On correction prediction in man and robot using the cerebellar model articulation controller
Master thesis

J. K. van Duijneveldt

September 16, 2019

Abstract

The subject of this thesis is to investigate whether the Cerebellar Model Articulation Controller (CMAC) can be used to anticipate controller corrections and increase performance by reducing delays in humanoid robots. This question can be divided into two subquestions. Firstly, whether the CMAC is a suitable architecture for the prediction of controller actions for a humanoid soccer robot. Using a 2D model of a robotic leg, the results of this thesis show that the CMAC can indeed learn to anticipate a corrective control signal 30ms ahead. Secondly, whether the architecture of the aforementioned setup can increase the performance of adequately passing a ball by reducing delays. The experiments show that the use of a CMAC can increase the performance of the robotic setup.

Keywords: (Artificial) Cerebellum, delay reduction, Cerebellar Model Articulation Controller (CMAC), Purkinje Cell.

Contents

1 Introduction

Humanoid robots are increasingly used in a wide range of situations. A special branch of sports are humanoid robots playing soccer [Missura and Behnke 2015]. Yet, robots are still commonly associated with slow and rough movement. This in opposition with the human body. Skilled athletes perform fast and very fluid motions. This competence can be largely attributed to the human cerebellum.

The complete planning and execution of a movement however, involves many different parts of the brain. The global planning of the movement is mainly governed by the prefrontal cortex. This process is mediated by the basal ganglia, which gauge the expected reward and mediate learning of adequate motion sequences. The task is broken down into a sequence of short actions as the information passes through the premotor cortex to the motor cortex. The motor cortex produces the impulses required for the specific movements. Finally, the cerebellum receives a copy of the efferent impulses. The cerebellum modifies and tweaks these impulses based on knowledge acquired during previous experiences, combined with measurements of the current situation, while executing the motion sequence [Kolb and Whishaw 2008].

To investigate whether this property is transferable to robotics, we first need know more about the functions and structure of the cerebellum in the human body.

The cerebellum is a part of the brain present in almost all vertebrates. In humans it is located at the lower back of the head. It derives its name from Latin, literally meaning little brain. Despite being generally smaller than the cortex (accounting for only ten percent of the brain volume in humans), its name detracts from its capabilities. Even with this small volume, the cerebellum contains eighty percent of all neurons in the brain [Herculano-Houzel 2009]. In the words of neuroscientist Richard Bergland, "We don't know exactly what the cerebellum is doing. But whatever it's doing, it's doing a lot of it."

Of course, with the current possibilities for research, we gained new information about what the cerebellum does. The cerebellum is primarily involved in motor control and more specifically in tasks requiring a certain amount of skill. Highly skill demanding tasks are mostly easy to learn but hard to master. Typical examples of cerebellar involvement are the mastery of techniques in sports and proficiency with musical instruments. Both examples display the cerebellums involvement in muscle memory and fine motor control. However, the functions of the cerebellum span a broader area in human behavior. For example, maintaining balance or more complex functions such as rhythmic perception and the regulation of higher cognition [Caligiore et al. 2017].

The difference between the cortex and the cerebellum is characterized by their contribution in the execution of a motion. Whereas the cortex dictates each movement to be performed, the cerebellum fine-tunes the planned motion trajectories and coordinates the groups of muscles required to realize the mo-

tion. In this sense, the cerebellum may be seen as the orchestra conductor in the brain, conducting the body, assuring each individual movement is performed in harmony to produce a fluid and precise motion by fine-tuning each movement and coordinating all involved muscles.

The literature report [Duijneveldt 2018] concludes that the cerebellum seems to be a promising system from a roboticists point of view. *The cerebellum is equipped with to neurological hardware to perform the following tasks:*

- Predict future events (and their timing) created by either the environment, or the creatures own actions [Bell, Han, and Sawtell 2008; Ebner and Pasalar 2008; Miall et al. 1993].
- Recognize patterns through the Purkinje cells, modulated by the molecular layer interneurons and Golgi cells [Albus 1975; Bell, Han, and Sawtell 2008].
- Undergo supervised learning from the cortex [Caligiore et al. 2017].
- Adapt to a teaching signal with long delays (80 ms) and wide variation (σ = 180 ms) [Safo and Regehr 2008].

The functions above are very promising to make movements of robots more responsive and fluid. This would be especially valuable in the robotic equivalent of sport. For example, a soccer robot from the humanoid league Robocup [Missura and Behnke 2015], where planned approaches change quickly and often.

1.1 Aim of the study

With the acquired knowledge of the cerebellum and its function in humans, the following question emerges:

Can the artificial cerebellum be used to anticipate controller correction and increase the performance of a soccer robot by reducing delays?

This research question can be broken down into two sub-question. The first question is to investigate whether an artificial cerebellum is a suitable architecture for the prediction of controller corrections at all. The second question is more specific to the given situation. Its goal is to investigate whether the aforementioned setup can increase the performance of passing the ball by reducing delays. In order to answer these questions, the following hypotheses are formulated:

- The first hypothesis tested in this thesis is that the cerebellum is a suitable architecture for the effective prediction of controller actions.
- The second hypothesis tested in this thesis is that an artificial cerebellum can increase performance by reducing delays.

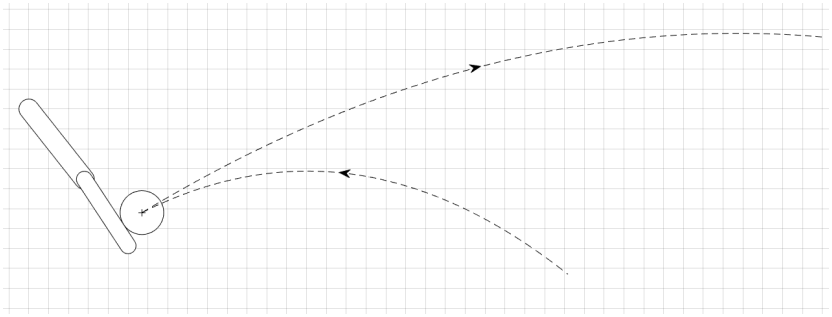

Figure 1: The robot receives the ball and kicks it back to a target located at 10m.

1.2 Research design

In order to test the hypotheses, a controlled environment needs to be established. The test setup consists of a simulated robot leg which task it is to kick an incoming ball to a fixed target located at a distance of 10 m from the robot (see figure 1). This test environment is the result of the following criteria:

- A simulated environment will be used to facilitate communication between the different setup components and to reduce the cost of the research.
- The setup must contain both kinematic and dynamic components to ensure a sufficiently complex system to discern learning.
- A disturbance must be introduced to the setup to provide a cue and incentive to react and learn.
- Appropriate delays must be included to represent reaction times. In a real situation, latencies can originate from sensors, actuators and controller computation time.
- The artificial cerebellum must learn to anticipate controller corrections through supervised learning. This is necessary to test the first hypothesis.
- The setup must produce a measurable performance. This is necessary to compare the situations with and without artificial cerebellum, and thereby test the second hypothesis.

1.3 Materials

These criteria have led to the following considerations. The desired controlled environment can be created in MATLAB. This is a well established programming language in the academic environment. It is a versatile framework which allows to simulate dynamics with one of its dedicated packages Simulink.

The initial plan for the test setup was to simulate a robot learning to walk towards an approaching ball and kick it to the target. This required a robot with five degrees of freedom. On initial attempts, teaching the robot to walk and to

kick a ball proved to be too complex. As a result, the setup was changed to a 2D robot fixed at the right place to kick the ball. The number of degrees of freedom was hereby reduced from five to two: the first degree of freedom is the rotational joint equivalent to the human hip. The second degree of freedom corresponds to the human knee. The objective, to kick the ball to a target placed at 10 meters from the robot, remains the same. The model and its implementation are further discussed in chapter 2.

1.4 Procedure and thesis organization

To test the hypotheses, the experiments are broken down to separate tasks. The tasks as described below can be used as a summary of each step and as a reading guide to this thesis.

The first step is to model the robot setup in MATLAB. This requires a more precise definition of the setup which is elaborated in chapter 2. From this setup, the equations of motion are derived using Kane's method [Kane and Wang 1965].

To control the robot setup, two different control schemes are implemented. The first controller plans the trajectories of the hip and knee joints. The necessary torques are calculated using inverse kinematics using the derived equations of motion. A second controller becomes active when the assumed trajectory of the ball is found to be incorrect. In this case, the initial planned trajectory for the hip and knee proves to be suboptimal. The corrective action of the second controller is therefore added to the original signal.

Once the control schemes are implemented, the setup can be tested. During this test, the robot tries to kick the ball to the target. However, shortly before the leg impacts with the ball, the position of the ball is changed in varying degrees. This allows the robot only a short time to compensate and adjust its planned trajectory. As a result, the robot may not fully succeed to kick to ball to its target. The resulting distances from the target serve as a control test to compare the situation with the artificial cerebellum to. Additionally, it shows the sensitivity of the setup to a disturbance in ball position.

As a first experiment, an artificial cerebellum is partially implemented and trained using the correction control scheme. During this experiment, the artificial cerebellum is trained off-line and does not yet provide a control signal to the actuators of the robotic leg. The artificial cerebellum is trained on 1 000 randomly generated initial conditions. For each initial condition, 10 different lateral disturbances are evaluated, resulting in a total of 10 000 trials. These trials are randomly permuted during training. For each trial the setup state over time is recorded as well as the output of the correction controller. After each trial, the artificial cerebellum is trained to reproduce the output of the correction controller using the setup state delayed by 30 ms as input. This way, the artificial cerebellum should produce a predicted correction controller action

30 ms ahead when its input (the setup state) is no longer delayed.

An evaluation of the first hypothesis can now be made. To evaluate the learning capabilities of the cerebellum, the Root Mean Square (RMS) error between the predicted and actual controller action is evaluated around the kick event. A low RMS error indicates that the artificial cerebellum correctly predicts the controller action

The pre-trained artificial cerebellum is then fully implemented in the setup during the second experiment. The output generated by the artificial cerebellum is added to the control signals sent to the actuator of the robotic leg. Since the input (setup state) to the artificial cerebellum is not delayed during the trials, the artificial cerebellum should predict the controller corrections 30 ms ahead. The initial conditions are divided into 900 training conditions and 100 validation conditions with 10 different lateral disturbances each.

Finally, a comparison can be made between the control test without the effect of the artificial cerebellum and the results of the second experiment which does include the effect of the artificial cerebellum. Since both the control test and the second experiment use the same initial conditions and lateral disturbances, the effect of the artificial cerebellum of the performance of the robotic leg can be evaluated by comparing the distances the ball is kicked in both situations.

2 Model

2.1 Overview and definition

The setup is reduced to a single leg with a hip and knee joint. See figure 2. The simulation to test the hypothesis is a ball kicking robot consisting of:

- A robot leg with two bodies parts: an upper leg and a lower leg. The upper leg is connected to the fixed global frame by a revolute joint. Its position is measured by the angle α measure in positive direction from the anatomical position. In anatomical description, a positive angular velocity $\dot{\alpha}$ would be equivalent to hip flexion. The lower leg is connected to the upper leg in knee by a revolute joint. Its angle β is defined in positive direction relative to the upper leg (see fig. 2). A positive angular velocity $\dot{\beta}$ would be equivalent to knee extension.

- The football is modeled as a disk with three degrees of freedom in the same plane as the robot leg.

The goal of the setup is for the robot leg to kick the ball to a target distance d_t of 10 meters. The incoming trajectories of the ball are different for each trial. The trajectories are disturbed just before impact, requiring a correction in the planned motion of the leg.

The reaction time available for the correction is limited and in some cases insufficient to fully correct the plan and reach the targeted distance. The task of the cerebellar controller is to learn from the previously executed corrections and react preemptively to the change of information.

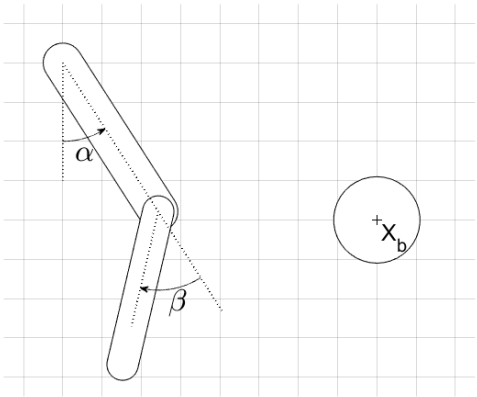

Figure 2: Definition of angles of the hip and knee joint.

2.2 Assumptions and values

Numerical values are required for the simulation. Some are based on real life situations, some are estimated guesses. The table below lists all the used values

Variable	Meaning	Value	Unit	Source
α	Angle of the thigh of the robot	-	rad	Variable
β	Angle of the shank of the robot	-	rad	Variable
$\dot{\alpha}$	Angular velocity of the thigh	-	rad.s^{-1}	Variable
$\dot{\beta}$	Angular velocity of the shank	-	rad.s^{-1}	Variable
g	Gravity	9.80665	m.s^{-2}	Literature[1]
g_i	Convective acceleration	-	-	Derived
$\mathbf{q}$	Generalized coordinates	-	-	Derived
$\mathbf{M}$	Mass matrix	-	-	Derived
d_t	Target distance of the kick	10	m	Arbitrary
l_1	Length of the upper leg	0.45	m	Literature[2]
l_2	Length of the lower leg	0.40	m	Literature[2]
m_1	Mass of the upper leg	8.4	kg	Literature[2]
m_2	Mass of the lower leg	3.8	kg	Literature[2]
I_1	Rotational inertia of the upper leg	0.2031	kg.m^2	Derived
I_2	Rotational inertia of the lower leg	0.0700	kg.m^2	Derived
$T_{1,max}$	Maximum hip torque	309.2	N.m	Literature[3]
$T_{2,max}$	Maximum knee torque	129.9	N.m	Literature[3]

[1] Default setting for MATLAB 2018a Simscape mechanism configuration
[2] Plagenhoef, Gaynor Evans, and Abdelnour 1983
[3] Nunome et al. 2006

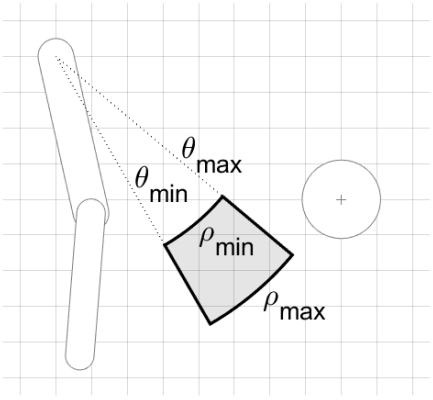

Figure 3: Region of possible locations of the ball at the time of the kick.

2.2.1 Initial conditions for the ball

The x and y-coordinates of the ball are defined such that the ball starts from the ground in front of the leg. The y coordinate is defined such that the ball touches the ground. Deformations of the ball are neglected. $y_0 = r_{ball}$. The x coordinate is selected randomly from a uniform distribution between 4 m and 10 m ahead of the hip joint. These two coordinates uniquely define a first point of the parabola.

To assure a valid parabola, the ball must pass through a region where the leg can reach it. A second point is selected from a section of the disk centered around the hip such that in polar coordinates, with the hip as origin,$\theta_{min} \leq \theta \leq \theta_{max}$ and $\rho_{min} \leq \rho \leq \rho_{max}$. An illustration of this region is shown in figure 3.

To assure a uniform distribution over the area, the probability distribution function (pdf) of ρ should take into account the changing length of the arc of radius ρ. More specifically, since the length of the arc is proportional to the radius ρ, the probability density must also be proportional to the radius.

$$pdf(\rho) = k\rho, \text{ with } k \in \mathbb{R}^* \text{ and } \rho \in [\rho_{min}, \rho_{max}] \tag{1}$$

The constant k can be calculated by using the property of the probability density function that the integral over its domain is equal to one:

$$\int_{-\infty}^{\infty} pdf(\rho) \cdot d\rho = \int_{\rho_{min}}^{\rho_{max}} k \cdot \rho d \cdot \rho = \frac{1}{2} \cdot k \cdot (\rho_{max}^2 + \rho_{min}^2) = 1 \tag{2}$$

$$k = \frac{2}{\rho_{max}^2 + \rho_{min}^2}$$

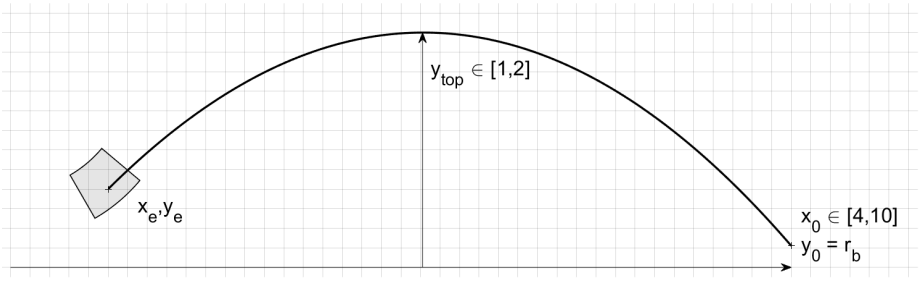

Figure 4: Example of a valid parabolic trajectory of the ball.

To generate random values for ρ such that its distribution coincides with the function established in equation (2), the inverse probability integral transform can be applied.

Theorem: If X has a uniform distribution on $[0, 1]$ and if Y has a cumulative distribution F_Y, then the random variable $F_Y^{-1}(X)$ has the same distribution as Y [Morgan and Devroye 1988]

Therefore, to sample ρ randomly according to equation (2), we calculate the cumulative distribution function (cdf) of ρ and apply its inverse to the uniform distribution $U(0, 1)$.

$$cdf(\rho) = \int_{-\infty}^{\rho} pdf(\rho) \cdot d\rho = \int_{\rho_{min}}^{\rho} k \cdot \rho \cdot d\rho = \frac{\rho^2 - \rho_{min}^2}{\rho_{max}^2 - \rho_{min}^2} \tag{3}$$

$$\rho_{random} \sim cdf^{-1}(U(0,1)) = \sqrt{(\rho_{max}^2 - \rho_{min}^2) \cdot U(0,1) + \rho_{min}^2} \tag{4}$$

Converting polar to Cartesian coordinates yields:

$$x_e = \rho \cdot \cos(\theta) \text{ and } y_e = \rho \cdot \sin(\theta) + 1 \tag{5}$$

This defines a second point on the parabola, which is still insufficient. For a unique definition, a third constraint can be added as the peak height of the ball. Selecting a peak height y_{peak} between 1 m and 2 m, we can assure the ball reaches its peak before entering the kick region. This ensures a negative vertical velocity at impact.
With the maximum height of the parabola known, the initial vertical velocity can be calculated by equating the initial vertical kinetic energy to the potential energy at the peak.

$$\frac{1}{2} \cdot m \cdot v_{y0}^2 = m \cdot g \cdot (y_{peak} - y_0) \implies v_{y0} = \sqrt{2 \cdot g \cdot (y_{peak} - y_0)} \tag{6}$$

To calculate the initial horizontal velocity, the time difference between initial position and the kick must be known. Since the vertical velocity at the top is zero by definition this is easily calculated:

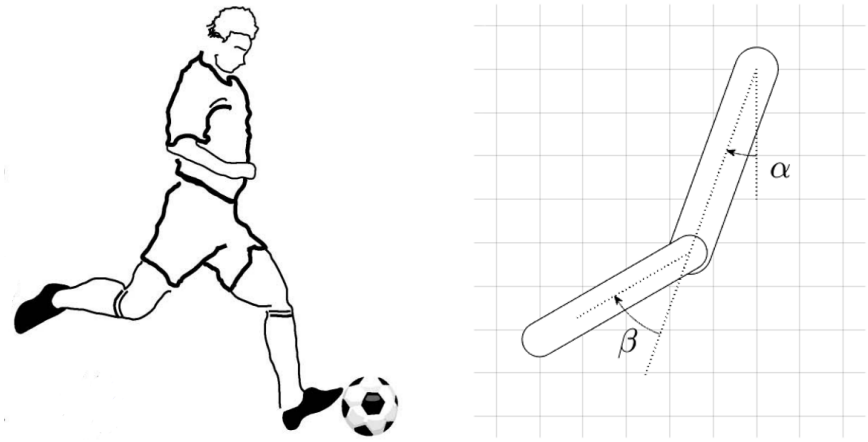

Figure 5: Left side: An illustration of an instep kick as performed by a human. Right side: The initial conditions of the robotic leg during the simulations. The initial angle of the hip α is equal to -20 degrees. The initial angle of the knee β is equal to -40 degrees.

$$\Delta t = \Delta t_{0-top} + \Delta_{top-e} = \sqrt{\frac{2 \cdot (y_{top} - y_0)}{g}} + \sqrt{\frac{2 \cdot (y_{top} - y_e)}{g}} \tag{7}$$

Since there are no forces in the horizontal direction, the initial horizontal velocity is simply the distance over time.

$$v_{x0} = \frac{x_e - x_0}{\Delta t} \tag{8}$$

2.2.2 Initial conditions of the leg

The initial position of the leg is chosen to be fixed for all trials. The angles are chosen arbitrarily to represent a position ready to kick, see figure 5. The hip is extended by an angle of 20 degrees from the anatomical position. The knee joint is flexed by an angle of 40 degrees from the anatomical position. Note that this results in negative values for the angles α and β.

The robotic leg starts from a standstill. Therefore the angular velocity of the hip $\dot{\alpha}$ and the angular velocity of the knee $\dot{\beta}$ are both equal to 0 rad.s^{-1}.

2.3 Kinematics

To simulate the motion of the robot leg, the system must be converted to a set of equations to be integrated by the solver.

2.3.1 Equations of motion

The equations of motion required to simulate the movements of the robot and the ball are derived using Kane's method [Kane and Wang 1965] as explained below. The reasons to choose this method are twofold. First of all, this method reduces the amount of coordinates to a strict minimum: the number of degrees of freedom. This eliminates configurations which would violate joint constraints. This increases the reliability of the simulations as there is no need to correct for drift in the constraints. Secondly, the structure of the derivation of the equations of motion is very straight forward. This allows for easier simplification of the equations which reduces the complexity the calculations.

Derivation of the equations: If the mass of the system is constant, Newton's second law of motion can be written in vector form as:

$$\mathbf{M}\ddot{x}_i - f_i = 0 \tag{9}$$

Where $\mathbf{M}$ is the mass matrix, x_i the global coordinates of the centers of mass and f_i the sum of external forces. Alternatively, using virtual velocities:

$$\delta\dot{x}_i^T(\mathbf{M}\ddot{x}_i - f_i) = 0 \tag{10}$$

From this point we want to express this equation in terms of generalized coordinates q_j. To do so, we must express the global coordinates x_i in terms of generalized coordinates q_j:

$$x_i = F_i(q_j) \tag{11}$$

We do the same for the velocity and acceleration:

$$\dot{x}_i = \frac{\partial F_i}{\partial q_j}\dot{q}_j = F_{i,j}\dot{q}_j \tag{12}$$

$$\ddot{x}_i = \frac{\partial F_i}{\partial q_j}\ddot{q}_j + \frac{\partial^2 F_i}{\partial q_j \partial q_k}\dot{q}_j\dot{q}_k = F_{i,j}\ddot{q}_j + F_{i,jk}\dot{q}_j\dot{q}_k \tag{13}$$

Substitution in equation (10) yields:

$$(F_{i,j}\delta q_j)^T(\mathbf{M}(F_{i,j}\ddot{q}_j + F_{i,jk}\dot{q}_j\dot{q}_k) - f_i) = 0 \tag{14}$$

Since the choice of virtual velocity is arbitrary, we obtain a set of equations:

$$F_{i,j}^T(\mathbf{M}(F_{i,j}\ddot{q}_j + F_{i,jk}\dot{q}_j\dot{q}_k) - f_i) = 0 \tag{15}$$

The vector $F_{i,jk}\ \dot{q}_j\dot{q}_k$ is often called convective acceleration and denoted g_i. Substituting and rearranging the terms yields the following equation:

$$F_{i,j}^T\mathbf{M}F_{i,j}\ddot{q}_j = F_{i,j}^T(f_i - \mathbf{M}g_i) \tag{16}$$

We can express this in reduced notation:

$$\bar{\mathbf{M}}\ddot{\mathbf{q}} = \bar{\mathbf{f}} \tag{17}$$

Where :

- $\bar{\mathbf{M}}$ is the reduced mass matrix $F_{i,j}^T \mathbf{M} F_{i,j}$.
- $\bar{\mathbf{f}}$ is the reduced force vector $F_{i,j}^T (f_i - \mathbf{M} g_i)$.
- $\ddot{\mathbf{q}}$ is the vector of generalized accelerations.

Application to the real system: We apply these to formulas the system as defined previously. A very important matrix in this method is the transformation matrix. It is the matrix containing all partial derivatives of the global coordinates x_i with respect to the generalized coordinates q_j:

$$F_{i,j} = \begin{bmatrix} \frac{\partial F_1}{\partial \alpha} & \frac{\partial F_1}{\partial \beta} \\ \frac{\partial F_2}{\partial \alpha} & \frac{\partial F_2}{\partial \beta} \\ \frac{\partial F_3}{\partial \alpha} & \frac{\partial F_3}{\partial \beta} \\ \frac{\partial F_4}{\partial \alpha} & \frac{\partial F_4}{\partial \beta} \\ \frac{\partial F_5}{\partial \alpha} & \frac{\partial F_5}{\partial \beta} \\ \frac{\partial F_6}{\partial \alpha} & \frac{\partial F_6}{\partial \beta} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} l_1 \cos(\alpha) & 0 \\ \frac{1}{2} l_1 \sin(\alpha) & 0 \\ 1 & 0 \\ \frac{1}{2} l_2 \cos(\alpha + \beta) + l_1 \cos(\alpha) & \frac{1}{2} l_2 \cos(\alpha + \beta) \\ \frac{1}{2} l_2 \sin(\alpha + \beta) + l_1 \sin(\alpha) & \frac{1}{2} l_2 \sin(\alpha + \beta) \\ 1 & 1 \end{bmatrix} \tag{18}$$

This allows us to calculate the reduced mass matrix.

$$\bar{\mathbf{M}} = F_{i,j}^T \mathbf{M} F_{i,j} = \begin{bmatrix} \mathbf{M}1 & \mathbf{M}2 \\ \mathbf{M}2 & \mathbf{M}3 \end{bmatrix} \tag{19}$$

Where:

$$\mathbf{M}1 = I_1 + \tfrac{1}{4} m_1 l_1^2 + I_2 + m_2 (l_1^2 + \tfrac{1}{4} l_2^2 + l_1 l_2 \cos(\beta)) \tag{20}$$

$$\mathbf{M}2 = I_2 + \tfrac{1}{4} m_2 l_2^2 + \tfrac{1}{2} m_2 l_1 l_2 \cos(\beta) \tag{21}$$

$$\mathbf{M}3 = I_2 + \tfrac{1}{4} m_2 l_2^2 \tag{22}$$

To obtain this expression, the trigonometric simplifications from (66) to (68) are used. The transformation matrix is also needed to calculate the convective accelerations:

$$g_i = \begin{bmatrix} \frac{\partial F_{1,1} \dot{\alpha}}{\partial \alpha} & \frac{\partial F_{1,2} \dot{\beta}}{\partial \beta} \\ \frac{\partial F_{2,1} \dot{\alpha}}{\partial \alpha} & \frac{\partial F_{2,2} \dot{\beta}}{\partial \beta} \\ \frac{\partial F_{3,1} \dot{\alpha}}{\partial \alpha} & \frac{\partial F_{3,2} \dot{\beta}}{\partial \beta} \\ \frac{\partial F_{4,1} \dot{\alpha}}{\partial \alpha} & \frac{\partial F_{4,2} \dot{\beta}}{\partial \beta} \\ \frac{\partial F_{5,1} \dot{\alpha}}{\partial \alpha} & \frac{\partial F_{5,2} \dot{\beta}}{\partial \beta} \\ \frac{\partial F_{6,1} \dot{\alpha}}{\partial \alpha} & \frac{\partial F_{6,2} \dot{\beta}}{\partial \beta} \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} l_1 \dot{\alpha}^2 sin(\alpha) \\ \frac{1}{2} l_1 \dot{\alpha}^2 cos(\alpha) \\ 0 \\ -l_1 \dot{\alpha}^2 sin(\alpha) - \frac{1}{2} l_2 \left(\dot{\alpha} + \dot{\beta}\right)^2 sin(\alpha + \beta) \\ l_1 \dot{\alpha}^2 cos(\alpha) + \frac{1}{2} l_2 \left(\dot{\alpha} + \dot{\beta}\right)^2 cos(\alpha + \beta) \\ 0 \end{bmatrix} \tag{23}$$

The only external force on the system is gravity acting in negative y-direction:

$$f_i = \begin{bmatrix} 0 \\ -m_1 g \\ 0 \\ 0 \\ -m_2 g \\ 0 \end{bmatrix} \tag{24}$$

Again using the simplifications from (66) to (68) we obtain:

$$\bar{\mathbf{f}} = F_{i,j}^T (f_i - \mathbf{M} g_i) =$$

$$\begin{bmatrix} l_1 l_2 m_2 \sin(\beta)\dot{\beta}(\frac{1}{2}\dot{\beta} + \dot{\alpha}) + g l_1 \sin(\alpha)(\frac{1}{2}m_1 + m_2) + \frac{1}{2} g l_2 m_2 \sin(\alpha + \beta) \\ \frac{1}{2} l_2 m_2 (g \sin(\alpha + \beta) - l_1 \sin(\beta)\dot{\alpha}^2) \end{bmatrix} \tag{25}$$

Finally, to obtain the complete equation of motion for the system, the expression for the reduced mass matrix (19) and the expression for the generalized force vector (25) are substituted in the reduced equation of motion (17). The resulting expression can be used directly for integration.

2.3.2 Forces acting of the system

Using the equations as derived above would simulate the robotic leg as if it were a double pendulum subjected to gravity. To simulate a robotic leg kicking a ball several more forces need to be added to the equations such as: motor torques actuating the joints, interaction forces between the robot and ball upon contact, and joint limits. These additional forces can be added to the system by adjusting equation (24), describing the external forces acting on the system:

$$f_i = \begin{bmatrix} 0 \\ -m_1 g \\ 0 \\ 0 \\ -m_2 g \\ 0 \end{bmatrix} + \begin{bmatrix} F_{x_1} \\ F_{y_1} \\ T_1 \\ F_{x_2} \\ F_{y_2} \\ T_2 \end{bmatrix} \tag{26}$$

Where: F_{x_1} is the force in x-direction acting on the center of mass (CoM) of the upper leg, F_{y_1} is the force in y-direction acting on the CoM of the upper leg, T_1 is the torque acting on the CoM of the upper leg, F_{x_2} is the force in x-direction acting on the CoM of the lower leg, F_{y_2} is the force in y-direction acting on the CoM of the lower leg, and T_2 is the torque acting on the CoM of the lower leg.

Joint limits: The range of motion of both the hip and knee are delimited by an upper and lower angle. To prevent the joints from extending or flexing beyond a reasonable region, a stiffness and damping system is introduced when the joint angles exceed their respective bounds. The torques applied to the hip joint T_{hip} and to the knee joint T_{knee} are defined as follows:

$$T_{hip} = \begin{cases} -k_1(\alpha - \alpha_{max}) - c_1\dot{\alpha}, & \text{if } \alpha > \alpha_{max} \\ -k_1(\alpha - \alpha_{min}) - c_1\dot{\alpha}, & \text{if } \alpha < \alpha_{min} \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

$$T_{knee} = \begin{cases} -k_2(\beta - \beta_{max}) - c_2\dot{\beta}, & \text{if } \beta > \beta_{max} \\ -k_2(\beta - \beta_{min}) - c_2\dot{\beta}, & \text{if } \beta < \beta_{min} \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

Where k_1 and k_2 are the joint stiffnesses in N.m.rad^{-1} and c_1 and c_2 are the damping coefficients in N.m.rad^{-1}.s.

Ball interaction: An important part of the simulation is the interaction between the ball and the lower leg of the robot during the kick. This entails for a choice of contact interaction between the leg and ball to be made. Four typical approaches can be roughly sorted by the duration of contact. In increasing order of contact duration:

1. Instant impulse transfer between bodies. The calculations are halted when two bodies collide.
2. Introduction of penetration penalty
3. Introduction of constraint forces (lagrangian)
4. Reduction of degrees of freedom

The contact duration between the leg and ball is relatively short but non-negligible. In a real interaction, the football would deform, storing potential energy. This energy is released when the ball moves away from the leg. This is comparable to the introduction of a penetration penalty (approach 2.) which is therefore the selected approach for the contact forces. Instead of the ball deforming, a virtual force is introduced between the leg and ball. The virtual force is scaled linearly with the depth of penetration, similar to a spring. Additionally, viscous damping is introduced to account for the energy loss due to the deformation of the ball.

With a choice of contact interaction made, the virtual force acting on the ball and the leg is calculated according to the following steps:

1. The ball position and velocity are expressed in the frame of the lower leg.
2. If the ball is in contact with the leg, the virtual force is calculated as a spring damper system: $F_x = -k_b x - c_b \dot{x}$.
3. The contact force is transformed back to the global frame.
4. The contact force is added to equations of motion.

Since the robot is simulated in two dimensions, the coordinate transformations can be described using the special euclidean group SE(2). Any transformation from one frame to an other can be described by a rotation R and a translation c:

$$X' = RX + c \tag{29}$$

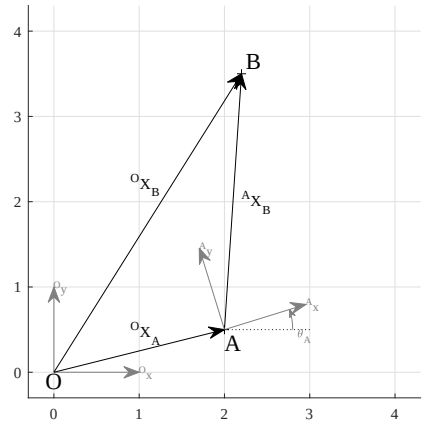

Figure 6: Transformation diagram

Where $R \in \mathbb{R}^{2\times 2}, \det R = 1, R^T R = I$ and $c \in \mathbb{R}^2$. These properties also allow the inverse transformation of X' into the original frame.

$$X = R^T X' - R^T c \tag{30}$$

Expression of the coordinates of B in the frame A.

$${}^A X_b = {}^A H_0 \, {}^0 X_b = R^T ({}^0 X_b - {}^0 X_A) \tag{31}$$

Expression of the velocity of B in the frame A.

$${}^A \dot{X}_b = \dot{R}^T ({}^0 X_b - {}^0 X_A) + R^T ({}^0 \dot{X}_b - {}^0 \dot{X}_A) \tag{32}$$

Transpose of rotation matrix is equal to the inverse.

$$R^{-1} = R^T = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{33}$$

The derivative of the transpose of the rotational matrix with respect to time.

$$\dot{R}^T = \begin{bmatrix} -\omega \sin(\theta) & \omega \cos(\theta) \\ -\omega \cos(\theta) & -\omega \sin(\theta) \end{bmatrix} \tag{34}$$

For simplification it is useful to note that:

$$\dot{R}^T R = \begin{bmatrix} -\omega \sin(\theta) & \omega \cos(\theta) \\ -\omega \cos(\theta) & -\omega \sin(\theta) \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} \tag{35}$$

Substitution in equation (32) yields:

$${}^A \dot{X}_b = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} {}^A X_b + R^T ({}^0 \dot{X}_b - {}^0 \dot{X}_A) \tag{36}$$

The equations (31) and (36) can be used to calculate the position and velocity of the ball with respect to the leg. Once the magnitude of the contact force between the ball and leg are determined, the force can be transformed back to the inertial reference frame using (31).

2.3.3 Control forces

To control the robot to kick the ball to the desired distance, an adequate signal must be provided to the actuators in the hip and knee joint. The actual torque provided by the actuators is a result of multiple controllers working together. A feedforward controller provides a scheduled torque profile for the hip and knee to follow a planned trajectory. A correction controller is added to provide an alternative course of action when a later disturbance to the ball is introduced. An artificial cerebellum is used to anticipate the action of the correction controller. Finally, a proportional derivative controller is used help the feedforward controller to remain on the planned trajectory.

Feed forward: The trajectory of the hip and knee joints are planned ahead along a third degree polynomial for both joints. This method is commonly called cubic interpolation. Assuming a third degree polynomial with coefficients a, b, c and d, we can express the angle of the hip α, its angular velocity $\dot{\alpha}$ and its angular acceleration $\ddot{\alpha}$ as a function of time t:

$$\begin{aligned} \alpha(t) &= at^3 + bt^2 + ct + d \\ \dot{\alpha}(t) &= 3at^2 + 2bt + c \\ \ddot{\alpha}(t) &= 6at + 2b \end{aligned} \tag{37}$$

A third degree polynomial has four degrees of freedom. This allows us to choose both the position and the velocity of two points in time. As shown in figure 7, the starting angle α_0 and velocity $\dot{\alpha}_0$ are given for the time t_0. The same holds for α_e and $\dot{\alpha}_e$ at t_e. Using the equations for the angle and angular velocity (37) this results in the following set of equations:

$$\begin{cases} \alpha_0 = \alpha(t_0) = at_0^3 + bt_0^2 + ct_0 + d \\ \alpha_e = \alpha(t_e) = at_e^3 + bt_e^2 + ct_e + d \\ \dot{\alpha}_0 = \dot{\alpha}(t_0) = 3at_0^2 + 2bt_0 + c \\ \dot{\alpha}_e = \dot{\alpha}(t_e) = 3at_e^2 + 2bt_e + c \end{cases} \tag{38}$$

The set of equation can be transformed into a non-homogeneous system of linear equations in a, b, c and d:

$$\begin{bmatrix} \alpha_0 \\ \alpha_e \\ \dot{\alpha}_0 \\ \dot{\alpha}_e \end{bmatrix} = \begin{bmatrix} t_0^3 & t_0^2 & t_0 & 1 \\ t_e^3 & t_e^2 & t_e & 1 \\ 3t_0^2 & 2t_0 & 1 & 0 \\ 3t_e^2 & 2t_e & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{39}$$

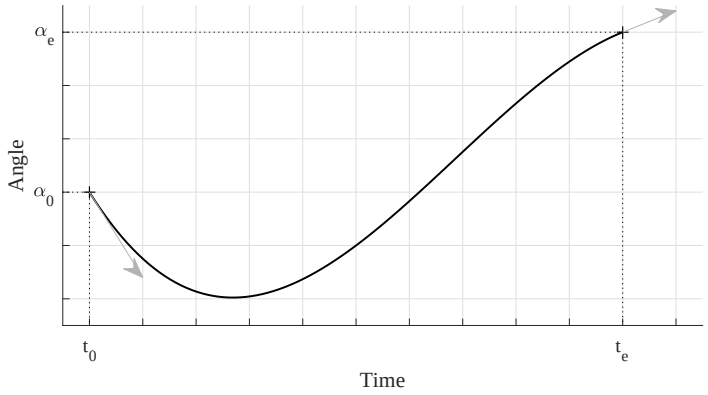

Figure 7: Exemple of a third degree polynomial with both angles (α_0 and α_e) and angular velocities ($\dot{\alpha}_0$ and $\dot{\alpha}_e$) prescribed in two points in time (t_0 and t_e).

Theorem: A $n \times n$ non-homogeneous system of linear equations has a unique non-trivial solution if and only if its determinant is non-zero.

Therefore, to verify that the system of equations (39) can be solved in a,b,c and d, we can simply calculate the determinant. Expanded down the forth column, its expressions simplifies as follows:

$$\begin{vmatrix} t_0^3 & t_0^2 & t_0 & 1 \\ t_e^3 & t_e^2 & t_e & 1 \\ 3t_0^2 & 2t_0 & 1 & 0 \\ 3t_e^2 & 2t_e & 1 & 0 \end{vmatrix} = -t_0^4 + 4t_0^3 t_e - 6t_0^2 t_e^2 + 4t_0 t_e^3 - t_e^4 = -(t_0 - t_e)^4 \qquad (40)$$

Since the starting time (t_0) is always different from kick time (t_e), there is always a trajectory along a third degree polynomial that satisfy the boundary conditions as defined in (38).

Using the same approach for the angle of the knee β, we can obtain the expressions of the desired angular accelerations $\ddot{\alpha}$ and $\ddot{\beta}$. Using inverse kinematics, the required torque on the hip and knee can be calculated. To do so, the equation of motion (16) must be expanded to include motor control as follows:

$$F_{i,j}^T \mathbf{M} F_{i,j} \ddot{\mathbf{q}}_{des} = F_{i,j}^T (f_i + \mathbf{C}\mathbf{T}_{FF} - \mathbf{M} g_i) \qquad (41)$$

Where $\mathbf{q}_{des}$ is the vector of desired angular accelerations as defined above, $\mathbf{C}$ is the control matrix and $\mathbf{T}_{FF}$ is the vector of feed forward control torques.

$$\ddot{\mathbf{q}}_{des} = \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{T}_{FF} = \begin{bmatrix} T_{hip} \\ T_{knee} \end{bmatrix} \tag{42}$$

Rearanging the terms of (41) and substituting the reduced notations (19) and (25) we get:

$$F_{i,j}^T \mathbf{C} \mathbf{T}_{FF} = \bar{\mathbf{M}} \ddot{\mathbf{q}}_{des} - \bar{\mathbf{f}} \tag{43}$$

This can be simplified by noting that:

$$F_{i,j}^T \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{44}$$

Resulting in the following equation for the torques:

$$\mathbf{T}_{FF} = \bar{\mathbf{M}} \ddot{\mathbf{q}}_{des} - \bar{\mathbf{f}} \tag{45}$$

Since $\bar{\mathbf{M}}$, is a function of α and β and $\bar{\mathbf{f}}$ is a function of α, β, $\dot{\alpha}$ and $\dot{\beta}$, the required torque for the hip and knee can be calculated.

Proportional derivative control: A proportional derivative (PD) controller is added to get back to the desired trajectory if the setup is disturbed, perturbing feed forward control. Disturbances may be the result of rounding errors during the integration of the equations of motion. Another source of disturbances could be noise from the artificial cerebellum introduced later. Since the role of the PD controller is to provide a small correction to get back on the planned trajectory, it is designed to be critically damped with a relatively long response time of τ=500 ms. For a damped harmonic oscillator of parameter x, the equation of motion is:

$$\ddot{x} + 2\zeta\omega_n \dot{x} + \omega_n^2 x = 0 \tag{46}$$

To obtain a critically damped system, the damping ratio ζ should be equal to one. Since the desired response time is also known (τ=500 ms), we can derive the natural frequency of the desired oscillator:

$$\omega_n = \frac{2\pi}{\tau} \approx 12.57 \text{ rad.s}^{-1} \tag{47}$$

In the robotic setup however, the states to be controlled (α and β), are not actuated independently, nor are they of unitary mass. Therefore, the action of the PD controller must be scaled accordingly. The proposed controller is implemented as follows, with the control matrix $\mathbf{C}$ as defined previously:

$$\mathbf{T}_{PD} = \mathbf{C}\bar{\mathbf{M}} \left(-2\zeta\omega_n \left(\dot{\mathbf{q}} - \dot{\mathbf{q}}_{des} \right) - \omega_n^2 \left(\mathbf{q} - \mathbf{q}_{des} \right) \right) \tag{48}$$

Perturbation correction attempt: During the second part of the experiments, the trajectory of the ball is altered just before the shin contacts the ball. This situation requires the addition of a perturbation correction controller. This controller is tasked to adjust the lower leg just before the ball is kicked.

The disturbance in the position of the ball is know only shortly before the kick. Therefore the perturbation correction controller has a limited time frame to react, and adjust the position and velocity of the lower leg. Since we want to measure the effect of the artificial cerebellum, the correction controller should be able to compensate for some, but not all possible lateral disturbances to the ball. For this reason the maximum magnitude of the correction controller is set to 10 Nm.

To maximize the effect of the correction controller, its output should always be maximal in either positive or negative direction while it is active. This yields two options: either accelerate for a certain period τ then decelerate until the kick, or first decelerate for a period τ and then accelerate until the kick. An example of the this strategy is shown in figure 8. It shows the possible states a unitary mass can reach within one second when a unitary force is applied. In solid blue lines are possible trajectories of the unit mass when a unit force is applied in positive direction for a period τ then the direction of the force is flipped to negative direction for the remainder of the time. In solid red, the opposite in applied. First the force is applied in negative direction, then switch to positive. The dashed black lines show the envelopes of the possible position and velocity states after 0.2, 0.4, 0.6, 0.8 and 1 seconds.

In the case of the robotic leg kicking the ball, the knee joint is almost extended at the time of the kick t_e. Therefore, strategy of extending the knee for a period τ then flexing in would result in the knee reaching its limit. To avoid this situation, the optimal correction is searched only by flexing the knee then extending it. The remaining search space is one dimensional. The optimal value of the initial deceleration period τ is searched between 0 and the remaining time before the kick $t_e - t$.

Artificial cerebellum: The last controller to be implemented is the artificial cerebellum. The architecture chosen for this application is the Cerebellar Model Articulation Controller (CMAC) proposed by J.S. Albus. The task of this controller is to anticipate the action of the correction controller defined above. Since the artificial cerebellum is the focus of this thesis, an in depth explanation of its function is provided in chapter 3.

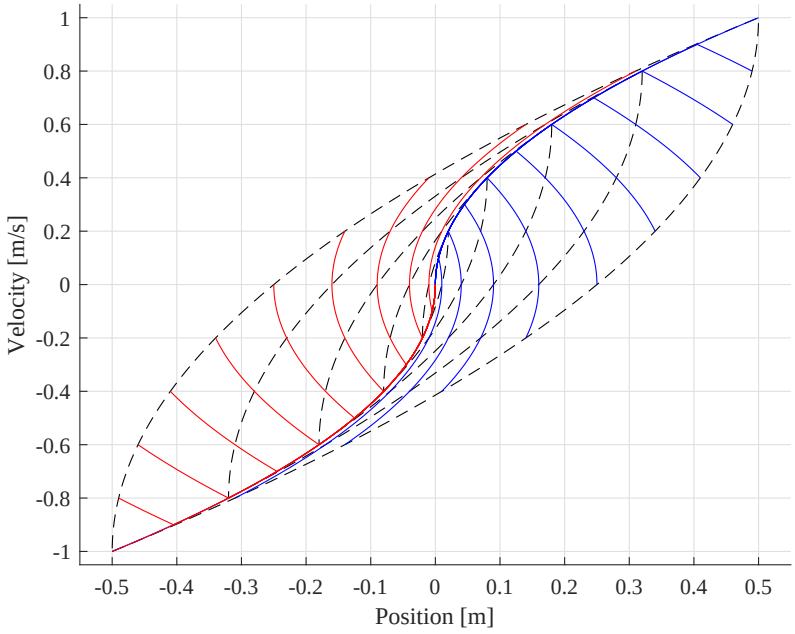

Figure 8: Reachable position and velocity states for a unitary mass subjected to a unitary force for one second. Dashed black lines: the envelopes of reachable positions in 0.2, 0.4, 0.6, 0.8 and 1 seconds. Solid blue lines: trajectories of the mass when a positive unitary force is applied for τ seconds followed by a negative unitary force for 1-τ seconds. Solid red lines: trajectories of the mass when a negative unitary force is applied for τ seconds followed by a positive unitary force for 1-τ seconds. The values for τ reach for 0 seconds to 1 second with step of 0.1 seconds.

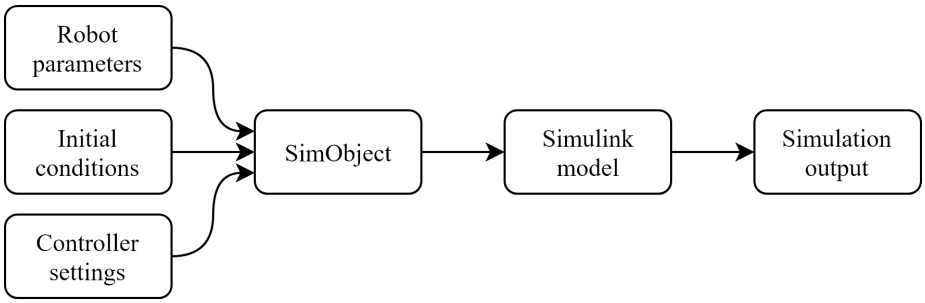

Figure 9: MATLAB simulation work-flow.

2.4 MATLAB implementation

To simulate the trajectories of the leg and ball, the equations of motion (17) are numerically integrated over time. The software chosen to implement the setup is MATLAB, which is well established in the academic community.

The MATLAB model is primarily based on a Simulink model with the simscape multibody library. The model is divided between the physics simulation and the motor control. The physics simulation calculates and updates the coordinates of each body as a function of the input torques on the hip and knee. The motor control block determines the torque on each joint, based on the available information: The position and velocity of the ball and robot joints.

At the top level of the MATLAB program, a simulation object (simObject) is created which contains all the parameter information necessary to perform the simulations in Simulink. For each simulated trial, the simObject contains the following information:

- The fixed physical parameters of the robot.
- The initial conditions of the robot and ball.
- The setting of the controllers including the artificial cerebellum.

The simObject is used to update and initialize the Simulink model which can then be run to produce the following output:

- The position, velocity and acceleration of the hip and knee.
- The position, velocity and acceleration of the ball in x and y-direction.
- The time of each data point.
- The output of the CMAC if available.
- The torque applied to the hip and knee by the controllers.

The data produced by Simulink model is returned to the top level of the program to be saved for further processing and analysis. The work-flow of the program is visualized in figure 9.

2.4.1 Integration method

To obtain the functions of the system states (α, β, x_b, and y_b), the equations of motion (17) are used. These equations, however, provide the accelerations of each state ($\ddot{\alpha}$, $\ddot{\beta}$, $\ddot{x}_b$, and $\ddot{y}_b$). Therefore, they must be integrated twice to obtain the positions.

The integration method used is the Dormand–Prince method, the default method in MATLAB. Being a fourth order Ordinary Differential Equation (ODE) solver with a fifth order error estimate, the method is referred to as ode45 within MATLAB. Due to the built-in error measurement step, this method is particularly suited to perform integrations with a varying time step. This allows larger time steps to be taken when the ball is far away, and smaller time steps when the ball is in contact with the leg, resulting in sudden high accelerations. Overall, this allows fewer function evaluations and a shorter simulation time.

2.4.2 Simulink optimizations

The Simulink model is built to be compatible with two optimizations provided by the MATLAB framework to decrease the simulation time. Firstly, the model is optimized to use the Fast Restart feature. This enables the Simulink model to be run multiple times with different initial condition without recompiling the model. Secondly the model is optimized to enable the use of the accelerator mode option in Simulink. This option shortens the simulation time by compiling parts of the model into an executable file.

3 Cerebellar Model Articulation Controller

The cerebellum is a part of the brain located at the lower back of the head as can be seen in figure 10. Nowadays, there are many models of artificial cerebellum trying to replicate the functions the cerebellum has in the human brain.

This chapter explains the origin of the selected cerebellar model, and is implementation in the setup. We must first establish an understanding of the functions and anatomy of the real cerebellum.

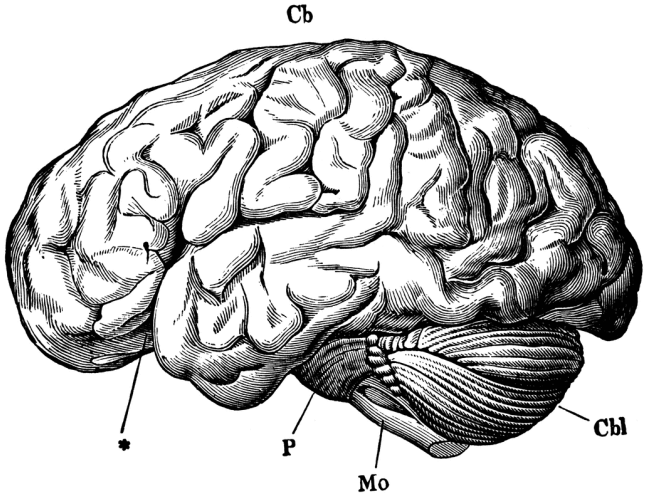

Figure 10: The brain from the left side. Labels: Cb, the cerebral hemispheres forming the main bulk of the fore-brain; Cbl, the cerebellum; Mo, the medulla oblongata; P, the pons Varolii; *, the fissure of Sylvius. Source: Martin 1884.

3.1 Functions of the biological cerebellum

In its 525 million years of evolution, the cerebellum has gathered a wide variety of functions. The initial function of the cerebellum is understood to be of predictive nature: using environmental cues, the cerebellum is able to anticipate future stimuli [Bell et al. 1997]. The functions of the cerebellum have later expanded to include fine motor control, balance, rhythmic perception and even higher cognition.

From the multitude of functions covered by the cerebellum, a selection of functions that are desirable for robotics are described below:

Fine motor control: Fine motor control, or the coordination of multiple muscle groups to accomplish precise tasks, is a desirable trait in robotics.

The execution of fine motor control tasks such as playing the piano or violin is one of the primary functions of the cerebellum [Kolb and Whishaw 2008].

In fact, the relative size of the cerebellum between species is a good indicator of dexterity in vertebrates [Sultan and Glickstein 2007]. A larger cerebellum is an indication for larger dexterity. It has also been shown that the degree of foliation (number of folds and ridges) correlates positively with the ability to use tools [Iwaniuk, Wylie, and Lefebvre 2009].

Muscle memory: Another aspect of cerebellar involvement in human movement is high skilled tasks. Typical examples are trained athletes and musicians. Their thousands of hours of training and exercises subjects the cerebellum to millions of variations of the same tasks [Kolb and Whishaw 2008]. The cerebellum associates the disturbances experienced during training with the corrections that were applied. After training, the cerebellum can adjust the motor commands to account for known disturbances. As a result, the tasks can be performed without cognitive overhead, by relying on the acquired skill.

If applied in the context of robotics, the primary controller may not have to take into account disturbances during the planning and execution of specific tasks. Instead, the corrections to the disturbances may be learned by an artificial cerebellum.

Rhythmic perception: Finally, the execution of these skilled task relies on proper timing of the movement. It has been shown that the cerebellum is also involved in rhythmic perception: the ability in estimating the duration of stimuli or the interval between stimuli [Kolb and Whishaw 2008].

3.2 Cellular anatomy of the biological cerebellum

The cellular structure of the cerebellum is very homogeneous over its entire surface and even varies very little between different species. At the cellular level however, the difference can be made between three distinct layers. From the surface to the center, one can distinguish the layers listed beneath. Additionally, the different layers and neuron types are visualized in figure 11.

- The molecular layer contains stellate cells and basket cells. Due to their location and similar function they are also called molecular layer interneurons (MLI). The molecular layer also contains the parallel fibers (granule cell axons) which run along the grooves of the cerebellum and orthogonal to the large dendritic trees of the Purkinje cells.
- The purkinje cell layer is a relatively thin but recognizable interface layer and almost exclusively contains the bodies of the Purkinje cells.
- The granular layer contains the most cells, mainly very small granule cells, some larger Golgi cells, unipolar brush cells (UBC), and finally, the Lugaro cells are also found in this layer.

As explained in the literature survey [Duijneveldt 2018], the functions of the different cells in the cerebellum can be described as follows:

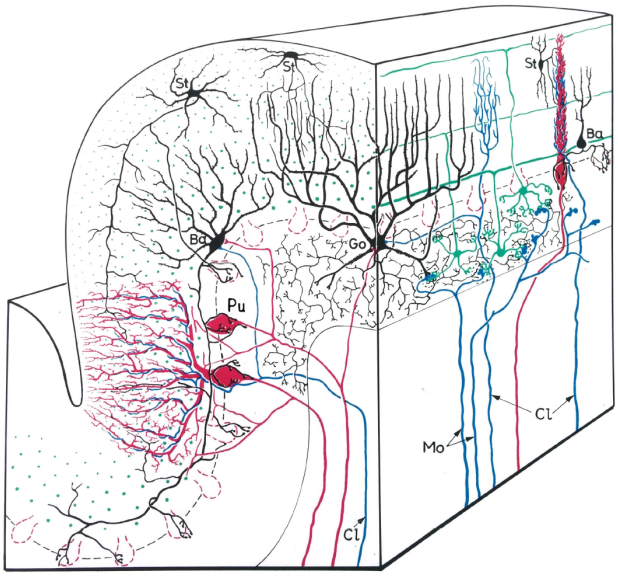

Figure 11: A stereo diagram of a section of the cerebellum adapted from Eccles, Ito, and Szentágothai 1967. Labels: Ba, basket cell; Cl, climbing fiber; Mo, mossy fiber; Pu, Purkinje cell; St, stellate cell.

Purkinje cells are arguably the most distinctive of all cells types in the cerebellum. Their somata are located in the Purkinje layer and their dendritic trees grow upward in the molecular layer. Each of the 15 million Purkinje cells has a dendritic tree forming a dense plane orthogonal to the grooves of the cerebellum. This plane is traversed by hundreds of thousands of parallel fibers (axons of granule cells) which synapse en passant. Since the Purkinje cells sum the weighted input from the parallel fibers, this synapse is often described as the dot product of their input, the parallel fibers. Another important afferent are the climbing fibers. Each Purkinje cell is accompanied by a single climbing fiber (originating from the inferior olive) covering the entire dendritic tree. Because of the numerous synapses, the climbing fiber has a very strong input on the Purkinje cell.

Granule cells are the most abundant neurons in the cerebellum and thereby the entire human body. Their soma is located in the granular layer where they receive excitatory input from the mossy fibers and inhibitory input from Golgi cells. The axons of the granule cells, called parallel fibers, grow outward to the molecular layer where they diverge medially along the grooves of the cerebellum. In the molecular layer, the up to 7mm long fibers traverse up to 1 110 dendritic trees orthogonally [Brand, Dahl, and Mugnaini 1976]. Due to their very small diameter (0.2μm–1.1μm), up to 200 000 parallel fibers can synapse with a single Purkinje cell [Wyatt, Tanapat, and Wang 2005].

Golgi cells are inhibitory cells acting on the granule cells. Since they receive their input from mossy fibers and parallel fibers, they function both as feed forward and feedback signal modulating parallel fiber activity. The commonly attributed function is the spatiotemporal reshaping of the input signal from the mossy fibers to the parallel fibers [Llinas and Negrello 2015]. Their inhibitory (feedback) effect on the granule cells serves to transform prolonged signals from the mossy fibers into a short burst in the parallel fibers. Neighboring Golgi cells in that same transversal plane tend to synchronize their activity. The feedback loop between the Golgi cells and granule cells may contribute to adequate timing of granule cell spikes [Vos et al. 1999].

Basket cells and Stellate cells are situated in the molecular layer where they receive input from the parallel fibers. Their axons synapse on the Purkinje cells and are inhibitory in nature. Since their function is very similar they are sometimes collectively referred to as molecular layer interneurons (MLI). Their inhibitory action on the Purkinje cells is needed to balance for the excitatory parallel fibers. This balance allows for simple patterns to be recognized. Since they receive the same input as the Purkinje cells and their axons diverge to up to 150 Purkinje cells, they are believed to increase the contrast of the Purkinje cell input in a similar way to unsharp masking in digital image editing. A functional difference between the basket and stellate cells is their relative size. The stellate cell are smaller and consequently operate at a different scale.

Lugaro cells are a special class of cells situated between Purkinje and granular layer. The axons of the Lugaro cells spread relatively far in the cerebellum and synapse with a over a hundred Golgi cells [Dieudonné and Dumoulin 2000]. They are normally silent, but they activate under the influence of serotonin[1]. When activated, they produce a rhythmic 5–15Hz inhibitory signal [Ito 2006].

As to their function in the cerebellum, different hypotheses are proposed. One possibility is that the Lugaro cells provide a feed-back control on Purkinje cells through the MLIs [Lainé and Axelrad 1998]. However, research by Vos et al. 1999 offers a different function for the Lugaro cells. They suggest that Lugaro cells (due to their large divergence on Golgi cells) may contribute to establishing synchronization among neighboring Golgi cells.

Unipolar brush cells (UBCs) are small interneurons located in the granular layer of the cerebellum. They receive input from mainly the mossy fibers but also feedback through the Golgi cells. Their excitatory output on the granule cells is thought to reshape the sensory input of the mossy fibers [Llinas and Negrello 2015].

[1]Serotonin is a neurotransmitter commonly associated with happiness, however, it plays a role in regulating many processes and fluctuations can affect learning, depression, and obsessive-compulsive disorders [Kolb and Whishaw 2008].

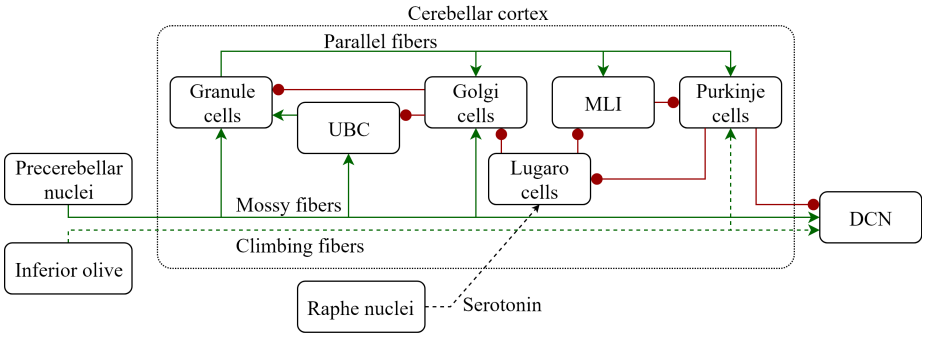

Figure 12: Schematic overview of the connectivity of the cells present in the cerebellum. Regular arrows indicate excitatory synapses whereas round arrows indicate inhibitory onces. Dashed lines are used to distinguish special connections such as the climbing fibers and the serotonin regulation.

3.2.1 Simple and complex spikes

The Purkinje cells are involved in cerebellar learning. They receive excitatory synapses from two sources: the granule cells and the mossy fibers. When excited by the granule cells, the Purkinje cells mostly produce a steady signal of action potentials called simple spikes. These regulate the normal processes of the cerebellum. However, when the situation deviates from the anticipated situation, the climbing fibers produce a excitatory signal. Due to the large number of synapses between the climbing fiber and Purkinje cell, this signal is extremely potent. This cause the Purkinje cell to produce a complex spike: A series of fast actions potentials followed by a refractory period. This interplay between simple and complex spikes is believed to be the basis of cerebellar learning [Pinzon-Morales and Hirata 2015].

3.3 Implementation of the artificial cerebellum

The Cerebellar Model Articulation Controller (CMAC) is a simple network inspired by the cerebellum. The model is first proposed by J. S. Albus in 1975, designed to capture the dominant synapses in the cerebellum. It models the interaction between the parallel fibers (carrying the sensory-motor information) and the Purkinje cells.

In essence, the CMAC is an injective function that associates each possible input (the robot state) to an output (the controller action). The implementation of the CMAC can be summarized in the following four steps:

- Discretize input state variables.
- Select activated Purkinje cells.
- Sum input from the activated cells.
- Adjust the weights of activated neurons if training

The input to the CMAC is the state space of the setup. In this particular setup, the input has eight dimensions: The angular positions of the hip and knee their derivatives, as well as the Cartesian coordinates of the ball and their derivatives. The input of the CMAC is a vector denoted x_i and is defined as follows:

$$x_i = \left[\alpha, \beta, \dot{\alpha}, \dot{\beta}, x_b, y_b, \dot{x}_b, \dot{y}_b\right]^T \tag{49}$$

3.3.1 Discretization

The first step of the CMAC is to convert the continuous variables $x_i \in \mathbb{R}^8$ to a discrete set $s_i \in \mathbb{N}^8$. Each state dimension i has three associated design parameters:

- min_i: the minimal allowed value of x_i.
- max_i: the maximal allowed value of x_i.
- r_i: the number of discrete values of s_i.

Therefore the proposed linear mapping is as follows:

$$s_i = \max\left(\min\left(\left\lfloor r_i \frac{x_i - min_i}{max_i - min_i}\right\rfloor, r_i - 1\right), 0\right) \tag{50}$$

This ensures each value of x_i between min_i and max_i to be uniformly distributed on $[0, r_i - 1]$. Values of input states x_i below min_i and above max_i are respectively mapped to 0 and $r_i - 1$.

3.3.2 Neuron positioning

Once the input state x_i is discretized, the most straightforward method of obtaining a controller value would be to associate a weight to each possible state s_i. A simple calculation of the number of possible states however shows that

this is not feasible for most situations. Indeed, using eight dimensions and a resolution of $r_i = 100$, equation (51) yields 10^{16} distinct states. Assigning a weight to each state using double format would require 80 petabytes of storage.

$$n_{states} = \prod_i r_i \tag{51}$$

Instead, the neurons (weights) are sparsely distributed over the i-dimensional space. The entire space is subdivided into i-dimensional hypercubes of size n_a and each hypercube is filled with the same number of n_a neurons. This reduces the total number of required neurons drastically: by a factor ${n_a}^{i-1}$. Finally, the spacing pattern of the neurons is an important choice. According to the work of Parks and Militzer 1991, the following rules apply:

a) The same pattern is repeated for each hypercube.
b) A single neuron lies on each hyperplane orthogonal to an edge of the hypercube.
c) The neurons are spaced uniformly across the entire space. Maximizing the distance between the closest neighboring neurons.

The proposed equation for the coordinates of all neurons N_j $(j = 1, 2, ..., n_a)$ in each hypercube is as follows:

$$N_{j,i} = \mod (j \cdot d_i, n_a) \tag{52}$$

The value of d_i is a design parameter and determines the offset between the neurons in each dimension. Optimal values of d_i are determined in the same work by Parks and Militzer 1991. As illustrated in figure 13, using the recommended value $d_i = (1, 2)$ for $n_a = 5$, the positions for the neurons are as follows:

$$N_0 = (0, 0),\ N_1 = (1, 2),\ N_2 = (2, 4),\ N_3 = (3, 1),\ N_4 = (4, 3)$$

Finally, to account for the sparsity of the neurons, an activation region is attributed to all neurons. Each neuron is activated if and only if the input s_i is within the activation region of the neuron. This region is an i-dimensional hypercube of size n_a surrounding each neuron. Figure 13 shows the activation regions in dashed outline for all N_0 neurons highlighted in gray. As a result of this activation region and the regular distribution of neurons, it is guaranteed that for every state s_i, there are exactly n_a neurons activated. The output of the controller is the sum of the weights of these n_a neurons.

The next step is to determine which neurons are activated given a certain discretized state s_i.

3.3.3 Neuron selection

Each neuron is activated if and only if the discretized input s_i is within the receptive region of the neuron. In other words, if the distance between the center of the neuron and s_i is smaller than the selected radius.

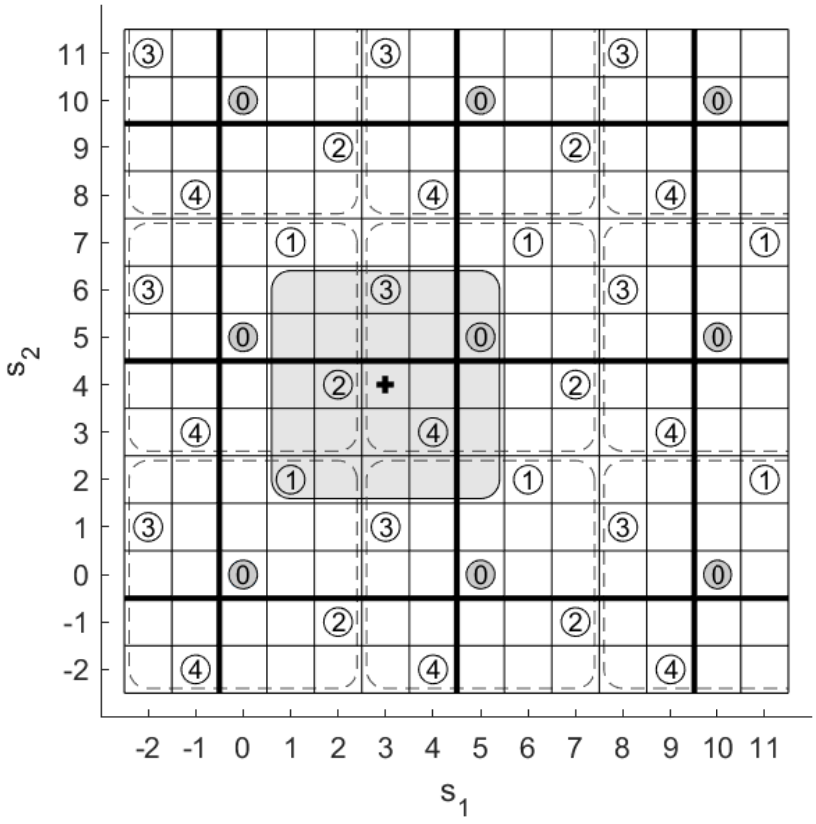

Figure 13: Example of the CMAC using two dimensions and a receptive region of size $n_a = 5$. The resolution of both dimensions is $r_1 = r_2 = 10$. The depicted state $s_i = (3, 4)$ activates the following neurons: $q_0 = (1, 1)$, $q_1 = (0, 0)$, $q_2 = (0, 1)$, $q_3 = (1, 1)$, $q_4 = (1, 1)$

Checking each neuron location for possible activation would be very time consuming and wasteful. Instead, the regular structure of the neurons can be leveraged to calculate the coordinates of all n_a neurons directly. The key in this algorithm is to notice that firstly, for each neuron index j, exactly one neuron is activated and secondly, all neurons of the same index are spaced regularly by an interval of n_a. As a result, we can assign new reduced coordinates q_j to each neuron based on its position within the grid of neurons sharing the same index j.

In figure 13, all neurons of index $j = 0$ are highlighted in gray. They form a 3×3 grid in which only the neuron at reduced coordinates $q_0 = (1, 1)$ is activated.

The reduced coordinates q_j can be simply be calculated by using the floor function provided the correct affine scaling of the discretized state s_i:

$$q_{j,i} = \left\lfloor \frac{s_i + N^*_{j,i}}{n_a} \right\rfloor \tag{53}$$

The coefficient n_a ensures an increase of index for each increase by n_a steps in the discretized state. The coefficient $N^*_{j,i}$ is a linear transformation, related to the neuron coordinates and ensures the step increase is located at the right index. It is defined as follows:

$$N^*_{j,i} = \mod \left(\left\lfloor \frac{n_a}{2} \right\rfloor - j \cdot d_i, n_a \right) \tag{54}$$

As an example using these equations, the reduced coordinates of the active neurons can be calculated as illustrated in figure 13. Using a receptive region of $n_a = 5$, a discretized state $s_i = (3, 4)$, and $d_i = (1, 2)$, we obtain the following reduced coordinates:

$$q_0 = (1, 1),\ q_1 = (0, 0),\ q_2 = (0, 1),\ q_3 = (1, 1),\ q_4 = (1, 1)$$

To be usable in a computer implementation the reduced coordinates must be associated with a memory location. To do so, a unique address a is attributed to each neuron:

$$a_j = \sum_{k=1}^{i} q_{j,k} \prod_{m=k+1}^{i} r_m \tag{55}$$

3.3.4 Hashing

The number of possible addresses a can still be significant, moreover, many states s_i may never be visited since they do not represent a plausible state for the robot. It would, therefore, be wastefull to store a weight for each address (neuron).

A popular technique to reduce memory requirements in such cases is hashing. This is the process of mapping data of an arbitrary size onto data of a fixed size.

Hashing is a commonly used method with CMAC controllers. A hash function maps an input to a set of fixed size. Since the addresses are integers, a simple hashing method that can be used is the modulo function. To map the addresses a to a total number n_h of possible hashed addresses h, the following equation can be used:

$$h_j = \mod (a_j n_a + j, n_h) \tag{56}$$

As a result of this operation, the resulting hashed addresses h are guaranteed to be in the set $[0, n_h - 1]$. When the set of inputs is larger than the set of possible hashes, collisions are guaranteed to occur. This happens when two different inputs produce the same hash. An example of collisions is shown in table 1, where six-digit numbers are represented by their last two digits.

Input	Operation	Hash
112233	112233 mod 100	33
123456	123456 mod 100	56
654321	654321 mod 100	21
123321	123321 mod 100	21

Table 1: Example of the modulo 100 hash function: Each input is represented by its last two digits. A collision can occur when different inputs produce the same hash.

There are different strategies to resolve collisions when they happen. These methods require the address a to be stored along with the weight associated with the address. Experience has shown that in the case of CMACs the occurrence of collisions may not severely impact the performance of the controller. Even though the collisions do introduce noise, the number of active neurons at each time are able to compensate.

As the last step of the CMAC algorithm, the hashed addresses h_j are used to retrieve the weights w_j from an indexed table. The output W of the CMAC is simply the sum of all the active weights for the given input:

$$W = \sum_j w_j \tag{57}$$

3.4 Learning algorithm

The weights of the of the cerebellar model articulation controller (CMAC) must be adjusted in order for the network to predict the actions of the controller. The network is therefore trained by repeatedly simulating the task under different conditions. After each simulated trial, the actual and predicted controller signals are evaluated and compared. Then, for each time-step, the weights of the CMAC are slightly adjusted to minimize the difference.

To obtain consistent training conditions, the input states x_i are resampled at a rate of 1 kHz. At each step, a total of n_a weights w_j are active to compute the CMAC output W. They are adjusted according to the following equation:

$$w_j \leftarrow w_j - \alpha \frac{e}{n_a} \tag{58}$$

Where e is the difference between the actual and desired output W of the CMAC. The coefficient α is the learning rate. This determines how fast the weights adapt to new information. The choice of learning rate is often a delicate balance. A high learning rate enables the network to adapt quickly to the presented information. However, the consequence of this adaptability is poor generalization of the problem.

The selection of an adequate learning rate in machine learning is always a delicate task. In this experiment, an appropriate learning rate was found empirically to be $\alpha = 0.001$. This value is relatively low to compensate for the fact that the same weights w_j may be adjusted multiple times due to the 1 ms sample time.

3.5 Design parameters

The final step of the implementation is to determine adequate design parameters for the artificial cerebellum.

The number of input dimensions (eight) and output dimensions (one) are imposed by the setup. Given the fixed input dimension, Parks and Militzer 1991 have determined optimal combinations of size of the receptive region n_a and spatial distribution of the neurons d_i. From this list, the size $n_a = 34$ was chosen for the receptive region. The corresponding optimal spatial distribution is $d_i = [1, 3, 5, 7, 9, 11, 13, 15]$

The number of possible hashes n_h determines the quantity of information the CMAC is able to store. Using an total number of 800 011 possible hashes, about 41 % of the values are used after training. Stored in double format this requires a little over 6.4MB of space. The chosen number is prime, which is often recommended for the modulo function to reduce the amount of structure in the obtained hashes.

For the discretization step, the maximum and minimum values of the input must be determined. Some values are straightforward, such as the joint limits. Other values such as the maximum velocities are found empirically by simulating different situations. The resulting maxima max_i and minima min_i are reported in table 2.

The resolution r_i of the discretization was optimized using a simple genetic algorithm. The CMAC was trained for varying values of r_i. The values of r_i that resulted in the lowest training error were then varied to continue the search.

Unit	rad	rad	rad/s	rad/s	m	m	m/s	m/s
min_i	-1.57	-2.36	-10.00	-10.00	0.00	0.00	-10.00	-3.50
max_i	2.09	0.05	10.00	10.00	5.00	2.00	-3.00	1.50

Table 2: Minimum and maximum values of each input dimension.

Unit	deg	deg	deg/s	deg/s	cm	cm	cm/s	cm/s
$\frac{max_i - min_i}{r_i}$	1.24	0.43	7.30	4.97	0.56	1.57	7.00	7.25

Table 3: Size of the discretization step for each input dimension.

This ultimately resulted in a discretization resolution of $r_i = [169, 318, 157, 231, 890, 127, 100, 69]$. This result of the optimization shows the relative importance of the angle of the knee discretized in steps of 0.43 degrees compared to 1.24 degrees per step for the angle of the hip, as can be seen in table 3.

With all the design parameters of the CMAC defined, the artificial cerebellum can be implemented into the robotic setup. The setup can than be simulated to test the hypotheses established in this thesis.

4 Experiments

The experiments presented in this chapter serve to test the research question: *Can an artificial cerebellum be used to anticipate controller corrections and increase the performance of a soccer robot by reducing delays?*

Firstly, the artificial cerebellum is trained to predict the actions of the correction controller. Secondly, the artificial cerebellum is integrated in the system and the performance of the combined system is evaluated.

4.1 Experiment I

During the first experiment, the artificial cerebellum is trained to anticipate the controller actions.

To train the Cerebellar Model Artificial Controller (CMAC), a total of 1 000 initial positions are generated using the method described in chapter 2. For each of the 1 000 initial conditions, the target velocities of feed forward controller are optimized to kick the ball as close to the target as possible. Once a suitable feed forward target is obtained, a lateral disturbance is introduced 200 ms before t_e, the time of the kick. For each initial condition, a total of 10 different lateral disturbances are introduced. The values of the disturbances range between -20 cm and +19.8 cm in x-direction. The lateral disturbance Δx can be calculated using the initial condition number $i \in [0, 999]$ and disturbance number $j \in [0, 9]$, using the formula below.

$$\Delta x(i, j) = -0.2 + 0.04j + 0.002(i \bmod 20) \tag{59}$$

For each trial i, and each disturbance number j, the corrective controller finds the optimal correction. This controller, however, has an arbitrary delay of 100 ms to represent the processing of the information to find an optimal correction. Therefore, the correction controller has only a 100 ms window available act.

$$T_{correction}(t) = \begin{cases} 0, & \text{if } \quad t < t_c \\ -10, & \text{if } t_c \leq t < t_s \\ 10, & \text{if } t_s \leq t \end{cases} \tag{60}$$

The goal of the CMAC is to compensate for the delay of the correction controller. Using a 30ms reduction, we obtain that any time t, the sum of the correction controller and the desired CMAC output is equal to the expected output of the correction controller 30 ms later:

$$T_{CMAC}(t) + T_{correction}(t) = T_{correction}(t + 30 \text{ ms}) \tag{61}$$

Rearranging the terms we obtain the equation for the desired output of the CMAC. The desired values of T_{CMAC} are used to train the artifical cerebellum.

$$T_{CMAC}(t) = T_{correction}(t + 30 \text{ ms}) - T_{correction}(t) \tag{62}$$

To ensure proper training conditions and avoid over-fitting to any particular situation, all sets of all initial conditions and disturbances (i, j) are permutated

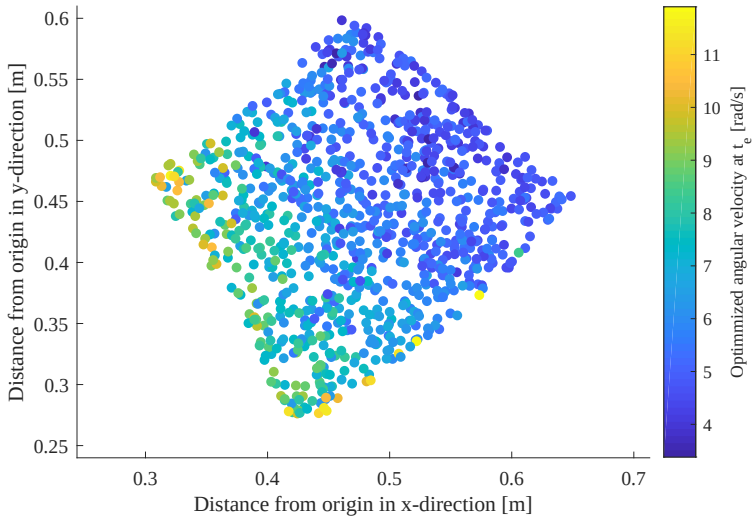

Figure 14: Results for the feed forward controller. Each dot represents the x and y-location of the planned kick for all 1 000 initial conditions. The color indicates the optimal velocity of the leg.

into a random order. This results in a total of 10 000 trials. For each trial, the correction attempt is simulated and the system states over time are saved. For consistency between trials, the system states are resampled to a fixed sample rate of 1 ms. These resampled states are used as training input for the CMAC. For each time step of 1 ms, the active weights are computed and summed. The difference between the obtained sum (CMAC output) and the desired output T_{CMAC} is the training error. The training error is used to adjust the active weights after being adjusted by the learning rate $\alpha = 0.001$.

Finally, the distances reached in this experiment are saved, to be used later as a baseline for comparison for the second experiment.

4.1.1 Results for experiment I

In order to conduct the experiments I and II the feed forward controller must be able to find a suitable solution in the undisturbed case. As shown in figure 14, the optimal velocities are fairly homogeneous. The target of 10 m is reached within a margin of ± 1.5 cm in 992 out of the 1 000 initial conditions. These cases are situated on the edge of reachable envelope of the leg. This can be observed indirectly in figure 14 as the optimization fails and the found optimum does not match neighboring points.

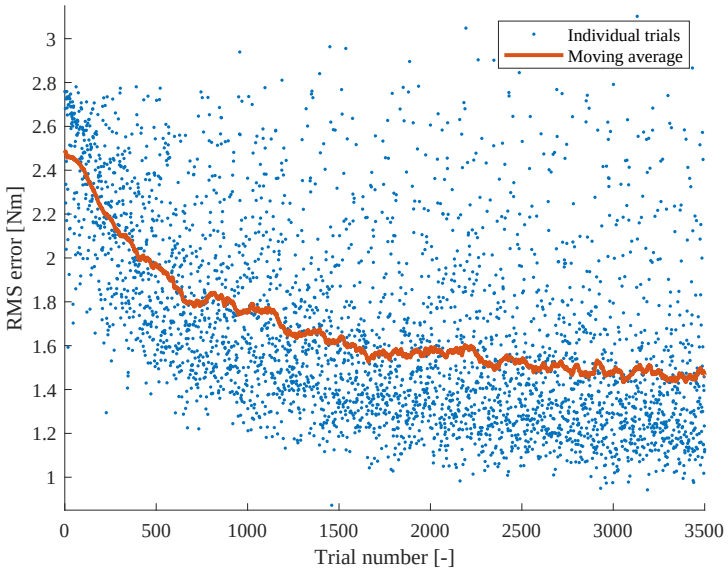

Figure 15: Training progress of the CMAC over the first 3 500 trials. Each dot represents an individual trial. The RMS error is measured before adjusting the weight. The moving average (solid line) is calculated over 200 trails centered around the current position.

The progress of the training can be evaluated by measuring the Root Mean Square (RMS) difference between the desired output of the CMAC and the actual output of the CMAC. Figure 15 shows the progress of the CMAC training process. To exclude the effect of measuring fitted data, the RMS error of the CMAC is measured before adjusting the weights to fit the desired output.

To illustrate the output of the trained CMAC, a sample trial is arbitrarily selected. The output of the CMAC to the selected trial is shown in figure 16. The CMAC output (before adjusting the weights) and the desired output are superimposed for comparison. As reference for the timing of the signal, the corrective signal is graphed as well as the a priori predicted timing of the kick.

Figure 17 shows the speed $\dot{s}$ of the input through discretized space. This speed is defined as follows. Firstly, the velocity vector in the input space is the first derivative of the position vector or input state:

$$\dot{x}_i = \left[\dot{\alpha}, \dot{\beta}, \ddot{\alpha}, \ddot{\beta}, \dot{x}_b, \dot{y}_b, \ddot{x}_b, \ddot{y}_b\right]^T \tag{63}$$

Secondly, to obtain the velocity of the input through the discretized space of the CMAC, the velocity vector $\dot{x}_i$ is scaled by the same linear coefficient as

in equation (50). This is equivalent to taking the first time derivative of the discretized state s_i is this were possible:

$$\dot{s}_i = \frac{r_i}{max_i - min_i}\dot{x}_i \tag{64}$$

Finally the speed $\dot{s}$ of the input through the discretizd space is calculated as the Manhattan distance or L_1-norm of the velocity vector $\dot{s}_i$.

$$\dot{s} = \sum_i |\dot{s}_i| \tag{65}$$

The graph of the speed $\dot{s}_i$ of the input through the discretized space is presented in figure 17. This speed is relatively steady between 3 ms^{-1} and 6 ms^{-1} before the leg touches the ball. During contact, the speed $\dot{s}$ briefly reaches 133 ms^{-1}.

4.1.2 Conclusion for experiment I

The results of experiment I show that the CMAC is able to learn to predict the correction controller output. This is supported by the decrease in Root Mean Square (RMS) error over the training trials as is shown in figure 15. The RMS error of the CMAC output shows a very clear decrease in the majority of the trials. The average RMS error drops from 2.49 Nm to 1.46 Nm. This leads to conclude that the cerebellar architecture is indeed suitable for the effective prediction of controller actions.

The output of the CMAC, as shown in figure 16, demonstrates that the artificial cerebellum is not capable of replicating the sharp transitions of the teaching target. This can be explained by how fast the input of the CMAC changes: for small changes in input, there is an overlap in activated neurons. In fact, given a receptive region n_a of 34 neurons and a speed $\dot{s}$ of 3 ms^{-1} through the discretized space, the CMAC is expected to activate an entirely distinct selection of neurons after at least $n_a/\dot{s} \approx 11$ ms. This seems to be consistent with the transitional behavior of the output of the CMAC observed in figure 16.

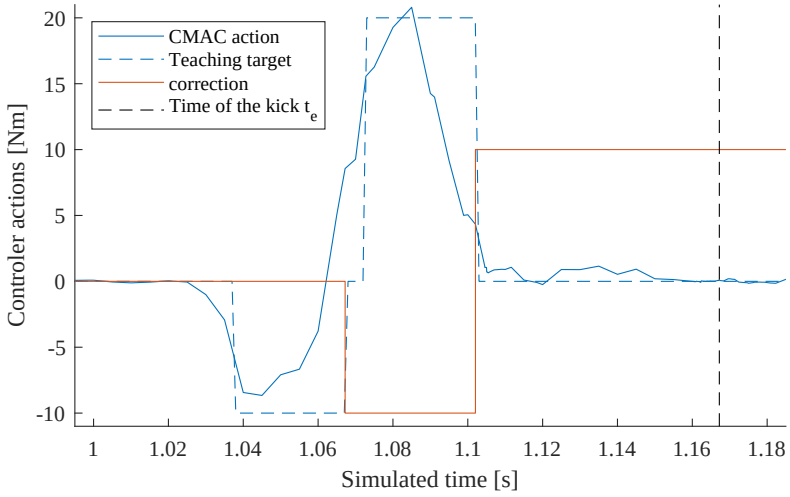

Figure 16: Typical output of the trained CMAC. In solid blue: the output of the CMAC. In dashed blue: the training signal or desired output. In solid red: the action of the correction controller for reference. In dashed black: the time t_e for reference.

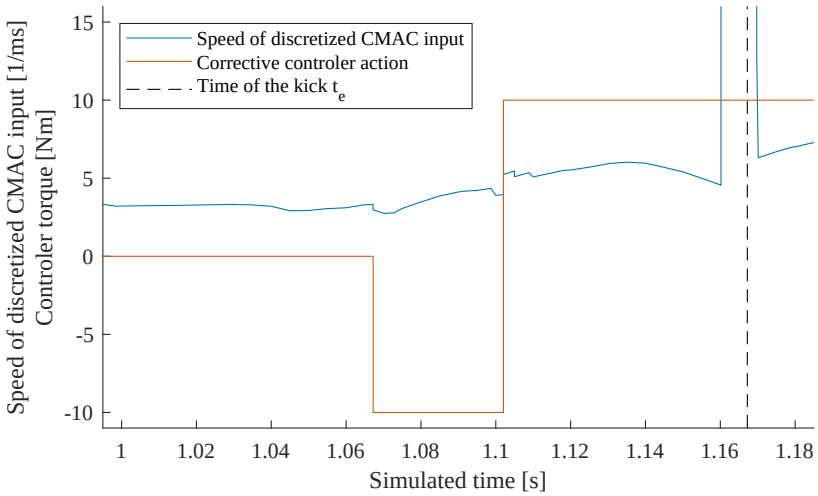

Figure 17: Speed of the input through the discretized space of the CMAC around the time of the correction and kick. Not visible on this graph, the speed briefly reaches a maximum of 133 ms^{-1} during the kick.

4.2 Experiment II

The CMAC from the first experiment is trained without feedback into the trials. Since the CMAC does have an effect on the system, the previously optimal correction must be adjusted somewhat. Consequently, the pre-trained CMAC must also be adjusted.

During the second experiment, the output of the CMAC is added to the integration during the trials. Again, the correction is optimized to get the ball as close to the target as possible. The resulting optimal trial is used as input to train the CMAC controller. As a result, during experiment II the CMAC is expected to take into account its own effect on the system. The entire system is simulated for 900 initial conditions each with 10 different values for the lateral disturbance. The same initial conditions and lateral disturbances as experiments I are used in experiment II. The initial conditions leg are fixed and the initial conditions of the ball are randomly distributed as defined in section 2.2. The lateral disturbances are space linearly according to (59).

To ensure proper training conditions and avoid over-fitting to any particular situation, all sets of all initial conditions and disturbances (i, j) are permutated into a random order. This results in a total of 9 000 trials. For each trial, the correction attempt is simulated, while the CMAC provides a 30ms prediction ahead of the correction controller. The system states over time are saved. For consistency between trials, the system states are resampled to a fixed sample rate of 1 ms. These resampled states are used as training input for the CMAC. For each time step of 1ms, the active weights are computed and summed. The difference between the obtained sum (CMAC output) and the desired output T_{CMAC} is the training error. The training error is used to adjust the active weights after being adjusted by the learning rate $\alpha = 0.01$.

To validate the results obtained during training, an additional 100 initial conditions are simulated, each with 10 different values selected according to equation (59).

4.2.1 Results for experiment II

The results of the second experiment are illustrated in figures 18 to 21. These figures either report the distance of the ball to the target (performance), or the difference (improvement) in distance between trials with identical conditions, with and without the CMAC. The improvement is defined as positive if the ball is closer to the target when the CMAC is added.

For some figures (18 and 22), the data is supplemented by interpolated values. To obtain a more dense and homogeneous dataset, the 10 disturbances simulated for each of the 900 initial conditions are used to linearly interpolate the performance for all possible disturbances $[-0.2, 0.198]$m. Where needed, the

data is also extrapolated linearly.

Figure 18 shows the performance of the robotic leg with and without CMAC for different lateral disturbances. In both cases, the performance is the highest for very small disturbances. In these cases the robotic leg almost always succeeds in kicking the ball to within a few centimeters of the target. respectively 93 % and 88 % of the trials are within 1.5 cm of the target for the trials without and with the CMAC. An asymmetry in clearly visible between negative disturbances (with the ball closer to the robot) and positive disturbances (with the ball further away from the robot). The decrease in performance is more gradual for increasingly negative disturbances and more abrupt for increasingly positive disturbances. It is also noteworthy that as the lateral disturbance increases, a large portion of the trials reaches a distance of about 10 meters to the target.

The results, previously presented in figure 18, are visually very similar. Therefore, to properly compare these two situations, the performance (distance to target) of both situations is plotted in the same graph as shown in figure 19. This figure shows a comparison of the performance without CMAC on the horizontal axis versus the performance with CMAC on the vertical axis. As a reference, the diagonal of equal performance is included as a solid black line. Since the data presented in figure 18 indicated different results depending on the lateral disturbance, the trials are divided into five categories for visualization (large positive, medium positive, small, medium negative, and large negative lateral disturbance).

Notable features in the graph are: Firstly, the band of points under the diagonal. This indicates trials where the performance is improved by the addition of the CMAC. Secondly, a vertical line of points at the origin. These points represent trials where the leg previously reached the target but, with the addition of the CMAC, no longer is on target resulting in a decrease in performance. Finally, in accordance with figure 18, the trials with a negative lateral disturbance are distributed relatively equally over the diagonal, whereas the trails with a positive disturbance tend to accumulate at the extremities.

A drawback of this representation is that the density of the points is not clearly visible in the figure. Moreover, quantitative values of the improvement are difficult to extract.

To obtain more quantitative results, a histogram of the improvements is presented in figure 20. This histogram shows a clear bimodal distribution. A large portion of the trials is unaffected and shows no improvement or the performance is slightly reduced by about 1 cm. A second portion of the trials shows an improvement in the performance of the robotic leg. The ball is shot to about 7 cm closer to the target. As suggested by figure 19, the increase in performance is strongly dependent on the direction of the lateral disturbance. This shows again in a second histogram of the improvements. When considering only negative disturbances Δx, the average improvement increases from 2.25 cm to 5.74 cm. Additionally, the probability of a trial with a decrease in performance equal or

greater than 20 cm drops from 3.00 % to 0.36 %.

The results of the training trials are confirmed by the validation trials. A histogram of the improvements of the trials over the validation data is shown in figure 21.

A final representation of the improvements can be seen in figure 22. This image shows the improvement obtained by adding the CMAC to the robotic leg. The image is produced using the augmented dataset as previously explained. Firstly, the most prominent feature is the large portion of trials that is unaffected (has an improvement of about 0 cm). Secondly, in the bottom-left, a large portion of the trials is improved by the addition of the CMAC. Up to 80 % of the trials has an improvement of at least 5 cm for large negative disturbances. Finally at the top-right of the image, a large portion of the trails has a large decrease in performance (negative improvement). For up to 10 % of the trials the performance is decreased by at least 20 cm.

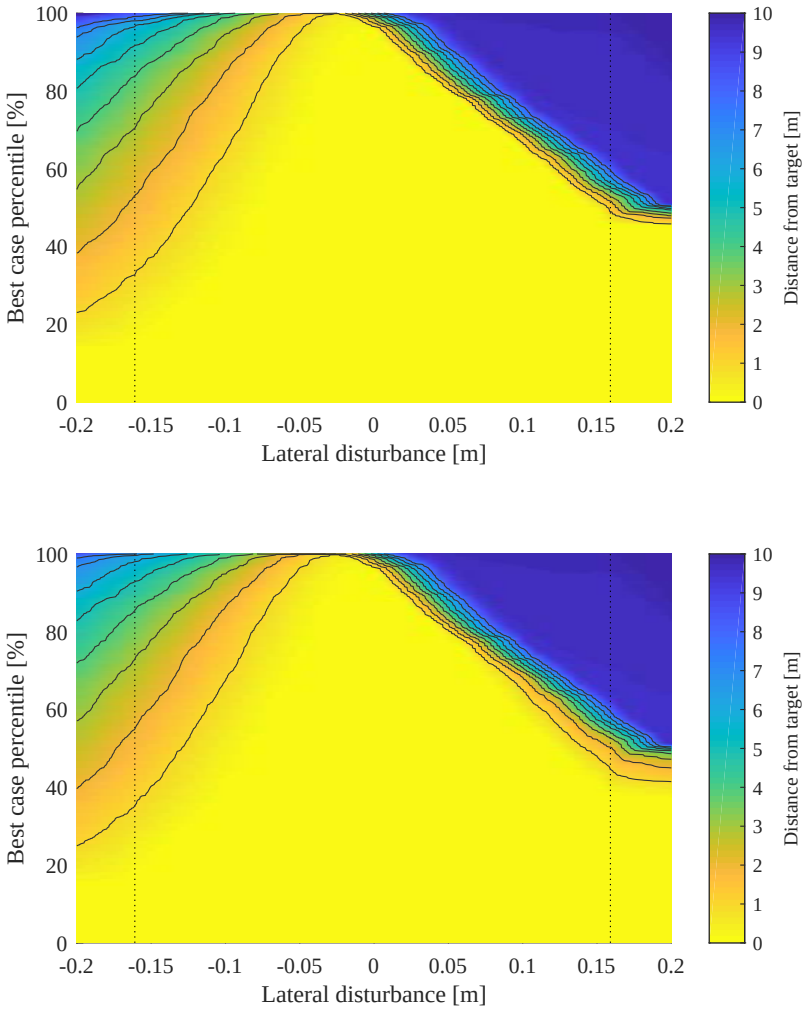

Figure 18: Top: The performance of the robotic leg without the cerebellar controller. Bottom: The performance of the robotic leg with the use of the CMAC. The data in this plot contains linearly interpolated data between the dotted vertical lines and partially contains extrapolated values outside the dotted lines. For each value of the disturbance, the performance (distance to the target) of all initial conditions are sorted and displayed by color.

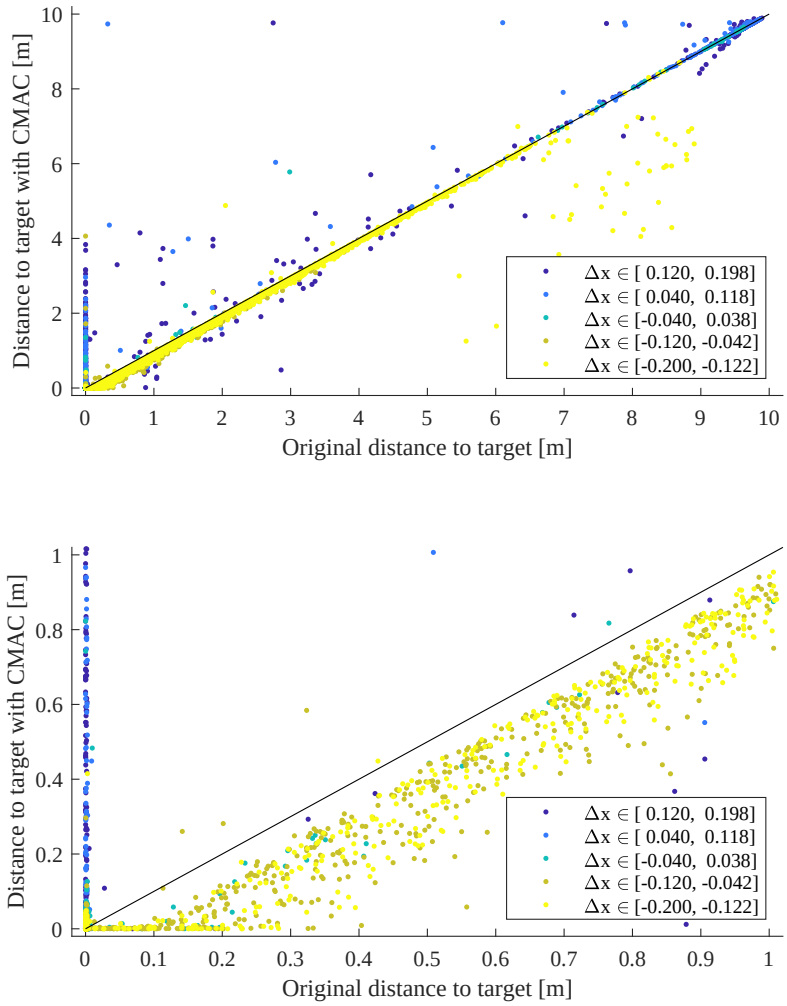

Figure 19: Top: Comparison of the distances to the target with and without the addition of the CMAC for all trials. Bottom: Close-up of the data for errors smaller than one meter.

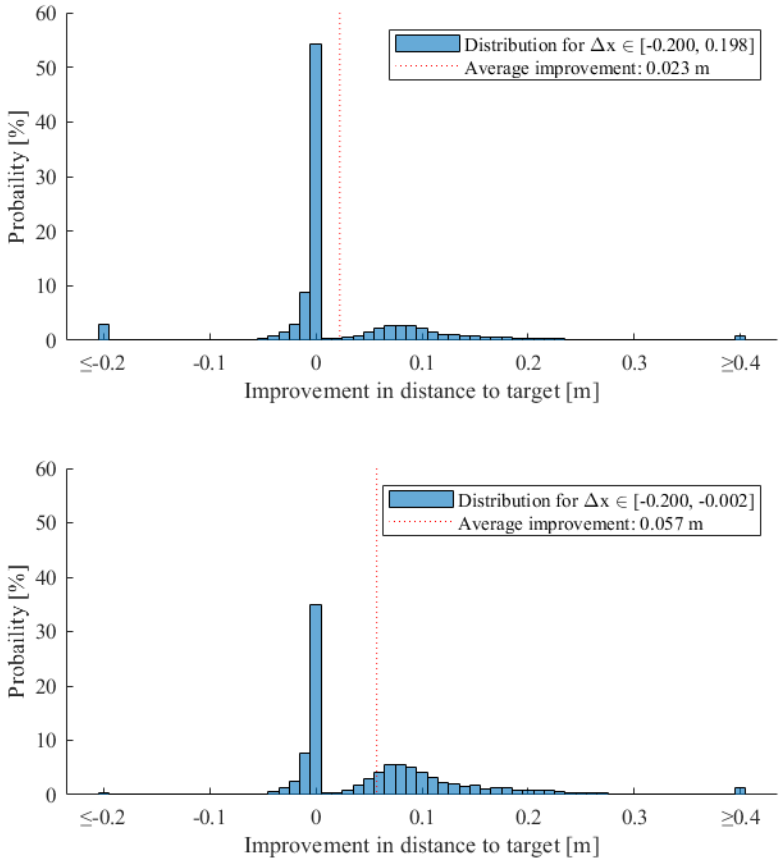

Figure 20: Histogram of the improvement in the distance to the target for all 900 training trials. The top histogram contains the improvements for all values of lateral disturbances. The bottom histogram contains the improvement only for trials with a negative lateral disturbance.

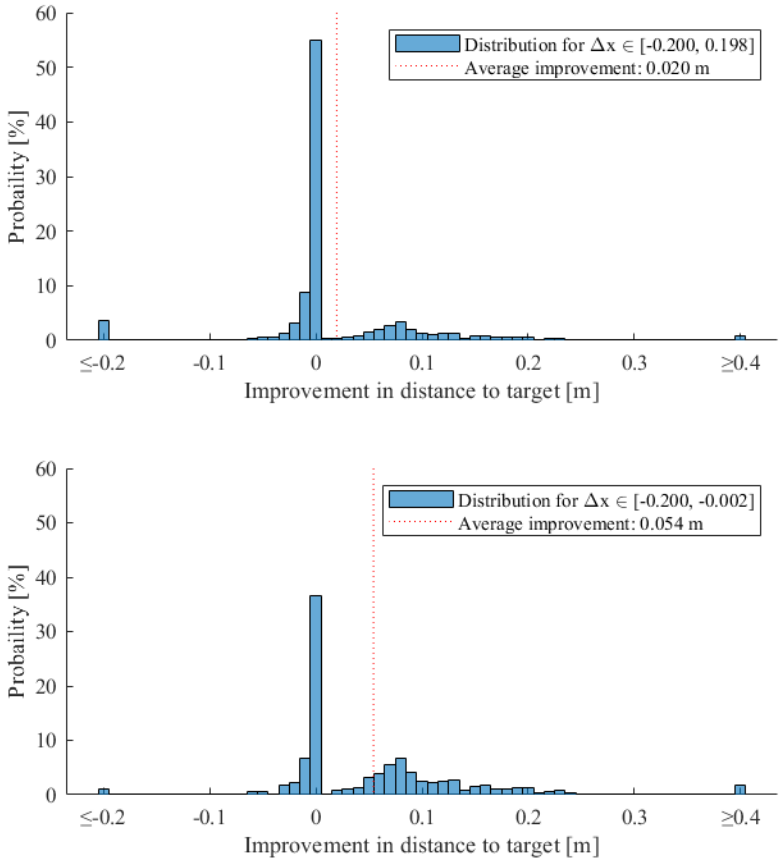

Figure 21: Histogram of the improvement in the distance to the target for all 100 validation trials. The top histogram contains the improvements for all values of lateral disturbances. The bottom histogram contains the improvement only for trials with a negative lateral disturbance.

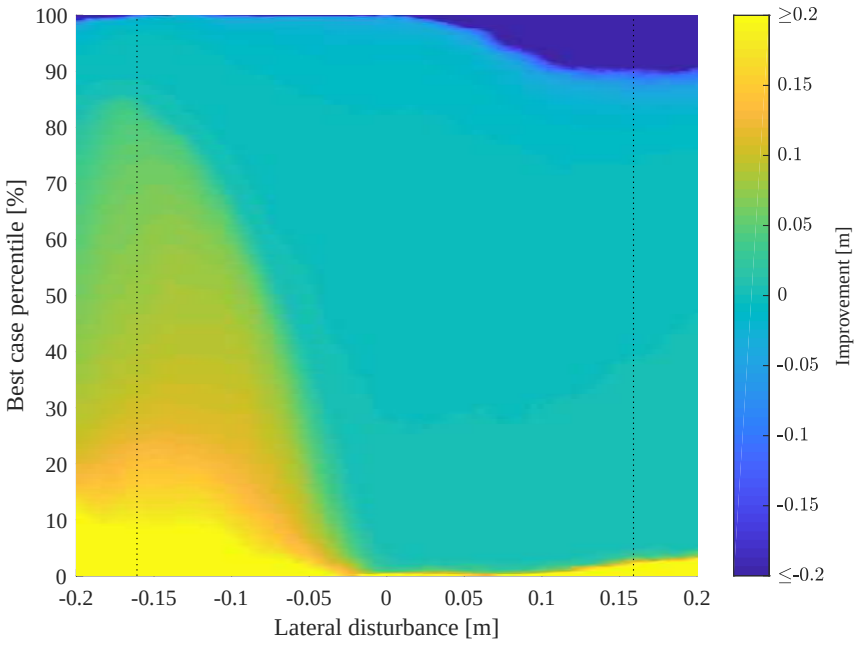

Figure 22: Image of the sorted improvement over 9 000 training trials. To obtain a dense and homogeneous dataset, the 10 disturbances simulated for each of the 900 initial conditions are used to linearly interpolate the performance over all possible disturbance. Where needed (outside the dotted lines), the data is also extrapolated linearly. The interpolated performances of the situation with and without CMAC are compared to obtain the improvement in meters. A positive improvement indicates a distance closer to the target. For each value of the disturbance, the improvements are sorted and displayed by color.

4.2.2 Conclusion for experiment II

The results from experiment II yield mixed results. As can be seen in figure 20, the improvement of the trials has a bimodal distribution. On the one hand, in 70 % of the cases, the performance of the robotic leg is unchanged or decreases slightly by about 1 cm. On the other hand, in 23 % of the cases the performance is increased between 2 cm and 20 cm.

Looking at figure 19, a large portion of the 70 % unaffected trials can be explained by at least two factors. Firstly, there are trials that already reach the target without the CMAC and therefore can not be improved. Secondly when the lateral disturbance is increased sufficiently , the trajectory of the ball no longer intersects with the reachable space of the leg. In these cases the leg misses the ball and the performance can not be improved, regardless of the addition of the CMAC.

Still, there are arguments to support the hypothesis that the artificial cerebellum does improve the performance of the robotic leg. Including all trials, the average improvement is positive with 2.25 cm. If only negative disturbances are considered, the average increases to 5.74 cm. In both cases, the mode for the improved trials is 7cm.

With the overall positive improvement, we can conclude that the artificial cerebellum can increase performance by reducing delays.

5 Discussion and recommendations

The cerebellum is a remarkable structure, supporting a great number of motor functions inn the human body. From an engineering point of view, the cerebellum seems a promising system and has inspired numerous models replicating its structure or behavior. The arguments for using the cerebellum as an inspiration are numerous. Some prize the predictive nature of the cerebellum, making it well suited to compensate delays [Paulin 1989; Miall et al. 1993]. Others value its learning abilities and leverage its structure for quick error correction and correct timing [Bamford et al. 2012; Luque et al. 2011].

Previous research into the possible engineering application of an artificial cerebellum has revealed several opportunities [Duijneveldt 2018]. It has been shown that the cerebellum is equipped with the neurological hardware to perform the following tasks:

- Predict future events (and their timing) created by either the environment or the creatures own actions [Bell, Han, and Sawtell 2008; Ebner and Pasalar 2008; Miall et al. 1993].
- Recognize patterns through the Purkinje cells, modulated by the molecular layer interneurons and Golgi cells [Albus 1975; Bell, Han, and Sawtell 2008].
- Undergo supervised learning from the cortex [Caligiore et al. 2017].
- Adapt to a teaching signal with long delays (80 ms) and wide variation (σ = 180 ms) [Safo and Regehr 2008].

From this it is reasonable to suggest that *the cerebellar architecture is able to accelerate decision making using contextual clues and learn from delayed feedback from the cortex.* To investigate this idea, two hypotheses were formulated. Firstly that the cerebellum is a suitable architecture for the effective prediction of controller actions, and secondly, that an artificial cerebellum can increase performance by reducing delays. To test these hypotheses, a robotic system with an artificial cerebellum has been designed, following requirements below:

- A simulated environment was used to facilitate communication between the different setup components and reduce the cost of the research.
- The setup contained both kinematic and dynamic components to ensure a sufficiently complex system to discern learning.
- A disturbance was introduced to the setup to provide a cue and incentive to react and learn.
- Appropriate delays were included to represent reaction times. In a real situation, latencies can originate from sensors, actuators and controller computation time.
- The artificial cerebellum must learn to anticipate controller corrections through supervised learning. This was necessary to test the first hypothesis.

- The setup must produce a measurable performance. This was necessary to compare the situations with and without artificial cerebellum, and thereby test the second hypothesis.

These requirements were met by a simulation of a humanoid robotic leg. The objective of the setup was to kick an incoming ball to a target located at 10 meters from the robot.

The first experiment conducted in this thesis was designed to test whether the cerebellum is a suitable architecture for the effective prediction of controller actions. The first setup, which was developed to test the first hypothesis, consisted of a robotic leg tasked with kicking a ball to a target located at 10 meters. By introducing a lateral disturbance to the position of the ball just before the impact, the robot is forced to perform a last-second correction. The artificial cerebellum, modeled after the Cerebellar Model Articulation Controller (CMAC) by J.S. Albus, was tasked to anticipate the performed correction.

The results of the first experiment, presented in chapter 4, show a decreasing error between the desired and actual output of the CMAC (see figure 15). This is indicative of the CMAC learning to replicate the desired signal. Additionally, the output of the artificial cerebellum can be seen in figure 16. Although the produced output showed less sharp transitions, the general shape and timing of the signal was reproduced. The lack in sharpness of the produced signal is likely a result of the limited speed of the input state through the input space relative to the size of the receptive region. This prevents big changes in the output but also provides robustness to noise in the trained weights.

In conclusion, the performance of the CMAC, as trained in the first experiment, is conform the expected performance. Therefore we can conclude the the cerebellum is a suitable architecture for the prediction of effective controller actions.

The second experiment conducted in this thesis was designed to test the second hypothesis: Can an artificial cerebellum increase the performance of a robotic leg passing a ball by reducing delays? For this hypothesis, the artificial cerebellum trained in the first experiment, was implemented in the setup. Different trials with the same initial conditions were simulated with and without the added effect of the artificial cerebellum. The performance of these situations was compared in chapter 4.

When the results of the second experiment were analyzed, a dichotomy was present in the improvements. In some case the performance of the setup, in kicking the ball to the desired target, was improved. In other cases, the performance seemed to decrease significantly. This dichotomy is very much visible in figure 22. When the lateral disturbance is negative (closer to the robot) a clear improvement can be observed in up to 80 % of the cases. The improvement in

distance to the target in these cases is around 7 cm. If the lateral disturbance is positive (away from the robotic leg), the addition of the artificial cerebellum does not seem to improve the results. In about 10 % of the cases the performance robotic setup is significantly reduced by more than 20 cm. These cases are most likely to occur when the trajectory of the ball is at the edge of the reachable domain of the robotic leg. The combination of the artificial cerebellum and correction controller seems to be less efficient in finding the right trajectory for the leg to kick the ball. If in these cases the leg misses the ball (compared to previously hitting the ball without the addition of the artificial cerebellum), the performance is greatly reduced.

In conclusion, the addition of an artificial cerebellum may increase the performance of a robotic leg kicking a ball to a target. However, the controllers provided to perform the desired correction must be compatible with the predictive action of the artificial cerebellum.

Suggestions for possible improvements or further research would include the following:

- A change in the control signal to be learned by the CMAC. A more rounded signal in accordance with the possible rate of change in the CMAC is likely to produce better learning.
- The controller used to perform the correction is very simple and very likely to be sub optimal. An improvement in the correction controller may result in better performance even for positive lateral disturbances to te position of the ball.
- Adaptive discretization of the input states may increase the performance of the CMAC. The discretized states are scaled linearly with the input states. Additional nonlinearity may increase the sensitivity in critical regions of the state space and increase the possible rate of change of the CMAC output.
- The sensitivity to less relevant dimensions could be reduced by decreasing the number of dimensions. Possible solution are the projection of the state space into a lower dimension or the extraction of modes or singular values. By reducing the number of dimension, the CMAC may also perform better in unfamiliar situations.

6 Appendix

6.1 Trigonometric identities

$$\cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta) = \cos(\alpha - \beta) \tag{66}$$

$$\cos(\alpha + \beta)\cos(\beta) + \sin(\alpha + \beta)\sin(\beta) = \cos(\alpha) \tag{67}$$

$$\sin(\alpha)^2 + \cos(\alpha)^2 = 1 \tag{68}$$

References

Albus, J. S. (1975). "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)". In: *Journal of Dynamic Systems, Measurement, and Control* 97.3, p. 220. ISSN: 00220434. DOI: 10.1115/1.3426922. URL: http://dynamicsystems.asmedigitalcollection.asme.org/article.aspx?articleid=1402197.

Bamford, Simeon A. et al. (2012). "A VLSI field-programmable mixed-signal array to perform neural signal processing and neural modeling in a prosthetic system". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 20.4, pp. 455–467. ISSN: 15344320. DOI: 10.1109/TNSRE.2012.2187933.

Bell, Curtis C., Victor Han, and Nathaniel B. Sawtell (2008). "Cerebellum-Like Structures and Their Implications for Cerebellar Function". In: *Annual Review of Neuroscience* 31.1, pp. 1–24. ISSN: 0147-006X. DOI: 10.1146/annurev.neuro.30.051606.094225. URL: http://www.annualreviews.org/doi/10.1146/annurev.neuro.30.051606.094225.

Bell, Curtis C. et al. (1997). "Synaptic plasticity in a cerebellum-like structure depends on temporal order". In: *Nature* 387.6630, pp. 278–281. ISSN: 00280836. DOI: 10.1038/387278a0.

Brand, S., A. L. Dahl, and E. Mugnaini (1976). "The length of parallel fibers in the cat cerebellar cortex. An experimental light and electron microscopic study". In: *Experimental Brain Research* 26.1, pp. 39–58. ISSN: 00144819. DOI: 10.1007/BF00235248.

Caligiore, Daniele et al. (2017). "Consensus Paper: Towards a Systems-Level View of Cerebellar Function: the Interplay Between Cerebellum, Basal Ganglia, and Cortex." In: *Cerebellum (London, England)* 16.1, pp. 203–229. ISSN: 1473-4230. DOI: 10.1007/s12311-016-0763-3. URL: http://www.ncbi.nlm.nih.gov/pubmed/26873754http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5243918.

Dieudonné, S and A Dumoulin (2000). "Serotonin-driven long-range inhibitory connections in the cerebellar cortex." In: *The Journal of neuroscience : the official journal of the Society for Neuroscience* 20.5, pp. 1837–48. ISSN: 1529-2401. URL: http://www.ncbi.nlm.nih.gov/pubmed/10684885.

Duijneveldt, Joris van (2018). *Intention prediction and contextual error correction*. Tech. rep. Delft: Delft University of Technology.

Ebner, Timothy J. and Siavash Pasalar (2008). "Cerebellum predicts the future motor state". In: *Cerebellum* 7.4, pp. 583–588. ISSN: 14734222. DOI: 10.1007/s12311-008-0059-3. arXiv: NIHMS150003.

Eccles, John C., Masao Ito, and János Szentágothai (1967). *The Cerebellum as a Neuronal Machine*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-662-13149-7. DOI: 10.1007/978-3-662-13147-3. URL: http://link.springer.com/10.1007/978-3-662-13147-3.

Herculano-Houzel, Suzana (2009). "The human brain in numbers: a linearly scaled-up primate brain". In: *Frontiers in Human Neuroscience*. DOI: 10.3389/neuro.09.031.2009.

Ito, Masao (2006). "Cerebellar circuitry as a neuronal machine." In: *Progress in neurobiology* 78.3-5, pp. 272–303. ISSN: 0301-0082. DOI: 10.1016/j.pneurobio.2006.02.006. URL: http://www.ncbi.nlm.nih.gov/pubmed/16759785.

Iwaniuk, Andrew N., Douglas R. Wylie, and Louis Lefebvre (2009). "The Comparative Approach and Brain-Behaviour Relationships: A Tool for Understanding Tool Use". In: *Canadian Journal of Experimental Psychology* 63.2, pp. 150–159. ISSN: 11961961. DOI: 10.1037/a0015678.

Kane, T. R. and C. F. Wang (1965). "On the Derivation of Equations of Motion". In: *Journal of the Society for Industrial and Applied Mathematics* 13.2, pp. 487–492. ISSN: 0368-4245. DOI: 10.1137/0113030. URL: http://epubs.siam.org/doi/10.1137/0113030.

Kolb, Bryan and Ian Q. Whishaw (2008). *Fundamentals Of Human Neuropsychology*. Sixth. Worth Publishers. ISBN: 0716795868. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3592660/.

Lainé, Jeanne and Herbert Axelrad (1998). "Lugaro cells target basket and stellate cells in the cerebellar cortex." In: *Neuroreport* 9.10, pp. 2399–403. ISSN: 0959-4965. DOI: 10.1097/00001756-199807130-00045. URL: http://www.ncbi.nlm.nih.gov/pubmed/9694235.

Llinas, Rodolfo and Mario Negrello (2015). "Cerebellum". In: *Scholarpedia* 10.1, p. 4606. ISSN: 1941-6016. DOI: 10.4249/scholarpedia.4606. URL: http://www.scholarpedia.org/article/Cerebellum.

Luque, N R et al. (2011). "Adaptive cerebellar spiking model embedded in the control loop: context switching and robustness against noise." In: *International journal of neural systems* 21.5, pp. 385–401. ISSN: 0129-0657. DOI: 10.1142/S0129065711002900. URL: http://www.worldscientific.com/doi/abs/10.1142/S0129065711002900http://www.ncbi.nlm.nih.gov/pubmed/21956931.

Martin, Henry Newell (1884). *The human body; an elementary text-book of anatomy, physiology and hygiene ... by H. Newell Martin.* New York, Henry Holt and company. DOI: 10.5962/bhl.title.27926. URL: http://www.biodiversitylibrary.org/bibliography/27926.

Miall, R. C. et al. (1993). "Is the cerebellum a smith predictor?" In: *Journal of motor behavior* 25.3, pp. 203–16. ISSN: 0022-2895. DOI: 10.1080/00222895.1993.9942050. URL: http://www.tandfonline.com/doi/abs/10.1080/00222895.1993.9942050http://www.ncbi.nlm.nih.gov/pubmed/12581990.

Missura, Marcell and Sven Behnke (2015). "Balanced Walking with Capture Steps". In: *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, pp. 3–15. ISBN: 9783319186146. DOI: 10.1007/978-3-319-18615-3_1. URL: http://link.springer.com/10.1007/978-3-319-18615-3{_}1.

Morgan, B. J. T. and L. Devroye (1988). "Non-Uniform Random Variate Generation." In: *Biometrics*. ISSN: 0006341X. DOI: 10.2307/2531615.

Nunome, Hiroyuki et al. (2006). "Segmental dynamics of soccer instep kicking with the preferred and non-preferred leg". In: *Journal of Sports Sciences*

24.5, pp. 529–541. ISSN: 0264-0414. DOI: 10.1080/02640410500298024. URL: http://www.tandfonline.com/doi/abs/10.1080/02640410500298024.

Parks, P.C. and J. Militzer (1991). "Improved Allocation of Weights for Associative Memory Storage in Learning Control Systems". In: *IFAC Proceedings Volumes* 24.8, pp. 507–512. ISSN: 14746670. DOI: 10.1016/S1474-6670(17)54222-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667017542226.

Paulin, Michael (1989). "A Kalman Filter Theory of the Cerebellum". In: *Dynamic Interactions in Neural Networks: Models and Data*, pp. 239–259. ISBN: 978-1-4612-4536-0. DOI: 10.1007/978-1-4612-4536-0_15. URL: http://link.springer.com/10.1007/978-1-4612-4536-0{_}15.

Pinzon-Morales, Ruben-Dario and Yutaka Hirata (2015). "A realistic bi-hemispheric model of the cerebellum uncovers the purpose of the abundant granule cells during motor control". In: *Frontiers in Neural Circuits* 9.May. ISSN: 1662-5110. DOI: 10.3389/fncir.2015.00018. URL: http://journal.frontiersin.org/article/10.3389/fncir.2015.00018.

Plagenhoef, Stanley, F. Gaynor Evans, and Thomas Abdelnour (1983). "Anatomical Data for Analyzing Human Motion". In: *Research Quarterly for Exercise and Sport*. ISSN: 21683824. DOI: 10.1080/02701367.1983.10605290.

Safo, Patrick and Wade G. Regehr (2008). "Timing dependence of the induction of cerebellar LTD." In: *Neuropharmacology* 54.1, pp. 213–8. ISSN: 0028-3908. DOI: 10.1016/j.neuropharm.2007.05.029. arXiv: NIHMS150003. URL: http://www.ncbi.nlm.nih.gov/pubmed/17669443http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2266067.

Sultan, Fahad and Mitchel Glickstein (2007). "The cerebellum: Comparative and animal studies". In: *Cerebellum* 6.3, pp. 168–176. ISSN: 14734222. DOI: 10.1080/14734220701332486. URL: http://link.springer.com/10.1080/14734220701332486.

Vos, B P et al. (1999). "Parallel fibers synchronize spontaneous activity in cerebellar Golgi cells." In: *The Journal of neuroscience : the official journal of the Society for Neuroscience* 19.11, RC6. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.19-11-j0003.1999. URL: http://www.ncbi.nlm.nih.gov/pubmed/10341267.

Wyatt, Krysta D., Patima Tanapat, and Samuel S.H. Wang (2005). "Speed limits in the cerebellum: constraints from myelinated and unmyelinated parallel fibers." In: *The European journal of neuroscience* 21.8, pp. 2285–90. ISSN: 0953-816X. DOI: 10.1111/j.1460-9568.2005.04053.x. arXiv: NIHMS150003. URL: http://www.ncbi.nlm.nih.gov/pubmed/15869526http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1201546.