FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Study and Simulation of Humanoid Robot Walking Algorithms

João Luis Pinto Rebelo

(ee09010@fe.up.pt)

Integrated Master in Electrical and Computer Engineering - Major in Automation, Specialization in Robotics

Supervisor: Paulo José Cerqueira Gomes da Costa (PhD)

July 2010

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2009/2010

A Dissertação intitulada

"STUDY AND SIMULATION OF HUMANOID ROBOT WALKING ALGORITHMS"

foi aprovada em provas realizadas em 21/Julho/2010

o júri

Presidente Professor Doutor José Carlos dos Santos Alves
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Andre Gustavo Scolari Conceicão
Professor do Departamento de Engenharia Elétrica da Escola Politécnica - UFBA

Professor Doutor Paulo José Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto (Orientador)

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - **João Luís Pinto Rebelo**

Faculdade de Engenharia da Universidade do Porto

Abstract

Humanoid robots are anthropomorphic mechanical structures that aim to mimic human capabilities and perform human-like tasks. Walking is one of the most important features of these machines since it allows them to move on irregular ground, climb stairs or step over obstacles.

The control of the humanoid walking movement is, however, a challenging task due to the many degrees-of-freedom and highly unstable dynamics. Currently, the most used solution for gait generation is based on the stability criteria known as Zero-Moment Point. Nevertheless, this method requires the complex robot dynamics to be modelled, generates non-optimal patterns and is not adaptable to highly dynamic environments.

For this reason new control techniques based on Central Pattern Generators (CPG) are being proposed. Using this method, no dynamic analysis is required, no trajectory is generated before hand and the robot is fully controlled by the limit cycle behavior of the neural oscillator. When implementing CPG systems architecture factors such as the oscillators mathematical model and the coupling typology have to be considered. So far no analysis was done to compare the performance of different non-linear oscillator models and there is no widely accepted architecture for this specific purpose.

In this work a simulated humanoid model is created using the SimTwo and Webots simulation environments. Using these simulators a simple trajectory-based gait is implemented and optimized using a genetic algorithm. This method shows that the two robots are capable of performing a walking motion.

Using the hip signals from the trajectory-based gait, an experimental comparison between the Van der Pol, Rayleigh, Duffing and Matsuoka oscillators performance is done. The best results were obtained by using the Matsuoka oscillator, which demonstrates that this is the most convenient model for this task. Connecting this oscillator model in a chain architecture, a genetic algorithm optimization is executed to determine the parameters for performing a walking motion. The final optimization was innefective and due to the convergence to a movement that did not correspond to a walking motion. To overcome this problem a supervised learning method, using the trajecory-based signals as reference, is used to entrain the Matsuoka oscillators. The obtained parameters are used to generate control signals that can effectively produce forward walking movement.

The results obtained in this work demonstrate the feasibility of using CPG-based techniques to perform biped locomotion. Nonetheless, the generated walk still shows some shortcomings and further improvements need to be done. A number of shortcoming were also found in the SimTwo simulator, which demonstratates that some of its functionalities, such as colision detection and friction models, still need to be refined.

Resumo

Robôs humanoides são estrutrutas antopomórficas cujo objectivo é reproduzir capacidades humanas e executar tarefas humanas. Caminhar é uma das propriedades mais importantes deste tipo de mecanismos, dado que lhes permite movimentarem-se em superfícies irregulares, subir ou descer escadas e ultrapassar vários obstáculos.

O controlo deste tipo de movimento é, no entanto, uma tarefa complexa devido aos muitos graus de liberdade e dinâmica inerentemente instável. Actualmente, a solução mais utilizada para gerar caminhadas é baseada no critério de estabilidade designado por Zero-Moment Point. No entanto, este método requer que a dinâmica do robô seja modelada, apresenta resultados ineficientes e não é facilmente adaptável a ambientes dinâmicos.

Por este motivo, novas técnicas de controlo baseadas em Central Pattern Generators (CPG) têm sido propostas. Utilizando estes métodos, não são necessárias análises da dinâmica, nenhuma trajectória é gerada antecipadamente e o movimento do robô é totalmente controlado pelo ciclo limite do oscilador. Quando se implementam sistemas baseados em CPG, factores como o modelo matemático dos osciladores e a topologia dos acoplamentos têm que ser considerados. Até á data nenhuma análise comparativa entre os vários modelos de osciladores foi feita e não existe uma arquitectura bem definida para este tipo de tarefa.

Neste trabalho um humanoide simulado em SimTwo e Webots foi utilizado para testar os algoritmos implementados. Numa primeira aproximação, uma caminhada baseada em trajectórias predefinidas foi implementada e optimizada utilizando um algoritmo genético. Este método demonstra que ambos os robôs são capazes de efectuar caminhadas.

Utilizando os sinais de controlo da anca obtidos a partir do método anterior, realizou-se uma comparação experimental entre os osciladores Van der Pol, Rayleigh, Duffing e Matsuoka. Os melhores resultados foram obtidos pelo oscilador Matsuoka o que demonstra que este modelo é o mais conviniente para este tipo de tarefa. Uma arquitectura baseada em CPG, utilizando o modelo de Matsuoka para gerar os sinais de controlo, foi implementada e um algoritmo genético utilizado para determinar os paramêtros necessários para gerar a caminhada. A optimização resultou numa convergência para um movimento errado, revelando-se este método ineficiente. Para ultrapassar esta limitação, um método de aproximação ás formas de onda anteriormente obtidas foi implementado para determinar os paramêtros do oscilador. Utilizando estes paramêtros foi possivel criar sinais que geram um movimento de caminhar.

Os resultados obtidos neste trabalho demonstram que é possível utilizar técnicas baseadas em CPG para realizar movimentos de caminhada. No entanto, as trajectórias geradas demonstram ainda alguns problema e, como tal, algumas melhorias são necessárias até este ser um método eficaz. O simulador SimTwo apresenta também limitações em algumas funcionalidades, nomeadamente na detecção de colisões e nos modelos de fricção que têm que ser corrigidas.

Acknowledgements

Primeiro gostaria de agradecer ao Prof. Paulo Costa não só por ter aceite ser o meu orientador neste tópico tão complexo, mas também por toda a disponibilidade demonstrada ao longo destes meses para longas conversas e pelos bons conselhos para enfrentar os pequenos e grandes obstáculos que foram surgindo ao longo do caminho.

Um grande obrigado também para todos os meus amigos que me ajudaram a manter a boa disposiçao e motivação ao longo deste trabalho. I would specially like to thank Manuel Cabral for the endless but always fruitful discussions about genetic algorithms, machine learning, satelittes and many other work-unrelated topics; to Sara Siconolfi for always having the right sentence on the right moment, and, most of all, for the fantastic friendship; às gentes de Lamego, em especial ao Tó Mané, Rodrigo e Ricardo, porque, por muitas voltas que a vida dê, os velhos amigos estão sempre presentes. To all the great friends that I am very lucky to have and are now in Denmark, Netherlands, Spain, Portugal and a bit all over this small world of ours, I can not name you all, but thank you very much.

A very special thank you goes to my girlfriend Adéla, because I am absolutely sure that without her unconditional help, support and patience, not only this thesis but a lot more would not have been possible.

Por fim, mas não menos importante, um enorme obrigado a toda a minha família. Em especial ao meu irmão pela excelente companhia, aos meus pais, cuja ajuda, compreensão e muita paciência tornou tudo isto possível e aos meus avós, simplesmente por serem como são e terem sempre dado um óptimo exemplo de esforço e dedicação.

O Autor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humanoid robots are anthropomorphic mechanical structures that aim to mimic human capabilities and perform human-like tasks. Several areas of study are in focus in humanoid robots, such as object recognition, artificial intelligence, man-machine interaction and locomotion planning and control.

The main motivation for the use of this type of robot is their suitability for performing tasks in a typical "human environment", due to their capability to perform not only human-like manipulation but also human-like locomotion [1]. Therefore, the goal of the studies in this area is to achieve human dexterity, efficiency, stability, effectiveness and flexibility [2].

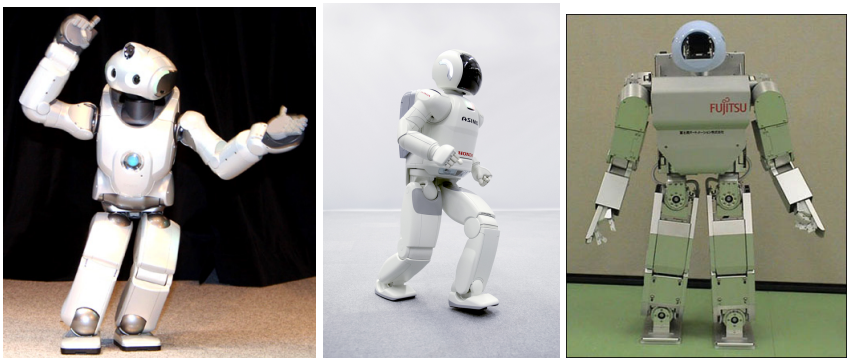
Walking is a particularly important feature of these machines, since it allows them to move on uneven ground or step over obstacles, with the advantage that feet usually provide a lot more friction with the surface when compared to a wheel. Furthermore wheeled robots are in general limited to horizontal movements, while a machine with legs can climb a ladder or use steps to travel up- or downwards [3].

This area has witnessed solid advances from the single leg hopping mechanism and quadrupeds introduced in 1986 [4] to the current commercial humanoid robots with the ability to walk and interact with the surrounding environment and even with people. Some of the most remarkable examples are Sony's QRIO, Honda's ASIMO and Fujitsu's HOAP-2, shown in Figure 1.1.

The control of the humanoid walking movement is a challenging task due to the many degrees-of-freedom and the nonlinear and hard to stabilize dynamics [2] and a reliable, flexible and widely accepted solution to solve this problem is still to be found.

Currently, the most used solution for gait generation is based on the stability criteria known as Zero-Moment Point. Nevertheless, this method requires the complex robot dynamics to be modeled and the generated patterns are non-optimal and not easily adaptable to highly dynamic environments [8]. For this reason new control techniques based on Central Pattern Generators (CPG) are being proposed.

CPG are constituted by coupled non-linear oscillators that can be used to generate robust and highly adaptable gaits. For the design of these systems factors such as architecture, oscillator mathematical model, coupling typology and the choice between torque and position control have



(a) (b) (c)

Figure 1.1: Humanoid robots (a) Sony's QRIO (Source:[5]) (b) Honda's ASIMO (Source:[6]) and (c) Fujitsu's HOAP-2 (Source:[7])

to be considered. All these design considerations are strongly interconnected and each of them has a strong impact in the gait generation performance.

Some oscillator models and architectures were proposed but, so far, no comparative work was done as to determine which model presents the best performance for this specific task. This is of the uttermost importance since the selection of a convenient oscillator is critical for the development of the more advanced locomotion capabilities of the humanoid robot.

With this in mind, the main objective of this work is to simulate different walking algorithms for humanoid robots. The focal point is on the use of the recently proposed architectures and control methods based on Central Pattern Generators. The specific objectives are: a) compare the performance of the different non-linear oscillator models and b) demonstrate the gait generation capabilities of the selected CPG oscillator and architecture.

To achieve the proposed objectives the first task for this project is to create a suitable test environment for the gait generation algorithms. Due to the high complexity and cost of real humanoid robots, two dynamics simulation environments based on a realistic physics engine are used. One of the used simulators is SimTwo [9], which is currently being developed at FEUP. The other is the commercial simulator Webots that is the basis for the humanoid simulation competition Robostadium. The use of these two simulators makes it possible to compare, validate and suggest improvements for SimTwo.

After implementing the simulation environments, a simple trajectory-based gait generation method is used to demonstrate the robot locomotion capabilities and generate reference trajectories. Using the reference signals a comparison between the studied non-linear oscillators is done, to determine which model can better approximate the reference signals.

The most suitable oscillator model is then implemented in a chain architecture and used to generate forward walking gaits for the humanoid robot. This demonstrates the possibility of doing sucessful locomotion control using a CPG.

According to the proposed objectives and methodology, this thesis is organized as follows: Chapter 2 presents a review of the work currently done using Central Pattern Generators for hu-

manoid robot locomotion; The theoretical background for the used simulation environments, optimization methods and low-level motor control is given in Chapter 3; the implemented humanoid model and simulation results are detailed in Chapter 4; the conclusion and open topics for future work are stated in Chapter 5; The SimTwo humanoid model implementation, the non-linear oscillators source code and the NAO DC motors datasheet are shown in Appendices A, B and C, respectively. Also, a CD containing all the relevant source-code, datasheets and an electronic version of this thesis is enclosed.

Chapter 2

Central Pattern Generators: a review

Biped locomotion is one of the major topics in the design and development of humanoid robots and has been the focus of several studies and research, from a variety of viewpoints, over the last decades. The objective of these methods is to generate a stable gait over a complete walking cycle [10]. The gait generation techniques can be broadly divided in four main categories: passive-dynamic, trajectory-based and central pattern generators [11].

In passive-dynamic walking, the biped walks down a slope without the use of any actuator. The mechanical parameters of the legs (e.g. length, mass, foot and shape) determine the characteristic and stability of the generated motion [12]. This idea was first applied in [13] and was used to develop a 2D biped with knee joints and curved feet. Even though this robot could only move down an inclined slope, this work demonstrated that the robot physical construction can be as important as the control system for the walking motion.

Trajectory-based methods consist of finding a set of pre-defined kinematics trajectories using a stabilization criterion to ensure that the gait is stable. Most of the successful walking algorithms implemented nowadays are trajectory-based and use the Zero-Moment Point (ZMP) criterion for motion generation with off-line trajectory determination and on-line balance compensation [14]. This stability criteria was introduced by Vukobratovic and it states that ”as long as the ground projection of the point of the resulting reaction forces, remains inside the polygon formed by the contact point of the feet on the ground, the walking is stable“ [15]. Even though this method is effective to guarantee stability of biped locomotion it requires precise modeling of robot dynamics and, to avoid singularities while solving the legs' inverse kinematics, the generated patterns usually result in a ”bent-knee“ posture. Moreover, due to the regular interaction between bipedal robot and unknown environment during the entire walking cycle, the use of pre-recorded trajectories may result in a poor walking performance [8].

The difficulties of the trajectory-based approach inspired some researchers to look into the biomechanics of the animal and human walking. The concept of Central Pattern Generators (CPG) originated from the neurophysiological studies of biological neural networks that produce coordinated multidimensional rhythmic signals, under the control of simple input signals, which are characteristic of the vertebrate and invertebrate animal motions [16].

In biological systems of legged locomotion, the high-level nervous system and spinal cord rhythmic system control the musculo-skeletal system in a hierarchical manner. The high-level nervous system is responsible for the motion learning and adaptation, i.e. the intelligence. The lower limbs move with a given coordinated motion under the Central Pattern Generators' driving [17]. Central Pattern Generators (CPG) are, in simple terms, networks of coupled non-linear oscillators that produce coordinated control signals. These networks present several properties that make them suitable for motion control [17]:

- They can produce periodic control signals even without any sensor or higher order inputs. This allow a robot to move on flat terrain only using the CPG control signals [18].
- It is a distributed control system in which each CPG unit controls one joint of a robot and the full network coordinates the complete a movement. Depending on the parameters modulation different gaits can be generated. This also allows gait transition during movement to be performed.
- Sensorial and higher level inputs can be connected to the system, thus making the gait adapt to the environment. Therefore, using the CPG inspired method, robots can either walk on flat terrain with an open loop control or adaptively walk on irregular terrain with a closed loop control [19].

Using this method, no dynamic analysis is required, no trajectory is generated before hand and the robot is fully controlled by the limit cycle behavior of the neural oscillator. This approach was explored not only to control bipeds [8, 20, 21, 18, 22], but also quadrupeds [23, 24], hexapods [25], centipedes [26] and other types of bio-inspired robots [27]. Nonetheless, the relation between the parameters and the generated outputs are difficult to determine due to the non-linear nature of the oscilators and unpredictable and potentially unstable behaviour can occur [28].

To apply CPG models to robot motion control, three main points have to be considered [29, 28]:

1. CPG model selection. Before constructing a CPG network an appropriate CPG model, depending on the desired type of locomotion, must be selected.
2. CPG network architecture. The number of CPG units, the type of couplings and the overall architecture must be defined.
3. Modulating parameters. The network output signals' characteristics and, consequently, the generated gait, are dependent on the CPG parameters values. Therefore, these parameters must be modulated according to the desired type of gait to be performed.

2.1 Oscillator models

Several different non-linear oscillator models have been used to implement biped locomotion, being the most common the Matsuoka [8, 30, 20], Van der Pol [24, 31] and Rayleigh [31]. The

Duffing oscillator model was proposed to describe nonlinear vibration modes and it has not been used to study gait generation processes. Nonetheless it shows the same limit cycle behavior as the previous oscillators.

2.1.1 Van der Pol oscillator

The Van der Pol oscillator is a type of relaxation oscillator originally used to describe limit cycles in electrical circuit using vacuum tubes [28]. The differential Equation 2.1 describes the response of this type of oscillator.

$$\ddot{x} + a\left(x^2 - p^2\right)\dot{x} + w^2 x = 0 \tag{2.1}$$

where:

- x, $\dot{x}$ and $\ddot{x}$ describe the states of the system
- $a > 0$ and p are the coefficients of the resistance
- w represents the oscillator input gain

The resistance is negative for a small amplitude of x, as given by $x^2 - p^2$, and is responsible for the generation of self-sustained oscillations.

Equation 2.2 shows the oscillator including couplings and feedback signals.

$$\ddot{x} + a\left(x^2 - p^2\right)\dot{x} + \sum_{i=0}^{n} w_i^2 x_i + \sum_{i=0}^{m} w_i^2 f_i = 0 \tag{2.2}$$

where:

- x, $\dot{x}$ and $\ddot{x}$ describe the states of the system
- $a > 0$ and p are the coefficients of the resistance
- f represents the feedback value
- w represents the connection gain

The adaption consists of including the outputs of the connected oscillator and feedback sources affected by a gain factor in the oscillator state calculation. Figure 2.1 shows the typical response of a Van Der Pol oscillator in the presence of a limit cycle.

2.1.2 Rayleigh oscillator

The Rayleigh oscillator is also a relaxation oscillator, very similar to the one presented by Van der Pol, and was created to model the sound generating principles of musical instruments [28]. The key difference between these two oscillator is that when the voltage (input signal) increases,

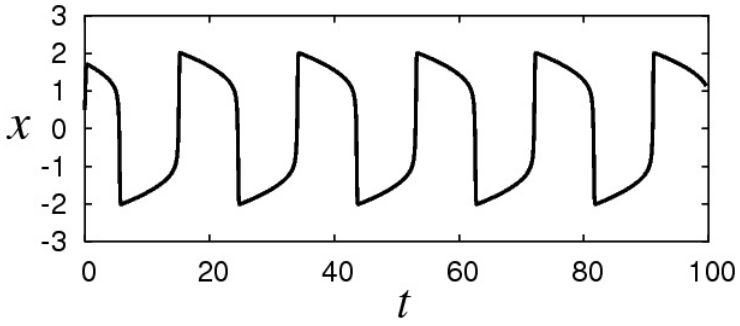


Figure 2.1: Van der Pol oscillator typical output (Source:[32])

the Van der Pol oscillator shows an increase in frequency, whereas the Rayleigh oscillator shows an increase in amplitude. Equation 2.3 details the output behavior of this type of oscillator.

$$\ddot{x} + a\left(\dot{x}^2 - p^2\right)\dot{x} + w^2 x = 0 \tag{2.3}$$

where:

- x, $\dot{x}$ and $\ddot{x}$ describe the states of the system
- $a > 0$ and p are the coefficients of the resistance
- w represents the oscillator input gain

Both the typical response and the inclusion of feedback and coupling between different oscillators is done in the same way as in the Van der Pol oscillator.

2.1.3 Duffing oscillator

From a physical point of view, the Duffing oscillator describe in Equation 2.4 can model the one-mode vibration of a suspended elastic cable driven by a periodic forcing [33].

$$\ddot{x} + p\dot{x} + qx + \beta x^2 + rx^3 = scos\left(\omega t\right) \tag{2.4}$$

where:

- p is the measure of damping
- β and r are nonlinearities
- q is the natural frequency
- s and ω represents the excitation signals' amplitude and frequency

This oscillator present the same limit cycle properties as the previously presented oscillators, however, due to the higher order terms it has richer dynamics. This type of response can potentially be used to effectively model the target joint angles needed for biped locomotion. The coupling and

feedback are included as in the Van der Pol and Rayleigh oscillator. Figure 2.2 shows an example of the Duffing oscillator response.

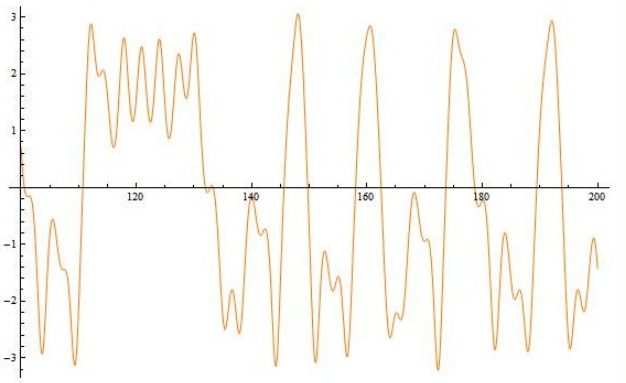


Figure 2.2: Duffing oscillator output example (Source: Adapted from [33])

2.1.4 Matsuoka oscillator

The most widely used CPG for humanoid locomotion is the one presented and analyzed by Matsuoka [34]. Equations 2.5 to 2.7 represent a neuron of a Matsuoka oscillator composed by n elements.

$$T_r \frac{dx_i}{dt} + x_i = -\sum_{n}^{j=1} a_{ij} y_j + s_i - b f_i \tag{2.5}$$

$$y_i = max(0, x_i) \tag{2.6}$$

$$T_a \frac{df_i}{dt} + f_i = y_i \tag{2.7}$$

where:

- x is a membrane potential of the neuron body
- T_r the rise time constant for a given step input
- a_{ij} is a weight of inhibitory synaptic connection from the $j-th$ neuron to the $i-th$ neuron
- s an impulse "rate" of the tonic or slowly varying input
- b the parameter that determines the steady-state firing rate for a constant input
- f is the degree of fatigue or adaption in the neuron
- y a "firing" rate or output of the neuron
- T_a the adaptation time constant

The feedback is included in the same way as the oscillator coupling by adding the feedback inputs to values multiplied by the inter-coupling a matrix.

Since the human motion is typical commanded by the response of two muscles, i.e. the extensor and the flexor, the two-neuron network typically used for movement control with each of the elements modeling the response of either muscle. This type of configuration is shown in Figure 2.3

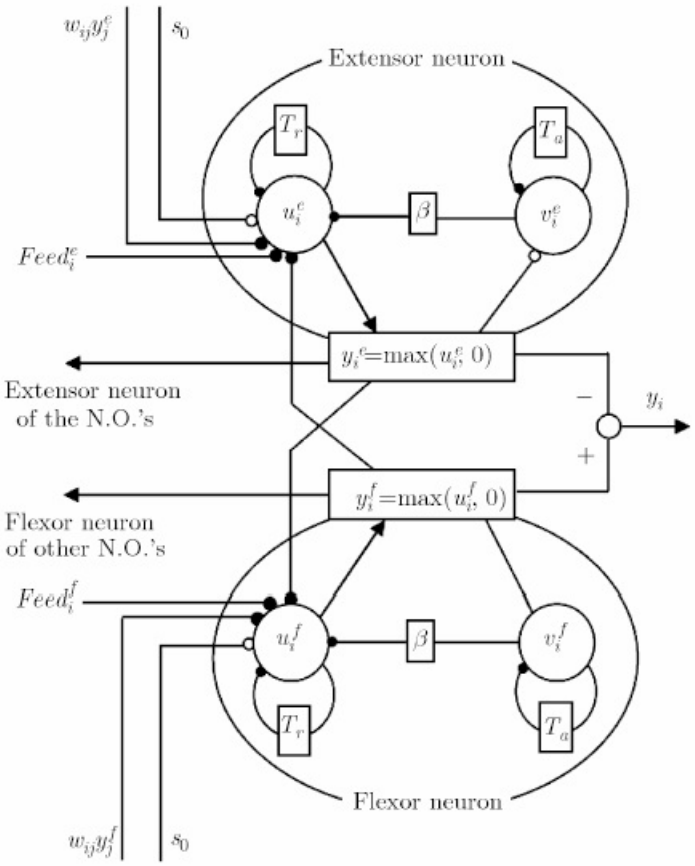


Figure 2.3: Two-neuron matsuoka configuration (Source:[28])

Figure 2.4 shows the output response of a two-neuron network with $T_r = 1, T_a = 12, b = 2.5, a_{12} = a_{21} = 1.5$ and $s_1 = s_2 = 5$.

The use of neural networks has also been attempted in [35, 36] but is a less common approach. The oscillators' mathematical equations are typically implemented in its continuous form and numerical methods are used to compute the solutions for a given set of parameters. Another possibility is the implementation in microcontrollers using a discretized version [37] or using analog hardware components [25]. A comparison between Rayleigh and Van der Pol oscillators for biped locomotion control [31] has shown the better performance of Rayleigh oscillators, however no extensive work was done using also the Matsuoka and other oscillators.

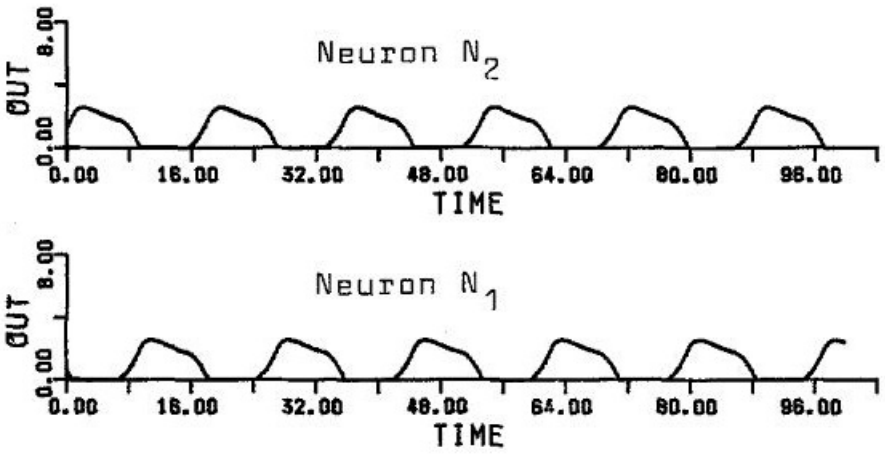
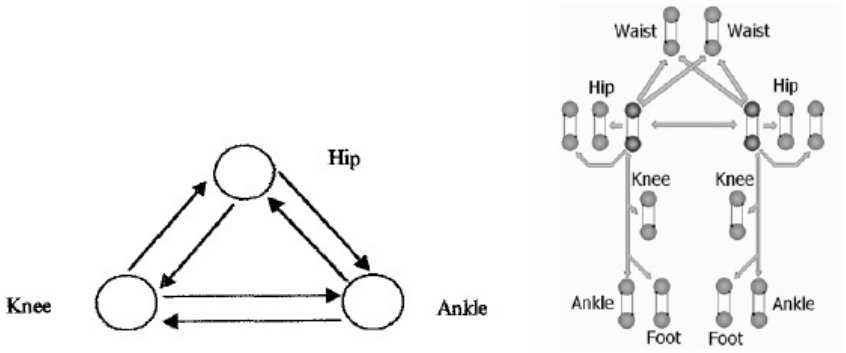


Figure 2.4: Matsuoka oscillator output example (Source:[34])

2.2 Architectures and parameter modulation

Currently, there are two types of CPG architectures found in literature, specifically, ring architecture, i.e the output of each oscillator is connected to the next one, with the last oscillator connected to the first, forming a ring or closed loop [38, 39] and chain along the legs, e.g. the the hip connects to the knee and the knee to the ankle [8, 30], being the latter the most used in biped locomotion. Figure 2.5 shows an example of ring and chain architectures, respectively.



(a) (b)

Figure 2.5: CPG control architectures (a) Ring connection (Source:[39]) and (b) Chain connection (Source:[30])

The use of the motor dynamics as part of the Matsuoka oscillator for generation of the position signal [41] and the introduction of virtual components between the CPG output and the motor control signal [22] have also been tried to improve the stability and reaction to external perturbation showing promising results.

Another property very common to animals and humans being is their capability to seamlessly change the type of gait, e.g. from walking to running. This property is also very little studied for robotic system and the only approach followed until now is the transition between sets of

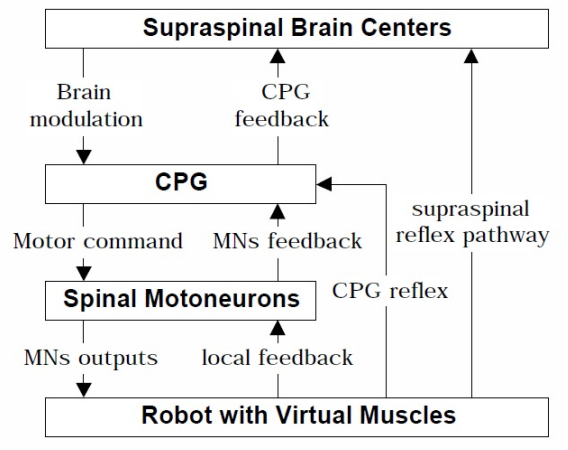


Figure 2.6: Motorneuron control architecture (Source:[40])

pre-defined parameters [42, 24]. This approach has been successfully used perform online gait transitions in quadrupeds but so far not in humanoids. A method to dynamically start a humanoid walking movement by adjusting the value of only of one of the inputs in a Matsuoka oscillator was proposed in [37].

Regardless of the CPG model or functionality that is to be implemented the main challenge is to adjust the parameters that generate the desired output signals. Since these networks are highly nonlinear it is usually very complicated to establish a direct relation between the the parameters and the describing factors of the outputs such as frequency, amplitude, phase relations between the neurons, and the overall waveform of the output signals [28]. The techniques used to determine these parameters can be divided in two main groups, supervised and unsupervised learning [29]. In unsupervised learning an optimization algorithm is used to optimize certain motion parameters. Most commonly found in literature is the optimization of a combination of speed and stability [41, 23], robot posture [43] or the estimated foot position during the walking cycle [44]. This method is not always capable of finding an adequate set of parameters to generate the desired motion. Therefore, the evolution is sometimes done in a stepwise manner, e.g. to evolve first to a simpler objective and then to the complete walking [41] or by including extra supporting structures in the robot to facilitate the walking for the first generations [30]. When supervised learning is used the desired outputs are assumed to be available, either from previous experiments or by adaptation from human measured data [45]. In this case either genetic algorithms are used to minimize the differences between the two data sets [45] or online entrainment algorithms are used [46]. A method to entrain a CPG using existing data was also developed [47] but can only be applied to a specific model.

Even though various studies have been made about CPG and their different features, several research topics are still open in the area of humanoid locomotion. There is currently no defined technique to select control architectures and oscillator models. It is also not clear if the CPG output

should be directly used to control the joint angles or if this represents a higher level of control and a layer should be included for the motor position or torque control. The inclusion of feedback and the online gait adaptation are also still very little studied and without these tools it will not be possible for humanoid robots to regularly operate in dynamically and highly unpredictable environments. Another missing key research is the study of an the end-to-end architecture to determine the relations between all the previously referred subsystems and the overall interaction with the external environment.

Chapter 3

Theoretical background

To provide the necessary framework for this thesis, this chapter briefly describes the theoretical concepts. In this sense, the optimization methods are detailed in Section 3.1. The low-level closed loop control is then presented in Section 3.2. An overview of the used simulation environments is given in Section 3.3.

3.1 Optimization methods

A major difficulty in designing CPG networks is the parameter modulation, due to the non-linear relation between the model parameters and the output signals. There is no well-established design methodology for these parameters determination and the main methods used are trial-and-error, numerical analysis and optimization algorithms [29].

Trial-and-error methods are usually inefficient and result is non-optimal solutions that are only adaptable to very specific situations. Also, due to the difficulty in establishing relationships between the parameters and outputs, numerical analysis methods are very difficult to use to obtain good results. Therefore, the use of optimization algorithms is the most used solution. These methods are also slow and can sometimes lead to sub-optimal solutions but they provide the best method for analysis of these type of systems [28]. Two of the used methods are Hill Climbing and Genetic Algorithms.

3.1.1 Hill climbing

The Hill Climbing algorithm is a simple local search method that is easy to implement and performs well in several situations, achieving reasonable results. The algorithm starts by generating a random (potentially poor) solution, and iteratively making changes to the solution that lead to better results. When the algorithm is not able to do further improvements on the solution, it terminates. Ideally, at this point, the solution is optimal or very close to optimal, even though this can not be guaranteed. Figure 3.1 shows an example of a suitable solution space for applying Hill climbing.

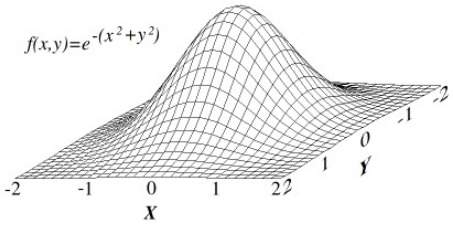


Figure 3.1: Hill climbing solution space example (Source: [33])

This algorithm can easily get stuck in local optimal solution, i.e., none of the neighbors has a better evaluation than the current solution or in an infinite loop through neighbor solutions. Figure 3.2 shows an example of a solution space that can lead to potential problem when using Hill Climbing. These two problems can be addressed by allowing the algorithm to occasionally accept worst solutions and by introducing memories to remember the previously found values.

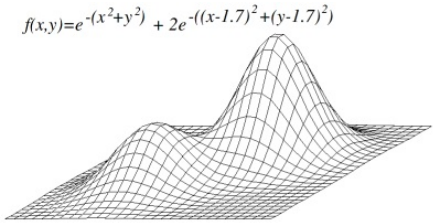


Figure 3.2: Hill climbing solution space example with local optimal solutions (Source: [33])

3.1.2 Genetic algorithms

A genetic algorithm (GA) is an optimization method inspired by the evolution of biological systems and based on global search heuristics and tries to overcome some of the limitations existing in the Hill Climbing and other optimization methods.

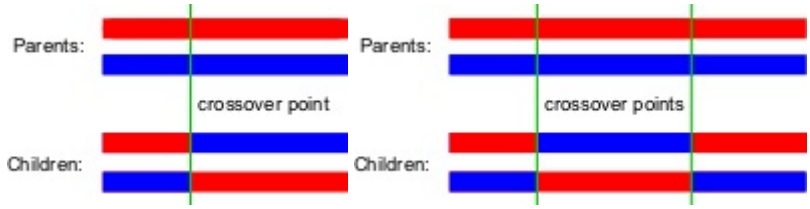
Given an initial population of candidate solutions (called individuals, creatures, or phenotypes), the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population.

The initial population is usually generated randomly, trying to cover most of the search space. Occasionally, the individuals are generated around areas in which the optimal solution is most likely to be found to increase the convergence speed. The population size depends on the problem to be solved, however, they commonly range from 5 to 20 times the number of variables to be optimized.

The fitness function always problem dependent and is calculated for each individual to measure the quality of the represented solution. When different objectives have to be combined, the fitness function is usually composed by a linear combination of each of them, with the weights representing the relative importance of each to the final results.

After evaluating an existing population, a certain number of individuals are selected based on their fitness to be recombined and create a new population. Two of the most used selection methods are roulette-wheel selection and tournament selection. In the roulette-wheel method each candidate solution is attributed a selection probability based on its fitness value. A random individual is then selected. This is analogous to a roulette in which each individual solution represent a pocket with a size proportional to the selection probability, hence the methods' name. In tournament selection a pre-defined number of randomly selected individuals are grouped together, being the one with the best fitness value selected for the next generation. Both of the functions are however stochastic and designed so that less fit solutions are also chosen in order to keep the diversity of the population and so avoid the convergence to local optimal solutions.

A new population is then created by recombining or mutating the selected individuals. The part of the population created by recombination is determined by the crossover ratio. Several crossover techniques exists, being the most used: one point-crossover in which a single crossover point on both parents' organism strings is selected; two-point crossover in which two points are selected on the parent organism and everything between them is swapped. Figure 3.3 shows these two crossover mechanisms.



(a) One-point crossover (b) Two-point crossover

Figure 3.3: Genetic algorithm crossover examples (Source: [33])

The remaing population is generated by mutation. Mutation is a process analogous to biological mutation in which one of the elements of the individual chromosome can change according to a given mutation probability. The purpose of mutation in GA is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution.

The newly created population is then evaluated and the algorithm iterates until a satisfactory solution is found or a stop criteria is reached (e.g. maximum number of generations). The algorithm does not guarantee that a global optimal solution has been found.

Figure 3.4 gives a general overview of the GA cycle.

3.2 Low-level control

When controlling motors, closed-loop controllers are very widely used, in order to make the overall system response immune to the presence of uncertainty and outside disturbances (e.g. battery voltage variations, load variations). The objective is to vary the voltage applied to the

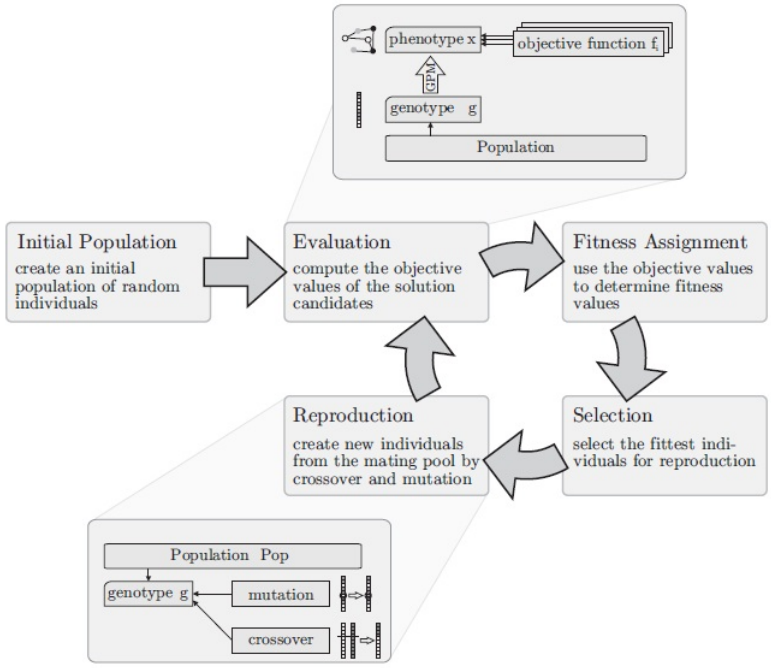


Figure 3.4: Genetic Algorithm cycle (Source:[48])

motor so that it moves at a specified speed or to a specific position. To accomplish this, it's necessary to measure the actual state of the motor and compare it with the desired state. The difference between them is called error and is then applied to a controller which will change the voltage applied to the motor. Figure 3.5 shows the typical closed-loop control block diagram.

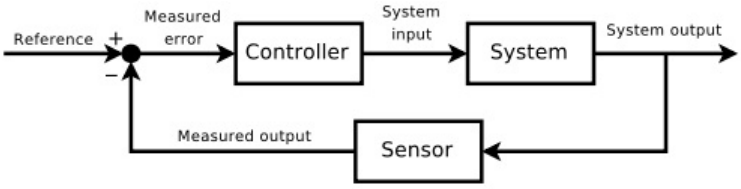


Figure 3.5: Typical position closed-loop control block diagram (Source: [49])

Typically, the controller used is a P, PI or PID controller. A PID controller is constituted by three distinct effects: Proportional, Derivative and Integral (Figure 3.6). Each of these parts has its own effect on the behaviour of the closed loop system

The proportional effect is used to deal with the present error, being K_p called the proportional gain. The higher K_p is, the faster will the system react to error, but a large value of K_p can eventually lead to system instability. The integral effect is used as a "memory" of the system. It integrates the values of the error over time and multiplies it by an integral gain designated by K_i, thus making an average and adding it to the control quantity. This integral effect removes the steady-state error.

The derivative effect is used as a prediction of the future state of the system. The derivative on the current point of the system is multiplied by a constant K_d designated as derivative constant

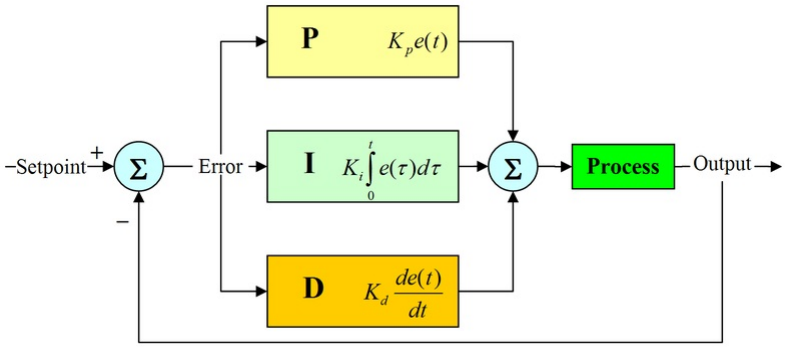


Figure 3.6: PID control (Source: [50])

and summed to the control quantity. The derivative effect is used to make the controller respond quicker to changes in the system. The resulting equation is, thus, (3.1) [51].

$$\begin{aligned} u(t) &= K\left[e(t) + \frac{1}{T_I}\int e(\tau)d\tau + T_D\frac{de(t)}{dt}\right] \\ &= K_p e(t) + K_i\int e(\tau)d\tau + K_d\frac{de(t)}{dt} \end{aligned} \tag{3.1}$$

This is the implementation of the PID controller in an analog/continuous way. But if a microcontroller is being used, it's necessary to implement it in a discrete/digital way using the Z-Transform domain instead of the S-Transform domain. Two possibilities are: to design the whole controller using digital control theory, or to try to discretize the analogical controller [52]. Following the second choice, the discretization can be done for small sampling times T_s. The derivative term, then, can be replaced by a first-order difference expression and the integral term, by a sum. Since the integral term should be a form a sum of all past errors, the error should be stored in every sampling time. The discrete recursive PID controller algorithm, derived from (3.1), becomes (3.2) [51].

$$u(k) = K\left[e(k) + \frac{T_s}{T_I}\sum_{i=0}^{k-1} e(i) + \frac{T_D}{T_s}(e(k) - e(k-1))\right] \tag{3.2}$$

3.3 Simulation environment

To test and analyze the behavior and performance of the implemented gait generation algorithms two dynamics simulation environments were used. The use of simulation environments for development and test in robotics provides has many benefits over the use of real robots. The main advantages of simulations are [11, 53, 28]:

- Easy to develop and test new robot models;
- Possibility to easily add, remove, and test different components;

- Easy testing of new algorithms;
- Less development and testing time;
- All tests can be done without damaging the real robot;
- No human intervention is needed to reinitalize test, which is particularly important for optimization algorithms that require several runs;
- Physical measures can be easily done
- On reliable and realistic simulation environments, the obtained results can be directly or very easily applied to the real robots;
- Control over the simulation time
- Less expensive than real robots

SimTwo [9] is a realistic simulation framework based on the Open Dynamics Engine (ODE) library [54] which provides the low-level dynamic realism by numerically simulating each body "mechanics" based on its physical properties. The simulator models the electrical actuators as DC motors driving hinge, universal or slider joints. Non-linear effects such as saturation, current limit and static and dynamic frictions are also taken into account, as well as user-defined gearboxes and controllers (e.g. PID-position or space-state). An Extensible Markup Language (XML) is used to define the simulator robots and environment.

Webots physical engine is also based on ODE, however it does not simulate the real behavior of DC motors. Each joint has instead a maximum force that can be applied, which corresponds to an ideal DC motor model without any non-linearity. A Virtual Reality Modelling Language (VRML) is used to define the simulated world and an advanced graphics engine is used to render realistic surfaces over the physics engine bounding boxes. Table 3.1 shows a comparison between the main features of the two softwares.

Table 3.1: SimTwo and Webots feature comparison

Feature	**SimTwo**	**Webots**
Physics Engine	ODE	ODE
Actuators	Simulated DC motors with gearbox and non-linearities effects	Maximum joint force
Sensors	Infrared distance sensor	Gyroscope, pressure sensor and accelerometer
Real-time	Yes, with the possibility of a fixed time acceleration	User-selectable
Graphics	Simple, only ODE shapes shown	Possibility of including 3D models of the real-robot
Scene definition	XML-type language	VRML-type language

Chapter 4

Simulation results and analysis

This chapter constitutes the core of the developped work, detailing the simulation model implementation and the obtained results and their analysis. In Section 4.1 the design and implementation of the simulated humanoid model is presented. Section 4.2 shows the results obtained when applying a trajectory-based method for gait generation. The results obtained with the CPG-based method are illustrated in Section 4.3. A comparison between the two simulation environments and the different walking generation methods is done in Sections 4.4 and 4.5, respectively.

4.1 Humanoid model

To obtain reliable results that can be implemented in a physical system in the future, the model is built based on Aldebaran's NAO [55] humanoid, shown in figure 4.1. This is also the current standard platform used in the Robocup competition, which allows the obtained results to be compared and validated against work done by other researchers.

Figure 4.1: Aldebaran's NAO (Source: [55])

The NAO SimTwo model used for this project is based on the RoboStadium robot included in the Webots simulator. Only minimal adjustments, in terms of parts and joint positions were done, thus allowing comparisons between the two simulators. The robot has been decomposed in 12

cuboids whose equivalent sizes and masses are detailed in Table 4.1. These parts are connected using the joint configuration depicted in Figure 4.2.

Table 4.1: NAO model body parts

Part ID	**Size**	**Position**	**Mass**
Trunk	$(0.1, 0.1, 0.18)$	$(0, 0, 0)$	1.217
Left thigh	$(0.08, 0.07, 0.14)$	$(-0.05, 0.0, -0.150)$	0.513
Left shank	$(0.08, 0.07, 0.12)$	$(-0.05, 0.0, -0.280)$	0.423
Left foot	$(0.085, 0.148, 0.04)$	$(-0.053, 0.02, -0.330)$	0.128
Left arm	$(0.05, 0.05, 0.08)$	$(-0.1, 0.0, 0.05)$	0.153
Left forearm	$(0.05, 0.05, 0.08)$	$(-0.1, 0.0, -0.01)$	0.077
Head	$(0.065, 0.065, 0.065)$	$(0.0, 0.0, 0.13)$	0.2

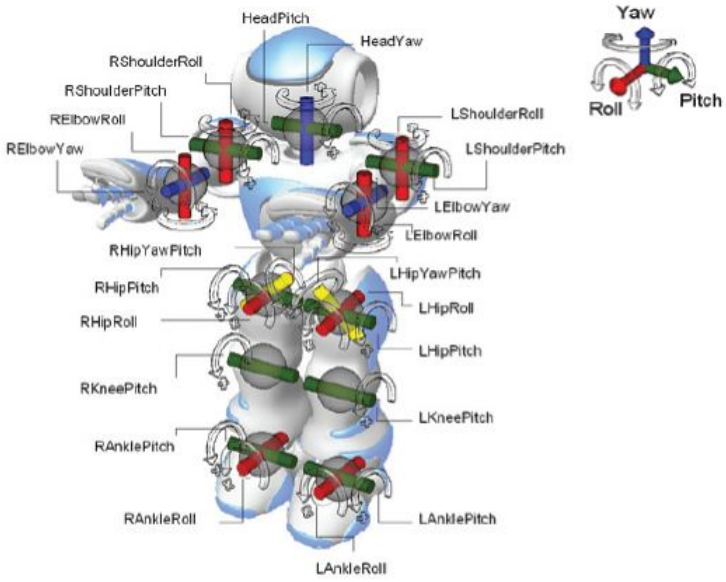


Figure 4.2: NAO's joint configuration (Source: [56])

The 45° degrees joint-configuration that provides the yaw movement of the leg can not be conveniently simulated in SimTwo, therefore it was removed. This is not an important issue since this joint is not used for either forward or backward motion. Also, as the objective of this project regards only walking, the arms were simplified to contain only the shoulder and elbow pitch joints. These simplifications contribute to an improvement in the simulation performance due to the decrease in degrees-of-freedom of the structure, without significantly affecting the properties under study. For SimTwo to accurately simulate the actuator and gearbox effects, the DC motor characteristics were implemented using the parameters as detailed in Table 4.2. The DC motors parameters correspond to the real motors used in the NAO robots as described by the manufacturer [57]. The final joint and axis configuration is shown in Table 4.3.

Figure 4.3 shows the NAO model both in SimTwo and Webots.

Table 4.2: NAO motor parameters

	M1	**M2**
Resistance (Ω)	$R_i = 6.44$	$R_i = 6.44$
Inductance (mH)	$L_i = 0.3e^{-3}$	$L_i = 0.3e^{-3}$
Torque constant (Nm/A)	$K_i = 21.3e^{-3}$	$K_i = 21.3e^{-3}$
Max. voltage (V)	$V_{max} = 18$	$V_{max} = 18$
Max. current (A)	$I_{max} = 0.598$	$I_{max} = 0.598$

Table 4.3: NAO model joints

Part ID	**Position**	**Joint type**	**Motor type**	**Gear ratio**
Left hip pitch	$(-0.05, 0, -0.115)$	Universal	M1	130.85
Left hip roll				201.3
Left knee pitch	$(-0.05, 0, -0.190)$	Hinge		130.85
Left ankle pitch	$(-0.05, 0, -0.290)$	Universal		130.85
Left ankle roll				201.3
Left shoulder pitch	$(-0.1, 0, 0.085)$	Hinge	M2	150.85
Left elbow pitch	$(-0.1, 0, 0.025)$	Hinge		150.85

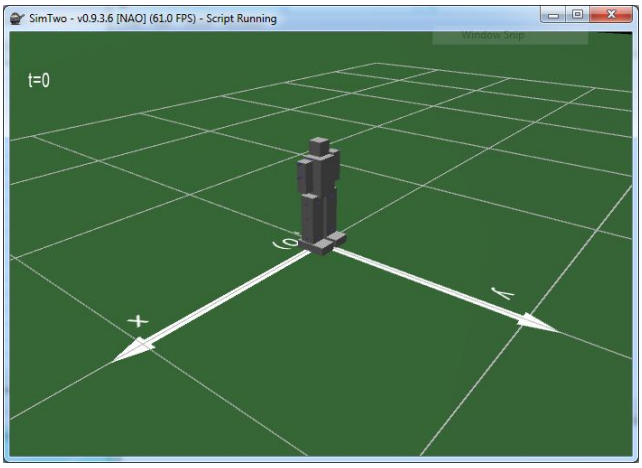
The high-level controller is implemented in Matlab for both simulators. For Webots the Matlab language is integrated in the simulation environment and a complete API is available. In SimTwo, Matlab comunicates with the simulator via UDP. Figures 4.4[1] and 4.5 illustrate the system architecture for SimTwo and Webots, respectively.

The SimTwo approach has the advantage of being much more flexible since it allows any program that provides a UDP connection to control the simulator either from local or remote locations. However, synchronization problems between the controller and the simulator can appear. The Webots architecture is less flexible but it provides a more direct integration of the controller with the simulation which can be beneficially, especially when running a big number of simulations, as is the case using genetic algorithms.

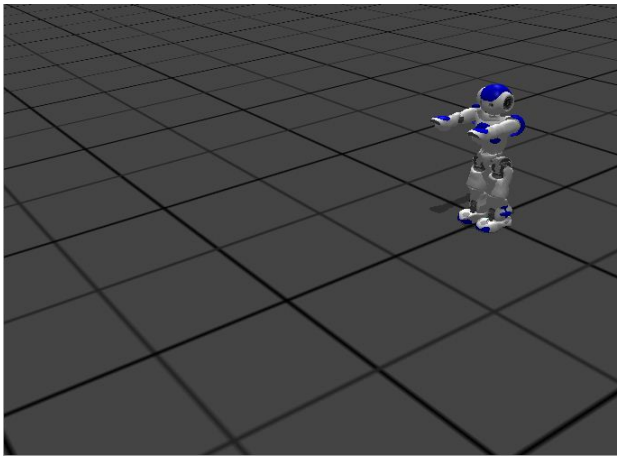
4.2 Trajectory-based gait

To validate the models implemented in SimTwo and Webots a simple trajectory-based gait generation is implemented. This method consists in creating a control signal by interpolating between ten predefined points over a full walking cycle. These points are determined for the left leg using a genetic algorithm optimization. It is known from existing data that, to perform a forward walk movement, the right and left leg signals are identical for roll joints and 90° out of

[1] All figures and tables are the author's own work



(a) SimTwo NAO model



(b) Webots NAO model

Figure 4.3: NAO simulated model

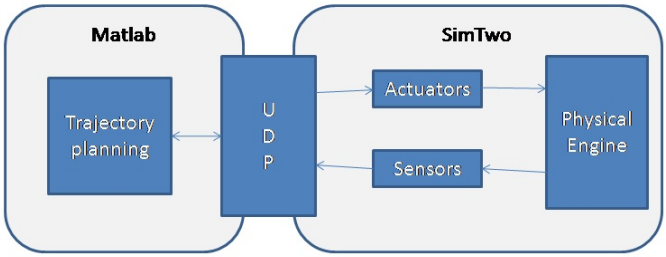


Figure 4.4: SimTwo control architecture

phase for pitch joints [45]. Therefore, this simplification is used to reduce the parameter search space and increase the convergence speed.

Since this method only describes the control signals when the robot is already in motion, a state-machine to do the movement initialization was implemented. During the initialization state all the robot joints are moved to the start position. The initialization time and position are also search parameters on the genetic algorithm. Figure 4.6 illustrates the implemented state-machine.

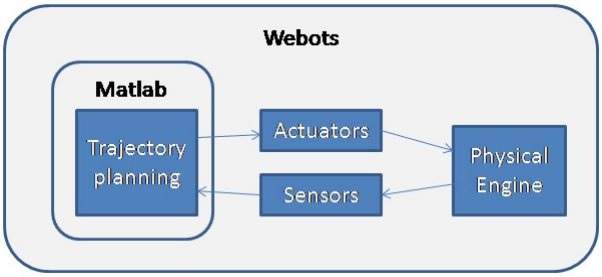


Figure 4.5: Webots control architecture

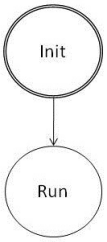


Figure 4.6: Gait generation state-machine

The fitness function used to evaluate the walking performance tries to maximize the robot standing time and forward walking distance, as well as minimize the lateral deviations from the straight walk. This function is represented in Equation 4.1. Note that all the genetic algorithms used in this work minimize the value of the fitness function.

$$Fitness = -4 \times t - d_f + \sum \|d_s\| \tag{4.1}$$

where t represents the total standing time, d_f represent the total forward distance and d_s the total lateral distance. The last variable was included to guarantee that the walk performed by the robot follows, as much as possible, a straight line.

A genetic algorithm with a population of 1500, roulette selection, crossover ratio of 0.7 and a mutation probability of 0.05 was executed. In both cases the algorithm converged to a walking motion in 45 generations. Figures 4.7 and 4.8 show the best walk performed by the humanoid in SimTwo and Webots[2], respectively. The best fitness value evolution is illustrated in Figure 4.9.

4.3 Central Pattern Generator based gait

4.3.1 Architecture and oscillator model selection

To generate coordinated signals that allow the humanoid robot to perform locomotion the CPG have to be interconnected to allow the phase and amplitude adjustment needed to generate convenient gaits. The architecture developed in this work is based on the typical chain configuration. The connections between hip, knee and ankle groups are uni-directional, whereas the connections

[2] All the videos can be accessed on http://paginas.fe.up.pt/~ee09010

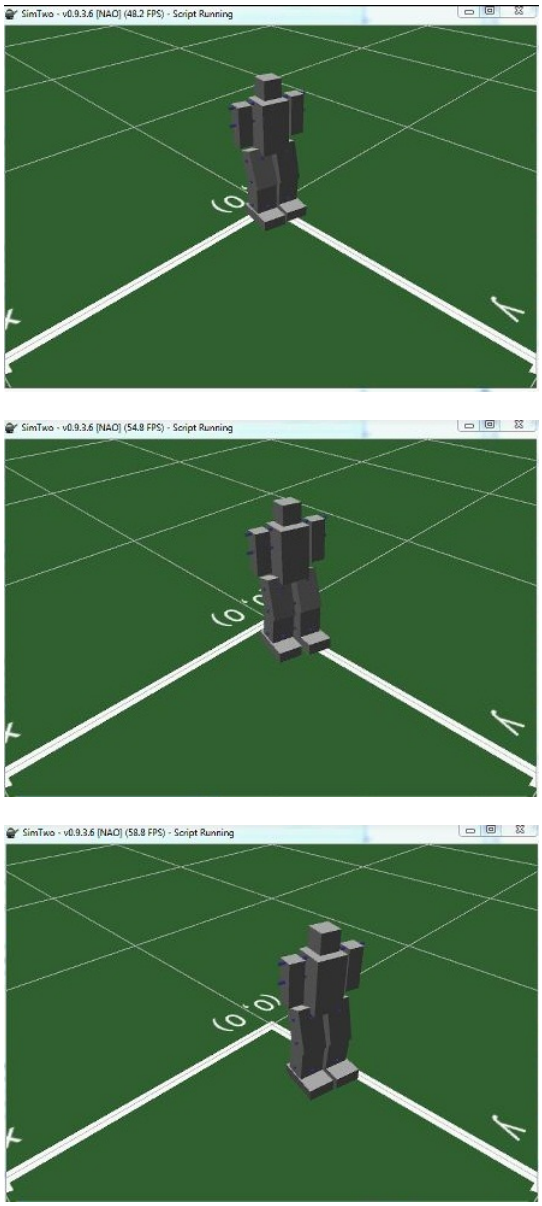


Figure 4.7: SimTwo trajectory-based gait

between joints of the same group, e.g. hip roll and hip pitch, are bi-directional. Figure 4.10 shows the disposition and connection of the different neurons.

Independently of the selected architecture an oscillator model has to be used to generate the target signals for the joint angles. Since, currently, there is no consistent work that compares the efficiency of each non-linear oscillator, the first step is to do a comparison between all the models proposed in section 2.1.

To perform this comparison, the hip control signals obtained for the trajectory-based gait are used as a reference. A genetic algorithm is used to minimize the difference between the CPG outputs and the reference signals. The difference is calculated as the square root of the sum of the

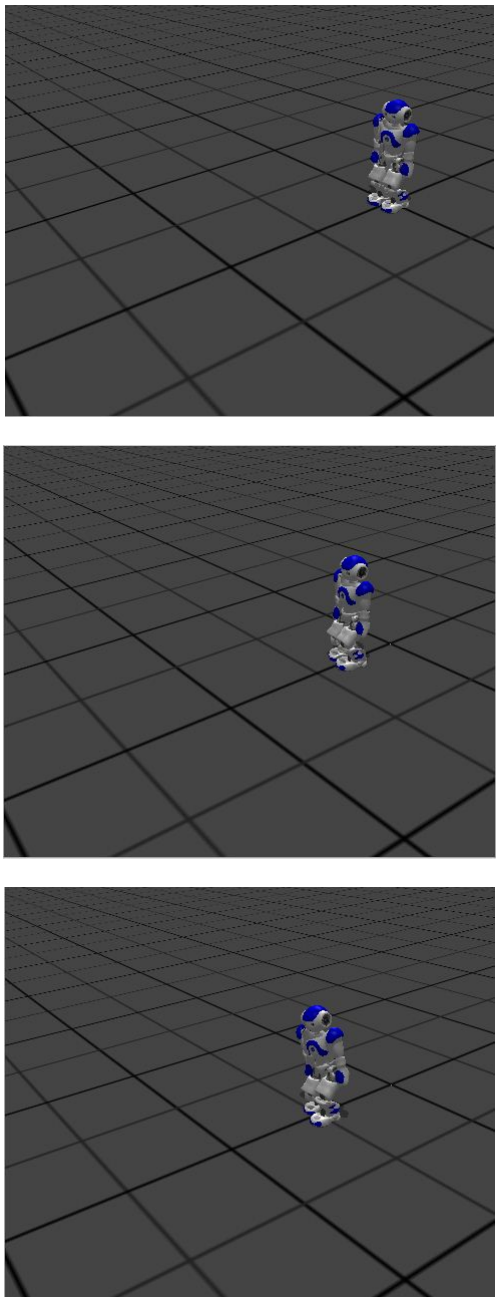


Figure 4.8: Webots trajectory-based gait

difference's square, as shown in Equation 4.2.

$$Fitness = \sqrt{\sum_{i=1}^{n} (X_i - \bar{X}_i)^2} \tag{4.2}$$

where n is the total number of points, X is the CPG output and $\bar{X}$ is the reference signal. All the reference signals were normalized, as to be centered around 0 and have an amplitude between -1 and 1. This should improve the optimization performance and guarantee unbiased results for

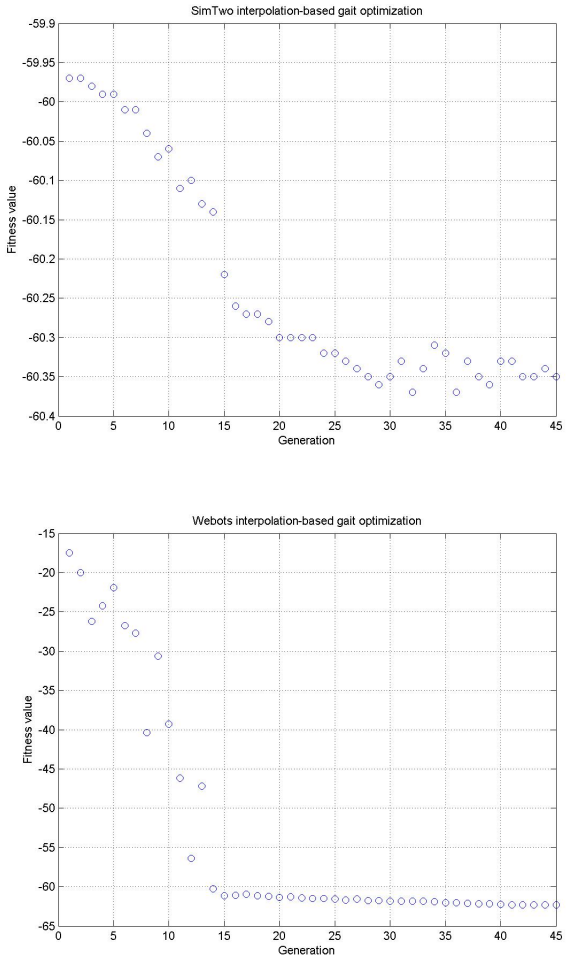


Figure 4.9: Trajectory-based gait fitness evolution

all the joints [45].

The results obtained from the approximation using Van der Pol, Rayleigh, Matsuoka and Duffing oscillators are presented in Table 4.4. The genetic algorithm was executed with populations of 15 times the number of search parameters, roulette selection, crossover ratio of 0.65 and a mutation probability of 0.05. All the optimization were done for 50 generations.

Table 4.4: Nonlinear oscillators performance comparison

Oscillator type	**Nr. parameters**	**Best approximation**
Ven der Pol	28	27.25
Rayleigh	28	26.32
Matsuoka	76	18.06
Duffing	44	19.42

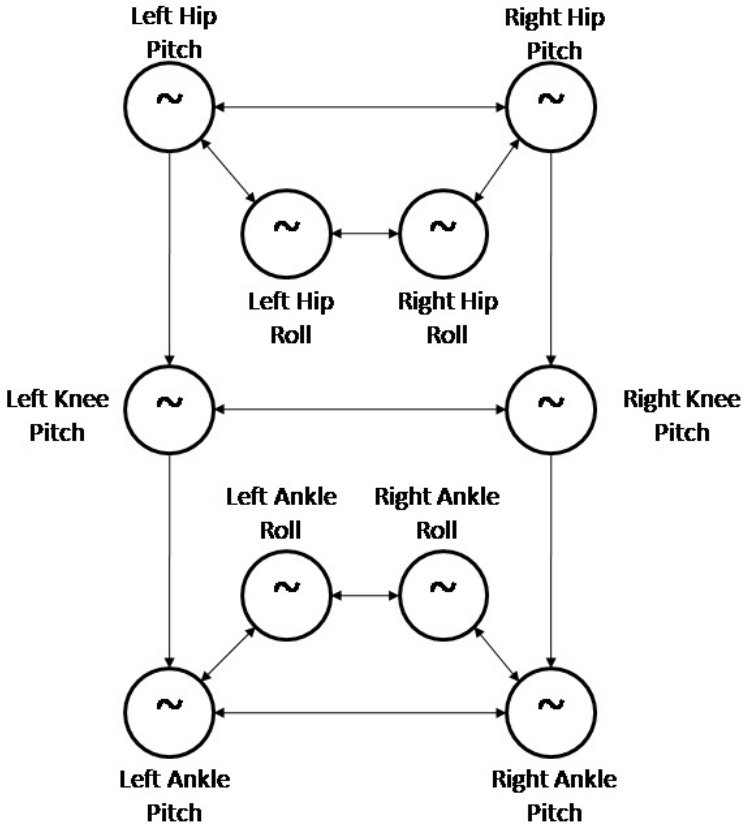


Figure 4.10: Central Pattern Generator architecture

From the obtained results it can be concluded that the Matsuoka oscillator is the one that can better approximate the reference signal.

4.3.2 Humanoid gait generation

Using the CPG architecture and model selected in the previous subsection, a genetic algorithm with the same fitness function and characteristics as the one shown in Equation 4.1 was used to determine the CPG parameters for forward walking motion. In this case there are a total of 650 parameters to be searched, therefore an initial population of 2500 individuals was used. Since each optimization can take several days to conclude, this method is only implemented in Webots. The resulting walk after 50 generations can be seen in Figure 4.11.

Analyzing the obtained results it can be observed that the algorithm converged to a type of motion that does not correspond to a walk, regardless of the fact that the final fitness value is equal to the best value obtained in the interpolation-based gait. This is a common problem inherent to genetic algorithms in which the results converge to optimal solutions that don't correspond to the expected output. In a first approach, penalty factors were included to prevent convergence to static gaits or constant outputs from the CPG. Nonetheless, no improvement in the obtained results was observed.

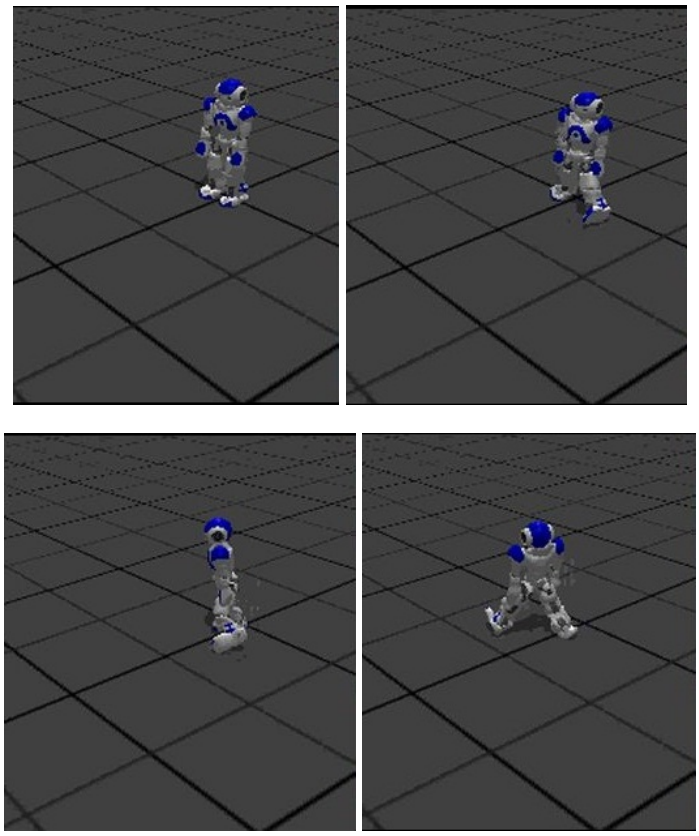


Figure 4.11: Webots CPG-based gait after total optimization

To address this problem a supervised learning mechanism, using the trajectory-based gait control signals as reference, was implemented. To increase the convergence speed of the genetic algorithm the approximations are done step-wise, starting by the hip and followed by the knee and ankle control signals. The approximations are done in a step-wise manner, starting by the hip joints, followed by the knee and ankle joints. This reduces the search-space of each optimization, thus improving the obtained results. Figures 4.12 to 4.14 show the final results of the approximation.

Using the entrained oscillator, a walking motion was generated as shown in Figure 4.15.

4.4 Simulation environments comparison

To compare the performances of each simulator, the developed trajectory-based gaits are used. The comparison is done in terms of maximum speed achieved by the robot, repeatability of the performed gait and total optimization time of the genetic algorithm. Table 4.5 presents the results of these metrics for both simulation environments. Figures 4.16 to 4.18 show the joint control signals for each simulator, when performing the best gait.

Comparing the obtained results, notorious differences between the resulting gaits in each simulation environment can be observed. Various reasons can explain these difference:

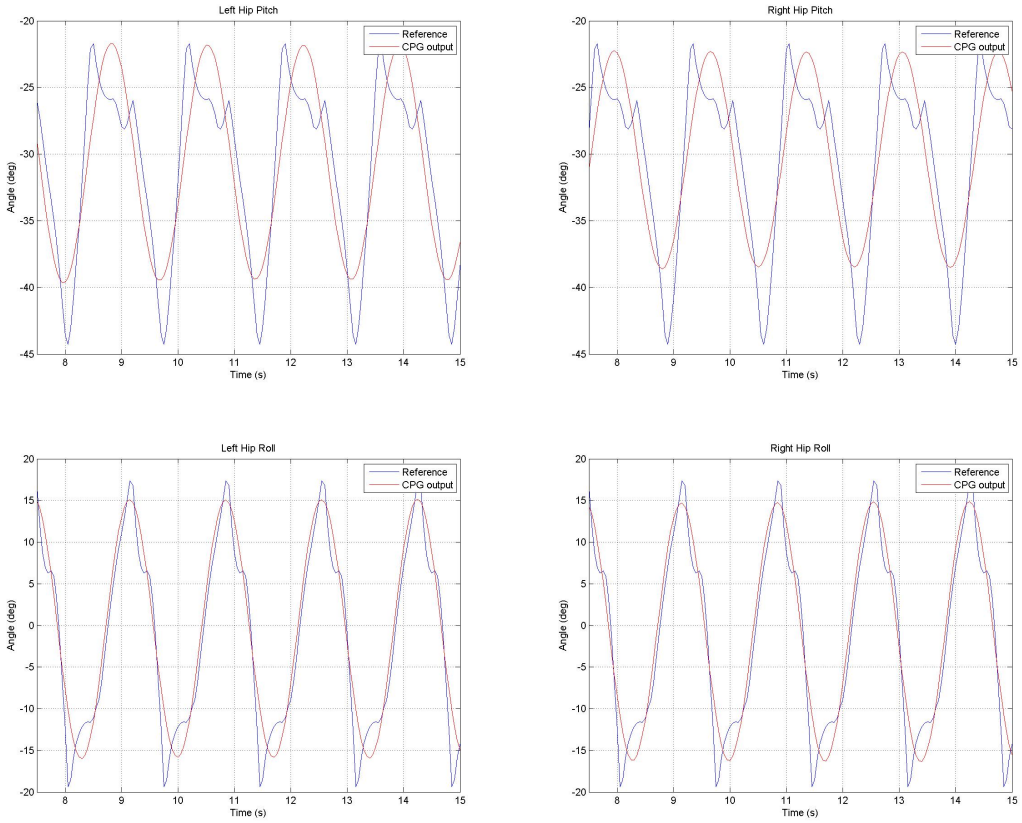


Figure 4.12: Hip control signals approximation

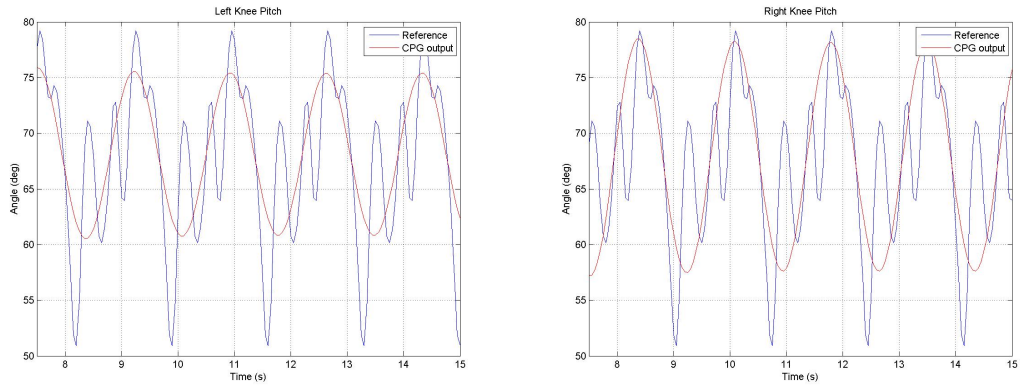


Figure 4.13: Knee control signals approximation

- Webots uses a simple P controller, whereas SimTwo implements a complete PID space-state controller. This difference can results in behaviour changes when following the reference signals;

- SimTwo implements a more realistic motor model that takes into account the non-linearities and gearbox effects,and Webots only considers maximum speed and force of each joint

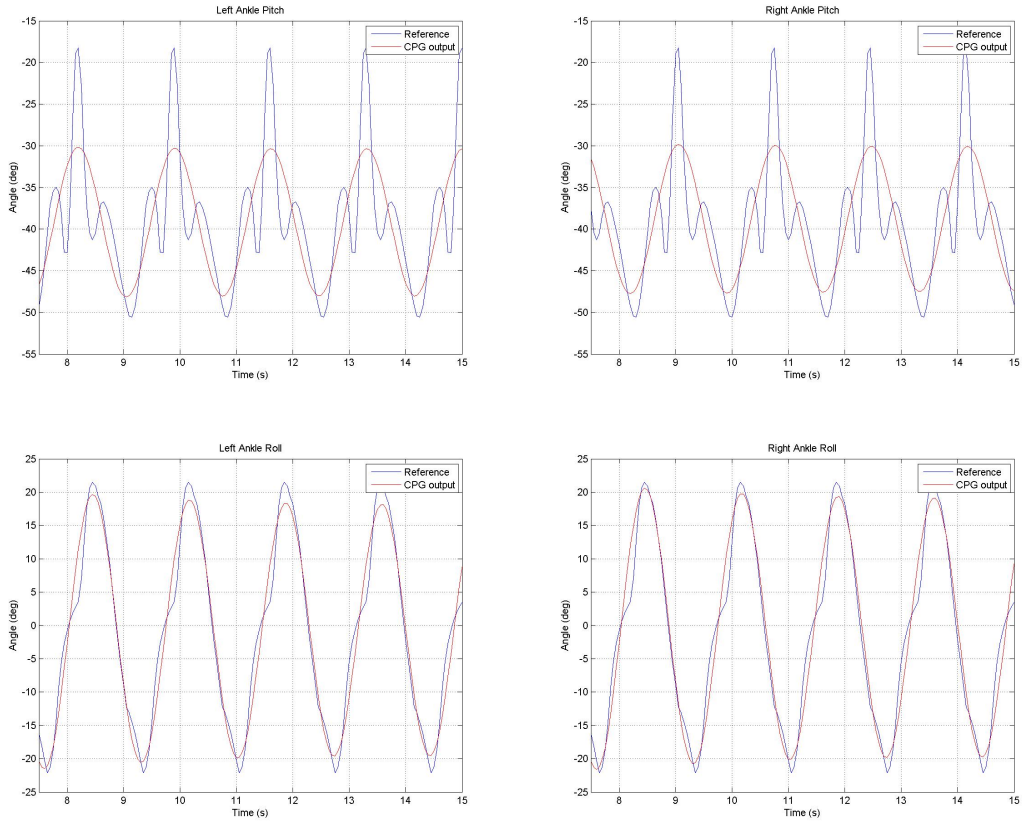


Figure 4.14: Ankle control signals approximation

Table 4.5: Simulator environement comparison

	SimTwo	**Webots**
Speed (m/s)	0.04	0.13
Repeatability	Low	High
Optimization time (hours)	36	12

- By analyzing the robot foot behavior, it is observed that the friction and collision detection mechanism behave differently in each simulator. It is not possible to do a detailed comparison since a detailed description or the source-code are not available. The low-repeatability can also be explained by these differences, since in the original ODE library the contact point forces are calculated in a random order and this can make errors propagate differently between simulation runs.

It is also interesting to note that the use of the ”as-fast-as-possible“ approach together with the Matlab API make the Webots optimization process approximatelly 3 times faster than the same operations with SimTwo.

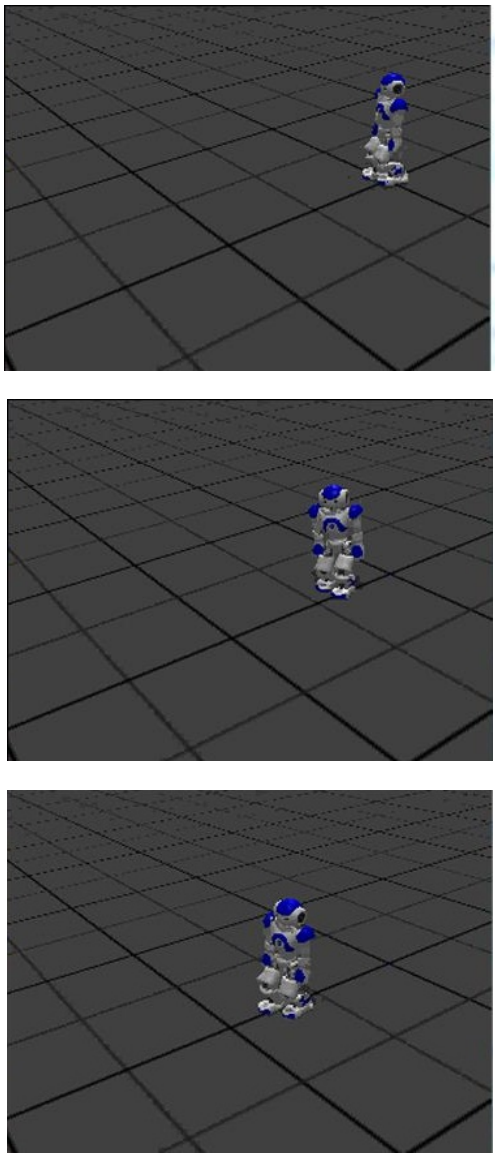


Figure 4.15: Webots CPG-based gait after approximation

Regardless of the existing differences, since no real hardware is available to make comparisons, it is not possible to determine which simulator has the most realistic behavior. Nevertheless, taking into account that Webots is a commercial and widely used simulator, it should be seen as a reliable baseline.

4.5 Gait generation methods comparison

Since the CPG-based gait was only implemented in Webots, the results obtained using this simulator are used to evaluate the performance of each of the generated motions. The metrics used for the comparison are the robot speed and the realism of the produced walk. The latter metric is mostly subjective but gives an idea about the motion quality. Also, Webots includes a demo

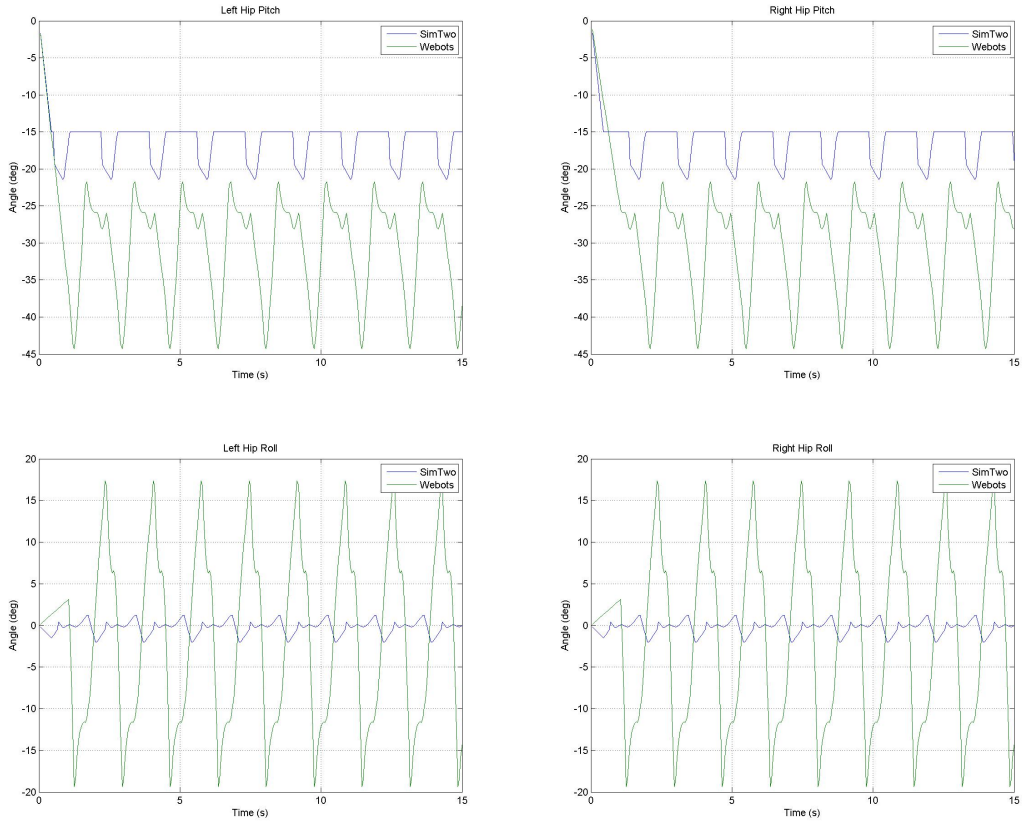


Figure 4.16: Hip control signals

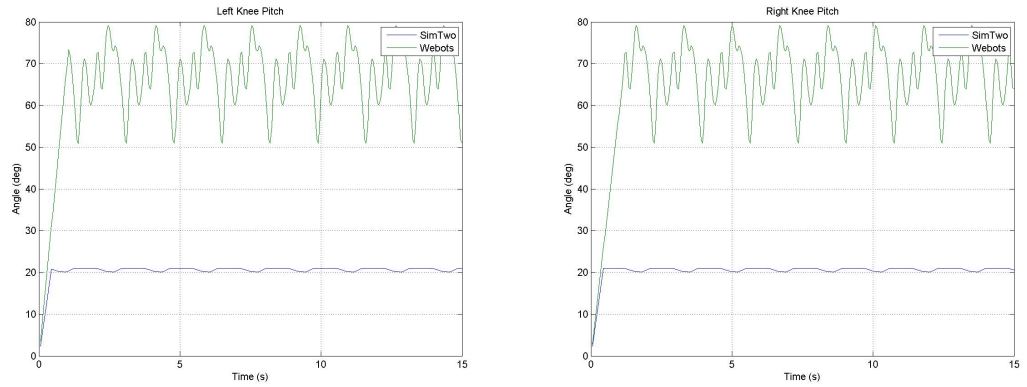


Figure 4.17: Knee control signals

with the standard NAO forward walking motion, which is used as the comparison reference point. Table 4.6 shows the results of the different gait types.

Since a supervised entrainment method was used to determine the CPG parameters that generate the gait, both the trajectory-based walk and CPG-based walk present very similar performances. They both allow the robot to move at approximately 0.14m/s, which is a very good

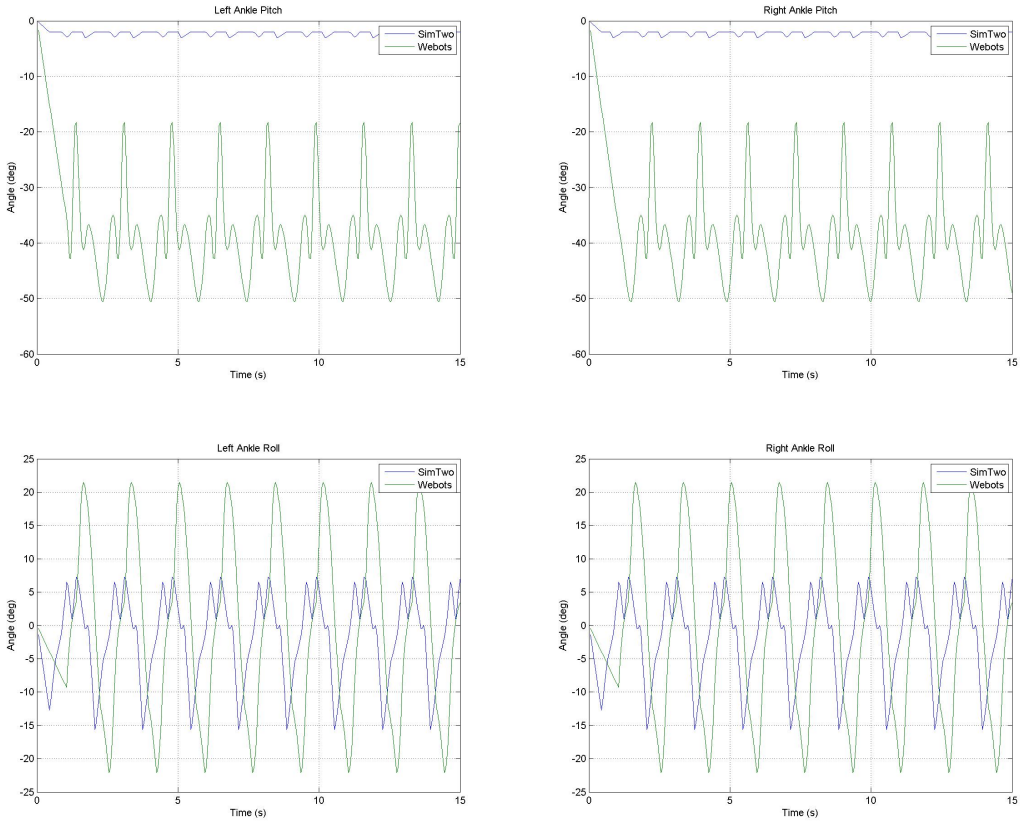


Figure 4.18: Ankle control signals

Table 4.6: Gait performance comparison

	Trajectory-based	**CPG-based**	**Standard NAO**
Speed (m/s)	0.14	0.14	0.16
Realistic	Yes	No	Yes

approximation to the standard 0.16m/s, of the standard walk. Nevertheless, the CPG-based walk shows a less realistic behavior when compared to the other implementation. This can be explained by the lower dynamics of the oscillator control signals, which can be observed in Figures 4.12 to 4.14.

Chapter 5

Conclusion

A NAO humanoid simulation model was successfully designed and implemented in the SimTwo and Webots simulation environments. Using a simple trajectory-based gait generation algorithm the humanoid walking motion was successfully performed.

Using the trajectory-based hip control signals as reference, a comparison between the Van der Pol, Rayleigh, Duffing and Matsuoka oscillator models was performed to identify the most convenient model to be used in the Central Pattern Generator. The comparison results show that the Matsuoka oscillator yields the best results.

Building a Central Pattern Generator using a chain architecture and the Matsuoka oscillator it was not possible to generate a walking motion using an unsupervised learning method based on a genetic algorithm. Even though the optimization converged to good fitness values, the resulting motion did not correspond to the expected gait. To overcome this problem a supervised learning mechanism, using the signals from the trajectory-based method as reference, was used to determine the correct Matsuoka oscillator parameters. With the approximated oscillator output it was possible to generate a forward walking motion without any further adaption.

From the results it can be concluded that improvements should be done in the SimTwo test environment. The ideal solution is to implement a model of a robot that is physically available and fine tune the dynamics simulation and DC motor behavior. Major improvements to the very limited friction and collision detection mechanisms of ODE should also be done. This is a very important first step, since a reliable and fully functional testing environment is of the uttermost importance to ensure that all the obtained results are reliable.

It can also be concluded that the direct use of the CPG output to control the joint position is a feasible solution for biped gait generation. Nevertheless, the implemented walk seems a bit unrealistic and would, most likely, not work in the real robot. In this case two approaches should be followed: improve the optimization method to be able to generate a walking motion without any reference trajectory; and to analyse the proposed motor-neuron mechanisms and perform a comparative analysis between this and the current technique. If the inclusion of the extra control layer provides the necessary dynamics to control the robot it could be a more suitable method for gait generation. Figure 5.1 shows a high-level overview of the overall locomotion system when

motor-neurons are used.

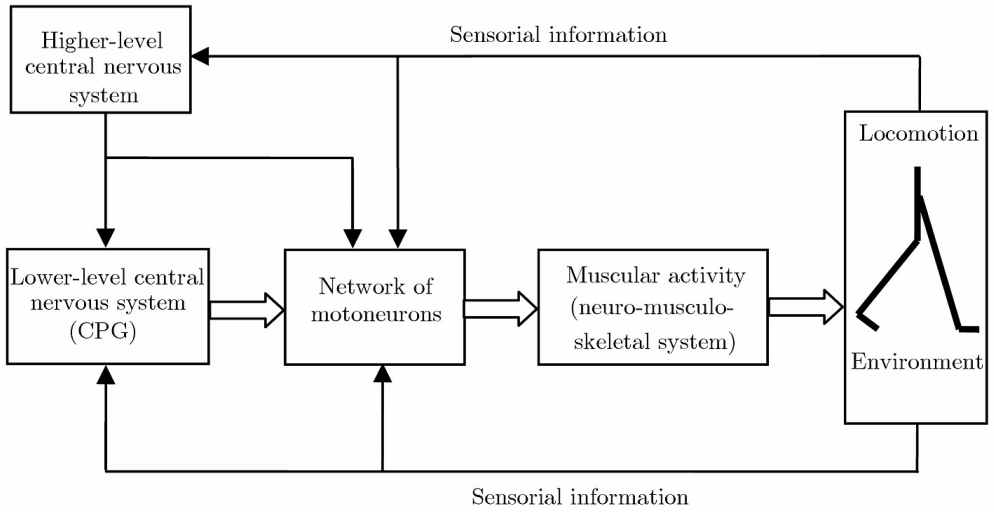


Figure 5.1: Humanoid locomotion overview (Source: [28])

When a stable open loop walk in a static environment is in place, the inclusion of feedback and dynamic gait transitions should be investigated. To evaluate the quality of the final generated gait, tests measuring the adaption to different slopes and ground types, the tolerance to external perturbations and the capability of gait transitions should be done. This is indispensable for the robot to operate conveniently in dynamic environments. Once all these methods have been studied and applied to simulation, work should be done in order to implement them in a real humanoid robot.

Appendix A

SimTwo humanoid model

This appendix contains the NAO humanoid model as described in the SimTwo XML model.

```
<?xml version="1.0" ?>
<robot>
   <kind value="NAO"/>

   <solids>
      <cuboid>
         <ID value="torso"/>
         <pos x="0" y="0" z="0"/>
         <size x="0.1" y="0.1" z="0.18"/>
         <mass value="1.2171"/>
         <desc Pt="Tronco"/>
         <desc Eng="Torso"/>
      </cuboid>

      <cuboid>
         <ID value="leftThigh"/>
         <pos x="-0.05" y="0.0" z="-0.150"/>
         <size x="0.08" y="0.07" z="0.14"/>
         <mass value="0.513"/>
         <desc Pt="CoxaEsq"/>
         <desc Eng="LeftThigh"/>
      </cuboid>

      <cuboid>
         <ID value="rightThigh"/>
         <pos x="0.05" y="0.0" z="-0.150"/>
```

```
    <size x="0.08" y="0.07" z="0.14"/>
    <mass value="0.513"/>
    <desc Pt="CoxaDir"/>
    <desc Eng="RightThigh"/>
</cuboid>

<cuboid>
    <ID value="leftShank"/>
    <pos x="-0.05" y="0.0" z="-0.255"/>
    <size x="0.08" y="0.07" z="0.12"/>
    <mass value="0.423"/>
    <desc Pt="CanelaEsq"/>
    <desc Eng="LeftShank"/>
</cuboid>

<cuboid>
    <ID value="rightShank"/>
    <pos x="0.05" y="0.0" z="-0.255"/>
    <size x="0.08" y="0.07" z="0.12"/>
    <mass value="0.423"/>
    <desc Pt="CanelaDir"/>
    <desc Eng="RightShank"/>
</cuboid>

<cuboid>
    <ID value="leftFoot"/>
    <pos x="-0.053" y="0.02" z="-0.330"/>
    <size x="0.085" y="0.148" z="0.04"/>
    <mass value="0.128"/>
    <desc Pt="PeEsq"/>
    <desc Eng="LeftFoot"/>
</cuboid>

<cuboid>
    <ID value="rightFoot"/>
    <pos x="0.053" y="0.02" z="-0.330"/>
    <size x="0.085" y="0.148" z="0.04"/>
    <mass value="0.128"/>
    <desc Pt="PeDir"/>
    <desc Eng="RightFoot"/>
```

```
</cuboid>

<cuboid>
   <ID value="leftArm"/>
   <pos x="-0.1" y="0.0" z="0.05"/>
   <size x="0.05" y="0.05" z="0.08"/>
   <mass value="0.153"/>
   <desc Pt="BracoEsq"/>
   <desc Eng="LeftArm"/>
</cuboid>

<cuboid>
   <ID value="rightArm"/>
   <pos x="0.1" y="0.0" z="0.05"/>
   <size x="0.05" y="0.05" z="0.08"/>
   <mass value="0.153"/>
   <desc Pt="BracoDir"/>
   <desc Eng="RightArm"/>
</cuboid>

<cuboid>
   <ID value="leftForearm"/>
   <pos x="-0.1" y="0.0" z="-0.01"/>
   <size x="0.05" y="0.05" z="0.08"/>
   <mass value="0.077"/>
   <desc Pt="AntebracoEsq"/>
   <desc Eng="LeftForearm"/>
</cuboid>

<cuboid>
   <ID value="rightForearm"/>
   <pos x="0.1" y="0.0" z="-0.01"/>
   <size x="0.05" y="0.05" z="0.08"/>
   <mass value="0.077"/>
   <desc Pt="AntebracoDir"/>
   <desc Eng="RightForearm"/>
</cuboid>

<cuboid>
   <ID value="head"/>
```

```
        <pos x="0.0" y="0.0" z="0.13"/>
        <size x="0.065" y="0.065" z="0.065"/>
        <mass value="0.1"/>
        <desc Pt="Cabeca"/>
        <desc Eng="Head"/>
    </cuboid>

</solids>

<articulations>
    <default>
        <draw radius="0.005" height="0.09" rgb24="8F0000"/>
        <!-- The default motors are the ones used
             for the lower body -->
        <motor ri="6.44" ki="21.3e-3" li="0.3e-3"
         vmax="18" imax="0.598" active="1"/>
        <friction bv="5e-3" fc="1e-3" coulomblimit="1e-3"/>
        <gear ratio="130.85" />
        <gear2 ratio="201.3" />
        <encoder ppr="1024" mean="0" stdev="0"/>
        <controller mode="state" kp="150" ki="0"
         kd="0" kf="0" active="1" period="5"/>
        <controller2 mode="state" kp="150" ki="0"
         kd="0" kf="0" active="1" period="5"/>
        <spring k="0" zeropos="0"/>
    </default>

    <joint>
        <ID value="left_hip"/>
        <pos x="-0.05" y="0.0" z="-0.115"/>
        <axis x="1" y="0" z="0"/>
        <axis2 x="0" y="-1" z="0"/>
        <gear ratio="130.85" />
        <gear2 ratio="201.3" />
        <connect B1="torso" B2="leftThigh"/>
        <limits Min="-100" Max="25"/>
        <limits2 Min="-25" Max="45"/>
        <type value="Universal"/>
        <desc Pt="CoxaEsq"/>
        <desc Eng="LeftHip"/>
```

```
</joint>

<joint>
   <ID value="right_hip"/>
   <pos x="0.05" y="0.0" z="-0.115"/>
   <axis x="1" y="0" z="0"/>
   <axis2 x="0" y="-1" z="0"/>
   <gear ratio="130.85" />
   <gear2 ratio="201.3" />
   <connect B1="torso" B2="rightThigh"/>
   <limits Min="100" Max="-25"/>
   <limits2 Min="-45" Max="25"/>
   <type value="Universal"/>
   <desc Pt="CoxaDir"/>
   <desc Eng="RightHip"/>
</joint>

<joint>
   <ID value="left_knee"/>
   <pos x="-0.05" y="0.0" z="-0.190"/>
   <axis x="1" y="0" z="0"/>
   <gear ratio="130.85" />
   <connect B1="leftThigh" B2="leftShank"/>
   <limits Min="0" Max="130"/>
   <type value="Hinge"/>
   <desc Pt="JoelhoEsq"/>
   <desc Eng="LeftKnee"/>
</joint>

<joint>
   <ID value="right_knee"/>
   <pos x="0.05" y="0.0" z="-0.190"/>
   <axis x="1" y="0" z="0"/>
   <gear ratio="130.85" />
   <connect B1="rightThigh" B2="rightShank"/>
   <limits Min="0" Max="130"/>
   <type value="Hinge"/>
   <desc Pt="JoelhoDir"/>
   <desc Eng="RightKnee"/>
</joint>
```

```
<joint>
   <ID value="left_ankle"/>
   <pos x="-0.05" y="0.0" z="-0.290"/>
   <axis x="1" y="0" z="0"/>
   <axis2 x="0" y="-1" z="0"/>
   <gear ratio="130.85" />
   <gear2 ratio="201.3" />
   <connect B1="leftShank" B2="leftFoot"/>
   <limits Min="-75" Max="45"/>
   <limits2 Min="-45" Max="25"/>
   <type value="Universal"/>
   <desc Pt="CoxaEsq"/>
   <desc Eng="LeftHip"/>
</joint>

<joint>
   <ID value="right_ankle"/>
   <pos x="0.05" y="0.0" z="-0.290"/>
   <axis x="1" y="0" z="0"/>
   <axis2 x="0" y="-1" z="0"/>
   <gear ratio="130.85" />
   <gear2 ratio="201.3" />
   <connect B1="rightShank" B2="rightFoot"/>
   <limits Min="-75" Max="45"/>
   <limits2 Min="-25" Max="45"/>
   <type value="Universal"/>
   <desc Pt="CoxaEsq"/>
   <desc Eng="LeftHip"/>
</joint>

<joint>
   <ID value="left_shoulder"/>
   <pos x="-0.1" y="0.0" z="0.085"/>
   <axis x="1" y="0" z="0"/>
   <connect B1="torso" B2="leftArm"/>
   <limits Min="-120" Max="120"/>
   <type value="Hinge"/>
   <desc Pt="OmbroEsq"/>
   <desc Eng="LeftShoulder"/>
```

```
    <motor ri="23.1" ki="16.6e-3"
     vmax="21" imax="0.248" active="1"/>
    <gear ratio="150.85"/>
    <friction bv="5e-3" fc="1e-2" coulomblimit="1e-2"/>
    <encoder ppr="4096" mean="0" stdev="0"/>
    <spring k="0" zeropos="0"/>
</joint>

<joint>
    <ID value="right_shoulder"/>
    <pos x="0.1" y="0.0" z="0.085"/>
    <axis x="1" y="0" z="0"/>
    <connect B1="torso" B2="rightArm"/>
    <limits Min="-120" Max="120"/>
    <type value="Hinge"/>
    <desc Pt="OmbroDir"/>
    <desc Eng="RightShoulder"/>
    <motor ri="23.1" ki="16.6e-3"
     vmax="21" imax="0.248" active="1"/>
    <gear ratio="150.85"/>
    <friction bv="5e-3" fc="1e-2" coulomblimit="1e-2"/>
    <encoder ppr="4096" mean="0" stdev="0"/>
    <spring k="0" zeropos="0"/>
</joint>

<joint>
    <ID value="left_elbow"/>
    <pos x="-0.1" y="0.0" z="0.025"/>
    <axis x="1" y="0" z="0"/>
    <connect B1="leftArm" B2="leftForearm"/>
    <limits Min="-120" Max="120"/>
    <type value="Hinge"/>
    <desc Pt="CotoveloEsq"/>
    <desc Eng="LeftElbow"/>
    <motor ri="23.1" ki="16.6e-3"
     vmax="21" imax="0.248" active="1"/>
    <gear ratio="150.85"/>
    <friction bv="5e-3" fc="1e-2" coulomblimit="1e-2"/>
    <encoder ppr="4096" mean="0" stdev="0"/>
    <spring k="0" zeropos="0"/>
```

```
        </joint>

        <joint>
           <ID value="right_elbow"/>
           <pos x="0.1" y="0.0" z="0.025"/>
           <axis x="1" y="0" z="0"/>
           <connect B1="rightArm" B2="rightForearm"/>
           <limits Min="-120" Max="120"/>
           <type value="Hinge"/>
           <desc Pt="CotoveloDir"/>
           <desc Eng="RightElbow"/>
           <motor ri="23.1" ki="16.6e-3"
            vmax="21" imax="0.248" active="1"/>
           <gear ratio="150.85"/>
           <friction bv="5e-3" fc="1e-2" coulomblimit="1e-2"/>
           <encoder ppr="4096" mean="0" stdev="0"/>
           <spring k="0" zeropos="0"/>
        </joint>

        <joint>
           <ID value="neck"/>
           <pos x="0.0" y="0.0" z="0.10"/>
           <axis x="0" y="0" z="1"/>
           <connect B1="torso" B2="head"/>
           <limits Min="-120" Max="120"/>
           <type value="Hinge"/>
           <desc Pt="Pescoco"/>
           <desc Eng="Neck"/>
           <motor ri="23.1" ki="16.6e-3"
            vmax="21" imax="0.248" active="1"/>
           <gear ratio="150.85"/>
           <friction bv="5e-3" fc="1e-2" coulomblimit="1e-2"/>
           <encoder ppr="4096" mean="0" stdev="0"/>
           <spring k="0" zeropos="0"/>
        </joint>

    </articulations>

</robot>
```

Appendix B

Nonlinear oscillators source code

This appendix contains the Matlab code used to implement the nonlinear oscillators.

B.1 Van der Pol oscillator

```
function dy = VdPOsc(t, y, params)

neuronOut = [y(1);
             y(3);
             y(5);
             y(7);];

neuronInter = k*neuronOut;

%Neuron 1
dy(1) = y(2);
dy(2) = a(1)*(p(1)^2-neuronInter(1)^2)*y(2)-(w(1)^2)*y(1);

%Neuron 3
dy(3) = y(4);
dy(4) = a(2)*(p(2)^2-neuronInter(2)^2)*y(4)-(w(2)^2)*y(3);

%Neuron 4
dy(5) = y(6);
dy(6) = a(3)*(p(3)^2-neuronInter(3)^2)*y(6)-(w(3)^2)*y(5);

%Neuron 5
dy(7) = y(8);
```

```
dy(8) = a(4)*(p(4)^2-neuronInter(4)^2)*y(8)-(w(4)^2)*y(7);
```

B.2 Rayleigh oscillator

```
function dy = RayleighOsc(t, y, params)

neuronOut = [y(2);
             y(4);
             y(6);
             y(8);];

neuronInter = k*neuronOut;

%Neuron 1
dy(1) = y(2);
dy(2) = a(1)*(p(1)^2-neuronInter(1)^2)*y(2)-(w(1)^2)*y(1);

%Neuron 3
dy(3) = y(4);
dy(4) = a(2)*(p(2)^2-neuronInter(2)^2)*y(4)-(w(2)^2)*y(3);

%Neuron 4
dy(5) = y(6);
dy(6) = a(3)*(p(3)^2-neuronInter(3)^2)*y(6)-(w(3)^2)*y(5);

%Neuron 5
dy(7) = y(8);
dy(8) = a(4)*(p(4)^2-neuronInter(4)^2)*y(8)-(w(4)^2)*y(7);
```

B.3 Matsuoka oscillator

```
function dy = MatsuokaOsc(t, y, params)

 %Neuron output matrix initialization
neuronOut = [max(0,y(1));
             max(0,y(3));
             max(0,y(5));
             max(0,y(7));
             max(0,y(9));
```

```
                max(0,y(11));
                max(0,y(13));
                max(0,y(15));];

%Neuron interconnection precalculation
neuronInter = a*neuronOut;

%Neuron 1
dy(1) = (-y(1)-neuronInter(1)-b(1)*y(2)+s(1))/tr(1);
dy(2) = (-y(2)+neuronOut(1))/ta(1);

dy(3) = (-y(3)-neuronInter(2)-b(2)*y(4)+s(2))/tr(2);
dy(4) = (-y(4)+neuronOut(2))/ta(2);

%Neuron 2
dy(5) = (-y(5)-neuronInter(3)-b(3)*y(6)+s(3))/tr(3);
dy(6) = (-y(6)+neuronOut(3))/ta(3);

dy(7) = (-y(7)-neuronInter(4)-b(4)*y(8)+s(4))/tr(4);
dy(8) = (-y(8)+neuronOut(4))/ta(4);

%Neuron 3
dy(9) = (-y(9)-neuronInter(5)-b(5)*y(10)+s(5))/tr(5);
dy(10) = (-y(10)+neuronOut(5))/ta(5);

dy(11) = (-y(11)-neuronInter(6)-b(6)*y(12)+s(6))/tr(6);
dy(12) = (-y(12)+neuronOut(6))/ta(6);

%Neuron 4
dy(13) = (-y(13)-neuronInter(7)-b(7)*y(14)+s(7))/tr(7);
dy(14) = (-y(14)+neuronOut(7))/ta(7);

dy(15) = (-y(15)-neuronInter(8)-b(8)*y(16)+s(8))/tr(8);
dy(16) = (-y(16)+neuronOut(8))/ta(8);
```

B.4 Duffing oscillator

```
function dy = DuffingOsc(t, y, params)
```

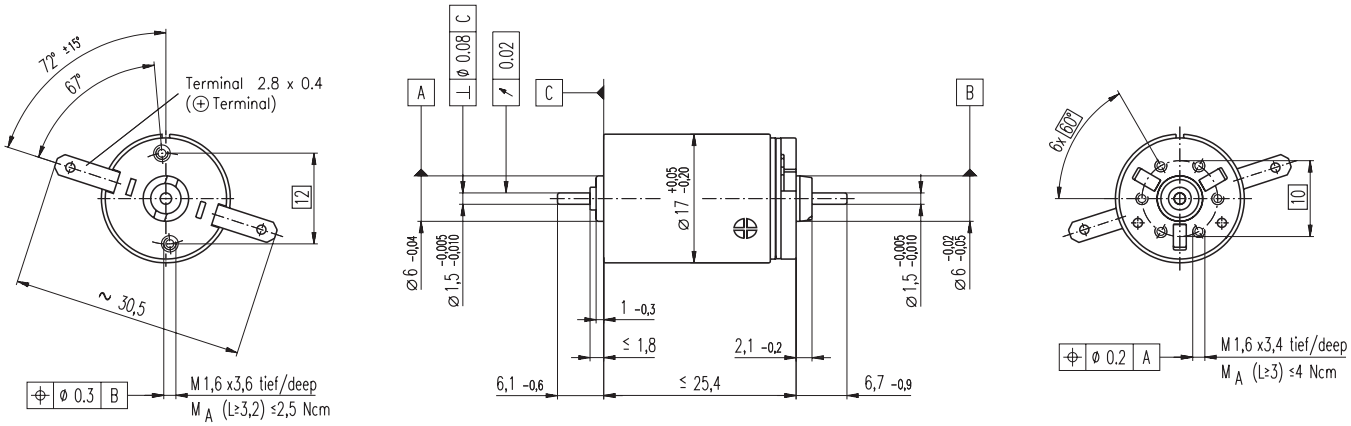
```
neuronOut = [y(1);
             y(3);
             y(5);
             y(7);];

neuronInter = k * neuronOut;

%Neuron 1
dy(1) = y(2);
dy(2) = -p(1)*y(2)-q(1)*y(1)-b(1)*y(1)^2- ...
    r(1)*y(1)^3+s(1)*cos(w(1)*t)-neuronInter(1)*y(3);

%Neuron 2
dy(3) = y(4);
dy(4) = -p(2)*y(4)-q(2)*y(3)-b(2)*y(3)^2- ...
    r(2)*y(3)^3+s(2)*cos(w(2)*t)-neuronInter(2)*y(1);

%Neuron 3
dy(5) = y(6);
dy(6) = -p(3)*y(6)-q(3)*y(5)-b(3)*y(5)^2- ...
    r(3)*y(5)^3+s(3)*cos(w(3)*t)-neuronInter(3)*y(3);

%Neuron 4
dy(7) = y(8);
dy(8) = -p(3)*y(8)-q(4)*y(7)-b(4)*y(7)^2- ...
    r(4)*y(7)^3+s(4)*cos(w(4)*t)-neuronInter(4)*y(1);
```

Appendix C

NAO DC motors datasheet

This appendix presents the datasheets of the motors used in the NAO humanoid robot.

RE-max 17 ∅17 mm, Graphite Brushes, 4.5 Watt



M 1:1

Stock program
Standard program
Special program (on request)

Order Number

			216008	216009	269571	216010	216011	216012	216013	216014	216015	216016	216017
Motor Data													
	Values at nominal voltage												
1	Nominal voltage	V	3.0	4.8	9.0	12.0	15.0	21.0	24.0	24.0	30.0	36.0	48.0
2	No load speed	rpm	11900	10800	11300	11200	11300	11900	11600	10400	10800	11500	10100
3	No load current	mA	116	65.4	36.4	27.1	22.0	16.6	14.2	12.4	10.4	9.31	6.0
4	Nominal speed	rpm	10900	8510	8190	8120	8240	8810	8570	7230	7660	8310	6790
5	Nominal torque (max. continuous torque)	mNm	1.43	2.72	3.90	3.91	3.88	3.84	3.87	3.88	3.84	3.76	3.71
6	Nominal current (max. continuous current)	A	0.720	0.720	0.555	0.414	0.333	0.248	0.214	0.191	0.157	0.137	0.089
7	Stall torque	mNm	18.2	13.1	14.6	14.5	14.5	15.1	14.9	13.1	13.5	13.9	11.7
8	Starting current	A	7.70	3.17	1.95	1.45	1.17	0.909	0.771	0.604	0.517	0.472	0.262
9	Max. efficiency	%	76	73	75	75	75	75	75	74	74	74	72
	Characteristics												
10	Terminal resistance	Ω	0.390	1.52	4.61	8.30	12.8	23.1	31.1	39.7	58.0	76.2	183
11	Terminal inductance	mH	0.0114	0.0349	0.114	0.206	0.314	0.558	0.759	0.956	1.38	1.75	4.04
12	Torque constant	mNm / A	2.37	4.14	7.49	10.1	12.4	16.6	19.3	21.7	26.0	29.4	44.5
13	Speed constant	rpm / V	4040	2310	1270	950	769	577	494	440	367	325	214
14	Speed / torque gradient	rpm / mNm	665	844	785	784	795	805	797	807	818	844	883
15	Mechanical time constant	ms	7.34	7.21	7.10	7.09	7.11	7.12	7.13	7.16	7.14	7.17	7.29
16	Rotor inertia	gcm²	1.05	0.816	0.864	0.864	0.854	0.844	0.854	0.848	0.834	0.811	0.788

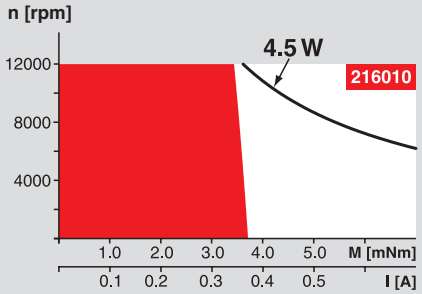
Specifications

	Thermal data	
17	Thermal resistance housing-ambient	35 K / W
18	Thermal resistance winding-housing	12 K / W
19	Thermal time constant winding	7.7 s
20	Thermal time constant motor	455 s
21	Ambient temperature	-30 ... +85°C
22	Max. permissible winding temperature	+125°C
	Mechanical data (sleeve bearings)	
23	Max. permissible speed	11900 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.012 mm
26	Max. axial load (dynamic)	0.8 N
27	Max. force for press fits (static)	35 N
	(static, shaft supported)	280 N
28	Max. radial loading, 5 mm from flange	1.4 N
	Mechanical data (ball bearings)	
23	Max. permissible speed	11900 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	2.2 N
27	Max. force for press fits (static)	30 N
	(static, shaft supported)	280 N
28	Max. radial loading, 5 mm from flange	7.8 N
	Other specifications	
29	Number of pole pairs	1
30	Number of commutator segments	7
31	Weight of motor	26 g

Values listed in the table are nominal.
Explanation of the figures on page 49.

Option
Ball bearings in place of sleeve bearings
Pigtails in place of terminals

Operating Range



Comments

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

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

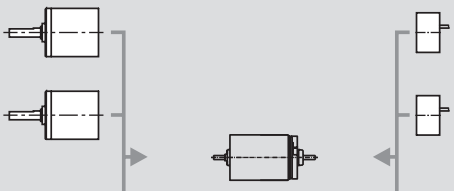
Assigned power rating

maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅16 mm
0.06 - 0.18 Nm
Page 223

Planetary Gearhead
∅16 mm
0.1 - 0.3 Nm
Page 224



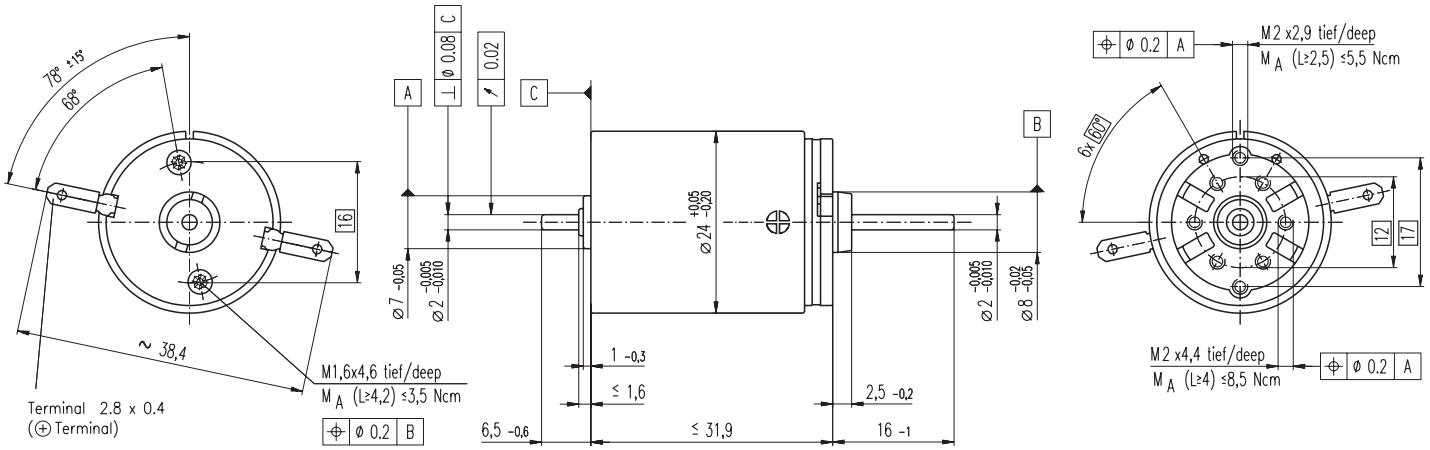
Encoder MR
32 CPT,
2 / 3 channels
Page 255

Encoder MR
128 / 256 / 512 CPT,
2 / 3 channels
Page 256

Recommended Electronics:

LSC 30/2	Page 276
EPOS 24/1	294
Notes	**18**

RE-max 24 ∅24 mm, Graphite Brushes, 11 Watt



M 1:1

Stock program
Standard program
Special program (on request)

Order Number

		222048	222049	222050	222051	222052	222053	222054	222055	222056	222057	222058	222059
Motor Data													
	Values at nominal voltage												
1	Nominal voltage	V	9.0	12.0	15.0	18.0	18.0	24.0	24.0	36.0	42.0	48.0	48.0	48.0
2	No load speed	rpm	8190	7540	8290	8870	8000	9310	7540	9120	8600	7410	5250	4730
3	No load current	mA	45.6	31.0	27.7	25.0	22.1	19.8	15.5	12.9	10.3	7.60	5.15	4.59
4	Nominal speed	rpm	6640	5870	6650	7240	6330	7670	5850	7450	6920	5690	3470	2960
5	Nominal torque (max. continuous torque)	mNm	10.8	12.4	12.4	12.3	12.3	12.1	12.2	12.1	12.0	12.0	12.0	12.1
6	Nominal current (max. continuous current)	A	1.08	0.853	0.752	0.666	0.598	0.516	0.421	0.335	0.270	0.204	0.144	0.131
7	Stall torque	mNm	57.7	56.9	63.6	67.7	59.5	69.3	55.3	66.6	62.0	52.4	35.7	32.8
8	Starting current	A	5.55	3.78	3.71	3.52	2.79	2.83	1.83	1.78	1.34	0.855	0.414	0.343
9	Max. efficiency	%	83	83	83	84	83	84	83	84	83	82	79	78
	Characteristics													
10	Terminal resistance	Ω	1.62	3.18	4.05	5.11	6.44	8.47	13.1	20.2	31.3	56.2	116	140
11	Terminal inductance	mH	0.0735	0.154	0.200	0.251	0.309	0.406	0.618	0.952	1.45	2.56	5.06	6.22
12	Torque constant	mNm / A	10.4	15.1	17.2	19.2	21.3	24.4	30.1	37.4	46.3	61.3	86.3	95.6
13	Speed constant	rpm / V	919	634	557	497	448	391	317	255	206	156	111	99.8
14	Speed / torque gradient	rpm / mNm	143	134	131	132	135	135	138	138	140	143	149	146
15	Mechanical time constant	ms	5.96	5.90	5.88	5.89	5.91	5.92	5.92	5.93	5.96	5.97	6.03	6.00
16	Rotor inertia	gcm²	3.97	4.22	4.28	4.26	4.17	4.17	4.11	4.11	4.07	4.00	3.88	3.92

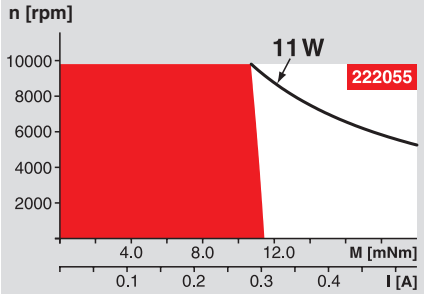
Specifications

	Thermal data	
17	Thermal resistance housing-ambient	24 K / W
18	Thermal resistance winding-housing	5.1 K / W
19	Thermal time constant winding	8.26 s
20	Thermal time constant motor	852 s
21	Ambient temperature	-30 ... +85°C
22	Max. permissible winding temperature	+125°C
	Mechanical data (sleeve bearings)	
23	Max. permissible speed	9800 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.012 mm
26	Max. axial load (dynamic)	1 N
27	Max. force for press fits (static)	80 N
	(static, shaft supported)	440 N
28	Max. radial loading, 5 mm from flange	2.8 N
	Mechanical data (ball bearings)	
23	Max. permissible speed	9800 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	3.3 N
27	Max. force for press fits (static)	45 N
	(static, shaft supported)	440 N
28	Max. radial loading, 5 mm from flange	12.3 N
	Other specifications	
29	Number of pole pairs	1
30	Number of commutator segments	9
31	Weight of motor	71 g

Values listed in the table are nominal.
Explanation of the figures on page 49.

Option
Ball bearings in place of sleeve bearings
Pigtails in place of terminals

Operating Range



Comments

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

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

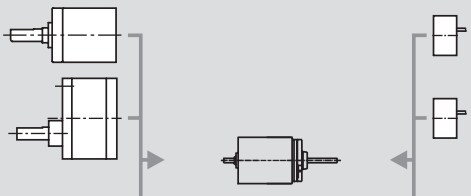
Assigned power rating

maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅22 mm
0.5 - 2.0 Nm
Page 231

Spur Gearhead
∅38 mm
0.1 - 0.6 Nm
Page 243



Encoder MR
32 CPT,
2 / 3 channels
Page 255

Encoder MR
128 / 256 / 512 CPT,
2 / 3 channels
Page 256

Recommended Electronics:

LSC 30/2	Page 276
ADS 50/5	276
ADS_E 50/5	277
EPOS 24/1	294
Notes	**18**

References

[1] M. Xie, Z. W. Zhong, L. Zhang, L. B. Xian, L. Wang, H. J. Yang, C. S. Song, and J. Li. A deterministic way of planning and controlling biped walking of loch humanoid robot. In *World Scientific*, 2008.

[2] Kemalettin Erbatur and Okan Kurt. Natural zmp trajectories for biped robot reference generation. *IEEE Transactions on Industrial Electronics*, 56(3):835–845, March 2009.

[3] Florian Hackenberg Bakk. Balancing central pattern generator based humanoid robot gait using reinforcement learning. Master's thesis, Graz University of Technology, 2007.

[4] Marc H. Raibert. *Legged robots that balance*. MIT Press, 1986.

[5] Sony. Qrio. Available online at http://www.sonyaibo.net/, Accessed on March 2010.

[6] Honda. Asimo. Available online at http://asimo.honda.com/, Accessed on March 2010.

[7] Fujitsu. Hoap-2. Available online at http://www.roboporium.com, Accessed on March 2010.

[8] Kai Feng, Chee-Meng Chew, Geok-Soon Hong, and Teresa Zielinska. Bipedal locomotion control using a four-compartmental central pattern generator. In *IEEE International Conference on Mechatronics & Automation*, pages 1515–1520, July 2005.

[9] Paulo Costa. Simtwo. Available online at http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo, Accessed on March 2010.

[10] Miomir Vukobratovic and Dusko Katic. Survey of intelligent control algorithms for humanoid robots. Robotics Laboratory, Mihailo Pupin Institute, 2002.

[11] Hugo Rafael de Brito Picado. Development of behaviors for a simulated humanoid robot. Master's thesis, University of Aveiro, 2008.

[12] Tad McGeer. Passive walking with knees. In *IEEE Robotics and Automation Conference*, 1990.

[13] Tad McGeer. Powered flight, child's play, silly wheels and walking machines. In *IEEE Robotics and Automation Conference*, 1989.

[14] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives. *Robotics and Autonomous Systems*, 47:79–91, 2004.

[15] M. Vukobratovic., B. Borovac, D. Surla, and D. Stokic. *Biped locomotion: Dynamics, Stability, Control and Aplication*. Springer-Verlag, 1990.

[16] Marilyn MacKay-Lyons. Central pattern generation of locomotion: A review of the evidence. *Physical Therapy*, 82:69–83, 2002.

[17] Jianjuen Hu. Learning control of bipedal dynamic walking robots with neural networks. Master's thesis, Massachusetts Institute of Technology, 1998.

[18] Guang Lei Liu, Maki K. Habib, Keigo Watanabe, and Kiyotaka Izumi. Cpg based control for generating stable bipedal trajectories under external perturbation. In *SICE Annual Conference*, pages 1019–1022, 2007.

[19] Torsten Reil and Phil Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6:159–168, 2002.

[20] Hironobu Inada and Kazuo Ishii. A bipedal walk using central pattern generator (cpg). Accessed online.

[21] Jiang Shan, Cheng Junshi, and Chen Jiapin. Design of central pattern generator for humanoid robot walking based on multi-objective ga. In *Proceedings of the 2000 International Conference on Intelligent Robots and Systems*, 2000.

[22] Woosung Yang, Nak Young Chong, Syungkwon Ra, ChangHwan Kim, and Bum Jae You. Self-stabilizing bipedal locomotion employing neural oscillators. In *8th IEEE-RAS International Conference on Humanoid Robots*, 2008.

[23] Milton Roberto Heinen and Fernando Santos Osório. Applying neural networks to control gait of simulated robots. In *10th Brazilian Symposium on Neural Networks*, 2008.

[24] Chengju Liu, Qijun Chen, and Jiaqi Zhang. Coupled van der pol oscillators utilised as central pattern generators for quadruped locomotion. In *Chinese Control and Decision Conference*, 2009.

[25] Paolo Arena, Luigi Fortuna, Mattia Frasca, and Luca Patané. A cnn-based chip for robot locomotion control. *IEEE Transactions On Circuits And Systems*, 52:1862–1871, 2005.

[26] Loic Matthey, Ludovic Righetti, and Auke Jan Ijspeert. Experimental study of limit cycles and chaotic controllers for the locomotion of centipede robots. In *IEEE/RSJ Conference on Intelligent Robots and Systems*, 2008.

[27] Dai bing Zhang, De wen Hu, Lin cheng Shen, and Hai bin Xie. A bionic neural network for fish-robot locomotion. *Journal of Bionic Engineering*, 3:187–194, 2006.

[28] Wu Qi Di, Liu Cheng Ju, Zhang Jia Qi, and Chen Qi Jun. Survey of locomotion control of legged robots inspired by biological concept. In *Science in China: Series F - Information Science*. Springer, 2009.

[29] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21/4:642–653, 2008.

[30] Almir Herali, Krister Wolff, and Mattias Wahde. *Central Pattern Generators for Gait Generation in Bipedal Robots*. I-Tech, 2007.

[31] Paschalis Veskos and Yiannis Demiris. Experimental comparison of the van der pol and rayleigh nonlinear oscillators for a robotic swinging task. In *AISB Conference on Adaptation in Artificial and Biological Systems*, pages 197–202, 2006.

[32] Scholarpedia. Van der pol oscillator. Available online at http://www.scholarpedia.org/wiki/images/b/b5/VdP-ts-e10.gif, Accessed on April 2010.

[33] Wolfram. Duffing oscillator demonstration. Available online at http://demonstrations.wolfram.com/DuffingOscillator/HTMLImages/index.en/popup_12.jpg, Accessed on April 2010.

[34] Kiyotoshi Matsuoka. Mechanisms of frequency and pattern control in the neural rhythm generators. In *Biological Cybernetics*. Springer-Verlag, 1987.

[35] Fumio Nagashima. A motion learning method using cpg/np. In *2nd International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[36] Jiang Shan and Fumio Nagashima. Neural locomotion controller design and implementation for humanoid robot hoap-1. In *Annual Conference of the Robotics Society of Japan*, 2000.

[37] Tee Wei In and Prahlad Vadakkepat. Hybrid controller for biped gait generation. In *2nd International Conference on Autonomous Robots and Agents*, pages 452–457, Palmerston North, New Zealand, December 2004.

[38] Armando Carlos de Pina Filho and Max Suell Dutra. Application of hybrid van der pol-rayleigh oscillators for modeling of a bipedal robot. *Mechanics of Solids in Brazil*, 1:209–221, 2009.

[39] Li Liu, Andrew B. Wright, and Gray T. Anderson. Trajectory planning and control for a human-like robot leg with coupled neural-oscillators. In *Proceedings of Mechatronics*, 2000.

[40] Jiang Shan and Fum. Biologically inspired spinal locomotion controller for humanoid robot. In *19th Annual Conference of the Robotics Society of Japan*, pages 517–518, 2001.

[41] Yurak Son, Yuka Kasai, Takashi Yasuno, Takuya Kamano, and Takayuki Suzuki. Generation of adaptive gait patterns for quadruped robot using cpg network. *Journal of Sinal Processing*, 8:455–460, 2004.

[42] Xu Guan, Haojun Zheng, and Xiuli Zhang. Biologically inspired quadruped robot biosbot: Modeling, simulation and experiment. In *2nd International Conference on Autonomous Robots and Agents*, pages 261–266, December 2004.

[43] Jeong-Jung Kim and Ju-Jang Lee. Gait adaptation method of biped robot for various terrains using central pattern generator (cpg) and learning mechanism. In *International Conference on Control, Automation and Systems 2007*, pages 10–14, 2007.

[44] Andrew L. Kun and W. Thomas Miller. Adaptive dynamic balance of a biped robot using neural networks. Accessed online.

[45] Hironobu Inada and Kazuo Ishii. Behavior generation of bipedal robot using central pattern generator(cpg) (1st report: Cpg parameters searching method by genetic algorithm). In *IEE/RSJ lntl Conterence on Intelligent Robots and Systems*, 2003.

[46] Ludovic Righetti, Jonas Buchli, and Auke Jan Ijspeert. Dynamic hebbian learning in adaptive frequency oscillators. In *Physica D: Nonlinear Phenomena*, pages 269–281. Elsevier, 2006.

[47] Simon Rutishauser, Alexander Sprowitz, Ludovic Righetti, and Auke Jan Ijspeert. Passive compliant quadruped robot using central pattern generators for locomotion control. In *2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 1930–1935, 2008.

[48] Thomas Weise. *Global Optimization Algorithms Theory and Application*. Available online at http://www.it-weise.de/, 2009.

[49] Wikipedia. Control theory. Available online at http://en.wikipedia.org/wiki/Control_theory, Accessed on April 2010.

[50] Wikipedia. Pid. Available online at http://en.wikipedia.org/wiki/PID_controller, Accessed on April 2010.

[51] R. Isermann. *Digital Control Systems*. Springer-Verlag, 1989.

[52] Embedded. Pid without a phd. Available online at http://www.embedded.com/2000/0010/0010feat3.htm, Accessed on April 2010.

[53] Milton Roberto Heinen and Fernando Santos Osório. Controle inteligente do caminhar de robôs móveis utilizando algoritmos genéticos e redes neurais artificiais. In *VI Encontro Nacional de Inteligência Artificial*, 2007.

[54] Russel Smith. Open dynamics engine. Available online at http://www.ode.org/, Accessed on March 2010.

[55] Aldebaran. Nao. Available online at http://www.aldebaran-robotics.com/en, Accessed on March 2010.

[56] Aldebaran. Nao robocup. Available online at http://www.aldebaran-robotics.com/en, Accessed on March 2010.

[57] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jérôme Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, and Bruno Maisonnier. Mechatronic design of nao humanoid. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 2124–2129, 2009.