University of Twente

EEMCS / Electrical Engineering
Control Engineering

ZMP based control in 3D passive dynamic walking

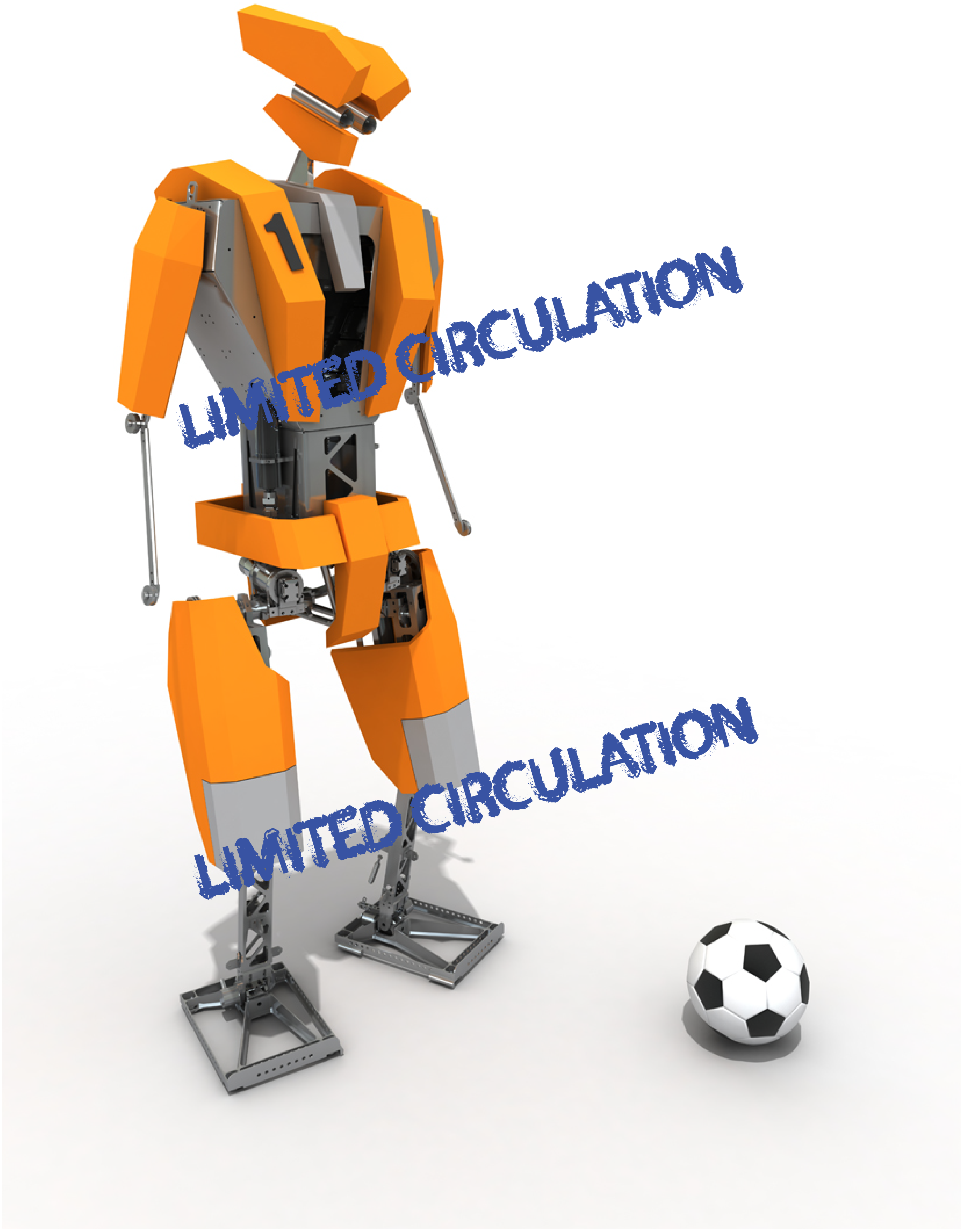
Peter Daemen

MSc report

Supervisors:
prof.dr.ir. S. Stramigioli
ir. G. van Oort
prof.dr.ir. J. van Amerongen

February 2008

Report nr. 001CE2008
Control Engineering
EE-Math-CS
University of Twente
P.O.Box 217
7500 AE Enschede
The Netherlands



Abstract

Within the context of the 3TU federation the 14 degrees of freedom (DOF) antropomorphic robot 'Tulip' is being developed with the goal to compete in the Robosoccer tournament of 2008 held in Suzhoo, China. Tulip will be a dynamic walker able to kick, goal-keep and walk. In order to gain a deeper insight in the dynamics of the design and to design and test motion patterns and controllers, a model has been developped and presented in this report, which will serve as a starting point for further work on the robot.

Tulip has 12 DOFs that are actively actuated and 2 DOFs that are passively actuated by torsional springs. Of this structure a 3D-dynamics model is built up using 20-sim's 3D Mechanics Editor. The main points of interest include the modelling of end-stops, ground contacts and the overall control structure, all incorporated into a re-usable, clear and efficient simulation-model.
As a test case for this model and as a first trial for the design specifications, a routine is developed for getting to an upright standing position from a prone position. Several motions were considered and a robust and stable algorithm is presented in this report.

Stability in biped robots in general has been well researched and for many robots. The zero moment point (ZMP) is a frequently used, and often discussed, criterion ensuring dynamic stability of the robot. Periodic gait stability for such walker usually results in small regions of stability. In this report a control strategy is proposed that reduces the necessity for trial and error establishment of variables, but moreover will produce gaits less sensitive to parameter variations.

Preface

This report contains the results of my final project in my masters studies in Electrical Engineering. Coming home from an internship working on a humanoid robot in Spain, I sat down with Edwin Dertien to discuss those results and possible graduation projects. As I was complaining a bit about the fact that I had not been dealing with locomotion or any other fundamental problem in Spain, I was extremely lucky to hear that in context of the 3TU federation a humanoid dynamic walker was being developped and that a simulation model and walking behaviour needed to be designed. Hence I ended up with this great project.

Throughout the months of hardship and suffering, which in fact were months of fun and challenging research, many people have popped up to help and advise, whom I would like to thank. Clearly I will not be able to mention them all here as it would get too long a list. So even if your name is missing: you are not forgotten.
Obviously this whole project would be impossible without the new great releases from Control Lab Products and I would like to thank them all, and Frank especially for the time they took to help out. Then of course there were my room-mates at CE8156, always sharing that essential coffeebreak, which was another fundamental building block of this thesis [1].

Some 2 years ago, prof. dr. ir. Stefano Stramigioli was the one who sparked my interest in robotics, made it possible for me to go on internship in Spain and, with his vision and ideas, led the project to where it is now. Another great many thanks go out to ir. Gijs van Oort, who always had a moment free when I needed him. Seldomly have I met any person who is able to say 'yes, no problem' that many times and still smile. He always offered constructive feedback and support which literally dragged me through the rough patches encountered. And last but not least I thank Anna, for enduring me bei glued to my laptop 24/7, working on this weird, perhaps juvenile boy-ish, topic that the normal world does not concern itself with.
All in all it was a great project, which took place in a great research deparment, for which I am very grateful.

Peter Daemen, January 2007

[1] http://rsna2005.rsna.org/rsna2005/V2005/conference/event_display.cfm?em_id=4418422

Contents

Chapter 1

Introduction

Ever since the first operational robot was employed at the General Motors plant in Ewing Township [12], robotics has evolved into a large and established field of research and commerce. In manufacturing, robots are paramount and we are slowly moving towards applications such as household robotics [8] and other task-settings which require both moving around in the human sphere as well as human interaction.

This move, albeit slow, shifts the focus from stationary robots, via mobile robotic platforms, to independent robots that can move around alongside of humans in their working and living environment. Especially this requirement and the incentive from the medical industry towards active prosthesis design, push research into the direction of biped locomotion in anthropomorphic robots.

1.1 Bipedal locomotion

Many institutes, both commercial as well as academic, are contributing to a worldwide progress of the field of humanoid robotics and biped locomotion. Examples of functioning humanoid robots are Honda's Asimo [9] and HRP-2. Both robots are independent humanoid robotic platforms, however their human likeness is lacking one important aspect: anthropomorphic efficient locomotion.

The principal reason for this is the concept of actuation. High gear-ratios, resulting in a non backdriveable system combined with stiff control and pre-computed gaits for every conceivable motion have led to very acceptable and interesting looking displays. However adorable the result of these techniques may be, it is very power-consuming as it disregards most of the oscillatory behaviour of for example a swinging leg. Development has branched into two main directions, which can be characterized by their stability criteria and control. There are many examples of robots with dynamically stable walking gaits, such as Asimo [9], RH-1 [1], HRP-2 [6] and many others.
This type of gait is developed in such fashion that at every moment of time, the robot

finds itself in a dynamic equilibrium: it does not fall, no matter what it is doing[1]. The mainstream criterion to ensure such stability is the Zero Moment Point (ZMP) criterion in combination with stiff set-point control on the principal joints of the robot. The ZMP criterion is used both in on-line and off-line applications; firstly to generate viable combinations of motion patterns and secondly online to deal with interaction forces like wind or a changing ground surface, such as steps up and down.

In contrast to such walkers, McGeer pioneered the study and development of completely passive walking in 1990 [11]. Starting off from the 'simplest walker', much research has been done on energy efficient walking. From planar walking frames on inclining slopes, the 'passive dynamic' principle of walking is applied to more complex 3D walkers, such as Denise[16] and others [3] and in the future the 3TU robot Tulip as well.

1.2 Context

The research at the Control Engineering department has been focused on determining the important dynamic characteristics of 2D dynamic walking frames. Dribbel has been developed [5] to study the important dynamic characteristics of its structure and the development of robust control strategies for such walking frames. Research on Dribbel has covered stability analysis of walking gaits [14], the influence of foot-shapes [15], the influence of ankle actuation [7] and other topics.

Especially [14] has started the movement from planar walkers to 3D walkers. This is a step which is being continued by participating in the 3-TU[2] development a 3D humanoid robot to compete in the Robot-soccer league of 2008 in Suzhou, China. Based upon previously developed walking robots at the Technical University of Delft and the work of Wisse and others[16], robot Tulip has been designed.

1.3 Goal

This thesis establishes a model and simulation structure of Tulip, in 20-sim. The purpose of the model will be the development of and experimentation with different motion patterns required for the RoboSoccer League of 2008 in Suzhou, therefore a strong focus will be on detail and commenting of the simulator code. Secondly a 3D walking control algorithm is sought based upon the transposition of the well known and widely discussed Zero Moment Point (ZMP) from the 'dynamically stable' to the 'passive dynamic' world.

[1]it may be noted that Asimo and Qrio [4] reportedly can run, which is not 'dynamically stable'
[2]Federation of Dutch Technical Universities

1.4 Report outline

This report sets out to provide a sound basis of understanding for continuation of work on this project, therefore at times the report follows a more qualitative than quantitative approach. Chapter chapter 2 will focus on the development and implementation of a 20-sim model of Tulip. Then chapter chapter 3 demonstrates an application of the developed model by proposing a recovery algorithm from a prone position. In chapter 4 a new gait-control strategy is proposed, incorporating the ZMP concept into a dynamic walker after which conclusions and recommendations are presented. In the appendices, detailed design specifications are included, of the robot, the motors, gearboxes, ground-contact pucks and of course the essential code of the simulator and the setpoint generation DLL.

Chapter 2

Model of 3D walker, Tulip

This chapter will elaborate on the structure and implementation of the model developed to simulate gaits and other motion patterns for the robot Tulip. Firstly the structure of the model is outlined, after which the 3D-dynamic model and its composition is discussed. Then section section 2.1.2 outlines the ground-contact model and finally the control and actuation structure is presented.

To develop and simulate a model for Tulip, the simulation package 20-sim 4.0.1[2] (20-sim) is used. The developed model can be divided into three main parts, as shown in figure 2.1. Overall control, the block 'Setpoints', generating the set-points and controller parameters for the control loop; control and actuation, containing a 14 dimensional control loop for all joints; and the mechanical model, in which the dynamics and all present motion constraints are modelled. The overall control will be discussed in chapter chapter 3 and again in chapter chapter 4, as its use is different in the two implementations that are presented in this report. The following sections will discuss the remaining submodels.

2.1 Mechanical model

2.1.1 'DynamicModel'

This block in figure figure 2.1 contains the automatically generated code from the 3D Mechanics Editor (3D-ME) included in the 20-sim package. In figure figure 2.2 two views of the 14 degrees of freedom (DOF) model of Tulip are shown, in which the center of mass (COM) of all bodies has been marked and world coordinate frame Ψ_0 is defined. Lastly he cylinders illustrate rotational joints along their center axis.
The models consists of 11 bodies with specified inertias, masses and relative positions to the connecting joints. These dimensions of the bodies have been extrapolated from the mechanical design made at the TU-Delft, as given in diagrams 1, 3 and 4 in appendix A. Additional to the design, 4 extra bodies, with negligible mass and inertia, have been added to accommodate more than one DOF between ankle and lower leg and between hip and upper leg. It should be noted that the visual representation in figure 2.2 a) of the robot is only geometrically correct and is chosen in a user-friendly way, but the shape of the bodies has no relation to the inertial properties or mass of the body itself. In figure

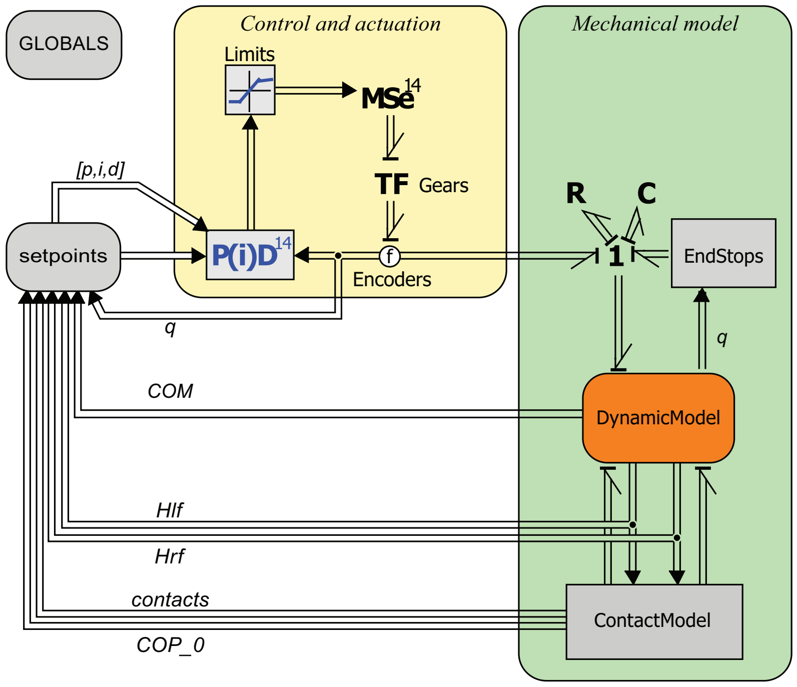
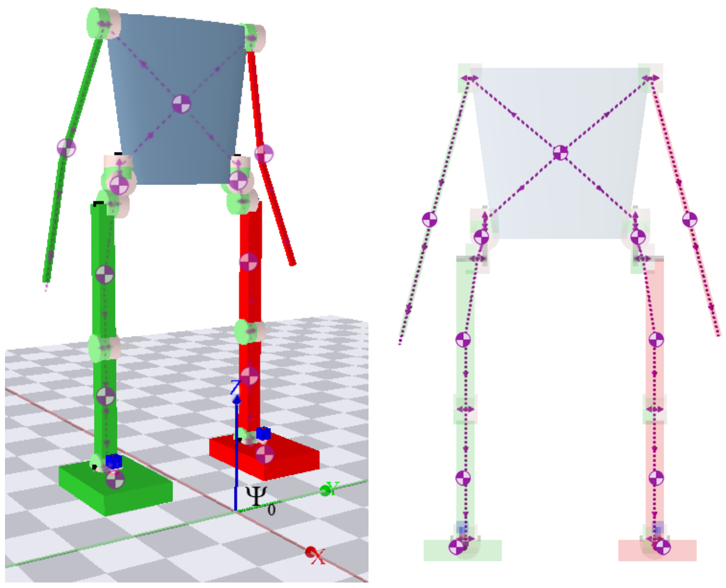


Figure 2.1: Overview of the used simulation model

2.2 b) the locations of the centers of mass (COMs) are shown, where it must be noted that all $\hat{x}_0$ components of H_i^0 are zero.
In addition to the dynamics of the robot, the 3D-ME allows for the insertion of "additional code". This has been used to enumerate the joints in a logical way, as denoted in appendix A. As the generated 20-sim code relies on the use of vectors to describe for example the 14-dimensional actuator-space, it is important to consequently order the joints in this vector to avoid confusion. Rather than being dependent on the internal workings of the code-generator for the numbering of the joints, appendix A presents the joints, their names and number which are defined in the 3D-Mechanics Editor.
It is important to note that each body has it's own coordinate system, located in the COM of the body and there is a world frame Ψ_0 defined by the 3DME. In the model there is one 6 DOF free moving joint, connecting the model to a body which is fixed to the world. This is done to be able to more easily define and manipulate initial positions of the robot with respect to the fixed world, throughout several versions of the editor. The location of this extra body is independent of Ψ_0 and the body is not important for the simulations.

The C and R elements that can be seen in figure 2.1 represent torsional spring acting and the R represent a damper on all joints. They have been implemented 14-dimensionally, with $K, \phi_0, R \in \mathbb{R}14$. Via matrix manipulations, fourteen springs are implemented, and fourteen dampers to model linear friction. This is done because one of the design choices has been to have the ankle-y-rotations to be passively actuated. For flexibility the springs have been implemented on all joints, but are generally only active on the ankle-y rotation. This actuation is called passive, because the present spring is not supplying energy to the



(a) x,y,z (b) y,z

Figure 2.2: The 3D mechanics model of Tulip, joints in zero-positions

system, only storing it. The choice for such a passive DOF has been made before the start of this project, based upon the experiences of the partners in the 3TU team.

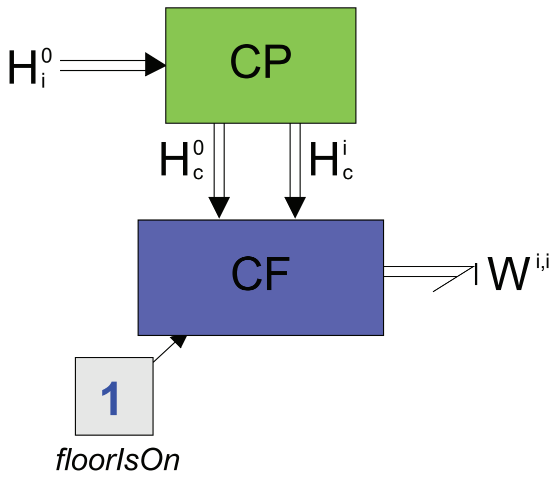
2.1.2 'Contactmodel'

After modelling the dynamics of the robot, the interaction of the model with the environment needs to be modelled, as the 3D-ME does not offer integrated floors and reaction forces. The component 'ContactModel' determines whether there is contact with the ground and the resultant force of this contact. In figure 2.3 one 'cell' out of the contact model has been shown. For each contact point to be modelled there exists such a combination of blocks and additionally, there is one block connected to both of the contact-model submodels for the feet of the robot, in order to pass on information about which foot is interacting with the floor and the center of pressure (COP).

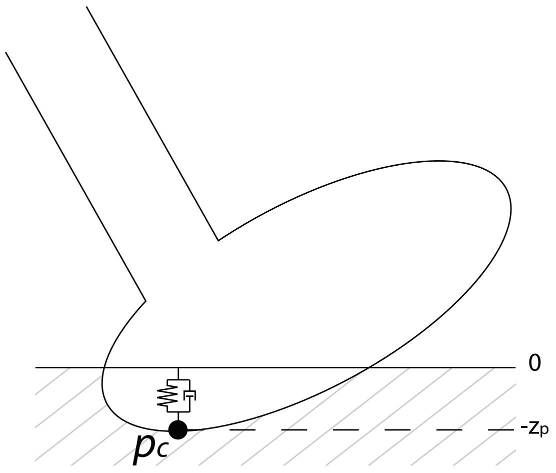
$W^{i,i}$ denotes a wrench [13] acting on body i, seen from its own coordinate frame, Ψ_i. Wrenches are generally defined as

$$W^{i,j} = [M_x \ M_y \ M_z \ F_x \ F_y \ F_z] \in \mathbb{R}^6 \tag{2.1}$$

where M_a and F_a are the moments around and the forces along axis a of coordinate frame Ψ_i, acting on body j. An important observation is that in 20-sim bond graphs, wrenches take the place of efforts, where in our 3D model the actuator which effectuates the interaction forces is defined such that $P.e = W^{i,i^T}$, a column vector.



(a) Schematic



(b) Contact model

Figure 2.3: The contact model

In figure 2.3(b) body i is defined, with frame Ψ_i at its COM and transformation matrix H_i^0 [13]. This means that if

$$p^a = [x_a, y_a, z_a, 1]^T$$

and

$$H_i^j = \left[\begin{array}{cc} R_i^j & p_i^j \\ 0^3 & 1 \end{array} \right],$$

then

$$p^j = H_i^j p^i, \tag{2.2}$$

where p^a denotes the point p expressed in coordinate frame Ψ_a, R_i^j denotes the 3D rotation from Ψ_i to Ψ_j and p_i^j is the translation that brings the origin of Ψ_i to Ψ_j.

Submodel 'CP', contact point, determines the lowest point of body i, p^c. Then frame Ψ_c is defined with p^c as the origin and the orientation along world frame Ψ_0. Knowing this transformation matrices H_c^0 and H_i^c, are calculated, being the coordinate change from Ψ_c to Ψ_0 and to Ψ_i, respectively.

Assuming a convex geometry of body i, submodel 'CF' checks H_c^0 to see if p^c is below the ground level. If it is not, then from the convex geometry the body it follows that there is no point below the floor, which means no contact and no reaction forces.
If p^c approaches the ground level[1] submodel CF focusses on the zero-crossing of $p^c[3]$ and upon that event, the contact model explained in the following paragraph is activated.

[1]For simulation purposes, the model forces the simulator to determine the zero crossing of the z-coordinate of p^c to prevent large 'dips' below the floor and large discontinuities in the reaction force

Compliant contacts

Submodel 'CF', contact force, calculates F_N, the normal force compensating gravity and the coupled frictional forces. The force is calculated according to a compliant contact model, which is justified by two reasons.
Firstly, [15] has found that for efficient passive dynamic gaits, compliant contact with the floor is beneficial. Therefore the feet of the robot have been designed such that they rest upon pucks, which have a K-value, as can be found in appendix C. The second argument is that the alternative solution, a rigid contact model, poses large problems in terms of simulation. A rigid contact would mean that, for example, during the two phases of a gait, swing and double-support, the dynamic equations would be fundamentally different. As the contact point becomes a fixed point on impact, during the gait, the dynamic equations alternate between an open kinematic chain for the swing phase and a redundant closed kinematic chain during the double support phase.
As an added advantage, compliant contacts have proven to be much simpler to simulate and less demanding on the simulator due to the avoidance of many extra discontinuous states. It must be noted however that the introduction of very stiff models alongside of a very loosely controlled passive dynamic walker will give rise to rather high computation times.

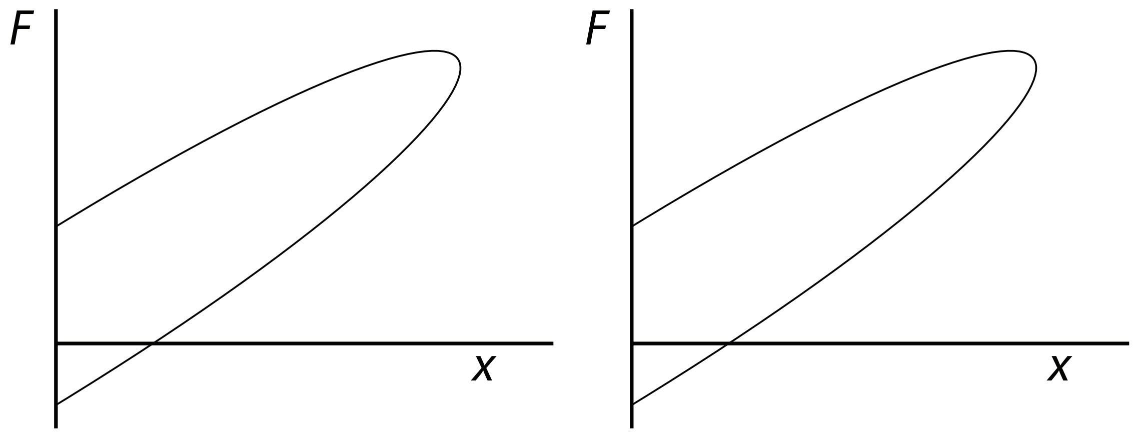
A compliant contact can be modelled as a spring and damper between p^c and the ground level, acting in the z-direction of Ψ_0. For efficient simulation, the model should be critically damped, making its parameters, spring constant K and damping factor D depend on the admissible steady-state ϵ_{ss}, mass M resting on the point, gravitational constant g by equation equation 2.3.

$$\begin{aligned} K &\geq Mg/\epsilon_{ss} \\ D &= 2KM^{1/2} \end{aligned} \tag{2.3}$$

Of course the 2 parameters are subject to fine-tuning towards a trade off between accurate simulation and fast simulation results. One of the problems one encounters is that it is not always clearly known which portion of the total mass of the robot will rest on the contact point, therefore how large is M in equation equation 2.3. However, taking the total mass as a worst case scenario will result in reasonable performance in simulation.

A problem arising from equation equation 2.3 is that when the body is moving to break contact a, the damper will counter-act such motions, resulting in unnatural 'stick'. In figure figure 2.4 a) the $F(x)$ curve of a body hitting the ground and then moving away agian is show. When $x = 0$ occurs for the second time, at point A, we can clearly see that the force exerted by the linear contact model is negative, potentially keeping the body on the ground. In order to overcome this we introduce a non-linearity into the damper, such that when P_c reaches the ground level, the forces are zero. This is achieved by

$$F_N = \begin{cases} -K(z - z_0) + D\dot{z}(z - z_0) & \text{if } z \leq H_f, \\ 0 & \text{if } z > H_f, \end{cases} \tag{2.4}$$



(a) Linear spring-damper model (b) Hunt-Crossley model

Figure 2.4: [F,x] diagram of linear spring/damper contact and a Hunt-Crossley contact

which is better known as the Hunt-Crossley model [10]. Equation equation 2.4 gives the implemented equations for determining F_N, the normal component of the contact force, counteracting gravity, of which the result is shown in figure 2.4 b), where $x = -(z - z_0)$.

Contact points and surfaces

Now for each body we need to choose an appropriate contact surface. In section section 2.1.2 we assumed the body shape to be convex, which is clearly not the case in any of the bodies of our robot. To ease the calculation of point P_c, we therefore choose convex shapes that approximate the body shapes, or relate well to the function the contact point has in maintaining the robot position with respect to the floor.

The arms of the robot are very thin, and the most important characteristic of the arm to be modelled is the presence of a fixed elbow. Due to this, as becomes clear in chapter 3, some advantages arise from the fact that we can push on the floor therefore we can choose the simplest convex contact surface for these four contacts: a point. No computation is required and the two required transformation matrices can be straightforwardly defined. However, as soon as the exact specifications of the protective outer hull are known however, some alterations may be needed. The contact point at the tip of the arms is sure, however, the shape of the upper arm might lead to an ellipsoid contact surface.

Because of the wider cross-section of the lower leg, the contact point on the top of the lower leg is chosen spherical to form a knee-cap so to speak. To determine the lowest point of a spherical contact point, one simply translates, in Ψ_0 but starting from the origin of the sphere, over the distance of the radius of the circle, in the direction of $\hat{z}_0$ as depicted in figure 2.5. Then P_c^0 is simply transformed to the body frame by multiplication with H_0^i.

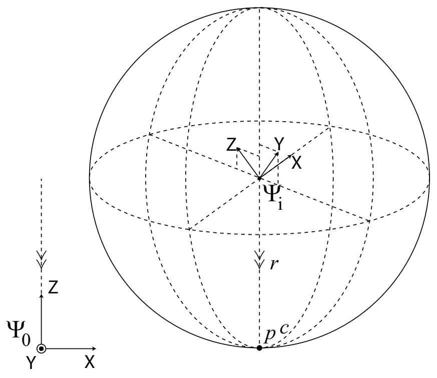


Figure 2.5: Determination of the lowest point of a sphere, implemented as in appendix E.1.2

For the torso we choose a rather flat ellipsoid, to approximate its shape. Determining P_c in this case will be done in two steps. Firstly we transform the total space, by the invertible linear transformation $Q^{sph}_{ell} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ such that the ellipsoid becomes a sphere. Then in this virtual space, we repeat the steps for a spherical contact surface and then apply the inverse transformation Q^{ell}_{sph} to obtain P_c, of which the total implementation is given in appendix E.1.3.

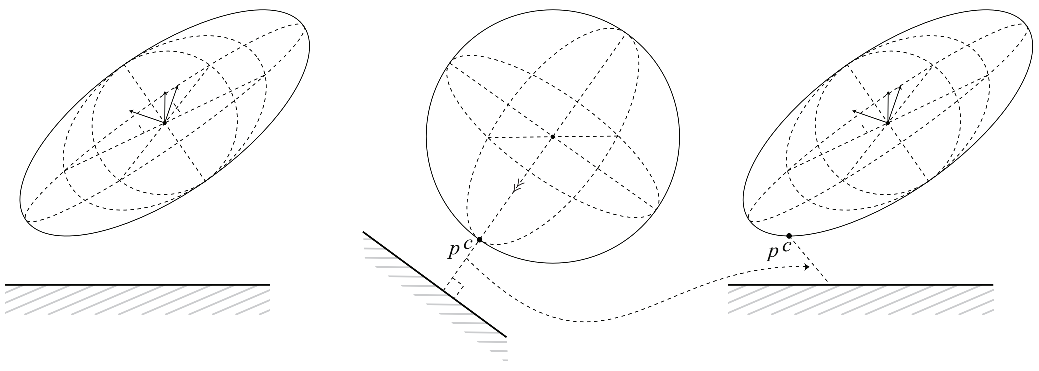


Figure 2.6: Determination of the lowest point of a ellipsoid

So far the feet have been assumed to be flat, which is not a convex shape. The best convex approximation of a flat surface is sphere with a very large radius, r. We then take a small portion of its surface and within this surface we try to find the lowest point with respect to Ψ_0. When the foot rotated that much that P_c calculated for the circle itself, would fall outside of this small portion, P_c for this foot will be located on the edge of this surface.

Depending on the size of the radius, the foot will exhibit some form of rolling behaviour, meaning that point p will move around the foot-sole. As for further application the location of p, as the centre of pressure (COP), it is important we need to verify whether the behaviour of p can be assumed to be reasonable and the choice of this contact surface is correct. Therefore compare our spherical flat foot with one contact to the situation in which there are four contacts at the corners of the foot.

In figure figure 2.7 we observe the two-dimensional case and we can see that a torque on the foot, as a result of a force somewhere on the rest of the body, will result in a larger force on the front contact point. This corresponds to the intuition that the pressure on the front foot increases when your center of mass moves forward. From equation 2.4 it follows that also in the case of two contact points there must be a slight rotation of the foot, as the force in a contact point in steady-state only depends on the z position with respect to the floor.

Knowing that in steady state all moments are zero, we can establish an expression for P_c for both foot models. In figure 2.7 we do this for the 2D case, where x denotes the position where the resultant F_r acts. We assume the rotations to be very small, justifying $x' = x$ For the four-point foot, we draw up the moments with respect to the ankle location and it follows that

$$\begin{aligned} F_1 &= \frac{1}{2}(F_z - \epsilon l * \sin\theta) \\ F_2 &= \frac{1}{2}(F_z + \epsilon l * \sin\theta). \end{aligned} \tag{2.5}$$

Setting all moments to zero for the four-point foot leads us to

$$\begin{aligned} F_z x_{fr} &= \frac{l}{2}(F_z - Kl\sin\theta) + \frac{l}{2}(F_z + Kl\sin\theta) \\ \rightarrow x &= \frac{Kl^2\sin\theta}{4F_z} \quad \tilde{=} \frac{Kl^2\theta}{4F_z}, \end{aligned} \tag{2.6}$$

while for the convex foot it easily follows that the foot will rotate until x is directly underneath the acting point of F_z. This straightforwardly delivers the expression

$$x = \pi r \sin\theta \quad \tilde{=} \pi r\theta \ , \tag{2.7}$$

which is unfortunately not the same as the expression in equation 2.6. In a dynamic situation, $F_z = f(t)$, which means that on impact, especially when using a compliant contactmodel, the transient behaviour of $F_z(t)$ influences the location of the COP, while the convex foot does not account for that.

We must note however, that apart from on impact, the variations in F_z can, while walking, be assumed to be rather small to the constant $m * g$ component. This means that for walking purposes, when the Hunt-Crossley equations are tuned such that the transient of F_z is fast, the model of the convex foot closely approximates the 4-point model Therefore we may conclude that this surface choice is proper and tuning of parameters will enable the model to approximate the real four-point-contac situation.

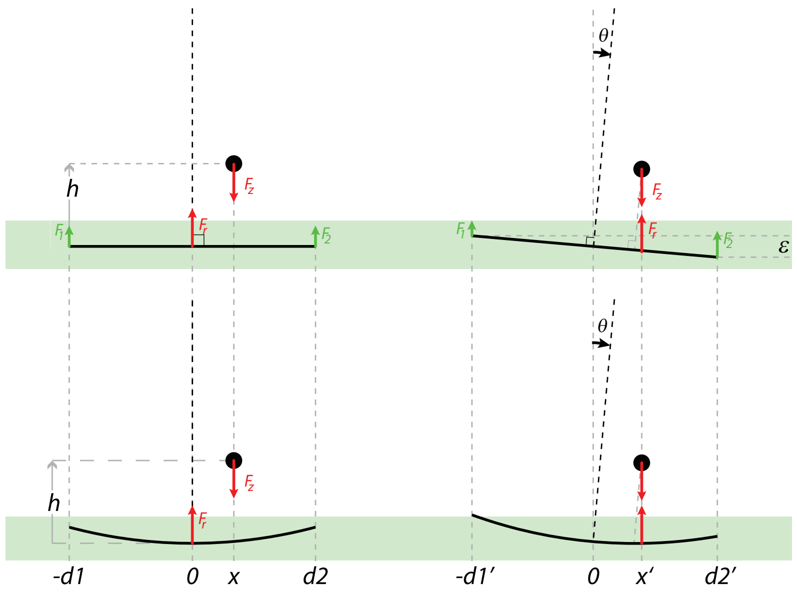


Figure 2.7: Foot rolling with 2 contact points (up) versus convex spherical foot

An important note is that when, for this model, a ZMP controller will be implemented the design must be made such that the location of the ZMP is not approximated by the COP, as this will not incorporate the dynamics of the robot into the ZMP position, only the configuration[2].

The first image of figure figure 3.3 shows a snapshot from a simulation which confirms that the outlined combination of contact points and surfaces keep the model in prone position leveled with respect to the floor. As for the implementation of the ellipsoid and the sphere, one might doubt the usefulness of such distinction. The main reason for it is that to model a total body that behaves as figure 3.3 would need more contact points to achieve e.g. the same behaviour for lying in prone and in supine position.

Contact friction

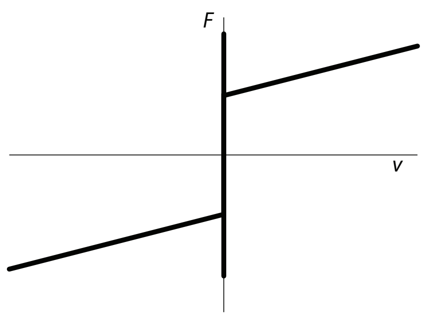
A standard friction model taking into account for static friction, coulomb friction, viscous friction and depends on the relative acceleration, velocity and force present between the two interacting bodies. Having calculated the magnitude of F_N, the other two, frictional, force components of contact wrench $W^{i,i}$ are computed by

$$F_{fr} = -Fn(sign(v_t r)\ (\mu_c + \mu_{st}|tanh(sv_{tr})| - \mu_c)e^{-v_{tr}v_{st}^{-2}} + v_{st}v_{tr}), \tag{2.8}$$

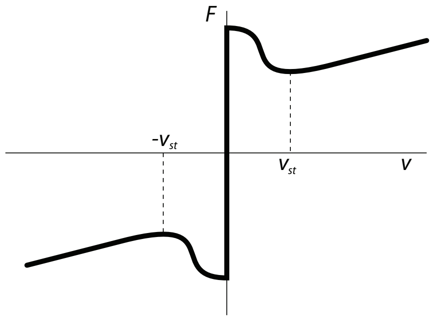
where

$$v_{tr} = \sqrt{\dot{P}0_x^{\ 2} + \dot{P}0_y^{\ 2}},$$

[2]This is because any motion in the robot will cause variations in the gCOM, which results in changes in x in the convex foot. However, the real COP also reacts to accelerations of the bodies, which is now disregarded



(a) Static, coulomb and viscous friction



(b) Friction with added Stribeck curve

Figure 2.8: The friction model

with static, viscous and coulomb friction coefficient μ_s, μ_v, μ_c, steepness of the coulomb-friction curve s and characteristic Stribeck velocity v_{st}.

Now we may note that the Stribeck effect only occurs in lubricated bearings, which in the case of ground-contacts is not present. However, to reduce the discontinuity between static friction and the Coulomb plus Viscous friction curves, the Stribeck curve serves us well with small v_{st}'s. Equation equation 2.8 therefore gives us the friction, along the direction of the velocity of the body.

The exact same friction model, with different constants, is applied to calculate the rotational friction, with the notion that only rotations around the z can can be calculated based on ground reaction forces. Rolling friction, around $[x, y]_0$, are assumed to be zero in the context of the ground reaction wrench, so there is only one non-zero component in the reaction moment, M_{GRF}.

2.1.3 Total wrench

The last step of submodel 'CF' is to incorporate all computed reaction forces and moments into the contact wrench $W^{c,i}$ and transform it to body coordinates $W^{i,i}$. This is done by aligning the translational friction with the velocity of the body and placing the resulting forces and moments in the co-vector representing the wrench,

$$W^{c,i} = \begin{bmatrix} 0 \\ 0 \\ M_{GRF} \\ \dot{p}^0_x v_{tr}^{-1} F_{fr} \\ \dot{p}^0_y v_{tr}^{-1} F_{fr} \\ F_N \end{bmatrix}^T , \tag{2.9}$$

with F_N, F_{fr} according to equation 2.4 and equation 2.8. Now transformation to Ψ_i is done by

$$W^{i,i^T} = Ad^T_{H^i_c{}^{-1}} W^{c,i^T}, \tag{2.10}$$

with the dual-adjoint mapping of coordinate changes for wrenches being

$$Ad^T_{H^b_a} = \begin{bmatrix} R^a_b & -R^a_b \tilde{p}^b_a \\ 0_3 & R^a_b \end{bmatrix}. \tag{2.11}$$

Ultimately we transpose the wrench to fit the power-bond $P.e = W^{i,i^T}$, as indicated before, to complete the contact model as is implemented in appendix E.2.

2.1.4 'EndStops'

As indicated in appendix A the range of motion of the robot is not unlimited in all joints. End-stops are therefore implemented, as given in appendix E.3. As an end-stop limiting the range of motion is the same phenomenon as a interaction with the ground, the reasoning is completely analogous to the previous section, except now taking place in $\mathbb{R}$ instead of $\mathbb{R}^6$.

The robot has 14 degrees of freedom and therefore 14 endstops. The implementation is adjusted to operate on the vector of 14 joint angles, even though some may have an unlimited range of motion, such as the rotation around $\hat{z}$ in the hip. First the incursion on the endstop is calculated and the force exerted by the end stop is then defined as

$$K, D, \bar{q}, \dot{q}, F_{stop} \in \mathbb{R}^{14}$$

$$F_{stop} = K \bar{*} \tilde{q} \; + \; D \bar{*} \dot{q} \bar{*} \tilde{q} \tag{2.12}$$

where $\bar{*}$ denotes elementwise multiplication and $\tilde{q}$ is the distance that q has overpassed the boundary posed by the endstop. Note that also here the Hunt-Crossley contact-model is applied for contact with the endstop. When a joint has no endstops, its corresponding element in vectors K, D is set to zero.

2.2 Control and actuation

Now that the physical system is modelled, a control loop can be implemented. Figure figure 2.9 shows the outline of this part of the model. The control of the walker consists basicly of two loops, the direct control loop, which is described first in this section, which is in turn governed by the setpoint generator, as a higher-level control loop, described at the end of this section.

A vector of 14 targets is provided to the controller, submodel "P(i)D" appendix E.4, of which the parameters are modified online by the 'Setpoints' submodel which also computes the setpoints. The implementation is a paralel PID controller, with tame differential part, in order to enable the switching on and off of individual, or all, parts of the controller without resulting in divisions by zero. This is done to enable the use of the control-loop for both rigidly actuated motion, with a stiff PID controller, 'passive dynamic' actuation in which a loosely set P(D) controller is generally used and even direct torque steering by the higher-level controller in the setpoint generator. The implementation follows the following general form

$$
\begin{aligned}
\vec{e} &= \vec{q_{set}} - \vec{q} \\
\vec{uP} &= \vec{K.} * \vec{error} \\
\vec{uI} &= \int (\vec{e.} * \vec{Ki}) dt \\
\vec{D_{state}} &= \int \vec{uD} dt \\
\vec{uD} &= (\vec{e.} * K - \vec{D_{state}}). * (\vec{Kd}./\vec{\beta}) \\
\vec{y} &= (\vec{uP} + \vec{uI} + \vec{uD}). * \vec{on}
\end{aligned}
\tag{2.13}
$$

where all vectors are $\in \mathbb{R}^{14}$, $\vec{y}$ are the fourteen control outputs of the controller, $\vec{\beta}$ denotes the tameness and the $\vec{on}$ vectors is used to switch on and off control on any joint without resetting the controller parameters.

The "f"-symbol, labelled 'encoders', represents a flow-sensor in the simulator, which is a bond-graph 1-junction with a signal representing the $P.f$ at that junction in the graph. In our model the flow-sensor is adapted to integrate the flow, taking into account the initial position of the robot, to deliver the joint angles. This is a construct for the simulator, as in reality, optical encoders directly deliver q for use in the control loop.
The "MSe" is the idealized bond-graph representation of an inertia-free motor delivering unlimited torque, depending on the input signal. The idealized motor is connected to a 'TF' which represents the gear-ratios present in the robot design, as given in the second diagram of appendix appendix A. Again idealized gears are taken, as later-on in the design-process, via the element R in the 'Mechanical model', various frictions can be

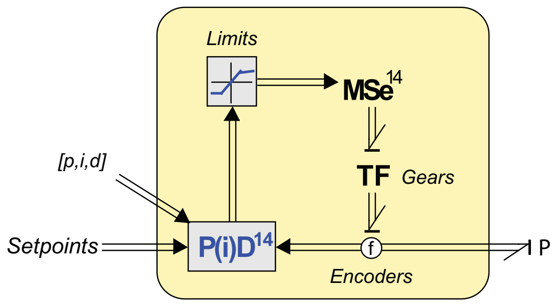


Figure 2.9: Control and actuation, taken from figure figure 2.1

added.

To ensure the designed motions will not require torques which are out of reach by the chosen motors,as indicated in appendix B, the input-signal of the “MSe” of the $P(i)D$ is limited by the “limits” block, which simply clips any of the desired moments to be delivered by the motors which are outside of their ranges. The limits have been based upon the maximum continuous torque which the motors can deliver, which can be considered an extremely conservative estimation, as the stall torque of a motor can be sustained shortly as well. However at this stage of the development, gaits and other motions should fit within the theoretic operating ranges, until tests have proven otherwise. Lateron strategies may be developed using the extra headroom the motor offers for short instances of time.

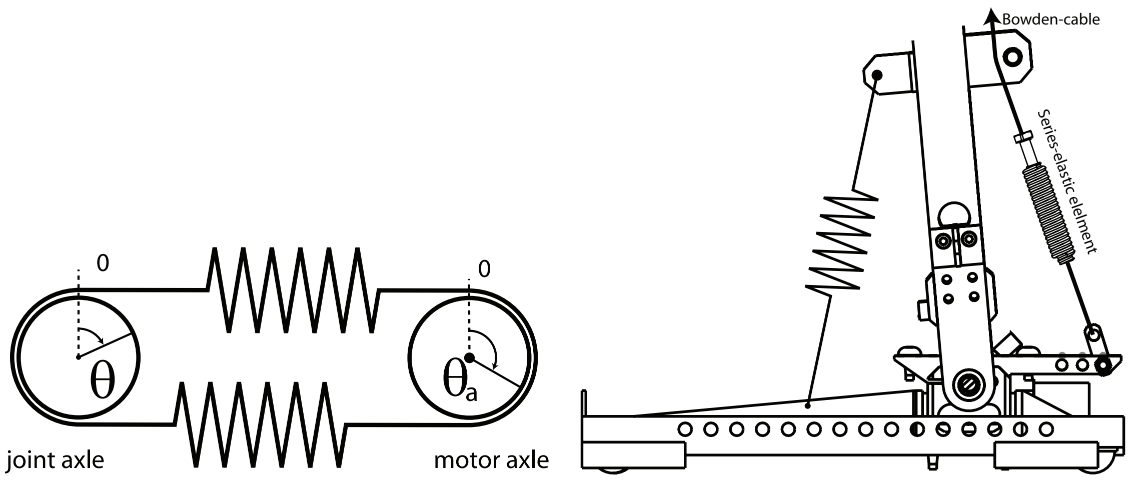
2.2.1 Series-elastic actuation

All actively actuated joints of Tulip, except the ankle-rotation around $\hat{y}$,have a series elastic element between the actuator and the actuated joint, as illustrated in figure figure 2.10(a). The ankle rotation is actuated via Bowden-cables[3] attached to the heels, much like the Achilles-tendon. Just as human tendons, Bowden-cables can hardly be used for pushing, but to overcome that problem, the cable is preloaded with an antagonistic spring. This concept is illustrated in figure figure 2.10(b). This actuation principle chiefly improves the isolation of the motor (Bowen cables hardly push) in a regular walking gait and additionally bring along the advantage of the motor being more flexible in its placement. In this case the ankle actuator is found in the upper body of the robot which resulted in an optimization of the weight-distribution, [16] [17]. However for our simulations we assume it to behave similarly to the other joints, which is fairly accurate except for a needed offset in the motor-torque of this joint for pre-loading the spring.

Using such actuation offers several advantages; it disconnects the motors from bodies on which impacts occur. When landing on the swing-foot during a walking gait, the shock is now absorbed by the mechanical structure and the elastic element in the drive-train. Secondly it absorbs other vibrations resulting in a much smoother, better looking, gait Thirdly, the fact that both the motor axle and the joint axle are equipped with encoders, we can, as the elastic element is a simple spring, measure the instantaneous torque being delivered to the joint at all times enabling the use of force-control.

Obviously, this choice introduces latency between the moment in which the desired torque is actually delivered to the joint. Choosing the K-value of the elastic element very high, allows for, when neglecting the inertia of the motor and damping in the cable and elastic element. However, obvously, this limits the chock-absorption towards the motors. The design of K will be done once the robots are built and a trade-off between latency

[3]Other examples of use: bicycle-brakes, throttle control in cars



(a) General schematic (b) Implementation on ankle

Figure 2.10: Series elastic actuation

and shock-absorption has to be made.

For this model, the series-elastic drive-train is a good justification for using a pure modulated source of effort to model the motors, the MSe-element in figure figure 2.9. This means that the motion profiles used here, as well as the joint torques, are only useable as concepts, not for direct code generation, until the inertiae of the motors and gearboxes are modelled together with the latency that the series-elastic drive-train introduces.

2.2.2 Set-point Generation

The motion profiles needed for the control loop are generated by a statemachine, which is implemented in C-code compiled into a DLL. The 20-sim simulator calls upon the DLL in the submodel "Setpoints" in figure figure 2.1.

The generator holds a 6×4 array of targets, t_1 in equation equation 3.2, for the motion profiles. At the same time another function is called, setting the P, I, D values of the controller to the desired setting, according to the state. In this way, for example a dynamic walking gait can be started after getting up on your feet. Every state corresponds to a row in this matrix.

State transitions should be defined for each task separately, so that different conditions can be used. For example while getting up, the following criterion is used:

$$\sum q_{target} - q < \epsilon, \tag{2.14}$$

so that as soon as all joints have reached their target position, the robot will start working towards the next desired posture. For a walking gait, the foot-contact signals provide a better state-transition condition.

The submodel takes care of the collection of all signals needed from the simulator and sorting them into the proper function calls of the DLL. In the initial equations, the setpoints matrix and the control-parameter matrix are preloaded into the DLL, so that this can be set via the simulator instead of external text-files or even within the code of the implementation.

The main reason for the separate implementation is that the equation-sorting that 20-sim does does not allow for easy implementation of state-machines. It would result in large files with deeply nested if-statements and it is, on top of that, unclear to the author which effects this will have on the sorting algorithm of 20-sim. The result of the separate implementation is in any case a much quicker simulation, eliminating many problems encountered in the beginning of this project. The implementation of both the DLL and the submodel are given in appendix appendix E.6, and are self-explanatory with the comments written in-line.

Chapter 3

Simulation Case: Getting up

In this chapter it is shown how the previously described model is used to simulate the robot recovering from lying on the floor in prone position, face-down, towards an upright position. The sequence of motions leading to an upright position is discussed after which simulation results are shown.

3.1 Recovery algorithm

The main goal is to end in a standing up position. There are many ways to get up and, contrary to initial intuition, this is not necessarily a 2D problem. Expanding into 3D allows for combination of motions of different joints and taking into account the proper 3D energy balance. However, in all cases, the main target is to move the COM of the robot over the feet and then straighten the legs. In our case we will approach this problem 2-dimensionally, exploiting the symmetry of the robot to ensure stability in the 3^{rd} dimension, and divide the motion into several phases, when reasoning backwards from the standing position:

1. Get on hands and 'knees'
2. Move into squatted position
3. Straighten out legs

Before starting the motion towards a hands and knees position, the feet are lifted of the ground just slightly by bending the knees. This limits the potential ground contacts to 3, torso and 2 knees, and moves the COM towards the knees. Having the COM as low as possible is beneficial for the motion, because that means that a larger portion of the total wait is lifted by the stronger motors.
Now we must choose in which direction the arms should rotate. From figure 3.1.i it becomes clear that when taking the shortest path to the desired angle, the point where the

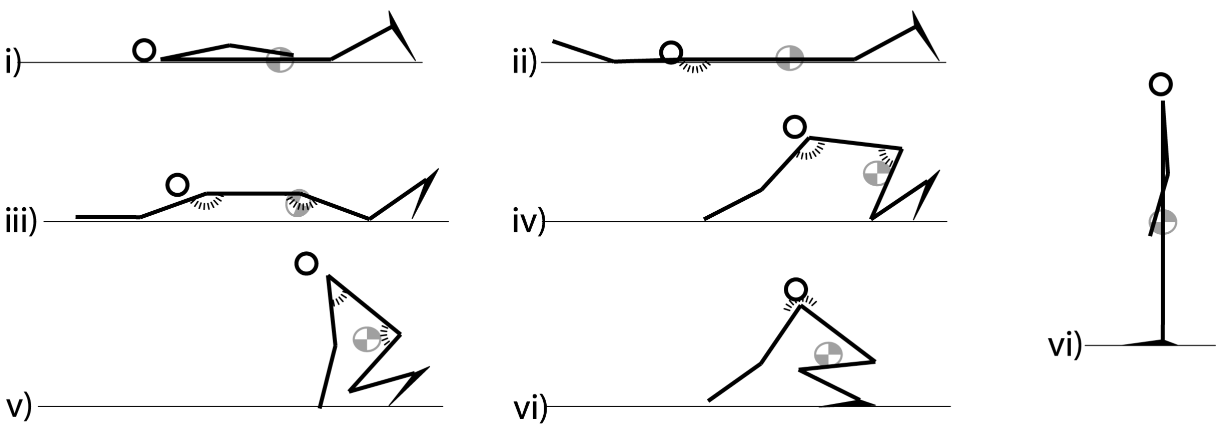


Figure 3.1: Target postures for getting up, with COM and active joints marked

tip of the arm will touch the floor will be very close to the COM, meaning that the shoulder should deliver enough torque to lift practically the full weight of the robot, resutling in a moment of $l_{arms}m_{total}g$, which it cannot, as follows from appendices appendix A and appendix B.

Having arrived to posture figure 3.1.ii), we can proceed in several ways. We can first start to with the arms only, with the hipjoint only or with both at the same time. “Arms only” is not an option, due to the weaker shoulder motors, and “hip only” would lead us into a position in which in the end again the arms need to carry the largest weight. Therefore we have chosen the posture iii) as the next step, which leaves the largest ‘working arm’ between the weakest joint, the shoulder, and the COM.

What can also be observed is that between posture *iii*) and *v*) the tips of the arms are dragged over the floor. Even though stability is guaranteed in this way, it is far from efficient. Therefore, when arriving at a position like *iv*) the ankle, knee and hip joint are flexed maximally, making the legs as short as possible. We can easily calculate the momentum needed for the robot to rotate around the front tip of the foot, landing on its feet but not toppling over backwards, given a non-inclining floor and zero interaction forces except for the ground contacts. For completeness we state that this momemtum has a minimum, the amount of energy to overcome the energetic maximum in the rotation around the foot-tip, and a maxiumum, the amount of energy needed to overcome the energetic maximum in the rotation around the heels.
As soon as the COM passes over the tip of the feet, a simple centre of pressure controller could be implemented to stabilize this motion using the knee joint. Stretching the knee joint will move the COM more towards the front of the feet as well as increase rotational inertia I_{robot}. Both effects result in increasing the energy needed to tip over backwards.

3.2 Motion profile

Throughout the motion, the PID controller of each joint is set to $\zeta = 0.7$ and a low bandwidth. The simulation as such is very computationally intensive, as it now contains throughout the simulation 3 very stiff ground contact models, 14 less stiff PID controllers and 2 much less stiff torsional springs in the ankle.

In order not to introduce vibrations into the simulation which may give rise to instabilities, a motion profile needs to be provided that is at least continuous in the acceleration profile. By choosing the shape of the jerk, $\dddot{q}$, to be continuous but not continuously differentiable, we guarantee $q(t)$ to be of class C^3 and $q(t) \in \mathbb{R}\forall t$. We choose

$$\dddot{q}(t) = sin\left(4\pi \frac{|t - \frac{\Delta_t}{2}|}{\Delta_t}\right), \tag{3.1}$$

with

$$\Delta_t = (t_1 - t_0).$$

When integrating this to obtain a $q(t)$ several integration constants are introduced, which can be found by applying boundary conditions such as beginning and end position, speed and acceleration. Doing so we find

$$q(t) = \begin{cases} q_0 & t \leq t_0, \\ q_0 + \frac{4\Delta_q}{\Delta_t^2}\left(\frac{\Delta_t}{4\pi}^2\left(cos(4\pi\frac{t-t_0}{\Delta_t}) - 1\right) + \frac{(t-t_0)^2}{2}\right) & t_0 < t \leq t_{mid}, \\ q_0 - \frac{4\Delta_q}{\Delta_t^2}\left(\frac{\Delta_t}{4\pi}^2\left(cos(4\pi\frac{t-t_0}{\Delta_t}) - 1 + (4\pi)^2\right) + \frac{(t-t_0)^2}{2} - \Delta_t(t - t_0)\right) & t_{mid} < t \leq t_1, \\ q_1 & t > t_1 \end{cases}, \tag{3.2}$$

with

$$\Delta_q = (q_1 - q_0),$$

$$t_{mid} = t_0 + \frac{\Delta_t}{2},$$

which results in figure 3.2 where the jerk, acceleration, velocity and position. Scaling of each magnitude is relative to the others, in other words $1[rad] = 1[rad\ s^{-1}] = 1[rad\ s^{-2}] = 1[rad\ s^{-3}]$. Additionally Δ_t can be chosen according to

$$\Delta_t = \sqrt{\frac{|8\Delta_q|}{\ddot{q}_{max}}},$$

limiting the maximum required acceleration required for the motion profile. In combination with the various inertias connected to each motor, we can straight-forwardly implement the motion profiles in a way that takes into account the maximum nominal continuous torque of the motors. However, in the case that there is more than 1 point

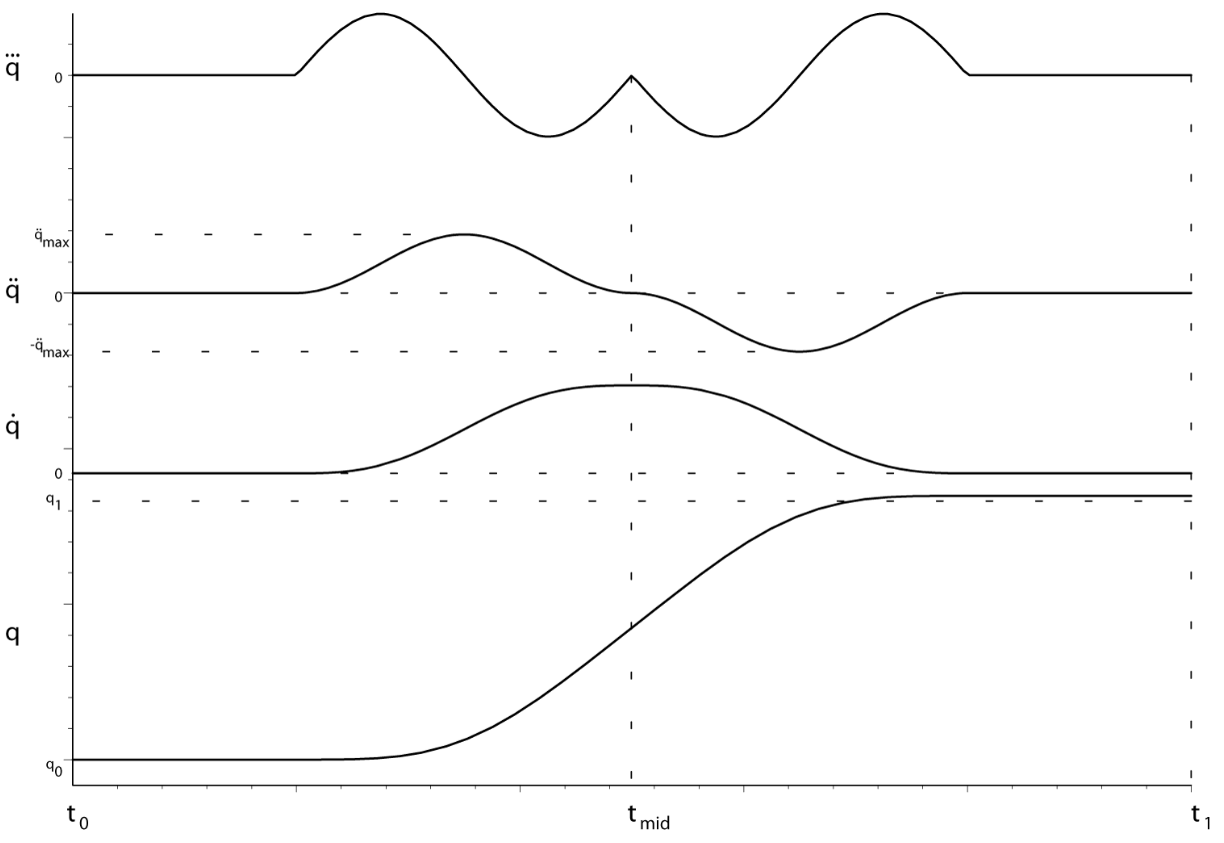


Figure 3.2: Jerk, accelleration, velocity and position

of interaction on the robot, actual limits on the maximum required torque are hard to compute without more detailed knowledge of the encountered frictions and other factors. Obviously this motion profile is not used for the shoulder joints at the moment of giving the impulse between postures iv) and vi) of figure 3.1.

3.3 Results

When running the simulation, the results are given in the following table and in figure figure 3.3. We can see that the algorithm works and have verified so far that the design of the robot, in combination with the friction parameters chosen during this simulation correspond. In other words, the limitations of the motors with respect to limb-length and other factors are chosen well and the robot will have enough power to execute this motion in real life as well, given that no unexpectedly large frictions occur in joints.

The 20-sim file "gettingUp.emx" contains the simulation as described in this chapter, "gettingUp.avi" shows the real-time execution of the recovery algorithm. The model is outlined in this chapter and it's implementation is to be found in appendix appendix E and more detailed results are given in appendix appendix D.

Average execution time:	$159.98[s]$	[1]
Simulated time:	$12[s]$	
Energy used	$150[J]$	

Table 3.1: Simulation results

#	$ankle_y$	$knee$	hip_y	$shoulder$
$i)$	-60	45	0	10
$ii)$	-60	45	0	180
$iii)$	-60	70	-20	200
$vi)$	-60	150	-150	260
$vii)$	0	0	0	0

Table 3.2: Joint positions for target postures

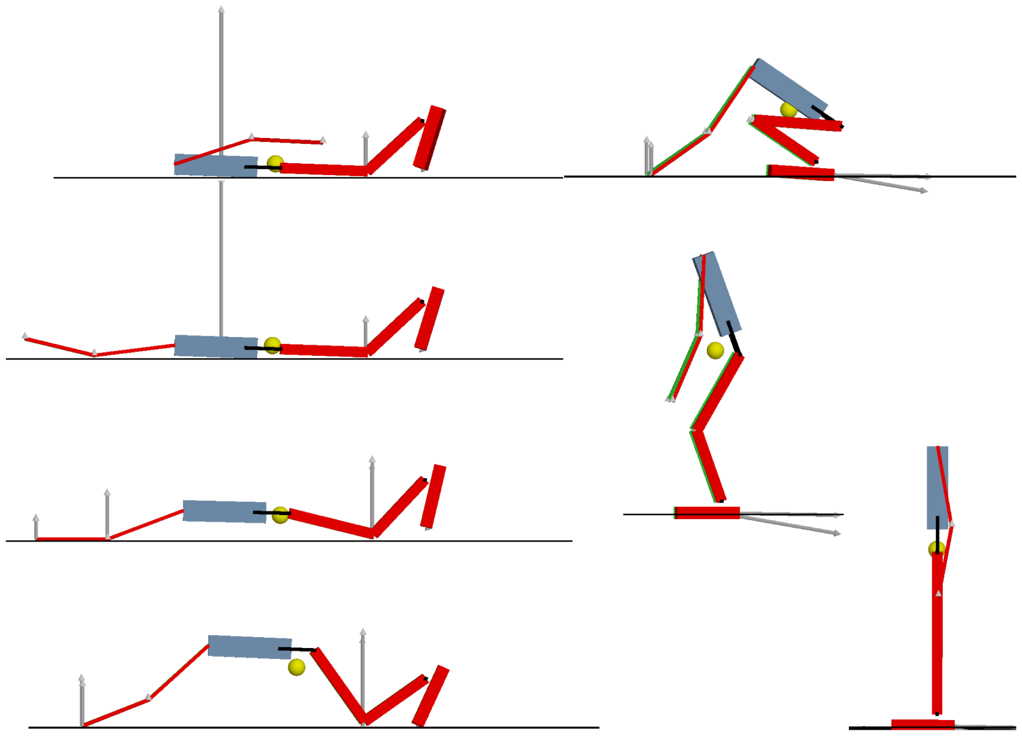


Figure 3.3: Screenshots of the model getting up on two feet

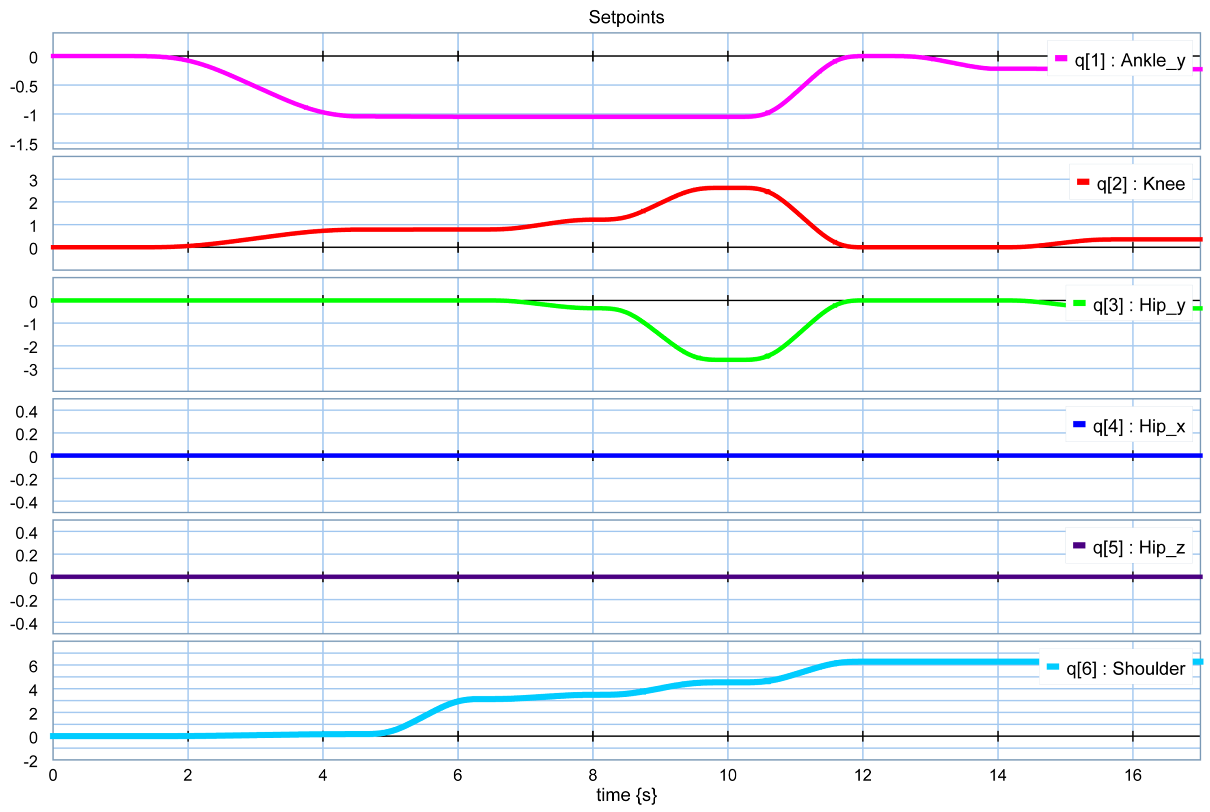


Figure 3.4: Joint motion profiles, left limbs

Chapter 4

Application of ZMP to a dynamic walker

Zero Moment Point control in passive dynamic walking

P.H.M. Daemen, ir. G. Van Oort, prof. dr. ir. S. Stramigioli

Abstract— **In humanoid biped locomotion, the Zero Moment Point (ZMP) is a widely used concept to ensure dynamic stability of the robot. The role the ZMP usually plays to achieve this, is that of a stability criterion, which relies on having full control over the motions of each joint. Applications exist both as an on-line control algorithm, as well as as a criterion for off-line setpoint generation.**
When considering passive dynamic motions, more soft control algorithms are used, which implicates that less rigid control over the motions of each joint can be guaranteed. Discrete controllers that evaluate a whole step and alter the execution of the consecutive step are available to create stable locomotion. In addition to this, this paper proposes a method for using the ZMP as an on-line, which means during the step's swing phase, gait analysis tool. The location of the ZMP is related to the overall motion of the robot body, which in turn is compared to the desired situation. By means of an application of the algorithm towards a humanoid robot in development, Tulip, it is shown that such reasoning results in a solution and produces achievable control outputs.

I. INTRODUCTION

The goal of this research has been is to implement and evaluate the concept of Zero Moment Point(ZMP) [8] into gait control of the 3D passive dynamic walker Tulip, which is being developed within the collaboration of the 3TU federation, to compete in the RoboSoccer league 2008 in Suzhou, China. The concept of ZMP will be introduced, outlined and reviewed towards its applicability in the context of passive dynamic walking and the robot Tulip, which is elaborated on in [2], specifically.
The type of application of the ZMP to achieve dynamic stability for humanoid robots is that of a stability criterion, which relies on having full control over the motions of each joint. When considering passive dynamic motions, more soft control algorithms are used, which implicates that less rigid control over the motions of each joint can be guaranteed. 3D walking gaits can be computed and executed for simple walking systems, and also for more complex walkers, stable gaits have been achieved. However, such gaits are based upon discrete controllers that evaluate a whole step and alter the execution of the consecutive step, to obtain stable locomotion.
In addition to such strategies, this paper proposes a method for using the ZMP as an on-line, which means during the step's swing phase, gait analysis tool. The location of the ZMP is related to the overall motion of the robot body, which in turn is compared to the desired situation. This paper aims to outline the foundation on which we start off to find such algorithms, that we can, based upon extremely simple walkers, achieve stable gaits when applying an on-line gait control algorithm. By applying this algorithm to the 14 DOF biped robot Tulip, it is shown that such reasoning converges to an optimizable solution and produces achievable control outputs.

This work is executed in the context of the Dutch 3TU federation, supervised by ir. G. Van Oort, prof. dr. ir. S. Stramigioli and prof. dr. ir. J. Van Amerongen

Peter Daemen is a graduate student at the Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands; phmdaemen@gmail.com

II. ANALYSIS

A. Dynamic walking

A dynamic walking gait is designed such that by using the potential and kinetic energies of the walker, a stable walking cycle is established with a minimum of energy introduced by actuation. We will consider the simplest walker that is possible, a compass walker as depicted in figure 1. It has massless legs, which rotate around the hip, with forward hip angle $\phi_h(t)$ and sideways leg-splay $\phi_s(t)$ and mass m as a point mass in the hip P_w^0. $T_w^{0,0} \in \mathbb{R}^6$ denotes the twist, [6], of the walker in Ψ_0 and both legs are in the plane spanned by m and $\hat{z}_0$. The stance leg is standing on the ground in the origin of Ψ_0, a left hand coordinate frame which has the x-axis parallel to the overall walking direction. For any frame Ψ_i, it's axes will be referred to as $\hat{x}_i$, $\hat{y}_i$ and $\hat{z}_i$

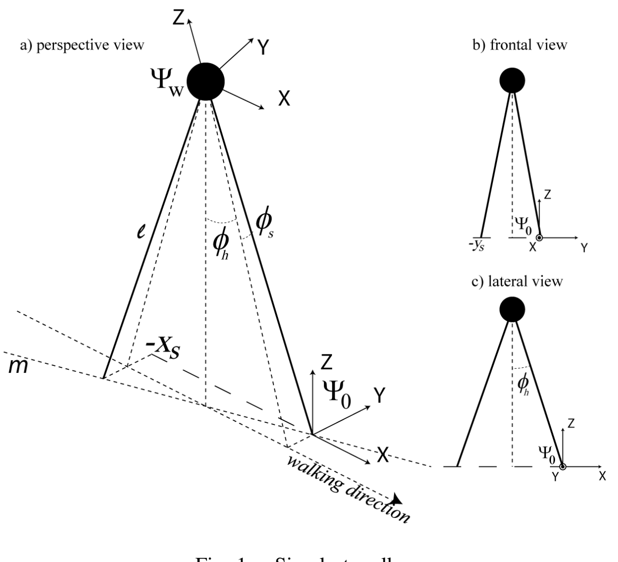


Fig. 1. Simplest walker

$$T_w^{0,0} = \begin{bmatrix} \omega \\ v \end{bmatrix}, \omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, v = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (1)$$

The sequence of positions of a walking gait is depicted in figure 2, where $\dot{\xi}$ denotes $\ell\omega_y$ and $\dot{\zeta}$ denotes $\ell\omega_x$. Figure 2 shows the simplest walker at t_0, with its mass having velocity vector $\dot{\xi}_0$ in the $[x,z]$ plane and $\dot{\zeta}_0$ in the $[y,z]$ plane. Then at t_{mid}, we can see that $\dot{\zeta}$ has reduced to zero and $\dot{\xi}$ to some minimum, while the position of the mass is raised Δh from its original height.
At t_{step}, the total step time, the walker's hip mass has reached a certain velocity, $\dot{\xi}_s$ and $\dot{\zeta}_s$, and impact occurs. Assuming instantaneous inelastic contact, on impact an impulse directed along the landing-leg instantaneously stops the rotation of the walker in its original direction and redirects it to rotate around the new stance-leg, giving rise to $\dot{\xi}'$ and $\dot{\zeta}'$. However, as the new velocities are a projection of the old velocities onto the tangent of the new sphere, they are reduced to

$$\begin{aligned} \dot{\xi}' &= \dot{\xi}_s \cos\phi_h^* \\ \dot{\zeta}' &= \dot{\zeta}_s \cos\phi_s^*, \end{aligned} \quad (2)$$

where $\phi^*_{(h,s)}$ denote the two angles at instant t_{step} for ϕ_h and ϕ_s. Hereby we have assumed the motions in the two planes to be uncoupled. To achieve a periodic gait $\dot{\xi}_0 = \dot{\xi}'$ must hold.

When considering the energy balance on impact, even for an idealized system without friction, it is obvious that energy needs to be inserted, to ensure that for the next swing phase, there is enough energy in the system to at overcome energetic maximum occurring at t_{mid}. This means that just before impact,

$$E_{kin} - E_{impact} \geq \Delta E_{potential}, \quad (3)$$

should be satisfied. From figure 2 we see that for the lateral view the on-impact balance looks like

$$\begin{aligned} E_{impact} &= \frac{1}{2}m(\dot{\xi}'^2 - \dot{\xi}^2) \\ E_{kin} &= \frac{1}{2}m(\dot{\xi})^2 \\ \Delta E_{potential} &= mg\Delta h, \end{aligned} \quad (4)$$

with m the mass of the walker, g the gravitational pull, Δh the height difference between the starting position and the mid-swing height, $\dot{\xi}$ the starting velocity and $\dot{\xi}'$ the after impact velocity. Analogous to equation 4 we have

$$\begin{aligned} T^* - T^*_{impact} &\geq \Delta V, \\ T^* &= \frac{1}{2}T^T \mathscr{I} T \\ T^*_{impact} &= \frac{1}{2}(T - T')^T \mathscr{I}(T + T') \\ \Delta V &= mG^0(P_w^0(t_{mid}) - P_w^0(t_0)), \end{aligned} \quad (5)$$

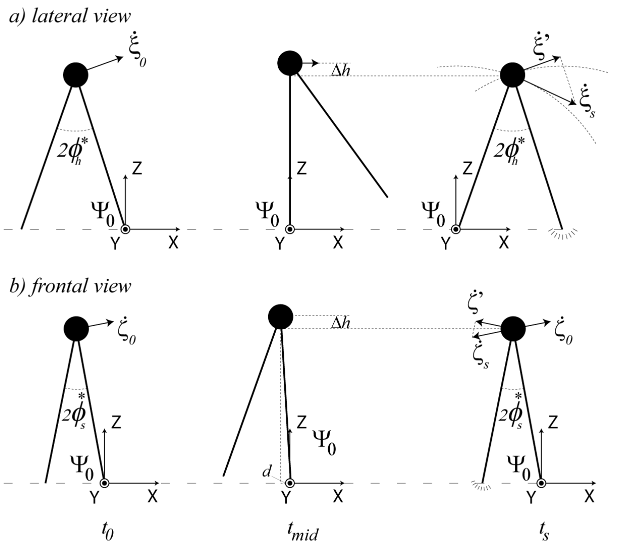


Fig. 2. Key moments in a gait: push-off t_0, mid-swing t_{mid} and impact t_{step}

with $G^0 = [\ 0\ 0\ g\ 0]$ being the gravitational potential field expressed in Ψ_0, $\mathscr{I}$ the inertia tensor of the walker, P_w^0 the walker's center of mass (COM) expressed in Ψ_0, T^* the pre-impact kinetic co-energy, T^*_{impact} the energy lost on impact, and ΔV the difference in potential energy between mid-swing phase, at t_{mid}, and the moment of impact t_i.

In Van Oort's [7] model of the simplest walker, energy injection takes place by extension of the pre-impact support leg. When the robot would be equipped with actuated feet, the energy injection can be done by pre-impact ankle push-off, as investigated by Franken [3], or ankle actuation during the swing phase. In the case of pre-impact push-off the contribution consists of potential energy, elevation of the stance leg due to push-off, and kinetic co-energy, and in the case of swing phase ankle actuation as pure kinetic co-energy T^*.
In figure 2 it becomes clear that the energy loss on impact depends on ϕ_s and ϕ_h, and the 'post-impact' velocities are dependant as well on their derivatives. When looking at figure 1, a special situation is depicted in which the impact occurs exactly so that

$$\begin{bmatrix} \phi_h' \\ \phi_s' \end{bmatrix} = \begin{bmatrix} -\phi_h^* \\ \phi_s^* \end{bmatrix}, \quad (6)$$

which means that the impact exactly takes place in-plane with both $\hat{z}_0$ and m, from figure 1. In this case, a symmetric gait, the components in the direction of m of the added and lost momentum cancel out and the $\hat{z}$-component remains, provided that during the swing phase the exact momentum which is going to be lost on impact is fed into the system. For this situation, Van Oort [7] has shown that without explicit knowledge of the limit cycle of the walker, a stable gait-controller can be derived by ensuring $\dot{\zeta}(t_{mid}) = 0$.

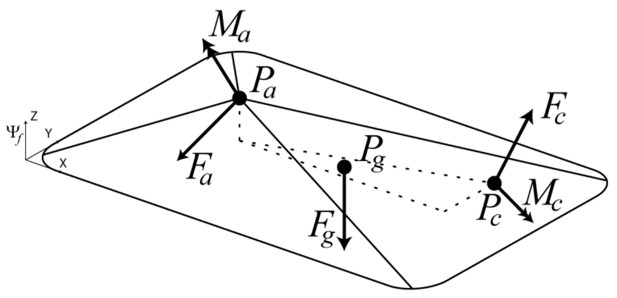


Fig. 3. Forces on the stance foot

This model for the simplest walker neglects many aspects of a realistic 3D walking robot, such as non-zero inertia of legs, knee-flexion and upper-body influences. For such more complicated models, the regions of stability/periodicity for parameter deviations are very small, whereas this simple walker can be stabilized more straightforwardly. The dynamics of the more complex body of humanoid robots may be seen as disturbances upon the dominant inverse pendulum behaviour. In order to achieve a periodic gait in the presence of such disturbances, a controller, such as the one proposed below, can be implemented. The goal of any gait-control is to decrease the sensitivity towards disturbances like pushes, steps down and slopes, as well as parameter deviations and variations, into which the 'missing dynamics' can be counted as well.

B. Dynamic stability and zero moment point

To analyse the dynamic stability of a bipedal robot, we must look at the relation between the wrenches acting upon its body and the wrench resulting from interaction with the ground. So far we have only discussed the simplest walker which has no feet, however it is useful to venture into an analyses of the forces and moments acting upon the stance-foot where afore mentioned wrenches interplay, as depicted in figure 3.

In figure 3 we define three coordinate frames Ψ_c, Ψ_a, Ψ_g having the same orientation. Their origins are located at points P_c, P_a, P_g, the points of action of the contact wrench W^c, the total body wrench W^a and the gravitational wrench W^g. The total body wrench expressed in Ψ_a is

$$W^{a,a} = [\, M_a \quad F_a \,] \qquad \text{with} \qquad M_a, F_a \in \mathbb{R}^3,$$

which is the resultant wrench of all dynamics of the rest of the body acting on the ankle. In other words, it constitutes all influences of the rest of the body on the foot, including any interaction forces encountered by any of the limbs of the robot. The foot's gravity wrench $W^{g,g}$ and the ground interaction wrench $W^{c,c}$ are defined as

$$\begin{aligned} W^{g,g} &= [0\,0\,0\,0\,0\,F_g] \\ W^{c,c} &= [M_c\,F_c], \end{aligned} \tag{7}$$

with

$$\begin{aligned} M_c &= [0\,0\,M_{fr_z}], \\ F_c &= [F_{fr_x}\,F_{fr_y}\,F_N] \text{ and} \\ F_g &= -m_f g \end{aligned}$$

where m_f is the mass of the foot, g the gravitational constant, F_{fr_*} denotes the translational friction components, M_{fr_z} denotes the rotational friction along the vertical axes with respect to the floor. In [8] Vukobratovic defines point P_z, with Ψ_z, on which $W^{z,z}$ compensates all moments around $\hat{x}_f$ and $\hat{y}_f$ acting on the foot, by means of a pure force.
The ZMP does not exist outside of the support polygon of the robot, which is spanned by either the outer edges of the stance foot alone, or of both feet in double-support positions. At P_z we must note that the moment around $\hat{z}$ is not necessarily zero, but as this degree of freedom (DOF) hardly influences the stability of biped robots, this moment disregarded. P_z can be found by equating the wrenches on the foot,

$$W^{f,a} + W^{f,g} + W^{f,c} = [0\,0\,\bullet\,\bullet\,\bullet\,0]$$

from which it straightforwardly follows that the ZMP expressed in Ψ_f is

$$\begin{aligned} P_z &= \begin{bmatrix} zmp_x \\ zmp_y \\ 0 \end{bmatrix} \\ &= \frac{1}{F_N} \begin{bmatrix} M_{a_y} + F_{a_x} p_{a_z} - F_{a_z} p_{a_x} + m_f g p_{g_x} \\ -M_{a_x} + F_{a_y} p_{a_z} - F_{a_z} p_{a_y} + m_f g p_{g_y} \\ 0 \end{bmatrix}, \end{aligned} \tag{8}$$

with $P_a^f = [p_{a_x}\; p_{a_y}\; p_{a_z}]^T$ and $P_g^f = [p_{g_x}\; p_{g_y}\; p_{g_z}]^T$ represent the coordinates of points P_a and P_g expressed in coordinate frame Ψ_f. We are now in need of an expression for $W^{a,a}$, which can be derived from figure 4.
Assuming zero interaction, we can state that the total wrench $W_t^{0,i}$ consists of the constraint wrench $W_q^{0,i}$ keeping the bodies connected via joint q together and the gravitational wrench $W_g^{0,i}$. Knowing that for the constraint wrenches, opposite reaction forces act upon the other connected body, we can find an expression for $W^{f,a}$ equate the wrench exercised body 1 by the ankle as

$$W_{q1}^{0,1} = W_t^{0,1} - W_g^{0,1} - W_{q2}^{0,1}$$

where $W_g^{0,i}$ denotes the gravitational wrench on body i and

$$-W_{q2}^{0,1} = W_{q2}^{0,2} = W_t^{0,2} - W_g^{0,2} - W_{q3}^{0,3} - W_{q4}^{0,4} - W_{q5}^{0,5}.$$

Assuming the absence of closed kinematic chains, a similar recursion can be made throughout the kinematic chain of a more advanced humanoid robot, which results in

$$(W^{f,a})^T = Ad_{H_f^0}^T \left(\sum_{i=1}^{n} W_t^{0,i} - \sum_{i=1}^{n} W_g^{0,i} \right)^T. \tag{9}$$

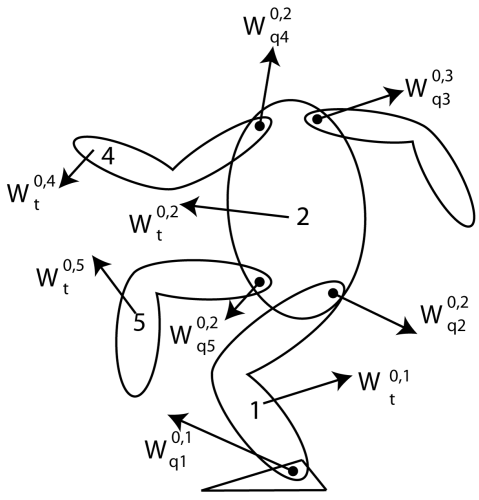


Fig. 4. Determining $W^{a,a}$, by means of constraint forces

This means to keep track of the exact motions of all bodies[1], which in simulation can be done, but in reality poses practical problems. A more realistic approach would be to directly measure the constraint forces in the ankle, instead of measuring the accelerations of all bodies, for example by positioning a 6 DOF force and torque sensor between the ankle and the foot.

Coincidently, while the robot is standing on the floor, it will in general not have just one point with which it interacts with the floor. We can however make a weighed sum of all F_N resulting from the ground-contact,

$$F_N = \sum_{k=1}^{n} F_{N_k} \text{ and } P_{grf} = \frac{1}{F_N} \sum_{k=1}^{n} F_{N_k} P_k,$$

where n is the number of contact points, F_{N_k} is the normal reaction force on point k and P_k denotes the position of contact point k. P_{grf} is also known as the centre of pressure (COP), the weighted average acting point of all normal forces interacting with the foot.

We know that when the robot is standing in a stable position, or executing dynamically stable motions, $W^{c,c}$ must act in P_z, to ensure that $W^{c,c}$ suffices to keep the robot standing. This makes in the dynamically stable situation P_z is equal to the COP in figure 3 and F_N the component along axis $\hat{z}$ of F_c. In fact, from the definitions of both ZMP and COP we can deduce that they almost always coincide except when the ZMP would fall outside of the support polygon, in which case the COP is at the edge of the polygon and the ZMP does not exist. This situation is an unstable situation, because $W^{c,c}$ cannot act on a point which would give it's forces a long enough arm to counteract the moments acting on the foot. This means the foot will start a rotation around the edge, or around the corner, where the COP is located at that moment. This is called an unpowered DOF, because no actuation is possible in that new DOF, which means that the system becomes underactuated. The foot, and with it the whole robot, will start an uncontrolled rotation.

[1]For interaction forces, equation 9 is augmented with $-\sum_{i=1}^{n} W_{ext}^{0,i}$

Our simplest walker, introduced before, has no actuated degrees of freedom to manipulate the location of the ZMP. Added to this, the foot of this walker is simply a point contact, which means that the ZMP and COP hardly ever coincide. Foot-placement is the only way of ensuring a periodic gait in this simple system.
However, more complex robotic gait generation often relies upon the criterion of maintaining the ZMP within the support polygon of the robot, to ensure dynamically stable motion planning, as for example in [1]. In other cases, control loops have been established, for example in Honda's Asimo [5], in which the ZMP is manipulated in order to maintain stability during gaits.

In [4], Goswami et al. define a point called the foot rotation indicator (FRI) in such a way that it always coincides with the ZMP, except for the fact that the FRI is allowed to exist outside of the support polygon, where Vukobratovic refers then to that point as the fZMP, fictitious ZMP. From this moment on, the point where the resultant W^c acts, P_c, will be referred to as the COP, and P_z, which is the FRI or (f)ZMP, simply as the ZMP, according to the definition of Vukobratovic' fZMP.

Multiplication of equation 8 with F_N results in M_{total}, the total moment in need of compensation to ensure dynamic stability. At the same time $P_c F_N$ denotes the amount of moment that is actually being compensated by the interaction with the ground. We define the unbalanced moment as

$$M_u = -F_N * ((P_z - P_c)), \tag{10}$$

which indicates the moment with which the robot is accelerated around the new unpowered DOF. In our simplest walker, we have no feet, therefore $P_c \triangleq P_a$ and P_z is a true (foot) rotation indicator of the walker.

III. Gait control

Having established an understanding of the concepts outlined so far, we can proceed to design a passive dynamic walking gait, which will be augmented by a gait controller. As stated at the end of section II-A, we will treat the additional dynamics of more complex robots and parameter variations as disturbances upon the gait. The ZMP accounts for the additional dynamics and forms a basis for this controller.

A. Gait design

When designing a gait, we need to define a set of parameters, out of which some may be chosen freely and others follow from those choices for the variables, which are defined in figure 2 and figure 1.

We start by selecting a target speed $\tilde{v}$ and a corresponding t_{step}, from which directly follows the forward step length x_s. Then we choose the sideways foot-placement with respect to the COM, see figure 2, d, realising we do not want to deviate too much from the $\hat{x}_0$ direction of walking. Hereby the targets for the hip, ϕ_h^*, ϕ_s^* are specified.

$$\begin{aligned}
\Rightarrow \quad & x_s = \tilde{v}\, t_{step} \\
& y_s = 2d \\
\Rightarrow \quad & \phi_h^* = \arcsin \frac{x_s}{2\ell \cos \phi_s^*} \\
& \phi_s^* = \arcsin \frac{y_s}{2\ell} \\
& \Delta h = \ell (1 - \cos \phi_h^* \sin \phi_s^*)
\end{aligned}$$

B. ZMP based control

Having established the parameters as given in equation 11 we can start to solve the equation of motion by looking at the boundary conditions that exist for the differential equation. Solving these equations using the twists and wrenches defined before will create a large and hard to solve differential equation, with boundary conditions

$$\begin{aligned}
H_w^0(\phi(t_{step})) &= \int_0^{t_{step}} \tilde{T}_w^{0,0}(t) H_w^0(t) dt + H_w^0(0) \qquad (11) \\
\frac{1}{2} T'^T \mathscr{I} T' &\geq mg\ell (1 - \cos \phi_h^* \cos \phi_s^*)
\end{aligned}$$

which combined with the dynamics rather quickly grows out of hand. Instead we choose to split up the problem into two dimensions, which results in a set of equations for the boundary conditions on impact. This split up will result in an imbalance in the energies, as stated before, but is justified as long as the difference between the forward and sideways velocities remains big enough.

These equations we will solve by finding a function $M_w(t)$ needed to achieve $\phi_h(t_{step}) = \phi_h^*$ while satisfying the requirement for a minimum $\omega'_w = \dot{\phi}(t) \hat{=} \frac{\xi'}{\ell}$, analogously to equation 2, plus stating that $\omega_0 = \omega'$, which means that we will add all energy required during the swing phase, and not on impact. We have to solve, for both dimensions,

$$\begin{aligned}
\omega' = \omega_0 &= \cos \phi^* \omega_{step} \qquad (12) \\
\omega_{step} &= \omega_0 + \frac{1}{I} \int_0^{t_{step}} M(t) dt \\
\Delta \phi = \phi_{step} - \phi_0 &= \int_0^{t_{step}} \left(\omega_0 + \frac{1}{I} \int_0^t M(\tau) d\tau \right) dt,
\end{aligned}$$

with additionally

$$\frac{1}{2} T'^T \mathscr{I} T' \geq mg\ell (1 - \cos \phi_h^* \cos \phi_s^*)$$

and

$$T' = T_w^{0,0'} = \begin{bmatrix} \omega_x \\ \omega_y \\ 0^4 \end{bmatrix},$$

which altogether provides a solvable optimal control problem, which can be minimized towards M_w, giving the optimal path for P_z during the swing phase.

Deviations from the ideal P_z should then result in changes of $W^{0,w}$. However, as stated before, we cannot achieve this in the simplest walker, so rearranging the foot-placement is the only other option. Several solutions for this exist, but in any case it will result in a change of direction, a different walking speed or both.

Now we take the simplest walker with a foot connected by a 2 DOF actuated ankle to the stance leg and we choose only to actuate the ankle with the aim to maintain the ideal P_z trajectory. This will lead to the foot placement that gives rise to the designed stable gait, with sufficient kinetic energy for a periodicly stable gait. However, from equation 10 we conclude that this can be only achieved and supported by actuation, while if $P_z = P_c$, in other words, as long as the ZMP remains within the support polygon of the robot. This limitation can be included in the optimal control problem of equation 12.

Using equation 8 we can define several control rules to control the ZMP and we can see that the ZMP can be used as an indicator of the resulting energy on impact. Therefore, the ZMP is in this case not a criterion for dynamic stability, but rather a criterion towards 'energy-direction' for the walker. At the moment that equation **??** is no longer satisfied, we can deviate from the chosen foot-placement, side-stepping or other countermeasures against falling, and in the next step change the injected energy at push-off to re-establish the initially designed gait.

IV. Application to a 3D passive dynamic walker

Having established the above interpretation of P_z we now evaluate this concept by means of a simulation case-study on the 3TU robot Tulip. The design and modelling of this robot is sufficiently discussed in [2].

A. Tulip ankle actuation

As indicated in [2] rotation around $\hat{x}_a$ is passively actuated by a torsional spring, which means we cannot control the dynamics of side-ways falling throughout the swing phase. Consequently, in the process of gait design, t_{step} now determines d, K and ϕ_0, being the spring constant and the zero position of the torsional spring. This means that to ensure stability, we need to devise a foot-placement strategy.
By choosing K and ϕ_0 we can manipulate the shape of the potential energy $V_{y,z}(\phi_s)$, as depicted in figure 5. A flatter curve results in a slower sideways falling acceleration in the (y,z)-plane, as well as in a smaller ζ', and kinetic co-energy, needed to achieve the mid-stance position, $COM_y = A_y - d$, as defined in figure 2. However, it also implies a higher sensitivity to disturbances, as any impulse destabilizing the gait is hardly counteracted. The lower graph in figure 5

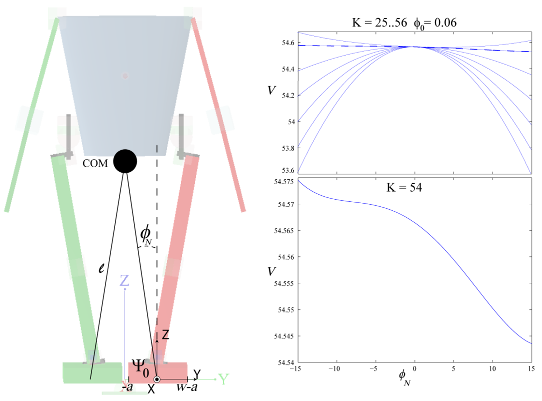


Fig. 5. Tulip vs. simple walker

shows the shape of the energy curve for the value of K that compensates gravity, while the zero position of the spring is only slightly off centre. The shape of this curve is what is desirable, ensuring an inward moment, even when the $\phi_s = 0$, however a steeper curve is beneficial.

The actual time required to pass the three gait phases from figure 2 is t_{step}', which is dependant on the excursion of the pendulum from its central position and ζ'. This means that, when we place the foot in such position that the initial excursion, $A_y - com_y$ results in falling time $t_s tep$, then we can rely on the gait to be periodic with period $P = 2t_{step}$, provided that the initial speed is correct. If we choose the target foot placement in the $\hat{y}_0$ direction as the measured $com_y + dy$, this is ensured while assuming only small side-ways disturbances to be present.

While $m = 11.6[kg]$ and $\ell = 0.48[m]$ for this robot, we obtain for variations of the torsional spring constant K and its zero position ϕ_0 the upper-right plot of figure 5, where the lower-right plot shows $K = 54$ in detail. The potential energy is of the form

$$V = mg\ell\cos\phi_N + \frac{1}{2}K(\phi_0 - \phi_N)^2, \tag{13}$$

where we can substitute the Taylor expansion $\cos x \doteq 1 - \frac{x^2}{2}$, which would give the flattest curve around $K = 54.6$, the dotted line in figure 5. Taking into account that $-aF_N \leq M_{a_x} \leq (w-a)F_n$ is limited by the foot width, we can define a region of stability, in terms of ϕ_N, where the t_{step} is influenced (positively) by the torsional spring, outside of this, the inverse pendulum reverts to free-fall, invalidating the chosen t_{step}. Another factor manipulating the energy curve in this plane would be to actuate the hip rotation around $\hat{x}$. For example, actuation in such manner that the upper-body stays levelled with the floor, effectively shortening ℓ slightly as the leg is rotated counter-clockwise with respect to the foot, which is what happens if you move the 'inverse pendulum' (COM) to $\phi_N = 0$.

ϕ_s	$10[deg]$	m_w	$11.5[kg]$
ℓ	$0.48[m]$	g	$9.81[ms^{-2}]$
$\tilde{v}$	$0.694[ms^{-1}]$	t_{step}	$0.5[s]$
x_s	$0.35[m]$	ϕ_h	$21.2[deg]$
w_0	$104[degs^{-1}]$	dy	$0.06[m]$
K	$58[Nmrad^-1]$	ϕ_N0	$-7[deg]$

TABLE I
GAIT PARAMETERS

B. Application of passive dynamic ZMP control

As one of the degrees of freedom which is essential for proper 3D ZMP manipulation is passively actuated, we are forced to split up the problem of gait stability in forward and sideways stability. The sideways stability was discussed in the previous section, and is established by proper design. Having selected the parameters for the torsional spring and the sideways foot placement, our t_{step} is fixed and the forward step length x_s follows from this, assuming a target walking velocity of $\tilde{v}$.

As can be seen in figure 5 the geometry of the robot imposes a minimum value for ϕ_s of the simplification to be used, to prevent the feet from overlapping. Choosing a target walking speed of $\tilde{v} = 2.5[\frac{km}{h}] = 0.68[\frac{m}{s}]$ results in the gait characteristics given in table I.

Now we can establish the desired trajectory for the ZMP in ankle coordinates $P^a_{zmp_x}$ by solving equation 12. Candidate function

$$M_x(t) = a\sin\frac{\pi * t}{t_{step}} + bsin\frac{2\pi t}{t_{step}}, \tag{14}$$

while substituting the values from table I makes

$$a = 1.115 \quad \text{and} \quad b = -13.95,$$

resulting in a pofile for zmp_x as given in figure 6.

The zmp_x is obtained by dividing equation 14 with a $F_N' \hat{=} m_w * g$, disregarding strong variations in F_N. The distances of the ankle towards the edges of the support polygon impose $-0.05 \leq zmp_x \leq 0.13$. A simple calculation shows that this profile satisfies these boundaries, which means that this profile of the ZMP can be supported and

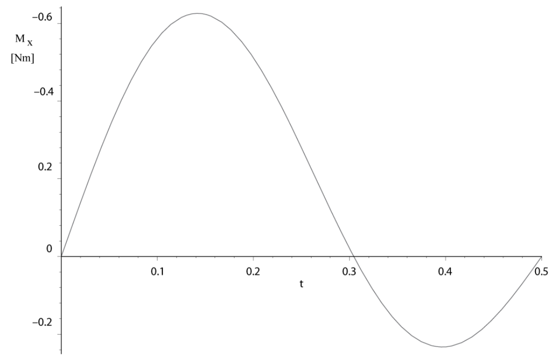


Fig. 6. Candidate profile for M_x, starting at t_0

Chapter 5

Conclusions and recommendations

The model as described in chapter 2 has proven sufficient to start simulating motion schemes for different operations for Tulip. The recovery algorithm is executed successfully and simulation times are at an acceptable level, to allow for iterative design of motions. A set of parameter-settings for the model is save in the different .emx-files which are aimed at accurate simulation, fine-tuning these settings towards the goal of the ongoing iterations can improve simulation times further. For example while creating a rough idea of a new recovery algorithm, the ground-contact parameters can be relaxed, whereas the creation of a stable gait needs a very stiff ground-contact for precision.

20-sim has proven to be an adequate environment for the simulation of this 14-DOF robot, especially in the versions 4.0.1 and beyond. However, the implementation of a DLL has improved the simulation stability. Removing the state machine and the non-linearities of the varying motion profile out of the equations of the simulator has reduced simulation time drastically as well as removed all instabilities.

The mechanical design of Tulip has been verified against the supposed 'critical task' of getting on its feet. As can be seen in appendix appendix D the motor torques stay well within specification, however it must be noted that this simulation was executed with extremely low, and purely static, joint-friction. This outlines the need for bushings and bearings on tulip to be carefully mounted such that the friction is as low as possible, with the exception of the passively actuated rotation around x in the ankle, where friction in the bushings might present a beneficial factor for the proposed walking gait.

As a model for a passive dynamic walker, a model for the simplest walker has been presented, from which design criteria for a periodic gait have been derived. Next to this the concept of the ZMP, which is widely applied in dynamically stable walkers, was discussed and expressed in generalized coordinates, twists and wrenches and an expression for the location of the ZMP has been formulated.
For the robot Tulip, the parameters for a periodic gait have been derived, based upon the compass walker model. Considering the extra dynamics of the actual robot, with respect to the compass walker, to be disturbances upon the model, the idealized ZMP trajectory

provides, together with the computed or measured actual ZMP of the robot can be used to form a control loop, controlling the gait during the swing-phase. Together with a simple sideways foot-placement strategy, it has been shown that the equations are solvable for robot Tulip and that the resulting computed torques are within the reasonable operating range.

Simulations to confirm the postulated theory have not been completed as of yet, due to implementation issues. The efficient calculation of the ZMP in 20-sim has proven to be less straightforward than expected. Practical implementation of the controller in the real robot faces similar computational problems. However, implementing a force and torque sensor for all 6 dimensions would deliver a very computationally inexpensive expression for the actual ZMP.

Simulations of the new walking gait have not been established, due to implementation difficulties. The efficient calculations, as shown, requires the knowledge of the twists and wrenches of all bodies in the kinematic chain, which gives rise to $6x8 = 48$ explicit differentiations. Implementing a F/T sensor around the ankle of the robot has proven equally challenging and a good solution to this problem has to be found.

It has however been shown that the concept of the ZMP indeed can be a tool in the design of passive dynamics gaits for complex robots. The fact that this is based upon a simple model of an inverse pendulum would reduce the amount of work and time spent on obtaining experimental data and parameter settings for periodic gaits. It is not suggested here that each ZMP stabilised gait will necessarily be an energetically optimum one. However, the necessity of different walking speeds and variations in stepsize resulting from for example a kicking task, would force the walker to execute energetically less favourable gaits. In such situations the ZMP control will find its optimum use.

The research on the walking behaviour of Tulip is at this stage not finished and will require some special attention. This project has resulted in a well documented model that can serve, together with this report, as a starting point for further work on Tulip and eventually other UT-humanoids. However, the study into the 3D walking behaviour has not been as far-stretching as was expected and aimed for. An attempt has been made to bridge the gap between the reasonably large region of stability for a 3D compass walker and the complex problem of a 3D 14-DOF walker, but especially in this area more focus is needed.

Bibliography

[1] M. Arbulu and C. Balaguer, *Real-time gait planning for rh-1 humanoid robot using local axis gait algorithm*, IEEE-RAS International Conference on Humanoid Robots (2007).

[2] Controllab Products B.V., *20-sim version 4.0*, http://www.20sim.com/, 2007.

[3] Steve Collins and Andy Ruina, *A bipedal walking robot with efficient and human-like gait*, IEEE International Conference on Robotics and Automation (2005), 1983–1988.

[4] Sony Corporation, *Worlds first running humanoid robot*, http://www.sony.net/SonyInfo/News/Press_Archive/200312/03-060E/.

[5] Edwin Dertien, *Realisation of an energy-efficient walking robot*, Master's thesis, University of Twente, June 2005.

[6] K. Kaneko et al., *Hrp-2*, Proceedings of the 2004 IEEE ICRA (2004).

[7] Michel Franken, *Ankle actuation for planar bipedal robots*, Master's thesis, University of Twente, April 2007.

[8] Bill Gates, *A robot in every home*, Scientific American (2007).

[9] Honda, *Asimo*, http://www.honda.com/asimo/.

[10] K.H. Hunt and F.R. Crossley, *Coefficients of restitution interpreted as damping in vibroimpact*, Applied Mechanics (1975), 440–445.

[11] Tad McGeer, *Passive dynamic walking*, International Journal of Robotics Research **9** (1990), no. 2, 62–82.

[12] Paul Mickle, *1961: A peep into the automated future*, http://www.capitalcentury.com/1961.html.

[13] S. Stramigioli and H. Bruyninckx, *Geometry and screw theory for robotics*, Tutorial during ICRA 2001 (2001).

[14] Gijs van Oort, *Strategies for stabilizing a 3d dynamically walking robot*, Master's thesis, University of Twente, june 2005.

[15] Eddy Veltman, *Compliant contact and ankle actuation of bipedal robots*, Master's thesis, University of Twente, May 2006.

[16] Martijn Wisse, *Essentials of dynamic walking, analysis and design of two-legged robots*, Ph.D. thesis, Technische Universiteit Delft, 2004.

[17] Yanzhen Xie, *Dynamic effects of an upper body on a 2d bipedal robot*, Master's thesis, University of Twente, June 2006.

Appendix A

Robot layout Tulip

3TU: Dutch robotics team

Tulip

10-01-2008

Body name

COM (x,y,z) [m]
mass [kg]
inertia (I_{xx}, I_{yy}, I_{zz})
*10e-3 [kg*m^2]

Joint name,, angle

Maxon motor-type
Maxon GP32C gear-ratio
position [x,y,z] [m]
stroke [deg]

Right arm
(0.012, -0.207, 0.584)
0.2
(5.53, 4.40, 0.48)

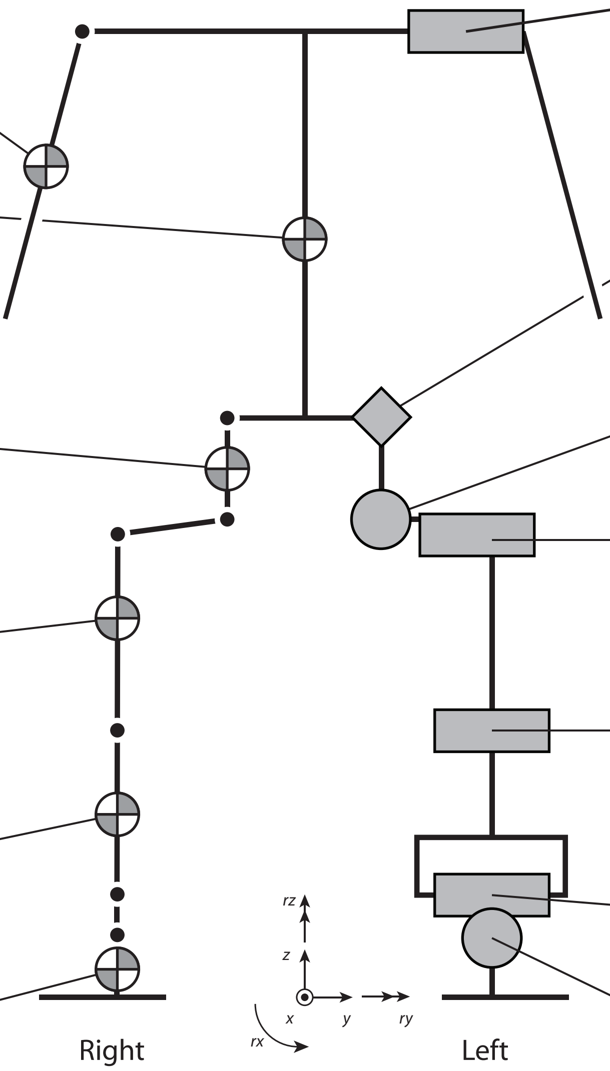
Torso
(0.000, 0.000, 0.675)
5
(13.34, 13.73, 2.86)

Right Hip
(0.000, 0.000, 0.675)
0.8
...

Right upperleg
(0.A, -0.157, 0.375)
1.75
(6.28, 7.28, 1.53)

Right lowerleg
(0.000, -0.157, 0.125)
0.25
(1.850, 1.820, 0.070)

Left foot
(0.030, -0.187, 0.005)
0.3
(0.083, 0.500, 0.600)



S_L: q6
RE 25, 20W
411 : 1
(0.000, 0.150, 0.800)
360

Hz_L: q5
RE 25, 20W
86 : 1
(0.000, 0.125, 0.563)
360

Hx_L: q4
RE 30, 60W
86 : 1
(0.000, 0.125, 0.538)
-30 : 90

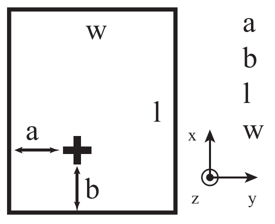
Hy_L: q3
RE 30, 60W
111 : 1
(0.000, 0.137, 0.500)
-160 : 90

K_L: q2
RE 60, 60W
111 : 1
(0.000, 0.157, 0.250)
0 : 150

Ay_L: q1
RE 60, 60W
68 : 1
(0.000, 0.157, 0.035)
-60 : 60

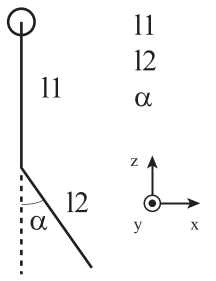
Ax_L: q13
torsional spring
(0.000, 0.157, 0.025)
-30 : 30

Footdesign (right foot)

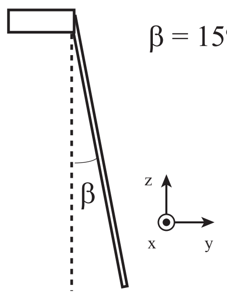


a	= 0.06 [m]
b	= 0.05
l	= 0.18
w	= 0.13

Armdesign:



l1	= 0.24 [m]
l2	= 0.21
α	= 20^o

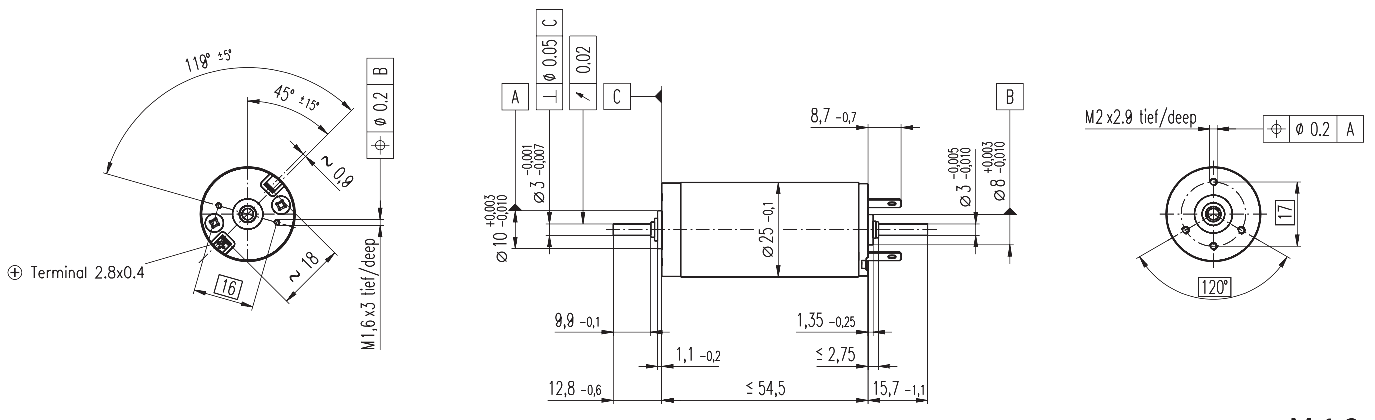


β = 15^o

Appendix B

Motor Specifications

RE 25 ∅25 mm, Graphite Brushes, 20 Watt



M 1:2

🟥	Stock program
☐	Standard program
▨	Special program (on request)

Order Number

according to dimensional drawing		118749	118750	118751	118752	118753	118754	118755	118756	118757
shaft length 15.7 shortened to 4 mm		302002	302003	302004	302005	302006	302007	302001	302008	302009
Motor Data										
Values at nominal voltage										
1 Nominal voltage	V	9.0	15.0	18.0	24.0	30.0	42.0	48.0	48.0	48.0
2 No load speed	rpm	10000	9650	10200	9550	9860	11100	10300	8230	5050
3 No load current	mA	110	60.7	53.9	36.9	30.5	25.2	20.1	15.2	8.51
4 Nominal speed	rpm	8980	8470	8890	8360	8680	9950	9190	7070	3870
5 Nominal torque (max. continuous torque)	mNm	11.1	20.6	23.1	26.7	27.2	27.6	28.4	29.4	30.8
6 Nominal current (max. continuous current)	A	1.50	1.50	1.47	1.17	0.983	0.799	0.667	0.548	0.352
7 Stall torque	mNm	244	237	233	257	263	299	280	222	136
8 Starting current	A	30.7	16.6	14.3	11.0	9.21	8.39	6.38	4.03	1.52
9 Max. efficiency	%	77	83	84	86	86	88	88	87	85
Characteristics										
10 Terminal resistance	Ω	0.293	0.902	1.26	2.19	3.26	5.00	7.53	11.9	31.6
11 Terminal inductance	mH	0.0275	0.0882	0.115	0.238	0.353	0.551	0.832	1.31	3.48
12 Torque constant	mNm / A	7.97	14.3	16.3	23.4	28.5	35.7	43.8	55.0	89.7
13 Speed constant	rpm / V	1200	669	585	407	335	268	218	173	106
14 Speed / torque gradient	rpm / mNm	44.1	42.3	45.3	38.1	38.2	37.5	37.4	37.6	37.5
15 Mechanical time constant	ms	5.36	4.58	4.49	4.28	4.20	4.13	4.11	4.10	4.09
16 Rotor inertia	gcm²	11.6	10.3	9.45	10.7	10.5	10.5	10.5	10.4	10.4

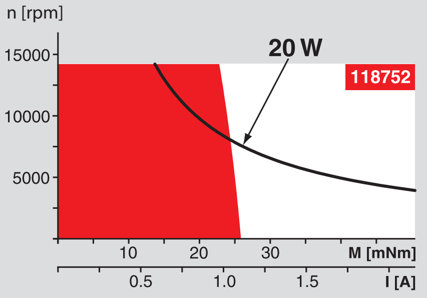
Specifications

	Thermal data	
17	Thermal resistance housing-ambient	14 K / W
18	Thermal resistance winding-housing	3.1 K / W
19	Thermal time constant winding	12.4 s
20	Thermal time constant motor	910 s
21	Ambient temperature	-20 ... +100°C
22	Max. permissible winding temperature	+125°C
	Mechanical data (ball bearings)	
23	Max. permissible speed	14000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	3.2 N
27	Max. force for press fits (static)	64 N
	(static, shaft supported)	270 N
28	Max. radial loading, 5 mm from flange	16 N
	Other specifications	
29	Number of pole pairs	1
30	Number of commutator segments	11
31	Weight of motor	130 g

Values listed in the table are nominal.
Explanation of the figures on page 47.

Option
Preloaded ball bearings

Operating Range



Comments

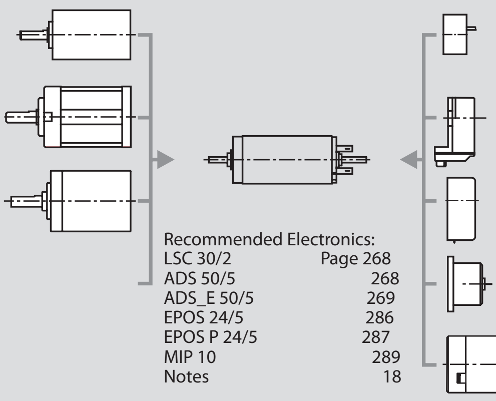
🟥 Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

☐ Short term operation
The motor may be briefly overloaded (recurring).

— Assigned power rating

maxon Modular System

Overview on page 16 - 21



Planetary Gearhead
∅26 mm
0.5 - 2.0 Nm
Page 226

Planetary Gearhead
∅32 mm
0.4 - 2.0 Nm
Page 228

Planetary Gearhead
∅32 mm
0.75 - 6.0 Nm
Page 229 / 231

Recommended Electronics:

LSC 30/2	Page 268
ADS 50/5	268
ADS_E 50/5	269
EPOS 24/5	286
EPOS P 24/5	287
MIP 10	289
Notes	18

Encoder MR
128 - 1000 CPT,
3 channels
Page 250

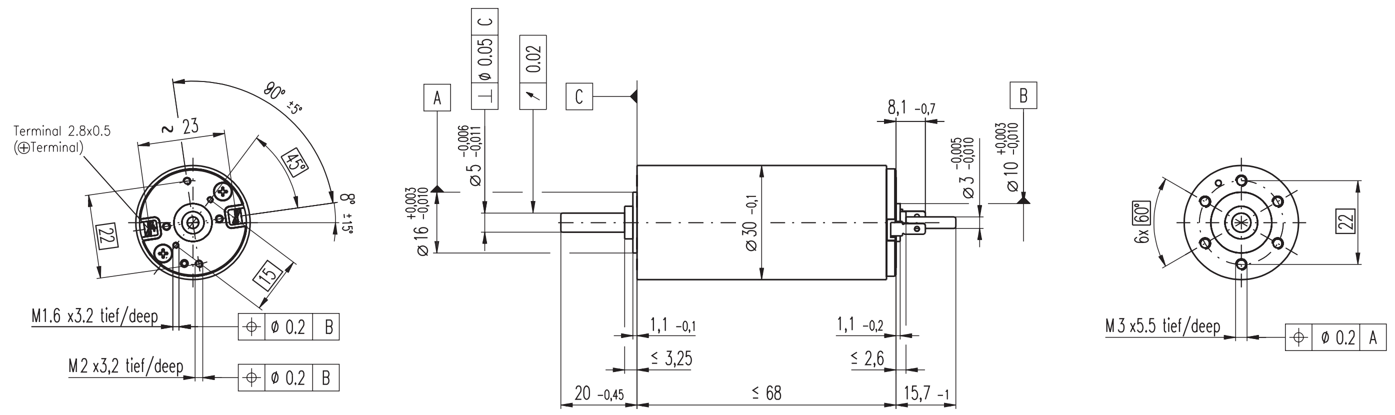
Encoder Enc
22 mm
100 CPT, 2 channels
Page 252

Encoder HED_ 5540
500 CPT,
3 channels
Page 254 / 256

DC-Tacho DCT
∅22 mm,
0.52 V
Page 263

Brake AB 28
∅28 mm
24 VDC, 0.4 Nm
Page 300

RE 30 ∅30 mm, Graphite Brushes, 60 Watt



M 1:2

🟥	Stock program
☐	Standard program
▨	Special program (on request)

Order Number								
according to dimensional drawing		310005	310006	310007	310008	310009		
shaft length 15.7 shortened to 8.7 mm		268193	268213	268214	268215	268216		
Motor Data								
Values at nominal voltage								
1 Nominal voltage	V	12.0	18.0	24.0	36.0	48.0		
2 No load speed	rpm	8170	8590	8810	8590	8490		
3 No load current	mA	300	212	164	106	78.5		
4 Nominal speed	rpm	7630	7900	8050	7810	7750		
5 Nominal torque (max. continuous torque)	mNm	51.7	75.5	85.0	83.4	88.2		
6 Nominal current (max. continuous current)	A	4.00	4.00	3.44	2.20	1.72		
7 Stall torque	mNm	844	991	1020	936	1020		
8 Starting current	A	60.5	49.8	39.3	23.5	19.0		
9 Max. efficiency	%	86	87	87	87	88		
Characteristics								
10 Terminal resistance	Ω	0.198	0.362	0.611	1.53	2.52		
11 Terminal inductance	mH	0.0345	0.0703	0.119	0.281	0.513		
12 Torque constant	mNm / A	13.9	19.9	25.9	39.8	53.8		
13 Speed constant	rpm / V	685	479	369	240	178		
14 Speed / torque gradient	rpm / mNm	9.74	8.71	8.69	9.22	8.33		
15 Mechanical time constant	ms	3.42	3.25	3.03	3.17	3.01		
16 Rotor inertia	gcm²	33.5	35.7	33.3	32.9	34.5		

Specifications

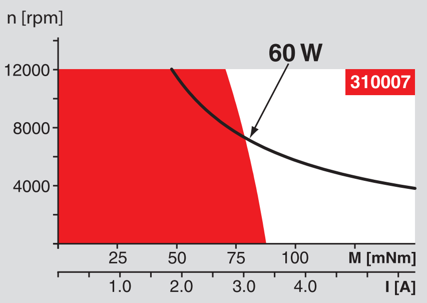
	Thermal data	
17	Thermal resistance housing-ambient	6.0 K / W
18	Thermal resistance winding-housing	1.7 K / W
19	Thermal time constant winding	16.2 s
20	Thermal time constant motor	714 s
21	Ambient temperature	-20 ... +100°C
22	Max. permissible winding temperature	+125°C
	Mechanical data (ball bearings)	
23	Max. permissible speed	12000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	5.6 N
27	Max. force for press fits (static)	110 N
	(static, shaft supported)	1200 N
28	Max. radial loading, 5 mm from flange	28 N
	Other specifications	
29	Number of pole pairs	1
30	Number of commutator segments	13
31	Weight of motor	238 g

Values listed in the table are nominal.
Explanation of the figures on page 47.

⚠ Tolerances may vary from the standard specification.

Option
Preloaded ball bearings

Operating Range



Comments

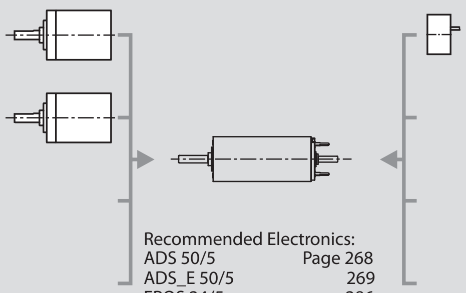
🟥	Continuous operation In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient. = Thermal limit.
☐	Short term operation The motor may be briefly overloaded (recurring).
—	Assigned power rating

maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅32 mm
0.75 - 4.5 Nm
Page 230

Planetary Gearhead
∅32 mm
1.0 - 6.0 Nm
Page 231



Encoder MR
256 - 1024 CPT
3 channels
Page 251

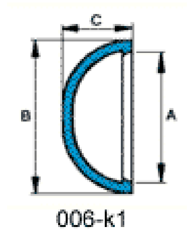
Recommended Electronics:

ADS 50/5	Page 268
ADS_E 50/5	269
EPOS 24/5	286
EPOS P 24/5	287
MIP 50	289
Notes	18

Appendix C

Compliant pucks for the feet

006-k1 - Cover cap



Prices in Euro

Dimensions in mm [✓ ●] Dimensions in inches ○

Article number	A	B	C	D	E	F	Gram	Price	Per	Price	Per	Price	Per
006 1082 599 01	18.8	20.6	10.2				0.7	40.40	1000	28.18	500		
covercaps type k1 --- Colour: black --- Material: Nylon-6 (PA-6)													

Let me try a sample first
I would like to order this part now
Previous size **Next size** **Print this page**

Colour:	**Colour number:**	
black	599	**XXX XXXX XXXXX**

Colour description	: jet black	■
Matches	: Reasonably matches RAL colour 9005	

Featured colours reserved. Due to the screen differences in colour may occur.

Material:	**Material number:**	
Nylon-6 (PA-6)	01	**XXX XXXX XXXXX**

General information
A strong, tough and durable material. Suitable for connecting elements and other technical components. Thanks to self lubricant properties ideal for slide bearings. Always has to acclimatize for a few days after injection moulding, taking approximately 3% moisture to obtain its normal strenght. Operational temperature can go up to 100-120°C temporarily for non-critical parts. Many nylons are self extinguishing.

Physical information

Property	**Value**	**DIN**
Relative density (gr/cm^3)	1.14	--
Tensile strength (MN/m^2)	55	53455
Elongation at break (%)	250	53455
Tensile modulus (MN/m^2)	950	53457
Notched impact strength (kJ/m^2)	35	53453
Ball indentation (MN/m^2)	82	53456
Application temperature (max ºC)	120	--
Volume resistivity (Ohm.cm)	10^15	53482
Dissapation factor tan. (10^3 Hz)	0.2	53483
Dielectric strength (MV/m)	35	53481
Flammability (UL94>1.6mm)	V2	--
Coefficient of friction (on steel) (--)	0.3	--

Chemical resistance

Resistant to	
Petrol	A
Benzene	A
Mineral oils	A
Vegetable oils	A
Weak alkalis	A
Strong alkalis	B
Weak acids	B
Strong acids	C

A = Good
B = Doubtful

Appendix D

Prone recovery results

In this appendix the results are given in more detail, represented as the setpoints, the applied motor torques and the used power versus the simulated time.

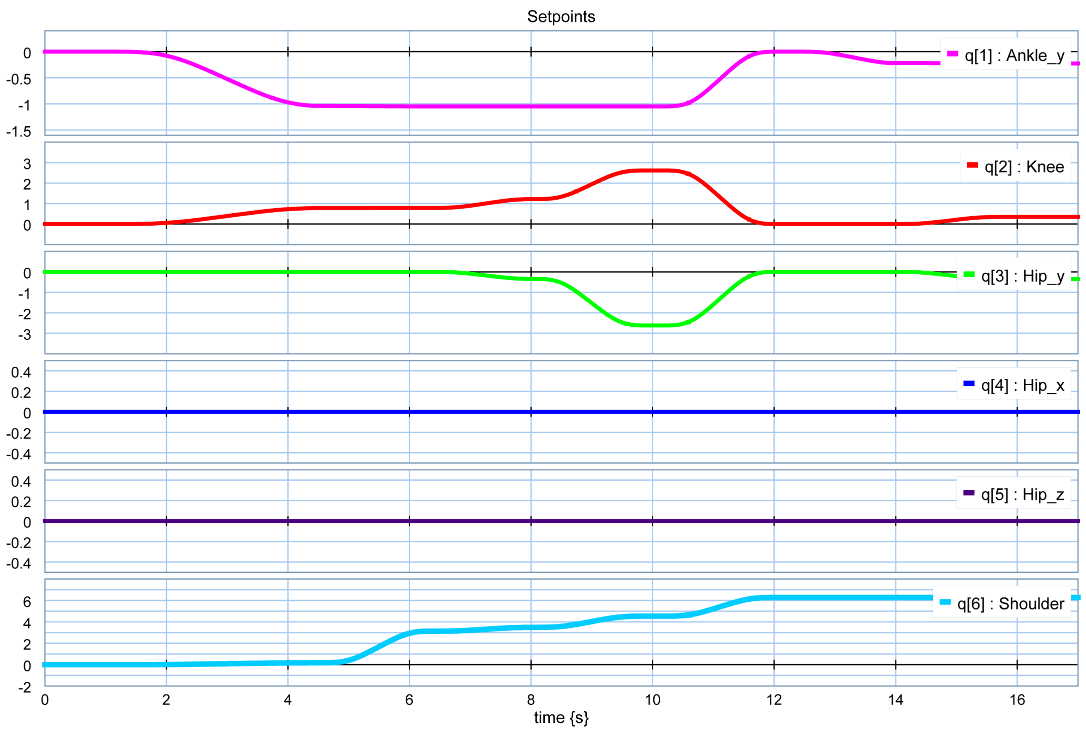


Figure D.1: Joint motion profiles, left limbs

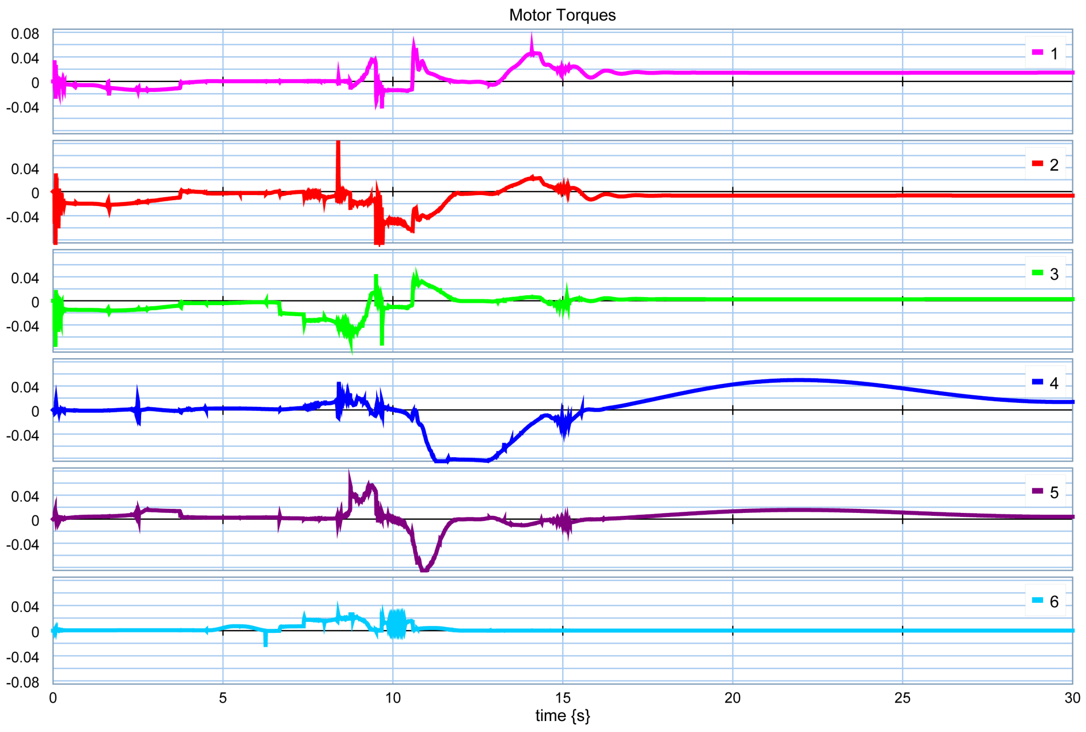


Figure D.2: Joint motion profiles, left limbs

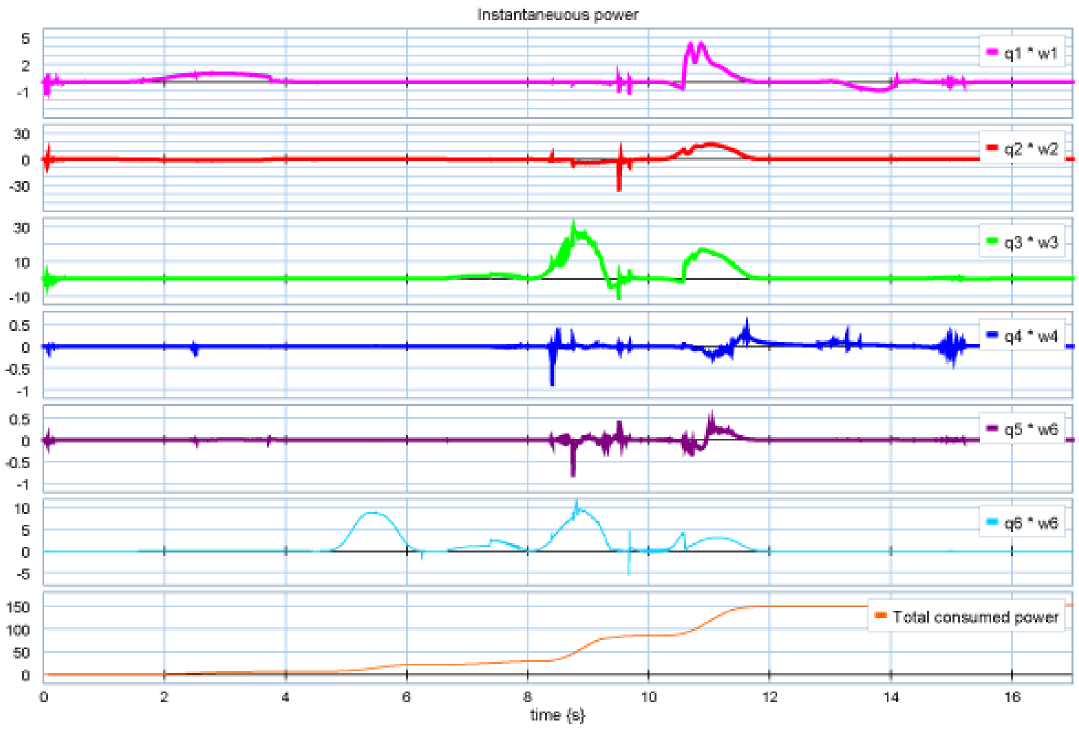


Figure D.3: Joint motion profiles, left limbs

Appendix E

20-sim models

- CP.emx: Contact point
- CP.emx: Contact sphere
- CP.emx: Contact ellipsoid
- CF.emx
- Endstops.emx
- COP.emx
- Setpoints.emx

E.1 Contact points

E.1.1 Contact point

```
/**Daemen 20071009: Generate a contact point for body b
  *
  * inputs:
  *    real Hb_0[4,4]   \\ H-matrix of body w.r.t. psi_0
  * outputs:
  *    real Hp_0[4,4]   \\ H-matrix of point P w.r.t. world-frame
  *    real Hp_b[4,4]   \\ H-matrix of point P w.r.t. body-frame
  *
  **/

variables
  real p_0[3];

equations

  p_0 = Hb_0[1:3,4];

  // Express the position of p in world coords as an H-matrix.
  // orientation of the frame is aligned to world orientation.

  Hp_0 = [ 1, 0, 0, p_0[1];
           0, 1, 0, p_0[2];
           0, 0, 1, p_0[3];
           0, 0, 0, 1];

  Hp_b = inverseH(Hb_0) * Hp_0;
```

E.1.2 Contact sphere

```
/**Daemen 20071009: Generate a contact point for body b
  *
  * inputs:
  *    real Hb_0[4,4]   \\ H-matrix of body w.r.t. psi_0
  * outputs:
  *    real Hp_0[4,4]   \\ H-matrix of point P w.r.t. world-frame
  *    real Hp_b[4,4]   \\ H-matrix of point P w.r.t. body-frame
  *
  **/
```

```
variables
  real p_0[3];

parameters
  real r;              \\ radius of the sphere

equations

  p_0 = Hb_0[1:3,4];

  // Express the position of p in world coords as an H-matrix.
  // orientation of the frame is aligned to world orientation.

  Hp_0 = [ 1, 0, 0, p_0[1];
           0, 1, 0, p_0[2];
           0, 0, 1, p_0[3] - r;
           0, 0, 0, 1];

  Hp_b = inverseH(Hb_0) * Hp_0;
```

E.1.3 Contact ellipsoid

```
/**Daemen 20071009:
  * Calculate the lowest point of an ellipse,
  *   with respect to the ground
  *
  * inputs:
  *   real Hb_0[4,4]   || H-matrix of body w.r.t. psi_0
  * outputs:
  *   real Hp_0[4,4]   || H-matrix of point P w.r.t. world-frame
  *   real Hp_b[4,4]   || H-matrix of point P w.r.t. body-frame
  *
  **/

variables
  real Q_sphere[4,4];
  real p_b[4];
  real xAxis[4], yAxis[4], z_n3[3], z_n4[4];
  real p_c[3];

parameters
  real rx = 0.03;
  real ry = 0.125;
  real rz = 0.1185;
equations

  // transformation ellipse to circle

  Q_sphere = [1/rx, 0, 0, 0; 0, 1/ry, 0, 0; 0,0,1/rz,0;0,0,0,1];

  // the 0 as the last element of the axis is because we
  // are only interested in rotation.
  // position could have been included, from H, but this
  // is faster:

  xAxis = Q_sphere * inverseH(Hb_0) * [1; 0; 0; 0];
  yAxis = Q_sphere * inverseH(Hb_0) * [0; 1; 0; 0];

  // calculate lowest point:

  z_n3 = cross(xAxis[1:3], yAxis[1:3]);
  z_n4 = [-z_n3[1]; -z_n3[2]; -z_n3[3]; norm(z_n3)]/norm(z_n3);
  p_b = inverse(Q_sphere) * (z_n4);

  //convert to contact frame

  p_c = Hb_0[1:3,1:4]*p_b;

  // Now p_c is the point of the body (or ellipsoid) that has the
  // minimum z-coordinate.

  Hp_0 = [ 1, 0, 0, p_c[1];
         0, 1, 0, p_c[2];
         0, 0, 1, p_c[3];
         0, 0, 0, 1];
  Hp_b = inverseH(Hp_0) * Hs_0;
```

E.1.4 Contact convex foot

```
/**Daemen 20071009:
  * Calculate the lowest point of a almost flat,
  *   convex foot with respect to the ground
  *
  * inputs:
  *   real Hb_0[4,4]   \\ H-matrix of body w.r.t. psi_0
  * outputs:
```

```
  *    real Hp_0[4,4]   \\ H-matrix p in world-frame
  *    real Hp_b[4,4]   \\ H-matrix p in. body-frame
  *
  * foot reference frame is at the center of the foot,
  *    on the ground.
  **/

parameters
  real l = 0.18;  // circumferential Length of the foot
  real w = 0.13;  // width of the foot
  real Ry = 10.0; // spheroid radius in y-direction.
  real Rx = 10.0; // spheroid radius in x-direction.

variables
  real lambda;          // The length of the foot in radians.
  real p_accent[3];
  real p_0[4];          // P, in world coordinates.
  real p_b[4];          // P, in local (body) coordinates.
  real xf_0[3];         // the foot's x-axis, in world coords.
  real yf_0[3];         // the foot's y-axis, in world coords.
  real zf_0[3];         // the foot's y-axis, in world coords.
  real theta_accent, theta;
  real gamma;
  real y_deviation_accent, y_deviation;

constants
  // Direction of the world's z-axis, expressed in world coords.
  real zw_0[3] = [0;0;1];

initialequations
  lambda = l/Rx;

equations
  xf_0 = Hb_0[1:3,1];
  yf_0 = Hb_0[1:3,2];
  zf_0 = Hb_0[1:3,3]; //check for foot inversion

  if (zf_0[3]<=0) then // Foot not upright
    // closest point is one of the corners.
    theta = limit( (Rx*xf_0[3]), -lambda/2, lambda/2);
    y_deviation = - limit( Rx * yf_0[3], -w/2, w/2);
  else
    p_accent = cross(yf_0, (cross(yf_0, zw_0)));
    theta_accent = -arcsin(  inner(p_accent, xf_0) / norm(p_accent));
    theta = limit(theta_accent,-lambda/2, lambda/2);

    // For sideways calculation of the contact point
    gamma = inner(yf_0, zw_0);
    y_deviation_accent = -Ry * gamma;
    y_deviation = limit(y_deviation_accent, -w/2, w/2);
  end;
  p_b = [ -Rx*sin(theta); y_deviation; Rx-Rx*cos(theta); 1];
  p_0 = Hb_0 * p_b;

  // Write P in world coords. Orientation aligned to world
  Hp_0 = [ 1, 0, 0, p_0[1];
         0, 1, 0, p_0[2];
         0, 0, 1, p_0[3];
         0, 0, 0, 1];
  Hp_b = inverseH(Hb_0) * Hp_0;
```

E.2 CF : calculate contact wrench

```
/**Daemen 20071009:
  * Calculate the ground reaction force, including friction
  *
  * inputs:
  *   real Hp_0[4,4]     H-matrix of p in world frame
  *   real Hp_b[4,4]     H-matrix of p in body frame
  *   boolean on         switch floor on/off
  * outputs:
  *   power P            6 DOF power bond
  *   boolean contact    indicates contact
  *
  * foot reference frame is at the center of the foot,
  *   on the ground.
  **/

variables
  boolean contact;
  boolean contactevent;
  boolean endcontactevent;
  real err[6];
  real Tp[6];    // Body twist expressed in point's coords
  real Wp[6];    // Body wrench expressed in point's coords
  real Fn;       // Normal force = Wp[6]
  real omega_x,omega_y,omega_z;
  real v_tr;     // the speed of point p over the plane
  real v_tr_temp;
  real Fr;       // rotational friction force;
  real Ft;       // translational friction force;
  real T_x;      //rolling friction
  real T_y;      //rolling friction
  real Zworld_body[4],Yworld_body[4];
  boolean FloorIsOn;
  real f_tot, t_tot;
parameters
  boolean global friction_rot\active;
  boolean friction_trans\active;
  real global hc_model\Kp, hc_model\Kd;
  real global friction_rot\mu_st;
  real global friction_rot\mu_c;
  real global friction_rot\mu_v;
  real global friction_rot\slope;
  real global friction_rot\omegaz_st;
  real global friction_trans\mu_st;
  real global friction_trans\mu_c;
  real global friction_trans\mu_v;
  real global friction_trans\slope;
  real global friction_trans\v_st;

constants
  real epsilon = 0.001;

initialequations
  contact=false;
  FloorIsOn = false;
  T_x = 0;
  T_y = 0;

equations
  // calculate the P in point coords:
  Tp = Adjoint(Hp_f) * P.f;

  contactevent=eventdown(Hp_0[3,4]);

  err = [0; 0; 0; 0; 0; Hp_0[3,4]];

  if (contactevent) then contact = true; end;
```

```
endcontactevent = eventup(Hp_0[3,4]);
if (endcontactevent) then contact=false; end;

// We may turn on floor, but not when foot is below floor
FloorIsOn = (On and (not FloorIsOn) and (not contact));

// We use the hunt-crossley model for the compliant contact
// F = -kp * x^n + -kd * x^n * xdot  , where we have n=1:

// Friction component for rotation around the local Z-axis.
// typical mu_st = 0.04; mu_c = 0.02; mu_v = 0.002.
//
// Translational: also a rough guess based on human feet.
// mu_st = 0.5; mu_c = 0.25; mu_v = 0.025.

omega_x = Tp[1];
omega_y = Tp[2];
omega_z = Tp[3];

v_tr_temp = norm(Tp[4:5]);
v_tr = if(abs(v_tr_temp)>epsilon) then
          v_tr_temp
      else
          sign(0.5+sign(v_tr_temp))*epsilon
      end;
  // prevent division by zero

if (FloorIsOn==1 and contact) then
  Fn = -hc_model\Kp*err[6] - hc_model\Kd* -err[6] *Tp[6];
  T_x = -0.1*omega_x;
  T_y = -0.2*omega_y;
  Fr = if (friction_rot\active)  then
    -Fn * (
      ( friction_rot\mu_c + (friction_rot\mu_st
        *abs(tanh( friction_rot\slope*omega_z ))
        - friction_rot\mu_c)
        * exp( -((omega_z / friction_rot\omegaz_st)^2 ))
      ) *  sign(omega_z) + friction_rot\mu_v * omega_z)
    else
      0
    end;
Ft = if (friction_trans\active) then
    -Fn * (
      ( friction_trans\mu_c + (friction_trans\mu_st
            *abs(tanh( friction_trans\slope*v_tr ))
            - friction_trans\mu_c)
            * exp( -((v_tr / friction_trans\v_st)^2 ))
          ) * sign(v_tr) + friction_trans\mu_v * v_tr
        )
      else
        0
      end;
    else
      Fn = 0;
      Fr = 0;
      Ft = 0;
    end;


    Wp =   [ T_x;
           T_y;
           Fr; // rotational friction
           Ft * (Tp[4]/v_tr);
           Ft * (Tp[5]/v_tr);
           Fn
         ];
    f_tot = norm(Wp[4:6);
    t_tot = norm(Wp[1:3]);
    P.e = transpose(Adjoint((Hp_f))) * Wp;

    Contact = FloorIsOn and contact;
```

E.3 Endstops

```
/**Daemen 20070614:
  *   14 DOF endstops
  *
  * inputs:
  *   real q[14]        || joint angles
  * outputs:
  *   power P           || Power bond
  *
  **/

variables
  real q_range[14,2], err[14,2];
  integer i, dof;

parameters
  real q_range_deg[14,2];   \\ joint motion range
  real stop_dynamics[14,2]; \\ end stop dynamics, [K,D]

initialequations
  err = [0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0;0,0];
  q_range = ( q_range_deg / 360 ) * 2 * pi;
  dof = rows(q_range);

code
  for i = 1 to dof by 1 do
    err[i,1] =
        if (stop_dynamics[i,1]<>0) then
          q[i] - limit(q[i], q_range[i,1],q_range[i,2])
        else
          0
        end;
    err[i,2] =
        if (err[i,1] <>0) then
          P.f[i] * err[i,1] //Hunt-Crossley
        else
          0
        end;
  end;

  P.e  = stop_dynamics .* err * [-1;-1];
```

E.4 PID

```
/**Daemen 20070729:
  * 14 DOF parallel PID, tame differential part
  *
  * inputs:
  *   real q[14]        joint angles
  *   real K[14]        joint angles
  *   real Kd[14]       joint angles
  *   real Ki[14]       joint angles
  *   real beta
  * outputs:
  *   output
  *
  **/

variables
  real error[14];
  real hidden uP[14],uI[14],uD[14],uDstate[14];
equations
  error = setpoint - q;
  uP = K .* error;
  uI = int (error .* Ki);
  uDstate = int (uD);
  uD = K .*error / beta  - uDstate .* ( Kd );
  output = (uP + uI + uD);
```

E.5 COP and contact

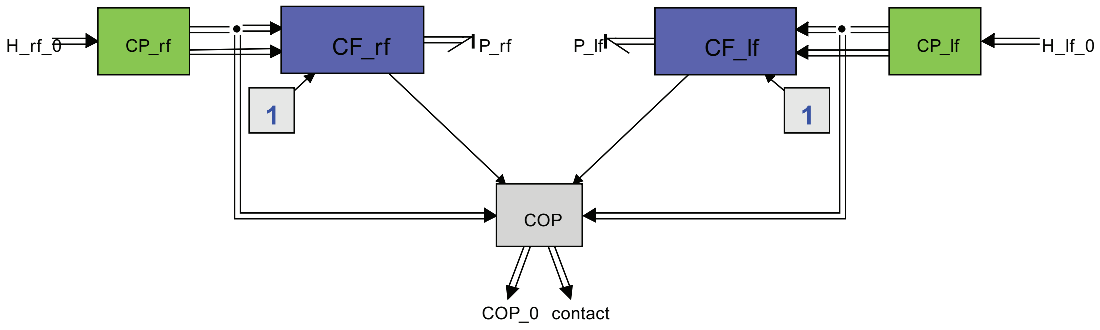


Figure E.1: Calculate COP of standing robot

```
/**Daemen 20070809:
  *   calculate COP and collect contact booleans
  *
  * inputs:
  *   real Hlf[4,4]    || left foot in world coords
  *   real Hrf[4,4]    || right foot in world coords
  *   real Fn_r, Fn_l  || normal forces, right and left
  * outputs:
  *   real COP[2,1]      COP planar coords in world coords
  *   real contact[2,1] contact booleans, [left, right]
  *
  **/

variables
```

```
  real cop_x, real cop_y;
  real Fn;
  real iscontact;
equations
  contact[1:2] = [(Fn_l>0); (Fn_r>0)];

  Fn = Fn_l + Fn_r;

  cop_x = Hrf_0[1,4] * Fn_r + Hlf_0[1,4] * Fn_l;
  cop_y = Hrf_0[2,4] * Fn_r + Hlf_0[2,4] * Fn_l;

  COP_0[1:2] =
      if (Fn>0) then
          [cop_x;cop_y] ./ Fn
      else
          [0;0]
      end;

  iscontact = if ( (Fn_l>0) or (Fn_r>0)) then 0.2 else 0.0 end;
```

E.6 Set-point generation

```
/**Daemen 20071023:
  * Generate setpoints according to statemachine
  *   implemented in daemen.dll
  *
  * inputs:
  *   q[14]          actual joint angles
  * outputs:
  *   q_set[14]      joint setpoints
  *
  * foot reference frame is at the center of the foot,
  *   on the ground.
  **/

parameters
  string filename = '../dll/daemen.dll';
  real epsilon = 0.03;
  real  targets[15,7] = [
   0, -60,  45,   0, 0, 0,  10, -60 , 45,   0, 0, 0,  10, 0, 0;
   1, -60,  45,   0, 0, 0, 180, -60,  45,   0, 0, 0, 180, 0, 0;
   2, -60,  70, -20, 0, 0, 200, -60,  70, -20, 0, 0, 200, 0, 0;
```

```
     3, -60, 150,-150, 0, 0, 260, -60, 150,-150, 0, 0, 260, 0, 0;
     4,   0,   0,   0, 0, 0, 360,   0,   0,   0, 0, 0, 360, 0, 0;
     5, -13,   0,   0, 0, 0, 360, -13,   0,   0, 0, 0, 360, 0, 0;
     6, -13,  20, -20, 0, 0, 360,   0,   0,   0, 0, 0, 360, 0, 0
    ];

variables
    real in_sp[2], y[14], in_state[16];
    real state[4], dummy;
    real target1[15],target2[15],target3[15],target4[15];
    real target5[15],target6[15],target7[15];

initialequations
    state = [0;0;0;0];

    //preload target postures
    target1 = targets[1:15,1];
    dummy = dll(filename,'load', target1);
    target2 = targets[1:15,2];
    dummy = dll(filename,'load', target2);
    target3 = targets[1:15,3];
    dummy = dll(filename,'load', target3);
    target4 = targets[1:15,4];
    dummy = dll(filename,'load', target4);
    target5 = targets[1:15,5];
    dummy = dll(filename,'load', target5);
    target6 = targets[1:15,6];
    dummy = dll(filename,'load', target6);
    target7 = targets[1:15,7];
    dummy = dll(filename,'load', target7);

code
    in_state[1:14] = (180/pi) * q;
    in_state[15:16] = [time; (180/pi)*epsilon];
    state = dll(filename,'stateMachine', in_state);

    in_sp = time;
      y = dll(filename,'setpoints',in_sp);

    q_set = (pi / 180) * y;
```

Appendix F

DLL C Code

```
#include <windows.h>
#include <vector>
#include <iostream>
#include <math.h>

#define DllExport __declspec( dllexport )

using namespace std;

extern "C" {

  double maxT[14];
  double inertia[14];
  double q_0[14];
  double targets[7][14];

  double q[14], t, t_start, t_mid, t_final, deltaT, setPoints[14];
  const double pi = M_PI;
  int state;

  DllExport int setpoints(double *inarr, int inputs,
                          double *q_out, int outputs, int major) {
    int n = 0;
    double dq = 0;

    t = inarr[0];

    //calculate outputs:
    for (n=0; n < 14; n++) {
      dq = targets[state][n]-q_0[n];
      if (t > t_final) {
        q_out[n] = targets[state][n];
      } else {
        if (t <= t_start) {
          q_out[n] = q_0[n];
        } else {
          if (t < t_mid ) {
            q_out[n] = q_0[n] + (4 / pow(deltaT,2))
                * dq * ( pow(deltaT/4/pi,2)
                  *(cos(4*pi*(t-t_start)/deltaT) - 1)
                + (t-t_start)*(t-t_start)/2);
          } else {
            q_out[n] = q_0[n] - (4 / pow(deltaT,2))
                * dq * ( pow(deltaT/4/pi,2)
                  *(cos(4*pi*(t-t_start)/deltaT) - 1 + 4*pow(pi,2))
                + pow(t-t_start,2)/2 - deltaT*(t-t_start));
          }
        }
      setPoints[n] = q_out[n];
      }
    }
    return 0; // return succesfull
  }

  DllExport int stateMachine(double *inarr, int inputs,
                          double *outarr, int outputs, int major) {
    int n;
    double error = 0.0;
    double maxdt = 0.0;
    double a_max = 0.0;
    double dq[14], dt;

    for (n=0; n<12; n++) {
      dq[n] = fabs(targets[state][n] - inarr[n]);
      error += dq[n];
    }

    if (error <= inarr[15] && inarr[14] > t_mid) {
      state += 1;

      for (n=0; n<14; n++) { // reset profile:
        q_0[n] = setPoints[n];
        dq[n] = fabs(targets[state][n] - q_0[n]);
        error += dq[n];
        a_max = 1; //maxT[n] / inertia[n];
        dt = sqrt( fabs(8*dq[n]) / a_max);
        if (dt > maxdt) maxdt = dt;
      }
      t_start = inarr[14];
      deltaT = 2;//maxdt;
      t_final = t_start + deltaT;
      t_mid = t_start + deltaT/2;
    }

    outarr[0] =  (double) state;
    outarr[1] =  error;
    outarr[2] = t_start;
    outarr[3] = deltaT;

    return 0;
  }

  DllExport int load(double *inarr, int inputs,
                     double *outarr, int outputs, int major) {
    for (n=1; n<15; n++) {
      targets[inarr[0]][n-1] = inarr[n];
    }
    outarr[0] = 1;
    return 0;
  }

  DllExport int Initialize()  {
    // do some initializations here.
    state = 0;
    q_0[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    t_start = 1;
    t_mid = 3;
    t_final = 5;
    deltaT = 4;
    return 0; // Indicate that the dll was initialized successfully.
  }

  DllExport int Terminate() {
    // do some cleaning here
    return 1; // Indicate that the dll was terminated successfully.
  }
}
```