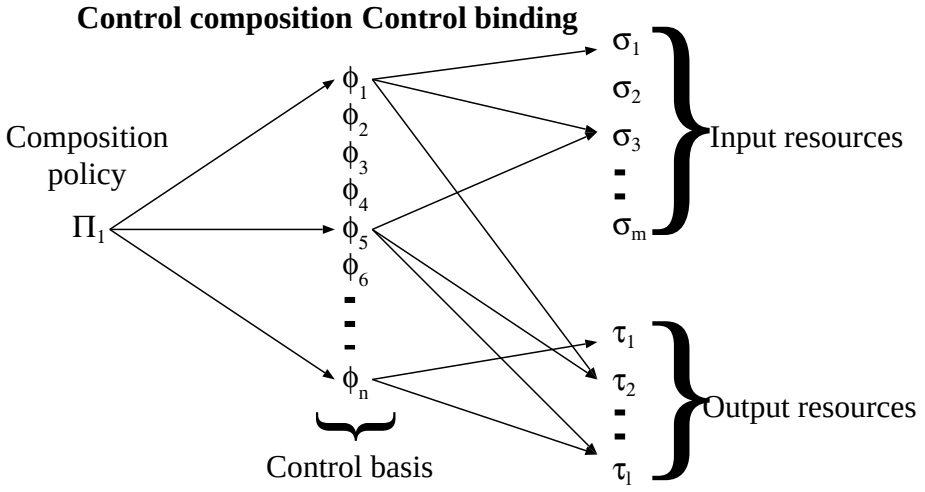


Figure 3.18: Illustration of controller binding and composition.

resources bound to it. The instance of a controller is denoted as

$$\phi_i \frac{\sigma_i}{\tau_i},$$

where σ_i denotes the set of input resources and τ_i denotes the set of output resources. If for instance $\sigma_1 \neq \sigma_2$, then most likely $\phi_i \frac{\sigma_1}{\tau_i}$ will not have the same effect as $\phi_i \frac{\sigma_2}{\tau_i}$, since the controllers do not use the same input resources.

What the actual input and output resources can be, depends on the specific elemental controller, but mostly they are the degrees of freedom corresponding to simple kinematic chains, like a robot arm or leg. It was found for the walking task, that there was no need to directly control individual degrees of freedom within the legs' kinematic chains. In this case, the resources are the foot positions $(\underline{1\,2\,3\,4})$, the position of the robot's centre of mass $(\underline{x\,y})$ and the robot's orientation $(\underline{\varphi})$. Note that abstract resources, such as the robot's orientation, can be used as well as joint angles. Also note that the references use a different leg numbering than in this study, i.e. the resources denoted $(\underline{1\,2\,3\,4})$ here are denoted $(\underline{2\,1\,3\,0})$ in the references.

The position controller

The position controller was originally used for reactive path-planning in manipulation tasks [29]. Depending on the assigned resources, the controller is capable of

either navigating a robot through a cluttered environment, or steering a robot leg away from an unsuitable foot-hold.

A potential field approach based on harmonic functions[19] is used to do path-planning in the configuration space. Harmonic functions have special properties, that guarantee that a gradient descent path leads to the goal, if a path exists. If there is no path, the gradient will be zero which is a condition that can be detected.

Obstacles and the goal are represented as boundary conditions with different values. When a new obstacle is detected, it is added as a new boundary condition and the field is recalculated, i.e. re-planning the path.

Formally, the problem can be cast as solving

$$\nabla^2 \Phi = 0, \text{ on } \Omega \subseteq \Re^n$$

with Dirichlet boundary conditions

$$\Phi \mid_{\partial\Omega} = \begin{cases} 0 & \text{goal} \\ 1 & \text{obstacle} \end{cases}$$

giving a solution that will be denoted Φ_D. The flow of this solution will be orthogonal to the obstacles. Using a von Neumann boundary condition

$$\frac{\Phi \mid_{\partial\Omega}}{\partial \mathbf{n}} = 0,$$

where $\mathbf{n}$ is the surface normal, another solution, Φ_N, is obtained where the flow will be parallel to the obstacles. Since Laplace's equation is linear, the solutions can be superimposed into

$$\Phi = k\Phi_D + (1-k)\Phi_N,$$

thereby varying the flow of the solution, i.e. how close to obstacles the robot will go.

Trajectories can be generated as

$$\dot{q} = K_\nabla \nabla \Phi,$$

where $\dot{q}$ is a vector with the desired velocities (in configuration space) and K_∇ is the velocity gain. To reactively handle obstacles that the robot hits, an admittance controller can be added by calculating the desired velocity as

$$\dot{q} = K_\nabla \nabla \Phi + K_{EQ} A(q) w,$$

[19] Harmonic functions are solutions to Laplace's eqation: $\nabla^2 \Phi = \sum_{i=1}^{n} \frac{\partial \Phi}{\partial q_i} = 0$ where q_i is a generalized coordinate.

where w is the external force in joint forces, e.g. joint torques, $A(q)$ is the admittance matrix and K_{EQ} is the admittance gain. For $n = 3$, a heuristic admittance relation can be given by the relation

$$A(q)w = -(\nabla\Phi \times (\nabla\Phi \times w)).$$

This simply means that the admittance induced velocity is orthogonal to the obstacle and to the gradient descent direction.

The contact controller

The contact controller attempts to achieve a stable stance, by minimizing the force and torque residuals, $F^T F$ and $M^T M$, where $F = \sum_{i=1}^{4} f_i - mge_z$ is the net force on the robot's trunk and M is the net torque on the center of mass. First the force residual is minimized through a gradient descent method until the force error is below a threshold. Then the torque residual is also minimized in a gradient descent method, see [25] for details.

The force from each foot is calculated as $f_i = J_i^{-T}\tau_i$, where

$$J_i^{-T} = \left(\frac{\partial r^{Trunk\rightarrow Foot_i}}{\partial q}\right)^{-T}$$

is the inverse transpose of the Jacobian of the vector between the trunk and the foot i. The joint torques, τ_i, are estimated by comparing signals (more or less motor current) in the servos' controllers to precalibrated force data.

The controller tries to place the output resource in such a way that the expressions are minimized. This requires an estimation of the surface orientation, that is obtained by using the foot as a probe. The foot is repeatedly moved downwards until the contact force exceeds a threshold. This position is then stored and several of these positions are used to fit a flat surface model to the data.

As an example, the goal of the contact controller

$$\phi_{1\underline{1}}^{1\,2\,3},$$

is to achieve a stable stance on the input resources, legs 1, 2 and 3, by moving the output resource, leg 1.

The posture controller The posture controller maximizes a heuristic posture measure,

$$m = \prod_{i<4}\left(p_i \prod_{j<3}\cos(\theta_{\epsilon_{ij}})\right),$$

based on the manipulability[20] measure,

$$p_i = \det\left(\sqrt{J_i J_i^T}\right)$$

where $\theta_{\epsilon_{ij}}$ is the joint angle for leg i and joint j, normalized to $[-\pi, \pi)$. It is implemented as a gradient descent method on the posture measure. An example of the posture controller is:

$$\phi_2 \underline{\varphi}^{1\,2\,3},$$

where the input resources are the kinematic chains of legs 1, 2 and 3 (i.e. in practice the foot positions), while the output resource is the yaw-angle of the robot. The goal of the controller is to rotate the robot around the robot's center, optimizing the posture measure.

Supervisors

In 1996 MacDonald [111] designed a supervisor to make the robot walk over flat horizontal terrain. The supervisor is illustrated in figure 3.19, where the result of the feedback map for each state is shown within the state. The solid lines indicate the current supporting polygon, while the dashed lines indicate the support polygon that the contact controller must achieve in order for the state transition to occur.

The precondition is that the robot is in a stable four-legged stance. In this gait, the robot walks in the x-direction, first moving the left rear leg, followed by the left front, right rear and right front leg.

Table 3.1 describes what happens in each state. This supervisor only walks straight ahead. In order to navigate around obstacles a supervisor for turning was also created. A composition of the two supervisors was then used, where the turning gait was used whenever the yaw error exceeds a threshold. The yaw error is the difference between desired heading and current heading as estimated by odometry.

Since the position controller is reactive and can incorporate new information about obstacles, the combined supervisors were capable of navigating to a goal point, while walking through an unknown obstacle field.

Going from flat terrain to irregular terrain

The supervisor for flat terrain was tested on irregular terrain, but no longer worked, since some of the transitions never occurred. The problem was solved using the same control basis, by modifying the supervisor and the contact controller:

[20] For a book on robotic control that includes manipulability, see the book by Murray, Li and Sastry [126].

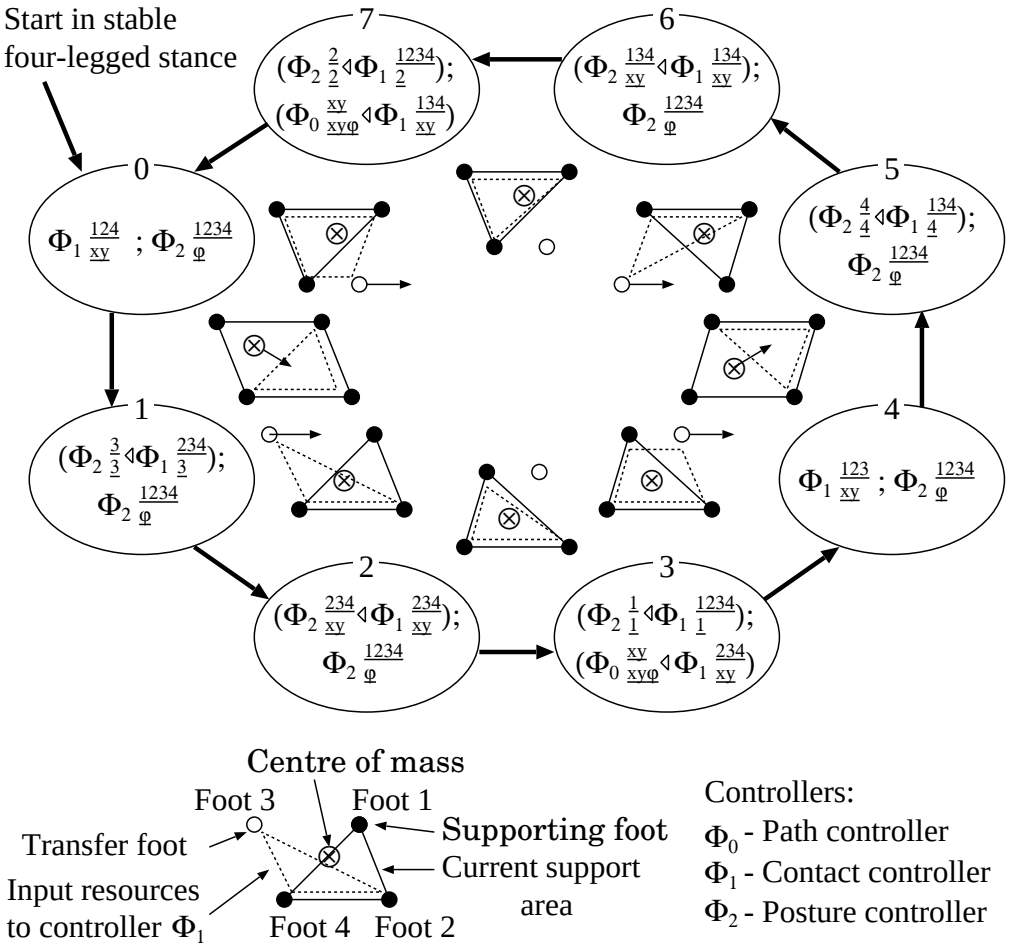


Figure 3.19: A supervisor for walking over flat terrain.

Table 3.1: Description of supervisor for flat terrain.

State	Description
0	The center of mass is moved, so that a stable stance is achieved using leg 1, 2 and 4. At the same time, the orientation is optimized.
1	Leg 3 is placed (kinematically optimized), so that the center of mass is also within the polygon of leg 2, 3, and 4. At the same time, the orientation is optimized.
2	The center of mass is kept within the supporting polygon of leg 2, 3 and 4, while it is moved to position that optimizes the manipulability measure. At the same time, the orientation is optimized.
3	Leg 1 is placed (kinematically optimized), so that the center of mass is within the polygon of leg 1, 2, 3, and 4. At the same time, keep the center of mass within the supporting polygon of legs 2, 3, and 4, while moving the center of mass according to the navigation controller.
4	The center of mass is moved, so that a stable stance is achieved using leg 1, 2, and 3. At the same time, the orientation is optimized.
5–7	These states are similar to the earlier states, but with different legs and support polygons.

- The probing behavior was added to the contact controller, in order to obtain information about surface orientation and location.
- Avoidance of footholds that are close to edges of surfaces was added, using the position controller on leg, subject to the contact controller.
- Vertical posture optimization was added.
- In four-legged stances, one foot was allowed to be dragged on the ground while the trunk was moved. This effectively increased the workspace of the trunk relative to the other footholds.
- Failed convergences was handled by adding two states.

Figure 3.20 illustrates the modified supervisor. This supervisor worked in experiments where the robot walked over unknown irregular terrain consisting of horizontal planes, each one cm high, placed on top of each other at irregular angles. An experiment took about 13 minutes, where the robot repeatedly used over 50% of the vertical workspace, when it crossed three planes at once.

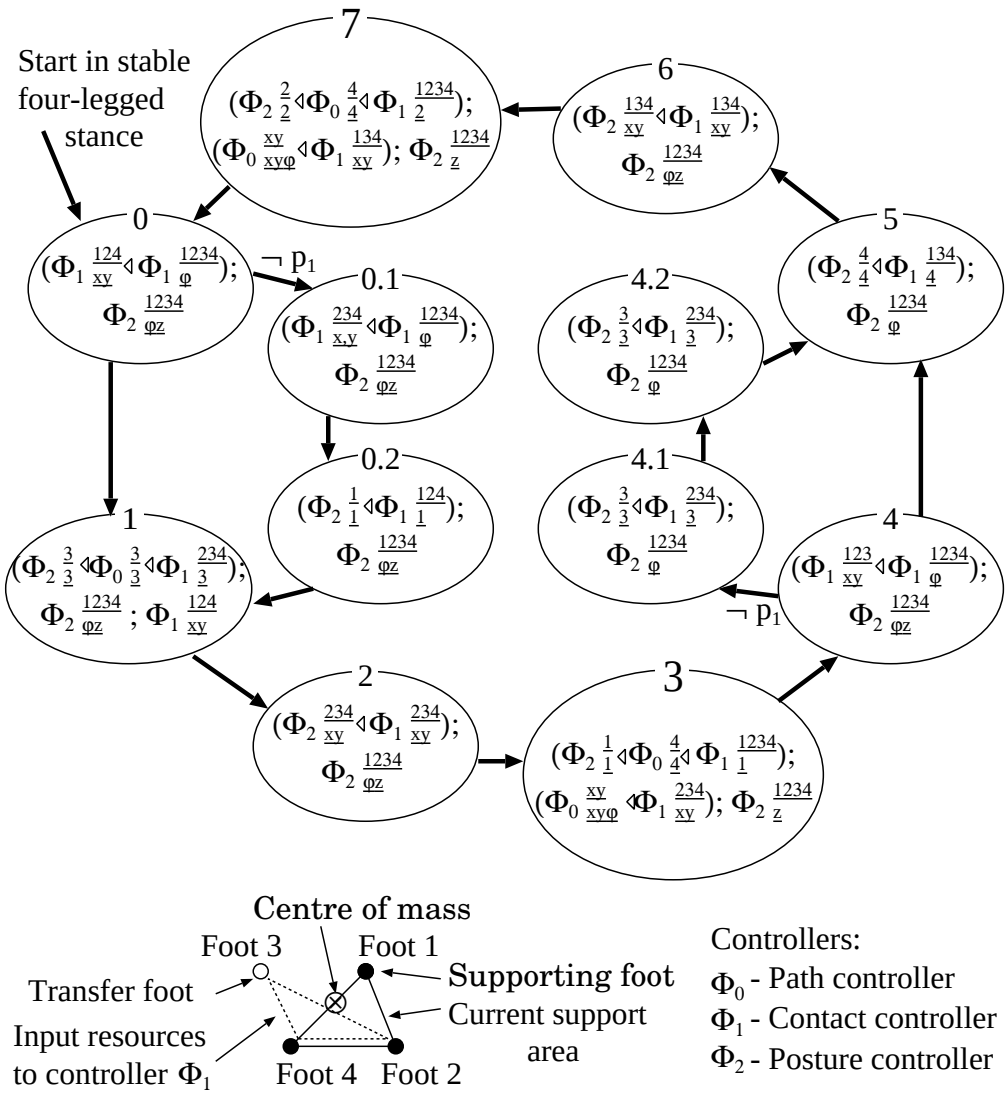


Figure 3.20: Supervisor for walking over irregular terrain.

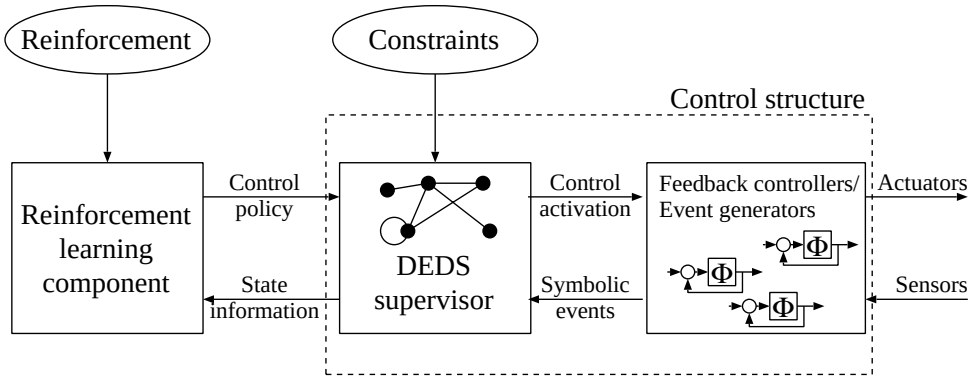


Figure 3.21: The control architecture.

3.3.3. Learning Thing to turn

The control architecture with a *reinforcement learning component* added is illustrated in figure 3.21. The purpose of the reinforcement learning component in the control architecture, is to learn a feedback map on the supervisor that achieves a specific task. As a demonstration of this technique, Huber and Grupen [80] let Thing use exploration with *Q-learning* to learn a counterclockwise turning gait. In that experiment, they only allowed controllers instantiated as

$$\phi_1 \underline{^{a\,b\,c}_{a}} \quad a \neq b \neq c \,\text{and}\, a, b, c \in \{1, 2, 3, 4\},$$

with the goal of reaching stable tripod stances and the controller

$$\phi_2 \underline{^{0\,1\,2\,3}_{\varphi}},$$

that optimizes the orientation of the robot. These 13 controllers can then be combined using the $\lhd$ operator, resulting in 1885 actions available from each state. A state in this experiment was defined as $p = (p_1, p_2, p_3, p_4, p_5)$ where the predicates p_1, p_2, p_3 and p_4 represent the convergence of controllers such as

$$\phi_1 \underline{^{1,2,3}_{*}} \qquad (* \text{ indicates any resource}),$$

i.e. stable tripod stances. The predicate p_5 indicates the convergence of

$$\phi_2 \underline{^{0,1,2,3}_{\varphi}}.$$

This gives 2^5 different states, with 1885 composed controllers to choose from at each state. Since exploration is done in the real world, it is important that the

exploration is limited to safe states, i.e. the constraint

$$p_1 \vee p_2 \vee p_3 \vee p_4$$

should be satisfied. Additional knowledge is incorporated through the constraint

$$\neg(p_1 \wedge p_3) \wedge \neg(p_2 \wedge p_4),$$

i.e. that two diagonally opposite tripod support patterns can not be used simultaneously. Controllers that could violate these constraints are not allowed, reducing the number of possible composite controllers available from each state to approximately 50. From each of the 16 stable states, transition is possible to on average about six states. In practice, the transitions will depend on kinematic limitations and the environment. The system can therefore be viewed as nondeterministic.

System identification was done by using a frequency count to approximate the function $p(x, a, y)$, i.e. the probability of transitioning to state y, when in state x and taking the action a.

The reinforcement learning was implemented as Q-learning, where the quality function

$$Q(x, a) = \sum_{y \in X} \left(p(x, a, y) Q(x, a, y) \right),$$

encodes the expected quality of taking action a from state x and the partial quality-function, $Q_t(x, a, y)$, is updated as

$$\begin{aligned} Q_t(x, a, y) &= (1 - \beta) Q_{t-1}(x, a, y) + \\ &\quad \beta \left(r_t + \gamma \max_{b \in A} Q_{t-1}(y, b) \right) \end{aligned}$$

at step t, where r_t is the reward for taking action a.

In the experiment, the reinforcement was defined as

$$r_t = \varphi_t - \varphi_{t-1},$$

i.e. the relative change in orientation in the current step. Initially, the robot was placed in a stable configuration and the exploration-level was initially 100% and incrementally decreased down to 10% in 1000 time steps. After approximately 500 control steps, that took 11 minutes, the turning rate settled down to fluctuate around about 0.36 radians/step.

The resulting supervisor contains a main cycle consisting of four states, where 98% of the transitions take place.

3.3.4. Summary and discussion

Walking is achieved by a supervisor that sequentially activates continuous feedback controllers, thus dividing the control problem into the design of a supervisor and a set of continuous controllers. Starting with a control basis, resources are bound to elemental controllers. The subject to operator $(\triangleleft)$ is then used to compose additional controllers and since the elemental controllers have well defined goals and characteristics, that will also be true for the composed controllers.

Based on these characteristics, formal DEDS methods are used to automatically generate a supervisor/observer over a predicate space representing convergence events. The problem is now to design a feedback map on the supervisor, that activates the correct sequence of controllers. In practice, the feedback map can be designed manually or found by reinforcement learning. Theoretical synthesis methods exist, but fail since the system is too large, complex and non-deterministic.

We consider the motion of the trunk to be "kinematically driven", since the controller moves the trunk by controlling the foot positions using inverse kinematics. However, this is not a deliberative system, since in the placing of a transferring leg, it is "pushed" ahead by the trunk through the use of the manipulability measure. The motion of the trunk is similarly the result of trying to optimize this measure. Balance is maintained, by always having a behaviour active that ensures a static balance criteria, whereas the support sequence emerges as a function of the encoded feedback map.

Performance

Thing walks very slowly, about four minutes for one metre on a flat surface, more for irregular terrain. The resulting gait is either sequential or non-sequential depending on the supervisor, but it is always aperiodic because of the way it is generated. An advantage is that DEDS methods can be used with additional knowledge about unwanted states to restrict the set of controllers that can be activated in a given situation, allowing a certain measure of safety to the system. Safety is especially important during unsupervised learning of feedback maps in the real world.

Experiments on the robot Thing demonstrate that this approach works for static walking and learning to turn. However, their implementation of the subject to constraint results in slow convergence and therefore slow walking. Furthermore, this particular control basis uses global resources and can not easily be distributed. On the other hand, the use of a control basis combined with the hybrid DEDS approach results in a very small set of parameters and reference trajectories that have to be tuned.

It is difficult to determine controllability of the abstract system model, e.g. if a specific task can be solved. One reason for this is the unknown environment, another is that the model depends on the chosen set of continuous controllers; it is not certain that the set is capable of achieving a specific task.

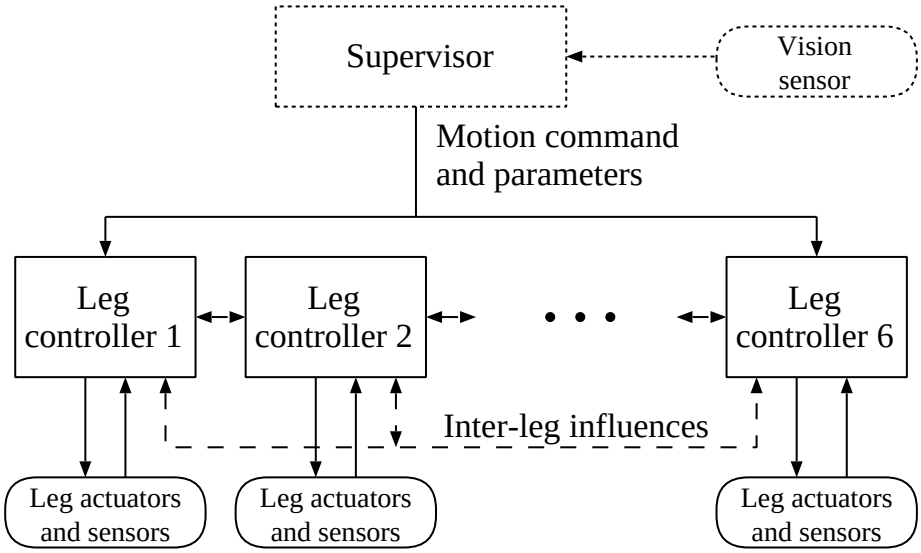


Figure 3.22: Overview of the Walknet control structure.

3.4. A biologically inspired controller

This section will describe the *Walknet* controller (figure 3.22). It is a very decentralized and modular controller that was designed at the University of Bielefeld (Germany) based on biological experiments by Cruse et al. [31]. They never designed an actual robot, but their simulation results and control principles have been used in collaboration with other groups such as at the Technische Universitet (TU) Munich [141, 142, 199] and at the Duisburg University [42].

First the simulation model will be briefly described, followed by the hexapod from TU Munich. Then the Walknet controller will be described and finally a summary and discussion will be given.

The simulation model

A simulated hexapod model, based on the stick insect (figure 3.23), was used for the design of Walknet. The legs are insect configured with three degrees of freedom per leg labeled, α, β, γ as illustrated in figure 3.23. The outputs of this controller are joint velocities, that are integrated in the simulation environment.

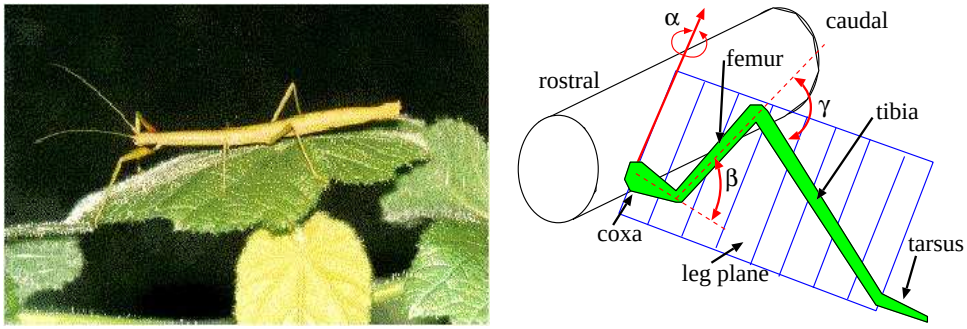


Figure 3.23: Image of a stick stick insect *(Carausius Morosus)*, about 80 mm long and 5 mm thick [14] and sketch of leg kinematics.

The TUM Hexapod

Figure 3.24 illustrates the TUM Hexapod that weighs about 23 kg and can carry about 5 kg. It uses decentralized leg controllers and some of the local coordinating mechanisms that will be described in the next section. The leg controllers are very similar in principle to Walknet, including a retract-and-elevate reflex. However, this system is based on state machines, not ANNs. Furthermore, Walknet relies on two phases for each leg cycle, whereas the TUM Hexapod uses four.

3.4.1. The Walknet controller

Behavioural, electrophysiological and neuroanatomical investigations led Cruse et al. [31] to the control structure illustrated in figure 3.22. The studies were done on the motor systems of stick insect's and these showed that the leg controllers are very independent and not directly controlled from a higher level. Instead, the leg controllers are coordinated through (six) influences (figure 3.25). Schmitz et al. [164] provide more information about the overall control structure and Cruse et al. [32] discuss the special use of positive feedback in the leg controllers, as explained later.

The dotted box in figure 3.22 illustrates a higher level that is not necessarily consistent with biological findings. It is supposed to deliver estimates of velocity and heading (assuming some kind of vision system) as well as the corresponding commands. However, all the remaining functionality lies within the individual leg controllers and the inter-leg coordination mechanisms. Each leg performs its own stepping motion and this, together with the coordination mechanisms, cause the

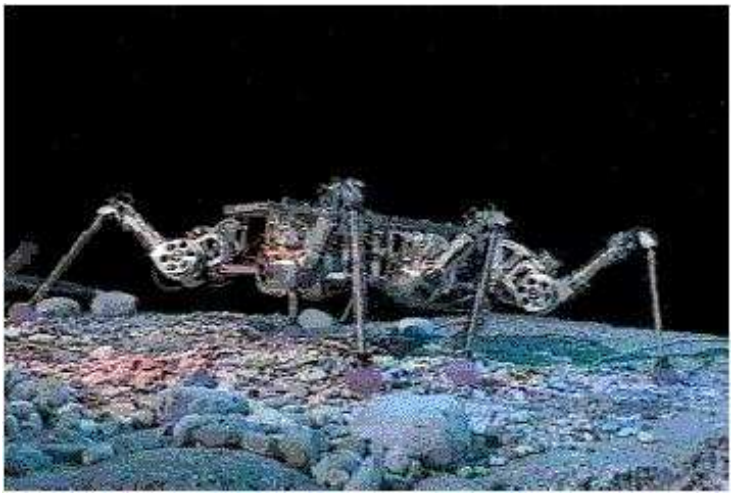


Figure 3.24: TUM hexapod, dimensions about $0.8 \times 1.0 \times 0.4$ metre (L $\times$ W $\times$ H) [14].

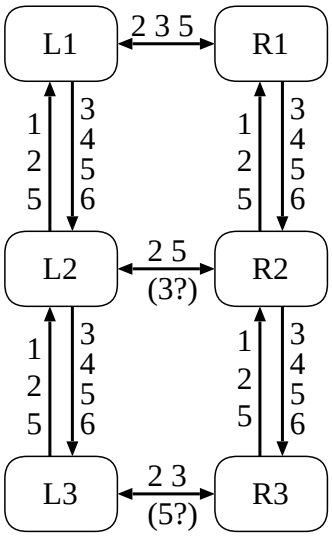
walking to emerge.

Trunk control The trunk height is controlled implicitly since each leg controller tries to maintain a specific height. This will also affect the attitude depending on the surface. Yawing and longitudinal velocity is controlled indirectly in each leg controller, through the desired velocities (yaw- and longitudinal) from the supervisor. In addition to desired velocity and yaw-rate, the supervisor also sends a "Walking on" signal that activates the legs.

Inter-leg coordination

The influence of inter-leg coordination mechanisms (figure 3.25) is stronger between ipsilateral legs, than between contralateral legs. No direct influences between the diagonal legs have been found. The influences are as follows [31]:

1. The start of a transfer phase is inhibited if the ipsilateral posterior leg is transferring (and up to 100 ms after footfall). This can cause a prolonged support phase.
2. The start of a transfer phase is excited if the ipsilateral posterior leg or the contralateral leg just entered the support phase.
   This can cause a shortened support phase.



1. Transfer phase inhibits start of transfer phase
2. Start of support phase excites start of transfer phase
3. Caudal positions excite start of transfer phase
4. Position influences position at end of return (“targeting”)

5a. Increased resistance increases force (“coactivation”)

5b. Increase load prolongs support phase

6. Treading-on-tarsus reflex

Figure 3.25: Inter-leg coordination mechanisms. L1 is the front left leg, L2 the middle left leg and so on.

3. The start of a transfer phase is more strongly excited, the further the leg is to the rear of a supporting ipsilateral anterior leg or contralateral leg. This causes the leg to start its transfer phase before the anterior leg.
4. The start of a support phase is “targeted” to occur next to the (supporting) ipsilateral anterior leg. This causes a follow-the-leader type of gait to emerge. Note that this influence also exists between the stick insect’s antenna and the front feet.
5. a) The force of a supporting leg increases if an adjacent leg encounters increased resistance.
   b) The support phase is prolonged, if the load of an adjacent leg increases.
6. If a foot is placed on top of another foot, the placed foot is repositioned slightly to the rear (to avoid stumbling).

The mechanisms 1-3 all produce the same effect: The nearly immediate re-establishment of coordination in the case of disturbances, i.e. the mechanisms are partially redundant. (Only) mechanisms 1-4 and 5b have been implemented in simulation.

Leg control

Figure 3.26 illustrates the leg controller that is based on ANNs. The three major components are as follows:

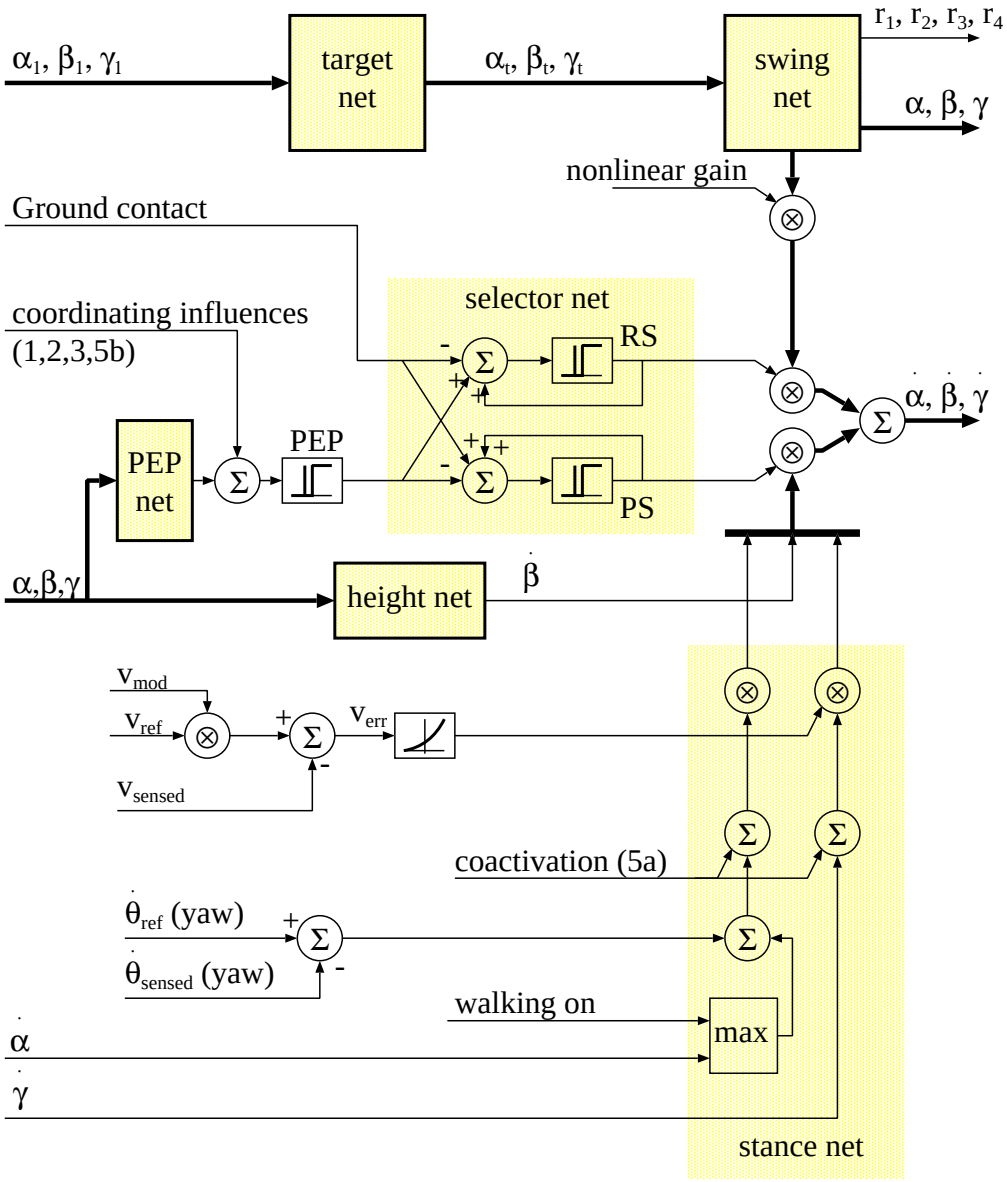


Figure 3.26: Walknet leg controller. The three major components are the stance, swing and selector net. Outputs from the stance net and the swing net are joint velocities. The selector net acts as a switch that sends either the output from the swing net or the output from the stance to the (simulated) leg joints.

- A *stance net* that controls the leg during the support phase.
- A *swing net* that controls the leg during the transfer phase.
- A *selector net* that switches between using the outputs of either the stance net or the swing net.

In essence, this is a simple switching controller with different modes depending on phase.

Selector net The selector net decides which set of outputs (joint velocities) that will be used.

- The *ground contact* (GC) input excites the support output unit (PS – power stroke), at the same time as it inhibits the transfer output unit (RS – return stroke).
- The posterior extreme position (PEP) input indicates that it is time to transition to the transfer phase. It excites the transfer output unit and inhibits the support output unit. The input comes from the *PEP-net*, that provides a value proportional between the actual foot position and the threshold position. It is influenced by the coordinating mechanisms (1-3) from the adjacent legs.

The output units receive self-excitation (i.e. positive feedback) to "lock" into a state. This was found to be less sensitive than using mutual inhibition from the output units.

Swing net The swing net controls the transfer phase of the leg and Cruse et al. [30] showed that to reproduce the stick insect's transfer motions it was enough with the following:

- A simple linear two-layer feed-forward net[21], with only 8 or 9 non-zero weights.
- Three target position inputs, either from the inter-leg mechanisms or a fixed configuration. These are in turn generated from the *target net*.
  - There is no explicit calculation of either leg's foot position in the target net, but the target angles will cause the transferring leg to be placed next to the anterior leg's foothold.
- A nonlinear compensation as a function of the distance to the target position, to modify the velocities so that they resemble that of the stick insect

[21] This net corresponds to a linear state feedback controller.

The swing net is able to generalize over a range of untrained situations. Extra inputs are also used to indicate mechanical disturbances, that cause a short retraction and elevation reflex. However, the attraction of the target position will eventually cause the foot to move forwards.

Stance net The stance net controls the leg during the support phase. The controller solves the problem of (kinematically) coordinating several joints (not just for this leg, but for all legs) by using high-pass filtered positive feedback [32]. The high-pass filter results in a velocity feedback, that causes velocity to saturate instead of going to infinity. However this approach is not used for the β joint, since gravity would otherwise collapse the robot. Instead, negative feedback is used on the β-joint to servo to a desired leg height (the current leg height is estimated by the *height net*).

The commanded yaw rate, $\dot{\theta}_{ref}$ (yaw), and velocity, v_{ref}, from the supervisor are also handled in this net. Note that it is not necessary to do anything special with these commands before they are used in the leg controllers.

- The yaw motion is controlled with a negative feedback correction that is added to the α-joint feedback signal (the sign is modified, depending on the side of the trunk).
- The velocity is controlled using negative feedback, where the error signal, v_{err}, is sent through the function

$$y(e) = \begin{cases} 1+e & e \geq 0 \\ \frac{1}{1-e} & e < 0 \end{cases}$$

  and then multiplied with the channels providing feedback for α and γ [164].

Because of the use of positive feedback, special care is taken to avoid going backwards. This is done by replacing the sensed α-joint velocity with a small value if it is too low (or negative).

3.4.2. Summary and discussion

The motion is kinematically driven in Walknet, but emergent from the individual leg controllers, since each leg constitutes a nonlinear oscillator that repeats a stepping motion. A special feature in this controller is the use of low-pass filtered positive feedback, to achieve coordination between legs and joints. This should also reduce the problems of large constraint forces, but that has not been studied. However, it is necessary with the negative feedback in the β-joint to resist gravity, as well as a

threshold on the α-joint to prevent the robot from going backwards. What would happen if the robot was supposed to walk on an inclined plane?

Balance is not considered explicitly, in fact the simulated robot sometimes falls and manages to rise without any knowledge that it has actually fallen. The support pattern emerges into a tripod or tetrapod gait depending on the locomotion velocity, through the combination of inter-leg influences and leg controllers.

Performance

This controller has really only been tested on a simulated robot[22], but the inter-leg influences have been used successfully by several other robots.

[22] See http://www.uni-bielefeld.de/biologie/Kybernetik/research/walk.html for some nice animations.

4. Analysis

The questions that we have tried to answer with this study can basically be phrased as follows

> How does the control of existing legged machines work and what, if any, are the common principles?

We have attempted to look beyond the implementation and hardware aspects of these machines as much as possible, with the intent of solely studying their actual control. Furthermore, we have tried to focus on how locomotion is generated, not on how the controllers have been designed.

To try and answer these questions, we have performed a literature study, where we have studied several legged machine controllers briefly, and a few in detail. The selection of the machines studied in detail, has been done to "spread" the views and get a good overview of at least seemingly very different controllers. There is of course no guarantee that we have covered the entire spectrum of controllers for legged machines. Examples of very well known legged machines that have not been covered in detail are Raibert's hopping robots. There is also relatively little coverage of bipeds[1].

A drawback with doing a pure literature study is the difficulty in obtaining all the necessary information in order to do detailed comparisons. It might be argued, and with good reasons, that this kind of analysis requires detailed simulations and experiments to produce valid conclusions and results. Unfortunately, this must remain a candidate for future work as it has, for practical reasons, been well beyond the scope of this work.

The problem of controlling legged locomotion is discussed next (section 4.1), followed by the results from the literature study (section 4.2) and finally a summary and discussion (section 4.3).

[1] The survey by Eriksson [39] covers a lot of bipeds.

4.1. Problem analysis

Studying how the control of a legged machine works is a complex problem with lots of different aspects. Here we will first discuss it (and narrow it down) in terms of goals, then in terms of compliance and damping, followed by coordination and slipping. Finally, the aspects for analyzing the problem and comparing control systems are decided upon.

4.1.1. Goals

In order to look at a control system and evaluate its performance, one must take the goal of the system into consideration. For legged machines, the goal is not always obvious or unique. Instead, the goals are often conflicting such as:

> Move from one place to another as fast as possible in the safest manner possible.

If we look at TITAN III (section 3.1.1) and the ASV (section 3.2.2), their goals can be thought to be that the trunk should follow a given path or trajectory. If the goal of the machine is to move approximately from one location to another, generating a detailed path might be a waste of time or unnecessarily restricting. Another question is whether it is important to track a desired trajectory really well, or if it is just the end result that matters. One argument for having a good tracking performance is of course that there might be obstacles that have to be avoided. A simple way to specify goals could be as follows:

- The goal is to travel from one location to another. This could also be further divided:
  - Little or no importance is placed on tracking performance.
  - High precision tracking is required for avoiding obstacles or machining operations (drilling, milling etc).
- The goal is to explore; i.e. wander around and cover an area as large as possible.

It is of course not necessary to specify a desired trajectory solely in spatial terms. The concept of impedance control might very well be applied to a desired trunk trajectory, thereby also specifying the dynamic behaviour of the trunk when subjected to disturbances (a human giving the machine a push).

Navigation and planning Is navigation a part of the legged locomotion controller? This could be considered the case if a map of the terrain is known, allowing detailed planning of footholds etc. Is this planning then part of the controller?

- To solve a task like jumping over an obstacle, it would seem prudent to plan ahead.
- For other tasks or gaits (i.e. at high speeds) it also makes sense to plan ahead and be confident that the task is achievable before attempting it.
- However, for other "gaits" (e.g. statically balanced) it is possible to take it step by step and back up if necessary, thus reducing the need for planning.

In our analysis, we have assumed that no more than short term planning is included as part of the legged machines controller.

Additional goals There are also goals that are more general for autonomous robots:

- Safety
- Performance
- Fault tolerance
- Mission completed

We have found very little information about this in the references and it seems to be a field that needs a great deal of work. It has therefore not been included in this study.

4.1.2. Compliance and damping

In reality, there is always some compliance[2] and damping in the combined system of robot, environment and controller. A few sources of damping and compliance are as follows:

- Soft surfaces (soft soil, mud, sand etc)
- Flexible links or rubber pads
- Compliance in the control ("weak" position controllers, impedance controllers)

[2] Gao and Song [45] point out that the stiffness of the system "distributes" the force even in the "statically indeterminate" cases.

Some examples of possible disadvantages with compliance and damping in the system are as follows:

- Weak links can be more difficult to control.
- Reduced bandwidth.
- Reduced energy efficiency due to damping.
- Resonance phenomena (i.e. chattering).

On the other hand, there are also advantages:

- Compliance accommodates position errors (in position-based controllers).
- Compliance stabilizes force control during intermittent contacts with hard surfaces.
- Compliance smoothes out impacts and reduces shocks (during footfall).

We believe the latter point is important for stability (especially for position controlled systems), even though it is rarely mentioned.

The question of advantages and drawbacks with compliance is actually very complex. As far as the analysis is concerned, the question is if the compliance and damping can be ignored as a major reason for why/how the robot walks or runs? When it comes to hopping robots, it is very obvious that it cannot be ignored. Raibert's hopping robots and others [2] exploit the compliance to reduce energy expenditure. In a case like TITAN III, we assume that damping is only necessary to damp out bouncing and accommodate position errors. TITAN VI for instance uses large damping soles to facilitate better performance on irregular ground.

If a combined stiffness/damping controller or an impedance controller is used, the question of the importance of the compliance is partially transformed into parameter dependency. In many cases it should be possible to either ignore the robot and the environment's compliance or include it with the controller's compliance.

In essence, we have tried to ignore this aspect due to lack of data, but it should require more study in the future. We believe this is an important aspect regarding overall system efficiency and stability, not merely about avoiding footfall chattering.

4.1.3. Coordination and slipping

A basic aspect of walking is that we are mostly interested in controlling the motion of the trunk. However, the machines have no actuators to control the trunk directly and therefore have to rely on gravity and limb motions. On the other hand, with more

than one supporting limb, closed kinematic chains arise when the legs are assumed to be connected to stiff surfaces by universal joints. To handle the new constraints, we address the problem of *coordination*. The term is in general [122] defined as

> the harmonious functioning of parts for effective results.

The walking community uses the term in different meanings:

- Coordination of joint motions within a limb: intra-limb coordination.
- Coordination of motions of all limbs: inter-limb coordination.
- Coordination of leg motions to achieve good (smooth) footfalls without chattering and large forces/impulses.
- Coordination of trunk motions with respect to the limbs or the environment (follow a path, avoid obstacles, constant velocity etc).
- Coordination of ground forces; avoiding too large vertical forces that cause sinking, or too small that cause slippage.

One reason to handle coordination is to avoid large constraint forces within the kinematic chains. These could cause slippage (or worse, mechanical failures). As discussed in the previous section, compliance is good in this sense. As to the problem of slipping, we do not always believe this is necessarily critical. In fact, we have seen dogs run (chasing a Frisbee), slipping constantly, and still achieve their mission (catching the Frisbee). On the other hand, when dogs do slip and fall they rarely have any problems standing up again... There are however other reasons why slipping could be bad.

- Bad energy efficiency.
- The foot may leave its working range too soon (sooner than planned).
- When using pure position control, this could cause unwanted yaw motions.
- Dead-reckoning becomes very difficult.
- Unexpected and jerky motions could occur, thus disturbing maching operations or causing mechanical stress.
- Cause damage to the terrain.

However, we do not consider this explicitly further, but instead focus on the aspects described in the next section.

4.1.4. Reducing the scope of the problem further

In addition to the problem of insufficient data in the references, there is also the question of what data is relevant. Where should the line be drawn between the control system, the robot and the environment? Consider the robot TITAN III (section 3.1.1), that uses a sampled control system at the lowest layer. This system implements (among other things) a P-controller that generates DC motor voltage based on a position error. What approximations can be made?

- Can the dynamics of the DC motor be ignored?
- Can the dynamics of using a sampled control system be ignored?
- Can the DC motor be considered a perfect velocity servo?
- Can the P-controller and actuator be considered a perfect position servo?

To answer these questions or to at least (approximately) motivate assumptions, information is needed about controller bandwidths and response times. This type of information is rarely available in the published papers and might not even ever have been measured. In practice, we have therefore assumed that when a joint is reported to be position controlled, this is achieved sufficiently[3] well. Furthermore, to achieve a higher level of abstraction, we have "defined" ideal inputs and outputs for a central part of each legged machine's controller. In the case of TITAN III, we decided that the ideal output of the controller was the position of a foot. Here we have assumed that the subsystem (consisting of sampled P-controller, DC motor and gears) achieves a perfect position servo. Note that we have also included the inverse kinematic problem of transforming foot position to joint angles in the position servo.

In short, the framework and limits of what to study was chosen as follows:

- If short-term planning of trunk motion exists, it is the highest[4] level of the controller.
- The output of commands to the actuators takes place in the lowest level of the controller.

Typical examples of ideal inputs are, desired path, desired velocity or goal point. On the lower level, we have tried to abstract "upwards" as in the TITAN III example above.

[3] I.e., we assume that there are no "side effects" that are important as to why or how the system walks.
[4] Long term planning is beyond the scope of this survey.

Assumptions, reduced scope and focus of the problem

To further limit the scope of the problem, we have chosen to mostly ignore the part of the control that is responsible for leg control during the purely transferring phase. Only some information about generation of leg reference trajectories have been included in the detailed examples. However, we do not consider the control of transferring legs a trivial task for several reasons:

- The base of the leg (the hip) is in general moving.
- There are in general temporal and spatial requirements on the footfall.
- There might be (known or unknown) obstacles or other spatial constraints (workspace limits).
- The position and characteristics of the surface with respect to the leg is mostly unknown.

In order to "decouple" this problem, we have made two assumptions:

- The motion of a transferring leg does not affect the trunk's motion.
- The motion of the trunk does not affect the motion of a transferring leg.

If the legs are massless (a common assumption) or very light, the first assumption is probably valid, but this is not always the case. Consider TITAN IV, where it was reported that the motion of the transferring legs had a deteriorating influence on the trunk motion at high speeds. There are also controllers (besides passive walking) that include the limb dynamics[5].

The second assumption requires a very robust leg controller in order to be valid, but we believe it is a reasonable assumption when using for instance high-gain position control. On the other hand, we also believe that the dynamics of transferring legs could be important[6] and exploited. Unfortunately, we have not been able to include this in the analysis, that instead has been focused on:

- The control of foot placement (selection of footholds).
- The control of leg sequence.
- The control of the trunk motions through the supporting legs.

[5] One method to take the limb dynamics into account, is to first select footholds and then plan the trunk motion so as to maintain the ZMP within the support area.

[6] We believe limb dynamics are important for energy efficency, but perhaps also for stability.

Analysis and comparisons aspects

To bring some kind of order and structure into the analysis, we have in addition to limiting the problem, decided to focus on a few different aspects or subproblems (remember that our aim is to try and understand how the controller "really" works).

- What determines the machine's balance?
- What determines the machine's motion, as seen from the controller's perspective?
- What determines the machine's support sequence? (foothold selection and sequence) What causes leg phase transitions?
- What, if any, "reflexes" are used?

The above questions have been chosen with the purpose of finding a framework of questions suitable for most of the control structures[7]. There are of course other aspects of comparisons that we have not concentrated on:

- Is the control structure distributed or centralized? How can it be implemented?
- Is the control structure flat or hierarchical? Is the divide and conquer strategy used? How is it used?
- What principles are used for control design?
- What principles are used to represent information — explicit or implicit?

4.2. Analysis results

The results from this literature study are divided into two parts. The first part compares and analyses the examples described in sections 3.1-3.4 according to the four questions just described. In the second part, common principles and methods are discussed, but categorized in a different way, with the purpose of guiding the reader to further references. This part is based both on the examples and on the more brief studies of several other controllers and miscellaneous references.

4.2.1. Comparison of examples

This section compares the control of the different examples with respect to the following points: motion, balance, support sequence and reflexes as discussed in the previous section.

[7] I.e. based on the brief study of several controllers.

TITAN IV is not discussed explicitly in this section, because the controllers of TITAN VI and TITAN IV are structurally very similar. For the same reason, RALPHY, SAP and BIPMAN will not be included in this comparison, because their controllers use principles similar to that of the ASV. Furthermore, the version of TITAN VI's controller that we compare in this section include the sky-hook suspension algorithm.

The control of Thing that is used in the comparison assumes the manually designed supervisor. See the corresponding example sections for details and more information about how TITAN IV compare to TITAN VI etc.

Balance

The brief study of several walking controllers showed that the four "balance methods" described below were approximately uniformly distributed[8]. The question we will try to answer can be phrased as follows:

> How does the machine maintain balance?

The answer varies from machine to machine.

Not considered explicitly Walknet does not actively control balance, but relies on a wide base of support and "luck". More accurately, the balance is an emergent property of the inter-leg influences, leg controllers and physics of the system. Not explicitly controlling balance is a seemingly simple strategy to implement, but may[9] require careful parameter tuning or training. In the case of passive walking, the parameters that have to be tuned are both initial conditions and physical parameters (link lengths, masses etc). In general, the overall kinematic configuration (e.g. a wide base of support and a low centre of mass) is of course also important. There are actually several systems[10] that walk without any behaviours or subsystems explicitly responsible for balance.

Static balance Another common method is to use a static balance strategy. Thing does this by having a behaviour responsible for static balance active all the

[8] We suspect that the strategy of static balance is the most common method today. The selection of controllers we studied is probably not a good "random" selecton.

[9] We have found that it is sometimes (in simulation) very easy to find parameters that result in dynamically balanced walking.

[10] Mechanical toys without any control at all is one example.

time (in parallel to the other behaviours). Since Thing actually walks very slowly, this is probably a rather well suited method.

TITAN III explicitly plans motions to satisfy a static balance criteria, but the attitude reflexes that modify the reference positions for the feet probably play a large part in achieving a good balance.

ZMP balance To go to a more dynamically adapted version of the static balance strategy, people have used the ZMP to do planning of trajectories. This is the strategy used by TITAN IV and TITAN VI, although they use slightly different algorithms. A drawback is that the ZMP criterion is only really valid for walking on planes. Furthermore it is only a necessary criterion (and not sufficient) that the planned trajectory is realizable and "stable". Here the machines were able to walk, so balance was maintained, but this was not guaranteed by the ZMP criterion. The fact that there were four-legged phases during the gait probably helped maintaining balance, but it is difficult to say with any certainty.

Force distribution The sky-hook suspension used by TITAN VI is an algorithm that explicitly controls balance by its virtual spring/damper implementation. The output of the algorithm is a desired force on the trunk, that is distributed between the supporting legs. On TITAN VI, only the vertical force was distributed, so in that sense one can consider it a specialized version of the balance servo on the ASV, where the legs use force control in all three directions. The ASV algorithm can be described as generating a desired acceleration on the trunk from position and velocity errors, that is subsequently converted to a force and torque to be distributed among the supporting legs.

A drawback with force distribution methods is that they are computationally heavy.

Support sequence

What causes the support sequence? In what order are the legs used?

Emergent support sequences In Walknet the actual sequence in which the legs are used emerge from the inter-leg influences and the leg controllers. The gait emerges as a tetra- or tripod gait depending on the speed. The "rules" for leg coordination are in that sense very distributed. Footholds are selected using the follow-the-leader strategy.

In Thing, the leg sequence is also implicitly encoded (in the supervisor feedback map). The manually encoded map implements the crawl gait as a standard leg sequence, but if a foothold cannot be found, another leg is used. Footholds are selected individually as solutions to optimization processes of static balance, manipulability and obstacles such as edges.

The emergent solution is elegant in the sense that the gait emerges from either the individual leg controllers, or the combination and sequencing of several controllers. Another example of a robot with good performance on irregular terrain using leg behaviours is the hexapod Robot II [148]. A drawback with these methods is the difficulty in enforcing or guaranteeing specific footholds and leg sequences if necessary. As an example, consider the case when a map of allowed (or forbidden) footholds is known (walking in a minefield...).

Algorithmic support sequences Purely algorithmic methods to generate the support sequence are used by the TITAN's and the ASV. The ASV has algorithms for several gaits, the TITAN III essentially use the crawl gait and TITAN VI use the intermittent trot gait. These methods work, but require computational power and good information about the environment to do the planning effectively. This kind of planning problem can be really difficult.

Sometimes the planning problem is avoided and the leg sequence and the footholds are fixed, as in SAP. This is actually used in several machines (bipeds, quadrupeds...), but we believe it will not work well over a large range of speeds, nor on irregular ground. On the other hand, it does allow the system to walk without solving the planning problem and is therefore at least useful in the early stages.

Motion

> What determines the machine's motion, as seen from the controller's perspective?

The motion of the robots are generated in the different examples as follows below. We have chosen to separate the cause of motion into *kinematically driven* and *force driven*.

- With a kinematically driven controller, we mean that the trunk motion is considered (by the controller) to be caused by the (spatial) motion of the limbs.
- With a force driven controller, we mean that the trunk motion is considered (by the controller) to be caused by the forces applied to the trunk by the limbs.

The ideas here are very close to position controlled, versus force controlled. However, we have chosen not to use these terms to prevent us from being too restricted in our thinking. Otherwise, we might only think of force controlled legs and forget impedance control or some kind of hybrid leg control scheme. In fact, leg impedance control might be used both by a kinematically driven and a force driven controller. The important point here is how the controller generates the reference signals to the limbs.

Kinematically driven control Most legged machines today can be considered kinematically driven. Among the detailed examples we have: Walknet, Thing, TITAN III, TITAN VI.

TITAN III uses a method that now has been around for a long time: Foot reference positions are calculated and planned so as to produce the desired trunk motion by solving an inverse kinematic problem. A drawback with this method is that rather accurate information about the terrain is required. However, in practice, the method is made more robust by the use of low level reflexes.

TITAN VI also uses an inverse kinematics based approach, but here the trunk motion is modified so as to keep the ZMP within the support pattern.

Thing also uses inverse kinematics to go from trunk motion to the motion of the feet, but there is no planning, the trunk motion is taken as the gradient of a harmonic potential. The cause of the motion is actually a bit special, since one can think of the trunk as "pushing" a transferring leg ahead of itself due to the manipulability measure. Similarly, the trunk can be thought of as "pushed" ahead by the supporting legs, to achieve the best manipulability measure under constraints such as static balance.

All three machines (TITAN III, TITAN VI and Thing) use a centralized, top-down approach: The motion is specified for the trunk and used (through inverse kinematics) to jointly control the foot motions. Walknet on the other hand, uses separate, individual controllers for each leg. The high-pass filtered positive feedback achieves coordination among the legs and avoids buildup of internal forces (avoids slipping etc).

Another question about the motion is related to the goal of the machine. Is it important to follow a specific trajectory? If precise tracking of the trajectory is required, explicit methods like the one used by TITAN III or the ASV (described next) are probably better suited. In Thing, Walknet and TITAN VI, it is difficult to specify a precise trajectory that the trunk should follow.

Force driven control The trunk motion of the ASV is considered to be caused by the force that is applied to the trunk by the supporting legs. From a desired trunk motion, a desired force on the trunk is calculated that is subsequently distributed as reference forces to the supporting legs. The supporting legs are thus modeled as force generators on a higher level. Note that this controller is also responsible for the balance, i.e. balance and motion control are solved by one and the same controller. The main drawbacks with this method are that it is computationally heavy and requires leg controllers capable of delivering the desired force. Furthermore, it could be difficult to achieve the required force tracking bandwidth.

Reflexes

All of the methods above use different kinds of reflexes[11] to improve the performance. Sometimes they are even a necessary part of the controller during normal operation. All of the machines have some kind of behaviour that maintains ground contact. Different methods are used however. For instance, TITAN III uses a contact switch to modify the vertical component of a foot's reference position. The ASV on the other hand maintains ground contact through force control.

Another reflex that most of the machines use is to retract-and-elevate a leg when it strikes an obstacle. There is of course an inherent problem with this reflex, when it comes to judging what the difference is between an obstacle and the ground. In Walknet, they recently added the method of using the relative position between to adjacent legs to "decide" if the obstacle is to be avoided or used as a foothold. We do not know if the ASV really has a reflex for this behaviour, but it does plan the motion of the transferring legs so as to avoid the other legs.

TITAN III and Thing have reflexes for finding suitable footholds by avoiding footholds close to edges. Thing is also a bit special in general, since one could view the entire controller as a set of reflexes (low level controllers) that are activated in sequences.

TITAN III and the ASV have a more global reflex for emergencies that stops the machine. In TITAN III, if something is not as it should[12], the entire machine is simply stopped. The ASV uses a more advanced method. When an emergency occurs[13], the current motion plan merely continues to execute, since the planner always maintains a plan to halt the vehicle at the end of the current motion plan.

In general, reflexes are often used to maintain balance. For instance, TITAN III

[11] By leg reflex, we mean a relatively simple behaviour on the leg.

[12] A typical example is that a foot has not acheived the support phase in time.

[13] A typical emergency in case of the ASV is that the force distribution algorithm fails to find an acceptable solution.

could be said to use leg reflexes to control the attitude and one could also consider the balance servo in the ASV and TITAN VI as an advanced reflex.

Reflexes seem like a useful way to increase the robustness of the system. Consequently, it might be wise to study animals (and other controllers) to see if there are any additional low level behaviours, or reflexes, that could be useful for legged machines.

4.2.2. Common solutions

This section is divided according to classes of methods and contains a lot of references to help the reader find more detailed information. It also tries to give an indication as to what the trend is; in what direction the research is heading.

Generation of reference trajectories

Algorithmic methods for motion planning based on static balance criteria[14] have been used for a long time. Early examples are Hirose et al. [66, 67, 72].

Lee and Shii [104] describe how a graphical method can be used to determine gait parameters and Kumar and Waldron [97] describe a modified wave gait that can be used as an adaptive gait. For information about a free gait algorithm, see for instance Song and Chen [173]. Graph search is another method that is used by Pal et al. [136, 137] to plan free gaits for quadrupeds and hexapods .

Zhang and Song describe a turning gait for a quadruped [35] and the generation of a discontinuous gait is discussed by Gonzalez de Santos and Jimenez [34]. Similar methods for gait planning can be found in the references: [22, 87, 88, 135, 138, 204].

A modern method to plan free gaits (based on ordinal optimization) is described by Chen et al. [19]. Yang and Kim [204] also discuss a recent method for robust hexapod gait planning that allows a fault in one leg to occur.

It is interesting to note that none of the references above contain any biped examples. We assume the reason for this is that static balance is relatively rarely used by bipeds.

Bipeds often use model based planning by typically assuming a *virtual inverted pendulum* (VIP) [89] (quadrupeds too [38, 205]). Since a leg is more often like a double pendulum, local feedback laws can be used to make the robot behave more like a VIP [43].

[14] The static stability margin is related to static balance, but there are other stability indices such as posture stability measures (indicates resistance against tipping), see for instance the reference [182].

In contrast to the methods above, coupled oscillators have also been used to generate motion references. Golubitsky et al. [49] discuss how networks of coupled oscillators (models for central pattern generators (CPG)) can be used to generate the gaits (i.e. leg phases). The method includes "smooth" gait transitions for quadrupeds, but can also be generalized to bipeds, hexapods etc. There is of course also earlier work on more or less the same theme: Collins and Stewart [27], Collins and Richmond [26] and Zielinska [213]. Instead of using several oscillators to generate leg phases, Venkataraman [189] suggests using only one oscillator and variable time delays as a model for a CPG.

Reference trajectories can of course also be learned off-line, using genetic algorithms as by Kawaji et al. [91]. Here rhythmic reference trajectories are learned for their biped's hip and the transferring leg. Alternatively, Stitt and Zheng [177] have used ANNs to directly learn the joint trajectories for their biped SD-2. They used an internal model of the robot's dynamics, so that stability measures then could be used as feedback to the network during training. Ilg and Berns [83] have used reinforcement learning to train the ANNs controlling the hexapod LAURON for leg transfer, support and coordination. Berns et al. [13] have also reported the use of an adaptive backpropagation for learning on the same robot.

Similar to this method, Shih et al. [168] use pattern parameterized trunk motions to find suitable biped motions, by studying the trajectory of the ZMP. They use eight pattern parameters and inverse kinematics in the trajectory generation. During the generation, they vary the parameters to see how the trajectory of the ZMP changes. Later [171], they optimize the trajectory by changing the trunk motion, in order to keep the ZMP at the center of the support area.

This method is now very close to the one used in TITAN IV, TITAN VI (see the corresponding example section) and other machines, where the motion of the trunk is generated so as to the keep the ZMP within the support pattern.

A trend within finding reference motions seems to be going towards solving two-point boundary-value problems (TPBVP), where the task is to find feasible repeating trajectories for one step using optimization as discussed by for instance Virk et al. [192]. Roussel et al. [162] also work with TPBVP, for which they assume piece-wise constant control inputs and try to find energy optimal biped trajectories. This method relies on a good model of the entire system, since the idea is to solve a shooting-problem and each "shot" requires simulating one step.

A novel method to plan the motion is discussed by Goodwine and Burdick [50], without explicitly planning footholds. It is based on an extension of trajectory generation methods for smooth systems. The outputs are the joint trajectories (no intermediate calculation of spatial motions), but the results are as of yet limited to quasi-static walking. In a later work [51] they also define *nonlinear gait controlla-*

bility for a legged robot to mean if a specified gait allows movement in any direction.

Control methods

A lot of control methods are used. In addition to the standard ones such as PID and local linear state-based feedback [169], more advanced methods are also used.

Tzafestas et al. [185] do a detailed comparison of sliding mode and computed torque control on a 5-link biped. They found that the sliding mode control gave better results for all their test cases, especially when the parameter uncertainty increased.

Other examples of advanced control methods include gain scheduling and fuzzy logic as used by Shih et al. [170]. Osuka [134] use gain scheduling to control the Emu robot to do sitting up and down. The interesting point is that they give some stability results based on a Lyapunov theorem (assuming that the scheduling parameter varies slowly enough).

A very recent control method is closely connected to the use of ZMP to do planning. Sorao et al. [175] present experimental results from their biped controller, where the ZMP is controlled on-line. The algorithm first generates a ZMP trajectory based on an arbitrary trunk CM trajectory using fuzzy logic. This ZMP trajectory is then controlled directly on-line.

It is quite common to use different controllers for different phases. A problem with using switching control strategies is that it is difficult to guarantee stability, but Zefran and Burdick [211] discuss an approach to stabilize equilibrium points in this case. The stabilization of periodic orbits (as is the case for walking) is still an open question according to them.

A problem with the control of walking robots is that the feet may slip or leave the ground. Genot and Espiau [48] discuss a control synthesis method for computing torques that incorporates this aspect. A foot can be forced to leave the ground, or ensured that it remains (i.e. horizontal forces are kept low enough based on a friction constraint). The input to the method is an admissible motion for specific body points. They do point out that the algorithm does not in general yield physically realizable torques, as well as difficulties with finding admissible motions.

This leads us to the classes of methods for doing force distribution where the control explicitly considers the forces at the different feet.

Force distribution methods

According to Lin and Song [108], there are two types of force distribution algorithms: compliant and rigid. A compliant algorithm is used by Gao and Song [45],

where they calculate the force distribution that (always) occurs due to the natural stiffness of the machine, actuators and environment. They point out that the elasticity of the structure will always distribute the forces in statically indeterminate cases. Furthermore they point out that this only allows zero-resultant force fields to be added as force distributions.

Rigid force distribution algorithms do not consider the system's elasticity and is much more widely used. Vertical force distribution was used already in the beginning of 80's on the OSU Hexapod, as described by Klein et al. [93] and some other early work is described in the references [46, 47, 92, 96].

Force distribution algorithms are still actively researched in terms of making them faster and more efficient. Recent work includes using CMACs for hybrid force/position control (Lin and Song [109]), quadratic optimization (Marhefka and Orin [114]) and a Compact QP method (Chen et al. [21]).

Force distribution is difficult to implement, since it is computationally heavy and requires force control. A more theoretic problem is how to decide on which force distribution is the most appropriate at a given occasion. Most methods rely on solving an optimization problem, but that still leaves us with the problem of selecting cost function.

Higher level control

Higher level control and selection of strategies is really beyond the scope of this survey and only three will be mentioned. We have already briefly mentioned the Real Time Criteria and Constraints Adaption (RTCA) architecture (section 3.2.3). The higher level strategy is based on classifying the current situation to select different constraints for the force distribution algorithm.

Partially related to this (and based on a very similar basic control structure) a rule-based real-time reasoning system is used to select and adapt parameters online [143]. A typical example is to detect slipping and alter parameters and behaviours in order to deal with it.

Another example of a higher level approach uses a parallel network (BeNet) of Real Time Augmented Automata (RTTA) [132]. It is used to generate motion patterns from action generators that control the hybrid quadruped Ballboy.

Analysis and simulation methods

Detailed analytic analysis is difficult, if not impossible, for complex systems such as legged machines. Some results have however been derived, mostly based on return maps and focusing on hopping/bouncing machines (Vakakis et al. [187],

Ringrose [161], Berkemeier [10]). The results have showed that the systems are sometimes asymptotically stable under different conditions. As an example, Berkemeier [10] has been able to derive and compare some stability results for a simplified model of quadruped bounding versus pronking. His results indicate that the system can be passively stable when designed with the proper parameters. Note that this is similar to passive walking, but that local feedback controllers are used in Ringrose's and Berkemeier's work, so there is control at a local level. The important point is that they have showed that even without a global controller, the overall limit cycle is asymptotically stable. This was of course not a big surprise, considering passive walking.

Due to the complexity, very little is done with pure analytic analysis and simulation is thus required. Methods for doing simulation faster is also a subject that is being actively researched, by for instance McMillan and Orin [119]. Today real-time simulation is even possible, Lee et al. [103]. Wong and Orin have even suggested [203] using faster than real-time simulation as part of the controller when jumping over obstacles.

Intermittent control — hopping

The control of Raibert's hopping robots [149] has not really been discussed in this report. Other hopping robots are the CMU "Bow Leg Hopping Robot" [16, 212] and the McGill SCOUT-robots [17].

The common aspect in the control of these robots is the fact that they are controlled intermittently, in the sense that control is only applied once every cycle. This is typically done by selecting the angle of the leg(s) at footfall. In this way, they could be considered aiming to do control on the return-map level. This kind of control is also researched (in theory only so far) by Saranli, Koditschek, Schwind and others [163, 165].

4.3. Summary and discussion

This report is based on a literature study of legged machine control systems, where over 20 controllers have been studied briefly, and seven[15] in detail. The controllers studied in detail have been described, compared and analyzed in order to understand how they work. Additionally, common principles and methods found during the

[15] Or more, depending on whether you count two controllers with virtually identical structures as one or two controllers.

study have also been briefly described. However, early in the study we encountered a problem:

> How much of the walking is part of the controller and how much is a part of the physics (c.f. passive walking).

Since this is a literature study, we have tried to handle this by first "abstracting" the controller from the system and then trying to understand how it works. We have therefore had to assume that the major cause of walking lies within the controller, and not with physics. As a consequence, we have more or less had to exclude passively walking systems and hopping robots (because physics is so important in these systems). However, we find it interesting that even though dynamic effects have been more or less neglected, we have still succeeded in understanding many controllers. This might of course just reflect the relatively small part physics play in todays controllers.

The control structures that we have studied are not exactly the same as what is computer controlled. Instead, we have tried to abstract the system by defining ideal controller inputs and outputs. Here we encountered further problems, since different controllers rarely use the same kind of inputs. Furthermore, the details of the input (or goal) was rarely described in detail[16]. However, we have during this study concluded that it is important to have a well defined input or goal. Especially for the purpose of comparing different controllers and structures. Furthermore, we have come to the conclusion that some focusing question are necessary in order to effectively compare and understand the different controllers. The questions we have chosen dealt with *balance*, *generation of support sequence*, *generation of motion* and *reflexes*. In our opinion, these questions served their purpose well.

Balance Only two methods that were specifically aimed at maintaining balance were found: static balance and trunk force control. A lot of machines do not explicitly consider balance. Instead, they rely on parameter tuning, training or learning. However, reflexes are also commonly used to help maintain balance (typically the attitude/posture).

Reflexes The *retract-and-elevate* reflex (i.e. if the foot hits an obstacle in the transfer phase, it is moved backwards, upwards and then forwards again), is one example of a useful behaviour. We believe it is worthwhile to incorporate some of the commonly used reflexes in both our walking robot, WARP1 and walking machines

[16] Sometimes the system did not even have a well defined input or goal.

in general. Further studies of reflexes used by animals and legged machines would probably also prove worthwhile.

Support sequence The support sequence was often generated using algorithmic methods (some of which considers the balance), or emergent from individual behaviours.

Motion The cause of the trunk's motion (as seen from the controller) was not easy to classify. However, we believe a classification into kinematically driven and force driven systems is reasonable, if perhaps a bit too simple. With the term kinematically driven controller, we wish to emphasize the fact that the trunk's motion is caused by the kinematic motion of the legs. Whereas in the case of force driven motion, the controller explicitly considers the force that the legs apply onto the trunk.

4.3.1. Discussion

Although methods for calculating support sequences, trajectory generation, static balance and ZMP etc etc are widely studied in the literature, we have found very little research on stability of legged machines. In fact, even the concept of stability is very loosely defined and mostly based on the concept of asymptotic stability. However, that requires knowledge about a desired system state (or limit cycle) and it is rarely obvious what this state should be. We therefore believe the term stability needs to be defined. Our impression is that the term today is used to mean the balance of the system, i.e. preventing the machine from falling. Another way of saying that we need stability is saying that ensuring goals such as safety, performance, fault tolerance and mission completed needs a lot more research.

From a safety aspect, the emergency halt system of the ASV seems like a very good idea. Similarly, the idea of faster than real-time simulation as a sort of imagination, preventing the robot from doing stupid things also seems like a good idea.

Trends

One trend that we observed within this literature study is that force distribution is actively being studied again. Another trend is the use of formulating "two point boundary value problems" (TPBVP) and solving them with advanced methods to find reference trajectories and/or control signals. These methods are deliberative, but the use of behaviour-based control is also becoming more popular.

Biology keeps influencing the research and coupled oscillator methods are no longer limited to outputting position references, but today also joint stiffness parameters in an effort to imitate biological actuators (muscles).

Impedance control and other advanced control methods are also becoming more and more popular.

Benchmarks

One problem we encountered when trying to compare different controllers was that we could not find any results (experimental or simulated) where principally different controllers had been used on the same machine. It is not even easy to find results for robots solving the same task.

Or, conversely, cases where the same controller has been used on different machines. It now seems obvious that we need benchmarks for legged machines, allowing us to do cross-comparisons between machines and controllers. Not just controllers with different sets of parameters, but also structurally and principally different controllers. We therefore believe there is a need for benchmarks, that allows us to quantify aspects such as balance and robustness. It is of course not very easy to come up with good benchmarks, considering that legged machines vary not only in physical dimension and actuators, but also in kinematic configuration and the number of legs. A beginning would at least be to define benchmarks for a given machine and testing different controllers on that machine. In this case, a simple benchmark could be the time required to finish a 100 metre race, including starting and stopping. A more interesting benchmark from a balance point of view, would be doing the same race while a stochastic force disturbs the system.

4.3.2. Conclusions

To sum up our overall impressions, we believe that it is important to specify what the goal/aim/input and output of the control system is? From a design point of view, a control system should include mechanisms for (or answer how) balance is maintained, support sequence generated and motion generated. Furthermore, we believe that fast reflexes are important to achieve good performance, especially in rough terrain.

Finally, we believe there is a need for benchmarks in order to compare and evaluate different legged systems. At least it would be fun with Olympic Games for legged machines.

4. Analysis

Part II.

The walking robot platform WARP1

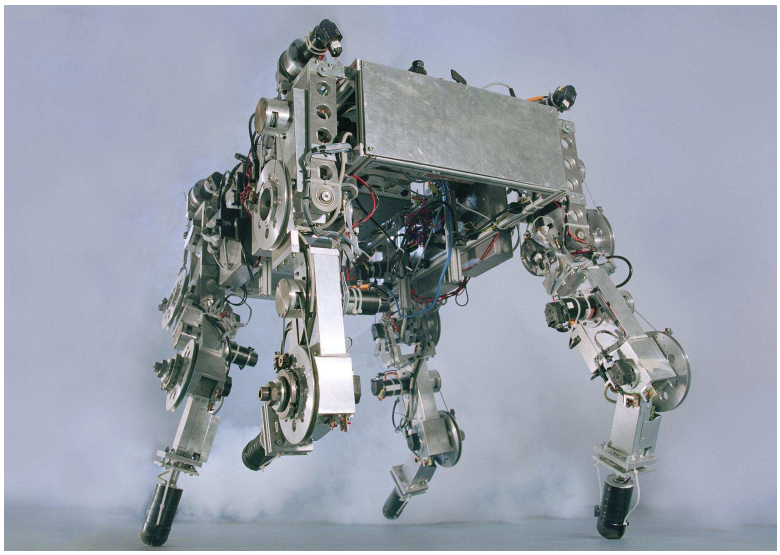


Figure 4.1: Photo of the walking robot WARP1.

Introduction to part II

This part will describe the four-legged walking robot platform WARP1 (figure 4.1). First, chapter 5 describes the platform in terms of its modular structure and then chapter 6 concludes this part by discussing the mathematical modeling of the robot. Chapter 5 is based primarily on the article by Ridderström et al. [155] with some updated material from Ingvast et al. [84]. Chapter 6 is new and unpublished.

Acknowledgements This work was supported by the Swedish Foundation for Strategic Research [176] through the Centre for Autonomous Systems [18] at the Royal Institute of Technology [100] in Stockholm.

Quite a large group of people have been involved in building and improving WARP1: Supervisors – Professor Jan Wikander and Doctor Tom Wadden; PhD students – Christian Ridderström, Freyr Hardarson, Lennart Pettersson, Johan Ingvast, Mats Gudmundsson and Henrik Rehbinder; Research engineers – Mikael Hellgren, Johan Wallström and Andreas Archenti; MSc students – Atsushi Ishi, Henrik Harju, Tomas Olofsson, Harald Karlsson and an entire advanced course in Machine Design and Mechtronics together with their teachers: Professor Sören Andersson, Kennet Jansson and Lars Wallentin.

5. The walking robot platform WARP1

This chapter describes a four-legged walking robot platform that has been developed within the Centre for Autonomous Systems [18]. It consists of a robot (section 5.1), computer architecture (section 5.2), control design tools (see chapter 7) and a basic control structure (section 5.3). Section 5.4 describes basic robot performance in terms of strength and speed.

Background and history The long-term objective is to provide an autonomous locomotion platform suitable for use in difficult terrain, where the autonomy implies that the platform should be self-contained in terms of its energy and computer requirements. The work started in 1996 and focused on the choice of locomotion platform. In 1997, two robot prototypes had been constructed: A prototype leg by Benjelloun [9, 120]; and a modular wheeled robot by Liander and Wallström [107]

Work then continued with legged systems, because of their potential for high mobility in very harsh terrain, as well as the challenging problems in creating such systems. A four-legged configuration was chosen, since this should allow both statically and dynamically balanced locomotion

The initial version[1] of the robot hardware was built by 39 undergraduate students during spring 1998, in an advanced course in Mechatronics and Machine Design [102, 197]. Work then continued by the researchers and WARP1 took its first steps in September 1999. However, the platform's mechanics and electronics were not robust and work on this kept on until next summer, when WARP1 was walking and trotting robustly (see `http://www.md.kth.se/~cas/movies`). Since then, the practical work has mainly focused on further software development.

[1] Initially the platform was called Sleipner3, but it was soon renamed to WARP1, since Sleipner is the name of Odin's eight-legged horse in the Norse mythology.

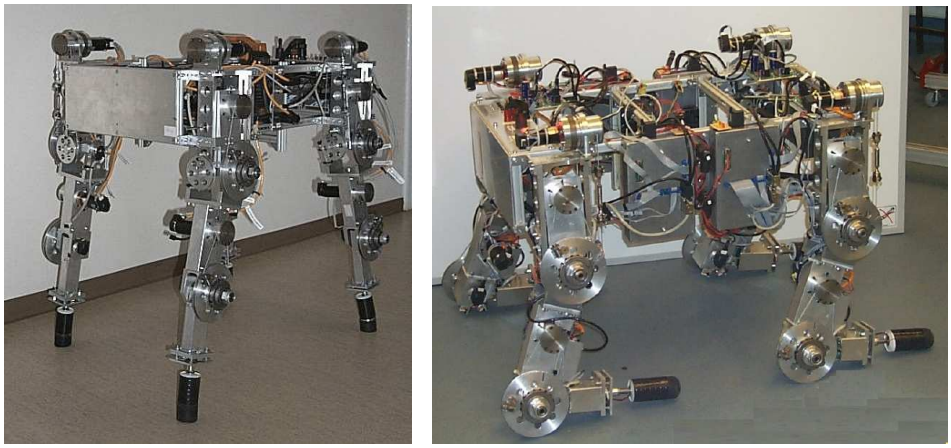


Figure 5.1: The robot WARP1 standing (left) and resting on its knees (right). In both photos the front of the robot is to the left. The left photo is from 1998 and the other from 2000, where one visible change is that aluminum boxes with computers have been mounted on the sides of the trunk (see page 123 for a more recent photo).

Research aim The initial research focus of the project has been on control methods for robust blind walking (walking without range finding sensors and/or terrain models) and mechatronic leg design (i.e. design of mechanics, actuators/transmissions and integration of control units). After blind locomotion has been achieved, range finding sensors and navigation/planning control might also be studied. Since different control methods will be investigated, it has been important to create a platform that makes testing and implementation of the methods easy.

Modularity The platform is modular in terms of:

- mechanics
- electronics
- computer systems
- control software.

Some of the modules are illustrated in figure 5.4, where the platform is divided into modules corresponding to external hardware, trunk and legs, and then each leg is further divided into three joints and a foot.

Table 5.1: Mass distribution, dimensions and power system parameters of WARP1

Part	Mass[kg]	Dimension [m]
Robot	59	$(\mathrm{l} \times \mathrm{w} \times \mathrm{h})$
– Trunk	20	$0.8 \times 0.4 \times 0.18$
– Leg	10	0.59
– Thigh	2.3	0.29
– Shank	2.1	0.30

System parameters		
System voltage	[V]	48
Maximum nominal motor power (approx.)	[kW]	1.6
Battery weight	[kg]	6
Battery capacity	[Ah]	7

5.1. Robot hardware

The robot has a cursorial leg configuration, where the kneecaps usually point forwards (figure 5.1). The modularity makes it easy to interchange the legs and achieve other configurations (such as all knees inwards or outwards). The mass distribution is about 20 kg for the trunk and 10 kg for each leg (table 5.1). The robot's approximate centre of mass is illustrated in figure 5.2, but it varies during walking due to the relatively large leg masses.

5.1.1. The trunk

The trunk is an aluminum frame built out of profiles with grooves that allow sliding nuts to fit. The sliding nuts make it easy to attach components and also to adjust their positions. Only the system voltage (48 Volt, table 5.1) is distributed to the different modules. It can come from an external power connection (up to 20 A), and/or an on-board NiMH battery pack. The battery pack can be used for shorter periods of self-powered operation corresponding to approximately 40 minutes of walking.

The legs are mounted with sliding nuts in the grooves of the trunk frame, making it possible to vary the leg attachment points in the longitudinal direction. Normally, the legs are placed as far apart as possible (figure 5.2), but the distance can be varied from 0.16 m to 0.65 m.

5.1.2. The leg

Figure 5.3 shows closeups of a leg and the hip joint. The legs have the same kinematical structure and each leg has three degrees of freedom: hip abduction/adduction (a/a), hip flexion/extension (f/e) and knee flexion/extension. The axes of revolution in the hip intersect and the knee can only overextend a few degrees (table 5.2). All the legs are identical, but mirrored in the sagittal plane. The distance from hip joint to knee joint is 0.29 m and 0.30 m from knee joint to foot pad.

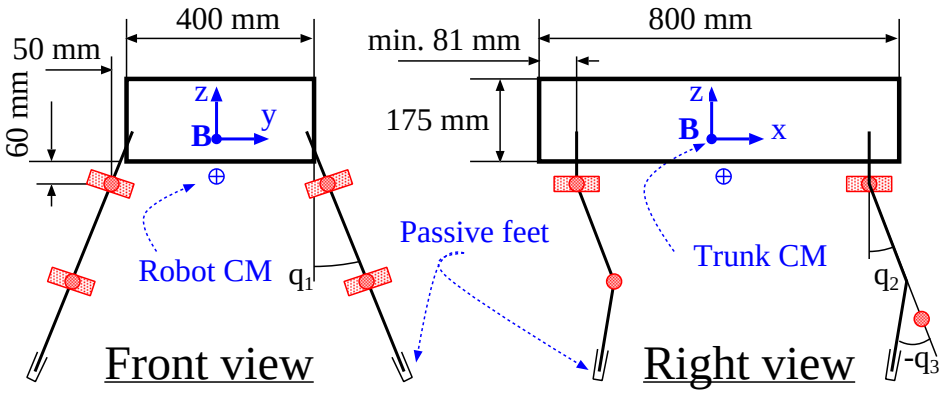


Figure 5.2: Kinematics and dimension of WARP1. The illustrated hip abduction/adduction angle, q_1 and hip flexion/extension angle q_2 are positive, whereas the illustrated knee flexion/extension angle, q_3, is negative.

5.1.3. The joint

The joint workspaces are limited by mechanical stops and tabulated in table 5.2 together with other data. Each joint is actuated by a Maxon DC motor via a Harmonic Drive connected to a cable and pulley system, where the lower pulley is attached to the limb (figure 5.3).

The cable reduction ratio is lower for the hip abduction/adduction joint than the flexion/extension joints. Furthermore, each flexion/extension joint can be fitted with a rubber torsion spring between the lower pulley and the limb. A spring increases the shock tolerance and the idea was to partly observe and control the joint torque through the spring deflection, similar to work by Pratt et al. [144]. The deflection is measured using two incremental encoders, one on the motor shaft and another on the joint shaft. Since the hip abduction/adduction joint cannot be fitted with an elastic element, it has only one encoder on the motor side. However, this method has not been used so far and the rubber springs were later also removed since they were considered too weak (it took about 0.25 seconds just to unload a fully loaded joint). Since the encoders are incremental, there is also a synchronization sensor (Hall-effect switch) that is used for calibration.

The pulley cables are steel wires (diameter 2.3 mm) that are tightened and fixed at the lower pulley. At the top pulley, forces are transmitted by friction (the cable is wound $2\frac{1}{2}$ revolutions) and by a small stopper soldered onto the wire. The stopper on the wire is fitted into a cavity in the pulley to prevent slippage.

Table 5.2: Data for hip abduction/adduction (a/a), hip flexion/extension (f/e) and knee.

Joint data		**Hip (a/a)**	**Hip (f/e)**	**Knee**
Workspace (0°= straight leg)	[°]	-30 – 30	-45 – 80	-110 – 5
Rated motor output	[W]	90	150	150
Motor current rise time	[ms]	0.7	0.4	0.4
Speed rise time at constant voltage	[ms]	20	10	10
Acceleration time to full speed at 5 A	[ms]	95	80	80
Harmonic drive reduction		100	100	100
Cable transmission reduction		2.5	2.85	2.85
Max. nominal angular velocity at 48 V	[rad/s]	4.8	2.6	2.6
Max. nominal torque at 5 A	[Nm]	50	91	91
Encoder resolution, joint shaft	[mrad]	NA	1.6	1.6
Encoder resolution, motor shaft	[mrad]	0.06	0.01	0.01
Joint stiffness (with rubber spring)	[Nm/rad]	NA	100	100
Joint stiffness (without rubber spring)	[Nm/rad]	1000	1000	1000
Motor and gear inertia (from joint side)	[kgm^2]	1.0	2.7	2.7

Speed rise time for a voltage step is about 15 ms and it takes about 90 ms to reach full speed (table 5.2) with the current limited to 5 A. Limiting the current is important, since the motors are strong enough to break the steel wires. Additional safety is provided by using joint range limit sensors (Hall-switch).

5.1.4. The foot

Figure 5.3 show's a photo of the foot. It's pad is a half sphere made out of rubber (diameter 50 mm), mounted on a linear bearing (range 0 mm to 26 mm) with a soft spring that extends the pad. A potentiometer measures the deflection and thereby provides both ground contact information as well as ground distance information just prior to an impending impact (the distance to the ground is measured along the direction of the shank). The foot is also equipped with a load cell that detects ground forces. Together, the two sensors more accurately detects ground contact.

5.2. Computer architecture

Figure 5.4 illustrates the computer architecture of the platform. Robot controllers can be developed and simulated on any computer with the appropriate software

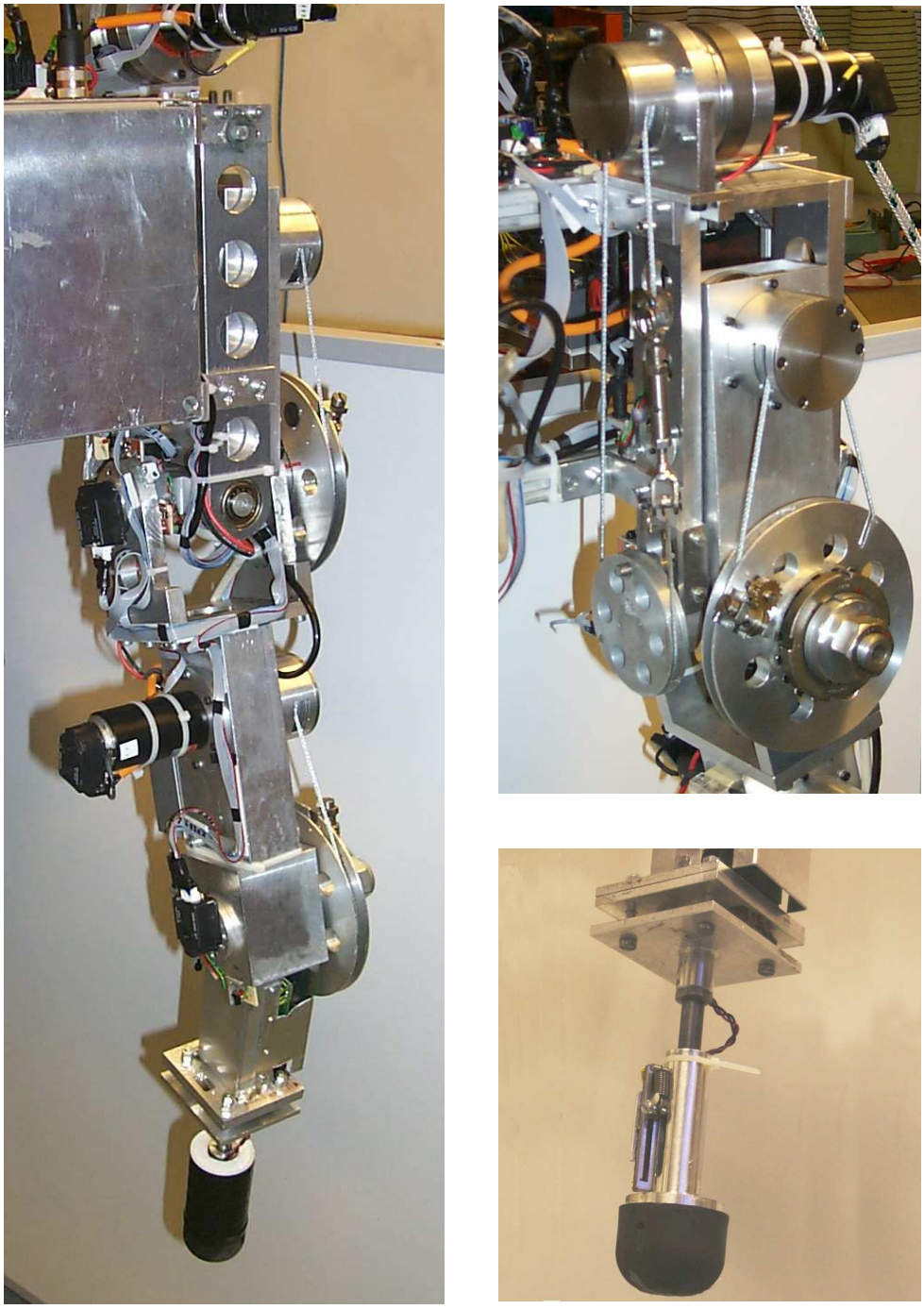


Figure 5.3: Photo of a leg and close-ups of hip joint and foot. The black plastic cover has been removed in the closeup of the foot.

Algorithm 1 Main loop in the ACN node's RUN state

1. Start AD conversion since that takes about $50\,u$s
2. Send ACN status message (every other cycle) and check for timeout
3. Sample encoder values and send encoder messages
4. Wait (up to a small time) for an actuator message and then set actuator outputs.
5. Send converted analog sensor values in current and foot sensor messages.

(chapter 7 describes the software tools). On the host computer, the controller can also be implemented for execution on the Development Control Node (DCN) to perform (real) experiments. During the experiments, the operator uses the host computer to control execution, modify parameters and log data etc. In addition to the interface through the host computer, the operator also steers the robot with a joystick and the Joystick Sensor Node (JSN). The JSN is based on a Siemens C167 microcontroller, whereas the DCN and the host computer both are standard PCs.

The system voltage and current is measured by the Trunk Sensor Node (TSN) that is mounted inside the trunk. The TSN is based on the same hardware and software as the JSN, but uses a specially designed IO-card to acquire data from

- one three-dimensional accelerometer
- one two-dimensional inclinometer and
- three rate gyros.

These data are then sent over a Controller Area Network bus (CAN bus) to the DCN.

The Actuator Control Nodes (ACN) functionally belong to the leg modules, but are mounted on the outside of the trunk (see figure 5.1). They are also similar to the JSN and TSN, but use another specially designed IO-card and PWM-driver card to acquire sensor data and control motor output. Communication with the DCN takes place over dedicated CAN busses, one for each ACN to provide enough bandwidth.

5.2.1. The ACN code and the CAN protocol

A simple state machine (figure 5.5) executes on the ACN, and similar state machines execute on the TSN and JSN. The nodes will automatically synchronize with the target computer's sample rate, since the RUN state waits (up to a small amount of time) for the message with actuator output values. Basically algorithm 1 is executed in the RUN state.

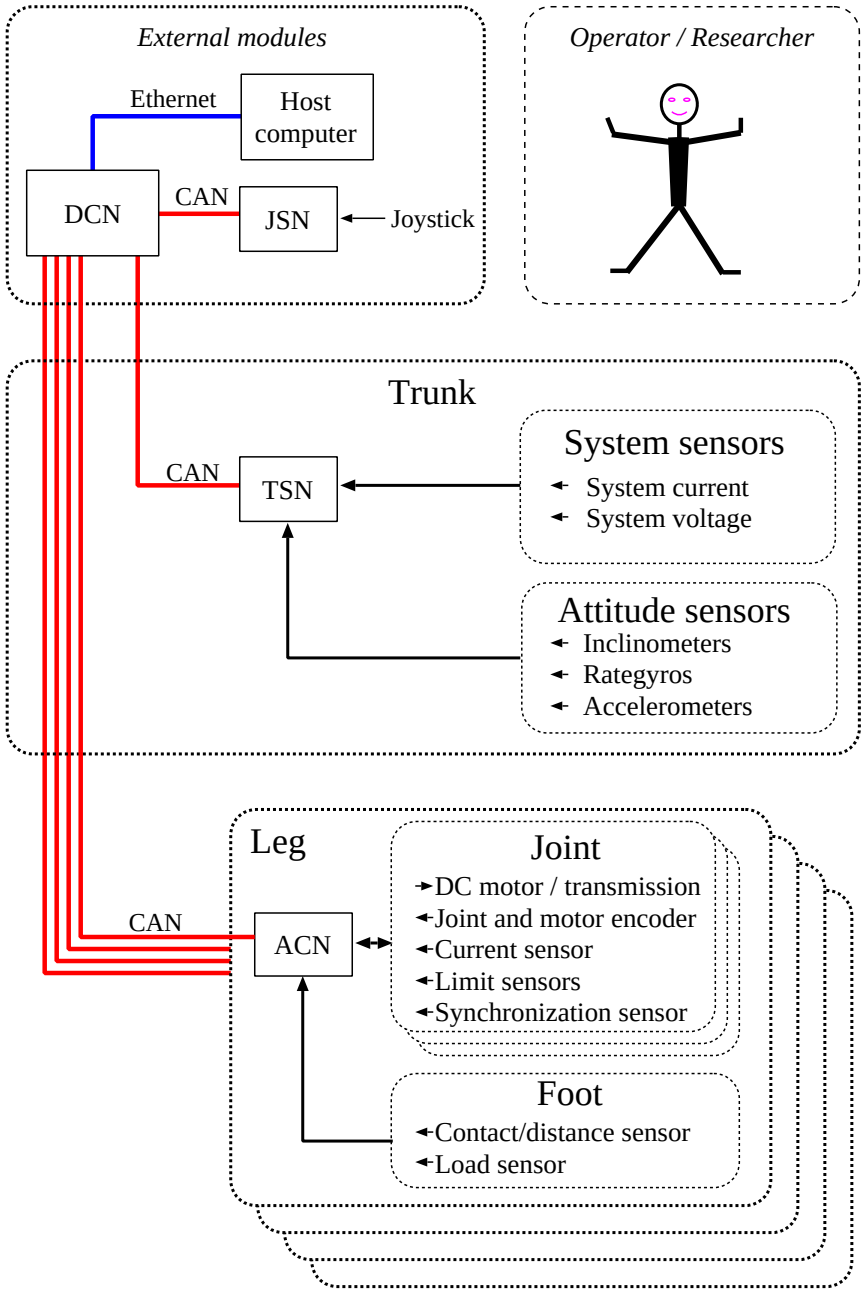


Figure 5.4: Platform modules and computer architecture where the DCN and host computer are standard PC.s. The JSN, TSN and ACN are microcontrollers based on a Siemens C167 processor. For details, see section 5.2.

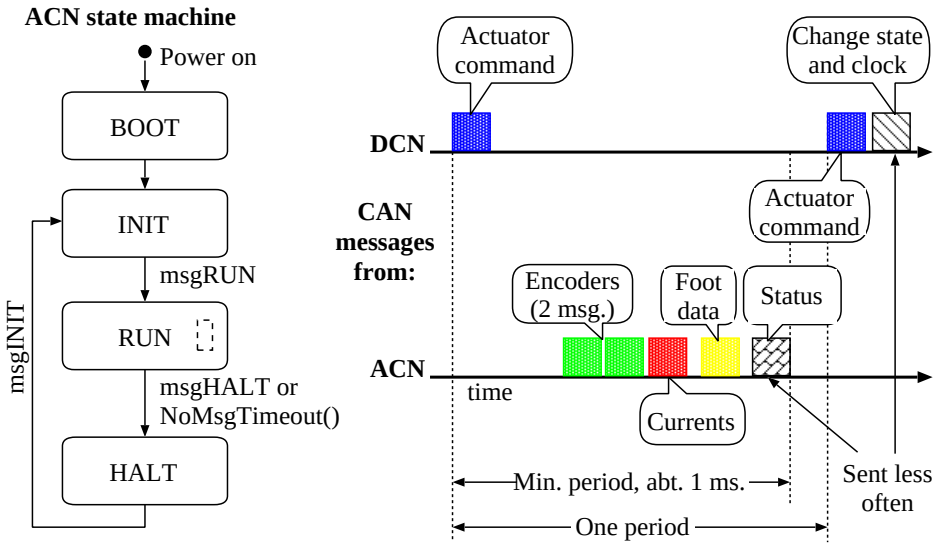


Figure 5.5: ACN state machine. and illustration of messages on CAN bus between DCN and an ACN. Up to seven CAN messages, with 384 bits of actual data (792 bits including overhead), might be transmitted during one sample.

In case no status messages are received within a few cycles, a timeout occurs and the state machine falls through to the HALT state. Otherwise the state machine remains in the RUN state until a Halt-message is received. Similarly, if the DCN fails to receive messages from any of the ACN:s, it stops and sends a Halt-message to the ACN:s.

The CAN bus protocols use a fixed message scheduling, illustrated for the ACN:s in figure 5.5. With this protocol (ACN) and hardware, the bus saturates at a sample rate of approximately 1000 Hz. In each cycle, up to seven messages are sent, corresponding to a rate of about 384 kbit/s of actual data. With the message overhead this comes to about 792 kbit/s, a little bit below the bus bit rate of 1000 kbit/s.

5.3. Control structure

Figure 5.7 illustrates the basic control structure with three levels: trunk, leg and joint level. When testing specific control ideas, parts of the structure are modified and this section will describe it's overall properties. The system is designed to allow a distributed implementation on several target computers. For instance, the system

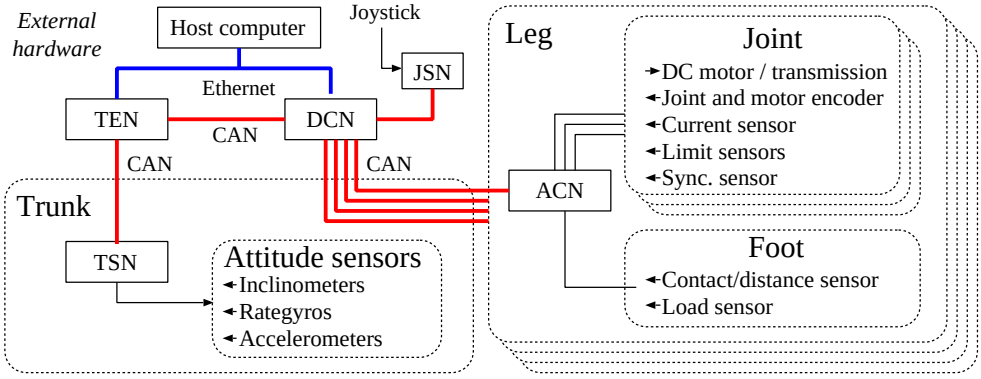


Figure 5.6: Illustration of computer architecture with two target computers, DCN and Trunk Estimation Node (TEN).

previously used two target computers as illustrated in figure 5.6. This allowed concurrent software development and testing against the ACN.s and the TSN.

Special care is taken to make sure that each major control module only uses signals which can be expected to be available in a distributed implementation. In the distributed situation only communication drivers have to be added to the interfaces of the control modules.

The trunk attitude is estimated by an algorithm designed by Rehbinder et al. [152, 154] that is executed in the *trunk observer*, using data from the TSN. Trunk motion and attitude is controlled by the *trunk controller*. However, except for Ridderström and Ingvast's [160] work on posture control, the trunk controller typically only generates parameterized position/velocity references for each *leg controller*.

The *leg observer* of leg l calculates the vector from the hip to the foot pad, r (and velocity, v) relative to the trunk frame. These estimates are then used (in a position control framework) by a simple combined stiffness/damping controller

$$\tau^l = J_l^T \cdot \{K^l \cdot \left(r^{ref} - r\right) + D^l \cdot \left(v^{ref} - v\right)\}, \quad J_l = \frac{^B\partial r}{\partial q_l} \tag{5.1}$$

that tracks the corresponding references r^{ref} and v^{ref}. In this control law, the Jacobian, J_l, is relative to the trunk with respect to the joint angles, q_l, and the matrices K^l and D^l denote the stiffness and damping coefficients. The output are three joint reference torques, τ^l, where the knee component is later modified by a virtual spring/damper to prevent the knee from overextending.

The torque reference is first limited in the *torque model* block (implemented as a

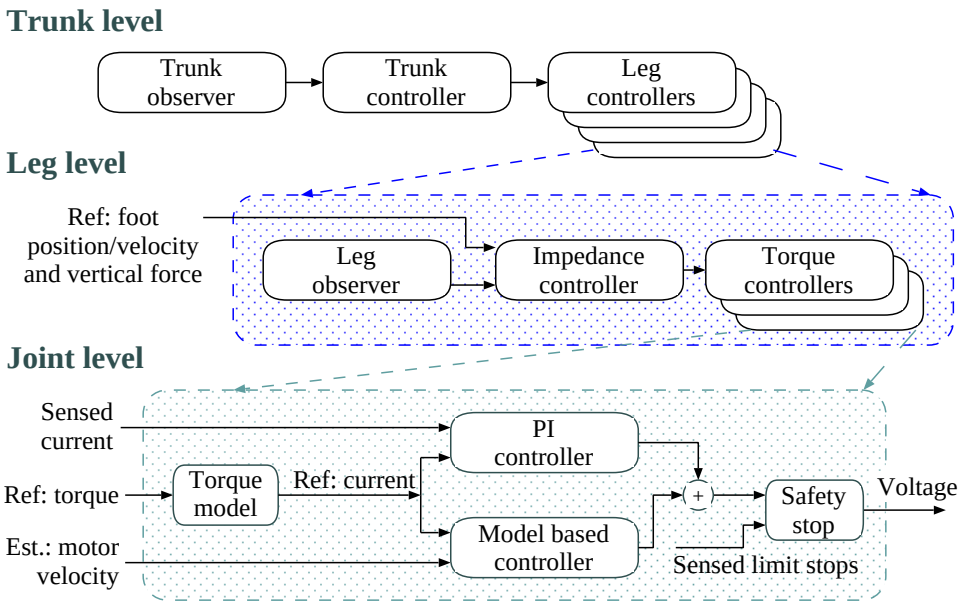


Figure 5.7: Overview of control structure.

current limit) that outputs a reference current. Then a feedforward DC motor model (based on inner resistance and back-EMF) is used together with a PI-controller to compute the desired voltage. Finally, if a limit sensor is active, the *safety stop* block limits the voltage in the direction of the corresponding mechanical stop.

5.4. Experiments

To demonstrate the speed of the legs, a simple experiment in which the robot performs walking motions was conducted. The robot was placed on a support with the legs in the air and two elliptic reference motions (periods 1.0 and 0.8 seconds) were tracked (figure 5.8).

When the period is 1.0 s, the leg tracks the reference (dashed line) well in both experiment (thick line) and simulation (thin line). The simulated and experimental trajectories have a small offset down to the rear due to gravity. The hip motor voltage (thin line) is close to being saturated and achieves a maximum joint speed of 2.4 rad/s, whereas the knee motor voltage (thick line) is below the saturation level. Just below the hip, the foot has a speed of 1.0 m/s.

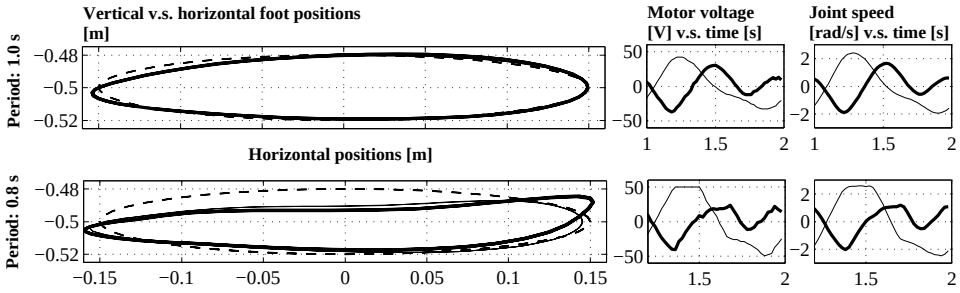


Figure 5.8: Walking motions (left), motor voltages (middle) and joint speeds (left). The motions are clockwise and start at "3 o'clock" and just before "3 o'clock".

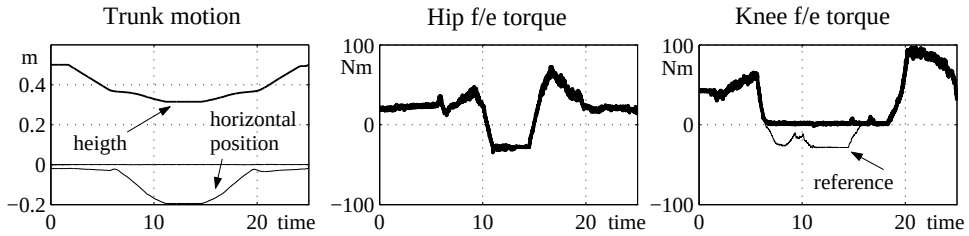


Figure 5.9: Trunk motion (left) and torques: hip f/e (middle) and knee f/e (right) during kneeling. The torques are estimated from the motor currents.

When the period is 0.8 s, the hip is unable to track the motions and saturates at a maximum joint speed of 2.5 rad/s. We can also note a small difference between the simulated result and the experimental result, probably due to poorly estimated parameters. The maximum speed of the foot is 1.25 m/s.

5.4.1. Resting on knees

To demonstrate the strength of the robot, a 50 kg payload was attached and leg reference trajectories were designed to make the robot go down on its knees and then up again (figure 5.9). The kneeling position is actually stable without any power to the actuators and could be used for "resting". Note that the knee flexion/extension reference torque (thin line) cannot be achieved because the joint limit sensor is active during the time interval $t \in [7, 17]$ s.

5.5. Discussion

The experiments have shown that the robot is strong and fast. It can go down and up on its knees with a 50 kg payload, as well as achieve joint speeds of 2.5 rad/s and foot speeds of 1.2 m/s.

Testing the robot has also shown limitations. For instance, the foot clearance is only about 0.19 m, due to a low working range in the knee. With the working range of a human knee (about 150°), clearance would be more than doubled. This will be taken into account in the design of the next leg.

The testing also taught us the importance of safety functions in order to avoid expensive mistakes. For instance, there are often very large position errors when the controller is started, and consequently a large control output. Examples of functions that increase safety are mechanical joint stops, limit sensors, current limiting and a dead-mans grip on the power supply to the motors. On the other hand, additional sensors etc increase the complexity and can introduce unexpected behaviours. For instance, the dead-mans grip is very convenient and consequently often used without stopping and then restarting the controller. This can be “dramatic” if the control error increased a lot while the motor was turned off. Even so, testing the robot and doing experiments have also shown that the control structure works well, and future control research will focus on the elements of this structure.

The tools combined with the computer architecture make it easy to move control blocks between the target computers. Similarly we expect it to be easy to expand the system with additional target computers, for vision and other long range sensors. The control frequency is limited by the CAN busses and not by the computational performance. The CAN bus to the ACN is filled at a control frequency of 1000 Hz (more than enough in practice), but the real bottleneck lies with the commercial CAN drivers on the DCN. With four legs, we have to execute the controllers on the DCN at 300-500 Hz (depending on their complexity). Fortunately, this bottleneck can be eliminated by rewriting the CAN drivers when we need to use more CAN messages or a higher control frequency, thus leaving us with a control frequency limit of a 1000 Hz.

6. Mathematical model of WARP1

This chapter primarily describes the derivation of the differential equations for a rigid body model of WARP1 using Kane's equations in a symbolic form. The rigid body model is too complex for derivation by hand, so the Sophia language by Lesser [106] was used (implemented for the computer algebra system Maple). However, it is the mathematical aspect of the derivation that will be described, not the actual Maple/Sophia-code used for the derivation.

The first section briefly discusses aspects of modeling walking robots, general assumptions and assumptions specific to WARP1. Then section 6.2 describes Kane's equations, followed by section 6.3 that describes the actual derivation of a rigid body model of WARP1. This section will also make the reader more familiar with the kinematics of WARP1 and the notation used to express kinematic relationships. The final section discusses the results and the author's experience of this work, as well as some notes about notation (section 6.4.2) and how to derive linearized equations of motion (section 6.4.3).

6.1. Models

Some kind of model of a legged robot is necessary in order to simulate the robot's behaviour, but it is also very useful for analysis and control design. A relatively complete simulation of a walking robot requires several kinds of models:

- Kinematic differential equations (kde) and dynamic differential equations (dde) of the robot's rigid body model (rbm).
- Actuator models
- Sensor models
- Model of control implementation (sampled control system) and communication system (delays)

- Models of the environment, such as
  - Model of the ground and foot/ground interaction, including ground geometry and characteristics
  - Disturbances (the robot pulls something or is pushed etc)

Simulation also requires choosing suitable initial values and/or a method to start the simulation (WARP1 usually starts simulation and real experiments in the same way — hanging in a rope and being lowered down until standing). For control design, on the other hand, it might be enough with just a partial kinematic description of the robot.

Note that there is (of course) a trade off between using detailed models and the speed with which the simulation runs. Therefore, models of sensors, control implemenation and communcation have in general only been included and used in special cases. On the other hand, the rigid body model and ground model are always used, whereas different types of actuator models have been switched between frequently. This chapter will focus on the mathematical aspects of the derivation of the rbm, whereas the other models are only briefly described.

6.1.1. Rigid body model

The following general assumptions are made about the rbm:

1. The main body, i.e. the trunk, has L articulated legs attached to it.
2. The force or torque applied at each joint is the output of an actuator model (or zero for an unactuated joint). Friction, backlash and elasticity in the power transmission are modeled in the actuator model. Similarly, the actuator model also includes models of the physical joint limits.
3. The interaction with the environment is accounted for by external forces (i.e. outputs of a ground model).

The last assumption implies that the rbm is modeled as an open mechanical loop regardless of the number of feet in contact with the ground. An alternative could have been to assume that a foot in contact with the ground creates a closed mechanical loop, with a different set of differential equations since the number of degrees of freedom has changed. However, the open loop alternative was chosen since it makes it easier to model slipping and a wide variety of ground characteristics.

Another modeling choice is whether the rbm differential equations should be in numeric or symbolic form. There are quite a few methods and tools that produce these equations numerically and some of those tools will be discussed in the next

chapter (chapter 7). Here the differential equations will be derived in symbolic form and although it might not be the best choice with respect to simulation performance, it keeps the door open for doing more advanced analysis.

The following assumption is more specific to the design of WARP1:

> Each leg is structurally identical and consists of two rigid bodies (a thigh and a shank) with rotational joints between the bodies.

This assumption implies that some mechanical details of WARP1 are ignored:

- A small rigid body in the mechanism of the hip joint is ignored, i.e. the joint is modeled as a two-dimensional joint rather than as two one-dimensional joints connected in series by a rigid body. Kinematically there is no difference since the joints' axes intersect.
- The foot is not modeled as a separate rigid body. Instead, the foot's mass and inertia is lumped with the shank, and the small motion of the foot's passive linear joint is ignored. (This passive joint is used as a sensor that measures the ground distance just before footfall).

The Maple/Sophia script that derives the rigid body model is specialized for WARP1. To speed up the derivation, it is assumed that the legs have identical kinematics, but not identical parameters such as link lengths and inertias. However, the script is not limited to WARP1 and can actually generate the rigid body model for a machine with L limbs, where each limb has 1–4 joints. This assumes limb kinematics similar to that of WARP1, but it is only necessary to change two or three rows, to use another sequence of rotational joints. Inserting linear joints is also easy.

6.1.2. Environment model

Terrain geometry is described using plane surfaces. We assume that the ground applies forces only on the supporting feet, no torques. This is motivated by a small footpad radius (about 2 cm).

The ground force is calculated as a function of penetration depth and velocity with some typical parameters illustrated in table 6.1. A linear spring/damper model is easy to use, but can result in discontinuous impact forces, due to damping and nonzero impact velocities. For the force perpendicular to the surface, we therefore use a modified version of a spring/damper model [130], when the foot penetrates the ground

$$F_z = -k_z z^n - d_z z \dot{z} \cdot (\text{Heaviside}(-\dot{z}))$$

Table 6.1: Two examples of ground model parameters

Type of ground	k_z $\left[\frac{\text{N}}{\text{m}}\right]$	d_z $\left[\frac{\text{Ns}}{\text{m}^2}\right]$	γ	d_h $\left[\frac{\text{Ns}}{\text{m}}\right]$	n
Weakly damped ground	70,000	2,000	0.7	2,000	1
Heavily damped ground	50,000	100,000	0.7	2,000	1

where k_z and d_z are the stiffness and damping coefficients, and z is the penetration depth. For the x and $y-$components, a smoothed viscous friction with a maximum limit based on the vertical force is used as in [15]

$$F_x = -\gamma F_z \frac{2}{\pi} \arctan\left(d_h \frac{\dot{x}}{\gamma F_z} \frac{\pi}{2}\right)$$

where γ is the friction coefficient and $\dot{x}$ is the velocity in the x-direction. F_y is calculated similarly.

6.1.3. Actuator model

Different actuator models are used (trading simulation speed against accuracy), ranging from ideal torque sources up to and including the DC motor's electrical and mechanical dynamics, viscous damping and linear spring/damping characteristics. Unmodeled (potential) problems include backlash, wire pulley dynamics and slippage as well as the motor's temperature dependence. Experience has shown that the motor parameters are quite temperature dependent.

The rotational acceleration of the link that the stator is attached to, is not modeled and the spring model is used to map spring deformation to output torque. This makes it is easy to add the actuator dynamics as extra states to the system, outside the rigid body model.

6.1.4. Sensor model

The robot has dual encoders to measure joint and rotor shaft angles as described in the previous chapter, but the discretization errors are ignored. Similarly, the sensors that measure motor current have not been modelled either. The inclinometers, rate gyros and accelerometers that are used to estimate orientation [153] have been modelled in simulation. Low-pass filters for the inclionmeters and offset plus noise for the rate gyros and the accelerometers, but this was mostly simulated during development of the attitude observer.

6.1.5. Control implementation model

In order to simulate a limited control frequency, we have added sampling delays in sensor sampling and actuator commands. Similarly, signals are delayed to reflect the fact that we do control over a CAN bus, but we do not model the corresponding jitter. However, the simulation runs substantially slower when including this model, so it is typically not used.

6.2. Kane's equations and Lesser's notation

The notation used in this thesis is primarily based on the interpretation of Kane's equations by Lesser [106]. Lesser takes a geometric approach in his book, where Kane and Levinson's book [90] is more algorithmical. Their book is the main reference for Kane's equations and Lennartsson's thesis [105] is an interesting reference regarding the efficiency of this method for deriving symbolic differential equations of rbm's.

Kane's equations are an improvement from the 1960's of work by Gibbs, Appell and others. In the equations, the *kinematic differential equations* (kde) are used to find a basis transformation for the tangent space, that allows the user to in some sense choose configuration and speed parameters independently. This integrated use of converting the second order differential system into two first order differential systems, the kde and the *dynamic differential equations* (dde), can produce a combined system that is easier to work with. The main feature however, is that the method is well suited to handle non-holonomic systems and it is easy to do multibody analysis. In Lesser's interpretation and notation, a stringent difference between geometric objects and the components that represent the object is included. An example of that will be given shortly, but first some basic terminology.

> A *reference triad* consists of three non co-planar vectors and a rigid body[1]. The directions, but not the origin of these vectors are assumed fixed relative to the rigid body. The vectors comprising the triad are *free*, a vector from point A to point B, is considered equal to any other vector with the same direction and length. The combination of a *fixed point* in the body together with the triad comprise a *reference frame*. A *standard triad* and *standard reference frame* uses orthonormal triad vectors.

[1] Not necessarily a physical body, just a set of points that behave as rigid body.

Now, to distinguish between a geometric object and it's component representation, let a vector $\mathbf{r}$ in a three-dimensional vector space with standard triad $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ be expressed as

$$\mathbf{r} = r_1\mathbf{a}_1 + r_2\mathbf{a}_2 + r_3\mathbf{a}_3$$

where $r_i \in \Re$ are the vector components relative to this base. Using matrix multiplication, this can be written as follows:

$$\mathbf{r} = \underbrace{[r_1\, r_2\, r_3]}_{=r^T} \underbrace{\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix}}_{=a} = r^T a = a^T r,$$

where $r^T = [r_1\ r_2\ r_3]$ and $a^T = [\mathbf{a}_1\ \mathbf{a}_2\ \mathbf{a}_3]$. If it will be necessary to indicate which reference triad the components refer to, this will be indicated by a left superscript, i.e. to indicate that the components with respect to triad a are used, we write:

$$\mathbf{r} = \underbrace{[{}^a r_1\ {}^a r_2\ {}^a r_3]}_{={}^a r^T} \underbrace{\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix}}_{=a} = ({}^a r)^T a = a^T\, {}^a r. \tag{6.1}$$

More often, it is useful to indicate what body the components refer to, and since bodies are here denoted by a capital letter, that letter is typically used as left superscript. However, since a triad fixed in body A is typically referred to as a, this only means that (6.1) is written as follows:

$$\mathbf{r} = \underbrace{[{}^A r_1\ {}^A r_2\ {}^A r_3]}_{={}^A r^T} \underbrace{\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix}}_{=a} = \left({}^A r\right)^T a = a^T\, {}^A r$$

An inner (dot) product, $\cdot$, between two matrices means that it acts on the components of the matrices in a matrix multiplication, as in this example:

$$a^T \cdot b = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{b}_1 & \mathbf{a}_1 \cdot \mathbf{b}_2 & \mathbf{a}_1 \cdot \mathbf{b}_3 \\ \mathbf{a}_2 \cdot \mathbf{b}_1 & \mathbf{a}_2 \cdot \mathbf{b}_2 & \mathbf{a}_2 \cdot \mathbf{b}_3 \\ \mathbf{a}_3 \cdot \mathbf{b}_1 & \mathbf{a}_3 \cdot \mathbf{b}_2 & \mathbf{a}_3 \cdot \mathbf{b}_3 \end{bmatrix}$$

A special case is

$$a \cdot a^T = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{a}_1 & \mathbf{a}_1 \cdot \mathbf{a}_2 & \mathbf{a}_1 \cdot \mathbf{a}_3 \\ \mathbf{a}_2 \cdot \mathbf{a}_1 & \mathbf{a}_2 \cdot \mathbf{a}_2 & \mathbf{a}_2 \cdot \mathbf{a}_3 \\ \mathbf{a}_3 \cdot \mathbf{a}_1 & \mathbf{a}_3 \cdot \mathbf{a}_2 & \mathbf{a}_3 \cdot \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

that results in a unit matrix.

The outer (or dyadic) product between two vectors is simply expressed by writing two vectors next to each other, $\mathbf{vw} = \mathbf{D}$, thereby forming a *dyad* operator. The dot product between a dyad and a vector results in a new vector as follows:

$$\begin{aligned} \mathbf{D} \cdot \mathbf{u} &= (\mathbf{vw}) \cdot \mathbf{u} = \mathbf{v} (\mathbf{w} \cdot \mathbf{u}) \\ \mathbf{u} \cdot \mathbf{D} &= \mathbf{u} \cdot (\mathbf{vw}) = (\mathbf{u} \cdot \mathbf{v}) \mathbf{w} \end{aligned}$$

and a matrix representation of $\mathbf{D}$ is easily obtained through

$$a^T \cdot \mathbf{D} \cdot a = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{D} \cdot \mathbf{a}_1 & \mathbf{a}_1 \cdot \mathbf{D} \cdot \mathbf{a}_2 & \mathbf{a}_1 \cdot \mathbf{D} \cdot \mathbf{a}_3 \\ \mathbf{a}_2 \cdot \mathbf{D} \cdot \mathbf{a}_1 & \mathbf{a}_2 \cdot \mathbf{D} \cdot \mathbf{a}_2 & \mathbf{a}_2 \cdot \mathbf{D} \cdot \mathbf{a}_3 \\ \mathbf{a}_3 \cdot \mathbf{D} \cdot \mathbf{a}_1 & \mathbf{a}_3 \cdot \mathbf{D} \cdot \mathbf{a}_2 & \mathbf{a}_3 \cdot \mathbf{D} \cdot \mathbf{a}_3 \end{bmatrix} = {}^A D,$$

where the left superscript denotes the reference triads.

Lesser combines the idea of stacking vectors into a column matrix, with the concept of a *system vector* which for a point mechanism would describe it's configuration. Quoting from Lesser [106, p. 97]:

> First consider a mechanism that is composed of K points, labeled by $1 \ldots K$, and interconnected in such a way as the configuration is determined by n generalized coordinates. Thus assume the existence of the position vectors $\mathbf{r}^{<1}(q,t)$,$\ldots \mathbf{r}^{<K}(q,t)$, where q stands for the set of quantities $q_1, q_2, \ldots q_n$. The $3K$ component vector that represents a configuration point is indicated by the convention of dropping the label from $\mathbf{r}^{<L}$, that is by writing:
>
> $$\mathbf{r}^{<} = \begin{bmatrix} \mathbf{r}^{<1} \\ \mathbf{r}^{<2} \\ \vdots \\ \mathbf{r}^{<K} \end{bmatrix}$$
>
> with each component of the K column being a position vector to the L^{th} point. $\ldots$ From now on we will use the name *Kvectors*.

For the point mechanism, each vector can be represented by three components and hence the mechanism by $3K$ components. For a rigid body mechanism, $6K$ components are needed. The point in configuration space can in general only move in a subspace which is locally referred to as the *tangent space* and the term *tangent vectors* will be used to denote vectors in that space. In fact, a linearly independent

set of tangent vectors forms a basis for the tangent space and these are important as will be described later.

Going back to Kvectors, the *system velocity vector*, describes the velocities and angular velocities of K rigid bodies in a mechanism and is written as follows:

$$\mathbf{v}^{<} = \begin{bmatrix} \mathbf{v}^{<1} \\ \boldsymbol{\omega}^{<1} \\ \vdots \\ \mathbf{v}^{<K} \\ \boldsymbol{\omega}^{<K} \end{bmatrix}.$$

where $\mathbf{v}^{<k}$ and $\boldsymbol{\omega}^{<k}$ are the velocity[2] and angular velocity of the k^{th} body. Similar to $\mathbf{v}^{<}$, a Kvector with the applied forces onto each rigid body can be written as follows:

$$\mathbf{F}^{<} = \begin{bmatrix} \mathbf{F}^{<1} \\ \mathbf{T}^{<1} \\ \vdots \\ \mathbf{F}^{<K} \\ \mathbf{T}^{<K} \end{bmatrix}.$$

Here, $\mathbf{F}^{<i}$ and $\mathbf{T}^{<i}$ means the sum of external forces and torques applied to the i^{th} body.

A commutative bilinear operator, $\bullet$, is defined for two Kvectors of equal size and defined as the sum of scalar products between the corresponding vectors. For $\mathbf{v}^{<}$and $\mathbf{F}^{<}$, the product is expanded as follows

$$\mathbf{v}^{<} \bullet \mathbf{F}^{<} = \sum_{i=1}^{K} \mathbf{v}^{<i} \cdot \mathbf{F}^{<i}$$

which here corresponds to the instantaneous power produced (or consumed) by the mechanism. However, note that not all products are physically meaningful (for instance, $\mathbf{F}^{<} \bullet \mathbf{F}^{<}$).

Assume that the vector space which $\mathbf{v}^{<}$ belongs to has the a basis of Kvectors, $\{\mathbf{a}_1^{<}, \mathbf{a}_2^{<}, \ldots, \mathbf{a}_n^{<}\}$ where n is the dimension of the vector space, then $\mathbf{v}^{<}$ could be

[2]The velocity can actually be the velocity of any *index* point of the body, but in this work the index point will always be the centre of mass of that rigid body.

expressed as

$$\mathbf{v}^{<} = v_1^{<}\mathbf{a}_1^{<} + v_2^{<}\mathbf{a}_2^{<} + \cdots + v_n^{<}\mathbf{a}_n^{<} = \begin{bmatrix} v_1^{<} & v_2^{<} & \dots & v_n^{<} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a}_1^{<} \\ \mathbf{a}_2^{<} \\ \vdots \\ \mathbf{a}_n^{<} \end{bmatrix} = v^{<T} a^{<}$$

where $v_i^{<} \in \Re$ are the scalar components and $a^{<}$ denotes the collection (column matrix) of base Kvectors.

6.2.1. Derivation of differential equations

The process of deriving the differential equations for a rbm is now surprisingly straight forward. Note that the time variant case is not described here, nor the case systems with non-holonomic constraints or redundant coordinates.

1. Formulate the system velocity vector, $\mathbf{v}^{<}$ as a function of generalized coordinates, q and $\dot{q}$.
2. The system velocity vector will be affine with respect to $\dot{q}$ and can be partitioned into
$$\mathbf{v}^{<} = \tau^T \dot{q} + \boldsymbol{\tau}_t^{<} \tag{6.2}$$
where $\tau^T = [\boldsymbol{\tau}_1^{<}, \boldsymbol{\tau}_1^{<}, \dots, \boldsymbol{\tau}_n^{<}]$ is a collection of *tangent vectors* that can be used as a local basis of the tangent space. $\boldsymbol{\tau}_t^{<}$ describes the time-variant part of $\mathbf{v}^{<}$ which is zero in the time in-variant case and will not be considered further.
3. Choose an invertible linear transformation[3], $W^{\beta\tau}$ that converts generalized velocities, $\dot{q}$, into generalized speeds[4], w, i.e.
$$w = W^{\beta\tau}\dot{q}.$$
where $W^{\beta\tau}$ can also be thought of as basis transformation from τ to a new basis β. As an example of how the transform could be chosen, assume we would like to express the velocity of a body, B as follows
$$\mathbf{v}^{<B} = a^T \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix},$$

[3] An affine transform, $u = W^{\beta\tau}\dot{q} + f$ is also possible, but will not be used in this work.
[4] Note that Lesser typically denotes generalized speeds with u but here the symbol w is used instead.

i.e. in simple terms of the generalized speeds w_1, w_2 and w_3 and an orthonormal triad $a^T = [\mathbf{a}_1\ \mathbf{a}_2\ \mathbf{a}_3]$. From (6.2), we have already determined $\mathbf{v}^{<}$ as a function of $\dot{q}$, and therefore also $\mathbf{v}^{<B}$ as a function of $\dot{q}$ and we have that,

$$\mathbf{v}^{<B} = \left[\boldsymbol{\tau}_1^{<B}, \boldsymbol{\tau}_1^{<B}, \ldots, \boldsymbol{\tau}_n^{<B}\right] \dot{q}.$$

We can now write the following system of equations:

$$a^T \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \left[\boldsymbol{\tau}_1^{<B}, \boldsymbol{\tau}_2^{<B}, \ldots, \boldsymbol{\tau}_n^{<B}\right] \dot{q}$$

where $\boldsymbol{\tau}_i^{<_B}$ is the velocity component of body B that is proportional to the generalized velocity $\dot{q}_i$. Premultiplying both sides with $a\cdot$ produces

$$\underbrace{a \cdot a^T}_{=I_{3\times 3}} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = a \cdot \left[\boldsymbol{\tau}_1^{<B}, \boldsymbol{\tau}_2^{<B}, \ldots, \boldsymbol{\tau}_n^{<B}\right] \dot{q}$$

where $a^T \cdot a = I$ and the right side is further expanded as follows:

$$\begin{aligned} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} &= a \cdot \left[\boldsymbol{\tau}_1^{<B}, \boldsymbol{\tau}_2^{<B}, \ldots, \boldsymbol{\tau}_n^{<B}\right] \dot{q} \\ &= \begin{bmatrix} a \cdot \boldsymbol{\tau}_1^{<B} & a \cdot \boldsymbol{\tau}_2^{<_B} & \ldots & a \cdot \boldsymbol{\tau}_n^{<_B} \end{bmatrix} \dot{q} \\ &= \begin{bmatrix} \mathbf{a}_1 \cdot \boldsymbol{\tau}_1^{<B} & \mathbf{a}_1 \cdot \boldsymbol{\tau}_2^{<B} & \ldots & \mathbf{a}_1 \cdot \boldsymbol{\tau}_n^{<B} \\ \mathbf{a}_2 \cdot \boldsymbol{\tau}_1^{<B} & \mathbf{a}_2 \cdot \boldsymbol{\tau}_2^{<B} & \ldots & \mathbf{a}_2 \cdot \boldsymbol{\tau}_n^{<B} \\ \mathbf{a}_3 \cdot \boldsymbol{\tau}_1^{<B} & \mathbf{a}_3 \cdot \boldsymbol{\tau}_2^{<B} & \ldots & \mathbf{a}_3 \cdot \boldsymbol{\tau}_n^{<B} \end{bmatrix} \dot{q} \end{aligned}$$

Doing similar choices for the remaining generalized speeds, we end up with a system of equations as follows:

$$u = W^{\beta\tau}\dot{q}.$$

With a valid choice where $\exists W^{\tau\beta} = \left(W^{\beta\tau}\right)^{-1}$, the kinematic differential equations are simply

$$\dot{q} = W^{\tau\beta} w. \tag{6.3}$$

4. Determine a new basis for the tangent space. Substituting (6.3) into (6.2) we

get

$$\begin{aligned} \mathbf{v}^{<} &= \tau^T \dot{q} \\ &= \tau^T W^{\tau\beta} u \\ &= \beta^T w \end{aligned}$$

where we identified the new basis β as

$$\beta^T = \tau^T W^{\tau\beta}.$$

5. Newton's equations of motions can be written as

$$\dot{\mathbf{p}}^{<} = \mathbf{F}_a^{<} + \mathbf{F}_c^{<}$$

where $\mathbf{F}_a^{<}$ and $\mathbf{F}_c^{<}$ are the applied and constraint forces respectively, and $\dot{\mathbf{p}}^{<}$ is the time derivative of the momentum $\mathbf{p}^{<}$ relative to an inertial frame, N,

$$\dot{\mathbf{p}}^{<} = \frac{{}^N\partial \mathbf{p}^{<}}{\partial t}.$$

The momentum can be written as

$$\mathbf{p}^{<} = \begin{bmatrix} m_1\mathbf{U} & & & & \\ & \mathbf{I}_1 & & & \\ & & \cdots & & \\ & & & m_K\mathbf{U} & \\ & & & & \mathbf{I}_K \end{bmatrix} \cdot \mathbf{v}^{<} = \begin{bmatrix} m_1\mathbf{v}^{<1} \\ \mathbf{I}_1 \cdot \mathbf{v}^{<1} \\ \cdots \\ m_K\mathbf{v}^{<K} \\ \mathbf{I}_K \cdot \mathbf{v}^{<K} \end{bmatrix}$$

where m_i is the mass of body i, $\mathbf{U}$ is a unit dyad and $\mathbf{I}_i$ is the inertia of body i around its index point, i.e. here its centre of mass.

6. Constraint forces vanish when projected onto the tangent space, i.e.

$$\beta_i \bullet \mathbf{F}_c^{<} = 0,$$

since the constraint forces by definition do not produce any power and the projection corresponds to power. This can be used to eliminate the constraint forces as follows:

$$\beta \bullet \dot{\mathbf{p}}^{<} = \beta \bullet \mathbf{F}_a^{<} + \underbrace{\beta \bullet \mathbf{F}_c^{<}}_{=[0,0,\ldots,0]^T}$$

and the dynamic equations have been derived. They will be linear in $\dot{w}$ and by extracting the coefficients of $\dot{w}$, the dynamic equations can be rewritten as

$$M\dot{w} = F$$

where F are the remaining terms.

The kde and dde have now been derived as

$$\dot{q} = W^{\tau\beta} w$$

and

$$M\dot{w} = F.$$

If it is desired, the dde could of course be further partitioned into a standard form such as:

$$M(q)\,\dot{w} + C(w,q) + g(q) = f$$

where the Coriolis forces, $C(w,q)$ and gravitational forces, $g(q)$ have been written separately.

6.3. Derivation of WARP1 rigid body model

We will now proceed with the derivation of a rbm of WARP1 following the manner just described. Figure 6.1 illustrates the rbm where the legs have been enumerated as follows:

1 – front left, 2 – front right, 3 – rear left and 4 – rear right leg,

but the actual enumeration is not important.

Figure 6.1 also illustrates the two main reference triads:

- N The world frame is denoted N and n denotes the triad $n = [\boldsymbol{n}_1\, \boldsymbol{n}_2\, \boldsymbol{n}_3]^T$ where $\mathbf{n}_3$ points upwards, i.e. $-\mathbf{n}_3$ has the same direction as the field of gravity.
- B The trunk fixed triad is denoted $b = [\boldsymbol{b}_1\, \boldsymbol{b}_2\, \boldsymbol{b}_3]^T$, where $\mathbf{b}_1$ points forward, $\mathbf{b}_2$ points left, and $\mathbf{b}_3$ points up.

Additionally, the figure illustrates the location of different points such as H_3, the hip's centre of rotation of leg 3. Similarly, the knee's centre of rotation of leg 3 is denoted K_3. This is summarized in table 6.2.

To make the notation more compact, vectors between points within a leg will only indicate the leg number with a subscript for one of the points, i.e. the vector from H_3 to K_3 will be denoted with $\mathbf{r}^{HK_3}$ instead of $\mathbf{r}^{H_3K_3}$. Additionally, vectors between points within one rigid body will of course be fixed relative to that body and table 6.3 lists these vectors which can be considered to represent the physical

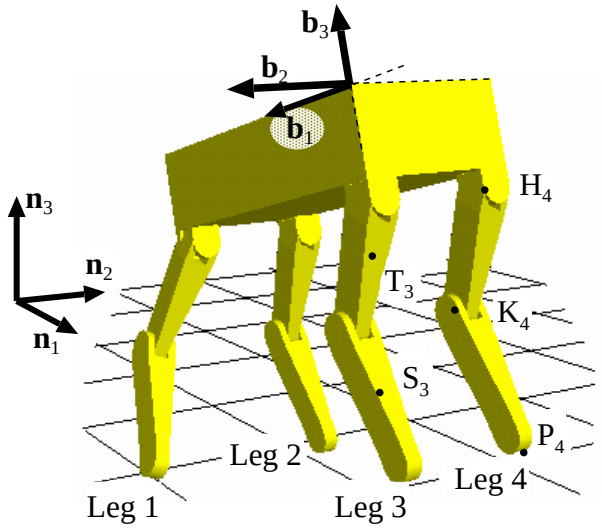


Figure 6.1: Illustration of the inertial frame N, the trunk fixed triad B and some of the points in the WARP1 rbm. Relative to the trunk, the vector $\mathbf{b}_1$ points forward, $\mathbf{b}_2$ to the left and $\mathbf{b}_3$ upwards. Only one point is drawn for each kind of point in a leg. For instance, P_4 that represents the pad of leg 4 is drawn whereas P_1, P_2 and $P_4$are not drawn. See table 6.2 for a complete listing and description of points in the rbm.

parameters of the rbm. As an example, consider, the vector from the trunk's centre of mass to the hip centre of rotation of leg 3:

$$\mathbf{r}^{BH_3} = {}^{B}r_1^{BH_3}\mathbf{b}_1 + {}^{B}r_2^{BH_3}\mathbf{b}_2 + {}^{B}r_3^{BH_3}\mathbf{b}_3 = {}^{B}r^{BH_3}b$$

where ${}^{B}r^{BH_3}$ would then be a column matrix with physical parameters, i.e. the vector components in the trunk frame.

6.3.1. Generalized coordinates and reference frames

The generalized coordinates are chosen as follows:

- Let $q^b = [q_x\ q_y\ q_z]^T$ be the generalized coordinates that describe the position of the trunk's centre of mass, i.e.

$$\mathbf{r}^{NB} = \mathbf{r}^{B} = b^T q^B \tag{6.4}$$

  where $\mathbf{r}^{NB}$ is denoted $\mathbf{r}^{B}$ to get a more compact notation.

Table 6.2: Description of points and bodies in the WARP1 rbm, see also figure 6.1.

Symbol	Description of point	Description of body
N	Origin of lab frame	Inertial reference body
B	Trunk centre of mass	Trunk
T_l	**T**high centre of mass of leg l	**T**high of leg l
S_l	**S**hank centre of mass of leg l	**S**hank of leg l
H_l	**H**ip rotation point in the hip joint of leg l	
K_l	**K**nee rotation point in the knee joint of leg l	
P_l	**P**ad ground contact point of leg l	

Table 6.3: List of body fixed vectors that represent physical parameters

Vector	Description
$\mathbf{r}^{BH_l}$	Vector from the trunk centre of mass to hip joint for leg l.
$\mathbf{r}^{HT_l}$	Vector from the hip joint to thigh centre of mass for leg l.
$\mathbf{r}^{HK_l}$	Vector from the hip joint to knee for leg l.
$\mathbf{r}^{KS_l}$	Vector from the knee joint to shank centre of mass for leg l.
$\mathbf{r}^{KP_l}$	Vector from the knee joint to pad of leg l.

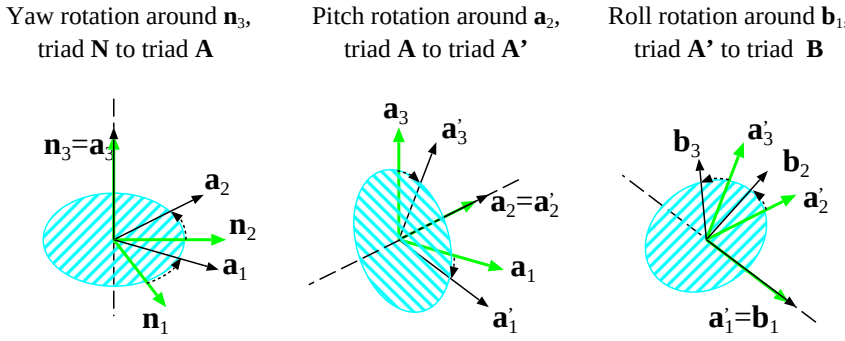


Figure 6.2: Transformation from the triad N to the triad B using the Yaw-pitch-roll parametrization.

- Let $q^o = [q_{yaw}\ q_{pitch}\ q_{roll}]^T$ be the generalized coordinates that describe the yaw, pitch and roll angles of the triad b relative to the triad n. The sequence of rotations is illustrated in figure 6.2. The angular velocity of B relative to N, denoted $\boldsymbol{\omega}^B$, can be found by summing the angular velocities of the individual rotations as follows:

$$\boldsymbol{\omega}^B = \dot{q}_{yaw}\mathbf{n}_3 + \dot{q}_{pitch}\mathbf{a}_2 + \dot{q}_{roll}\mathbf{b}_1. \tag{6.5}$$

- Finally, let $q_{H_{aa},l}$, $q_{H_{fe},l}$ and $q_{K_{fe},l}$ denote the angle of rotation (starting from the trunk): hip abduction/adduction, hip flexion/extension and knee flexion/extension for leg l.

However, since the complexity of the calculations increase rapidly with the number of links and joints in series, it is useful to reduce the effect of that by not letting $q_{K_{fe},l}$ simply be the knee angle. Instead, let the knee angle, i.e. the angle between the thigh and the shank be expressed as $q_{H_{fe},l} - q_{K_{fe},l}$.

We introduce the following reference frames (see figure 6.3):

H_l Hip reference frame for leg l, $h^l = \left[\mathbf{h}_1^l\ \mathbf{h}_2^l\ \mathbf{h}_3^l\right]^T$. H_l is B rotated around $\mathbf{b}_2$, i.e. $\mathbf{h}_2^l = \mathbf{b}_2$. The rotation has a constant offset of $\frac{\pi}{2}$ radians, i.e. $q_{H_{aa},l} = 0 \Rightarrow \mathbf{h}_1^l = -\mathbf{b}_3$ which means that $\mathbf{h}_1^l$ points downwards.

T_l Thigh reference frame for leg l, $t^l = \left[\mathbf{t}_1^l\ \mathbf{t}_2^l\ \mathbf{t}_3^l\right]^T$. T_l is H_l rotated around $\mathbf{h}_3^l$, i.e. $\mathbf{t}_3^l = \mathbf{h}_3^l$ and $\mathbf{t}_1^l$ points along the thigh.

S_l Shank reference frame for leg l, $s^l = \left[\mathbf{s}_1^l\ \mathbf{s}_2^l\ \mathbf{s}_3^l\right]^T$. S_l is H_l rotated around $\mathbf{h}_3^l$, i.e. $\mathbf{s}_3^l = \mathbf{h}_3^l$ and $\mathbf{s}_1^l$ points along the shank.

The estimated values for the parameter vectors, body masses and inertias can now be expressed in their respective body fixed triads; these values are shown in table 6.4 and table 6.5. However, the longitudinal position of WARP1's legs can easily be modified, and $\mathbf{r}^{BH_l}$ may therefore vary. In table 6.4 the most commonly used values are shown.

6.3.2. System velocity vector

We will now define the system velocity vector, $\mathbf{v}^<$. Since it is a just a column matrix with velocities of the different bodies, it is practical to partition it into $L + 1$ parts as follows:

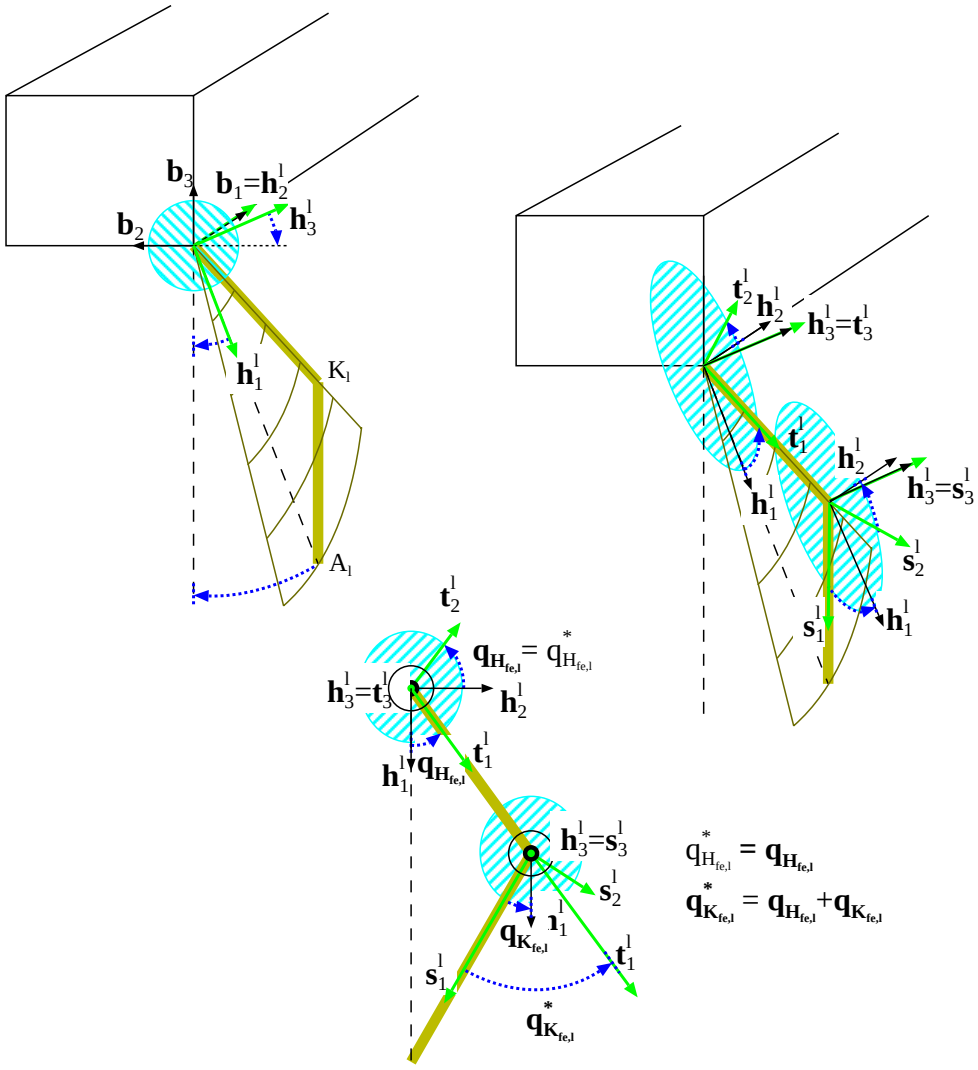


Figure 6.3: Illustration of leg kinematics and reference triads for leg 4 ($l = 4$). The rotations are "positive" according to the right hand rule. The hip abduction/adduction joint rotates an angle $q_{H_{aa},l}$ around $\mathbf{h}_2^l = \mathbf{b}_1$ (the illustrated angle is negative). For leg 4 this causes the leg plane to rotate outwards in the left figure. The figure to the right illustrates the leg's rotations within the leg plane, where the hip flexion/extension joint is rotated an angle $q_{H_{fe},l}$ around $\mathbf{h}_3^l$ (the illustrated angle is positive) and the knee flexion/extension joint is rotated an angle $q_{K_{fe},l}$ (the illustrated angle is negative). Finally, only the leg plane is shown in the bottom figure. Note that $q_{K_{fe},l}$ is not really the knee angle, which is denoted $q^*_{K_{fe},l}$. The relationsship between the "relative" joint angles and the generalized coordinates is also illustrated.

Table 6.4: Estimated values of physical parameter vectors for WARP1, where l stands for the leg number, $l \in [1, L]$. Most values were estimated by Friede and Kylström [41].

Vector	Value	Unit
$\mathbf{r}^{BH_l}$	$-\left(0.02 + (-1)^l\ 0.31\right)\ \mathbf{b}_1 + (-1)^l\ 0.25\,\mathbf{b}_2 - 0.14\,\mathbf{b}_3$	[m]
$\mathbf{r}^{HT_l}$	$0.15\,\mathbf{t}_1^l - (-1)^l\ 0.02\,\mathbf{t}_3^l$	[m]
$\mathbf{r}^{HK_l}$	$0.29\,\mathbf{t}_1^l$	[m]
$\mathbf{r}^{KS_l}$	$0.05\,\mathbf{s}_1^l - (-1)^l\ 0.03\,\mathbf{s}_3^l$	[m]
$\mathbf{r}^{KP_l}$	$0.30\,\mathbf{s}_1^l$	[m]
$\mathbf{I}^B$	$1.6\,\mathbf{b}_1\mathbf{b}_1 + 3.4\,\mathbf{b}_2\mathbf{b}_2 + 4.3\,\mathbf{b}_3\mathbf{b}_3$	$\left[\mathrm{kg\,m^2}\right]$
$\mathbf{I}^T$	$0.003\,\mathbf{t}_1^l\mathbf{t}_1^l + 0.03\,\mathbf{t}_2^l\mathbf{t}_2^l + 0.03\,\mathbf{t}_3^l\mathbf{t}_3^l$	$\left[\mathrm{kg\,m^2}\right]$
$\mathbf{I}^S$	$0.001\,\mathbf{s}_1^l\mathbf{s}_1^l + 0.02\,\mathbf{s}_2^l\mathbf{s}_2^l + 0.02\,\mathbf{s}_3^l\mathbf{s}_3^l$	$\left[\mathrm{kg\,m^2}\right]$

Table 6.5: Mass and inertia of WARP1 body parts.

Symbol	Value	Unit	Description
m^B	37.6	[kg]	Trunk mass
m^T	3.7	[kg]	Mass of thigh l
m^S	2.0	[kg]	Mass of shank l
$\mathbf{I}^B$	$1.6\,\mathbf{b}_1\mathbf{b}_1 + 3.4\,\mathbf{b}_2\mathbf{b}_2 + 4.3\,\mathbf{b}_3\mathbf{b}_3$	$\left[\mathrm{kg\,m^2}\right]$	Inertia of trunk
$\mathbf{I}^{T_l}$	$0.003\,\mathbf{t}_1^l\mathbf{t}_1^l + 0.03\,\mathbf{t}_2^l\mathbf{t}_2^l + 0.03\,\mathbf{t}_3^l\mathbf{t}_3^l$	$\left[\mathrm{kg\,m^2}\right]$	Inertia of thigh l
$\mathbf{I}^{S_l}$	$0.001\,\mathbf{s}_1^l\mathbf{s}_1^l + 0.02\,\mathbf{s}_2^l\mathbf{s}_2^l + 0.02\,\mathbf{s}_3^l\mathbf{s}_3^l$	$\left[\mathrm{kg\,m^2}\right]$	Inertia of shank l

$$\mathbf{v}^{<} = \begin{bmatrix} \mathbf{v}^{<0} \\ \mathbf{v}^{<1} \\ \vdots \\ \mathbf{v}^{<L} \end{bmatrix} \tag{6.6}$$

where $\mathbf{v}^{<0}$ is the velocity and angular velocity of the trunk and $\mathbf{v}^{<l}$ are the velocities and angular velocities of the rigid bodies in leg l. $\mathbf{v}^{<0}$ is expressed as follows

$$\mathbf{v}^{<0} = \begin{bmatrix} \mathbf{v}^B \\ \boldsymbol{\omega}^B \end{bmatrix} \tag{6.7}$$

and to find $\mathbf{v}^B$ we just differentiate (6.4) relative to frame N, i.e.

$$\mathbf{v}^B = \frac{{}^N\partial \mathbf{r}^B}{\partial t} = \frac{{}^N\partial b^T q^B}{\partial t} = \frac{{}^B\partial b^T q^B}{\partial t} + \boldsymbol{\omega}^B \times \mathbf{r}^B = b^T \dot{q}^B + \boldsymbol{\omega}^B \times \mathbf{r}^B$$

and then $\mathbf{v}^{<0}$ is simply

$$\mathbf{v}^{<0} = \begin{bmatrix} \mathbf{v}^B \\ \boldsymbol{\omega}^B \end{bmatrix} = \begin{bmatrix} b^T \dot{q}^B + (\dot{q}_{yaw}\mathbf{n}_3 + \dot{q}_{pitch}\mathbf{a}_2 + \dot{q}_{roll}\mathbf{b}_1) \times \mathbf{r}^B \\ \dot{q}_{yaw}\mathbf{n}_3 + \dot{q}_{pitch}\mathbf{a}_2 + \dot{q}_{roll}\mathbf{b}_1 \end{bmatrix}$$

where $\boldsymbol{\omega}^B$ was replaced using (6.5).

Expressing $\mathbf{v}^{<l},\ l = [1..L]$ is done in a similar way, but require too much space to be described here.

6.3.3. Choosing generalized speeds

The size of the differential equations is not only affected by the choice of generalized coordinates, but also by the choice of generalized speeds. Therefore we wish the trunk angular velocity to be expressed in the following manner using generalized speeds:

$$\boldsymbol{\omega}^B = w_{roll}\mathbf{b}_1 + w_{pitch}\mathbf{b}_2 + w_{yaw}\mathbf{b}_3 = b^T w^o, \tag{6.8}$$

i.e. we can expand this into

$$w_{roll}\mathbf{b}_1 + w_{pitch}\mathbf{b}_2 + w_{yaw}\mathbf{b}_3 = \dot{q}_{yaw}\mathbf{n}_3 + \dot{q}_{pitch}\mathbf{a}_2 + \dot{q}_{roll}\mathbf{b}_1$$

that by premultiplying with $b\cdot$ expands to the following matrix equation:

$$\begin{bmatrix} w_{roll} \\ w_{pitch} \\ w_{yaw} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \cdot (\dot{q}_{yaw}\mathbf{n}_3 + \dot{q}_{pitch}\mathbf{a}_2 + \dot{q}_{roll}\mathbf{b}_1) \\ \mathbf{b}_2 \cdot (\dot{q}_{yaw}\mathbf{n}_3 + \dot{q}_{pitch}\mathbf{a}_2 + \dot{q}_{roll}\mathbf{b}_1) \\ \mathbf{b}_3 \cdot (\dot{q}_{yaw}\mathbf{n}_3 + \dot{q}_{pitch}\mathbf{a}_2 + \dot{q}_{roll}\mathbf{b}_1) \end{bmatrix}.$$

Simplifying and replacing the dot products result in the following system of equations:

$$\begin{aligned} w_{roll} &= \dot{q}_{roll} - \sin(q_{pitch})\,\dot{q}_{yaw} \\ w_{pitch} &= \cos(q_{roll})\,\dot{q}_{pitch} + \cos(q_{pitch})\sin(q_{roll})\,\dot{q}_{yaw} \\ w_{yaw} &= \cos(q_{roll})\cos(q_{pitch})\,\dot{q}_{yaw} - \sin(q_{roll})\,\dot{q}_{pitch} \end{aligned}$$

This is solved into:

$$\begin{aligned} \dot{q}_{yaw} &= \frac{\cos(q_{roll})}{\cos(q_{pitch})} w_{yaw} + \frac{\sin(q_{roll})}{\cos(q_{pitch})} w_{pitch} \\ \dot{q}_{pitch} &= \cos(q_{roll})\, w_{pitch} - \sin(q_{roll})\, w_{yaw} \\ \dot{q}_{roll} &= w_{roll} + \tan(q_{pitch})\sin(q_{roll})\, w_{pitch} + \tan(q_{pitch})\cos(q_{roll})\, w_{yaw} \end{aligned}$$

Notice that $\dot{q}^o$ is linear in the generalized speeds as expected and therefore can be written as:

$$\dot{q}^o = \begin{bmatrix} \dot{q}_{yaw} \\ \dot{q}_{pitch} \\ \dot{q}_{roll} \end{bmatrix} = \begin{bmatrix} 0 & \frac{\sin(q_{roll})}{\cos(q_{pitch})} & \frac{\cos(q_{roll})}{\cos(q_{pitch})} \\ 0 & \cos(q_{roll}) & -\sin(q_{roll}) \\ 1 & \tan(q_{pitch})\sin(q_{roll}) & \tan(q_{pitch})\cos(q_{roll}) \end{bmatrix} \begin{bmatrix} w_{roll} \\ w_{pitch} \\ w_{yaw} \end{bmatrix}$$

Let us now, similar to the choice of $\boldsymbol{\omega}^B$ in equation (6.8), choose to express $\mathbf{v}^B$ in this manner:

$$\mathbf{v}^B = w_x \mathbf{b}_1 + w_y \mathbf{b}_2 + w_z \mathbf{b}_3 = b^T w^B \tag{6.9}$$

Since

$$\begin{aligned} \mathbf{v}^B &= \frac{{}^N\partial \mathbf{r}^B}{\partial t} = \frac{{}^B\partial \mathbf{r}^B}{\partial t} + \boldsymbol{\omega}^B \times \mathbf{r}^B \\ &= \left\{ \mathbf{r}^B = b^T q^B \Rightarrow \frac{{}^B\partial \mathbf{r}^B}{\partial t} = b^T \dot{q}^B \right\} \\ &= b^T \dot{q}^b + b^T w^o \times b^T q^B \end{aligned}$$

it is now easy to solve for the generalized velocities as a function of the generalized speeds, i.e.:

$$\dot{q}^B = w^B - w^o \times q^B.$$

Now we make a simple choice for the remaining generalized speeds,

$$\dot{q}_{i,l} = \frac{\partial q_{i,l}}{\partial t} = w_{i,l},\ i \in \{H_{aa}, H_{fe}, K_{fe}\} \text{ and } l \in [1, L]$$

and we have the kde:

$$\begin{aligned}
\dot{q}^B &= w^B - w^o \times q^B \\
\dot{q}^o &= \begin{bmatrix} 0 & \frac{\sin(q_{roll})}{\cos(q_{pitch})} & \frac{\cos(q_{roll})}{\cos(q_{pitch})} \\ 0 & \cos(q_{roll}) & -\sin(q_{roll}) \\ 1 & \tan(q_{pitch})\sin(q_{roll}) & \tan(q_{pitch})\cos(q_{roll}) \end{bmatrix} w^o \\
\dot{q}_{H_{aa},l} &= w_{H_{aa},l} \quad l \in [1, L] \\
\dot{q}_{H_{fe},l} &= w_{H_{fe},l} \quad l \in [1, L] \\
\dot{q}_{K_{fe},l} &= w_{K_{fe},l} \quad l \in [1, L]
\end{aligned} \tag{6.10}$$

6.3.4. System momentum vector

Given $\mathbf{v}^<$ it is easy to derive the system momentum vector, $\mathbf{p}^<$. First partition $\mathbf{p}^<$ in the same manner as $\mathbf{v}^<$, i.e. let

$$\mathbf{p}^< = \begin{bmatrix} \mathbf{p}^{<_0} \\ \mathbf{p}^{<_1} \\ \vdots \\ \mathbf{p}^{<_L} \end{bmatrix}$$

where $\mathbf{p}^{<_0}$ is the momentum and angular momentum of the trunk and $\mathbf{p}^{<_l}$ correspondingly for leg l. Let m^B denote the trunk mass and $\mathbf{I}^B$ a dyad that represents the trunk inertia. It can be written as

$$\mathbf{I}^B = b^T\, {}^B I^B b$$

where ${}^B I^B$ is the inertia matrix of the trunk relative to the trunk. Then $\mathbf{p}^{<_0}$ can be written as:

$$\mathbf{p}^{<_0} = \begin{bmatrix} m^B \mathbf{v}^B \\ \mathbf{I}^B \cdot \boldsymbol{\omega}^B \end{bmatrix}. \tag{6.11}$$

The time derivative of the momentum relative to the frame N is denoted $\dot{\mathbf{p}}^<$,

$$\dot{\mathbf{p}}^< = \frac{{}^N \partial \mathbf{p}^<}{\partial t}.$$

It can also be partitioned

$$\dot{\mathbf{p}}^{<} = \begin{bmatrix} \dot{\mathbf{p}}^{<_0} \\ \dot{\mathbf{p}}^{<_1} \\ \vdots \\ \dot{\mathbf{p}}^{<_L} \end{bmatrix} = \begin{bmatrix} \frac{{}^N\partial \mathbf{p}^{<_0}}{\partial t} \\ \frac{{}^N\partial \mathbf{p}^{<_1}}{\partial t} \\ \vdots \\ \frac{{}^N\partial \mathbf{p}^{<_L}}{\partial t} \end{bmatrix}.$$

From (6.11) we can find $\dot{\mathbf{p}}^{<_0}$ with some algebra. First note that

$$\dot{\mathbf{p}}^{<_0} = \begin{bmatrix} \frac{{}^N\partial m^B \mathbf{v}^B}{\partial t} \\ \frac{{}^N\partial \mathbf{I}^B \boldsymbol{\omega}^B}{\partial t} \end{bmatrix}$$

and then expand that as follows

$$\frac{{}^N\partial m^B \mathbf{v}^B}{\partial t} = m^B \frac{{}^B\partial \mathbf{v}^B}{\partial t} + \boldsymbol{\omega}^B \times m^B \mathbf{v}^B = m^B b^T \dot{w}^B + m^B b^T \left(w^o \times w^B\right)$$

$$\frac{{}^N\partial \mathbf{I}^B \cdot \boldsymbol{\omega}^B}{\partial t} = \frac{{}^B\partial \mathbf{I}^B \cdot \boldsymbol{\omega}^B}{\partial t} + \boldsymbol{\omega}^B \times \left(\mathbf{I}^B \cdot \boldsymbol{\omega}^B\right) = \mathbf{I}^B \cdot b^T \dot{w}^o + b^T w^o \times \left(\mathbf{I}^B \cdot b^T w^o\right),$$

and $\dot{\mathbf{p}}^{<_0}$ can now be expressed using generalized speeds as follows

$$\dot{\mathbf{p}}^{<_0} = \begin{bmatrix} m^B b^T \dot{w}^B + m^B b^T \left(w^o \times w^B\right) \\ \mathbf{I}^B \cdot b^T \dot{w}^o + b^T w^o \times \left(\mathbf{I}^B \cdot b^T w^o\right) \end{bmatrix}.$$

6.3.5. Tangent vectors

The tangent vectors can be found from $\mathbf{v}^{<}$ by taking partial derivatives with respect to the generalized speeds (since $\mathbf{v}^{<}$ is linear in w), i.e.

$$\boldsymbol{\tau}_i^{<} = \frac{\partial \mathbf{v}^{<}}{\partial w_i}. \tag{6.12}$$

Partitioning $\boldsymbol{\tau}_i^{<}$ as $\mathbf{v}^{<}$ (6.6), we let

$$\boldsymbol{\tau}_i^{<} = \begin{bmatrix} \boldsymbol{\tau}^{<_0} \\ \boldsymbol{\tau}^{<_1} \\ \vdots \\ \boldsymbol{\tau}^{<_L} \end{bmatrix}$$

and therefore together with (6.12) we see that

$$\boldsymbol{\tau}_i^{<} = \begin{bmatrix} \boldsymbol{\tau}_i^{<0} \\ \boldsymbol{\tau}_i^{<1} \\ \vdots \\ \boldsymbol{\tau}_i^{<L} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{v}^{<0}}{\partial w_i} \\ \frac{\partial \mathbf{v}^{<1}}{\partial w_i} \\ \vdots \\ \frac{\partial \mathbf{v}^{<L}}{\partial w_i} \end{bmatrix}.$$

However, from (6.7) we easily find the first partial derivatives:

$$\boldsymbol{\tau}_i^{<0} = \frac{\partial \mathbf{v}^{<0}}{\partial w_i} = \begin{bmatrix} \frac{\partial w^B b^T}{\partial w_i} \\ \frac{\partial w^o b^T}{\partial w_i} \end{bmatrix} = \begin{bmatrix} \frac{\partial w_x \mathbf{b}_1 + w_y \mathbf{b}_2 + w_z \mathbf{b}_3}{\partial w_i} \\ \frac{\partial w_{roll} \mathbf{b}_1 + w_{pitch} \mathbf{b}_2 + w_{yaw} \mathbf{b}_3}{\partial w_i} \end{bmatrix}$$

and find that

$$\begin{aligned} \boldsymbol{\tau}_x^{<0} &= \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{0} \end{bmatrix} \quad \boldsymbol{\tau}_{roll}^{<0} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_1 \end{bmatrix} \quad \boldsymbol{\tau}_{H_{aa},l}^{<0} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \\ \boldsymbol{\tau}_y^{<0} &= \begin{bmatrix} \mathbf{b}_2 \\ \mathbf{0} \end{bmatrix} \quad \boldsymbol{\tau}_{pitch}^{<0} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_2 \end{bmatrix} \quad \boldsymbol{\tau}_{H_{fe},l}^{<0} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \\ \boldsymbol{\tau}_z^{<0} &= \begin{bmatrix} \mathbf{b}_3 \\ \mathbf{0} \end{bmatrix} \quad \boldsymbol{\tau}_{yaw}^{<0} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_3 \end{bmatrix} \quad \boldsymbol{\tau}_{K_{fe},l}^{<0} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{6.13}$$

6.3.6. Applied forces and torques

The applied forces and torques can (also) be partitioned as follows:

$$\mathbf{F}^{<} = \begin{bmatrix} \mathbf{F}^{<0} \\ \mathbf{F}^{<1} \\ \vdots \\ \mathbf{F}^{<L} \end{bmatrix}$$

where $\mathbf{F}^{<0}$ are the applied forces onto the trunk part and $\mathbf{F}^{<l}$ are the forces on leg l. For the trunk, these forces include

$$\tau_{H_{aa},l}\mathbf{b}_2 \text{ and } \tau_{H_{fe},l}\mathbf{h}_3^l$$

that are the torques applied by the hip actuators (abduction/adduction, $\tau_{H_{aa},l}$, and flexion/extension, $\tau_{H_{fe},l}$) onto the thigh of leg l. These torques will cause a reaction force onto the trunk and $\mathbf{R}^{<0}$ can be written as:

$$\mathbf{F}^{<0} = \begin{bmatrix} -m^B g \mathbf{n}_3 \\ \sum_{l=1}^{4} \left(-\tau_{H_{aa},l}\mathbf{b}_2 - \tau_{H_{fe},l}\mathbf{h}_3^l \right) \end{bmatrix}$$

where $-m^B g\mathbf{n}_3$ is the force of gravitation.

Note that the ground forces are not shown explicitly in $\mathbf{F}^{<_0}$. They are instead included in the applied forces $\mathbf{F}^{<_l}$, where they act onto the lowest rigid body of the leg.

6.3.7. The dynamic differential equations

The dde can now be found by a projection as follows

$$\boldsymbol{\tau}_i^{<} \bullet \left(\dot{\mathbf{p}}^{<} - \mathbf{F}^{<}\right) = 0,\ i \in [1, \text{degrees of freedom}] \tag{6.14}$$

where the left side is expanded according to the partitioning as before:

$$\boldsymbol{\tau}_i^{<} \bullet \left(\dot{\mathbf{p}}^{<} - \mathbf{F}^{<}\right) = \sum_{l=0}^{L} \boldsymbol{\tau}_i^{<_l} \bullet \left(\dot{\mathbf{p}}^{<_l} - \mathbf{F}^{<_l}\right). \tag{6.15}$$

Expressions grow quickly, and as an example, the single term corresponding to $l = 0$, $i = x$, i.e.

$$\boldsymbol{\tau}_x^{<_0} \bullet \left(\dot{\mathbf{p}}^{<_0} - \mathbf{F}^{<_0}\right)$$

expands to

$$\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{0} \end{bmatrix} \bullet \left(\begin{bmatrix} m^B b^T \dot{w}^B + m^B b^T \left(w^o \times w^B\right) \\ \mathbf{I}^B \cdot b^T \dot{w}^o + b^T w^o \times \left(\mathbf{I}^B \cdot b^T w^o\right) \end{bmatrix} - \begin{bmatrix} -m^B g\mathbf{n}_3 \\ \sum_{l=1}^{4} \left(-\tau_{H_{aa}}^l \mathbf{b}_2 - \tau_{H_{fe}}^l \mathbf{h}_3^l\right) \end{bmatrix} \right)$$

but this will fortunately reduce substantially when we apply the "fat" dot product, i.e.:

$$\begin{aligned} \boldsymbol{\tau}_x^{<_0} \bullet \left(\dot{\mathbf{p}}^{<_0} - \mathbf{R}^{<_0}\right) &= \mathbf{b}_1 \cdot \left(m^B b^T \dot{w}^B + m^B b^T \left(w^o \times u^B\right) - \left(-m^B g\mathbf{n}_3\right)\right) \\ &= m^B \left(\dot{w}_x^B + w_{pitch} w_z - w_{yaw} w_y + g\mathbf{b}_1 \cdot \mathbf{n}_3\right). \end{aligned}$$

This is still only one term of the sum (6.15) and it should be obvious why using a computer algebra tool is useful.

The result of (6.14), together with the kinematic equations (6.10) finally give us the system of differential equations.

6.4. Discussion and details of derivation

This chapter showed how the differential equations describing the rigid body model of a legged robot can be derived. Since the expressions become big and complicated, the computer algebra system (cas) Maple was used together with the Sophia-language by Lesser [106].

These derivations would have been very tedious to do by hand, but we would like to emphasize that a cas does not eliminate the need for manual derivations. One (subjective) reason is that we seem to gain some understanding from the system by working with the derivations manually on a high and abstract level. Another reason is that deriving an expression in two different ways, often produce expressions that appear to be very different. For instance, it is very easy to directly differentiate a vector $\mathbf{r}$ relative to a triad N, i.e.

$$\frac{{}^{N}\partial \mathbf{r}}{\partial t}$$

can be directly evaluated in Sophia/Maple. However, evaluating

$$\frac{{}^{B}\partial \mathbf{r}}{\partial t} + \boldsymbol{\omega}^{B} \times \mathbf{r}$$

typically produced more a compact expression in our derivation of the rbm. The expressions are algebraically equal of course, but Maple may not be able to simplify the first expression as much as the second.

The choice of generalized coordinates and speeds also affects the size of the expressions. For the coordinate corresponding to the knee angle, expressions are drastically reduced by choosing the rotation angle between triads h^l and s^l, rather than between t^l and s^l. These alternatives are shown at the bottom of figure 6.3, where $q^*_{H_{fe},l}$ and $q^*_{K_{fe},l}$ denote the corresponding relative angles. One reason for the difference in size is that expressions such as $\sin(q_1 + q_2)$ are often expanded into $\sin q_1 \cos q_2 + \cos q_1 \sin q_2$ during symbolic evaluation and it is difficult for the cas to find the reverse simplifcation. More importantly, differentiating the expression increases its size exponentially, i.e.

$$\frac{\partial \sin(q_1 + q_2)}{\partial t} = (\dot{q}_1 + \dot{q}_2)\sin(q_1 + q_2) \quad \text{whereas} \quad \frac{\partial \sin q}{\partial t} = \dot{q}\sin q.$$

One drawback with this coordinates choice is that sensors typically measure the knee's relative angle, i.e. the angle between the thigh's extension and the shank. It is therefore necessary to perform a substution when using relative values, i.e.

$$\begin{aligned} q_{H_{fe,l}} &= q^*_{H_{fe},l} \\ q_{K_{fe},l} &= q^*_{K_{fe},l} - q^*_{H_{fe},l} \end{aligned}$$

This substitution is also important to perform before deriving the Jacobians used in the leg controllers (see equation 5.1 on page 134), since the knee actuators apply torque between the thigh and the shank.

The fact that the legs are structually identical was also taken advantage of. This means that once $\mathbf{v}^{<1}$ in equation 6.6 has been derived, the parameters and coordinates in $\mathbf{v}^{<1}$ specific to leg 1 can be substituded for the parameters and coordinates specific to leg 2, thereby creating $\mathbf{v}^{<2}$ through a simple substitution. Strictly speaking this is not necessary and the actual Maple/Sophia code can do the derivation using either substitutions or full derivations.

6.4.1. Details of derivation and generality

The Maple/Sophia implementation actually models the foot as a separate rigid body, and also allows a flexion/extension ankle to be included if so desired (one early foot prototype included a rotational ankle joint that could be used to sense the orientation of the ground).

The actual implemenation is done with L as the number of legs, which is set to $L = 4$ for WARP1, but it was also used with $L = 0, 1$ and 2 during the development and derivation. For instance, setting $L = 0$ simply produces the equations of motions for a rigid body, which are known and this can then be used to verify the result of the derivations. In a similar manner, and also to verify the results, the actual number of joints in the legs has also been varied from 0 to 4 in the derivation.

It also seems straight forward to add more joints and different joints (e.g. linear). However, with more joints, the scheme for naming points and symbols needs to be modified

6.4.2. A note about notation

It is quite annoying to work out a suitable notation for complex systems — you tend to run out of subscripts and superscripts... Here, an anthropomorphic approach was taken, where **S** was used for shank, **T** for thigh etc. However, if for instance each leg would have several more joints, a more general and indexed notation could have been used as follows:

$B_{l,j}$ denotes the j^{th} rigid body on leg l and it's centre of mass and B_0 for the trunk.

$R_{l,j}$ denotes the point of rotation of joint j on leg l.

This gets messy when referring to specific triads:

n denotes the triad fixed in the inertial frame, $n^T = [\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3]$.

$b^{l,j}$ denotes the triad fixed in the j^{th} body in leg l and b^0 for the trunk, $b^{l,jT} = \left[\mathbf{b}_1^{l,j}, \mathbf{b}_2^{l,j}, \mathbf{b}_3^{l,j}\right]$.

As a comparison, consider $\mathbf{r}^{HK_l}$ (**h**ip to **k**nee on leg l) expressed in components relative to a triad fixed in the thigh. In the notation used so far, this is written as:

$$\mathbf{r}^{HK_l} = {}^{H_l}r^{HK_lT}t^l = {}^{H_l}r_1^{HK_l}\,\mathbf{t}_1^l + {}^{H_l}r_2^{HK_l}\,\mathbf{t}_2^l + {}^{H_l}r_3^{HK_l}\,\mathbf{t}_3^l$$

whereas the more general notation could be written as follows:

$$\mathbf{r}^{R_{l,1}R_{l,3}} = {}^{B_{l,1}}r^{R_{l,1}R_{l,3}T}b^{l,1} = {}^{B_{l,1}}r_1^{R_{l,1}R_{i,3}}\,\mathbf{b}_1^{l,1} + {}^{B_{l,1}}r_2^{R_{l,1}R_{l,3}}\,\mathbf{b}_2^{l,1} + {}^{B_{l,1}}r_3^{R_{l,1}R_{l,3}}\,\mathbf{b}_3^{l,1}$$

where $R_{l,1}$ means hip joint, $R_{l,3}$ means the knee joint and $B_{l,1}$ means the thigh link.

6.4.3. Obtaining linearized equations of motion

The linearized equations of motions are interesting in order to study the system and also for control design. Using a cas on (6.10) and (6.14) it is straight forward to obtain linearized equations in a standard form such as

$$\dot{x} \;=\; \mathbf{A}x + \mathbf{B}u \tag{6.16}$$

It is a matter of performing the following steps on on (6.10) and (6.14):

- Replace references to external ground forces with the output of a model that actually calculates the forces based on q and w.
- Replace q and w with $q + q_0$ and $w + w_0$, where q_0 and w_0 are the points around which the linearization takes place.
- Perform the actual linearization with respect to q and w on (6.10) and (6.14), results in the following system of equations (with matrix sizes indicated for the case of WARP1):

$$\begin{aligned} \dot{q} &= [\mathbf{A}_{kde}]_{18\times 36}\begin{bmatrix} q \\ w \end{bmatrix}_{36\times 1} + [\mathbf{B}_{kde}]_{18\times 12}\,u + [h_{kde}]_{18\times 1} \\ [\mathbf{M}_{dde}]_{18\times 18}\,\dot{w} &= [\mathbf{A}_{dde}]_{18\times 36}\begin{bmatrix} q \\ w \end{bmatrix}_{36\times 1} + [\mathbf{B}_{dde}]_{18\times 12}\,u + [h_{dde}]_{18\times 1} \end{aligned}$$

  where h_{kde} and h_{dde} are the remaining "constants" from the linearization. When (q_0, w_0) corresponds to an equilibrium h_{kde} and h_{dde} vanish.

- The matrices can then be combined into

$$\begin{bmatrix} I & \\ & \mathbf{M}_{dde} \end{bmatrix} \begin{bmatrix} \dot{q} \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{kde} & \\ & \mathbf{A}_{dde} \end{bmatrix} \begin{bmatrix} q \\ w \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{kde} & \\ & \mathbf{B}_{dde} \end{bmatrix} u + \begin{bmatrix} h_{kde} \\ h_{dde} \end{bmatrix}$$

  to get (6.16) in an implicit form.

This method has been used to obtain a linearization of (6.10) and (6.14) together with a simple linear ground model. However, this is a "brute force" approach and the resulting matrices are difficult to interpret, perhaps with the exception of A_{kde},

$$A_{kde} = \begin{bmatrix} [A_{kde}]_{1\ldots6\times1\ldots6} & [\mathbf{0}]_{6\times12} \\ [\mathbf{0}]_{12\times6} & [\mathbf{I}]_{12\times12} \end{bmatrix}$$

and

$$[A_{kde}]_{1\ldots6\times1\ldots6} = \begin{bmatrix} 1 & & & -z_0 & y_0 \\ & 1 & z_0 & & -x_0 \\ & & 1 & -y_0 & x_0 \\ & & & \sin(q_{roll,0}) \,/\, \cos(q_{pitch,0}) & \cos(q_{roll,0}) \,/\, \cos(q_{pitch,0}) \\ & & & \cos(q_{roll,0}) & \sin(q_{roll,0}) \\ & & 1 & \tan(q_{pitch,0}) \sin(q_{roll,0}) & \tan(q_{pitch,0}) \cos(q_{roll,0}) \end{bmatrix},$$

where $q_0^B = [x_0 \; y_0 \; z_0]^T$ and $q_0^o = [q_{yaw,0} \; q_{pitch,0} \; q_{roll,0}]^T$. As a comparison of the size of the expressions, Maple derives the (6.10) and (6.14) in a few seconds, whereas the linearization takes about half a minute on a workstation[5]. A faster derivation, and perhaps also a more compact result, ought to be possible using the same kind of partitioning scheme as in the derivation of dde.

[5] Pentium IV, 1.4 GHz

Part III.

Complex systems and control design

Introduction to part III

This part will describe a method for control design, where different tools have been combined into a method. This method was developed over a relatively long period of time as a way to handle the complexity involved in working with the WARP1 robot system (chapter 5). First chapter 7 describes this method and then chapter 8 describes an example for a robot arm with three actuators, showing the computer algebra code that was used.

Chapter 7 is based on an article by Ridderström and Ingvast [159], but material has been added: partly to give a more detailed description, and partly because of more recent work. In particular, a large part of the method was primarily tailored for the development of WARP1, but it has now been extracted as a separate tool kit.

To give an example of what can be done with these tools, it took about four hours to apply this method to a simple robot arm (figure 8.1) where the following was done:

- A computer algebra system was used to analytically derive the following:
  - a rigid body model of the robot arm with three actuators.
  - a simple control expression for the robot arm.
  - an expression that is used to animate the robot arm during simulation.
- Export the above model and expressions to another tool where,
  - a simulation model was assembled, including the control and animation expression.
  - the robot was simulated and controlled, while it was animated at the same time.

In the case of WARP1, this would have been followed by automatically implementing the controller, running experiments and evaluating the results.

Acknowledgements This work was supported by the Swedish Foundation for Strategic Research [176] through the Centre for Autonomous Systems [18] at the Royal Institute of Technology [100] in Stockholm. However, this work had not been possible without standing on the shoulders of others, i.e. by using specific tools such as the Sophia language by Prof. Martin Lesser with the `exmex()` procedure by Dr. Anders Lennartsson and Dr. Jesper Adolfsson. I would also like to thank Johan Ingvast for collaborating with me in this, and especially in the design of the class `ExpData`.

6.

7. Combining control design tools

Good tools are needed in order to develop robotic systems efficiently. Today, in addition to CAD/CAM, there are tools for model derivation, control design and implementation. There are also tools for exporting models to a control design environment, as well as from control design to implementation (rapid prototyping tools). It is however, still difficult to combine these tools, especially when working with large systems (i.e. with lots of states, signals and parameters). We have therefore combined, interfaced and augmented some of these tools into a method that bridges the gaps between analytic model derivation and control implementation, with special support for handling large system.

In the method, analytically derived functions are used for control design, simulation, visualization and evaluation, as well as for implementation. The method and tools have been used with the robot WARP1 (chapter 5), and also tested in simulation on two robot arms.

7.1. Introduction

When developing large and complicated mechanisms that are to be controlled, i.e. robots, there is a need for tools that can aid the designer. Figure 7.1 illustrates a control design process, where analysis precedes modeling and design. The next step is to simulate the robot's behaviour and perform experiments, which produce data that need to be visualized and evaluated. To aid us with this process, we would like to have tools that help us with tasks such as model derivation, simulation, evaluation/visualization and control implementation. Please note that the term controller in this chapter may also mean, or include the observer, i.e. we will write "controller" instead of "controller and observer".

Today, multibody systems can be simulated with graphical tools such as ADAMS [121], DADS [28] and Envision [36] that use numeric methods. However, there are

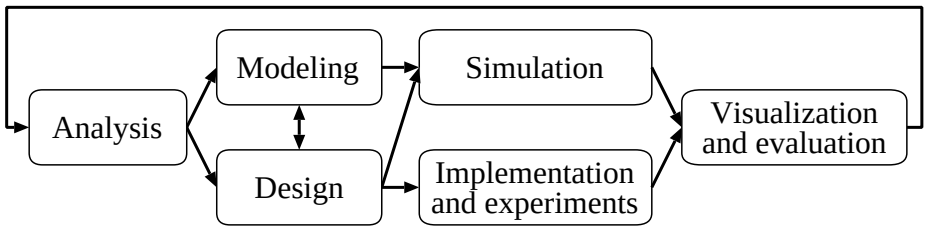


Figure 7.1: Illustration of control design process.

also computer algebra systems (cas[1]) that can be used to derive analytic (robot) models for analysis and/or (numeric) integration, i.e. simulation. One example is the combination of a general cas such as Maple [198], Mathematica [202] and Macsyma [113] together with the Sophia language [106]. Another is described by Murray et al. [126] for Mathematica.

In contrast to the CAD tools above, the analytic tools use textual input to describe the model. However, Hardell [63] has worked on using CAD models as input to Sophia programs. An advantage with analytic methods is the possibility to derive expressions representing forward kinematics and linearized models. These can then be used to design and test controllers in simulation using tools such as MATLAB/Simulink [179] and MATRIXx [85]. Finally, to automatically implement and test the controller, there are rapid prototyping tools such as dSPACE [37], WinCon [147], xPC Target [151] and OPAL-RT [133]. There are thus a lot of tools that aid the designer, but so far only the rapid prototyping tools have been well integrated. Furthermore, with all of these tools it is tedious to evaluate and and keep track of data from large and complicated systems.

The next section will disucss the four-legged robot WARP1 (chapter 5) as an example of a large and complicated system.

7.1.1. WARP1 — a complicated system

Consider figure 7.2 that shows a typical block diagram with a controller, observer and a rigid body model (rbm). Here, the rbm is of WARP1 (chapter 5) and it has 12 inputs, over 40 outputs, 36 internal states and it uses about 140 parameters.

The shaded area in the figure could for instance correspond to the “simple” controller described in section 5.3 that uses about 400 scalar parameters values. Of

[1]Since the Centre for Autonomous Systems is abbreviated CAS, the lower case form is used for computer algebra systems here.

Table 7.1: The number of operations required to evaluate some expressions used in WARP1's controller and for visualization/evluation. The last row shows the cost of evaluating the implicit differential equations of the rbm once.

Function	Add.	Mul./div.	Sin/cos	Note
${}^{B}v^{HP_1}$	25	46	6	Foot velocity relative to trunk
${}^{N}v^{HP_1}$	129	236	12	As ${}^{B}v^{HP_1}$, but expressed in an inertial coordinate system
${}^{N}r^{CM}$	1148	1794	30	Position of the robot's centre of mass in an inertial frame
rbm field	7068	7185	38	See caption above.

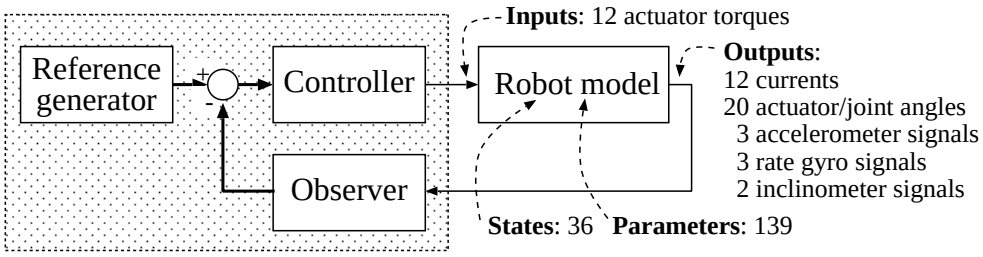


Figure 7.2: A typical block diagram illustrating an imagined controller and observer for WARP1. The blocks within the shaded area could correspond to the control structure in figure 5.3 on page 133. Major inputs and outputs of the robot model are also listed.

course, a lot of these parameters may be redundant and perhaps a quarter of the parameters might suffice if the four legs were assumed to have identical parameter values. On the other hand, most of the sensors need individual calibration data. In addition, requiring the parameters to be identical for the legs would cause problems in the future, e.g. if a single leg was modified in some way.

For WARP1, plotting signals and states is insufficient in order to understand the simulation results. Instead, animation of the robot is necessary, as well as being able to plot complicated functions of the data from simulations and experiments. Table 7.1 shows the cost[2] of evaluating some functions related to control, visualization/evaluation and the rbm.

[2]This is *after* Maple has attempted to minimize the cost by introducing intermediate variables.

7.1.2. Large and complicated systems

A few aspects of a large and complicated system in general are listed below:

- The robot model or controller has lots of: inputs; outputs; states; parameters; or internal signals[3]. A lot of signals and states may need to be logged.
- It is difficult to interpret the outputs and states directly. Evaluating and visualizing the system's behaviour requires large and complicated expressions.
- The controller needs functions that are described by large and complicated expressions.

Some of the problems are of a more "practical" nature because of the abundance of signals, states and parameters. It is difficult to manually keep track of them when working with the system, e.g. creating the simulation and controller. Specifically, it is difficult to provide the required signals and parameters in the correct order to control modules and models. It is also difficult to keep track of what signals are output, and in which order, as well as when accquiring logged data from simulations and experiments.

In fact, just having to do and take care of so many things causes problems, beacuse of all the small and trivial errors that pile up. Reducing and automating the designer's work is therefore important. It is also important to support a modular design strategy, i.e. allowing a divide-and-conqure strategy.

During our work with WARP1 we faced several of the these problems and have therefore automated parts of the design process in order to make it simpler and more efficient. In addition to combining and interfacing tools that are useful for specific tasks (e.g., using a cas for deriving and working with large and complicated expressions), we have created additional functions to handle some of the problems described above. One important principle has been to try and remove the need for doing things more than once:

- Specify numeric data and parameter values once, i.e. be able to refer to parameters by the same names and symbols in the different tools.
- Export derived expressions and models automatically rather than manually retyping expressions.
- Use the same control module for both simulation and experiment, avoid separate implementations.
- Create resuable modules (control modules, animation modules etc)

[3] Internal signals, e.g. the controller is modular with lots of signals between modules. In figure 7.2 this would be the signals between the blocks within the shaded area.

The tools and methods will now be described in detail.

7.2. Development tools and method

The development method is described by algorithm 2 and illustrated in figure 7.3. These are some of the more important tools of the method:

Maple is a well known computer algebra system (cas) [198].

MATLAB is an integrated environment that combines numeric computation with graphics and a high-level programming language [179]. Additional MATLAB tools are called toolboxes:

Simulink is a MATLAB toolbox that provides a simulation and prototyping environment for modeling, simulating and analyzing dynamic systems. It has a graphical interface for creating and working with block diagrams.

To import systems of differential equations into Simulink, so called System-functions (*S-functions)* are used. An S-function is a computer language description (in this case C code) of systems that can be continous, discrete or hybrid. Or simply a direct feedthrough system where the outputs only depend on the inputs, i.e. no internal states. Note that the S-functions have to be compiled before they can be used in Simulink.

Figure 7.4 illustrates how the data/information is passed between the different tools. Note that the same Simulink model file (denoted [.mdl] in the figure) is used for both simulations and experiments. This is possible since the robot is represented by an instance of a block in a Simulink library, where the library is either a simulation library or an experiment library. Simulation will use libraries that model the robot, and implementation will use different libraries with interfaces to the robot's hardware.

The use of the tools and methods will now be described in more detail.

7.2.1. Analytical derivation

Using the cas Maple with the Sophia language the rbm can be derived as in chapter 6. It is easy to work with vector objects in the Sophia language, since vectors contain both components and the name of the reference triad. This means that once the user has specified how different coordinate systems are related, vectors can be

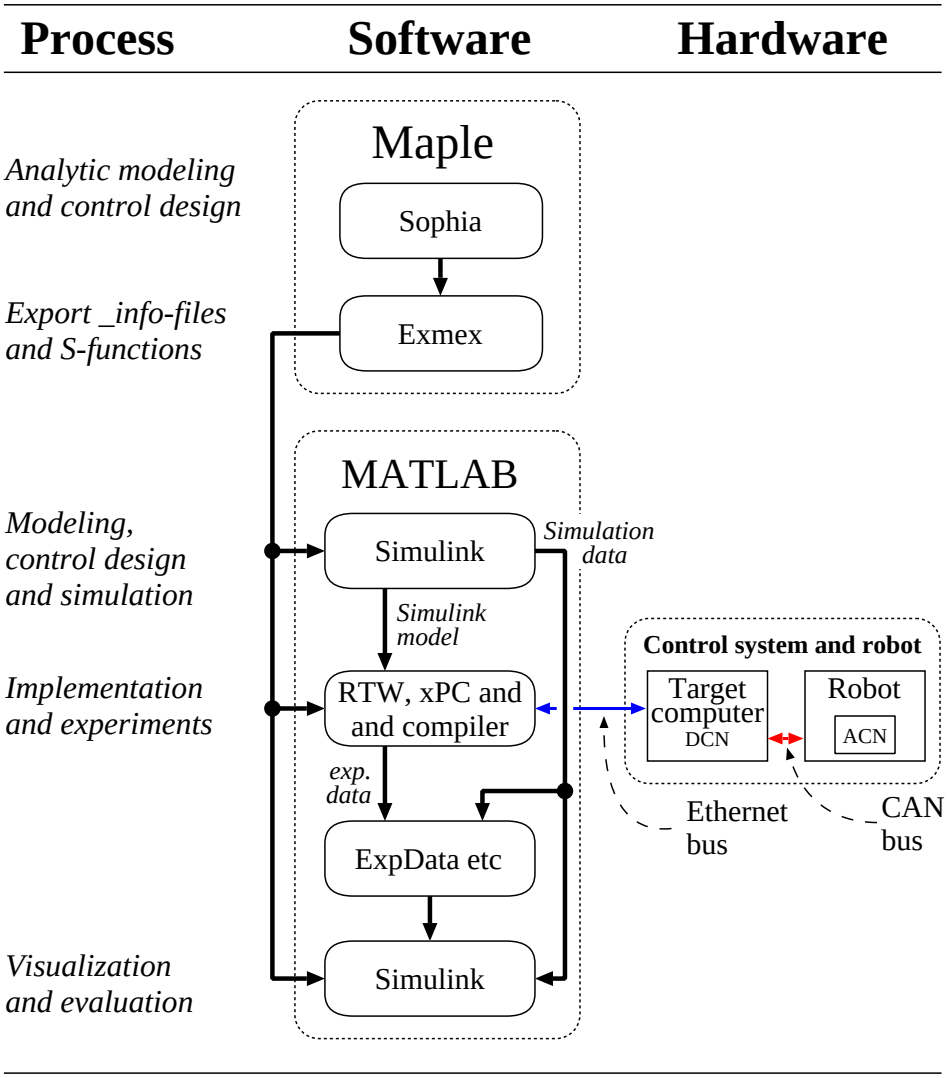


Figure 7.3: Maple and MATLAB/Simulink are used in the modeling and control design phase. Maple-expressions are exported to MATLAB/Simulink, where models and controllers are then integrated and tested in simulation. Finally, the MATLAB-toolboxes Real-Time Workshop and xPC Target are used to automatically build the controller.

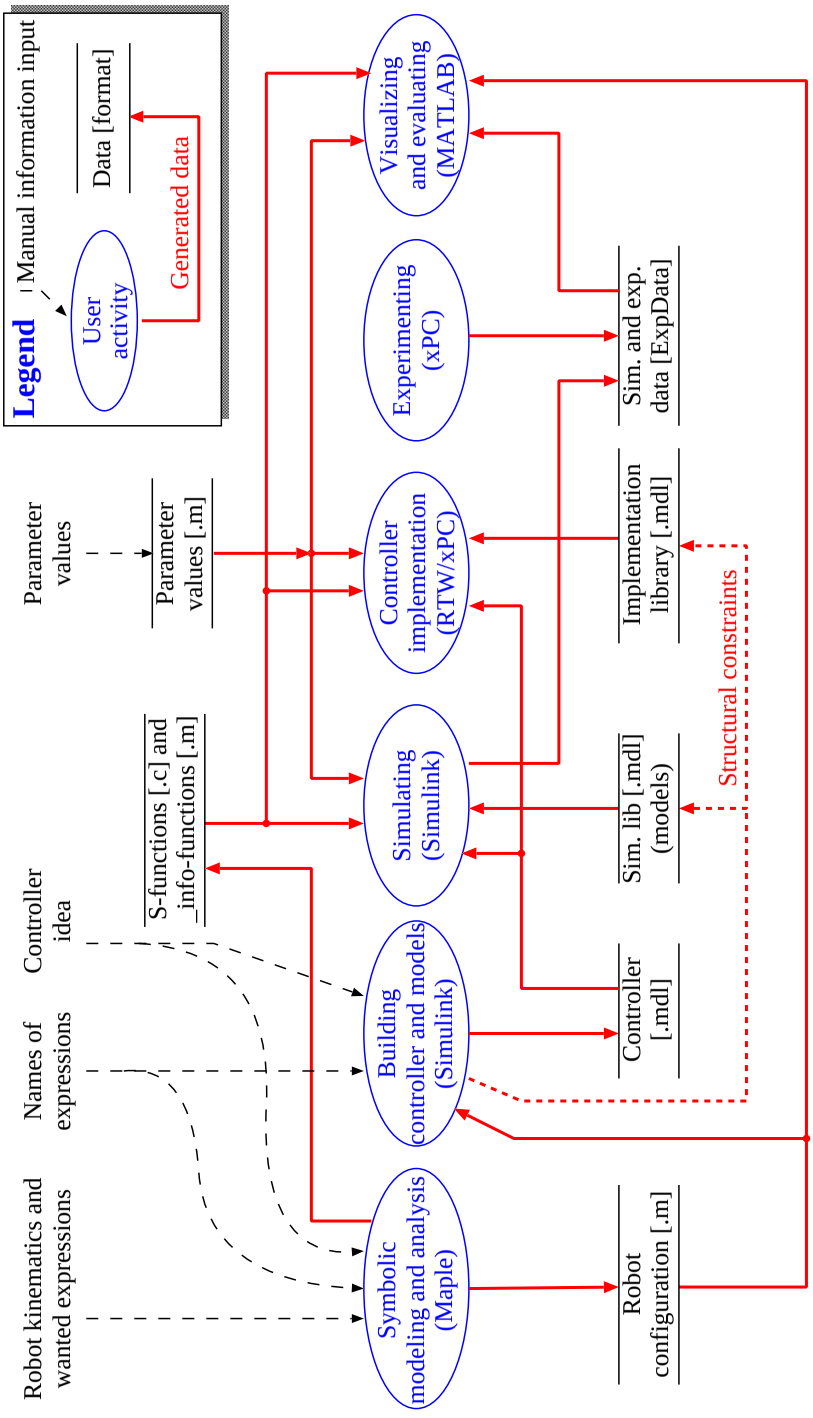


Figure 7.4: Manual user input and major dataflows in design process, details in text.

Algorithm 2 Development tools and method

1. Maple is used to derive and work with the large and complicated expressions.
   a) The Sophia language [106] is used to analytically derive the rbm and useful expressions such as forward kinematics and Jacobians.
   b) The `exmex()` procdure [105] is used to export expressions as S-functions.
   c) Signal and parameter information about the S-function is also exported.
2. A simulation model of the system is assembled in Simulink, where the controller is also created and simulations are performed to test it.
3. The controller is automatically implemented by using the MATLAB toolboxes Real-Time Workshop (RTW) and xPC Target (xPC).
   a) RTW generates controller C-code from a Simulink block diagram.
   b) The C-code is built with real-time kernel and drivers from xPC and then downloaded to the target computer (DCN in figure 5.4 on page 132).

   xPC also has an interface to run experiments and upload data from MATLAB.
4. Data produced from either simulations or experiments are evaluated and visualized in MATLAB/Simulink using additional expressions derived in the cas.

added with a simple operation, as illustrated below (figure 6.2 on page 152 shows the relationsship between triad N and A).

```
> #Define coordinate system relationship
  chainSimpRot([[N,A,3,alpha]]):
  r1 := N &ev [a,0,0]:  #Define vector r1
  r2 := A &ev [b,0,0]:  #Define vector r2
  r1 &++ r2;            #Add the vectors
```

$$[[a\cos\alpha + b, -a\sin\alpha, 0]\,, A]$$

As an example, it takes about 30 lines of Sophia code to derive the equations for a SCARA robot consisting of three rigid bodies, connected by one linear joint and two revolute joints (figure 7.7),

The code that derives the rbm of WARP1 is in principle similar to the script used for a SCARA robot. However, it is about ten times as large, partly due to the need to handle a variable (and therefore larger) structure, and partly to be user friendly by using “human like” names such as hip, thigh and knee for different parts of the legs.

7.2.2. Exporting models and expressions

The Maple procedure `exmex()` is used to export systems of ordinary differential equations and expressions as C-files in Simulink's S-function format. The procedure is a part of the Sophia package and is primarily written by Lennartsson [105], with major portions of the Simulink parts by Jesper Adolfsson.

Since the C-files are in the S-function format they will also work[4] with RTW and xPC. A drawback with `exmex()` is the lack of support for passing the required parameters to the S-function when used in Simulink. This could have been solved by acquiring the parameter values from the MATLAB environment and the parameters with their numeric values before exporting the function. However, this would mean loosing a lot of flexibility and in in the case of WARP1, each function would have to be created and exported separately for each leg.

A new Maple module was therefore created, that for each S-function also creates an auxiliary *information* function (a MATLAB .m-file, here denoted _info-function since *_info* is appended to the name of the S-function). This module, also called *MexFcn*, parses the expression that is to be exported in order to automatically determine the parameter names etc, before the original `exmex()` procedure is used to create the C-file. For example, say that we want to export the following expression:

$$u = u(q) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_1 (Q_1 - q_1) \\ k_2 (Q_2 - q_2) \end{bmatrix}$$

where $q = [q_1\, q_2]^T$ is the input, $u = [u_1\, u_2]^T$ is the output and the remaing symbols are parameters. This expression can be exported by the following Maple code:

```
> MexFcn:-New("name", u(q=[q1,q2])=[ k1*(Q1-q1),
                                     k2*(Q2-q2)],
                          , Simulink);
```

The newly created _info-function contains the information in the following list as well as some miscellanous information such as creation date etc:

- A list of the parameters used by the function (`k1`, `k2`, `Q1` and `Q2` in the example above)
- A list of inital values required by the function if it has internal states (there are no states in the example above)
- A list of inputs that the function needs (a signal called "`q`" with two components in the example above)

[4] To be picky, a trivial change of how integer and floating variables are defined was necessary before the S-functions worked with both Simulink and RTW.

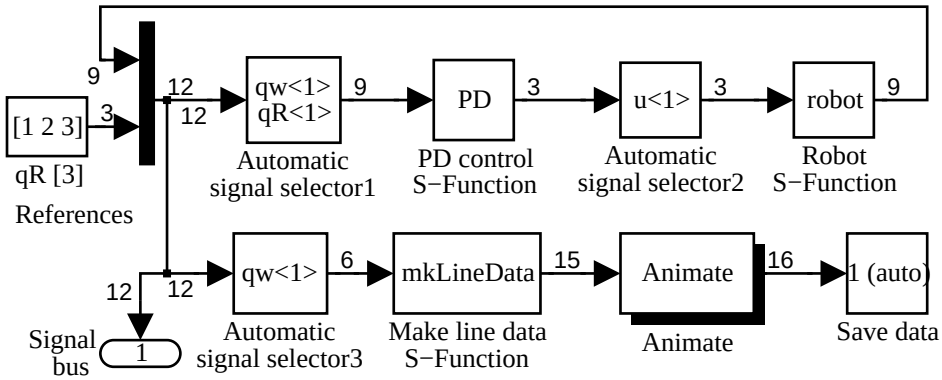


Figure 7.5: Example of a Simulink block diagram. In the upper row, there is a constant block, a multiplex block, two automatic signal selector blocks and two S-function blocks. The two S-functions are named *PD* and *robot*, and their corresponding C-files and _info-files are PD.c, robot.c, PD_info.m and robot_info.m. In the lower row, there is another S-function, *makeLineData*, that produces data for the Sub-System block called "Animate".

- A list of outputs of the function (a signal called "u" with two components in the example above)

The list of the parameter names that are required by the S-function can then be used to extract the values from some kind of database. The simplest way of doing this is to let the parameter names correspond to variable names in MATLAB's *base* workspace. A more intelligent way is to put the parameter values in a structure where the field names correspond to the parameter names. Note that both ways provide for (and assume) that the parameter names are identical in Maple and MATLAB. Access to the _info-function is done through an interface function, `getSFcnInfo()`.

7.2.3. Model and control assembly

The controller and models are put together in Simulink's graphical environment, where an exported S-function is included by adding a *S-Function block*. It is not actually the C-file that is used, but rather a compiled version of it. Figure 7.5 shows an example of an assembled Simulink block diagram model. The S-function blocks take as arguments the name of the S-function as well as values for any initial conditions and parameters. With the help of the _info-file, both the initial conditions

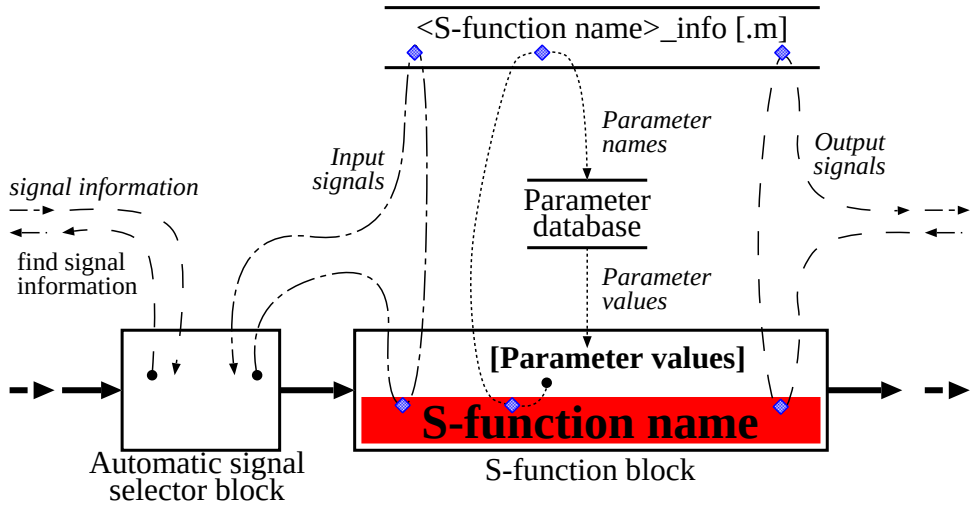


Figure 7.6: Flow of information using an _info-file. The flows start at a dot and information is extracted at the diamonds

and the parameters can be extracted from a parameter database as illustrated by the dotted line in figure 7.6.

Supplying S-functions with the correct signals can also be difficult, especially if the input consists of several signals. Thanks to the signal information from the _info-function, the *automatic signal selector* blocks in figure 7.5 handle this. The standard Simulink blocks could not be used for this purpose and we created these blocks together with a MATLAB function, `findSignals()`, that basically does the following (dash-dotted and dashed lines to the left in figure 7.6):

1. Get the S-function name from the S-function block.
2. Get the list of input signals that the S-function needs from the _info-function.
3. Get information about the signals coming into the the automatic signal selector block using `findSignals()`.
4. Select the required components.

Figure 7.6 also shows (dashed line to the right) the use of the _info-function when `findSignals()` finds an S-function and extracts information about output signals.

Remember that the block representing the robot (in the case of WARP1) is an instance from a library, either a simulation library or an implementation library. This is important since it allows switching between simulation and implementation

by just changing the library search path. Similarly, it makes it easy to use robot models of different complexity with the same controller. However, this requires the structures of the libraries to be identical.

For a complex system with several sub-systems that are structurally identical, it also speeds up the design by letting them be instances of the same reference system from a library. However, this also causes problems with keeping track of all the signals from the different sub-systems. Fortunately this is easily solved with signal selector blocks where they are given an argument saying which *occurence* of a signal to select. E.g., with several legs each sending out one signal with a specific name, the second occurence of that signal will be from the second leg connected to the bus.

If the control system is intended for distribution, it is also helpful to create busses similar to the expected real communication busses. Then it is possible to distribute the control system by first breaking the busses with communication blocks representing for instance a CAN bus and then implement the different parts of the controller on different target computers.

7.2.4. Visualization and evaluation

Simple 3D objects such as lines and plane surfaces are animated in Simulink to visualize the robot and environment (figure 7.7). Another tool, Envision/IGRIP[5], has been used for better graphics and more complicated environments. A drawback with that tool is that the user has to specify the robot model again. In contrast, the Simulink animation is based on data from functions derived in Maple that produce point positions (e.g. joint positions) that lines/surfaces are drawn between. This means that changing limb kinematics, or even the number of limbs, is automatically reflected.

Similarly, other functions that are useful for visualization and evaluation can also be derived for different number of legs and joints, automatically. Some examples are expressions that draw lines animating ground forces, joint torques and trunk velocities.

The large amount of signals can also result in a lot of logged data (either from simulation or experiments). If each signal is logged separately, the user has to keep track of a lot of variables. On the other hand, if the data is just lumped together, it becomes tedious and error prone for the user to keep track of what component of the data vector that corresponds to what signal. We therefore expanded the functionality that allows us to select the desired signal within a combined Simulink bus, to do the same from the command line. For instance, the commands

[5] Envision/IGRIP is actually an integrated environment for robot design, simulation and off-line programming, but was in this context only used for visualization and environment modeling.

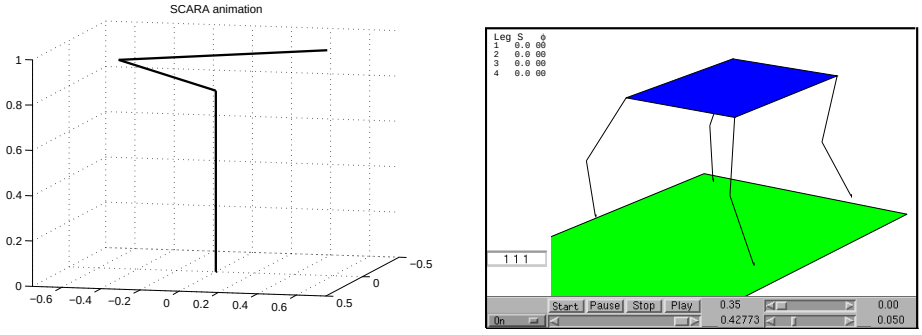


Figure 7.7: SCARA animation (left) and WARP1 animation (right).

```
simData = ExpData('trot', 0.01, [-10 0]);
expData = ExpData(target, 0.01, [-10 0]);
```

acquires the latest ten seconds of output data at a sample rate of 0.01 seconds from a simulation (first line) and a real experiment (second line). The argument *'trot'* refers to a Simulink model, whereas *target* refers to an xPC-object representing the target computer that has just tested the same control model against the real robot. Different signals are then easy to access as members of *simData* or *expData*, for example

```
time = expData.t;
```

7.2.5. Control implementation and hardware

RTW is used together with xPC to automatically generate the controller C-code and build it. Note that control expressions generated from the analytical model and used in the simulation are automatically included. In fact, we use the same Simulink model file for both simulation and implementation.

The resulting controller C-code, with a small real-time kernel, is then ready for execution on the target computer. The control program uses xPC's I/O libraries to communicate with I/O-cards in the target PC. Using communication cards (for CAN-busses etc) it becomes possible to work with distributed systems. In that case, only encapsulation blocks in Simulink have to be created that convert between Simulink signals and the bus communication protocol.

It is not necessary to use the xPC toolbox with this method. There are also other toolboxes (Real-Time Windows etc) and products (dSPACE etc) that would work because they all use the S-function format as an interface.

7.2.6. Experiments

Experiments are performed using xPC and Simulink's external simulation mode, i.e. without leaving the Simulink environment. This allows us to create a crude graphical "User Interface". Since the experiments are performed from Simulink, logging and retrieving data to MATLAB is easy.

The next section will present some results from using these models with the previously described tools.

7.3. Results and performance

The results and performance discussed below are based on our experience from working with WARP1, and numerical data are given for that case. A typical design process would now be as follows. First the robot model is derived analytically (in the case of WARP1 as described in section 6.3, using an automated script) and then exported to Simulink, where the model is tested[6] with local joint controllers. The next step is to test the joint controllers against the real robot, thereby verifying that both the method and the robot works properly. Then, the designer gets an idea for a controller (such as the one in section 5.3) and returns to the analytical environment to derive functions for the controller. These are also exported to Simulink where they are used to build a controller that is tested in simulation and then against the robot.

To give an example of how long it takes to evaluate a "large" expression, the position of the robot's centre of mass, ${}^{N}r^{CM}$ (see table 7.1) takes about $60\,\mu$s to evaluate on the target PC (a Pentium 350 MHz). It takes about 20 seconds to execute the script that derives these expressions and the rbm for WARP1 on a Sun Ultra/60. However, poorly chosen generalized coordinates and adding complicated constraints can increase this time significantly.

Several levels of detail are possible in the simulation, e.g. in the case of WARP1 the actuator models range from ideal torque sources to including models of motor current and rotor dynamics. Simulating one second of walking takes about four seconds on a Sun Ultra/60, when assuming "ideal actuators". Including actuator dynamics (electric and mechanic) approximately doubles the required time, and calculating all visualization and evaluation expressions during simulation increases the required time with about 25%.

[6] Further testing can be done by analytically reducing model complexity and study special cases, and for WARP1, by using the automated script to reduce the n:o legs and the n:o degrees of freedom per leg.

7.3.1. Simple controller again

The simple controller (5.1 on page 134) described in chapter 5 uses expressions ($^{B}r^{HP_l}$, $^{B}v^{HP_l}$ and J_l), that are all automatically generated. This means that in theory the leg controller should work even if the leg kinematics are changed in the analytical model. A mechanical designer could therefore easily test different ideas for leg kinematics (using an actual controller) with very little extra work. In practice, the real hardware would of course have to be changed similarly, and there are limits to how the kinematics could be changed. For instance, it is not clear what would happen if extra degrees of freedom were added, since this controller only uses feedback on the foot's position (not its orientation).

7.4. Discussion and summary

In this chapter we have shown how to combine and interface several tools to bridge the gap between automatic model derivation and control implementation. This method was used and developed while developing our four-legged robot. The automatic model derivation is capable of producing analytic models for entire classes of robots (not just a specific robot type). Furthermore, the designer is aided by being able to analytically derive, generate and use various expressions.

The rigid body model derived in Maple is not only used for simulation, e.g. in the case of WARP1, the kinematics is also used for control design. Yet another use is debugging, since in Maple, the analytic model can be investigated by reducing the equations through linearization or by looking at special cases.

It is very useful for the designer to be able to easily derive expressions from the model. In our application, we automatically derived and exported expressions that were used in the controller and for performance evaluation, but we also see the possibility to use this for mechanical design. The ability to easily generate and export expressions saves the designer a lot of time, compared to deriving them by hand and then manually implementing them in the controller or simulator. Additionally, changes in the robot's structure are automatically reflected in other tools for purposes such as visualization, simulation and expressions for control design.

The generality is unfortunately not supported in full by all tools. For instance, Simulink's graphical interface makes it difficult to design more generally applicable controllers. The problem of too many signals was handled by creating signal busses from which we used special methods to extract the desired components.

Another aspect is optimality with respect to speed and size. We have focused on doing the tools generic, not optimal. Since expressions are exported as C-code, optimization by Maple and the compiler(s) will affect simulation and implementa-

tion performance. Similarly, the performance of RTW and xPC's real-time kernel will also affect the implementation performance, i.e. the maximum frequency with which we can execute the controller. However, with a target PC using a Pentium 350 MHz CPU, calculating the control signal from the controller in section 5.3 only takes about 140 μs.

One improvement to this chain of tools would be a closer connection to other CAD tools, in order to extract parameter values and perhaps also the robot structure. Another improvement would be adding specific support for standard control methods such as linear state feedback controllers.

In this paper, we worked with a four-limbed robot and a simulated SCARA robot, but we believe this method can easily be extended to other structures. We found the method to be useful and believe it has advantages over other tools for robot simulation, control and implementation. Furthermore, we are convinced that these tools and methods will be very valuable for future work, validating models and developing more advanced controllers.

To summarize the method, it uses Maple/Sophia to derive models and expressions for analysis, control design and simulation in MATLAB/Simulink. RTW and xPC are then used to implement the controller and perform experiments. Note that no low-level coding is necessary, once the tools have been integrated and combined. As a final example, we had an idea for a simple trunk attitude controller that distributes desired vertical leg forces based on the estimated attitude. Reusing parts from the simple controller previously described, it took less than 90 minutes to go from idea to simulation and perform the first experiments (see chapter 11).

8. Maple/Sophia/MexFcn example

This chapter shows an example of Maple/Sophia code and an auxiliary information file. The purpose is to demonstrate the use of Maple, Sophia and the MexFcn module, as well as give a concrete example of the information exported to MATLAB/Simulink. This example is based on a real robot arm (figure 8.1) and table 8.1 shows the amount of Maple/Sophia code needed for different parts of the first two steps in the list below. In total, it took about four hours to do the following:

1. Use Maple/Sophia code to derive:
   - a rigid body model of the robot arm.
   - a simple control expression for the robot arm.
   - an expression that is used to animate the robot arm during simulation.
2. Export the rbm and expressions to MATLAB/Simulink.
3. Assemble a simulation model based on the exported rbm and the expressions for control and animation.
4. Run simulations where the robot is controlled and animated.

The focus of this example is on the use of the computer algebra system and exporting the expressions, not on assembly of the simulation model (nor on implementation). Figure 7.5 on page 180 in the previous chapter actually shows a Simulink block diagram based on this example, where the original hierarchical structure of the block diagram has been collapsed to produce a smaller figure.

Table 8.1: Lines of Maple code and n:o statements used in the robot example.

Part of Maple/Sophia code	Lines	Statements
Initialization, i.e. load Sophia, MexFcn etc	3	4
Define kinematics and derive the dynamic equations	36	30
Define signals, i.e. names and component symbols	3	3
Export three S-Functions	15	11

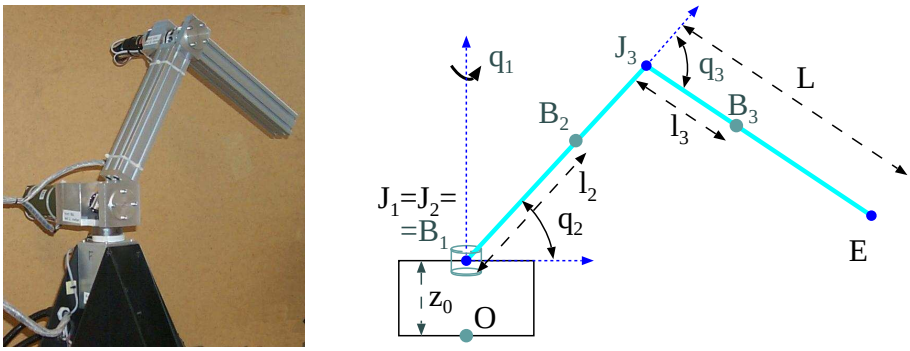


Figure 8.1: Photo of the robot arm that this example is based on, and illustration of the rigid body model.

8.1. Description of the robot arm

Figure 8.1 illustrates the robot's rbm that is comprised of three rigid bodies, B_1, B_2 and B_3. The robot's base is fixed and all three joints are rotational, where the first and second joint (counting from from the base) coincide spatially. The points are denoted as follows:

- B_1, B_2 and B_3 denote the centres of mass of the three bodies B_1, B_2 and B_3 respectively.
- J_1, J_2, and J_3 denote joints 1, 2 and 3 respectively of the arm, i.e. the axis of rotation intersect the point.
- E denotes the endpoint of the arm.

The following standard reference triads will be used:

- $f0$ is a triad fixed in the inertial system, where its third axis is in the opposite direction of gravity.
- $f1$, $f2$ and $f3$ are three triads fixed in the bodies B_1, B_2 and B_3 respectively.

The triads and the generalized coordinates q_1, q_2 and q_3 are defined through a series of simple rotations as follows:

- The triad $f1$ is the triad $f0$ rotated q_1 radians around $f0$'s third axis (vertical). $f1$'s first vector is parallel to the horizontal projection of the arm.

- The triad $f2$ is the triad $f1$ rotated q_2 radians around $f1$'s second axis. $f2$'s first vector is parallel to $\overline{J_2 J_3}$, i.e. from the second joint to the third joint.
- The triad $f3$ is the triad $f2$ rotated q_3 radians around $f2$'s second axis. $f3$'s first vector is parallel to $\overline{J_3 E}$, i.e. from the third joint to the arm endpoint.

8.2. The example code

The commented code for this example will now be shown. This code clears memory and loads software modules (e.g. MexFcn) and packages (Sophia and `exmex()`).

```
> restart:  # Clear memory
  cat(getenv("CAS"),"/share/maple/util/OS_Tools.mpl"):  read(%):
  OS_Tools:-Read("ChrTools","MexFcn","SophiaExtras");
```

8.2.1. Define kinematics and derive dynamic equations

Declare (to Sophia) that the generalized coordinates and generalized velocities depend on time.

```
> dependsTime(q1, q2, q3, w1, w2, w3):
```

Define the kinematic differential equations, kde:

```
> kdeL := [q1t = w1, q2t = w2, q3t = w3]:
```

Define the rotational relationsship between the triads.

```
> chainSimpRot([ [f0, f1, 3, q1],
                 [f1, f2, 2, q2],
                 [f2, f3, 2, q3] ]):
```

Define position vectors from the origin of the base to different points:

```
> r0  := f0 &ev [0,0,0]:           # The origin..
  # Joint 1 is a distance 'z0' above the origin
  rJ1 := r0 &++ (f0 &ev [0,0,z0]):
  rJ2 := rJ1:  # Joint 2 coincides with joint 1
  rJ3 := rJ2 &++ (f2 &ev [L,0,0]):  # Joint 3
  rE  := rJ3 &++ (f3 &ev [L,0,0]):  # Endpoint of arm
  rB1 := rJ1:  # B1 coincides with joint 1
  rB2 := rJ1 &++ (f2 &ev [l2,0,0]):
  rB3 := rJ2 &++ (f3 &ev [l3,0,0]):
```

Derive the velocities of the points relative to the inertial frame, and substitute `q1t`

with w1, i.e. use the kde. Then simplify the result.

```
> for P in [J1, J2, J3, E, B1, B2, B3] do
      v||P := &simp subs(kdeL, f0 &fdt r||P);
  end do:
```

Create the system velocity vector, vK, by collecting the velocities of the centres of mass and the angular velocities of the rigid bodies into a Kvector. The Sophia operator &aV is used to extract the angular velocity between to triads.

```
> vK := &Ksimp subs(kdeL, &KM[vB1, f0 &aV f1,   # Vel. for body 1
                             vB2, f0 &aV f2,   # Vel. for body 2
                             vB3, f0 &aV f3]):# Vel. for body 3
```

Find the tangent vectors from the system velocity vector. Sophia has a function that does this, using the fact that the system velocity vector can be written as an affine function of the generalized speeds, w1, w2 and w3.

```
> beta := KMtangents(vK, [w1, w2, w3]):
```

Define dyad operators representing inertia of the bodies with respect to their respective centre of mass and triad:

```
> I1 := EinertiaDyad(I1x, I1y, I1z, 0, 0, 0, f1):
  I2 := EinertiaDyad(I2x, I2y, I2z, 0, 0, 0, f2):
  I3 := EinertiaDyad(I3x, I3y, I3z, 0, 0, 0, f3):
```

Define a system vector with momentums, pK, and derive the time differential of the momentums, pKt. The Sophia operators &** and &o are used for scalar multiplications and inner products.

```
> pK :=  &Ksimp(subs(kdeL, &KM[m1 &** vB1, I1 &o (f0 &aV f1),
                               m2 &** vB2, I2 &o (f0 &aV f2),
                               m3 &** vB3, I3 &o (f0 &aV f3)])):
  pKt := &Ksimp(subs(kdeL, f0 &Kfdt pK )):
```

Define applied external forces and moments, and then assemble them into a Kvector. The masses of the bodies B1, B2 and B3 are denoted m1, m2 and m3 respectively, while g denotes the constant of gravity. The torques applied by the actuators at the joints are denoted u1, u2 and u3 respectively. Note that if the actuator at joint three applies a torque of u3 onto body B3, then a torque -u3 will be applied to body B2.

```
> F1 :=  f0 &ev [0,0,-m1*g]:  # Force onto body 1
  F2 :=  f0 &ev [0,0,-m2*g]:  # Force onto body 2
  F3 :=  f0 &ev [0,0,-m3*g]:  # Force onto body 3
  # Moments onto bodies 1, 2 and 3
  M1 := (f1 &ev [0,0,u1]) &- (f2 &ev [0,u2,0]):
  M2 := (f2 &ev [0,u2,0]) &- (f3 &ev [0,u3,0]):
  M3 :=  f3 &ev [0,u3,0]:
  # Assemble forces and moments into a Kvector
  F := &KM [F1, M1, F2, M2, F3, M3]:
```

Derive the dynamic differential equations by projecting the difference of the applied forces and momentum differential onto the tangent vectors. Sophia has an operator, `&kane`, which does this for us. The system of equations will be in an implicit form that can then be solved for the accelerations, i.e. solving for `w1t`, `w2t` and `w3t`.

```
> # List of implicit equations
  ddeImplicitL := simplify(beta &kane (F &-- pKt)):
  # Make sorted list of explicit equations
  ddeExplicit := solve(convert(ddeImplicitL,set),{w1t,w2t,w3t}):
  ddeExplicitL := ChrTools:-SortEq(ddeExplicit,lhs):
```

8.2.2. Define commonly used signals

Define signals by specifying their names and the symbols that comprise them.

```
> Signals[qw]:= qw=[q1,q2,q3,w1,w2,w3]:  # States
  Signals[qR]:= qR=[qR1,qR2,qR3]:  # References
  Signals[u] :=  u=[u1,u2,u3]:  # Control signal
```

8.2.3. Export robot model

The rbm will now be represented by a *MexFcn-object*, that is then exported as an S-function (robot.c) and an information file (robot_info.m, see section 8.3). First a new object is created

```
> arm := MexFcn:-New():  # Create new MexFcn-object
```

Then the system of differential equations, as well as the definition of the required input signals are assigned to the object

```
> arm:-systems := [[kdeL], [ddeExplicitL]]:
  arm:-inputs := Signals[u]:  # Input signals
```

and finally the output of the S-Function is defined

```
> arm:-outputs := [Signals[qw], "rE"=&simp subs(kdeL, f0 &to rE)]:
```

This is enough information to automatically determine what parameters this S-function will require, which is done by the method `arm:-Export()` when it creates the information file and uses `exmex()` to create robot.c.

```
> arm:-Export("mex/robot", Simulink):
```

8.2.4. Export animation function

The S-function, mkLineData.c, is also exported. It outputs the coordinates of points in the robot arm that will be used to animate the robot during simulation.

```
> mkLineData := MexFcn:-New(inputs = Signals[qw]):
  mkLineData:-outputs :=
    LineData=map(x->f0 &to x, [r0, rJ1, rJ2, rJ3, rE]):
  mkLineData:-Export("mex/mkLineData", Simulink):
```

8.2.5. Export simple PD-controller

Define and export a simple PD controller for the joints.

```
> PD := MexFcn:-New(inputs = (Signals[qw], Signals[qR]) ):
  PD:-outputs := [ u = [ k1*(qR1-q1) + d1*(0-w1),
                         k2*(qR2-q2) + d2*(0-w2),
                         k3*(qR3-q3) + d3*(0-w3) ] ]:
  MexFcn:-Export("mex/PD", Simulink);
```

8.2.6. Exporting to MATLAB

The MexFcn-objects can also be exported for direct use in MATLAB, e.g. from the MATLAB prompt. This example creates and exports a simple function in one line:

```
> MexFcn:-Export("mex/f", y(x)=sin(x+phi), MATLAB):
```

In this case, another function, `exmat()`, is used instead of `exmex()` for creating the C-file. The information file is still used and allows the user to easily create objects in the MATLAB environment that can be used as normal functions.

```
>>Par = struct('phi', 0.3); % Structure with parameters
>>f = MexFcn('mex/f', Par); % Create MATLAB MexFcn-object
>>f(1.2)
ans =
         0.9975
```

8.3. Information file of the robot model

The information file, robot_info.m, looks like this (slightly trimmed):

```
% ROBOT_INFO    Return S-function information for robot
% Filename:     mex/robot_info.m
% Created at:   2003-02-14, 19:08:56
% By:           chr
% Stx:  y = robot_info(type, S, index)
```

```
% Fcn:  Return S-function information depending on 'type'
% In:   type    = One of the following:
%  'p'   - List of Names of function parameters
%  'ic'  - List of names of initial condition values
%  'i'   - String with names of input signals
%  'o'   - String with names and widths of output signals
%  'h'   - List of strings describing the function
%  'x'   - String listing the states of the S-function
%  'xt'  - String listing the state derivatives
%
function y = robot_info(type)
  switch type
   case 'p',            % Names of parameter
    y = {'I1z' 'I2x' 'I2y' 'I2z' 'I3x' 'I3y' 'I3z' ...
         'L' 'g' 'l2' 'l3' 'm2' 'm3' 'z0'};

   case 'ic',           % Names of initial conditions
    y = {'q1_0' 'q2_0' 'q3_0' 'w1_0' 'w2_0' 'w3_0'};

   case 'i',            % Names of input signals
    y = 'u';

   case 'o',            % Names and widths of outputs
    y = 'qw 6 rE 3';

   case 'x',            % String with state names
    y = {'q1' 'q2' 'q3' 'w1' 'w2' 'w3'};

   case 'xt',           % String with state derivatives
    y = {'q1t' 'q2t' 'q3t' 'w1t' 'w2t' 'w3t'};

   case 'h',            % Help text and description
    y = { ...
  'robot was automatically exported from Maple' ...
  'Inputs:     u(1:3) = [u1, u2, u3]' ...
  'Parameters: I1z I2x I2y I2z I3x I3y I3z L g l2 ...'...
  'Outputs:    qw(1:6) = [q1, q2, q3, w1, w2, w3]' ...
  '            rE(1:3) = [cos(q1)*L*(cos(q2)+cos(q...'...
  'Unused inputs: u1 u2 u3' ...
  'States:     q1 q2 q3 w1 w2 w3' ...
  'I.C.:       q1_0 q2_0 q3_0 w1_0 w2_0 w3_0' ...
  'State der.: q1t q2t q3t w1t w2t w3t' ...
  'Sys. of de: [q1t = w1, q2t = w2, q3t = w3]' ...
  '            [w1t = (u1-4*sin(q3)*cos(q2)^2*co...'};

   otherwise, error('Unknown argument');
  end
```

8. Maple/Sophia/MexFcn example

Part IV.

Stability of statically balanced robots with compliance

Introduction to part IV

Part IV studies the situation of statically balanced robots when compliance is also considered. Typically, the static balance criterion only considers that the centre of mass is projected within the support area. In chapter 9 it is shown that when the combined system is not stiff enough, a so called “statically stable” stance is actually unstable. The chapter studies a model corresponding to a planar symmetric robot standing on a compliant surface. A criterion for when it is stable is derived analytically for this case, and also experimentally verified. Chapter 10 then extends this criterion to radially symmetric configurations and further analyzes the asymmetric planar configuration.

In chapter the ground is no longer considered to be the cause of compliance. Instead, it originates with the robot controller and two controllers are discussed: a Cartesian position control of the feet; and a posture controller that uses force control in the vertical direction. Both of controllers are analyzed for a planar robot model in symmetric stance on stiff ground. The posture controller has been tested on WARP1 and some experimental results from this are also included for completeness. Finally, this part is concluded with a summary and discussion in chapter 12.

Chapter 9 and 10 are partly based on work by Ridderström published in [158] and [157]. The chapters also contain new results such as an analysis of the asymmetric planar case. The posture controller in chapter has been presented in a paper by Ridderström and Ingvast [160].

Acknowledgements This work was supported by the Swedish Foundation for Strategic Research [176] through the Centre for Autonomous Systems [18] at the Royal Institute of Technology [100] in Stockholm.

I would also like to thank Takafumi Kinoshita and Andreas Archenti for their help with the experiments, and Johan Ingvast and Chandana Paul for their valuable input.

8.

9. A statically balanced planar stance on a compliant surface

A legged robot is said to be *statically balanced* (or *statically stable*), if a line from its centre of mass in the direction of gravity passes through the support region (see chapter 2.4 on page 38 for details). However, this does not always guarantee stability. In fact, even when standing still with all joints locked and statically balanced, it can still fall over! All it takes is a sufficiently compliant surface. An example of this will be shown in this chapter, where in a planar and symmetric case, the robot would fall over if

$$\frac{mg}{2k} > \frac{r^2}{h} \tag{9.1}$$

where mg is the weight, $2k$ is the vertical stiffness from the two legs, r is half the width of the support region and h is the vertical distance from the feet to the centre of mass. (A geometrical note — the left side of this inequality corresponds to the distance that the robot has sunk into the soft surface if a linearly elastic surface is assumed.) This condition does not contradict McGhee and Frank's [117] original definition of static stability, since that assumes a *stationary horizontal support surface* — clearly not the case when standing on a compliant surface. Other stability criteria have been suggested since then. Messuri and Klein [123] defined the *energy stability margin* (ESM), as the minimum energy needed to tip over a (rigid) vehicle. Nagy et al. [129] later defined both the *rigid stance stability measure* and the *compliant stance stability measure* (CSSM), where the former is equivalent to ESM. The CSSM is one of the few measures for stability on compliant terrain. More recent results on energy measures have been produced by Yoneda and Hirose [74, 75, 206].

This criterion (9.1) is valid for a static balance strategy, where motions are planned so that the projection of the centre of mass remains within the support area (see sections 3.1.1 and 3.3 for examples). In order to use this kind of result for other balance methods, it would at least have to be extended to include some kind of

equivalence of "control stiffness". A commonly used strategy for bipeds is based on the *zero-moment point*, first introduced by Vukobratović and Stepanenko [194, 195]. It can be thought of as a dynamic version of the static balance criterion that includes the inertia of the robot.

Gao and Song [45] actually worked on force distribution, but their definition of a *leg stiffness matrix*, shows how the compliance due to the ground can be lumped together with the compliance due to the leg's structure and actuators. They note that the stiffness matrix is actually a function of the foot's position with respect to the trunk. One can think of the stiffness matrix as a generalization of two springs connected in series. It should be possible to include compliance from the controller in a similar manner. However, there are differences between ground compliance, and compliance that depend on the robot's orientation.

9.1. Models

The robot model used to study stability needs to include compliance. This compliance can come from the ground (soft terrain), the feet, compliance in the mechanical structure (legs and trunk) as well as from the actuators and transmission. For example, in our quadruped robot WARP1 (figure 9.1a), there is definitely compliance in the feet (rubber pads) and in the transmission system based on steel wires. The robot weighs about 60 kg and the feet are small, so on soft terrain it would sink down. See chapter 5 for a detailed description of the robot.

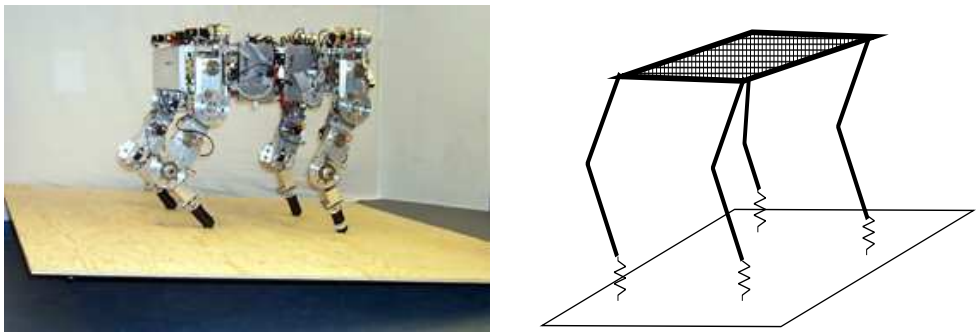


Figure 9.1: a) WARP1 on a balance board. b) Ideal legged locomotion machine on a horizontal compliant surface.

A model of such a system easily becomes very complex. Kinematic and dynamic differential equations for WARP1 have been derived (chapter 6) in a symbolic

form, but the size and complexity makes analysis of these equations difficult. Consider instead as a first step the model shown in figure 9.1, with massless legs and foot contacts that are approximated as point contacts. Furthermore, all compliance is considered lumped together as soft terrain. However, even that model becomes complicated and a planar approximation will be used first. One motivation for this planar model is to imagine a four-legged robot standing with the legs close together in the lateral direction, but with a much longer distance between the front and rear legs. Then, looking at the imagined robot from the front, it is approximated with the model shown in figure 9.2.

A stability analysis of the planar model (figure 9.2) is then performed symbolically using computer assisted algebra tools in a standard manner as follows:

- Derive differential equations for the model of the mechanical system.
- Determine equilibria.
- Linearize around the equilibria.
- Determine eigenvalues and analyze them to determine stability.

9.1.1. A planar model

Even in the case of a planar model, there are several choices including

- inverted pendulum with torque spring
- with or without horizontal springs
- vertical spring only

Here the third model is discussed, but the others have also been analyzed to varying extents, producing similar results.

The planar model (figure 9.2) is assumed to be rigid and standing on two identical vertical springs and dampers (stiffness=k, damping=d) at the points C_1 and C_2. The feet have sunk the distances δ_1 and δ_2 into the ground. The width of the base is $2r$, and the parameter γ determines the relative position of the point A on the base line $\overline{C_1C_2}$. When $\gamma = 0$, the configuration is symmetric. The point B represents the system's centre of mass, located a distance h above the base line. To simplify understanding of the geometry, the angle α is included. Here, we will consider the case when $\gamma = 0$.

The generalized coordinates are δ_1, δ_2 and q_1 where q_1 is the horizontal position of the centre of mass. Since only vertical forces play any role in this model, q_1

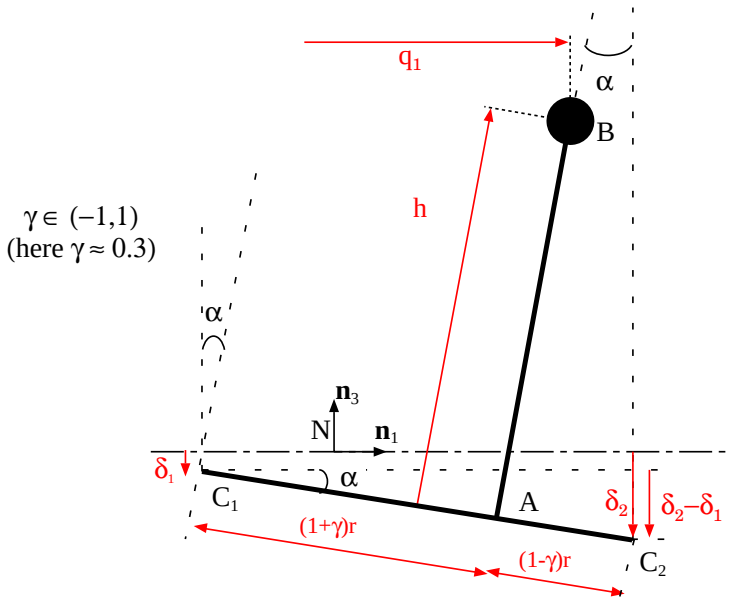


Figure 9.2: Planar model of robot standing on horizontal soft terrain. The dash-dotted line represents the ground, into which the left and right feet have sunk the distances δ_1 and δ_2 respectively.

will be irrelevant in the final equations. The computer assisted algebra software Sophia *[106]* with Maple [198] were used in the derivations (see chapter 7). Only the results are included to save space.

The generalized speeds were chosen as follows

$$\begin{aligned} w_1 &= \boldsymbol{\omega} \cdot \mathbf{n}_2 \\ w_2 &= \frac{{}^N\partial \mathbf{r}^{NB}}{\partial t} \cdot \mathbf{n}_3 \\ w_3 &= \frac{{}^N\partial \mathbf{r}^{NB}}{\partial t} \cdot \mathbf{n}_1 \end{aligned}$$

i.e. w_1 is the angular velocity, w_2 is the vertical velocity of the centre of mass and w_3 is the horizontal velocity. To derive the kinematic differential equations, these equations were simply solved for $\dot{\delta}_1$, $\dot{\delta}_2$ and $\dot{q}_1$.

The dynamic equations are not shown because they are large, complicated and would not contribute anything. The equilibrium points can be found by solving the following equations (obtained through the dynamic equations as usual, i.e. setting

all velocities and accelerations to zero). To simplify the equations, the following coordinates are introduced:

$$\begin{aligned} \delta &= \frac{\delta_2 - \delta_1}{2r} \\ \overline{\delta} &= \frac{\delta_2 + \delta_1}{2r} \end{aligned}$$

where δ corresponds to the difference in displacement and $\overline{\delta}$ to the average displacement. In these coordinates, the equilibrium equations can be written as follows

$$\begin{aligned} mg &= 2k\overline{\delta}r \\ h\overline{\delta}\delta &= \sqrt{1-\delta^2}\left(\delta - \gamma\overline{\delta}\right) r \end{aligned}$$

which have the following solutions when γ=0

$$\begin{aligned} \overline{\delta} &= \frac{mg}{2kr} = \frac{ar}{h} \\ \delta &= 0 \text{ or } \delta = \pm\sqrt{1-a^2} \end{aligned} \tag{9.2}$$

where the dimensionless parameter group

$$a = \frac{mgh}{2kr^2}$$

has been introduced. Three equilibrium points can exist only if $a < 1$ since δ must be real. The equilibrium δ=0 corresponds to a vertical equilibrium, i.e. α=0 ($\delta_1 = \delta_2 = \frac{mg}{2k}$), whereas $\delta = \pm\sqrt{1-a^2}$ corresponds to an equilibrium where $\alpha = \pm \arcsin\left(\sqrt{1-a^2}\right)$. When a is small, this corresponds to large α, implying that the centre of mass is outside the support area, and that the equilibria are invalid for a real robot. A detailed analysis based on the constraint that δ_1 and δ_2 cannot be negative, shows that $\delta = \pm\sqrt{1-a^2}$ is only valid when it corresponds to the centre of mass being directly above one of the feet. In that case, $\delta_1 = 0$ and $\delta_2 = \frac{mg}{k}$ or vice versa.

The next step in the analysis is to derive a linearized system around the equilibrium point corresponding to δ=0 and $\overline{\delta} = \frac{ar}{h}$. A reduced system, where the irrelevant horizontal dynamics has been removed, is as follows

$$\dot{x} = \begin{pmatrix} 0 & 0 & -r & -1 \\ 0 & 0 & r & -1 \\ \frac{rk(a-1)}{I_y} & \frac{rk(1-a)}{I_y} & -\frac{2r^2d}{I_y} & 0 \\ \frac{k}{m} & \frac{k}{m} & 0 & -2\frac{d}{m} \end{pmatrix} x$$

where $x = \left(\delta_1 - \frac{ar^2}{2},\, \delta_2 - \frac{ar^2}{2},\, w_1,\, w_2\right)^T$. The eigenvalues for this system are as follows

$$\lambda_{1,2} \quad = \quad -\frac{d \pm \sqrt{d^2 - mk}}{m}$$

and

$$\lambda_{3,4} = -\frac{rd \pm r\sqrt{r^2d^2 + 2\,(a-1)\,I_y k}}{I_y}$$

from which we easily see that the real parts of $\lambda_{1,2}$ will be negative if only m and k are positive. Similarly, if $a < 1$ the real parts of $\lambda_{3,4}$ will also be negative if in addition I_y is positive. Since m, k, d and I_y are all positive, we end up with the conclusion that the equilibrium point is asymptotically stable if and only if $a < 1$. Doing the same for the remaining two equilibrium points shows that they will be unstable if $a < 1$.

Note that the equilibrium point can be unstable even though the projection of the centre of mass is actually in the centre of the support, directly in contrast with what you would get from a static stability analysis. To verify this conclusion, equipment was built and experiments performed.

9.1.2. Comparison with CSSM

The criterion (12.1) has been compared with the requirement that the CSSM must be positive, and it was found that they produce very similar conclusions. There were some differences, especially for small r. However, the main difference between these criteria is that (12.1) verifies local stability, whereas the CSSM only approximates a measure of global stability.

9.2. Experimental verification with a test rig

It is difficult to experimentally find an unstable equilibrium. The system was therefore started in a stable equilibrium and the parameter a was slowly varied by increasing h until the equilibrium point becomes unstable. This is illustrated later in figure 9.5, where the height at which the system became unstable is plotted as a function of the radius. Movies of these experiments can be found at `http://www.md.kth.se/~cas/movies/balance_rigg`.

9.2.1. The equipment

The experiment rig is illustrated with a sketch and a photo in figure 9.3. It consists of an aluminum framework with a movable mass at the centre. The rig is hanging from the ceiling in two steel wires, each connected in series with a spring. The lower part of each spring is then connected by two strings to the ends of the horizontal bar. When the parameters are close to those of the bifurcation point, the system becomes extremely sensitive to perturbations. It was therefore impossible to manually change the height of the movable weight and a DC motor at the top of the vertical bar, together with a wire-and-pulley system, was used instead.

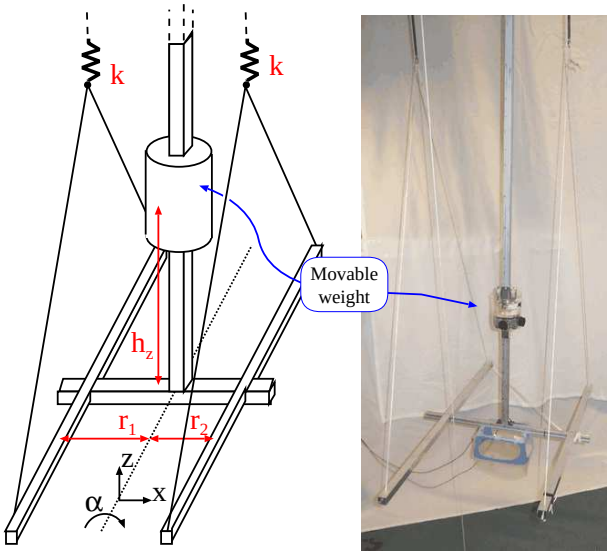


Figure 9.3: Sketch and photo of the balance rig

The only interaction with the environment is through the steel wires from the ceiling, the electrical wires and air friction. Suspending the rig in wires virtually eliminated dry friction. The two horizontal parallel bars are there in order to avoid rotations around the wrong axis. This method produced a very undamped system, so a damper was attached to the bottom of the rig — a part of the rig was simply submerged in water.

9.2.2. The method

Using the rig it is now possible to slowly raise the movable mass, thereby effectively changing the parameter h and consequently a.

In each experiment two springs and a specific r were chosen. The distance between the parallel bars was adjusted to match r, as well as the distance between the ceiling attachment points. Then the lengths of the steel wires were adjusted so that the equilibrium was vertical, i.e. the vertical bar was parallel to a lode. However, as the height is increased, small errors in the trimming of the wire lengths cause large deviations of the equilibrium. Therefore the lengths of the steel wires were repeatedly fine-tuned until the equilibrium was vertical even with higher heights.

Then the movable weight was raised slowly and in small steps (with pauses in between) until the rig tipped over. This procedure was repeated several times for each r.

Sensors

To increase the accuracy, a measuring system based on ultrasound was used to measure the height of the movable mass as well as the changes in the vertical bars orientation.

Estimation of parameters

The ultrasonic measurement system was also used to measure the stiffness of the springs and the steel wires. This was done by hanging an object with known mass in the spring (or steel wire) and making it vibrate vertically. The measured motion was then be fitted to a simple spring/damper model to obtain the stiffness.

Lengths and masses were measured in a straight forward way, and the centre of mass for the fixed part of the rig was obtained by balancing the rig on a sharp edge. The measured distance between the movable weight and the base of the rig, h_z, was then used to calculate the systems centre of mass.

Accuracy and uncertainty

The largest errors in the measured data stems from the uncertainties of the stiffnesses and the position of the centre of mass. Additionally, close to the bifurcation point, something that resembled dry friction was encountered during the experiment. This was probably caused by the dynamometers that were used as springs. There was some chafing between two of the tubes inside the dynamometers.

Table 9.1: Measured data of the balance rig

Parameter			Uncertainty	
Movable weight	m_1	1.00	0.01	kg
Total mass	m	4.00	0.01	kg
Combined stiffness	k	371	18	N/m
Spring stiffness	k_{spr}	378	8	N/m
Rig/wire stiffness	k_{sys}	9800	1000	N/m

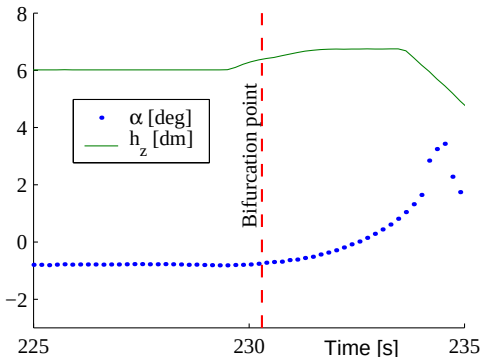


Figure 9.4: The bifurcation time was manually determined from graphs such as this (zoomed in around the bifurcation point). For each set of parameters the experiment was repeated several times and the graph studied.

Extraction of data

The actual point when the bifurcation occurred was determined by studying a graph of the orientation (α), and deciding when it starts to fall over (figure 9.4). Then the height of the movable mass at this time was used to calculate the height of the centre of mass.

9.2.3. Results of experiment

Performing experiments with different values for the radius produced data that was consistent with the theory. Table 9.1 shows the results for different values of the parameter r and the corresponding measured h and estimated value of a. The last column contains an estimate of the maximum error that should be possible in the estimate of a, taking into account estimates of parameter uncertainties.

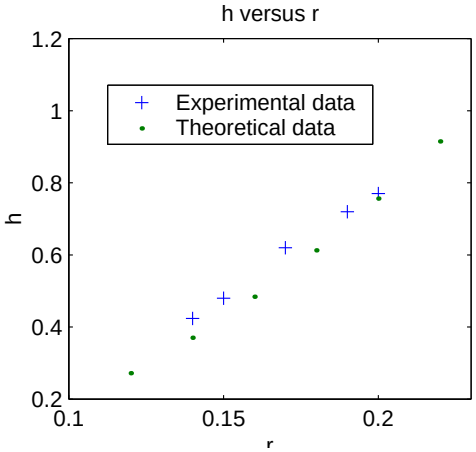


Figure 9.5: Plot of experimental result compared to theoretical results. The plot shows the height at which the system became unstable, h_c, as a function of the radius r. Several measurements of h_c were done for each r. The dots correspond to the theoretical limit, and the '+'-symbols to the h_c as measured in the experiments.

The result agrees fairly well with the hypothesis that the bifurcation occurs when $a = 1$. It is disturbing that the relative errors are all positive, since that could indicate a systematic error in the experimental method (or an error in the hypothesis). Another way of illustrating this data is shown in figure 9.5, where the measured height at the bifurcation is plotted as a function of the parameter r.

9.2.4. Variation of γ

In this experiment, the width of the base was kept constant and γ was varied. It seems that the bifurcation height, i.e. when the system becomes unstable, does not vary significantly with respect to γ! However, with a larger magnitude (of γ), it became increasingly difficult to trim the system to obtain the stable vertical equilibrium

9.3. Domain of attraction

This section investigates a subset of the domain of attraction for the symmetric equilibrium. It is a subset since we assume that all initial velocities are zero and that the system initially is in a vertical equilibrium. First consider the case of a standing

on a hard surface. The domain of attraction for this case corresponds to the initial projection of the centre of mass being inside the support area. This is expressed by the following condition

$$\mathbf{n}_1 \cdot \mathbf{r}^{C_1 B} > 0 \text{ and } \mathbf{n}_1 \cdot \mathbf{r}^{BC_2} > 0$$

that simply states that the horizontal distances from B to the contact points must be positive. With some trigonometry we can show that this is equivalent to

$$(1+\gamma)\frac{r}{h} + \tan\alpha > 0 \text{ and } (1-\gamma)\frac{r}{h} - \tan\alpha > 0$$

which can be rewritten as

$$\left|\tan\alpha + \gamma\frac{r}{h}\right| < \frac{r}{h}.$$

With $\gamma = 0$ it is easy to see for what initial angles the stiff system will be stable, but what about the compliant system?

Simulations were run to answer this question and the results are shown in figures 9.6-9.8. In the figures, each symbol represents the result of a simulation, where a dot means it was stable and an $\times$ that it fell over. The vertical position of each symbol indicates the value of a for that simulation and the horizontal position is related to the initial stability margin. A symbol's coordinates are determined in this way

$$\left(r - \mathbf{n}_1 \cdot \mathbf{r}^{BC_2}\big|_{t=0}\,,\, a\right).$$

When $\alpha_0 = 0$ the horizontal position is 0 and it increases with α_0. If the horizontal position is r, the initial stability margin is zero. A robot standing on a stiff surface would only fall if starting outside $(-r, r)$. This limit is illustrated by dotted vertical lines in the figures.

Furthermore, the initial velocities were all zero and the initial vertical position was chosen so that the sum of the ground forces equals the weight, i.e. in a vertical equilibrium.

Given a value for a, the stiffness was calculated for each simulation. In figure 9.6, the height is $h = 1$ and the width is $r = 0.1$. For these parameters $a \approx 1$ corresponds to $k \approx 500$. Figure 9.6 seems to indicate that as long as $a < 1$, the result is the same as for when standing on a hard surface. That this is not true is shown by figure 9.7 where $a \approx 1$ and we can see that starting with a small, but positive, stability margin may still result in a fall.

This effect is more evident in figure 9.8 where $h = 0.5$, corresponding to $k \approx 240$ when $a \approx 1$, i.e. the ground is softer. Still, it seems that unless the system is very compliant, it is in practice enough to require that $a < 1$ together with the normal condition for static balance.

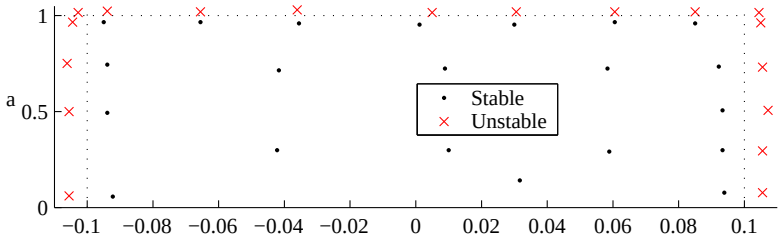


Figure 9.6: Simulation results when $h = 1$ and $r = 0.1$ for different initial conditions (horizontal axis) and parameters a.

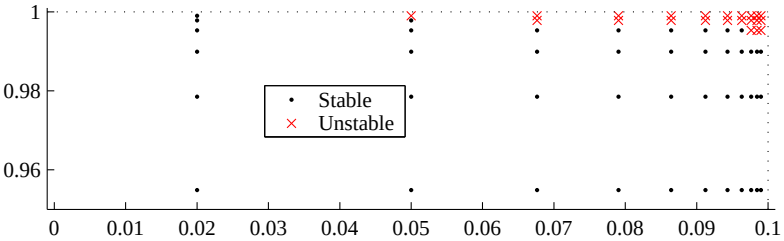


Figure 9.7: Simulation results when $h = 1$ and $r = 0.1$ for $a \approx 1$.

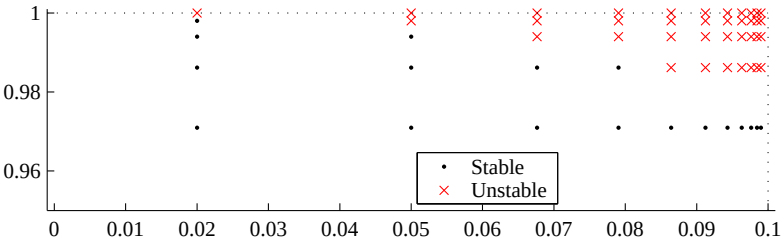


Figure 9.8: Simulation results when $h = 0.5$ and $r = 0.1$ for $a \approx 1$.

10. Extensions — radially symmetric and planar asymmetric stances

This chapter continues the analysis of a robot standing on a compliant horizontal surface. The criterion from the previous chapter is extended to three dimensions for two-, three- and four-legged radially symmetric stances. For the symmetric planar case, a simple "proof" and and intuitive explanation is presented. It is also suggested that local asymptotic stability could be determined in a simple way by comparing the "torsion stiffness due to gravity", with the "torsion stiffness due to compliance".

Finally, the asymmetric planar case is studied further by visualizing equilibria as a surface that depends on two dimensionless parameters. Furthermore, the stability of points on this surface are numerically determined and shown. Finally, an example is given of how this surface can be used to reason about robot motions that are statically balanced when compliance is included.

10.1. Radially symmetric models

The model in the previous chapter was planar and we will now try to work with a more complete model. Consider the *ideal legged locomotion machine*, (McGhee and Frank [117]) illustrated in figure 9.1b. A rigid body is assumed with massless legs and the foot contacts are approximated as point contacts, where the force exerted by the legs must be directed into the supporting surface. Additionally, consider all compliance to be lumped together as linear elasticity in the supporting surface. However, even this model is complicated.

Instead, figure 10.1 illustrates models of two-, three- and four-legged radially symmetric stances, where the trunk has mass m, with the centre of mass at the point B. The inertia parameters I_x, I_y and I_z are expressed relative to B. The supporting surface has stiffness k and damping d, at each of the L feet that are

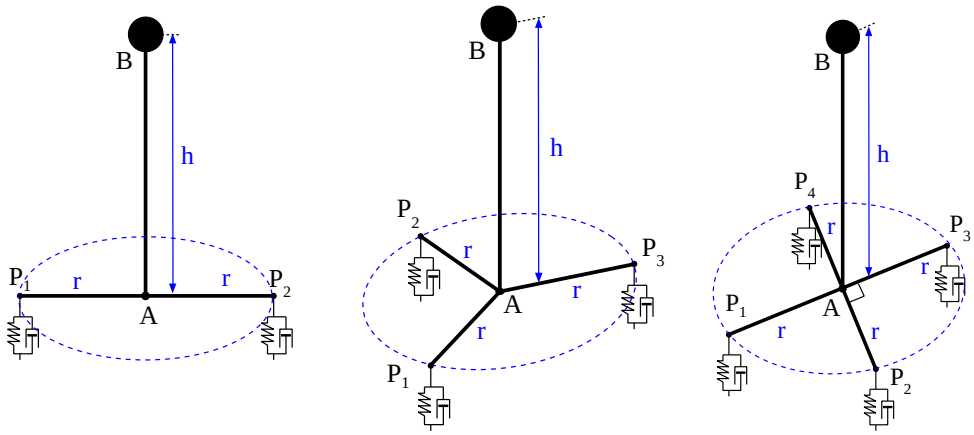


Figure 10.1: Models of two-, three- and four-legged radially symmetric stances.

placed evenly on a circle with radius r around the point A. The distance from A to B is denoted by h, and g is the constant of gravity directed downward. The parameters m, g, k, I_x, I_y, I_x, h and r are all assumed to be positive.

For $L = 2$, the criterion is

$$\frac{mgh}{Lkr^2} = a < 1 \tag{10.1}$$

and maybe this naturally extends to L legs? To determine this, the differential equations were derived for a robot with L legs supporting a rigid body, i.e. there are six degrees of freedom. The generalized coordinates are chosen so that

$$\mathbf{r}^{NB} = q_1\mathbf{n}_1 + q_2\mathbf{n}_2 + q_3\mathbf{n}_3$$

and q_4, q_5 and q_6 give a yaw-pitch-roll transformation from N to B. The generalized speeds are chosen so that

$$\frac{{}^N\partial\mathbf{r}^{NB}}{\partial t} = w_1\mathbf{n}_1 + w_2\mathbf{n}_2 + w_3\mathbf{n}_3$$

and that

$$\boldsymbol{\omega} = w_6\mathbf{n}_1 + w_5\mathbf{n}_2 + w_4\mathbf{n}_3.$$

However, since purely vertical forces are assumed, the equations become simpler around the vertical axis, and along the horizontal axes. During the analysis of the

planar model, several linear ground models were tested. For the horizontal and symmetric case, it seemed enough to only use the vertical forces (for a non-horizontal surface this is obviously not true).

As in chapter 10, the stability analysis was done symbolically using computer assisted algebra tools in the following standard manner:

- Derive differential equations for the model of the mechanical system.
- Determine the symmetric equilibrium (the vector $\overline{BA}$ is parallel to gravity).
- Linearize around the equilibrium.
- Determine eigenvalues and analyze them to determine stability.

10.1.1. Two-legged stances — planar and 3D

A repeated analysis of the planar model using a 6-dof model produces the same result, except that it can now fall "out of the plane". That aside, the relevant eigenvalues are as follows:

$$\lambda_{1,2} = -\frac{d \pm \sqrt{d^2 - mk}}{m}$$

$$\lambda_{3,4} = -\frac{rd \pm \sqrt{r^2 d^2 + 2\left(a_2 - 1\right) I_y k}}{I_y}$$

where

$$a_2 = \frac{mgh}{2kr^2}. \tag{10.2}$$

It is easy to see that the real parts of $\lambda_{1,2}$ will be negative. The real parts of $\lambda_{3,4}$ are negative if $a_2 < 1$, leading to an asymptotically stable equilibrium (ignoring the singular case of zero eigenvalues).

10.1.2. A simpler proof

Why does the system become unstable? In order to intuitively understand that, consider figure 10.2a, illustrating the planar model in equilibrium. In figure 10.2b it has been perturbed by a small angle $\Delta\alpha$ and the corresponding approximate displacements are illustrated. The approximate torque change, ΔM, around the point A can be written as follows

$$\Delta M = \Delta\alpha mgh - \Delta\alpha r^2 k - \Delta\alpha r^2 k.$$

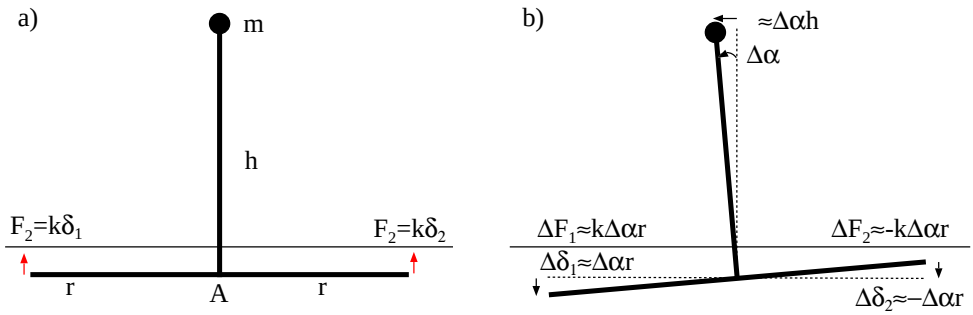


Figure 10.2: a) Symmetric equilibrium configuration. b) Perturbed configuration.

The torsion stiffness can then be calculated as follows

$$\frac{\Delta M}{\Delta \alpha} = mgh - 2r^2k = 2r^2k\left(\frac{mgh}{2r^2k} - 1\right) = 2r^2k\,(a-1)\,.$$

When $a < 1$, the stiffness is negative and the equilibrium is stable (since the torque change will counteract the perturbation). Alternatively, think of the problem directly in terms of stiffness due to the ground, versus stiffness due to gravity. The two springs acting at the feet comprise a torsion spring with stiffness

$$k_\tau = 2r^2k.$$

Now rewrite the stability condition as follows

$$\frac{mgh}{k_\tau} = a < 1 \quad \Leftrightarrow \quad mgh < k_\tau.$$

The torsion stiffness due to the ground, k_τ, must be greater than the stiffness due to gravity, mgh. This is similar to the well known example of a buckling beam. It seems to have been overlooked in the research of legged machines. There is of course one major difference: typically only unidirectional ground forces are possible for legged machines (as is assumed for an ideal legged locomotion machine). For instance, a stable equilibrium for the buckling beam can correspond to a situation where one of the feet are above the ground.

10.1.3. Three- and four-legged stances

For the three-legged stance, the following eigenvalues correspond to the symmetric equilibrium:

$$\begin{aligned}
\lambda_i &= -\frac{1}{2m}\left(3d \pm \sqrt{9d^2 - 12mk}\right),\ i = 1,2 \\
\lambda_i &= -\frac{r}{4I_y}\left(3dr \pm \sqrt{(3dr)^2 - 24kI_y\,(1-a_3)}\right)\ \ i = 3,4 \\
\lambda_i &= -\frac{r}{4I_x}\left(3dr \pm \sqrt{(3dr)^2 - 24kI_x\,(1-a_3)}\right)\ \ i = 5,6
\end{aligned}$$

where

$$a_3 = \frac{3mgh}{2kr^2}. \tag{10.3}$$

For the four-legged stance, the following eigenvalues correspond to the symmetric equilibrium:

$$\begin{aligned}
\lambda_i &= -\frac{2}{m}\left(d \pm \sqrt{d^2 - mk}\right),\ i = 1,2 \\
\lambda_i &= -\frac{2r}{I_x}\left(dr \pm \sqrt{(dr)^2 - 2kI_x\,(1-a_4)}\right)\ \ i = 3,4 \\
\lambda_i &= -\frac{2r}{I_y}\left(dr \pm \sqrt{(dr)^2 - 2kI_y\,(1-a_4)}\right)\ \ i = 5,6
\end{aligned}$$

where

$$a_4 = \frac{mgh}{2kr^2}. \tag{10.4}$$

The first two eigenvalues correspond to vertical modes, and the last four to tipping modes. Reasoning as in the planar case, we see that the equilibrium is stable when

$$a_L < 1,\ L = 3,4.$$

10.2. An L-legged stance — using the stiffness

A torsion stiffness can be calculated in a manner similar to that in section 10.1.2, generalized to three dimensions. In the three-legged stance, the stiffness is $\frac{2}{3}kr^2$ and in the four-legged stance, the stiffness is $2kr^2$. Comparing this to a_3 and a_4, indicates that the stability can be predicted by comparing the torsion stiffness to mgh.

The calculation of the torsion stiffness $\mathbf{k}_\tau$ around the point A is briefly described below. In these radially symmetric stances, the torsion stiffness should be independent of the horizontal axis around which it is calculated (except of course in the two-legged case, where tipping "out-of-the-plane" corresponds to no stiffness at all). Using the computer algebra system, the stiffnesses were calculated by evaluating

$$\mathbf{k}_\tau = \sum_{l=1}^{L} \mathbf{r}^{AP_l} \times \left(\mathbf{K} \cdot \Delta\mathbf{r}^{AP_l}\right)$$

where $\mathbf{r}^{AP_l}$ is a vector from the point A to foot l, $\mathbf{K} = k\mathbf{n}_3\mathbf{n}_3$ represents the ground stiffness and $\Delta\mathbf{r}^{AP_l}$ is a displacement due to a small rotation. The displacement is calculated as

$$\Delta\mathbf{r}^{AP_l} = \left.\frac{{}^N\partial\mathbf{r}^{AP_l}}{\partial t}\right|_{q=q^*} \cdot \sum_{i=1}^{3} \frac{\mathbf{n}_i\mathbf{n}_i}{\mathbf{n}_i \cdot \boldsymbol{\omega}},$$

where the dyad $\sum_{i=1}^{3} \frac{\mathbf{n}_i\mathbf{n}_i}{\mathbf{n}_i\boldsymbol{\omega}}$ eliminates the angular velocity components created by the time differentiation where $q = q^*$ denotes the equilibrium point.

10.3. Analysis of asymmetric configuration

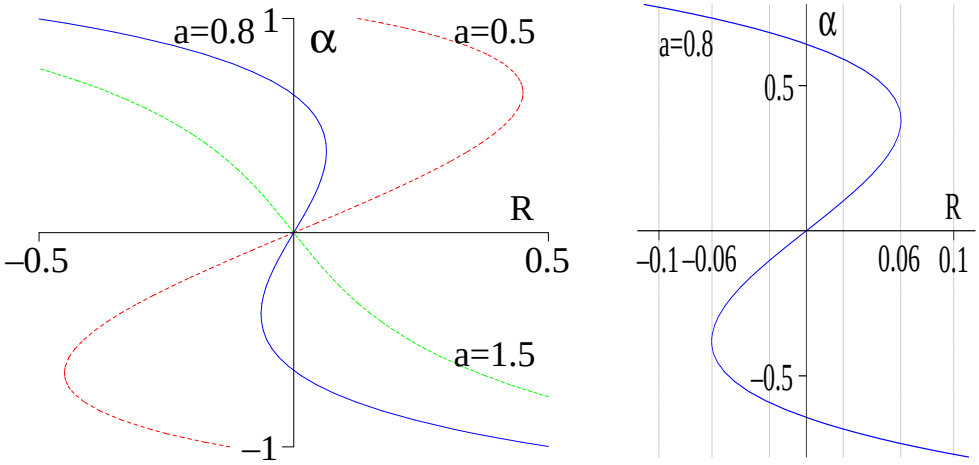
The planar asymmetric case ($\gamma \neq 0$) will now be studied further. It is more difficult to analyze and the results are also more difficult to visualize. Some of the figures will show three-dimensional surfaces that can be difficult to "see".

The model was illustrated in figure 9.2 but its differential equations have been derived using the same generalized coordinates as in section 10.1. By introducing dimensionless parameter groups, the equations of equilibrium can be written as follows

$$\begin{aligned} R + \tan\alpha - \frac{\sin\alpha}{a} &= 0 \\ q_3 + R\sin\alpha - \cos\alpha + \frac{1}{k'} &= 0 \end{aligned}$$

where $R = \gamma r/h$, $k' = 2kh/mg$ and q_3 is the vertical position of the trunk's centre of mass. Note that these equations only represent a "real" equilibrium, if it corresponds to the robot being supported by both feet. Consequently, some of the equilibria do not exist for an actual robot, but this will be ignored here.

The equilibrium equations need to be solved for both α and q_3, but we can note that the first equation only depends on α, and once α is known the solution for q_3 is



(a) Curves for three values of a

(b) Curve for $a = 0.8$

Figure 10.3: The curves correspond to (10.6) for different values of a. Intersections with a vertical line correspond to solutions of (10.5) and if $a > 1$, there is only one intersection. **(b)** shows several vertical lines for different values of R.

trivial. The first equation can be rewritten as

$$R = -\tan\alpha + \frac{\sin\alpha}{a} \tag{10.5}$$

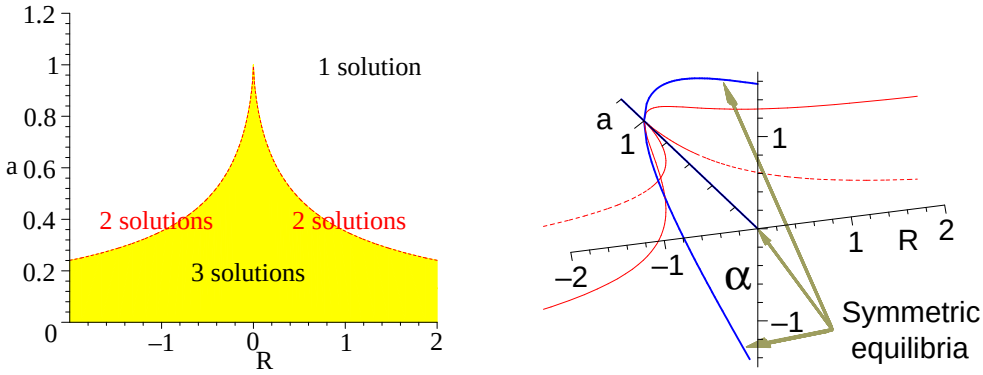
where solutions to this equation corresponds to intersections between a vertical line (at $x = R$) and a curve

$$x(\alpha) = -\tan\alpha + \frac{\sin\alpha}{a},\ y(\alpha) = \alpha \tag{10.6}$$

as shown in figure 10.3. Note that depending on the value of a, the vertical line can intersect the curve either one, two or three times. Consequently the number of equilibria also vary between one, two and three.

Figure 10.4 shows how the number of equilibria depend on the parameters (α and R), where there are three equilibria below the curve, two equilibria on the curve and only one equilibria above the curve. A bifurcation occurs when crossing this curve and the corresponding equilibrium that bifurcates is also shown in figure 10.4.

The complete equilibrium surface is shown in figure 10.5, where the symmetric equilibrium solutions have been drawn on the surface as three thick curves. Given



(a) Number of equilibria

(b) 3D-view of equilibrium trajectory

Figure 10.4: a) There are two equilibria along the dashed curve, one above and three below. This curve is also shown in 3D, where the vertical axis is the equilibrium angle and the symmetric equilibria are drawn. The thick curve traces out the equilibria that bifurcates when changing regions in **(a)**.

a point in the parameter plane, the equilibria correspond to intersections between a vertical line going through that point and the equilibrium surface.

As in the symmetric case, it is possible to linearize the system around an equilibrium point and study the eigenvalues. Deriving the linearized system for an arbitrary point is straight forward[1], but the expressions for the eigenvalues are too complicated to make much sense. However, it is still possible to numerically evaluate the eigenvalues at various points on the equilibrium surface. This is illustrated in figure 10.6, where light dots indicate that all the eigenvalues have negative real parts at that point. The light dots represent stable equilibria, and initial conditions close to such a point means it will remain there. From the figure, it looks like all points "inside" the bifurcation curve are stable. However, even though this seems very likely, the figure is not a strict proof. Calculating the eigenvalues required assigning specific values to d, I_y and r, so for other values of these parameters the figure might look different.

Now assume that the figure is representative for a real robot standing on a soft surface, or even explicitly generated for the parameters of that robot. For a specific robot, the dimensionless parameters are now completely determined by the position

[1]Not by hand, but using a computer algebra system.

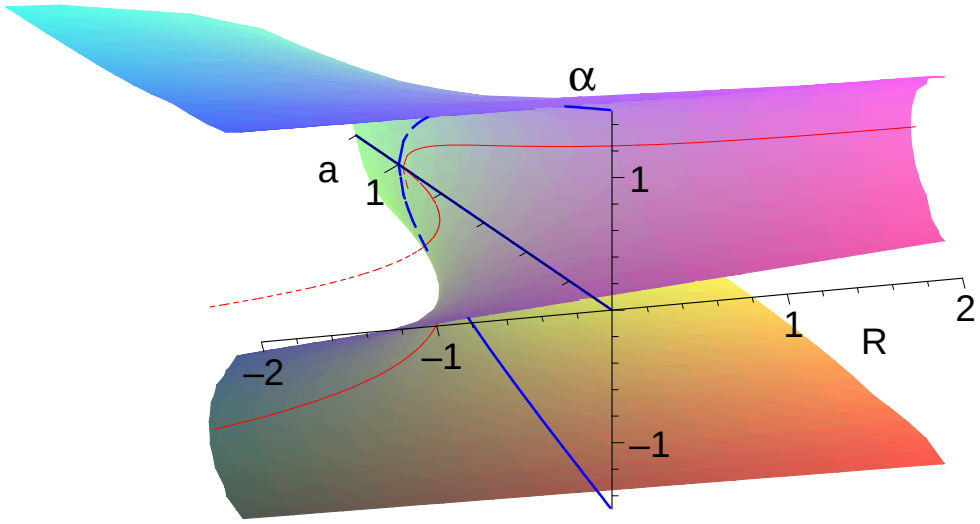


Figure 10.5: Equilibrium surface for planar model. The three thick curves at $R = 0$ correspond to the symmetric equilibria solutions (9.2).

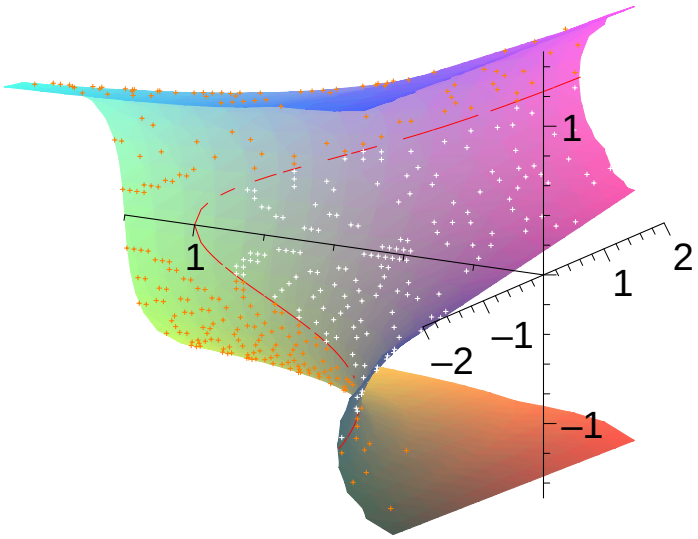


Figure 10.6: Equilibrium surface for planar model where the white dots indicate that all eigenvalues at that point have a negative real part.

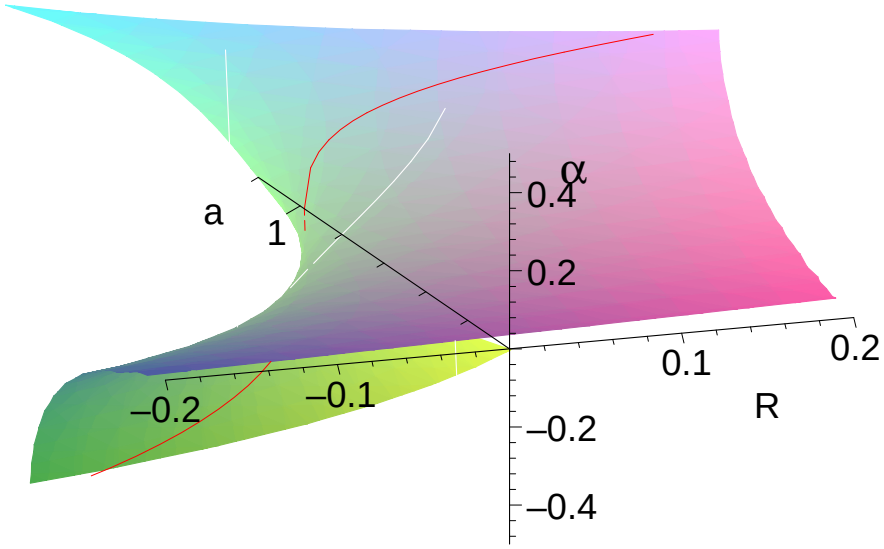


Figure 10.7: The white line shows the equilibria corresponding to $a = 0.5$, $r = 0.1$ and $\gamma = -0.9 \ldots 0.9$, i.e. as the trunk slowly moves from one side to the other.

of the trunk relative to the feet. Furthermore, assume that the robot is in a stable equilibrium. This equilibrium will correspond to a point on the equilibrium surface in figure 10.5. By varying the parameters slowly enough, the robot will remain at (or very close to) the equilibrium point as that points move along the surface. An example of this is shown in figure 10.7 where the equilibrium trajectory corresponds to $a = 0.5$, $r = 0.1$ and $\gamma = -0.9 \ldots 0.9$. This curve gives the real motion of the robot and if the equilibrium undergoes a bifurcation the robot will fall. It should be noted that the system is likely to become more and more sensitive as it approaches the edges of the stable part of the equilibrium surface. Also remember that it has been assumed that both feet are in ground contact.

11. Compliance in the control

In this chapter, the ground is no longer assumed to be compliant. Instead, the compliance is "caused" by the fact that controllers use low feedback gains. Two controllers, that have both been tested on WARP1, are analyzed using planar approximations. In the first case, the feet are position controlled in Cartesian coordinates and in the second case, a posture controller uses force control in the vertical direction. Some experiment results from when this posture controller was tested on WARP1 are also included at the end of this chapter.

11.1. Cartesian position control of the feet

This controller is only intended to keep the feet still. A Cartesian position control law, for a force between the trunk and a foot l, can be written as:

$$\mathbf{f}^l = \mathbf{P} \cdot \left(\mathbf{r}^{BP_l,ref} - \mathbf{r}^{BP_l} \right) - \mathbf{D} \cdot \mathbf{v}^{BP_l}$$

where $\mathbf{P}$ is the stiffnes and $\mathbf{D}$ the damping. The Cartesian system is assumed to coincide with the trunk triad and the control law can therefore be written as:

$$^B f^l = {}^B K^l \left({}^B r^{BP_l,ref} - {}^B r^{BP_l} \right) - {}^B D^l {}^B v^{BP_l} \tag{11.1}$$

where we use diagonal stiffness and damping matrices:

$$^B K^l = \begin{bmatrix} k_1 & & \\ & k_2 & \\ & & k_3 \end{bmatrix} \text{ and } {}^B D^l = \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \end{bmatrix}. \tag{11.2}$$

11.1.1. Implementation on WARP1

This kind of control law (but with velocity references and a non-constant position reference) has been used with WARP1 to make it both crawl and trot [84]. See

section 5.3 on page 133 for an overview of the control structure, where the torque control law (5.1) already contains (11.1). Note that (5.1) is expressed in terms of vectors between the hip (H_l) and the feet. However, this does not change anything as long as we ensure that the reference satisfy this equation:

$$\mathbf{r}^{BP_l,ref} = \mathbf{r}^{BH_l} + \mathbf{r}^{HP_l,ref}.$$

As for the Jacobian, we have that

$${}^{B}J^{l} = \frac{\partial^B r^{BP_l}}{\partial q} = \frac{\partial^B r^{BH_l}}{\partial q} + \frac{\partial^B r^{HP_l}}{\partial q} = \frac{\partial^B r^{HP_l}}{\partial q}$$

since $\mathbf{r}^{BH_l}$ is fixed relative to the trunk.

11.1.2. Analysis

A planar version of this controller will now be analyzed. As in section 10.1 we assume that the robot can be modeled as an ideal legged locomotion machine. This time the ground is stiff and we assume that the feet do not slip. The parameters m and g are again the trunk mass and constant of gravity, but h and r are now used in the position references

$$\begin{aligned} \mathbf{r}^{BP_1,ref} &= -r\mathbf{b}_1 - h\mathbf{b_3} \\ \mathbf{r}^{BP_2,ref} &= r\mathbf{b}_1 - h\mathbf{b_3} \end{aligned}$$

where h is the desired trunk height and $2r$ is the width of the stance (and robot, since the desired position is under the hips).

The dynamic model uses the following generalized coordinates:

$$\mathbf{r}^{NB} = q_1\mathbf{b}_1 + q_3\mathbf{b}_3 \text{ and } q_2 = \alpha$$

and generalized speeds

$$w_i = \dot{q}_i,\ i = 1 \ldots 3.$$

Deriving the system's differential equations and determining the (symmetric) equilibrium gives that it is:

$$q^* = \left[0, 0, h - \frac{mg}{2k_3}\right]^T,$$

where the trunk has sunk down a distance

$$\frac{mg}{2k_3}$$

relative to the desired height. The displacement is due to the compliance in the controller rather than the ground this time. Linearizing around the equilibrium produces these dynamic differential equations:

$$\begin{aligned} I_y\dot{w}_1 &= c_1 q_1 + 2d_1\left(\frac{mgh}{k_3} - \frac{I_y}{m} - h^2 - \frac{(mg)^2}{2k_3^2}\right) w_1 + \\ &\quad \left(2r^2hk_3 - mgr^2 + gI_y\right) q_2 + d_3 r^2\left(2h - \frac{mg}{k_3}\right) w_2 \\ I_y\dot{w}_2 &= \left(mg + 2hk_1 - \frac{k_1}{k_3}mg\right) q_1 + \left(2hd_1 - \frac{d_1}{d_3}mg\right) w_1 - 2r^2\left(d_3 w_2 - k_3 q_2\right) \\ m\dot{w}_3 &= -2k_3 q_3 - d_3 w_3 \end{aligned}$$

where the coefficient c_1 is

$$c_1 = \frac{(mg)^2}{2k_3}\left(1 - \frac{k_1}{k_3}\right) + mgh\left(2\frac{k_1}{k_3} - 1\right) - 2k_1\left(h^2 + \frac{I_y}{m}\right).$$

Two eigenvalues are easy to find for this system:

$$\lambda_{1,2} = \frac{-d_3 \pm \sqrt{d_3^2 - 2mk_3}}{m}$$

but the remaining four eigenvalues are the roots of a large fourth degree polynomial. A necessary criterion for the real parts of the eigenvalues to be negative, is that the polynom's 0^{th} coefficient is positive. This criterion can be written as follows

$$0 < 2k_3\left((mg)^2(k_1 - k_3) + 2k_1k_3\left(2k_3r^2 - mgh\right)\right)$$

which requires a positive horizontal stiffness, i.e.

$$k_1 > 0.$$

To further simplify the expression, let us replace the parameters with dimensionless equivalents, i.e.

$$\begin{aligned} k_1 &= \frac{mg\acute{k}_1}{h} \\ k_3 &= \frac{mg\acute{k}_3}{h}. \end{aligned}$$

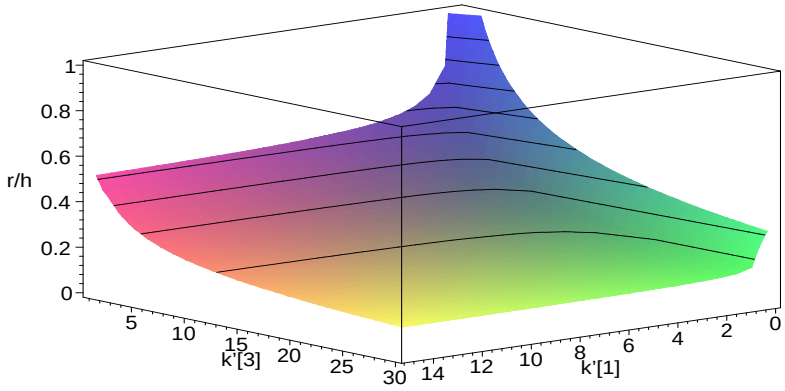


Figure 11.1: The surface illustrates the critical point for different values of the parameters k_1, k_3 and $\frac{r}{h}$. The criterion is satisfied for points above the surface. High stiffnesses allow a narrower stance, and for $\acute{k}_1 > 1$, the vertical stiffness is more important than the horizontal stiffness.

This is inserted into the inequality and simplified as follows:

$$\begin{aligned} 0 &< \acute{k}_1 - \acute{k}_3 + 2\acute{k}_1\acute{k}_3\left(2\acute{k}_3\frac{r^2}{h^2} - 1\right) \\ \acute{k}_3 &< \acute{k}_1\left(1 + 2\acute{k}_3\left(2\acute{k}_3\frac{r^2}{h^2} - 1\right)\right) \\ \frac{1}{\acute{k}_1} &< \frac{1 + 2\acute{k}_3\left(2\acute{k}_3\frac{r^2}{h^2} - 1\right)}{\acute{k}_3} \end{aligned} \tag{11.3}$$

This criterion is illustrated in figure 11.1, showing a surface that satisfies the equation

$$\frac{1}{\acute{k}_1} = \frac{1 + 2\acute{k}_3\left(2\acute{k}_3\frac{r^2}{h^2} - 1\right)}{\acute{k}_3}. \tag{11.4}$$

The criterion is satisfied when the parameters correspond to a point above the surface. We can see that with high stiffnesses, the stance can be narrower and also that the vertical stiffness is more important than the horizontal stiffness (for $\acute{k}_1 > 1$ anyway).

11.1.3. Comparison with WARP1

It is now possible to test if (11.3) is satisfied for some typical control parameters that are actually used during the control of WARP1. Since WARP1 has four legs and this was a planar model, the control parameters from WARP1 must be doubled. A typical leg control stiffness is about 4 kN/m vertically and 10 kN/m horizontally, corresponding to $\acute{k}_1 \approx 7$ and $\acute{k}_3 \approx 17$. For these values, the critical value of $\frac{r}{h}$ is about 0.19, but typically WARP1 use a wider stance with a ratio of 0.5. This agrees with the fact that WARP1 is easily capable of standing using these values (in both simulation and reality).

A few preliminary simulations have also been done where the width (of both stance and trunk) was reduced. As the parameters approached the critial point, WARP1 fell, and seemed to fall faster as it was closer. However, only a few cases were simulated using full models of WARP1 and the simulations are time consuming. It is probably not feasible to verify the surface in figure 11.1 in this way, but requires simulations using a reduced model. The result is still interesting and before the analysis we did not expect it to be so difficult for WARP1 to remain standing using this controller when the stance was narrow.

Note that this criterion is unlikely to be accurate for WARP1 since it does not model the actuator dynamics, multiple rigid bodies nor implemenation of the Cartesian stiffness/damping control.

11.2. A simple force based posture controller

A simple force based posture controller [160] will now first be described and then analyzed in a similar manner. The posture controller is based on a few simple ideas:

1. Use force control vertically and position control horizontally
2. Use simple rules to distribute leg forces
3. The supporting surface should not have to be horizontal, planar or even static
4. The height is not easily defined for a legged robot, use average of “hip heights”.

The goal of the controller is to track a desired height and attitude (roll and pitch angles), i.e. a desired posture. This is to be done without any knowledge of the ground type or orientation. Furthermore, only on-board sensors will be used. In one of the experiments, the ground is not even static since the robot is actually standing on a balancing board that is being tilted back and forth (figure 11.2).

Methods to control balance similar to the one described in this paper have been used before, see for instance section 3.2.1 (ASV) and section 3.1.3 (Sky-Hook suspension). There has been more research on this subject for bipeds [86, 145]. Similar

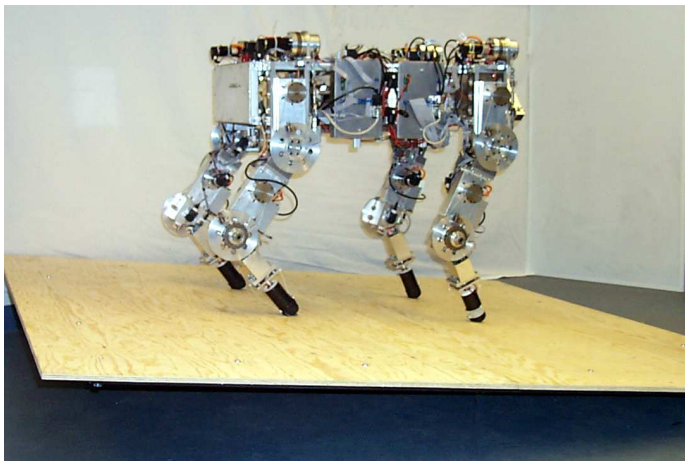


Figure 11.2: WARP1 on a balance board.

to the method used in the ASV, we would like to control the trunk's posture by distributing the forces applied onto the trunk by the legs. However, we do this in a simpler way, by using a hybrid control of horizontal foot positions and vertical forces. Another method to distribute the forces is to formulate a quadratic optimization problem, like Chen et al. [20]. This method has mostly been used in simulation ([82] [20]), with the ASV as a rare exception.

Simply put, in order to increase the trunk height, all vertical leg forces are increased by the same amount. Similarly, to pitch the trunk forward, more vertical force is applied by the rear legs and less by the front legs. However, there are some problems. One of them is the need for a good estimate of the trunk's orientation and height. Further, when the ground is unknown, it is not obvious how to define the height. Here it is defined as the average of the hip heights.

11.2.1. Posture observer

The trunk orientation is estimated by a high-gain algorithm based on data from onboard sensors (three rate gyros, two inclinometers and one accelerometer). Two sets of gains are used in the algorithm depending on how abruptly the trunk moves (measured by the accelerometer). During abrupt motions, the inclinometer gain is much smaller than during smooth motions. Rehbinder [153] contains more information.

The trunk position[1] is estimated relative to the average position of the feet, de-

[1] I.e. its centre which is assumed to coincide with the trunk's centre of mass.

noted G. It is calculated as follows

$$^{B}r^{GB} = -^{B}r^{BG} = -\frac{1}{L}\sum_{l=1}^{L} {}^{B}r^{BP_l}.$$

where L is the number of legs (i.e. $L = 4$) and $^{B}r^{BP_l}$. The *height* is now defined as $^{B}r_3^{GB}$ and the estimated velocity of the trunk is calculated in the same way, i.e.

$$^{B}v^{GB} = -\frac{1}{L}\sum_{l=1}^{L} {}^{B}v^{BP_l} = -\frac{1}{L}\sum_{l=1}^{L} \frac{\partial {}^{B}r^{BP_l}}{\partial t}.$$

Note that this is usually not the same as the "real" velocity, i.e. the velocity relative to the inertial frame, N. That velocity could be calculated as follows

$$\frac{^{N}\partial \mathbf{r}^{GB}}{\partial t} = {}^{N}\boldsymbol{\omega}^{B} \times \mathbf{r}^{GB} + \frac{^{B}\partial \mathbf{r}^{GB}}{\partial t}$$

where $^{N}\boldsymbol{\omega}^{B}$ is the angular velocity of the trunk and it is assumed that the point G does not move.

11.2.2. Posture controller

Two parts comprise the posture controller. The first part consists of PID-controllers that calculate the desired force and moment on the trunk. A constant term mg is added to the desired force in order to compensate for gravity. The desired force is calculated as follows:

$$^{B}f^{B,ref} = k_P e_p + k_D \dot{e}_p + k_I \int e_p \partial t + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

where e_p and $\dot{e}_p$ are the position and velocity error, i.e.

$$\begin{aligned} e_p &= {}^{B}r^{GB,ref} - {}^{B}r^{GB} \\ \dot{e}_p &= {}^{B}v^{GB,ref} - {}^{B}v^{GB}. \end{aligned}$$

Note that the velocity reference is relative to the trunk's coordinate system, not the inertial frame. Similarly, the feedforward term, mg, is also in the trunk's coordinate system. This will produce a small error when the trunk is not horizontal, but it is ignored. The desired moment on the trunk, $^{B}m^{B,ref}$, is calculated similarly, based on the error in trunk orientation and angular velocity.

The second part of the controller calculates what forces the legs should apply, ${}^B f^{BP_l,ref}$ $l = 1 \ldots L$. A simple distribution rule is used here that only distributes the vertical leg forces as follows

$$ {}^B f_3^{BP_l,ref} = -\frac{1}{L}\left({}^B f_3^{B,ref} + \frac{{}^B m_1^{B,ref}}{{}^B r_2^{BP_l}} - \frac{{}^B m_2^{B,ref}}{{}^B r_1^{BP_l}} \right). $$

This satisfies the force balance equations at least when the robot is horizontal. It is assumed that the leg controllers keep the feet directly below the hips. The actual foot positions can of course also be used, but "division-by-zero" must then be handled if a foot is directly below the trunk's centre of mass.

11.2.3. Implementation on WARP1

The implemenation of this posture controller uses the control structure described in section 5.3 on page 133. However, the control law for calculating joint torques, τ^l, is different from (5.1):

$$ \begin{aligned} \tau^l = \quad \tau^{fc} + \left({}^B J^l\right)^T \cdot \{ & {}^B K^l \cdot \left({}^B r^{BP_l,ref} - {}^B r^{BP_l}\right) + \\ & + {}^B D^l \cdot \left({}^B v^{BP_l,ref} - {}^B v^{BP_l}\right) + \\ & + {}^B f^{BP_l ref} \} \end{aligned} \tag{11.5} $$

The term τ^{fc} has been added. It is a *knocker*, i.e. a small amplitude square wave (40 Hz) that reduces the effect of static friction. Only diagonal ${}^B K^l$ and ${}^B D^l$ were used here. The reference trajectories are constant, with the feet placed directly below the hips. However, the vertical position and velocity components, ${}^B r_3^{BP_l}$, ${}^B r_3^{BP_l,ref}$ etc., are not used since the corresponding stiffness and damping coefficients are set to zero.

$$ {}^B K^l = \begin{bmatrix} k_L & & \\ & k_L & \\ & & 0 \end{bmatrix} \text{ and } {}^B D^l = \begin{bmatrix} d_L & & \\ & d_L & \\ & & 0 \end{bmatrix}. $$

Similarly, only the vertical component of the reference force, ${}^B f_3^{BP_l ref}$, is nonzero.

11.2.4. Analysis of posture controller

The assumptions for this model are the same as in section 11.1.2 and the model uses the same generalized coordinates and speeds. Deriving the system of the differential equations and determining the equilibrium gives that it is:

$$ q^* = (0, 0, h). $$

Table 11.1: Control parameters in planar model of posture controller.

Function in posture controller	Stiffness	Damping
Generation of vertical reference force	k_B	d_B
Generation of reference torque	k_α	d_α
Control of horizontal foot position	k_L	d_L

There is no lowering of the height of the trunk due to gravity because of the compensation term in the posture controller. The linearized system around this equilibrium looks like this:

$$\begin{aligned} I_y\dot{w}_1 &= -2\left(\frac{I_y}{m}+h^2\right)(k_L q_1 + d_L w_1) + h\,(q_2 k_\alpha + d_\alpha w_2) - mghq_1 + gI_y q_2 \\ I_y\dot{w}_2 &= -d_q w_2 + 2hk_L q_1 + mgq_1 - k_q q_2 + 2d_L u_1 h \\ m\dot{w}_3 &= -k_B q_3 - d_B w_3 \end{aligned}$$

where the control parameters are explained in table 11.1.

Two eigenvalues are easy to find for this system:

$$\lambda_{1,2} = \frac{-d_B \pm \sqrt{d_B^2 - 4mk_B}}{2m}$$

but the remaining four eigenvalues are once more roots of a large fourth degree polynomial. The necessary criterion that the 0^{th} coefficient is positive is this time

$$2k_L k_\alpha - 2hk_L mg - (mg)^2 > 0$$

which requires a positive horizontal stiffnes, i.e.

$$k_L > 0.$$

Solving the inequality for q_α gives this result:

$$k_\alpha > mgh + \frac{(mg)^2}{2k_L}.$$

Inserting the parameters actually used in the experiments that will be described in the next section, show that the inequality is satisfied. For roll motions, k_α is about five times larger than required, and for pitch motions it is about 30 times larger (different parameter values are used in the two directions). Note that these control parameters were not just tuned to be able to stand up, but also to track steps in the references. Furthermore, the parameters were not extensively tuned for the experiments.

11.3. Balance experiments

All of the experiments were performed with the same control parameters, only the reference trajectories differ. Step changes are applied to the posture references in the first three experiments. In the last experiment, the posture reference is constant, but the robot is standing on a balance board that is tilted.

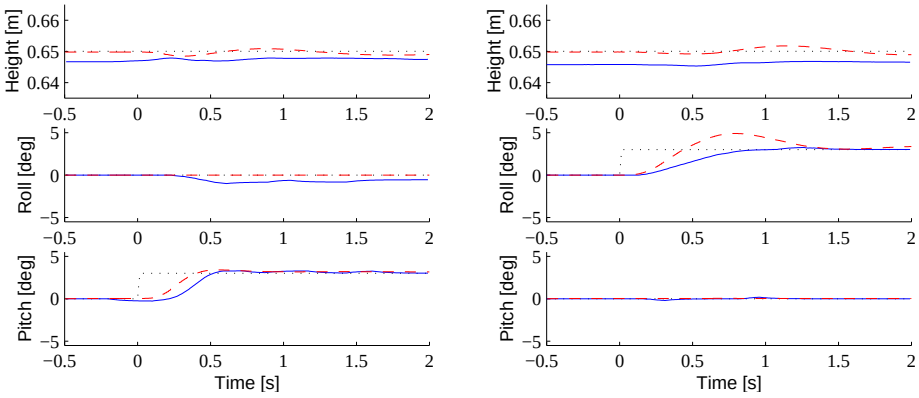
11.3.1. Step response of the pitch and roll angle

In these two experiments we applied a change to the pitch and roll reference angles (figure 11.3a and b). The pitch response is almost twice as fast as the roll response. This could be due to the forward offset of the robot's centre of mass. During the step in pitch angle, there is a deviation ($< 1^o$) on the roll angle due to coupling. The error is slowly corrected by the integration term. There is a similar (but smaller) deviation on the pitch angle during the step in the roll angle. The height deviation is small (<5 mm) in both experiments and simulation. There are only small deviations in the roll/pitch angles in simulation. It seems reasonable that the friction at least partly causes the behaviour in the experiments. Another cause could be the fact that the individual joints have quite different mechanical characteristics, e.g. friction. Friction is probably also the reason why the response in simulation is faster (and overshoots in 11.3b). The parameters where tuned for the experiments and not the simulation.

11.3.2. Step response of the height

Figure 11.4a illustrates the response to a change in step height. In the experiment, the robot has not quite reached its desired height before the reference step occurs. Then, after the initial response, the height increases more or less linearly as the integration term grows. We believe this behaviour is caused by friction in the joints combined with an integration term that is too small. The total power consumed by the actuators during the initial response rise from about 16 W to a peak of 90 W and then drops down to 9 W. Less power is consumed afterwards, because the knees are less bent. The increased potential energy accounts for about half of the used energy, the other half is probably dissipated by friction and resistance in the motor coils. There was no friction included in the simulation, which might account for the differences.

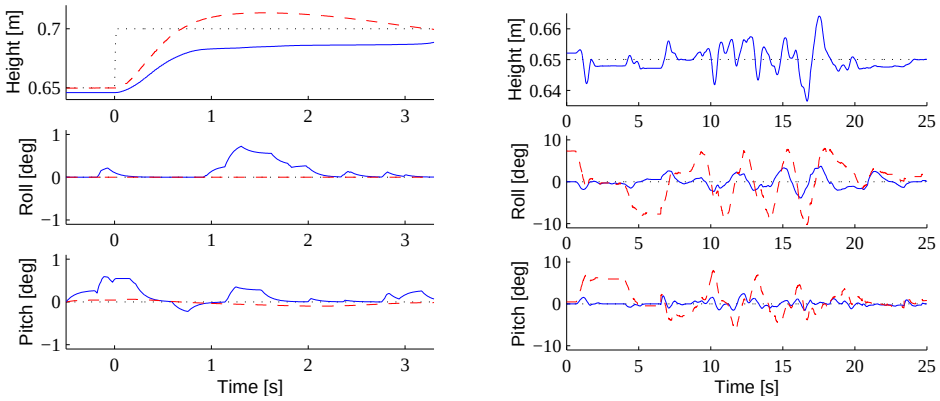
There are some disturbances in the roll and pitch angles in the experiment, with less deviations on the pitch angle. There are also small deviations in these angles on the simulated motions ($< 0.1\,^\circ$).



(a) Step in reference, pitch angle

(b) Step in reference, roll angle

Figure 11.3: a) Posture response to a step change in reference pitch angle (dotted line). Solid lines are experimental data and the dashed lines are data from simulation. b) Posture response to a step change in reference roll angle (dotted line). Solid lines are experimental data and the dashed lines are data from simulation.



(a) Step in reference height

(b) Response on balance board

Figure 11.4: a) Posture response to a step change in reference height (dotted line). Solid lines are experimental data and the dashed lines are data from simulation. b) Posture response (solid lines) to large disturbances of the balance board (dashed lines).

11.3.3. Standing on a balance board

The robot is placed on a 2×2 m balancing board (figure 11.2) that has a 0.13 m long rod attached underneath. This allows the board to rotate around the rod's ground contact point. The board is held by hand during the experiment and manually rocked back and forth along the board's pitch and roll axes. The board orientation is estimated using inverse leg kinematics and the estimated trunk orientation.

Figure 11.4b) shows the posture response from pitching and rolling board motions of about $\pm 8^\circ$ with velocities up to 20°/s. It looks good (see *http://www.md.kth.se/~cas/movies/balance)* and the pitch error is smaller here as well. That might partly be due to the slightly smaller disturbances.

12. Summary and discussion

We begin by briefly summarizing the results in this part and then discussing them. Finally results from the other parts are discussed.

12.1. Summary of stability analysis

In chapter 9 it was analytically shown that a planar symmetric statically balanced stance on a compliant surface is actually only stable if

$$\frac{mgh}{2kr^2} = a < 1. \tag{12.1}$$

This criterion was verified in experiments with a special test rig. The question of domain of attraction was also investigated through simulations, where it was shown that the compliance in the ground also decreases the domain of attraction. However, that effect is minor unless the compliance is large.

Chapter 10 then extended the planar case to 3D for radially symmetric stances, showing stability if:

$$a_L < 1 \tag{12.2}$$

for two, three- and four legs ($L = 2,\ 3$ and 4). A more intuitive proof of the planar case was also given, as well as an interpretation of a_L, as the ratio between stiffness due to gravity, and stiffness due to ground compliance. The asymmetric planar case was also investigated and the surface of possible equilibria was visualized in 3D as a function of two dimensionless parameters (a_2 and $r\gamma$). After separating the equilibrium surface into stable and unstable equilibria, it was discussed how a robot moving slowly will be statically balanced, and its motion will therefore correspond to a curve on the equilibrium surface.

In chapter 11 the ground was no longer considered to be the cause of compliance. Instead, it originates with the robot controller and two controllers were discussed: a

Cartesian position control of the feet; and a posture controller that uses force control in the vertical direction. Both controllers were analyzed for a planar robot model in symmetric stance on stiff ground. Here the eigenvalues were not useful, but a necessary condition for stability was found. This criterion must be satisfied for the Cartesian position control to be stable:

$$\frac{1}{\acute{k}_1} < \frac{1 + 2\acute{k}_3 \left(2\acute{k}_3 \frac{r^2}{h^2} - 1 \right)}{\acute{k}_3}$$

where $\acute{k}_1$ and $\acute{k}_3$ are the dimensionless horizontal and vertical control stiffnesses. The corresponding criterion for the posture controller is

$$k_\alpha > mgh + \frac{(mg)^2}{2k_L}$$

where k_α is the angular stiffness of the trunk control, and k_L is the horizontal stiffness of the control of foot position. Both criteria were compared with actual control values that have been used with WARP1, and both criteria were satisfied.

12.2. Discussion of stability analysis

One way the results for a compliant surface could be used is to include them in the process of gait planning that is based on the static balance criterion. The criterion could be used to give an upper boundary on the allowed height, or a lower boundary on the width of the stance. Using this for planning would require an estimate of the compliance that can be difficult to obtain — especially for areas where the robot has not been before. Fortunately the support width dominates over the stiffness, e.g. a 20% decrease in stiffness is compensated by a 11% increase in width. For robots with a low centre of mass, this will probably not be a problem. A biped on the other hand, usually has a high centre of mass and a narrower stance, especially a biped standing on one foot.

A more important use for this criterion is that it indicates when static balance is not enough and an active balance is needed. Note however that this shifts the problem to the balance controller, which in the case of the posture controller have to satisfy a similar criterion.

The results for the controllers mean that low control stiffness can cause problems, but unfortunately it is not possible to arbitrarily increase the control stiffness. For the posture controller, using too large parameters made it unstable and for the

position control chattering occurred. Furthermore, it is not always desired to position control the feet very stiffly since this could cause slippage and a buildup of internal forces when walking on rough terrain.

12.2.1. Implications for ZMP

The ZMP criterion (see section 2.5.2) is often used as a modern version of the static balance criterion. It is used to plan a reference trajectory where the ZMP remains in the interior of the support area. However, the static balance criterion for a compliant surface (12.2) implies that a robot whose reference trajectory satisfies the ZMP criterion can also fall. The proof is in fact identical, since a planned motion for the robot to stand still corresponds to static balance. Even simpler, any normal rigid object placed on a sufficiently compliant surface can tip over.

12.2.2. Ideas for the future

This part has barely scratched the surface of possible results. It could be fruitful to create a kind of library with this kind of criteria for various "standard" cases such as soft terrain and different controllers. Especially if there is some way in which the criteria can be combined when there are several sources of compliance. In fact, studying what happens in this case is one of the more important things that should be investigated. For further research in this direction, it is probably useful to look at the results on the properties of spatial stiffness matrices [23], and at the concept of compliant grasps and pasive force closure [167]. Related to the idea that it is enough to study the stiffness to determine stability, a paper by Koditschek's [94] is a good start.

Another thing that should be investigated is how the type of ground affects the results. What happens on real terrain that is not linearly elastic? The local stability analysis holds of course, but it is not obvious that the robot will fall over (although it's unclear what the alternative could be, a limit cycle for instance is unlikely without energy being added to the system). As for the analysis of compliance in the controllers, it needs to be extended to asymmetric configurations and general foot placements.

Other questions that should be investigated include:

- How does removing the assumption of massless legs affect the results?
- How robust are the results to modeling errors?

Finally, it would be very interesting to see how compliance affects the stability of motions that have been planned using a ZMP criterion. When the planned motion no

longer is static, does this require a stiffer ground than just standing still? One way to answer this question is to run simulations of a planar system such as in figure 9.2, but let h and γ be functions of time (this corresponds to the trunk moving relative to the feet).

12.3. Summary and discussion of part I-III

Part I gave an introduction to the field of walking robots, where information was collected from various sources and descriptions of controllers for walking robots were presented in a more coherent framework. Thus making it easier to compare and understand how the controllers work. The main result from this part is in my opinion the choice of questions/aspects for how to analyze and try to understand the controllers: Generation of trunk motion; Maintaining balance, Generation of leg sequence and support patterns; and Reflexes. A big problem was that articles very rarely contained complete information. Using several papers as a source for one controller gives a more complete picture, but also introduces discrepancies since details in the controller and/or the robot have changed.

In part II, an attempt has been made to provide a comprehensive and detailed description of our four-legged robot in terms of mechanics, electronics and basic control structure. However, it probably does still not contain enough information for someone else to easily reproduce simulations and experimental results. The complexity of the system is one of the reasons for this. As an example, the dynamic model is simply too large to fit in an article, the C-file used to export it contains almost 10000 lines.

Because of the complexity, it is also impossible to derive the model by hand so instead a description is given of how it is derived using a computer algebra system (cas). Since it is basically a program that derives the model, it was possible to make the program more general. In this case, it is easy to make the program derive a model for certain classes of robots.

The complexity is also a reason why we need to use advanced tools for working with this system. These are the main tools and methods that have been used:

- A computer algebra system (Maple) with the Sophia language to derive kinematic and dynamic models of mechanical systems.
- A graphical environment to design, model and simulate dynamical systems (MATLAB/Simulink).
- A rapid prototyping tool that automatically implements a graphically designed controller by converting it into C code (Real-Time Workshop).

However, the tools need to interface with each other, and it is not practical to manually enter a model as large as the WARP1 rigid body model. Fortunately, we found a tool (`exmex()`) that could do this. Complexity also caused problems within MATLAB and Simulink because so many signals and parameters are used. It was not practical to manually manage the order with which these are used, so extra functionality was added to do this for us. As an added benefit, it is now also much easier specify, transfer and use symbolical expressions derived in Maple with MATLABSimulink. In fact, creating relatively simple expressions is often quicker to derive symbolically and then transfer, than enter by hand. Additionally, not having to enter large expressions by hand reduce the risk of entering them incorrectly.

On the other hand, there is actually a drawback with using derived expressions. If there is an error in the original expression, there will most likely be an error in the derived expression. What you loose is the independent implementation of the same function, that allow you to compare the results. It is also the comparison of independent results that makes it so important to be able to compare results from simple analytic models, with complicated numerical simulations and with experimental results.

This combination was also very useful in the last part, that is theoretical rather than about control design. For instance, the computer algebra system allows you to work with large and complicated differential equations. The numerical tool is necessary to interpret and visualize the results when you end up with a complicated result.

12. Summary and discussion

Part V.

Appendices, index and bibliography

A. Publications and division of work

Below is a lists publications where I am the primary author or a co-author. I have been deeply involved in the writing process of all the papers except [10], where my contribution is minor. I am the only author and contributor to [156–158]. The main contribution to [159] is mine, but I wrote it with Ingvast. We also wrote [155] and [84] about WARP1, that is the result of several persons combined effort. My contribution to [154] and [5] is writing and experiments.

Publications

[1] Christian Ridderström. Stability of statically balanced, radially symmetric stances for legged robots on compliant surfaces. In *Int. Conf. on Climbing and Walking Robots*, Paris, France, September 2002.

[2] Christian Ridderström. Stability of statically balanced stances for legged robots with compliance. In *Int. Conf. on Robotics and Automation*, Washington DC, USA, 2002.

[3] J. Ingvast, C. Ridderström, and J. Wikander. The four legged robot system WARP1 and its capabilities. In *Second Swedish Workshop on Autonomous Systems*, Stockholm, Sweden, October 2002. See `http://www.md.kth.se/~cas/publications`.

[4] Christian Ridderström and Johan Ingvast. Combining control design tools — from modeling to implementation. In *Int. Conf. on Robotics and Automation*, pages 1327–1333, 2001.

[5] J. Ingvast, C. Ridderström, F. Hardarson, and J. Wikander. Improving a trotting robot's gait by adapting foot trajectory offsets. In *Int. Conf. on Climbing and Walking Robots*, pages 711–718, Karlsruhe, Germany, September 2001.

[6] Christian Ridderström and Johan Ingvast. Quadruped posture control based on simple force distribution — a notion and a trial. In *Int. Conf. on Intelligent Robots and Systems*, pages 2326–2331, 2001.

[7] C. Ridderström, J. Ingvast, F. Hardarson, M. Gudmundsson, M. Hellgren, J. Wikander, T. Wadden, and H. Rehbinder. The basic design of the quadruped robot Warp1. In *Int. Conf. on Climbing and Walking Robots*, Madrid, Spain, October 2000.

[8] Christian Ridderström. Legged locomotion control — a literature survey. Technical Report TRITA-MMK 1999:27, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, November 1999. ISSN 1400-1179.

[9] Henrik Rehbinder and Christian Ridderström. Attitude estimation for walking robots. In *Int. Conf. on Climbing and Walking Robots*, September 1999.

[10] Freyr Hardarson, Bengt Eriksson, Christian Ridderström, Tom Wadden, and Jan Wikander. Experiments with impedance control of a single compliant leg. In *Int. Conf. on Climbing and Walking Robots*, September 1999.

B. Special references

⋆ R. McNeil Alexander. *Optima for animals*, chapter 3, pages 45–64. Princeton University Press, revised edition, 1996.

⋆ M. G. Bekker. *Introduction to Terrain-Vehicle Systems*. University of Michigan Press, Ann Arbor, 1969.

⋆ Matthew D. Berkemeier. Modeling the dynamics of quadrupedal running. *Int. J. of Robotics Research*, 17(9):971–985, 1998.

⋆ Karsten Berns. The walking machine catalogue. `http://www.fzi.de/divisions/ipt/WMC`.

⋆ J. Furusho and A. Sano. Sensor-based control of a nine-link biped. *Int. J. of Robotics Research*, 9(2):83–98, April 1990.

⋆ D. M. Gorinevsky and A. Yu. Shneider. Force control in locomotion of legged vehicles over rigid and soft surfaces. *Int. J. of Robotics Research*, 9(2):4–23, April 1990.

⋆ A. Goswami. Foot rotation indicator (FRI) point: A new gait planning tool to evaluate postural stability of biped robots. In *Int. Conf. on Robotics and Automation*, pages 47–52, Detroit, MI, may 1999.

⋆ S. Hirose, Y. Fukuda, and H. Kikuchi. The gait control system of a quadruped walking vehicle. *Advanced Robotics*, 1(4):289–323, 1986.

⋆ Neville Hogan. Impedance control: An approach to manipulation. *J. Dynamic Systems, Measurement and Control*, 107(1):1–20, 1985.

⋆ C. A. Klein, K. W. Olson, and D. R. Pugh. Use of force and attitude sensors for locomotion of a legged vehicle over irregular terrain. *Int. J. of Robotics Research*, 2(2):3–17, 1983.

⋆ Martin Lesser. *The analysis of complex nonlinear mechanical systems*. World Scientific, P O Box 128, Farrer Road, Singapore 9128, 1995.

⋆ T. McGeer. Passive dynamic walking. *Int. J. of Robotics Research*, 9(2):62–82, 1990.

⋆ R. B. McGhee. Some finite state aspects of legged locomotion. *J. Math. Biosciences*, 2:67–84, 1968.

⋆ R. B. McGhee and A. A. Frank. On the stability properties of quadruped creeping gaits. *J. Math. Biosciences*, 3:331–351, 1968.

⋆ Domini A. Messuri and Charles Klein. Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion. *IEEE Robotics and Automation Magazine*, RA-1(3), September 1985.

⋆ Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

⋆ E. Muybridge. *Animals in Motion*. New Dover Edition, Dover Publications, Inc., New York, 1957. First published in 1899.

⋆ Marc H. Raibert. *Legged robots that balance*. The MIT Press, 1986.

⋆ Shin-Min Song and Kenneth J. Waldron. *Machines that Walk*. MIT Press, Cambridge, MA, 1989.

⋆ D. J. Todd. *Walking machines – an introduction to legged robots*. Kogan Page Ltd, London, 1986.

⋆ M. Vukobratovic, A. A. Frank, and D. Juricic. On the stability of biped locomotion. *IEEE Trans. Biomedical Eng.*, BME-17(1):25–36, January 1970.

⋆ M. Vukobratovic and J. Stepanenko. On the stability of anthropomorphic systems. *J. Math. Biosciences*, 15:1–37, 1972.

⋆ M. Vukobratovic and J. Stepanenko. Mathematical models of general anthropomorphic systems. *J. Math. Biosciences*, 17:191–242, 1973.

Index

References

[1] LyX the document processor. `http://www.lyx.org`.

[2] M. Ahmadi and M. Buehler. Stable control of a simulated one-legged running robot with hip and leg compliance. *IEEE-Transactions-on-Robotics-and-Automation*, 13(1):96–104, February 1997.

[3] R. McN. Alexander. Optimum walking techniques for quadrupeds and bipeds. *J. of Zoology (London)*, 192:97–117, 1980.

[4] R. McN. Alexander. The gaits of bipedal and quadrupedal animals. *Int. J. of Robotics Research*, 3(2):49–59, 1984.

[5] R. McN. Alexander. Optimization and gaits in the locomotion of vertebrates. *Phys. Rev.*, 69(4):1199–1227, 1989.

[6] R. McNeil Alexander. *Optima for animals*, chapter 3, pages 45–64. Princeton University Press, revised edition, 1996.

[7] R. McNeill Alexander. Vertical movements in walking and running. *J. of Zoology (London)*, 185:27–40, 1978.

[8] M. G. Bekker. *Introduction to Terrain-Vehicle Systems*. University of Michigan Press, Ann Arbor, 1969.

[9] Karim Benjelloun. Design and construction of the M3L robot leg. Technical report, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1997. TRITA-MMK 1997:21, ISSN 1400-1179, ISRN KTH/MMK–97/21–SE.

[10] Matthew D. Berkemeier. Modeling the dynamics of quadrupedal running. *Int. J. of Robotics Research*, 17(9):971–985, 1998.

[11] K. Berns, W. Ilg, M. Deck, and R. Dillman. The mammalian-like quadrupedal walking machine BISAM. In *Int. Workshop on Advanced Motion Control Proc.*, pages 429–433, 1998.

[12] Karsten Berns. The walking machine catalogue. `http://www.fzi.de/divisions/ipt/WMC`.

[13] Karsten Berns, Rüdiger Dillmann, and Stefan Pekenbrock. Neural networks for the control of a six-legged walking machine. *Robotics and Autonomous Systems*, 14:233–244, 1995.

[14] Biological Cybernetics. `http://www.uni-bielefeld.de/biologie/Kybernetik/research/walk.html`.

[15] A. J. Van Den Bogert, H. C. Schamhadt, and A. Crowe. Simulation of quadrupedal locomotion using a rigid body model. *J. of Biomechanics*, 22(1):33–41, 1988.

[16] Ben Brown and Garth Zeglin. The bow leg hopping robot. In *Int. Conf. on Robotics and Automation*, pages 781–786, 1998.

[17] M. Buehler, R. Battaglia, A. Cocosco, G. Hawker, J. Sarkis, and K. Yamazaki. SCOUT: A simple quadruped that walks, climbs, and runs. In *Int. Conf. on Robotics and Automation*, pages 1707–1712, 1998.

[18] Centre for Autonomous Systems. `http://www.cas.kth.se`. Numerisk Analys och Datalogi, Kungliga Tekniska Högskolan, S-100 44 Stockholm, Sweden.

[19] Chun-Hung Chen, Vijay Kumar, and Yuh-Chyuan Luo. Motion planning of walking robots using ordinal optimization. *IEEE Robotics and Automation Magazine*, June 1998.

[20] Jeng-Shi Chen, Fan-Tien Cheng, Kai-Tarng Yang, Fan-Chu Kung, and York-Yih Sun. Optimal force distribution in multilegged vehicles. *Robotica*, 17(pt.2):159–172, March-April 1999.

[21] Jeng-Shi Chen, Fan-Tien Cheng, Kai-Tarng Yang, Fan-Chu Kung, and York-Yin Sun. Solving the optimal force distribution problem in multilegged vehicles. In *Int. Conf. on Robotics and Automation*, pages 471–476, 1998.

[22] D. J. Cho, J. H. Kim, and D. G. Gweon. Optimal turning gait of a quadruped walking robot. *Robotica*, 13(6):559–564, 1995.

[23] Namik Ciblak. *Analysis of Cartesian stiffness and compliance with applications*. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia 30332, May 1998.

[24] T. Karčnik and A. Kralj. Gait dynamic stability assessment in a sagittal plane. In *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine., 18th Annual International Conference of the IEEE*, volume 2, pages 467–468, November 1996.

[25] Jefferson A. Coelho and Roderic A. Grupen. Online grasp synthesis. In *Int. Conf. on Robotics and Automation*, pages 2137–2142, 1996.

[26] J. J. Collins and S. A. Richmond. Hard-wired central pattern generators for quadrupedal locomotion. *Biological Cybernetics*, 71:375–385, 1994.

[27] J. J. Collins and I. Stewart. Hexapodal gaits and coupled nonlinear oscillator models. *Biological Cybernetics*, 68:287–298, 1993.

[28] Computer Aided Software Inc. DADS. `http://www.cadsi.com`.

[29] C. I. Connolly and R. A. Grupen. The applications of harmonic functions to robotics. *J. Robotic Systems*, 10(7):931–946, October 1993.

[30] H. Cruse, Ch. Bartling, J. Dean, T. Kindermann, J. Schmitz, M. Schumm, and H. Wagner. Coordination in a six-legged walking system. Simple solutions to complex problems by exploitation of physical properties. In Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan Pollack, and Stewart W. Wilson, editors, *From animals to animats*, volume 4, pages 84–93. MIT Press, 1996.

[31] H. Cruse, D. E. Brunn, Ch. Bartling, J. Dean, M. Dreifert, T. Kindermann, and J. Schmitz. Walking: A complex behavior controlled by simple networks. *Adaptive Behavior*, 3(4):385–418, 1995.

[32] Holk Cruse, Christian Bartling, and Thomas Kindermann. High-pass filtered positive feedback for decentralized control of cooperation. In F. Moran, A. Moreno, J. J. Merelo, and P. Chacon, editors, *Advances in Artificial Life*, pages 668–678. Springer-Verlag, 1995.

[33] Harry Dankowicz, Jesper Adolfsson, and Arne B. Nordmark. Existence of stable 3d-gait in passive bipedal mechanisms. *Trans. ASME J. Biomechanical Engineering*, ?(?):?, 1999. Subm. to.

[34] Pablo Gonzalez de Santos and Maria A. Jimenez. Generation of discontinuos gaits for quadruped walking vehicles. *J. Robotic Systems*, 12(9):599–611, 1995.

[35] Chang de Zhang and S. M. Song. Turning gait of a quadrupedal walking machine. In *Int. Conf. on Robotics and Automation*, volume ?, pages 2106–2112, 1991.

[36] Deneb Robotics Inc. Envision. `http://www.deneb.com`.

[37] dSPACE GmbH. dSPACE. `http://www.dspace.de`.

[38] Takashi Emura and Akira Arakawa. Attitude control of a quadruped robot during two legs supporting. In *Int. Conf. on Adv. Rob.*, pages 711–716, 1991.

[39] Bengt Eriksson. A survey on dynamic locomotion control strategies for legged vehicles. Technical Report TRITA-MMK 1998:1, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1998. ISSN 1400-1179.

[40] El F. Hafi and P. Gorce. Walking dynamic control under unknown perturbation. In *Int. Conf. on Systems, Man and Cybernetics*, pages 3538 – 3543, 1998.

[41] Johan Friede and Kim Kylström. Determination of mechanical parameters for the warp 1 robot using experimental methods, simulations and real-time control. Master's thesis, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1998. TRITA-MMK 1998:52MME666, ISSN 1400-1179, ISRN KTH/MMK–98/9–SE.

[42] M. Frik. Adaptive neural control of a walking machine. In *7th German-Japanese Seminar on Nonlinear Problems in Dynamical Systems — Theory and Applications*, 1996.

[43] J. Furusho and M. Masubuchi. A theoretically motivated reduced order model for the control of dynamic biped locomotion. *J. Dynamic Systems, Measurement and Control*, 109:155–163, June 1987.

[44] J. Furusho and A. Sano. Sensor-based control of a nine-link biped. *Int. J. of Robotics Research*, 9(2):83–98, April 1990.

[45] X. C. Gao and S. M. Song. Stiffness matrix method for foot force distribution of walking vehicles. In *Int. Conf. on Robotics and Automation*, volume 3, pages 1470–1475, 1990.

[46] J. F. Gardner. Force distribution in walking machines over rough terrain. *J. Dynamic Systems, Measurement and Control*, 113:754–758, 1991.

[47] John F. Gardner. Characteristics and approximations of optimal force distributions in walking machines on rough terrain. In *Int. Conf. on Advanced Robotics*, pages 613–618, 1991.

[48] F. Génot and B. Espiau. On the control of the mass center of legged robots under unilateral constraints. In *CLAWAR '98 First International Symposium*, pages 9–14, 1998.

[49] Martin Golubitsky, Ian Stewart, Pietro-Luciano Buono, and J. J. Collins. A modular network for legged locomotion. *Physica D*, 115:56–72, 1998.

[50] Bill Goodwine and Joel Burdick. Trajectory generation for kinematic legged robots. In *Int. Conf. on Robotics and Automation*, pages 2689–2696, 1997.

[51] Bill Goodwine and Joel Burdick. Gait controllability for legged robots. In *Int. Conf. on Robotics and Automation*, pages 484–489, 1998.

[52] P. Gorce. Dynamic control of bipeds using postural adjustment strategy. In *Int. Conf. on Systems, Man and Cybernetics*, pages 453–458, 1997.

[53] P. Gorce and El F. Hafi. Modelling of human body control scheme and learning in stepping motion over an obstacle. In *Int. Conf. on Intelligent Robots and Systems*, pages 64–69, 1998.

[54] P. Gorce and M. Guihard. On dynamic control of pneumatic bipeds. *J. Robotic Systems*, 15(7):421–433, 1998.

[55] P. Gorce, O. Vanel, and C. Ribreau. Equilibrium study of "human" robot. In *Int. Conf. on Systems, Man and Cybernetics*, pages 1309–1314, 1995.

[56] P. Gorce, C. Villard, and J. G. Fontaine. Grasping, coordination and optimal force distribution in multifinger mechanisms. *Robotica*, 12:243–251, 1994.

[57] D. M. Gorinevsky and A. Yu. Shneider. Force control in locomotion of legged vehicles over rigid and soft surfaces. *Int. J. of Robotics Research*, 9(2):4–23, April 1990.

[58] A. Goswami. Foot rotation indicator (FRI) point: A new gait planning tool to evaluate postural stability of biped robots. In *Int. Conf. on Robotics and Automation*, pages 47–52, Detroit, MI, may 1999.

[59] M. Guihard and P. Gorce. Joint impedance control applied to a biped pneumatic leg. In *Int. Conf. on Systems, Man and Cybernetics*, pages 1114–1119, 1996.

[60] M. Guihard and P. Gorce. A new way to tackle position/force control for pneumatic robots. In *Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 603–610, 1996.

[61] M. Guihard, P. Gorce, and J. G. Fontaine. SAPPHYR: legs to pull a wheel structure. In *Int. Conf. on Systems, Man and Cybernetics*, volume 2, pages 1303–1308, 1995.

[62] Freyr Hardarson. Locomotion for difficult terrain. Technical Report TRITA-MMK 1998:3, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, April 1998. ISSN 1400-1179, ISRN KTH/MMK–98/9–SE.

[63] Christian Hardell. An integrated system for computer aided design and analysis of multibody systems. *Engineering with Computers*, 12(1):23–33, 1996.

[64] Simon Haykin. *Neural Networks*. Prentice Hall, 1994.

[65] M. Hildebrand. Symmetrical gaits of horses. *Science*, 150:701–708, 1965.

[66] S. Hirose, Y. Fukuda, and H. Kikuchi. The gait control system of a quadruped walking vehicle. *Advanced Robotics*, 1(4):289–323, 1986.

[67] S. Hirose, H. Kikuchi, and Y. Umetani. The standard circular gait of a quadruped walking vehicle. *Advanced Robotics*, 1(2):143–164, 1986.

[68] S. Hirose, T. Masui, H. Kikuchi, and Y. Fukuda. TITAN III a quadruped walking vehicle — its structure and basic characteristics. In *2ndISRR*, pages 325–331, 1985.

[69] S. Hirose, K. Yoneda, and Ibe Arai. Design of prismatic quadruped walking vehicle TITAN VI. In *5th Int. Conf. on Advanced Robotics*, pages 732–738, 1991.

[70] S. Hirose, K. Yoneda, K. Arai, and T. Ibe. Design of a quadruped walking vehicle for dynamic walking and stair climbing. *Advanced Robotics*, 9(2):107–124, 1995.

[71] S. Hirose, K. Yoneda, R. Furuya, and T. Takagi. Dynamic and static fusion control of quadruped walking vehicle. In *Int. Conf. on Intelligent Robots and Systems*, pages 199–204, 1989.

[72] Shigeo Hirose. A study of design and control of a quadruped walking vehicle. *Int. J. of Robotics Research*, 3(2):113–133, 1984.

[73] Shigeo Hirose and Osamu Kunieda. Generalized standard foot trajectory for a quadruped walking vehicle. *Int. J. of Robotics Research*, 10(1):3–12, February 1991.

[74] Shigeo Hirose, Hideyuki Tsukagoshi, and Kan Yoneda. Normalized energy stability margin: Generalized stability criterion for walking vehicles. In *CLAWAR '98 First International Symposium*, pages 71–76, 1998.

[75] Shigeo Hirose, Hideyuki Tsukagoshi, and Kan Yoneda. Normalized energy stability margin and its contour of walking vehicles on rough terrain. In *Int. Conf. on Robotics and Automation*, pages 181–186, 2001.

[76] Shigeo Hirose and Kan Yoneda. Toward development of practical quadruped walking vehicles. *J. of Robotics and Mechatronics*, 5(6):494–504, 1993.

[77] Hirose and Yoneda Lab. `http://mozu.mes.titech.ac.jp/`.

[78] Neville Hogan. Impedance control: An approach to manipulation. *J. Dynamic Systems, Measurement and Control*, 107(1):1–20, 1985.

[79] Manfred Huber and Roderic A. Grupen. A hybrid discrete event dynamics systems approach to robot control. Technical Report 96–43, Dept. of Computer Science, Univ. of Massachusetts, USA, October 1996.

[80] Manfred Huber and Roderic A. Grupen. A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems*, 22(4–3):303–15, December 1997.

[81] Manfred Huber and Roderic A. Grupen. A control structure for learning locomotion gaits. In *7th Int. Symposium on Robotics and Applications*, Anchorage, AK, 1998. TSI Press.

[82] Min-Hsiung Hung, D. E. Grin, and K. J. Waldron. Force distribution equations for general tree-structured robotic mechanisms with a mobile base. In *Int. Conf. on Robotics and Automation*, volume 4, pages 2711–2716, 1999.

[83] Winfried Ilg and Karsten Berns. A learning architecture based on reinforcement learning for adaptive control of the walking machine LAURON. *Robotics and Autonomous Systems*, 15:321–334, 1995.

[84] J. Ingvast, C. Ridderström, and J. Wikander. The four legged robot system WARP1 and its capabilities. In *Second Swedish Workshop on Autonomous Systems*, Stockholm, Sweden, October 2002. See `http://www.md.kth.se/~cas/publications`.

[85] Integrated Systems Inc. MATRIXx. `http://www.isi.com`.

[86] Satoshi Ito and Haruhisa Kawasaki. A standing posture control based on ground reaction force. In *Int. Conf. on Intelligent Robots and Systems*, 2000.

[87] K. Jeong, T. Yang, and J. Oh. A study on the support pattern of a quadruped walking robot for aperiodic motion. In *Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 308–313, Pittsburgh, PA, August 1995.

[88] Kyung-Min Jeong and Jun-Ho Oh. An aperiodic z type spinning gait planning method for a quadruped walking robot. *Autonomous Robots*, 2(2):163–173, 1995.

[89] Shuuji Kajita and Kazuo Tanie. Experimental study on biped dynamic walking in the linear inverted pendulum mode. In *Int. Conf. on Robotics and Automation*, pages 2885–2891, 1995.

[90] Kane and Levinson. *Dynamics: Theory and application*. McGraw-Hill, 1985. ISBN 0-07-037846-0.

[91] Shigeyasu Kawaji, Ken'ichi Ogasawara, and Masaki Arao. Rhytm-based control of biped locomotion robot. In *Int. Workshop on Advanced Motion Control Proc.*, pages 93–97, 1998.

[92] C. A. Klein and S. Kittivatcharapong. Optimal force distributions for the legs of a walking machine with friction cone constraints. *IEEE Robotics and Automation Magazine*, 6(1):73–85, 1990.

[93] C. A. Klein, K. W. Olson, and D. R. Pugh. Use of force and attitude sensors for locomotion of a legged vehicle over irregular terrain. *Int. J. of Robotics Research*, 2(2):3–17, 1983.

[94] D. Koditschek. The application of total energy as a lyapunov function for mechanical control systems, 1989.

[95] E. I. Kugushev and V. S. Jaroshevskij. Problems of selecting a gait for an integrated locomotion robot. In *4th Int. Conf. Artificial Intell.*, pages 789–793, Tbilisi, Georgian SSR, USSR, 1975.

[96] Vijay R. Kumar and Kenneth J. Waldron. Force distribution in closed kinematic chains. *IEEE Robotics and Automation Magazine*, 4(6):657–664, December 1988.

[97] Vijay R. Kumar and Kenneth J. Waldron. Adaptive gait control for a walking robot. *J. Robotic Systems*, 6(1):49–76, 1989.

[98] A. Kun and W. T. Miller, III. Adaptive dynamic balance of a biped robot using neural networks. In *Int. Conf. on Robotics and Automation*, pages 240–245, 1996.

[99] A. L. Kun and W. T. Miller, III. Unified walking control for a biped robot using neural networks. In *IEEE ISIC/CIRA/ISAS Joint conf.*, pages 283–288, Gaithersburg, MD, September 1998.

[100] Kungliga Tekniska Högskolan. (Royal Institute of Technology). `http://www.kth.se`. S-100 44 Stockholm, Sweden.

[101] Laboratory for Perceptual Robotics, Univ. of Massachusetts. `http://www-robotics.cs.umass.edu/thing`.

[102] Anna-Karin Larde, Mattias Olsson, Kennet Jansson, Lars Wallentin, and Sören Andersson. Sleipner 3 — a student project in machine elements and mechatronics. Technical Report TRITA-MMK 1998:12, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1998. ISSN 1400-1179.

[103] Kwan-Pyo Lee, Tae-Wan Koo, and Yong-San Yoon. Real-time dynamic simulation of quadruped using modified velocity transformation. In *Int. Conf. on Robotics and Automation*, pages 1701–1706, 1998.

[104] Tsu-Tian Lee and Ching-Long Shih. A study of the gait control of a quadruped walking vehicle. *IEEE Robotics and Automation Magazine*, RA-2(2):61–69, June 1986.

[105] Anders Lennartsson. *Efficient Multibody Dynamics*. PhD thesis, Dept. of Mechanics, KTH, 100 44 Stockholm, Sweden, 1999.

[106] Martin Lesser. *The analysis of complex nonlinear mechanical systems*. World Scientific, P O Box 128, Farrer Road, Singapore 9128, 1995.

[107] Fredrik Liander and Johan Wallström. Development of a robot prototype with wheeled locomotion for difficult terrain. Master's thesis, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1997. TRITA-MMK 1997:? MME650.

[108] B.-S. Lin and S.-M. Song. Dynamic modeling, stability and energy efficiency of a quadrupedal walking machine. In *Int. Conf. on Robotics and Automation*, pages 367–373, 1993.

[109] Yi Lin and Shin-Min Song. Learning hybrid position/force control of a quadruped walking machine using a CMAC neural network. *J. Robotic Systems*, 14(6):483–499, 1997.

[110] Willard S. MacDonald. Design and implementation of a four-legged walking robot. Senior honors thesis, (ece) dept., Univ. of Massachusetts, USA, may 1994.

[111] Willard S. MacDonald. Legged locomotion over irregular terrain using the control basis approach. Master's thesis, University of Massachusetts Amherst, USA, 1996.

[112] Willard S. MacDonald and Roderic A. Grupen. Building walking gaits for irregular terrain from basis controllers. In *Int. Conf. on Robotics and Automation*, pages 481–486, April 1997.

[113] Macsyma Inc. Macsyma. `http://www.macsyma.com`.

[114] Duane W. Marhefka and David E. Orin. Quadratic optimization of force distribution in walking machines. In *Int. Conf. on Robotics and Automation*, pages 477–483, 1998.

[115] T. McGeer. Passive dynamic walking. *Int. J. of Robotics Research*, 9(2):62–82, 1990.

[116] R. B. McGhee. Some finite state aspects of legged locomotion. *J. Math. Biosciences*, 2:67–84, 1968.

[117] R. B. McGhee and A. A. Frank. On the stability properties of quadruped creeping gaits. *J. Math. Biosciences*, 3:331–351, 1968.

[118] R. B. McGhee and G. I. Ishwandhi. Adaptive locomotion of a multilegged robot over rough terrain. *IEEE Trans. Systems, Man, and Cybernetics*, 9(4):176–182, 1979.

[119] Scott McMillan and David E. Orin. Forward dynamics of multilegged vehicles using the composite rigid body method. In *Int. Conf. on Robotics and Automation*, pages 464–470, 1998.

[120] The mechanical design of a 3 degrees of freedom compliant robot leg. Karim benjelloun. Master's thesis, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1997. TRITA-MMK 1997 MME646, ISSN 1400-1179.

[121] Mechanical Dynamics Inc. ADAMS. `http://www.adams.com`.

[122] Merriam-Webster's Collegiate Dictionary. Encyclopedia Britannica Online, `http://www.eb.com`.

[123] Domini A. Messuri and Charles Klein. Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion. *IEEE Robotics and Automation Magazine*, RA-1(3), September 1985.

[124] N. K. M'Sirdi, N. Manamani, and N. Nadjar-Gauthier. Methodology based on CLC for control of fast legged robots. In *Int. Conf. on Intelligent Robots and Systems*, pages 71–76, 1998.

[125] Nacer K. M'Sirdi, Marina Guihard, and Jean-Guy Fontaine. Identification and control of pneumatic driven robot. In *Int. Conf. on Systems, Man and Cybernetics*, volume 3, pages 722–727, Le Touquet, France, October 1993.

[126] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[127] E. Muybridge. *Animals in Motion*. New Dover Edition, Dover Publications, Inc., New York, 1957. First published in 1899.

[128] P. Nagy, D. Manko, S. Desa, and W. Whittaker. Simulation of postural control for a walking robot. In *IEEE Int. Conf. on System Engineering*, pages 324–329, aug 1991.

[129] P. V. Nagy, S. Desa, and W. L. Whittaker. Energy-based stability measures for reliable locomotion of statically stable walkers: Theory and application. *Int. J. of Robotics Research*, 13(3):272–287, June 1994.

[130] H. M. Lankarani; P.E. Nikravesh. A contact force model with hysteresis damping for impact analysis of multibody systems. *Trans. of the ASME, J. of Mechanical Design*, 112:369–76, 1990.

[131] N. Nishikawa, T. Murakami, and K. Ohnishi. An approach to stable motion control of biped robot with unknown load by torque estimator. In *Int. Workshop on Advanced Motion Control Proc.*, pages 82–87, 1998.

[132] T. Oka, M. Inaba, and H. Inoue. Describing a modular motion system based on a real time process network model. In *Int. Conf. on Intelligent Robots and Systems*, pages 821–827, 1997.

[133] OPAL-RT Technologies Inc. OPAL-RT. `http://www.opal-rt.ca`.

[134] Koichu Osuka. Control theoretic approach to motion control of legged robot. In *Int. Workshop on Advanced Motion Control Proc.*, pages 88–92, 1998.

[135] D. Pack and H. Kang. An omnidirectional gait control using a graph search method for a quadruped walking robot. In *Int. Conf. on Robotics and Automation*, volume 1, pages 988–993, Nagoya, Japan, May 1995.

[136] Prabir K. Pal and K. Jayarajan. Generation of free gait —- a graph search approach. *IEEE Robotics and Automation Magazine*, 7(3):299–305, 1991.

[137] Prabir K. Pal, Vivek Mahadev, and K. Jayarajan. Gait generation for a six-legged walking machine through graph search. In *Int. Conf. on Robotics and Automation*, volume 2, pages 1332–1337, 1994.

[138] Junmin Pan and Junshi Cheng. Gait synthesis for quadruped robot walking up and down slope. In *Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 532–536, July 1993.

[139] David W. Payton and Thomas E. Bihari. Intelligent real-time control of robotic vehicles. *Communications of the ACM*, 34(8):49–63, 1991.

[140] Lennart Pettersson. Control system architectures for autonomous agents. Technical Report TRITA-MMK 1997:22, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, April 1997. ISSN 1400-1179, ISRN KTH/MMK-97/22-SE.

[141] F. Pfeiffer and H. Cruse. Bionik des Laufens — technische Umsetzung biologischens Wissens. *Konstruktion*, pages 261–266, 1994. In German.

[142] F. Pfeiffer, H.-J. Weidemann, and P. Danowski. Dynamics of the walking stick insect. *IEEE Control Systems Magazine*, 112:9–13, feb 1991.

[143] R. Prajoux and L. de SF. Martins. A walk supervisor architecture for autonomous four-legged robots embedding real-time decision-making. In *Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 200–207, 1996.

[144] G. A. Pratt, M. M. Williamson, P. Dilworth, and J. Pratt. Stiffness isn't everything. In *Preprints of the fourth International Symposium on Experimental Robotics(ISER'95)*, Stanford, California, USA, July 1995.

[145] Jerry Pratt and Gill Pratt. Intuitive control of a planar bipedal walking robot. In *Int. Conf. on Robotics and Automation*, 1998.

[146] Dennis R. Pugh, Eric A. Ribble, Vincent J. Vohnout, Thomas E. Bihari, Thomas M. Walliser, Mark R. Patterson, and Kenneth J. Waldron. Technical description of the adaptive suspension vehicle. *Int. J. of Robotics Research*, 9(2):24–42, 1990.

[147] Quanser Consulting Inc. WinCon. `http://www.quanser.de`.

[148] Roger D. Quinn and Kenneth S. Espenschied. Control of a hexapod robot using a biologically inspired neural network. In Randall D. Beer, Roy E. Ritzmann, and Thomas McKenna, editors, *Biological Neural Networks in Invertebrate Neuroethology and Robotics*, Neural Networks: Foundations to Applications, chapter XVI, pages 365–381. Academic Press Inc., 1993.

[149] Marc H. Raibert. *Legged robots that balance*. The MIT Press, 1986.

[150] Peter J. G. Ramadge and W. Murray Wonham. The control of discrete event systems. *Proc. IEEE*, 77(1):81–98, jan 1989.

[151] RealTech AG. RealLink/32. `http://www.realtech.ch`. now xPC Target from [179].

[152] Henrik Rehbinder. *State Estimation and Limited Communication Control*. PhD thesis, The Royal Inst. of Technology, 100 44 Stockholm, Sweden, 2001. TRITA-MAT-01-OS-09, ISSN 1401-229.

[153] Henrik Rehbinder and Xiaoming Hu. Nonlinear pitch and roll estimation for walking robots. In *Int. Conf. on Robotics and Automation*, volume 3, pages 2617–2622, 2000.

[154] Henrik Rehbinder and Christian Ridderström. Attitude estimation for walking robots. In *Int. Conf. on Climbing and Walking Robots*, September 1999.

[155] C. Ridderström, J. Ingvast, F. Hardarson, M. Gudmundsson, M. Hellgren, J. Wikander, T. Wadden, and H. Rehbinder. The basic design of the quadruped robot Warp1. In *Int. Conf. on Climbing and Walking Robots*, Madrid, Spain, October 2000.

[156] Christian Ridderström. Legged locomotion control — a literature survey. Technical Report TRITA-MMK 1999:27, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, November 1999. ISSN 1400-1179.

[157] Christian Ridderström. Stability of statically balanced, radially symmetric stances for legged robots on compliant surfaces. In *Int. Conf. on Climbing and Walking Robots*, Paris, France, September 2002.

[158] Christian Ridderström. Stability of statically balanced stances for legged robots with compliance. In *Int. Conf. on Robotics and Automation*, Washington DC, USA, 2002.

[159] Christian Ridderström and Johan Ingvast. Combining control design tools — from modeling to implementation. In *Int. Conf. on Robotics and Automation*, pages 1327–1333, 2001.

[160] Christian Ridderström and Johan Ingvast. Quadruped posture control based on simple force distribution — a notion and a trial. In *Int. Conf. on Intelligent Robots and Systems*, pages 2326–2331, 2001.

[161] Robert P. Ringrose. *Self-Stabilizing Running*. PhD thesis, Massachusetts Inst. of Technology, February 1997.

[162] L. Roussel, C. Canudas de Wit, and A. Goswami. Generation of energy optimal complete gait cycles for biped robots. In *Int. Conf. on Robotics and Automation*, pages 2036–2041, 1998.

[163] U. Saranli, W. J. Schwind, and D. E. Koditschek. Toward the control of a multi-jointed, monoped runner. In *Int. Conf. on Robotics and Automation*, volume 3, pages 2676–2682, 1998.

[164] Josef Schmitz, Thomas Kindermann, Michael Schumm, and Holk Cruse. Simplifying the control of a walking hexapod by exploiting physical properties. In *European Mechatronics Colloquium, Biology and Technology of Walking*, pages 296–303, 1998.

[165] W. J. Schwind and D. E. Koditschek. Characterization of monoped equilibrium gaits. In *Int. Conf. on Robotics and Automation*, volume 3, pages 1986–1992, 1997.

[166] Jana Košecká and Ruzena Bajcsy. Discrete event systems for autonomous mobile agents. *Robotics and Autonomous Systems*, 12:187–198, 1994.

[167] Amir Shapiro, Elon Rimon, and Joel W. Burdick. Passive force closure and its computation in compliant-rigid graps. In *Int. Conf. on Intelligent Robots and Systems*, pages 1769–1775, 2001.

[168] C. L. Shih, Y. Z. Li, S. Chung, T. T. Lee, and W. A. Gruver. Trajectory synthesis and physical admissibility for a biped robot during the single-support phase. In *Int. Conf. on Robotics and Automation*, pages 1646–1652, 1990.

[169] Ching-Long Shih and William A. Gruver. Control of a biped robot in the double-support phase. *IEEE Trans. Systems, Man, and Cybernetics*, 22(4):729–735, July 1992.

[170] Ching-Long Shih, William A. Gruver, and Yun Zhu. Fuzzy logic force control for a biped robot. In *IEEE Int. Symposium on Intelligent Control*, pages 269–274, Arlington, Virginia, USA, August 1991.

[171] Ching-Long Shih, Yun Zhu, and William A. Gruver. Optimization of the biped robot trajecotry. In *Int. Conf. on Systems, Man and Cybernetics*, pages 899–903, 1991.

[172] M. Sobh, J. C. Owen, K. P. Valvanis, and D. Gracani. A subject-indexed bibliography of discrete event dynamic systems. *IEEE Robotics and Automation Magazine*, 1(2):14–20, 1994.

[173] Shin-Min Song and Yaw-Dong Chen. A free gait algorithm for quadrupedal walking machines. *J. of Terramechanics*, 28(1):33–48, 1991.

[174] Shin-Min Song and Kenneth J. Waldron. *Machines that Walk*. MIT Press, Cambridge, MA, 1989.

[175] K. Sorao, T. Murakami, and K. Ohnishi. A unified approach to ZMP and gravity center control in biped dynamic. In *Int. Workshop on Advanced Motion Control Proc.*, page 112, 1997.

[176] Stiftelsen för Strategisk Forskning. (Swedish Foundation for Strategic Research). `http://www.stratresearch.se`. Box 70483, S-107 26 Stockholm, Sweden.

[177] Steve Stitt and Yuan F. Zhen. Distal learning applied to biped robots. In *Int. Conf. on Robotics and Automation*, volume 1, pages 137–142, 1994.

[178] A. Takanishi, H. Lim, M. Tsuda, and I. Kato. Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface. In *Int. Conf. on Intelligent Robots and Systems*, pages 323–330, 1990.

[179] The MathWorks Inc. MATLAB, Simulink, xPC Target etc. `http://www.mathworks.com`.

[180] D. J. Todd. *Walking machines – an introduction to legged robots*. Kogan Page Ltd, London, 1986.

[181] R. Tomović. A general theoretical model of creeping displacement. *Cybernetica*, 4, 1961.

[182] Hideyuki Tsukagoshi, Shigeo Hirose, and Kan Yoneda. Maneuvering operations of a quadruped walking robot on a slope. *Advanced Robotics*, 11:359–375, 1997.

[183] C. Tzafestas, M. Guihard, and N. K. M'Sirdi. Two-stage adaptive impedance control applied to a legged robot. In 3, editor, *Int. Conf. on Intelligent Robots and Systems*, pages 173–178, 1995.

[184] Costas S. Tzafestas, Nacer K. M'Sirdi, and N. Manamani. Adaptive impedance control applied to a pneumatic legged robot. *J. of Intelligent and Robotic Systems*, 20:105–129, 1997.

[185] S. Tzafestas, M. Raibert, and C. Tzafestas. Robust sliding-mode control applied to a 5-link biped robot. *J. of Intelligent and Robotic Systems*, 15(1):67–133, 1996.

[186] Logistical vehicle off-road mobility. Fort Eustis, Va.: U.S. Army Transportation Combat Developments Agency, feb 1967. (Project TCCO 62-5).

[187] A. F. Vakakis, J. W. Burdick, and T. K. Caughey. An interesting strange attractor in the dynamics of a hopping robot. *Int. J. of Robotics Research*, 10(6):606–618, December 1991.

[188] O. Vanel and P. Gorce. Adaptive criteria for biped dynamic stability under external perturbations. In *Int. Conf. on Intelligent Robots and Systems*, pages 192–199, 1996.

[189] S. T. Venkataraman. A simple legged locomotion gait model. *Robotics and Autonomous Systems*, 22:77–85, 1997.

[190] C. Villard, P. Gorce, and J. G. Fontaine. Study of the dynamic behavior of ralphy. In *Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 1765–1770, Yokohama, Japan, July 1993.

[191] Claude Villard, Phillipe Gorce, Jean-Guy Fontaine, and Jacques Rabit. RALPHY: A dynamic study of a quadruped robot. In *Int. Conf. on Systems, Man and Cybernetics*, pages 106–111, 1993.

[192] G. S. Virk and D. R. Harvey. Dynamically stable legged robots. In *CLAWAR '98 First International Symposium*, pages 335–342, 1998.

[193] M. Vukobratovic, A. A. Frank, and D. Juricic. On the stability of biped locomotion. *IEEE Trans. Biomedical Eng.*, BME-17(1):25–36, January 1970.

[194] M. Vukobratovic and J. Stepanenko. On the stability of anthropomorphic systems. *J. Math. Biosciences*, 15:1–37, 1972.

[195] M. Vukobratovic and J. Stepanenko. Mathematical models of general anthropomorphic systems. *J. Math. Biosciences*, 17:191–242, 1973.

[196] Kenneth J. Waldron, Vincent J. Vohnout, Arrie Pery, and Robert B. McGhee. Configuration of the adaptive suspension vehcile. *Int. J. of Robotics Research*, 3(2):37–48, 1984.

[197] Lars Wallentin, Kennet Jansson, and Sören Andersson. SLEIPNER3 — A four legged robot platform. In Kjell Andersson and Jan-Gunnar Persson, editors, *NordDesign '98*, pages 289–297, Stockholm, Sweden, August 1998.

[198] Waterloo Maple Inc. Maple. `http://www.maplesoft.com`.

[199] H.-J. Weidemann, F. Pfeiffer, and J. Eltze. The six-legged TUM walking robot. In *Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 1026–1033, 1994.

[200] David Wettergreen, Chuck Thorpe, and Red Whittaker. Exploring Mount Erebus by walking robot. *Robotics and Autonomous Systems*, 11(3-4):171–185, December 1993.

[201] D. M. Wilson. Insect walking. *Annu. Rev. Entomol*, 11:103–121, 1966.

[202] Wolfram Researh Inc. Mathematica. `http://www.wolfram.com`.

[203] Ho Cheung Wong and D. E. Orin. Control of a quadruped standing jump over irregular terrain obstacles. *Autonomous Robots*, 1(2):111–129, 1995.

[204] Jung-Min Yang and Jong-Hwan Kim. A strategy of optimal fault tolerant gait for the hexapod robot in crab walking. In *Int. Conf. on Robotics and Automation*, pages 1695–1700, 1998.

[205] K. Yoneda and S. Hirose. Dynamic and static fusion gait of a quadruped walking vehicle on a winding path. In *Int. Conf. on Robotics and Automation*, pages 143–148, 1992.

[206] K. Yoneda and S. Hirose. Tumble stability criterion of integrated locomotion and manipulation. In *Int. Conf. on Intelligent Robots and Systems*, pages 870–876, 1996.

[207] K. Yoneda, H. Iiyama, and S. Hirose. Intermittent trot gait of a quadruped walking machine. In *Int. Conf. on Robotics and Automation*, volume 4, pages 3002–3007, 1996.

[208] Kan Yoneda and Shigeo Hirose. Dynamic and static fusion gait of qudruped walking vehicle on a winding path. *Advanced Robotics*, 9(2):125–136, 1995.

[209] Kan Yoneda, Hiroyuki Iiyama, and Shigeo Hirose. Sky-Hook suspension control of quadruped walking vehicle. In *Int. Conf. on Robotics and Automation*, pages 999–1004, 1994.

[210] Kan Yoneda, Hiroyuki Ilyama, and Shigeo Hirose. Trot gait of quadruped walking machine on a rough terrain. In *Proc. of ROBOMEC'94*, pages 389–392, 1994. in Japanese).

[211] Milos Zefran and Joel W. Burdick. Stabilization of systems with changing dynamics by means of switching. In *Int. Conf. on Robotics and Automation*, pages 1090–1095, 1998.

[212] Garth Zeglin and Ben Brown. Control of a bow leg hopping robot. In *Int. Conf. on Robotics and Automation*, pages 793–798, 1998.

[213] Teresa Zielinska. Coupled oscillators utilized as gait rhythm generators of a two-legged walking machine. *Biological Cybernetics*, 6(2):263–273, March 1996.