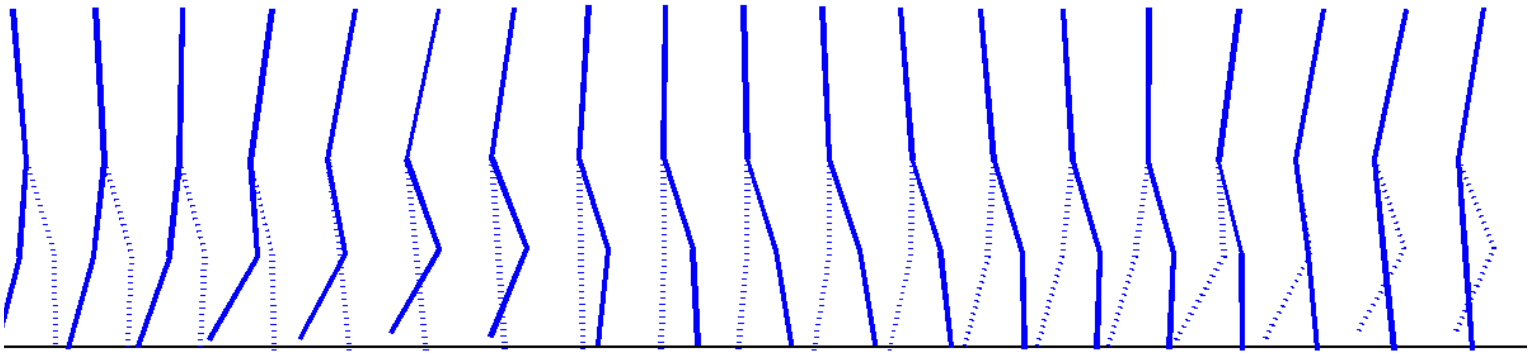
Helsinki University of Technology Control Engineering Laboratory

Espoo 2004 Report 138

SIMULATION TOOL OF A BIPED WALKING ROBOT MODEL

Olli Haavisto Heikki Hyötyniemi



TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D´HELSINKI

Helsinki University of Technology Control Engineering Laboratory

Espoo March 2004 Report 138

SIMULATION TOOL OF A BIPED WALKING ROBOT MODEL

Olli Haavisto Heikki Hyötyniemi

Abstract: In this report a simulation tool for a biped robot model is presented. The tool uses Matlab/Simulink environment and contains simulation blocks for the biped and two controllers. A graphical user interface for model simulation and animation is also included.

Keywords: biped, robot, model, walking, Matlab, Simulink, data-based modeling, PD control, clustered regression control

Helsinki University of Technology
Department of Automation and Systems Technology
Control Engineering Laboratory

Distribution:
Helsinki University of Technology
Control Engineering Laboratory
P.O. Box 5500
FIN-02015 HUT, Finland
Tel. +358-9-451 5201
Fax. +358-9-451 5208
E-mail: control.engineering@hut.fi
http://www.control.hut.fi/

ISBN 951-22-7018-8
ISSN 0356-0872

Picaset Oy
Helsinki 2004

Technical notes

The simulation tool was created in Windows XP using Matlab 6.5 and Simulink 5.0 (Matlab Release 13), and it works also at least in Matlab 6.5.1 and Simulink 5.1 (Release 13 with service pack 1).

In this document, Simulink model and block names are typed in *italic*, and names of the Matlab m-files, functions and variables in `truetype` font. In the Simulink models, blocks having a mask are drawn with a drop shadow. Additionally, connection lines with bigger dimension than one are thick and the dimension is show.

The simulation tool and this document are available on the Internet at

`http://www.control.hut.fi/publications/haavisto-2004/simulator/`.

Contents

Chapter 1

Introduction

This document describes the biped robot simulation tool for Matlab/Simulink that was developed during a master's thesis work [2] in the HUT Control Engineering Laboratory in 2003. The tool enables the simulation of the exact dynamics of a two-dimensional biped robot model on a walking surface. The simulation environment contains a simulation block for the biped model, PD controller block for controlling the model and data-based controller block which implements the so-called clustered regression control. A graphical user interface for simulation and biped motion animation is also included.

The biped structure modeled is a five-link, two-dimensional walking robot with a torso and knees but no ankles. The walking surface can be defined as a sequence of points connected with straight lines. The interaction between the biped and the ground is modeled using external forces acting on each leg tip when the leg touches the ground. This enables the use of one seven degrees of freedom dynamic model to describe the dynamics of the system in all situations. The ground contact forces are calculated by separate PD controllers.

The PD control method presented for controlling the gait is quite cumbersome, its main purpose being only to enable data collection from the walking biped system. Using this data, the clustered regression model structure is taught and applied to the biped control.

Chapter 2

Biped model structure

The biped model was chosen to be quite simple in order to make it possible to calculate the exact mathematical dynamic equations. The model can, however, describe the walking motion quite well and is therefore used as such or slightly modified also in many other researches (for example, see [1]).

2.1 Biped dynamics

The two-dimensional biped consists of five links which are connected with frictionless joints. The identical legs have knee joints between the shank and thigh parts, and one rigid body forms the torso. Figure 2.1(a) shows the model structure and variables used.

As the system can move freely in the x-y-plane and contains five links, it has seven degrees of freedom. The corresponding seven coordinates are selected according to Figure 2.1(a):

$$q = [x_0, y_0, \alpha, \beta_L, \beta_R, \gamma_L, \gamma_R]^T \tag{2.1}$$

The coordinates (x_0, y_0) fix the position of the center of mass of the torso, and the rest of the coordinates describe the joint angles. The link lengths are denoted as (l_0, l_1, l_2) and masses as (m_0, m_1, m_2). The centers of mass of the links are located at the distances (r_0, r_1, r_2) from the corresponding joints.

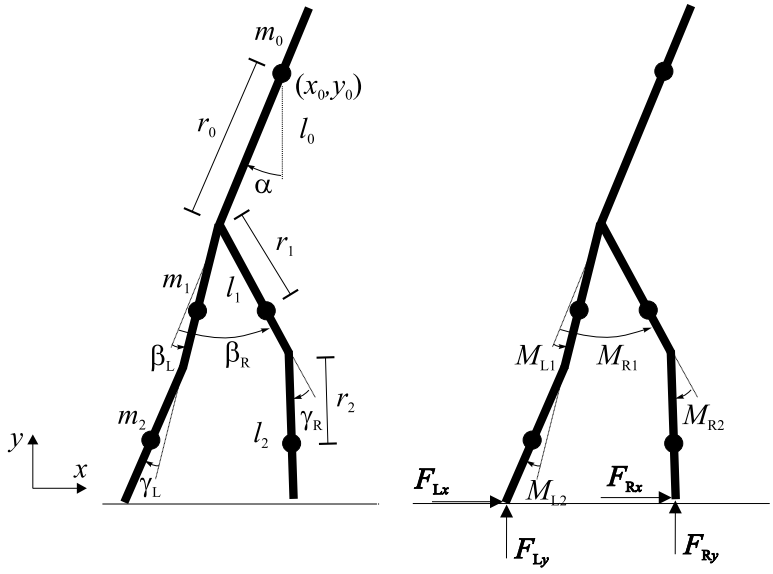
The model is actuated with four moments

$$M = [M_{L1}, M_{R1}, M_{L2}, M_{R2}]^T, \tag{2.2}$$

two of them acting between the torso and both thighs and two at the knee joints (Fig. 2.1(b)). The walking surface is modeled using external forces

$$F = [F_{Lx}, F_{Ly}, F_{Rx}, F_{Ry}]^T \tag{2.3}$$

that affect the both leg tips. When the leg should touch the ground, the corresponding forces are switched on to support the leg. As the leg rises, the forces are zeroed.



(a) (b)

Figure 2.1: Biped model coordinates and constants (a) and external forces (b).

Using Lagrangian mechanics, the dynamic equations for the biped system can be derived:

$$A(q)\ddot{q} = b(q, \dot{q}, M, F). \tag{2.4}$$

Here $A(q) \in \mathbb{R}^{7\times7}$ is the *inertia matrix* and $b(q, \dot{q}, M, F) \in \mathbb{R}^{7\times1}$ is a vector containing the right hand sides of the seven partial differential equations. The closed form formulas for both A and b are listed in Appendix A.

2.2 Ground contact forces

The ground surface is modeled as a set of x, y points that are connected with straight lines. Figure 2.2 shows a sketch of one leg in touch with the ground.

A new coordinate system (x', y') is now defined so that its origo is the next ground point to the negative x direction of the leg tip. The axis x' is aligned tangential to the ground surface and y' equals the surface normal direction as shown in Figure 2.2.

When the leg tip touches the ground at the point $(x'_0, 0)$, normal and tangential forces are applied to it. The normal force is calculated as an output of a PD controller, so that the situation is analogous to a spring-damper system:

$$F_n = -k_y y'_G - b_y \dot{y}'_G, \tag{2.5}$$

where y'_G is the current (negative) leg tip y' coordinate, k_y the ground normal elastic constant and b_y the normal damping ratio. Additionally the normal force is limited to positive values to prevent the leg sticking to the ground.

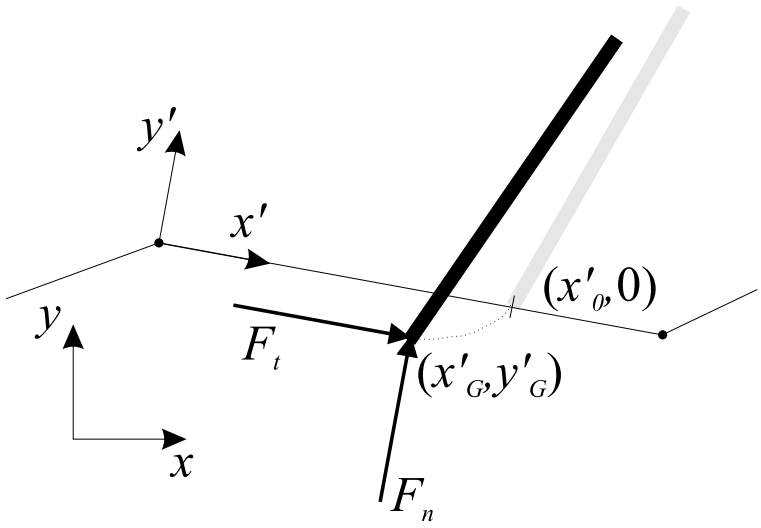


Figure 2.2: The leg tip touches the ground in point $(x'_0, 0)$ (grey) and penetrates it. The current position of the leg tip is (x'_G, y'_G) (black). Note that the penetration is exaggerated for clarity.

In the tangential direction the force F_t acting is caused by friction. The static friction force is determined using a PD controller similar to the normal force calculation. Now the nominal value, however, is the initial touching point x'_0:

$$F_t = -k_x(x'_G - x'_0) - b_x\dot{x}'_G, \tag{2.6}$$

where k_x and b_x are the ground tangential properties. If the required force exceeds the maximum static friction force

$$F_{t,max} = \mu_s F_n, \tag{2.7}$$

μ_s being the static friction coefficient, the leg starts to slide. In that case the tangential force is

$$F_t = \mu_k F_n, \tag{2.8}$$

where μ_k is the kinetic friction coefficient. The stored value of x'_0 is continuously set to the corresponding leg tip x'_G position during the sliding.

After the normal and tangential forces are computed, they need to be projected to the original coordinate system (x, y) to attain the dynamic model input forces (2.3).

2.3 Knee angle limiting

The knee angles are limited to a user-defined range. This limitation works quite similar to the ground contact normal force calculation. When the knee angle is exceeding the given maximum value or going under the minimum value, a PD controller is switched on. The controller output is added to the corresponding knee joint moment to prevent the joint rotating over (or under) the limit value. Typically one wants that the knee cannot rotate under the zero angle of γ_L (or γ_R), that is, bend to the "wrong" direction.

Chapter 3

Simulink blocks

The simulation tool consists of three main Simulink blocks found in the library model *bipedlib.mdl*. The blocks are the actual biped model (*Biped model* block) and two controllers (*PD control* and *Clustered regression control*). This chapter gives an overview of these blocks and their parameter definitions, but a more complete description of the internal structures of the blocks is given in Appendix B.

3.1 *Biped model* block

The *Biped model* block simulates the complete walking robot system. It includes biped dynamics, ground support force calculation and knee angle limiter. The parameters for the system can be defined on the parameter dialog box of the block. Figure 3.1 shows the *Biped model* Simulink block and its dialog box.

The parameters for the block define the robot dimensions, system initial state and properties of the knee limiter and walking surface (ground). Table 3.1 lists all the parameters, their descriptions and units. An example parameter definition file is presented in Appendix C.1. Note that most of the parameters are Matlab structure variables with specific fields.

The robot link properties and initial coordinate values are defined according to Figure 2.1(a). The `gcstate` field of the initial state structure contains the friction force calculation information for both (L, R) leg tips. The initial touching coordinate values are store as variables x'_{0L} and x'_{0R}. $sliding_L$ and $sliding_R$ binary flags define whether the leg is currently sliding (1) or not (0).

The ground surface points are collected to the ground properties variable field `ground` so that the first row contains the x-values in ascending order and the second row the corresponding y-values.

The system is simulated as a continuous process, but the input and output signals are discretized using zero order hold with the defined sample time.

The input for the block is the 4×1 moment vector (2.2) and the output the state of

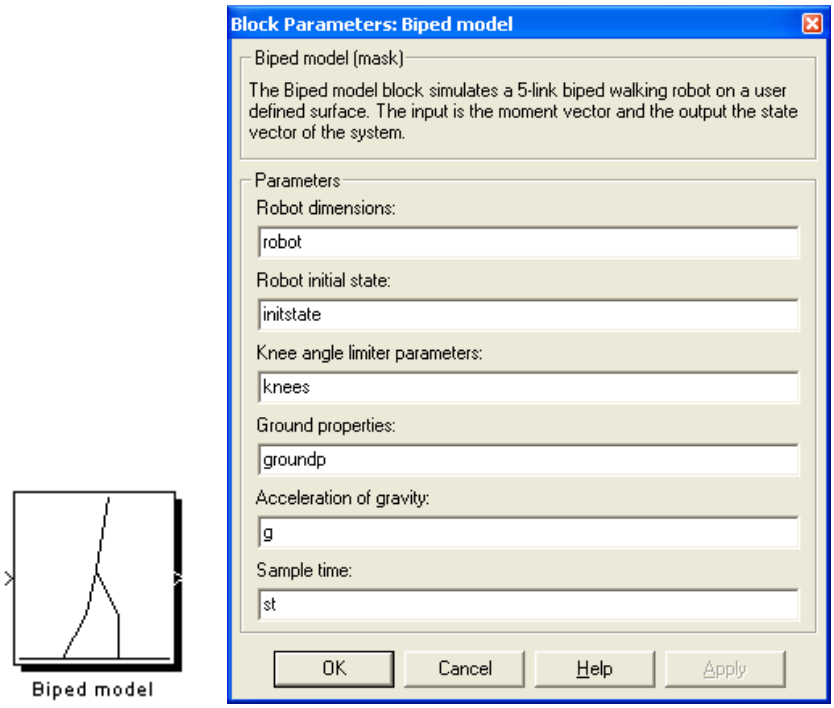


Figure 3.1: *Biped model* block and the parameter dialog box.

the system. The output signal contains the coordinate values q, the corresponding time derivatives $\dot{q}$ and the leg tip touch sensor values s_L, s_R. If the leg is touching the ground, the sensor value equals to 1, otherwise 0. So the total output signal is the 16×1 vector

$$[q^T, \dot{q}^T, s_L, s_R]^T. \tag{3.1}$$

The internal structure of the *Biped model* block is presented in Appendix B.1.

3.2 *PD control* block

In order to test the *Biped model* block and get the system walking, a discrete PD controller was developed. The control is based on four independent SISO controllers, which function according to reference signals. As the biped moves, the references are updated to maintain the gait.

The walking movement can be divided into two main phases: *double support phase* (DSP), in which both legs are touching the ground and *single support phase* (SSP), in which only one leg is touching the ground. As two consecutive steps are always symmetrical with respect to the left and right leg, it suffices to handle only the case where the left leg, for example, is the swinging leg and right leg the stance leg. Now one step consists of a DSP with the left leg as the rearmost leg and a SSP where the left leg swings. The next step can then be described changing the left and right leg.

The creation of the reference signals was designed so that the left leg is always the swinging

Table 3.1: Biped model block parameter definitions.

field	description	units
Robot dimensions:		
`l`	robot link lengths $[l_0, l_1, l_2]$	m
`r`	center of mass distances $[r_0, r_1, r_2]$	m
`m`	link masses $[m_0, m_1, m_2]$	kg
Robot initial state:		
`coordinates`	$[x_0, y_0, \alpha, \beta_L, \beta_R, \gamma_L, \gamma_R]$	m, (rad)
`speeds`	$[\dot{x_0}, \dot{y_0}, \dot{\alpha}, \dot{\beta_L}, \dot{\beta_R}, \dot{\gamma_L}, \dot{\gamma_R}]$	m/s, (rad)/s
`gcstate`	$[x'_{0L}, sliding_L, x'_{0R}, sliding_R]$	m
Knee angle limiter parameters:		
`kk`	elastic constant	Nm/(rad)
`bk`	damping ratio	Nms/(rad)
`min`	minimum angle for γ_L and γ_R	(rad)
`max`	maximum angle for γ_L and γ_R	(rad)
Ground properties:		
`ground`	surface points $\begin{bmatrix} x_1, x_2, \dots \\ y_1, y_2, \dots \end{bmatrix}$	m
`ky`	normal elastic constant	kg/s^2
`by`	normal damping ratio	kg/s
`kx`	tangential elastic constant	kg/s^2
`bx`	tangential damping ratio	kg/s
`mus`	static friction coefficient μ_s	
`muk`	kinetic friction coefficient μ_k	
Additional parameters:		
	acceleration of gravity	m/s^2
	sample time	s

leg. When a new DSP starts, the left and right leg signals are in the state input and reference output switched.

Each of the four separate PD controllers takes care of its own control variable given the corresponding reference signal value. The controlled variables are:

- α: torso angle.
- $\Delta\beta = \beta_R - \beta_L$: difference of the thigh angles.
- γ_L: left leg knee angle.
- γ_R: right leg knee angle.

The control signals of the knee angles are directly used as the *Biped model* input moments M_{L2} and M_{R2}. The output of the $\Delta\beta$ PD controller affects positively the right thigh moment M_{R1} and negatively the left thigh moment M_{L1}. The torso angle control was

separated from the actual gait control so that it is possible to decide the used angle independently. The control signal, however, needs to be added to the thigh moments. During the DSP the control signal is affecting the both thigh moments, but during the SSP only to the stance leg moment.

Figure 3.2 shows the PD controlled biped Simulink model. The input signal for the controller is the state of the biped system. The ouput signal is directly the moment vector (2.2). The parameter `st` defines the discrete sample time (in seconds) used in the controller.

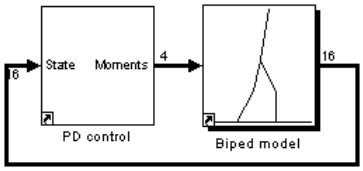


Figure 3.2: The PD controlled *Biped model*.

The internal structure of the controller is divided in two parts: The reference signal updating and the PD control (see Appendix B.2). The reference signal creation logic and PD controller parameters were tuned by hand, so the result is far from the optimum. The biped parameters used in the simulations are defined in the Matlab m-file `bipedparams.m` and the *PD controller* block parameters in the file `pdparams.m` (Appendix C.2). The reference signal creation, however, is based on the hardwired structure and parameters of the *PD controller/Create references* subsystem. So the presented controller works only with the specified biped parameter values and merely gives an example of how the biped system can be controlled.

3.3 *Clustered regression control* block

In the master's thesis [2], a data-based control method called *clustered regression* was used to model the mapping from the walking biped outputs to the input moments. The model structure is a combination of local, linear principal component regression (PCR) models that are distributed along the system operating trajectory. During one step (DSP and SSP) all these local models or *operating points* are gone through. The control estimate of the regression model in every time step is calculated as a combination of all the local PCR model outputs.

To form the clustered regression mapping, sample input-output-data from the walking trajectory is needed. This data is first divided into *clusters*, each of which is corresponding to one operating point and local model. Based on the data in the cluster, each local PCR model is calculated. When the total regression model estimate is formed as a weighted average of the local ones, the estimates of the closest local models are given the biggest weight, whereas the further ones are practically ignored. Figure 3.3 shows a sketch of the clustered regression model structure.

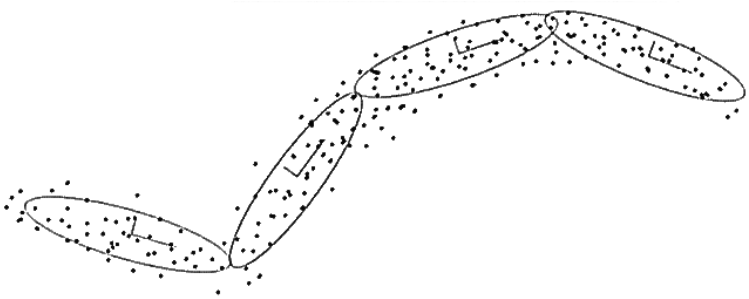


Figure 3.3: The clustered regression model structure consists of local PCR models (ellipses) located along the operating trajectory of the system. The dots represent the collected sample data points.

The regression model can directly be used to control the biped system when the estimated control signal is connected to the biped model input (Figure 3.4).

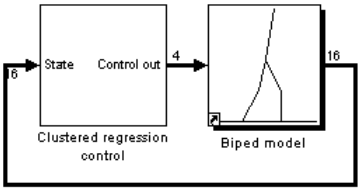


Figure 3.4: The clustered regression controlled *Biped model*.

The *Clustered regression control* block (Fig. 3.5) is built around a *Clustered regression* block which implements the clustered regression structure. The input and output signals need, however, some processing like mean-zeroing and scaling. Also the left and right leg signals are switched in every other step as was the case in the PD controller.

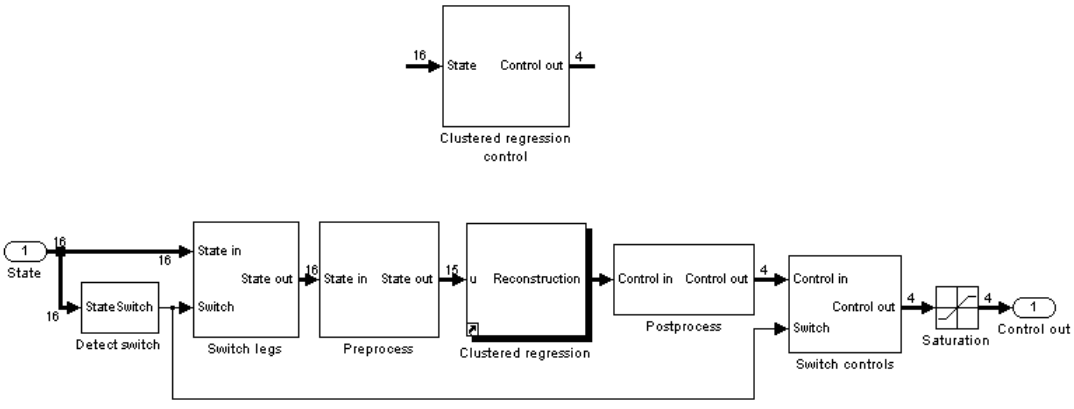


Figure 3.5: The *Clustered regression control* block.

The actual clustered regression structure is defined in the dialog box of the *Clustered regression* block (Fig. 3.6).

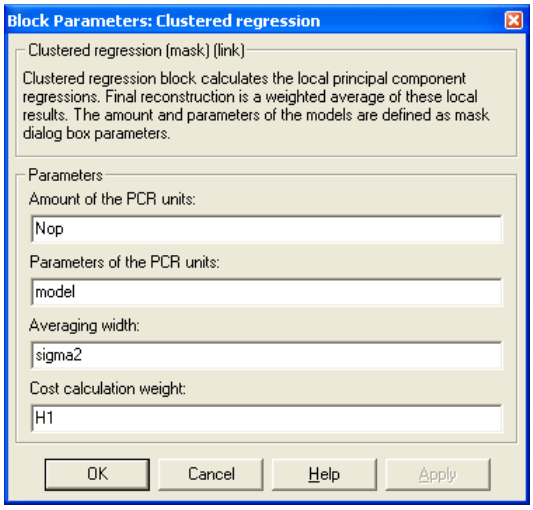


Figure 3.6: The dialog box of the *Clustered regression* block.

The parameters are:

- Amount of the PCR units: Defines how many PCR units the block includes.
- Parameters of the PCR units: Structure array containing the fields described in Table 3.2
- Averaging width: Parameter defining the width of the local results averaging function.
- Cost calculation weight: Weight used in the PCR modeling error calculation.

The PCR unit parameters are the properties of the data cluster connected to each operating point. The latent variable data used in the PCR is denoted as x, and, when scaled to unit variance, as z.

Table 3.2: Clustered regression structure (`model` structure array variable) contains the fields listed for each local PCR model.

field	description
`Cu0`	Mean of the cluster input data u
`Cy0`	Mean of the cluster output data y
`Pxx0`	Inverse of the latent data x covariance matrix
`Rxu0`	Covariance of the signals x and u
`Ryz0`	Covariance of the signals y and z

The *Clustered regression* block initial structure is updated dynamically every time the block is simulated. This ensures that the number of the PCR blocks agrees with the dialog box parameter values. The updating is done in the block mask initialization code.

The regression structure is created off-line using data collected from the PD controlled biped gait. The Matlab function `teach` (Appendix C.3) first preprocesses the simulated data and

then calculates the operating point locations and PCR models. The outputs of the function are the clustered regression model structure (Table 3.2) and the data properties structure. Table 3.3 shows the "data properties" variable fields, which are used for the input and output data processing.

Table 3.3: Teaching data properties (`dataprop` structure).

field	description
`inputmean`	Mean vector of whole input data
`inputvariance`	Variance of whole input data
`outputmean`	Mean vector of the output data
`outputvariance`	Variance of the output data

When the clustered regression control is used, the model is loaded and the other parameters of the simulink model are defined. An example of a *Clustered regression control* block parameter definition m-file (`crcparams.m`) is presented in Appendix C.4.

Detailed description of the *Clustered regression control* block initial structure is given in Appendix B.3. Clustered regression itself is explained in the publication [2].

Chapter 4

Graphical user interface

A graphical user interface (GUI) was created for simulating a Simulink model including the *Biped model* block and an arbitrary controller. The GUI enables the biped model simulation with specified parameters and the animating of the system behaviour. It is also possible to save all the simulation data or only one state of the system to the Matlab workspace. Figure 4.1 shows the GUI.

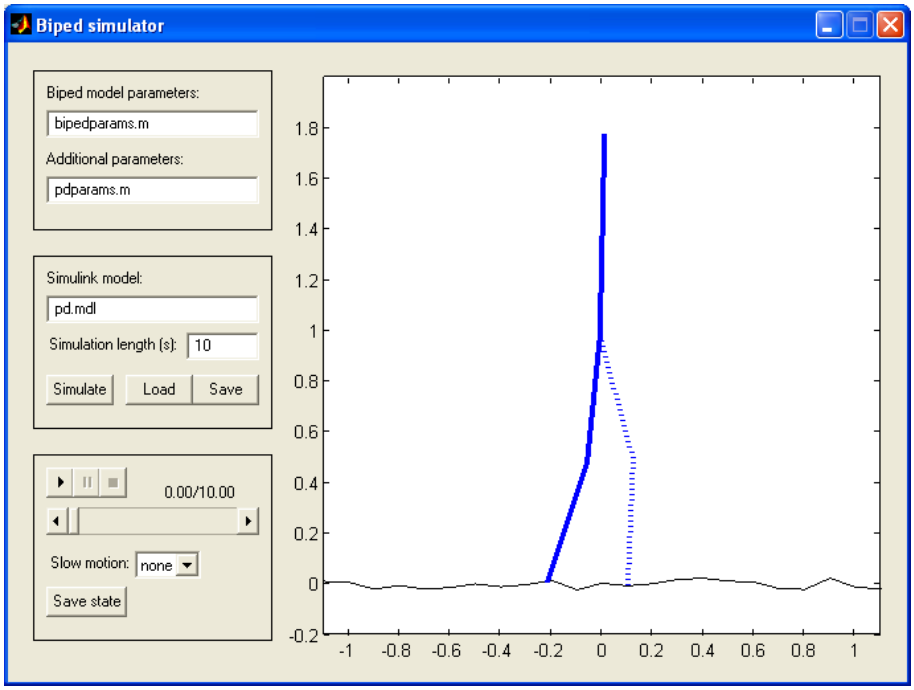


Figure 4.1: The graphical user interface for a controlled biped system simulation.

The GUI has the following functionalities:

- Biped model parameters: A Matlab m-file defining the *Biped model* block parameters. This file is executed before the simulation. An example file is presented in Appendix C.1.

- Additional parameters: A Matlab m-file containing other parameter definitions, for example for the controller block. The file is executed before the simulation.
- Simulink model: Simulink model to be simulated. The model must contain the *Biped model* block.
- Simulation length: Maximum simulation time in seconds. The simulation can be stopped earlier using a *Stop Simulation* block in the Simulink model.
- `Simulate` button: Executes the biped parameters, additional parameters and simulates the model.
- `Load` button: Loads simulation data stored in the Matlab workspace variable `data` (see below).
- `Save` button: Saves the simulated data to the Matlab workspace variable `data` (see below).
- Animation buttons (`play, pause, stop`): Show the animation of the system behaviour.
- Animation time and slider: Show the current animation state and animation total length (in seconds).
- Slow motion: Selects the animation slowing factor.
- `Save state` button: Saves the current system state to the Matlab workspace structure variable `currentstate`. The structure has the same fields as the "Robot initial state" parameter in the *Biped model* block dialog box.

The `data` variable stores the simulation data collected in the *Biped model* block. The `data` structure contains the following fields:

- `state`: A matrix whose rows are the state vectors of the system, that is, the *Biped model* block outputs (3.1) during the simulation.
- `controls`: The corresponding input vectors of the *Biped model* block.
- `st`: Sample time used in the simulation (in seconds).
- `robot`: The robot dimensions structure (see Table 3.1).
- `groundp`: The ground properties structure (see Table 3.1).

Chapter 5

Step-by-step examples

This section provides detailed examples of how to use the simulation tool and the related Simulink models. It is assumed that the simulation tool Matlab files are located in the current Matlab work directory.

5.1 PD controlled gait

The Simulink model *pd.mdl* simulates the PD controlled biped using *bibedlib.mdl* library blocks *Biped simulator* and *PD control*. Follow the steps below to simulate the biped model controlled by the provided default PD controller:

1. Open the simulink model *pd.mdl*.
2. Enter `simulator` in the Matlab command line to open the simulator GUI.
3. Check that the "Biped model parameters" field is set to `bipedparams.m`. This default file contains the parameter definitions for the *Biped model* block (see Appendix C.1). The parameters are referred to on the *Biped model* block dialog box. To show it, you can double click the *Biped model* block in the Simulink model.
4. Check that the "Additional parameters" field is set to `pdparams.m`. The file defines parameters for the *PD control* block (see Appendix C.2).
5. Check that the "Simulink model" field is set to `pd.mdl`.
6. Enter the appropriate simulation time to the "Simulation length" field.
7. Press `Simulate` button to simulate the model. When the simulation is completed, a beep is given and the time used for the simulation is printed to the Matlab command window. In case there were some errors during the simulation and the GUI is not responding, you have to close and reopen it before a new simulation.
8. To watch the animation of the biped behaviour use the animation buttons (`play, pause, stop`) or the animation slider.

5.2 Clustered regression controlled gait

Clustered regression control is based on the clustered regression model which is learned from the sample data collected from the PD controlled system. A ready clustered regression model structure is saved in the file `pdmodel.mat`, but it can be created as follows:

1. Simulate the *pd.mdl* model for about 100 s to create enough sample data.
2. Use `Save`-button to save the data to the Matlab workspace variable `data`.
3. Calculate the clustered regression model and data properties by entering the command
   `>> [model, dataprop] = teach(data)`
   to the Matlab command line.
4. Save the results to a file (here named as `mymodel.mat`):
   `>> save mymodel.mat model dataprop`

To simulate the clustered regression controlled biped, use the Simulink model *crc.mdl*:

1. Open the *Clustered regression control* block parameter definition file `crcparams.m` in the Matlab editor and find the line `load pdmodel.mat`. If you want to use the model file created above, change the line to `load mymodel.mat`.
2. Change in the simulator GUI the "Additional parameters" field to `crcparams.m` and "Simulink model" to `crc.mdl`.
3. Set the simulation length and simulate the model.

Bibliography

[1] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. R. Westervelt, C. Canudas-de-Wit, and J. W. Grizzle. RABBIT: A testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, October 2003.

[2] O. Haavisto. Kävelevän robottimallin kehittäminen sekä sen datapohjainen mallitus ja säätö. Master's thesis, Helsinki University of Technology, 2004.

[3] H. Hyötyniemi. Hebbian and Anti-Hebbian Learning: System theoretic approach. *Neural Networks (submitted)*, 2004.

Appendix A

Dynamic model formulas

The dynamic model of the biped has the form:

$$A(q)\ddot{q} = b(q, \dot{q}, M, F). \tag{A.1}$$

The structure of the biped and the parameters used are shown in the figure 2.1. The vector q contains the generalized coordinates, F the ground support forces and M the joint moments. As the system has seven degrees of freedom, there exists also seven partial differential equations. In the following, the exact formulas of the inertia matrix $A(q)$ and the right hand side vector $b(q, \dot{q}, F, M)$ are presented.

$A(q)$:

$$\begin{aligned}
A_{11} &= m_0 + 2m_1 + 2m_2 \\
A_{12} &= 0 \\
A_{13} &= (-2m_1 r_0 - 2m_2 r_0)\cos(\alpha) + (-l_1 m_2 - m_1 r_1)\cos(\alpha - \beta_L) - l_1 m_2 \cos(\alpha - \beta_R) \\
&\quad - m_1 r_1 \cos(\alpha - \beta_R) - m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) - m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
A_{14} &= (l_1 m_2 + m_1 r_1)\cos(\alpha - \beta_L) + m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) \\
A_{15} &= (l_1 m_2 + m_1 r_1)\cos(\alpha - \beta_R) + m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
A_{16} &= -m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) \\
A_{17} &= -m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
A_{21} &= 0 \\
A_{22} &= m_0 + 2m_1 + 2m_2 \\
A_{23} &= (2m_1 r_0 + 2m_2 r_0)\sin(\alpha) + (l_1 m_2 + m_1 r_1)\sin(\alpha - \beta_L) + l_1 m_2 \sin(\alpha - \beta_R) \\
&\quad + m_1 r_1 \sin(\alpha - \beta_R) + m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) + m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
A_{24} &= (-l_1 m_2 - m_1 r_1)\sin(\alpha - \beta_L) - m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) \\
A_{25} &= (-l_1 m_2 - m_1 r_1)\sin(\alpha - \beta_R) - m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
A_{26} &= m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) \\
A_{27} &= m_2 r_2 \sin(\alpha - \beta_R + \gamma_R)
\end{aligned}$$

$$
\begin{aligned}
A_{31} &= (-2m_1 r_0 - 2m_2 r_0)\cos(\alpha) + (-l_1 m_2 - m_1 r_1)\cos(\alpha - \beta_L) - l_1 m_2 \cos(\alpha - \beta_R) \\
&\quad - m_1 r_1 \cos(\alpha - \beta_R) - m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) - m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
A_{32} &= (2m_1 r_0 + 2m_2 r_0)\sin(\alpha) + (l_1 m_2 + m_1 r_1)\sin(\alpha - \beta_L) + l_1 m_2 \sin(\alpha - \beta_R) \\
&\quad + m_1 r_1 \sin(\alpha - \beta_R) + m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) + m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
A_{33} &= 2{l_1}^2 m_2 + 2m_1 {r_0}^2 + 2m_2 {r_0}^2 + 2m_1 {r_1}^2 + 2m_2 {r_2}^2 + (2l_1 m_2 r_0 \\
&\quad + 2m_1 r_0 r_1)\cos(\beta_L) + (2l_1 m_2 r_0 + 2m_1 r_0 r_1)\cos(\beta_R) \\
&\quad + 2m_2 r_0 r_2 \cos(\beta_L - \gamma_L) + 2l_1 m_2 r_2 \cos(\gamma_L) + 2m_2 r_0 r_2 \cos(\beta_R \\
&\quad - \gamma_R) + 2l_1 m_2 r_2 \cos(\gamma_R) \\
A_{34} &= -{l_1}^2 m_2 - m_1 {r_1}^2 - m_2 {r_2}^2 + (-l_1 m_2 r_0 - m_1 r_0 r_1)\cos(\beta_L) \\
&\quad - m_2 r_0 r_2 \cos(\beta_L - \gamma_L) - 2l_1 m_2 r_2 \cos(\gamma_L) \\
A_{35} &= -{l_1}^2 m_2 - m_1 {r_1}^2 - m_2 {r_2}^2 + (-l_1 m_2 r_0 - m_1 r_0 r_1)\cos(\beta_R) \\
&\quad - m_2 r_0 r_2 \cos(\beta_R - \gamma_R) - 2l_1 m_2 r_2 \cos(\gamma_R) \\
A_{36} &= m_2 r_2 (r_2 + r_0 \cos(\beta_L - \gamma_L) + l_1 \cos(\gamma_L)) \\
A_{37} &= m_2 r_2 (r_2 + r_0 \cos(\beta_R - \gamma_R) + l_1 \cos(\gamma_R)) \\
A_{41} &= (l_1 m_2 + m_1 r_1)\cos(\alpha - \beta_L) + m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) \\
A_{42} &= (-l_1 m_2 - m_1 r_1)\sin(\alpha - \beta_L) - m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) \\
A_{43} &= -{l_1}^2 m_2 - m_1 {r_1}^2 - m_2 {r_2}^2 + r_0(-l_1 m_2 - m_1 r_1)\cos(\beta_L) \\
&\quad - m_2 r_0 r_2 \cos(\beta_L - \gamma_L) - 2l_1 m_2 r_2 \cos(\gamma_L) \\
A_{44} &= {l_1}^2 m_2 + m_1 {r_1}^2 + m_2 {r_2}^2 + 2l_1 m_2 r_2 \cos(\gamma_L) \\
A_{45} &= 0 \\
A_{46} &= m_2 r_2 (-r_2 - l_1 \cos(\gamma_L)) \\
A_{47} &= 0 \\
A_{51} &= (l_1 m_2 + m_1 r_1)\cos(\alpha - \beta_R) + m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
A_{52} &= (-l_1 m_2 - m_1 r_1)\sin(\alpha - \beta_R) - m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
A_{53} &= -{l_1}^2 m_2 - m_1 {r_1}^2 - m_2 {r_2}^2 + r_0(-l_1 m_2 - m_1 r_1)\cos(\beta_R) \\
&\quad - m_2 r_0 r_2 \cos(\beta_R - \gamma_R) - 2l_1 m_2 r_2 \cos(\gamma_R) \\
A_{54} &= 0 \\
A_{55} &= {l_1}^2 m_2 + m_1 {r_1}^2 + m_2 {r_2}^2 + 2l_1 m_2 r_2 \cos(\gamma_R) \\
A_{56} &= 0 \\
A_{57} &= m_2 r_2 (-r_2 - l_1 \cos(\gamma_R)) \\
A_{61} &= -m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) \\
A_{62} &= m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) \\
A_{63} &= m_2 r_2 (r_2 + r_0 \cos(\beta_L - \gamma_L) + l_1 \cos(\gamma_L)) \\
A_{64} &= m_2 r_2 (-r_2 - l_1 \cos(\gamma_L)) \\
A_{65} &= 0 \\
A_{66} &= m_2 {r_2}^2 \\
A_{67} &= 0
\end{aligned}
$$

$$
\begin{aligned}
A_{71} &= -m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
A_{72} &= m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
A_{73} &= m_2 r_2 (r_2 + r_0 \cos(\beta_R - \gamma_R) + l_1 \cos(\gamma_R)) \\
A_{74} &= 0 \\
A_{75} &= m_2 r_2 (-r_2 - l_1 \cos(\gamma_R)) \\
A_{76} &= 0 \\
A_{77} &= m_2 {r_2}^2
\end{aligned}
$$

$b(q, \dot{q}, F, M)$:

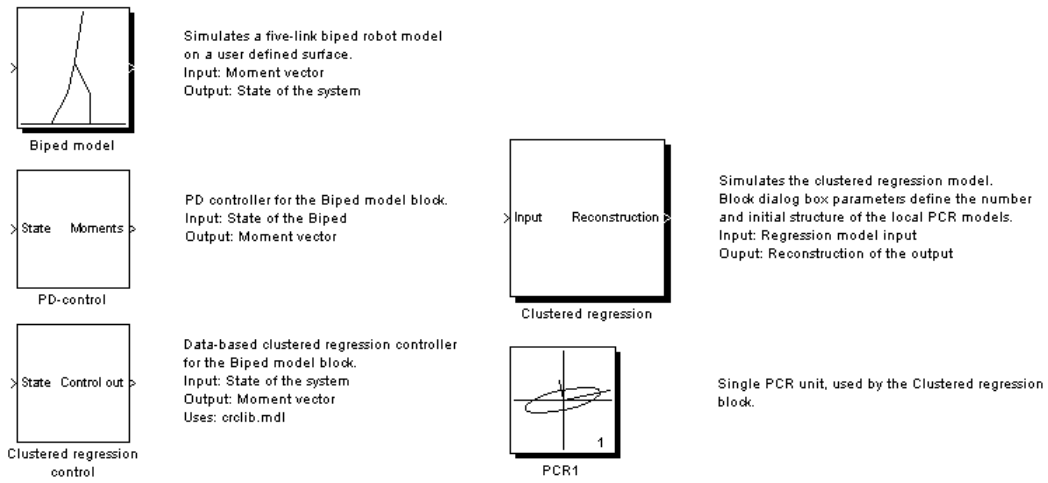
$$
\begin{aligned}
b_1 = &-2\dot{\alpha}^2 m_1 r_0 \sin(\alpha) + F_{Rx} - \dot{\beta}_R^2 m_1 r_1 \sin(\alpha - \beta_R) - \dot{\alpha}^2 l_1 m_2 \sin(\alpha \\
&- \beta_L) + F_{Lx} - \dot{\gamma}_L^2 m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) - \dot{\alpha}^2 m_1 r_1 \sin(\alpha - \beta_R) \\
&- \dot{\beta}_R^2 l_1 m_2 \sin(\alpha - \beta_R) - \dot{\alpha}^2 l_1 m_2 \sin(\alpha - \beta_R) - \dot{\gamma}_R^2 m_2 r_2 \sin(\alpha \\
&- \beta_R + \gamma_R) - \dot{\beta}_L^2 m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) - \dot{\beta}_L^2 m_1 r_1 \sin(\alpha - \beta_L) \\
&- \dot{\beta}_R^2 m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) - \dot{\alpha}^2 m_1 r_1 \sin(\alpha - \beta_L) \\
&- \dot{\beta}_L^2 l_1 m_2 \sin(\alpha - \beta_L) - \dot{\alpha}^2 m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
&- \dot{\alpha}^2 m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) + 2\dot{\alpha}\dot{\beta}_R l_1 m_2 \sin(\alpha - \beta_R) \\
&+ 2\dot{\beta}_R \dot{\gamma}_R m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) - 2\dot{\alpha}\dot{\gamma}_R m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
&+ 2\dot{\alpha}\dot{\beta}_L m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) + 2\dot{\alpha}\dot{\beta}_L m_1 r_1 \sin(\alpha - \beta_L) \\
&+ 2\dot{\alpha}\dot{\beta}_R m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) + 2\dot{\alpha}\dot{\beta}_L l_1 m_2 \sin(\alpha - \beta_L) \\
&- 2\dot{\alpha}^2 m_2 r_0 \sin(\alpha) + 2\dot{\alpha}\dot{\beta}_R m_1 r_1 \sin(\alpha - \beta_R) \\
&+ 2\dot{\beta}_L \dot{\gamma}_L m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) - 2\dot{\alpha}\dot{\gamma}_L m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) \\
b_2 = &-\dot{\beta}_L^2 m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) + F_{Ry} - 2g m_2 - \dot{\gamma}_R^2 m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
&- \dot{\beta}_R^2 m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) - \dot{\alpha}^2 m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
&- \dot{\gamma}_L^2 m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) - \dot{\alpha}^2 m_1 r_1 \cos(\alpha - \beta_R) \\
&- \dot{\beta}_R^2 m_1 r_1 \cos(\alpha - \beta_R) - \dot{\alpha}^2 (2m_1 + 2m_2) r_0 \cos(\alpha) - \dot{\alpha}^2 l_1 m_2 \cos(\alpha \\
&- \beta_R) - \dot{\beta}_R^2 l_1 m_2 \cos(\alpha - \beta_R) + F_{Ly} - \dot{\alpha}^2 m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) - 2g m_1 \\
&+ 2\dot{\alpha}\dot{\beta}_R l_1 m_2 \cos(\alpha - \beta_R) + 2\dot{\beta}_R \dot{\gamma}_R m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
&- 2\dot{\alpha}\dot{\gamma}_R m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) + 2\dot{\alpha}\dot{\beta}_R m_2 r_2 \cos(\alpha - \beta_R + \gamma_R) \\
&- 2\dot{\alpha}\dot{\gamma}_L m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) + 2\dot{\beta}_L \dot{\gamma}_L m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) \\
&+ 2\dot{\alpha}\dot{\beta}_L m_2 r_2 \cos(\alpha - \beta_L + \gamma_L) + 2\dot{\alpha}\dot{\beta}_R m_1 r_1 \cos(\alpha - \beta_R) \\
&- (\dot{\alpha}\dot{\beta}_L(-2l_1 m_2 - 2m_1 r_1) + \dot{\alpha}^2 (l_1 m_2 + m_1 r_1) + \dot{\beta}_L^2 (l_1 m_2 \\
&+ m_1 r_1)) \cos(\alpha - \beta_L) - g m_0
\end{aligned}
$$

$$
\begin{aligned}
b_3 = {} & \dot{\gamma}_R^2 l_1 m_2 r_2 \sin(\gamma_R) + F_{Ry} l_2 \sin(\alpha - \beta_R + \gamma_R) - F_{Lx} l_1 \cos(\alpha - \beta_L) \\
& - F_{Rx} l_1 \cos(\alpha - \beta_R) - F_{Lx} l_2 \cos(\alpha - \beta_L + \gamma_L) + F_{Ly} r_0 \sin(\alpha) + F_{Ry} r_0 \sin(\alpha) \\
& + F_{Ly} l_1 \sin(\alpha - \beta_L) + F_{Ly} l_2 \sin(\alpha - \beta_L + \gamma_L) - g m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
& - \dot{\gamma}_R^2 m_2 r_0 r_2 \sin(\beta_R - \gamma_R) - \dot{\beta}_R^2 m_2 r_0 r_2 \sin(\beta_R - \gamma_R) \\
& - g m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) + \dot{\gamma}_L^2 l_1 m_2 r_2 \sin(\gamma_L) \\
& - \dot{\gamma}_L^2 m_2 r_0 r_2 \sin(\beta_L - \gamma_L) - \dot{\beta}_R^2 m_1 r_0 r_1 \sin(\beta_R) \\
& - \dot{\beta}_L^2 m_2 r_0 r_2 \sin(\beta_L - \gamma_L) - \dot{\beta}_R^2 l_1 m_2 r_0 \sin(\beta_R) \\
& - \dot{\beta}_L^2 m_1 r_0 r_1 \sin(\beta_L) - g l_1 m_2 \sin(\alpha - \beta_R) + F_{Ry} l_1 \sin(\alpha - \beta_R) - g m_1 r_1 \sin(\alpha \\
& - \beta_R) - g l_1 m_2 \sin(\alpha - \beta_L) - g m_1 r_1 \sin(\alpha - \beta_L) - \dot{\beta}_L^2 l_1 m_2 r_0 \sin(\beta_L) \\
& - (F_{Lx} r_0 + F_{Rx} r_0) \cos(\alpha) - 2\dot{\alpha}\dot{\gamma}_L m_2 r_0 r_2 \sin(\beta_L - \gamma_L) \\
& + 2\dot{\beta}_L \dot{\gamma}_L m_2 r_0 r_2 \sin(\beta_L - \gamma_L) + 2\dot{\alpha}\dot{\beta}_L m_2 r_0 r_2 \sin(\beta_L \\
& - \gamma_L) + 2\dot{\alpha}\dot{\beta}_R m_1 r_0 r_1 \sin(\beta_R) - 2 g m_2 r_0 \sin(\alpha) \\
& + 2\dot{\alpha}\dot{\gamma}_R l_1 m_2 r_2 \sin(\gamma_R) + 2\dot{\alpha}\dot{\beta}_R m_2 r_0 r_2 \sin(\beta_R - \gamma_R) \\
& - 2\dot{\alpha}\dot{\gamma}_R m_2 r_0 r_2 \sin(\beta_R - \gamma_R) + 2\dot{\alpha}\dot{\gamma}_L l_1 m_2 r_2 \sin(\gamma_L) \\
& - 2\dot{\beta}_L \dot{\gamma}_L l_1 m_2 r_2 \sin(\gamma_L) - 2 g m_1 r_0 \sin(\alpha) \\
& + 2\dot{\alpha}\dot{\beta}_R l_1 m_2 r_0 \sin(\beta_R) + 2\dot{\alpha}\dot{\beta}_L l_1 m_2 r_0 \sin(\beta_L) \\
& + 2\dot{\alpha}\dot{\beta}_L m_1 r_0 r_1 \sin(\beta_L) - 2\dot{\beta}_R \dot{\gamma}_R l_1 m_2 r_2 \sin(\gamma_R) \\
& + 2\dot{\beta}_R \dot{\gamma}_R m_2 r_0 r_2 \sin(\beta_R - \gamma_R) - F_{Rx} l_2 \cos(\alpha - \beta_R + \gamma_R) \\
b_4 = {} & M_{L1} + F_{Lx} l_1 \cos(\alpha - \beta_L) + F_{Lx} l_2 \cos(\alpha - \beta_L + \gamma_L) - F_{Ly} l_1 \sin(\alpha - \beta_L) \\
& + g l_1 m_2 \sin(\alpha - \beta_L) + g m_1 r_1 \sin(\alpha - \beta_L) - \dot{\alpha}^2 l_1 m_2 r_0 \sin(\beta_L) \\
& - \dot{\alpha}^2 m_1 r_0 r_1 \sin(\beta_L) - \dot{\alpha}^2 m_2 r_0 r_2 \sin(\beta_L - \gamma_L) \\
& - 2\dot{\alpha}\dot{\gamma}_L l_1 m_2 r_2 \sin(\gamma_L) + 2\dot{\beta}_L \dot{\gamma}_L l_1 m_2 r_2 \sin(\gamma_L) \\
& - \dot{\gamma}_L^2 l_1 m_2 r_2 \sin(\gamma_L) - F_{Ly} l_2 \sin(\alpha - \beta_L + \gamma_L) + g m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) \\
b_5 = {} & M_{R1} + F_{Rx} l_1 \cos(\alpha - \beta_R) + F_{Rx} l_2 \cos(\alpha - \beta_R + \gamma_R) - F_{Ry} l_1 \sin(\alpha - \beta_R) \\
& + g l_1 m_2 \sin(\alpha - \beta_R) + g m_1 r_1 \sin(\alpha - \beta_R) - \dot{\alpha}^2 l_1 m_2 r_0 \sin(\beta_R) \\
& - \dot{\alpha}^2 m_1 r_0 r_1 \sin(\beta_R) - \dot{\alpha}^2 m_2 r_0 r_2 \sin(\beta_R - \gamma_R) \\
& - 2\dot{\alpha}\dot{\gamma}_R l_1 m_2 r_2 \sin(\gamma_R) + 2\dot{\beta}_R \dot{\gamma}_R l_1 m_2 r_2 \sin(\gamma_R) \\
& - \dot{\gamma}_R^2 l_1 m_2 r_2 \sin(\gamma_R) - F_{Ry} l_2 \sin(\alpha - \beta_R + \gamma_R) + g m_2 r_2 \sin(\alpha - \beta_R + \gamma_R) \\
b_6 = {} & M_{L2} - F_{Lx} l_2 \cos(\alpha - \beta_L + \gamma_L) + \dot{\alpha}^2 m_2 r_0 r_2 \sin(\beta_L - \gamma_L) \\
& - \dot{\alpha}^2 l_1 m_2 r_2 \sin(\gamma_L) + 2\dot{\alpha}\dot{\beta}_L l_1 m_2 r_2 \sin(\gamma_L) \\
& - \dot{\beta}_L^2 l_1 m_2 r_2 \sin(\gamma_L) + F_{Ly} l_2 \sin(\alpha - \beta_L + \gamma_L) - g m_2 r_2 \sin(\alpha - \beta_L + \gamma_L) \\
b_7 = {} & M_{R2} - F_{Rx} l_2 \cos(\alpha - \beta_R + \gamma_R) + \dot{\alpha}^2 m_2 r_0 r_2 \sin(\beta_R - \gamma_R) \\
& - \dot{\alpha}^2 l_1 m_2 r_2 \sin(\gamma_R) + 2\dot{\alpha}\dot{\beta}_R l_1 m_2 r_2 \sin(\gamma_R) \\
& - \dot{\beta}_R^2 l_1 m_2 r_2 \sin(\gamma_R) + F_{Ry} l_2 \sin(\alpha - \beta_R + \gamma_R) - g m_2 r_2 \sin(\alpha - \beta_R + \gamma_R)
\end{aligned}
$$

Appendix B

Simulink block initial structures

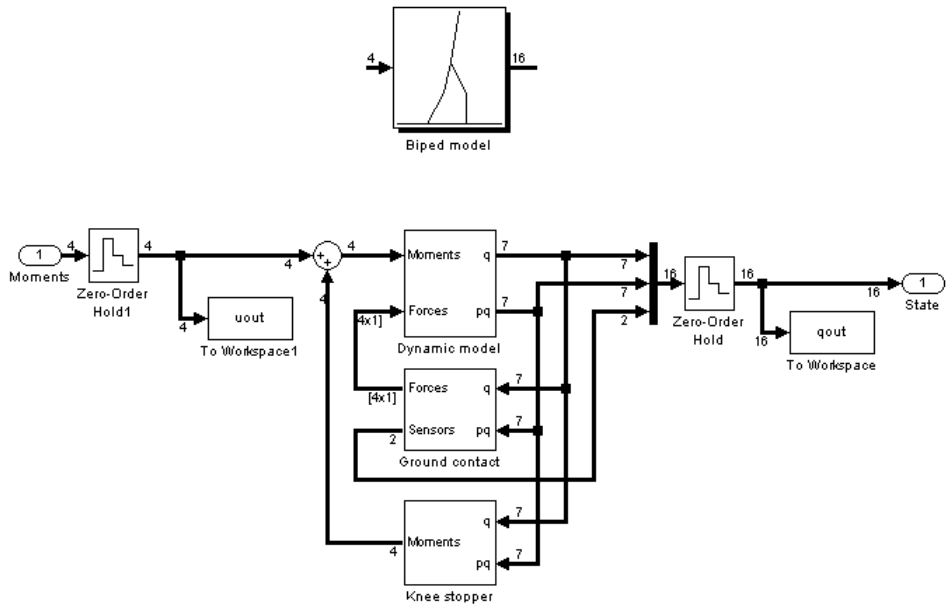
This appendix presents the Simulink blocks created for the biped simulation and control. *Biped model*, *PD control* and *Clustered regression control* blocks are located in the library model *bipedlib.mdl*. The *Clustered regression control* block uses also the library model *crclib.mdl*, which contains the *Clustered regression* block.



bipedlib.mdl *crclib.mdl*

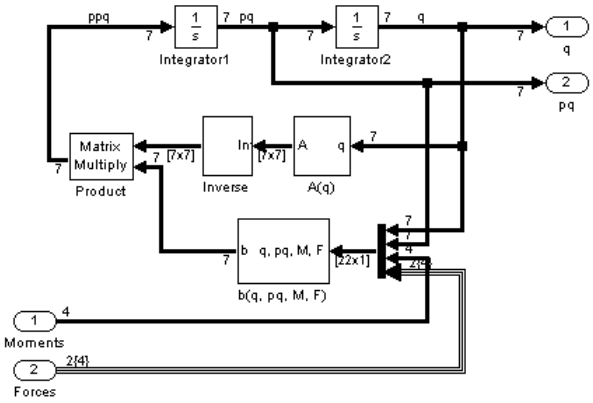
In the following, the blocks and subsystems belonging to them are gone through and explained one by one. The titles show the locations of the subsystems in the current block structure.

B.1 *Biped model*



The *Biped model* block simulates continuously the complete biped walker on a surface. The input and output signals are discretized using zero order hold and saved to the Matlab workspace. The most important part of the block is the *Dynamic model* subsystem, which simulates the dynamic equations (2.4). The other two subsystems are modeling the ground contact forces and knee angle limiter moments.

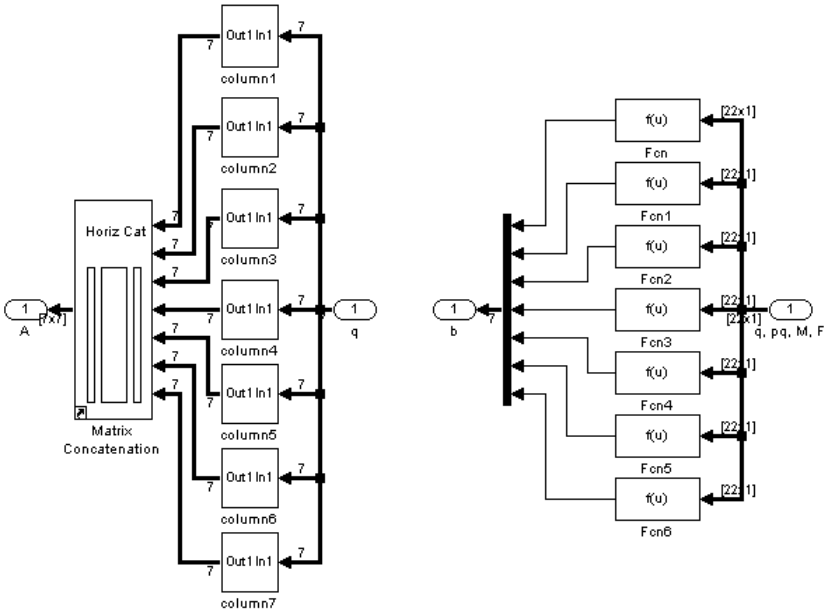
Biped model/Dynamic model



This block calculates the biped model dynamic equations (2.4). The inputs are the joint moments (2.2) and ground contact support forces (2.3), and the outputs the coordinates q (2.1) and their time derivatives $\dot{q}$. The b vector is multiplied by inversion of the inertia matrix A to form the acceleration vector $\ddot{q}$. Initial values for the *Integrator* blocks are set according to

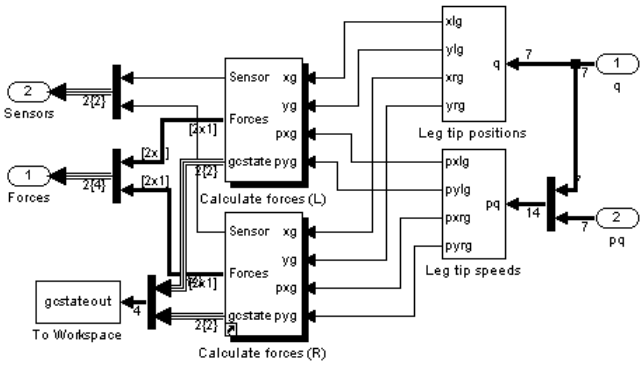
the robot initial state structure defined on the *Biped model* block dialog box (see Figure 3.1 and Table 3.1).

Biped model/Dynamic model/A(q) and *b(q, pq, M, F)*



These blocks calculate elementwise the inertia matrix $A(q)$ (left) and the vector $b(q, \dot{q}, M, F)$ (right) using Simulink *Fcn* blocks and the formulas of Appendix A. Each *column* block contains seven *Fcn* blocks, that is, one for each element of the matrix.

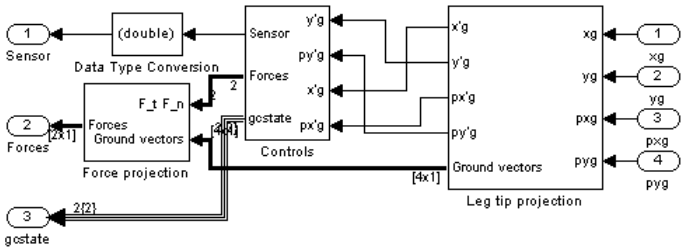
Biped model/Ground contact



The *Ground contact* block determines the ground support forces and touch sensor values for both leg tips. The input is the state of the biped system. The leg tip cartesian coordinates (x, y) and speeds are first calculated using *Fcn* blocks. Then the legs are handled separately by two identical *Calculate forces* subblocks, which take the corresponding initial

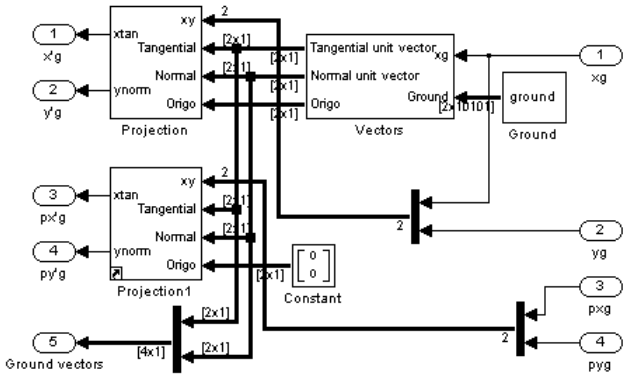
`gcstate` variables as their mask parameters. The `gcstate` vector is saved to the Matlab workspace at every sampling interval. It is needed only, if the user wants to save a certain state of the simulation (`Save state` button).

Biped model/Ground contact/Calculate forces (L)



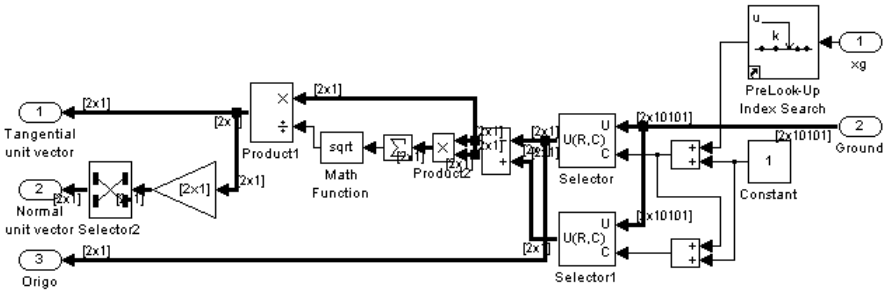
This block first projects the leg tip coordinates to the local (x', y') coordinate system (*Leg tip projection*), then calculates the tangential and normal forces (*Controls*) and finally projects them to the support forces (2.3) (*Force projection*). Also the current `gcstate` and sensor values are given as the output signals.

Biped model/Ground contact/Calculate forces (L)/Leg tip projection



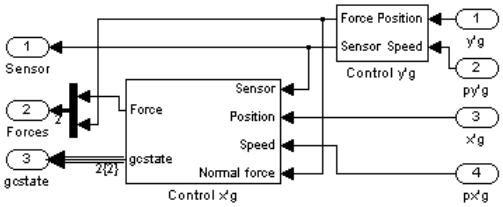
To calculate the projection, the unit coordinate vectors of the (x', y') coordinate system are first determined (*Vectors*). In the identical *Projection* blocks the given origo is first subtracted from the `xy` input vectors (the position and speed of the leg tip), then the projection is carried out using dot products. The `ground` variable has the ground points defined in the *Biped model* block parameters.

Biped model/Ground contact/Calculate forces (L)/Leg tip projection/Vectors



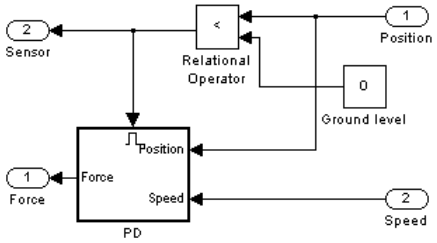
The coordinates of the ends of the current ground segment are selected from the ground surface matrix `ground` using Matlab *PreLook-Up Index Search* and *Selector* blocks. Then the (x', y') coordinate axis unit vectors and origo are calculated.

Biped model/Ground contact/Calculate forces (L)/Controls



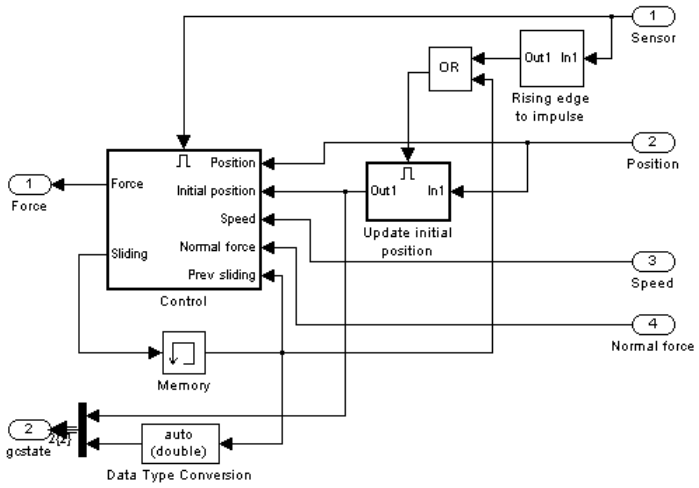
The *Controls* block includes control blocks for tangential and normal forces. The inputs are the current position and speed of the leg tip in the (x', y') coordinate system, and the outputs the touching sensor values, the normal and tangential forces and the `gcstate` variable.

Biped model/Ground contact/Calculate forces (L)/Controls/ Control y'g



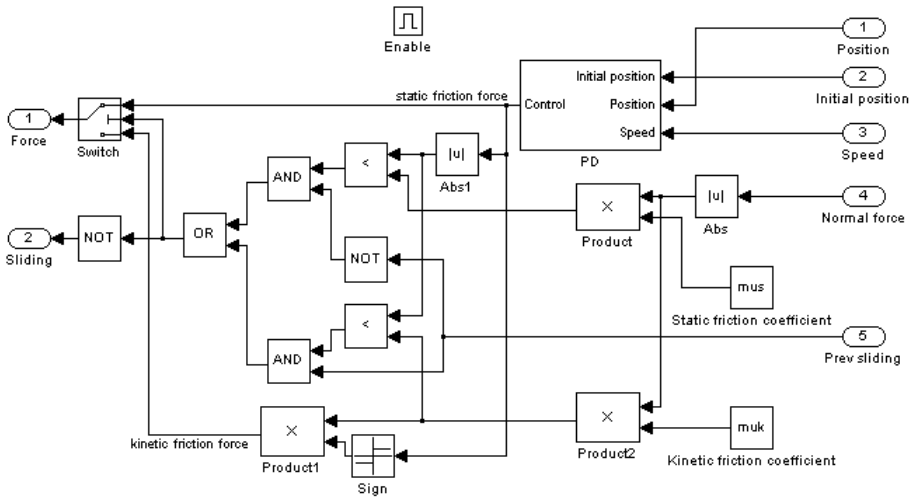
The normal force is calculated when the leg tip position is below the zero level. The *PD* block output equals to the equation (2.5) and is restricted to positive values. If not enabled, the *PD* block output is zero.

Biped model/Ground contact/Calculate forces (L)/Controls/ Control x'g



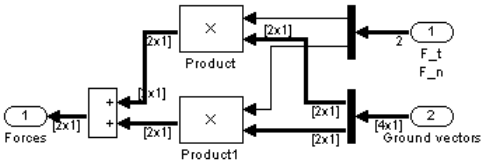
The *Control x'g* block stores the initial position of the leg tip x'_{0L} (or x'_{0R}) in the enabled subsystem *Update initial position*. The value is updated when the leg first touches the ground or if the leg was sliding at the previous simulation step. The `gcstate` variables define the initial states of the subsystem block output (x'_{0L}, x'_{0R}) and the *Memory* block (sliding$_L$, sliding$_R$).

Biped model/Ground contact/Calculate forces (L)/Controls/ Control x'g/Control



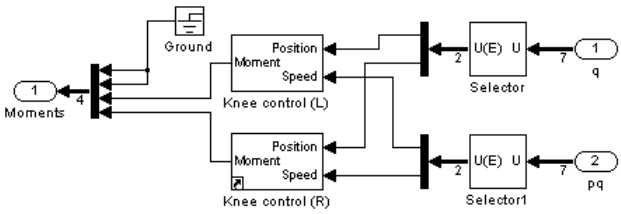
The *Control* block decides whether the leg should slide or not and calculates the corresponding tangential control force. If the leg is not sliding, the force is the output of the *PD* block according to the equation (2.6). When the leg is sliding, the force is calculated using the equation (2.8).

Biped model/Ground contact/Calculate forces (L)/Force projection



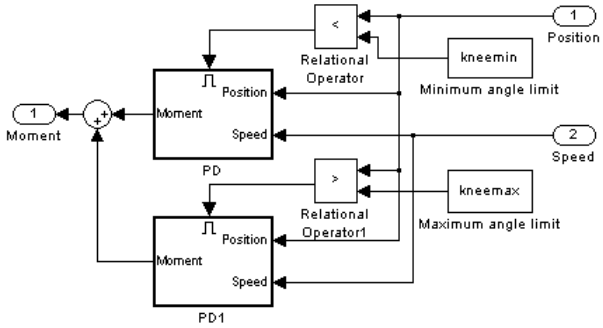
The *Force projection* block projects the normal and tangential forces acting to the leg tip to the external forces (2.3). The `Ground vectors` input contains the normal and tangential ground unit vectors.

Biped model/Knee stopper



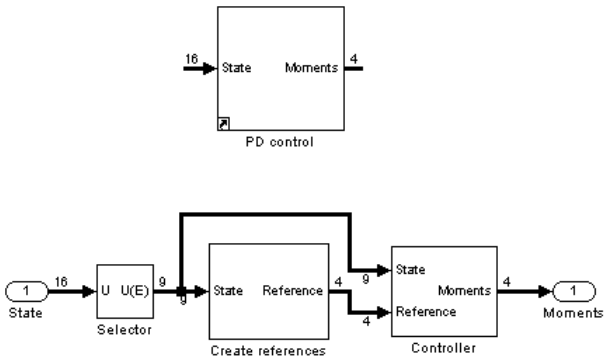
The *Knee stopper* block adds a moment value to the knee joint moments if the angle of the joint does not stay in the given limits. Both knees are controlled with similar *Knee control* blocks.

Biped model/Knee stopper/Knee control (L)



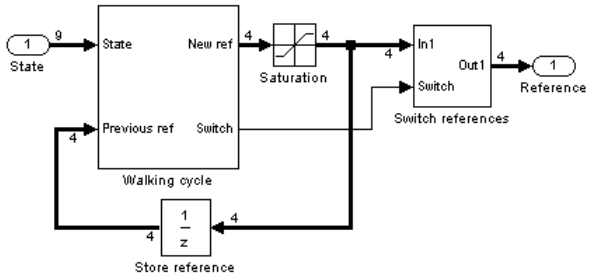
The given knee angle limits are stored in `kneemin` and `kneemax` variables. If the knee angle goes below the lower limit, the *PD* block gives the limiting moment value, which is restricted to positive values. The maximum angle limiter works similarly, but the moment value is now always negative.

B.2 *PD control*



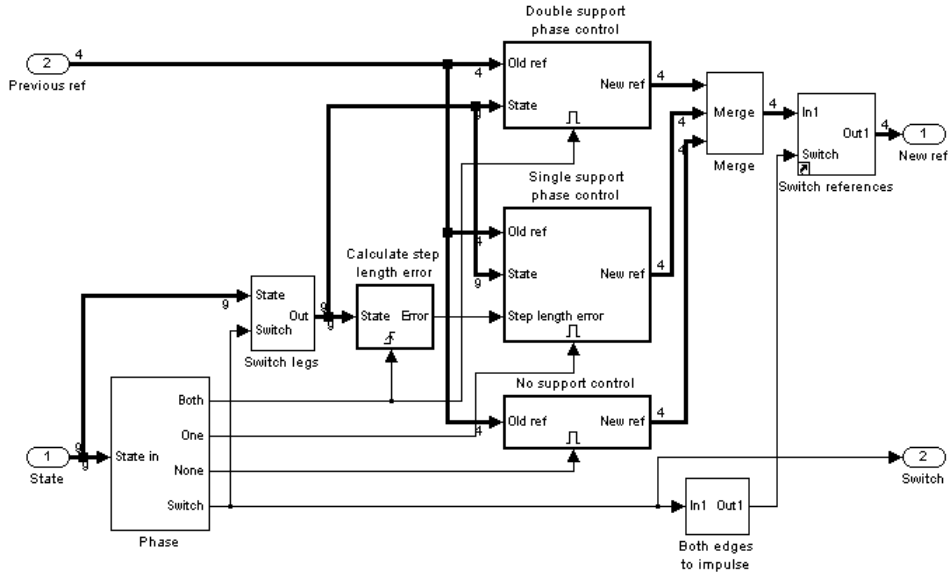
The *PD control* block uses the state of the system to calculate new moment values which control the biped gait. First the reference signals are calculated based on the system coordinates and touching sensor values. Then the input moments for the *Biped model* block are calculated using the references and coordinates.

PD control/Create references



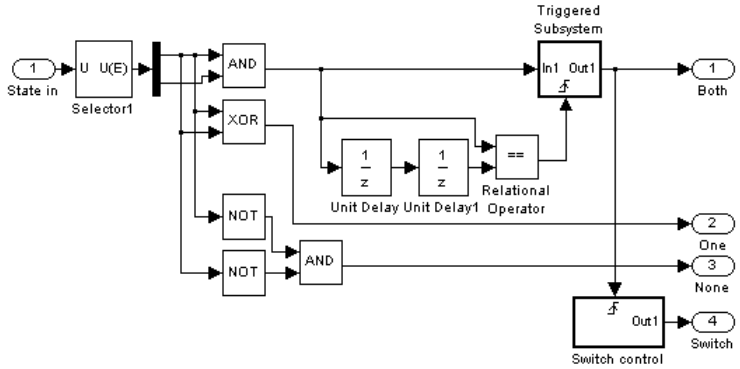
The reference signal calculation is made by updating the previous reference values according to the current system state (*Walking cycle* subsystem). The output signal leg values are switched during every other step to create a left-right symmetrical gait (*Switch references* subsystem). The initial reference values (`ref`) defined in the `pdparams.m` are used as the initial values for the *Store reference* block.

PD control/Create references/Walking cycle



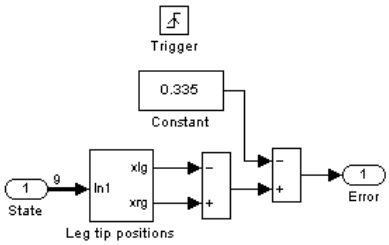
The updating of the reference signals depends on the phase of the step, which is determined using the sensor values of the state signal. The *Phase* block selects the corresponding enabled subsystem to update the references and gives also the binary `switch` signal, which shows whether the legs should be switched or not. The reference signals are switched once inside the update cycle at the beginning of each step.

PD control/Create references/Walking cycle/Phase



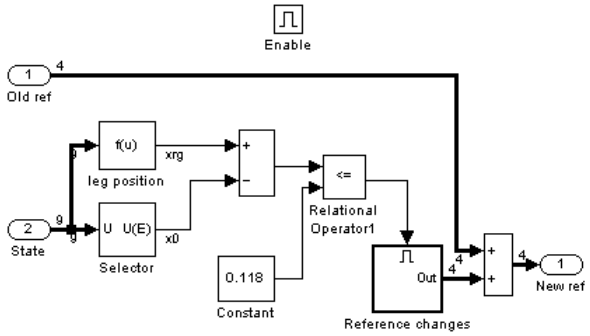
The *Phase* block first selects the sensor values from the state signal and determines then the number of legs on the ground. The `Both` output is enabled only when the swinging leg has touched the ground for two time steps. This tries to ensure that the new double support phase really has begun. When triggered with a rising edge the *Switch control* switches its output between 1 and 0. The value is 1 when the left leg is swinging and 0 when the right leg is swinging.

PD control/Create references/Walking cycle/Step length error



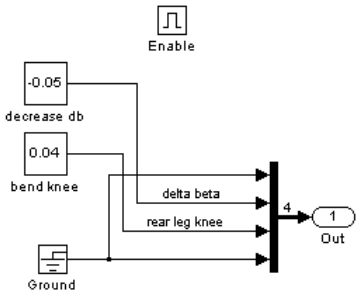
The *Step length error* block calculates the difference between the leg tip distance and a nominal value. The calculation is updated every time when a new DSP starts.

PD control/Create references/Walking cycle/Double support phase control



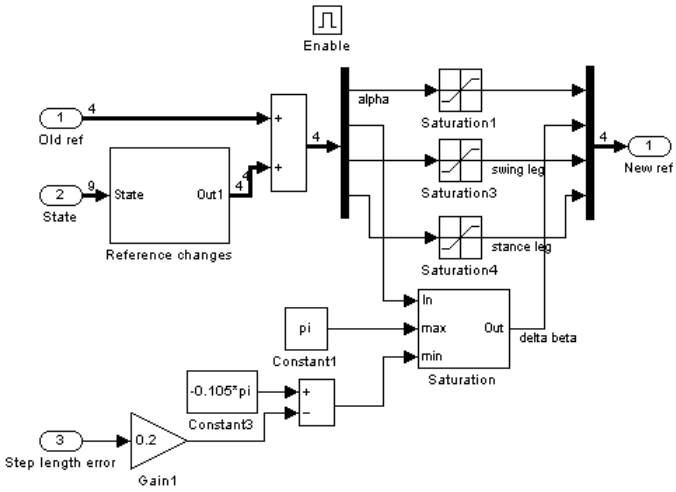
When both legs are touching the ground, the updating of the reference signals is done in the *Double support phase control* block. As long as the difference of the torso center of mass and the front leg tip is bigger in the x-direction than the specified constant value, the references are not changed. After the torso has swung forward enough, the *Reference changes* block is enabled and its outputs summed to the previous reference values.

PD control/Create references/Walking cycle/Double support phase control/Reference changes



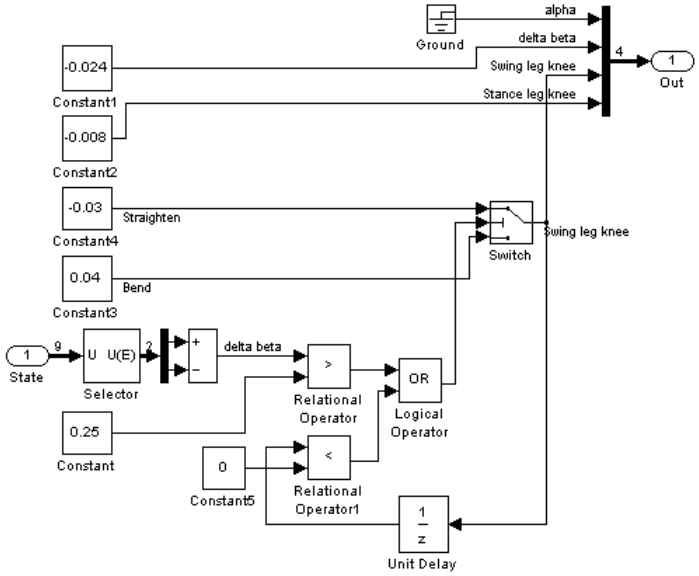
The reference changes are simply constant values, which are added to the old values. In the end of the DSP the angle between the thighs is decreased and the rear leg knee is bend in order to lift the rear leg from the ground.

PD control/Create references/Walking cycle/Single support phase control



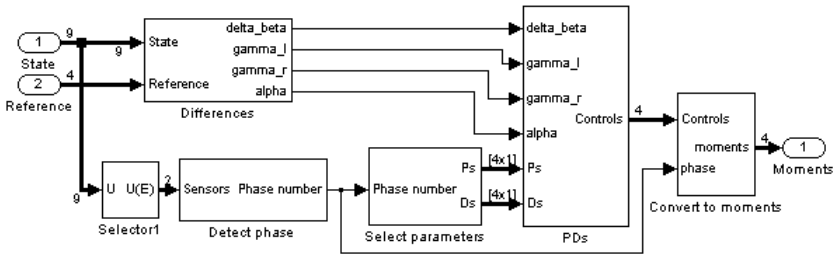
In the SSP the reference changes are also done in the *Reference changes* block. Additionally, the reference values are saturated to certain upper and lower limits. Specially the lower limit of the $\Delta\beta$ angle is calculated as a function of the previous step length error. This works like a feedback control stabilizing the gait.

PD control/Create references/Walking cycle/Single support phase control/Reference changes



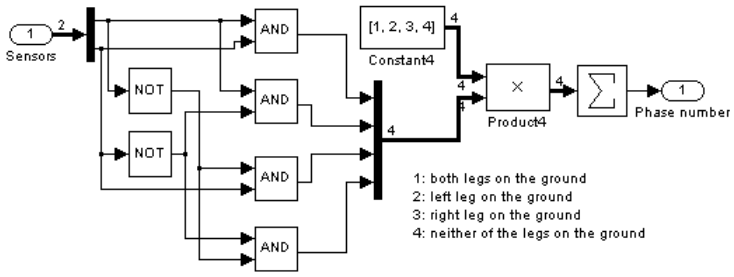
The constant reference changes are added to the reference signals during the SSP. The thigh angle difference $\Delta\beta$ is decreased all the time to force the swing leg forward and the stance leg knee is straightened to lift the biped. During the first part of the SSP the swing leg knee is bend to prevent the leg tip touch the ground, but when the leg has swung forward enough, the knee starts to straighten. Using the *Unit delay* block it is also ensured that when the straightening once has begun, it cannot be stopped.

PD control/Controller



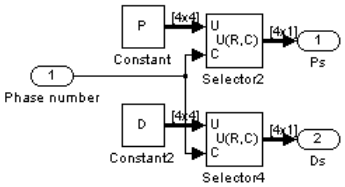
The *Controller* block first calculates the differences of the controlled variables and reference values. The parameters for the PD controllers are chosen according to the phase of the step. After the control signals are computed, they are converted to the moments (2.2).

PD control/Controller/Detect phase



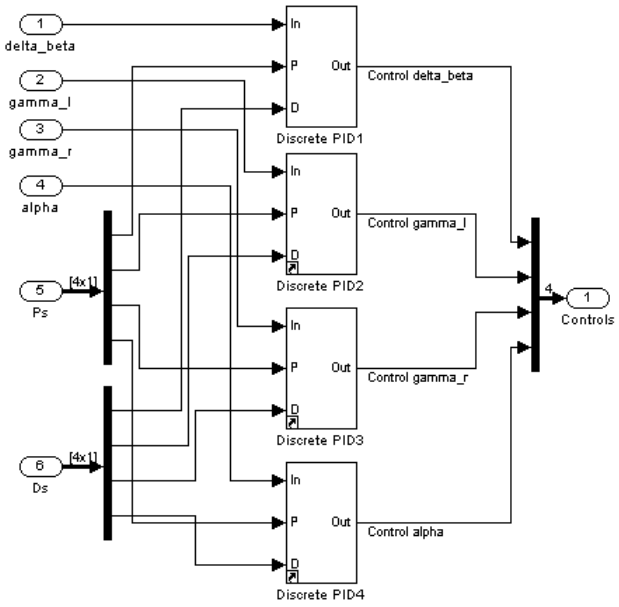
The phase detection is based on the sensor values. The output is a number (1–4) indicating which legs touch the ground.

PD control/Controller/Select parameters



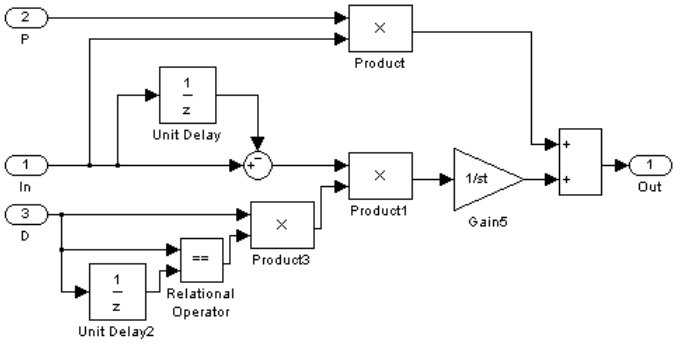
The right parameters for the current phase are selected taking the corresponding column from the matrices `P` and `D`.

PD control/Controller/PDs



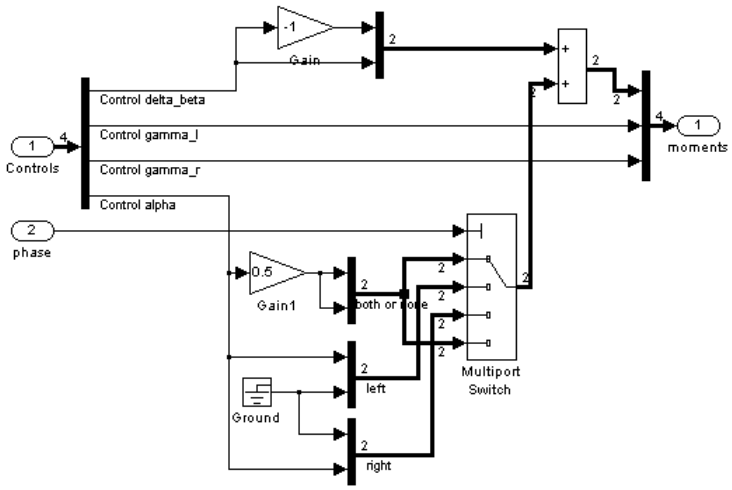
The four identical PD controllers take the difference signals and the parameters (P, D) as inputs and give the corresponding control signals as outputs.

PD control/Controller/PDs/Discrete PD1



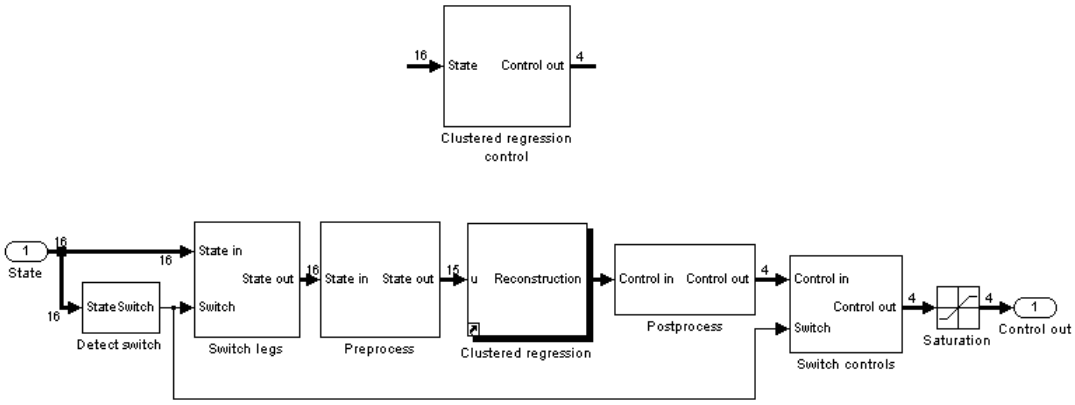
The only special feature in the discrete PD controllers is the speed term zeroing. When the D parameter is changed the speed term is zeroed for one time step in order to avoid peaks in the output signal.

PD control/Controller/Convert to moments



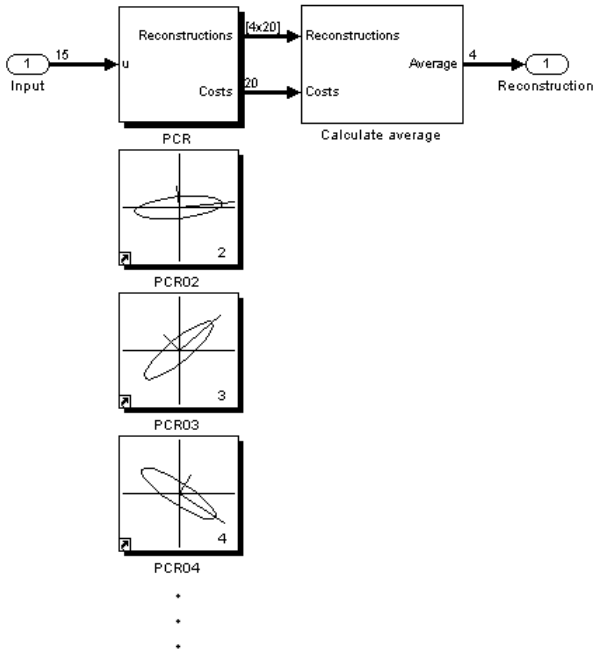
Because all of the controlled variables cannot be affected straight through the inputs of the biped system, a conversion of the control values is needed. The $\Delta\beta$ control signal is divided to both thigh moments, and the torso angle control signal is added to the thigh moment of the leg that is touching the ground. If both legs are in contact with the ground, the control is divided equally between the two moments.

B.3 *Clustered regression control*



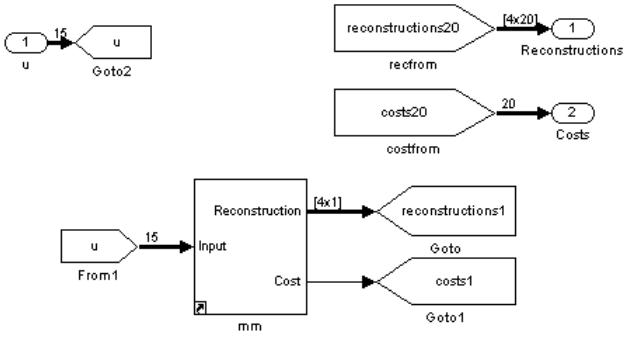
The *Clustered regression control* block applies the clustered regression model for the biped control. The input for the block is the state of the biped model and the output the control signal. In the *Switch legs* and *Switch controls* subsystems the left and right leg signals are switched in every other step according to the `switch` signal created in the *Detect switch* block. The x_0 coordinate is removed from the input signal in the *Preprocess* subsystem, in which also the signal is mean-zeroed and scaled to unit variance using the `dataprop` structure field values (see Table 3.3). After the *Clustered regression* output is calculated, the mean and variance of the estimate are restored in the *Postprocess* block, correspondingly.

Clustered regression control/Clustered regression



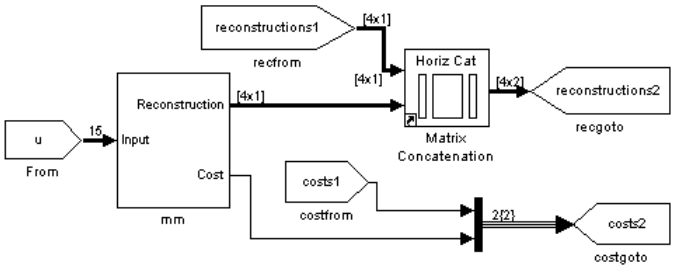
The *Clustered regression* block is linked from the library *crclib.mdl*. The structure of the is updated so that the number of the *PCR* subsystems, each representing one local model of the regression structure, corresponds to the dialog box parameters (Fig. 3.6). All of the *PCR* blocks are identical except the first one, which collects the results from the other blocks. The connections between the blocks are made using Simulink *Goto* and *From* blocks, so no connection lines are needed. In the *Calculate average* block the local estimates are averaged to form the whole regression model estimate.

Clustered regression control/Clustered regression/PCR



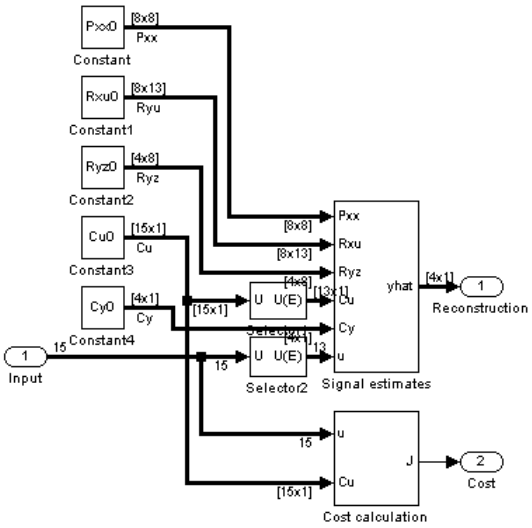
In this first *PCR* block the input is directed to the "u" goto tag, which is then used in every *PCR* block. The actual PCR calculation is done in the *mm* block (linked from *crclib/PCR1/mm*), its outputs being the regression estimate and the corresponding cost value. The results are sent to the next *PCR* block, which combines its own results with the previous results increasing the dimension of the reconstruction and cost signals by one. From the last *PCR* block (in this case number 20) the signals are directed back to the first block.

Clustered regression control/Clustered regression/PCRxx



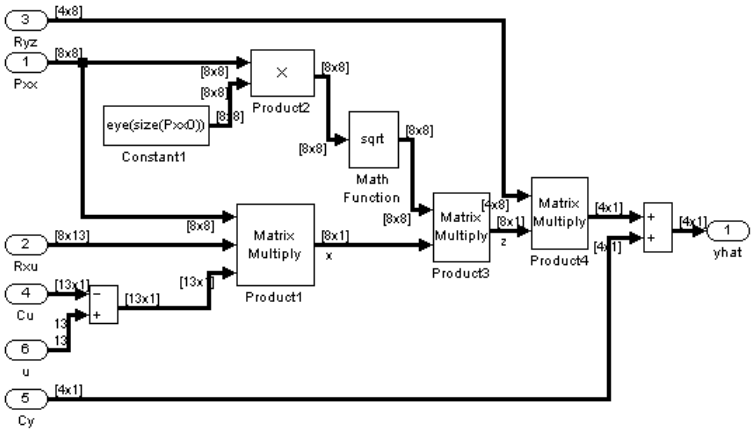
All the rest PCR calculation blocks (*PCR02, PCR03, PCR04,* etc.) are linked copies of the *crclib/PCR1* block, and the *Goto* and *From* tag definitions are updated in each blocks mask initialization commands. The reconstruction and cost signals are obtained from the previous PCR block and sent with the current PCR model estimate and cost to the next PCR block.

Clustered regression control/Clustered regression/PCR/mm



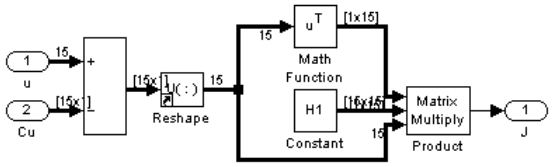
The *mm* block includes the PCR (*Signal estimates*) and cost calculation subblocks. The local regression model is given by the matrix and vector constants correspondingly to Table 3.2.

Clustered regression control/Clustered regression/PCR/ mm/Signal estimates



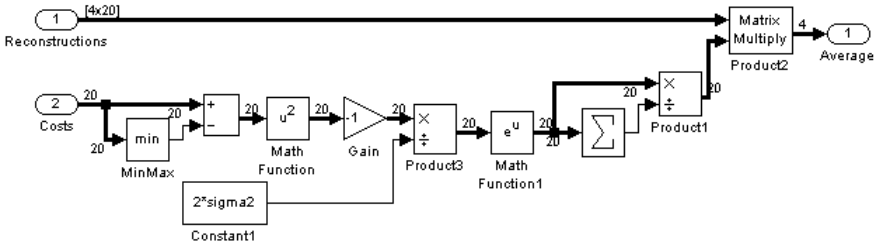
The PCR estimate is calculated using the data variances and means defined in the *Clustered regression* block parameters. For more information see [2] or [3].

Clustered regression control/Clustered regression/PCR/ mm/Cost calculation



The cost corresponding to the estimate depends on the distance from the input point u to the center of the cluster C_u. The *Clustered regression control* block parameter `H1` is used in this subsystem.

Clustered regression control/Clustered regression/ Calculate average



The average of the local estimates is a weighted sum of the local PCR model estimates. The weights are calculated using the corresponding cost values.

Appendix C

Matlab codes

C.1 bipedparams.m

This file defines the default parameters for the *Biped model* block.

```
% Parameters for the Biped model block
%
% Olli Haavisto, 2004

% sample time
st = 0.01;

% robot link lengths (m)
l0 = 0.8; % torso
l1 = 0.5; % thigh
l2 = 0.5; % shank
% center of mass distances (m)
r0 = l0/2; % torso (from hip)
r1 = l1/2; % thigh (from hip)
r2 = l2/2; % shank (from knee)
% link masses (kg)
m0 = 5; % torso
m1 = 2; % thigh
m2 = 1; % shank
% fill in the structure
robot.l = [l0, l1, l2];
robot.r = [r0, r1, r2];
robot.m = [m0, m1, m2];

clear l0 l1 l2 r0 r1 r2 m0 m1 m2

% acceleration of gravity (m/s^2)
g = 9.81;

% initial state
x00 = 0;        % (m)
y00 = 1.3749;   % (m)
```

```
a0  = -0.0851;  % (rad)
bl0 = -0.1084;  % (rad)
br0 = 0.2075;   % (rad)
cl0 = 0.2096;   % (rad)
cr0 = 0.2016;   % (rad)

initstate.coordinates = [x00, y00, a0, bl0, br0, cl0, cr0];

% initial speeds
px00 = 0.4867;  % (m/s)
py00 = -0.1135; % (m/s)
pa0  = -0.3789; % (rad/s)
pbl0 = -0.7304; % (rad/s)
pbr0 = -1.0236; % (rad/s)
pcl0 = 0.6357;  % (rad/s)
pcr0 = 0.0386;  % (rad/s)

initstate.speeds = [px00, py00, pa0, pbl0, pbr0, pcl0, pcr0];

clear x0 y0 a0 bl0 br0 cl0 cr0 px0 py0 pa0 pbl0 pbr0 pcl0 pcr0

% ground force calculation initial state
initstate.gcstate = [0,1,0,1];

% ground shape
% - row 1: x-coordinates
% - row 2: y-coordinates
groundp.ground = [-5:0.1:1000];
groundp.ground = [groundp.ground;zeros(size(groundp.ground))];

% ground parameters
% - normal direction
groundp.ky = 10000; % elastic constant
groundp.by = 500; % damping ratio
% - tangential direction
groundp.kx = 10000; % elastic constant
groundp.bx = 500; % damping ratio
% friction coefficients
groundp.muk = 0.6; % kinetic
groundp.mus = 1.2; % static

% knee stopper parameters
knees.kk = 1000; % elastic constant
knees.bk = 100; % damping ratio

% maximum and minimum knee angles (rad)
knees.max = 160*pi/180;
knees.min = 0;
```

C.2 pdparams.m

This file defines the default parameters for the *PD control* block. Note that the reference updating logic parameters are set inside the block.

```
% Parameters for the PD control block
%
% Olli Haavisto, 2004

% initial reference
ref = [0.02, 0.3261, 0.0314, 0.2199]';

% robot link lengths
l0 = robot.l(1); % torso
l1 = robot.l(2); % thigh
l2 = robot.l(3); % shank
% mass center locations
r0 = robot.r(1); % torso (from hip)
r1 = robot.r(2); % thigh (from hip)
r2 = robot.r(3); % shank (from knee)

% sample time
st = 0.01;

% thighs
Pb1=60;     % both legs on the ground
Pb2=70;     % one leg on the ground
Pb3=Pb2;
Pb4=2.5;    % neither of the legs on the ground
% knees
Pc1=40;     % both legs on the ground
Pc2=30;     % only this leg on the ground
Pc3=10;     % only the other leg on the ground
Pc4=2.55;   % neither of the legs on the ground
% torso
Ph1=40;
Ph2=40;
Ph3=Ph2;
Ph4=2.5;

P=[Pb1, Pb2, Pb3, Pb4;      % db
    Pc1, Pc2, Pc3, Pc4;     % cl
    Pc1, Pc3, Pc2, Pc4;     % cr
    Ph1, Ph2, Ph3, Ph4];    % head

clear Pb1 Pb2 Pb3 Pb4 Pc1 Pc2 Pc3 Pc4 Ph1 Ph2 Ph3 Ph4

% thighs
Db1=1;      % both legs on the ground
Db2=6;      % one leg on the ground
Db3=Db2;
Db4=0.25;   % neither of the legs on the ground
% knees
Dc1=0.5;    % both legs on the ground
```

```
Dc2=2;      % only this leg on the ground
Dc3=0.1;    % only the other leg on the ground
Dc4=0.05;   % neither of the legs on the ground
% torso
Dh1=2;
Dh2=2;
Dh3=Dh2;
Dh4=0.4;

D=[Db1, Db2, Db3, Db4;      % db
    Dc1, Dc2, Dc3, Dc4;     % cl
    Dc1, Dc3, Dc2, Dc4;     % cr
    Dh1, Dh2, Dh3, Dh4];    % head

clear Db1 Db2 Db3 Db4 Dc1 Dc2 Dc3 Dc4 Dh1 Dh2 Dh3 Dh4
```

C.3 `teach.m`

This function calculates the clustered regression model structure. The only required parameter is the sample data structure (`data`), which is used as teaching data for the model. The data structure variable should have at least the fields `state` and `control` similar to the data structure saved using the GUI `save` button. The rest of the parameters are optional and define the properties of the clustered regression structure and amount of data used. Note that by default some data from the beginning is ignored.

```
function [model, dataprop] = teach(data, Nop, n, epochs, starti, stopi)
% [model, dataprop] = teach(data, Nop, n, epochs, starti, stopi)
%
% Calculates the clustered regression model for the clustered regression
% controller Simulink block.
% Parameters:
% - data: sample data saved from the simulator GUI
% - Nop: number of operating points (clusters) (optional, default=20)
% - n: number of latent variables (optional, default=8)
% - epochs: number of iterations of the clusterization algorithm
%               (optional, default=10)
% - starti: first data index to be used (optional, default=2001)
% - endi: last data index to be used (optional, default=end)
% Returns;
% - model: clustered regression model structure
% - dataprop: input and output sample data means and variances
% Uses: preprocess.m, convertlr.m, operatingpoints.m, separate.m,
% calculatemodels.m
%
% Olli Haavisto, 2004

if nargin<6
    stopi = length(data.state);
end
if nargin<5
    starti = 2001;
```

```
end
if nargin<4
    epochs = 10;
end
if nargin<3
    n = 8;
end
if nargin<2
    Nop = 20;
end

% preprocess data and get the data properties (mean and variance)
[pdata, dataprop] = preprocess(data, starti, stopi);
% calculate operating point locations and clusterize data
[I, model] = operatingpoints(pdata, Nop, epochs);
% calculate the submodels
model = calculatemodels(pdata, model, I, n);
```

C.4 crcparams.m

This file defines the parameters used by the *Clustered regression control* block. First, a ready clustered regression structure and data properties structure are loaded from a file. Then the rest of the parameters for the clustered regression calculation are defined. The `initswitch` variable determines which leg is initially swinging.

```
% Parameters for the Clustered regression controller block
%
% Olli Haavisto, 2004

%%%%%%%% clustered regression parameters %%%%%%%%
% local model parameters ('dataprop' and 'model' structures)
load pdmodel.mat
% number of local models
Nop = length(model);
% averaging function width
sigma2 = 0.05;
% cost calculation weight
H1 = diag([ones(1, 6), zeros(1,7), 1, 1]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%% controller parameters %%%%%%%%%%%%%
% initially the left leg is swinging
initswitch = 1;
```

HELSINKI UNIVERSITY OF TECHNOLOGY CONTROL ENGINEERING LABORATORY

Editor: H. Koivo

Report 126 Kaartinen, J.
Data Acquisition and Analysis System for Mineral Flotation. October 2001.

Report 127 Ylén, J.-P.
Measuring, Modelling and Controlling the pH value and the Dynamic Chemical State. November 2001.

Report 128 Gadoura, I. A., Suntio, T.
Implementation of Optimal Output Characteristic for a Telecom Power Supply - Fuzzy-logic approach. April 2002.

Report 129 Elmusrati, M. S.
Power Control and MIMO Beamforming in CDMA Mobile Communication Systems. August 2002.

Report 130 Pöyhönen, S., Negrea, M., Arkkio, A., Hyötyniemi, H.
Comparison of Reconstruction Schemes of Multiple SVM's Applied to Fault Classification of a Cage Induction Motor. August 2002.

Report 131 Pöyhönen, S.
Support Vector Machines in Fault Diagnostics of Electrical Motors. September 2002.

Report 132 Gadoura, I. A.
Design of Robust Controllers for Telecom Power Supplies. September 2002.

Report 133 Hyötyniemi, H.
On the Universality and Undecidability in Dynamic Systems. December 2002.

Report 134 Elmusrati, M. S., Koivo, H. N.
Radio Resource Scheduling in Wireless Communication Systems. January 2003.

Report 135 Blomqvist, E.
Security in Sensor Networks. February 2003.

Report 136 Zenger, K.
Modelling, Analysis and Controller Design of Time-Variable Flow Processes. March 2003.

Report 137 Hasu, V.
Adaptive Beamforming and Power Control in Wireless Communication Systems. August 2003.

Report 138 Haavisto, O., Hyötyniemi, H.
Simulation Tool of a Biped Walking Robot Model. March 2004.

ISBN 951-22-7018-8

ISSN 0356-0872

Picaset Oy, Helsinki 2004