Goal-Based Control and Planning in Biped Locomotion Using Computational Intelligence Methods

Juan José Figueredo Uribe

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
Bogotá D.C., Colombia
2012

Goal-Based Control and Planning in Biped Locomotion Using Computational Intelligence Methods

Juan José Figueredo Uribe

Thesis submitted as partial fulfillment of the requirements for the degree of:
Magister en Ingeniería de Sistemas y Computación

Tutor:
Jonatan Gómez Perdomo, PhD

Research Area:
Computational Intelligence
Research Group:
Grupo de Investigación en Vida Artificial - Alife

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
Bogotá D.C., Colombia
2012

Resumen

Este trabajo explora la aplicación de campos neuronales, a tareas de control dinámico en el domino de caminata bípeda.

En una primera aproximación, se propone una arquitectura de control que usa campos neuronales en 1D. Esta arquitectura de control es evaluada en el problema de estabilidad para el péndulo invertido de carro y barra, usado como modelo simplificado de caminata bípeda. El controlador por campos neuronales, parametrizado tanto manualmente como usando un algoritmo evolutivo (EA), se compara con una arquitectura de control basada en redes neuronales recurrentes (RNN), también parametrizada por por un EA. El controlador por campos neuronales parametrizado por EA se desempeña mejor que el parametrizado manualmente, y es capaz de recuperarse rápidamente de las condiciones iniciales más problemáticas.

Luego, se desarrolla una arquitectura extendida de control y planificación usando campos neurales en 2D, y se aplica al problema caminata bipeda simple (SBW). Para ello se usa un conjunto de valores óptimos para el parámetro de control, encontrado previamente usando algoritmos evolutivos. El controlador óptimo por campos neuronales obtenido se compara con el controlador lineal propuesto por Wisse et al., y a un controlador óptimo tabular que usa los mismos parámetros óptimos. Si bien los controladores propuestos para el problema SBW implementan una estrategia activa de control, se aproximan de manera más cercana a la caminata dinámica pasiva (PDW) que trabajos previos, disminuyendo la acción de control acumulada.

campos neuronales, neurocontrol, robótica evolutiva, vida artifical, control óptimo, caminata bípeda, péndulo invertido, caminata bípeda pasiva.

Abstract

This work explores the application of neural fields to dynamical control tasks in the domain of biped walking.

In a first approximation, a controller architecture that uses 1D neural fields is proposed. This controller architecture is evaluated using the stability problem for the cart-and-pole inverted pendulum, as a simplified biped walking model. The neural field controller is compared, parameterized both manually and using an evolutionary algorithm (EA), to a controller architecture based on a recurrent neural neuron (RNN), also parametrized by an EA. The non-evolved neural field controller performs better than the RNN controller. Also, the evolved neural field controller performs better than the non-evolved one and is able to recover fast from worst-case initial conditions.

Then, an extended control and planning architecture using 2D neural fields is developed and applied to the SBW problem. A set of optimal parameter values, previously found using an EA, is used as parameters for neural field controller. The optimal neural field controller is compared to the linear controller proposed by Wisse et al., and to a table-lookup controller using the same optimal parameters. While being an active control strategy, the controllers proposed here for the SBW problem approach more closely Passive Dynamic Walking (PDW) than previous works, by diminishing the cumulative control action.

neural fields, neurocontrol, evolutionary robotics, artificial life, optimal control, biped walking, inverted pendulum, passive dynamic walking

Contenido

1 Introduction

1.1 Motivation

Dynamical bipedal walking has been a key objective in robotics since its origins, due to the human curiosity about artificial anthropomorphic beings which gave rise to the robot concept itself [11]. As could be seen in most industrialized countries, the industrial manipulators have found a wide adoption, and there is little space to a major boost in the area [76]. Although, the interest is shifting from an industrial point of view to a more domestic one [3], where robots can be seen as additional aids to human daily tasks.

But, in order to accompany humans, the robot must be able to fluidly move through all the environments in which the human can, and those environments are devised to adapt well to anthropomorphic beings: factories, vehicles, houses, sidewalks, and shopping malls, among others. This way, a robot made to perform well in arbitrary environments will have a great advantage if it is anthropomorphic, so that it could serve well as an personal assistant [18].

The interest on biped robotics is not only for bio-robotics itself. Another reason to research anthropomorphic motion is the understanding of human morphology, mechanics and control, from a medical point of view, where robotics could serve as a testing scenario to both theories and technologies concerning human motion (for an example see [70]) and, probably, provide technological aids and substitutes to body parts when an impairment is present [29].

Another motivation to the research of biped walking is related to the fact that anthropomorphic motion planning and control is a complex problem that includes nonlinear and non-holonomic systems [6], complex computing tasks, adaptability to unknown and unstructured environments [14], among others, and is useful to test different mechanical, electronic, computing and control techniques applicable to diverse areas. As it is remarked by Craig [17], it should not be forgot that the predominant dynamics algorithm for open-chain mechanisms was developed [60] and refined [50] while working in biped walking problems.

Among the different problems faced in anthropomorphic motion, biped walking is one of the most difficult because its intrinsic instability. In contradistinction to wheeled mobile robotics and stable legged robotics, biped robotics must allow

locally unstable motion in order to attain a fluid locomotion. Additionally, there are several problems in biped locomotion other than stability. A bipedal walker must be capable to choose the best path to reach an objective, avoid obstacles, tolerate high perturbations, perform well in unstructured environments, and move with an optimal energy consumption.

1.2 Goals

While already a problem broadly studied, the problem of biped walking is still open for several key goals not currently entirely satisfied, such as optimal energy consumption, objective-based planning and walking in less structured environments. However, there are recent, and not so recent, notable contributions in this field obtained by diverse methods: active control, passive dynamic walking, and computational intelligence; which are giving a stronger basis for major advances in the field, fueled by a strong academic and commercial motivation.

Nonetheless, here we study a narrower problem, in which biped robotics combines with dynamical neural neurons and evolutionary robotics to pursue biped walking control by emergence, i.e. without the explicit specification of those parameters required in order to achieve the desired behavior with the control scheme chosen.

In a intuitive way, the problem studied can be broadly described as to provide a biped robot with an optimal locomotion. Nonetheless, since the optimality referred can be thought as a performance measure, it is convenient to be more specific in its definition. Based on the preliminaries in previous sections, some possible relevant goals to be accomplished by biped locomotion are:

- To conserve static stability: The ability to conserve equilibrium while in static state or in quasi-static motion.
- To walk conserving dynamic stability: The ability to walk, or locomote without leaving the contact with the floor, at velocities high enough to make considerable the inertial forces maintaining stability. That means the robot should walk indefinitely without falling if there are no obstacles in its way.
- To locally minimize energy consumption: For a given path minimize the energy consumption of the gait pattern.
- To react to external perturbations: Sense and compensate external perturbations that affect locomotion, including external forces and changes in environment parameters.

- To plan gait trajectories to attain specific objectives: The ability to choose a suitable trajectory to reach a desired state, not only in terms of path, but also joint coordination.
- To move across unstructured environments: Perform biped locomotion in environments with variable conditions which are not previously given to the robot, and plan according to the perception of those variable conditions.
- To globally minimize energy consumption: For a given objective in an arbitrary environment, select and perform the gait trajectory that minimizes energy consumption.

Although non strictly, the order of goals presented has in mind a growing complexity required to satisfy them. That implies that in some cases it would be useful to satisfy, at least partially, a previous one before the one following it.

The last goal proposed, the global minimization of energy consumption, has not been completely satisfied by any of the works reviewed so far, and the goals of motion across unstructured environments and planning of trajectories to attain specific objectives have been taken into account by few of them. Particularly, in the area of computational intelligence it is necessary first to address consistently the problem of planning previous to studying the remaining goals.

Therefore, the specific problem defined is to perform the motion planning and control of biped locomotion in a computer-simulated environment using a goal-oriented integrated architecture based on computational intelligence. The biped model defined is a rigid body linked model, with the lesser number of joints required to move in a two-dimensional environment.

The motion planning is understood as the process to detail a motion task into a sequence of reference actions, while the motion control is the process that transforms a sequence of reference actions into control signals applied to robot actuators (applying torque accordingly to each joint).The computer-simulated environment is essentially a physics engine in which the dynamical experiments can be performed, given a model of the system, environmental conditions and a set of inputs.

1.3 Main Contributions

The goal-oriented control architecture using neural fields proposed here, is the main contribution of this work. It shows that neural fields may be used as a useful component to build motion controllers, taking into account that they are also biologically plausible.

In order to be able to move consistently across an unstructured environment and to be useful for any specific task it is required that the biped can pursue specific goals. Being able to perform both planning and control is important for two reasons. The first one is to allow a uniform and hierarchical implementation across the architecture, allowing the same adaptation strategies and analysis tools to be applied to both planning and control systems, and performing the planning task dynamically in the same time scale in which the control system is working. The second one is to be able to change softly from one operation mode to another when goals have changed, controlling undesired high frequency components of the reference control signal.

An additional but important contribution, related to the goal of minimization of global energy consumption, is the optimization by means of evolutionary algorithms and neural fields of the previous work on pseudo-Passive Dynamic Walking controllers (see [67]) for the Simplest Biped Walking model (as proposed in [24]). With this contribution, we are advancing towards what could be the most valuable goal on biped walking control.

The proposed problem faces dynamic biped walking in structured environments without perturbations. With the contribution on goal-oriented planning and its integration to motion control using a computational intelligence approach, and furthermore with the improvement on energy consumption, it is expected that this work provides a framework such that the motion across unstructured environments and the global energy minimization for general environments will fall in the field of study in short time.

1.4 Contents Overview

The contents of this work are organized as follows:

- First some preliminaries are presented, including:
  - The problem of biped walking and some general approaches to solve it;
  - An introduction to neural fields.
  - An introduction to evolutionary algorithms.
  - And finally, a survey of computational intelligence applied to biped robotics.
- Then, a first approach to a neural field control architecture is shown. It is used to solve the stability problem for the inverted pendulum, and compared against a recurrent neural network controller. Also, it is shown a second

neural field controller, where adaptation using evolutionary algorithms is applied.

- Later, an extended neural field control and planning architecture is developed and applied to the stability problem for the Simplest Biped Walking model. In this chapter we show how the control architecture may be used at a planning level by changing the control policy to be applied, and also how it is able to minimize the global energy consumption. The neural field control architecture is compared to the linear controller proposed by Wisse et al., and to the optimized but not biologically inspired Sliding-mode controller proposed by us.

- Finally, some conclusions are drawn and possibilities for future work are presented.

2 Preliminaries

2.1 Introduction

Dynamical bipedal walking has been a key objective in robotics since it was first conceived, due to the initial idea of robots as anthropomorphic machines with human-mimicking behavior [11]. In spite of the advent of industrial robotics and a modern academic and industrial conception more oriented towards robotics as a production augmenting value [76], there are not only important reasons to pursue anthropomorphic robots and understanding of bipedal walking, but an increasing interest on non-industrial robots (for which the term service robot is sometimes used) [3].

Most of the environments existing are devised to adapt well to humans: factories, vehicles, houses, sidewalks, and shopping malls, among others. This way, a robot made to perform well in arbitrary environments will have a great advantage if it is anthropomorphic, and therefore it could serve well as an personal assistant [18]. Another reason to research anthropomorphic motion is the understanding of human morphology, mechanics and control, from a medical point of view, where robotics could serve as a testing scenario to both theories and technologies concerning human motion (for an example see [70]) and, probably, provide technological aids and substitutes to body parts when an impairment is present [29]. A third motivation is related to the fact that anthropomorphic motion planning and control is a complex problem that includes nonlinear and non-holonomic systems [6], complex computing tasks, adaptability to unknown and unstructured environments [14], among others, and is useful to test different mechanical, electronic, computing and control techniques applicable to diverse areas.

Among the different problems faced in anthropomorphic motion, biped walking is one of the most difficult because its intrinsic instability. In contradistinction to wheeled mobile robotics and stable legged robotics, biped robotics must allow locally unstable motion in order to attain a fluid locomotion. Additionally, there are several problems in biped locomotion other than stability. A bipedal walker must be in capacity of choose the best path to reach an objective, avoid obstacles, tolerate high perturbations, perform well in unstructured environments, and move with an optimal energy consumption.

2.2 Biped Walking

Biped walking is, from a robotics standpoint, the articulated motion of several bodies with motion between them coordinated by actuation torques in their joints. It is easier to model biped walking as a rigid bodies system, but the nature of biological tissues is best modeled by viscoelastic bodies. Also, it is easier to model biped walking with the most reduced number of degrees of freedom (i.e. the minimum number of independent parameters required to completely describe the sate of the system), but at the organ level there are at least 6-DOF in each leg (in the hip: Flexion-Extension, Abduction-Adduction, External-Internal Rotation; in the knee: Flexion-Extension; and in the ankle: Plantarflexion-Dorsiflexion, Pronation-Supination) for a total of 12-DOF in the lower extremities. Though, a biologically complete description of joint states would need a lot more degrees of freedom: there are about 650 skeletal muscles in human body, each one composed of several muscle fibers (myocytes) independently innervated. This way proper selection of model body and joint properties is crucial in modeling walking.

2.2.1 Static and Dynamic Stability

One important element in biped walking is stability. Generally speaking, there are two ways of defining stability in biped locomotion. The first method, assumes locomotion as a quasi-static motion and, supposing negligible inertial momentum in body links, the stability is given by the location of body center of mass (CoM). If it is inside of the convex hull containing all points in the support area (i.e. the supporting polygon) it is said to be stable, otherwise it is said to be unstable. An example of a statically stable biped robot is shown in Fig. **2-1**. This principle applies generally under low accelerations in the single-support phase, because the support polygon in this phase is limited to the contact area of the supporting feet, and therefore the robot has a very limited movement range in order to conserve stability. In the double-support feet it can be used reasonably to model stability under higher accelerations, because the support area is wider and the stability is less sensible to dynamic forces.

The second method, takes in account the inertial forces due to dynamical motion, and its general definition is still and open problem [4]. However, a limited way to determine biped dynamic equilibrium is given by the projection of the dynamical equivalent to CoM: the Zero Moment Point (ZMP). The Zero Moment Point is the point where the reaction force with the supporting element would produce a zero moment. If the ZMP is located inside the supporting polygon, destabilizing moments due to dynamic locomotion would be compensated by the

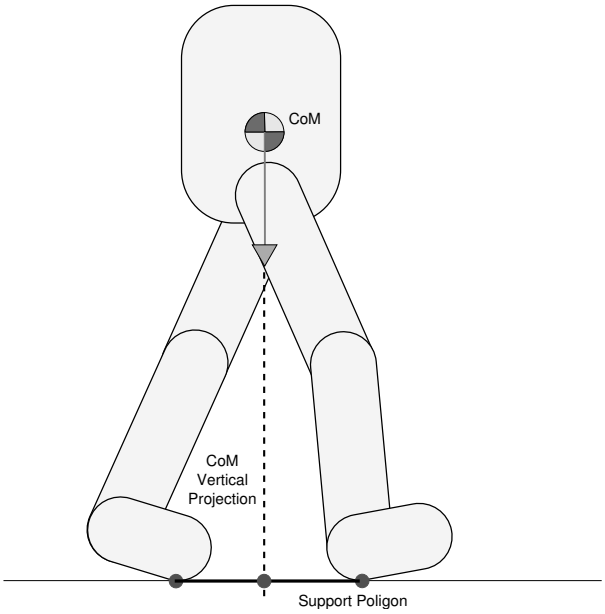


Figura 2-1: Static Equilibrium by CoM vertical projection over support polygon.

support reaction. An illustration of a robot which is not in equilibrium is shown in Fig. **2-2**. The ZMP is the projection over the floor of the total force acting over the CoM. When accelerations become relevant, the reference frame placed in the CoM is not anymore an inertial frame and, therefore, appears an inertial force opposed to CoM acceleration, which, with the weight force, yield a total force not normal to the floor. In order to be in moment equilibrium, the projection over the floor of the total force must coincide with the floor reaction point, that is, the point where the equivalent reaction force is applied.

Biped locomotion can be thought as a rhythmic and symmetric progression of trunk and limbs movement. In a complete gait cycle each leg passes trough a stance phase, in which its toe is in floor contact, and a swing phase, in which the foot is balancing without floor contact. Between them is a double support phase, where both feet are conforming the support area. Double support phase dynamics is more complex than that of the single support because of the additional constraints that yields a parallel kinematic chain. Thereby, a usual modeling technique is to model the dynamics of single support and a punctual transition condition for the double support [24]. However, double support dynamics modeling could be representative since it accounts for 20% of the gait cycle.

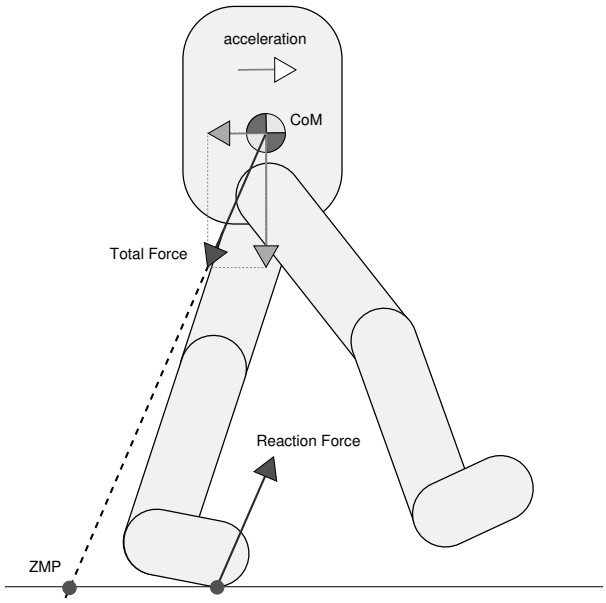


Figura 2-2: Dynamic Equilibrium not attained because of ZMP and floor reaction point not coincident. The robot is falling backwards.

2.2.2 Biological Considerations on Biped Locomotion

In order to build suitable controllers for biped locomotion it is necessary to define what aspects, or features, are most important to good performance, robustness, stability, adaptability, and optimality of walking. All of these properties are attained by biological biped walking and therefore it is the main inspiration for building models as well as controllers of locomotion.

There are several models of biped walking, and each of them remarks one or more important features. A very good review and analysis of biped locomotion models can be found in [63]. Vaughan reviews six models of bipedal walking: Bipedal walking as an evolutionary adaptation of hominids, minimization of energy consumption by displacing the CoM along an optimal path, progressive learning with risk of falling minimization, spinal cord inter-neurons acting as rhythmic central pattern generators, neural system training along with bio-mechanical system and environment adaptation, and feedback control in powered dynamic locomotion. The neurobiological aspects of motion control have been examined in [20]. Duysens remarks the reflex response of sensor inputs in motion control and presents tree levels for neural motion control: Feedback control in motoneurons, in terms of contribution of reflex action over motoneuron signal intensity; feedback control in central pattern generator flexor-extensor centers, relevant to movement synchronization and reflex response to perturbation and loads; and higher level

control, referred to conscious control of locomotion. Duysens states that despite of the relative low understanding of the high level control participation in locomotion, the two lower levels are complex and rich enough to be studied an applied to robot design (i.e. the so-called *spinal robot*, because it would have the control architecture expected in a human with transected spinal cord).

2.3 Neural Fields

Neural fields arise as a tissue level model of neural populations in brain. It has been proposed by Wilson and Cowan [65] and detailed by Amari [1] in the particular case of lateral inhibition. In this model, neural population in considered continuum in which exists a dynamical evolution equation where the mean activation potential evaluated in one place is affected by its neighborhood according to a so-called Mexican hat function (as noted by Coombes [16] better called wizard hat function) in which close neighbors act as exciters and distant ones act as inhibitors.

The base model, as presented by Amari [1] for the multiple layer case is:

$$\tau_i \frac{\delta u_i(x,t)}{\delta t} = -u_i + \sum_{j=1}^{m} \int w_{i,j}(x,x';t-t') f_j\left(u_j(x',t')\right) dx' dt' + h_i + s_i(x,t) \quad (2\text{-}1)$$

Where τ is a temporal constant of synaptic decay rate, $u_i(x,t)$ is the average membrane potential of the neurons located at position x at time t on layer i (where x can be 1-dimensional, 2-dimensional or even of higher dimension). The average intensity of connection from neurons on layer j at y to neurons on layer i at x is modeled with $w_{i,j}(x,y)$, $f_j(\cdot)$ is the saturating output function which is monotonically non decreasing. The deviation of the average stimulation potential at place x at time y of layer i is represented by $s_i(x,t)$, and $h_i = \bar{s_i} - r_i$ is the sum of the average stimulation potential an the resting potential of layer i.

There are several assumptions that produce simplifications over the previous model. One of them is to include the additional dependence of the time lag of signals $t' = |x - x'|/v$ where v is the velocity of an action potential [65]. Nonetheless, while not stated otherwise it will not taken into account the time lag, as well as the multiple layers. The non-homogeneous terms will also be merged $S(x,t) = h + s(x,t)$. This way, the resulting equation takes the form (for x n-dimensional):

$$\tau \frac{\delta u(x,t)}{\delta t} = -u + \int_{\mathbb{R}^n} w(x,x') f\left(u(x')\right) dx' + S(x,t) \quad (2\text{-}2)$$

For further simplification, temporarily the connection kernel $w(x, x')$ will be considered as isotropic and homogeneous, so that it only depends on the norm of the vector difference $\|x - x'\|$ i.e. $w(\|x - x'\|)$. Amari found diverse stable-state solutions for the one-dimensional case (isotropic and homogeneous), where the model is:

$$\tau \frac{\delta u(x,t)}{\delta t} = -u + \int_{-\infty}^{\infty} w(|x - x'|) f(u(x')) dx' + S(x,t) \tag{2-3}$$

2.4 Evolution and Adaptation

Evolutionary algorithms are a set of population-based heuristic search and optimization techniques. They maintain a population, and apply a set of operators or transformations over its members. Those operators are typically inspired on biological evolution and usually include selection, reproduction and mutation, among others. The operators are dependent of the evaluation of a performance function called fitness function. Generally, fitness function evaluation may include, from a simple numerical evaluation, to a complex simulation, in order to get the performance criterion which its optimization is pursued.

The pseudo-code of a general evolutionary algorithm is as follows:

Algorithm 1 *EvolutionaryAlgorithm*

```
1: P ← Generate initial population of size N
2: Evaluate fitness for each individual in P
3: repeat
4:     P' ← Apply operators to P
5:     Evaluate fitness for each individual in P'
6:     P ← Select N individuals in P' according to a selection scheme
7: until Termination condition is met
```

The most predominant form of an evolutionary algorithm is embodied by genetic algorithms. They most frequent genotypical representation is a bit sequence, although other representations can be used. Usually they are implemented with a generational replacement of population, but in some situations it is useful to conserve a small set of the better individuals across generations in a steady-steady replacement.

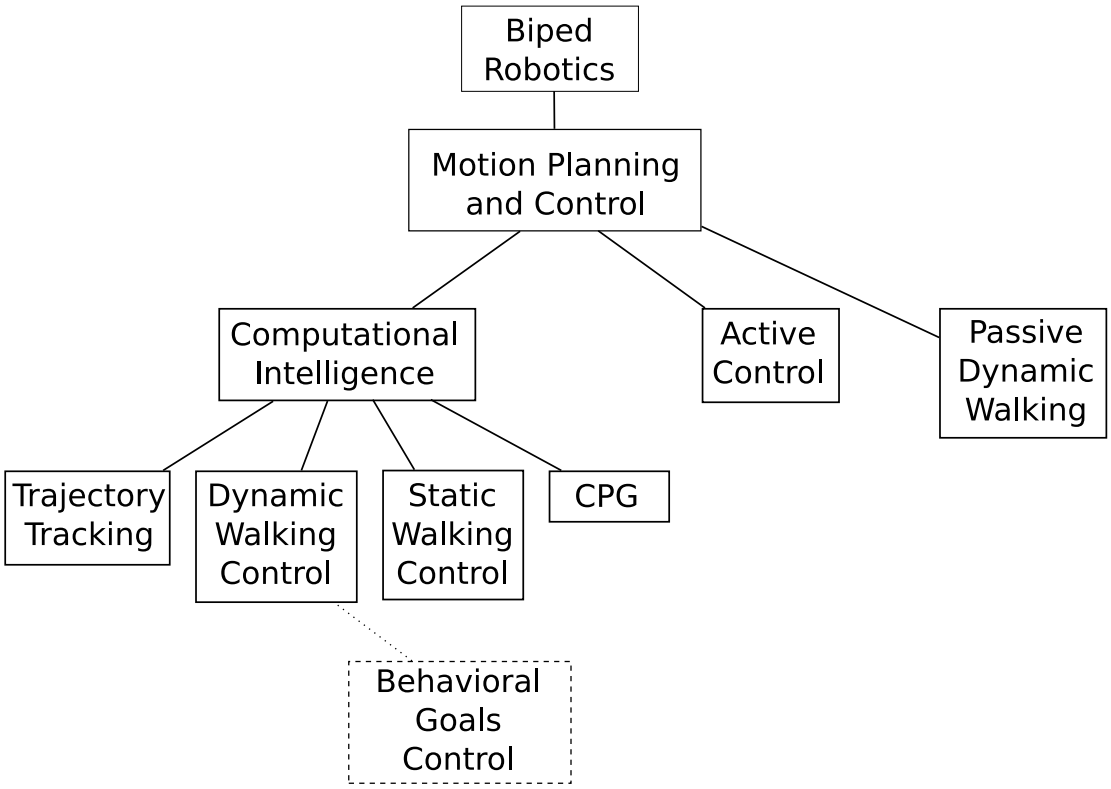


Figura 2-3: A taxonomy tree of motion planning and control methods in biped walking

2.5 Computational Intelligence Applied to Biped Robotics: A Survey

Here is presented a tentative taxonomy of the previous works made in the area of motion planning and control methods in biped walking. Three major approaches are identified: Computational Intelligence, Active Control and Passive Dynamic Walking. Fig. **2-3** shows for a graphical representation of the taxonomy with emphasis in computational intelligence methods.

Before focusing on computational intelligence methods, the two other methods mentioned are briefly presented.

Active Control refers to the persistent control of the joint actuation and state applying control theory. It includes position, velocity, acceleration and torque control techniques. Also, several design methods can be used, including some derived from linear ones but applied to nonlinear systems. In the methods used are included methods based on frequency response, performing the optimization of a performance criterion (e. g. H_∞ control), or applying state feedback. Another

simpler method used is the tuning of parameters of a PID controller. Also it is possible to a given extent use local linear approximations and directly use linear control methods. Active control gives the best trajectory tracking and accuracy, provides methods to evaluate stability (such as Lyapunov stability analysis), and can be vertically integrated to higher motion planning algorithms, but its main deficiency is its very high energy consumption due to the persistent control of joint actuation, even when optimal control is applied.

The other major approach is Passive Dynamic Walking (PDW). In PDW it is took the opposite approach by totally suppressing any joint actuation. The idea is that the actuation given by gravity force over a biped robot standing over a slightly inclined surface, in conjunction with its natural dynamics, must achieve stable gait patterns. This is attained by using passive elements as springs and dampers and a careful mechanical design guided by a detailed analysis by dynamic systems theory, including state space modeling, phase transitions, and probably other techniques such as Poincaré maps and Lyapunov stability analysis and, mostly, dynamic simulation. In order to obtain gait pattern in flat surfaces it can be applied reinforcement learning in a control system so as to learn how to simulate the slight actuation given by the gravity in the inclined surface case. The main advantage of this method is its very low energy consumption, and its main disadvantage is its low flexibility to track general paths, particularly those including vertical movements, and its high sensibility to environment conditions.

Next, each one of the three major approaches to the problem that apply computational intelligence are examined. These are Central Pattern Generator methods, Dynamic Walking Control methods, Static Walking Control methods, and Trajectory Tracking methods. This classification emphasizes the way in which the problem is solved and not the sub-field of computational intelligence used.

2.5.1 Central Pattern Generator (CPG) Methods

A Central Pattern Generator (CPG) is a system which is supposed to give the coordinated rhythmic stimulation to joints required to generate a gait pattern. It is inspired from the spinal motor center find in animals such as mammals, and it is usually implemented using a kind of neural networks called oscillatory neural networks.

Neural oscillators are a special type of artificial neural networks (ANNs) described by two essential elements: the dynamical properties of the individual neuron, and the type of coupling between neurons. Each neuron has its dynamics described by a set of differential equations.

A more specific model of neural oscillators couples neurons in pairs. It is

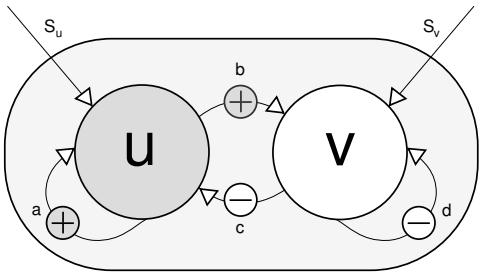


Figura 2-4: Topology of an Amari-Hopfield neural oscillator

based in the fact that most muscle fibers have one excitatory center for flexion and another for extension, and them are inhibiting between them. This way, the two coupled neurons can be represented by a set of differential equations. An example of this arrangement is the Amari-Hopfield model, for which a graphical representation is shown in Fig. presented here as shown in [47]:

$$\tau\dot{u} = -u + af(u) - cf(v) + S_u(t)$$
$$\tau\dot{v} = -v + bf(u) - df(v) + S_v(t)$$

In which u and v are the neuron potentials, $S_u(t)$ and $S_v(t)$ are the respective neuron inputs as function of time, and $f(u)$ and $f(v)$ are neuron outputs after applying the transfer function:

$$f(x) = \frac{1 + tanh(\mu x)}{2}$$

The parameters a, b, c and d characterize the neural oscillator behavior. This kind of arrangement is typically followed in the CPG control methods, where each neural oscillator (neuron pair) is used to stimulate a single joint, and the mutual interaction of neural oscillators conform the CPG.

A classical author in the application of CPGs to motion control of biped locomotion following the methodology presented above is Taga, who in his preliminary work of 1991 (see [62]) applied the concept of coupled neural oscillators, remarking the self-organizing properties of the neural system with the physical system and the environment, which gives some disturbance rejection that keeps the equilibrium of biped walking.

In general terms, the neural oscillator can be thought not only as the coupling of two neurons, but more globally, the rate of neuron activation potential variation can be a function of the network inputs, its current activation potential and a

special term called fatigue. Also the rate of variation of fatigue is function of the current fatigue and its current output. A general expression for the system of differential equations that rule the dynamics of a individual neuron is [10]:

$$\begin{aligned} \tau_r \dot{x_i} &= -x_i + \sum_{j=1, j\neq i}^{n} a_{ij} y_j - b f_i + s_i \\ \tau_a \dot{f_i} &= -f_i + y_i \\ y_i &= H_1(x_i) x_i \\ H_1(x) &= \begin{cases} 1 & x \geq 0, \\ 0 & x < 0, \end{cases} \end{aligned}$$

Here, x_i denotes the activation potential of the ith neuron, y_i its output, a_{ij} the connection weight from neuron j to neuron i, f_i is the fatigue strength, b is the coefficient of fatigue, s_i is a bias factor, and τ_r and τ_a are time constants.

With this model Cao and Kawamura used an oscillatory neural network as CPG to generate biped walking patterns. The net composition includes a single neuron for each articulation, and all neuron are connected between them (i. e. all the networks is a single coupled neural oscillator). The neuron activation function is given by the set of equations presented above. A connectivity matrix, with inhibitory and excitatory weights (composed by the different terms a_{ij}) was evolved using an genetic algorithm to obtain a gait pattern. The chromosome is a linear arrange of matrix elements in binary representation in the form:

$$< \underbrace{b_0 b_1 \dots b_7}_{a_{12}} \ \underbrace{b_8 \dots b_{15}}_{a_{13}} \ \dots\dots \ \underbrace{b_{440} \dots b_{447}}_{a_{87}} >$$

The fitness function used is dived in two partial qualitative functions and a third which evaluates the correctness of the gait, applying three genetic algorithms, one nested in another (authors called it hierarchical evolutionary algorithm). This way they were able to control eight joints and generate successful gait patterns.

Next is presented relevant previous works that mainly use Central Pattern Generators as motion strategy.

Kun and Miller [37] implemented a control for biped robot walking of a 10-DOF robot, for which they used a CPG that uses a heuristic to generate sequences, synchronizing the performance in the joints with the natural dynamics of the robot, and receiving correction parameters in the lateral and frontal motion given by CMAC neural networks as inputs. Additionally, a third CMAC network is trained to help control the robot in the double support phase. The natural dynamics of the robot is fundamental for the control system given its dependency

on directional balance to stabilize the movement. They used temporary difference learning to train networks. This pioneer work is validated implementing it on a real biped robot.

Venkataraman [64] made a complete compilation of the important elements in the CPGs in the field of biology and the robotics, and proposes an alternative method for biped, quadruped and hexapod locomotion patterns generation. The method has relation with the general implementation using oscillatory neural networks, but unlike these, it employs a single linear patterns generator, with which the movements are generated in each extremity using filters of finite dimension (adjustments of phase lead and phase lag filters) to simulate the delays that produce the coordinated movement necessary for locomotion. The oscillator is modeled including elements of the Van der Pool oscillator to obtain a nonlinear system with a robust and stable focus. There are presented as result patterns for the gait types already mentioned, proving that they can be obtained with a simple architecture.

Benbrahim and Franklin [8] developed a CPG using CMAC neural networks and simultaneously applying reinforcement and supervised learning. They use a central network, the generator, aided by a set of peripheral control networks in parallel with observation networks, whose inputs are relevant parameters that act as gait restrictions, among them body posture and height of body mass center. Peripheral control networks act only if its correspondent observation network detects a improper behavior from central network for a specific restriction. Control system output is determined by a Gaussian function with median located in the CPG output and standard deviation that initially has a high level and reduces as solution converges. Dynamic control necessary to follow the reference provided by CPG is easily implemented with a set of PID controllers. In the real application, the control scheme was aided by a posture bang-bang control (i. e. on-off control). It was required a previous training in order to obtain suitable network training time. This novel approach allows to prevent that CPG errors cause a general gait failure, not only providing robustness but accelerating the learning process.

Hasegawa et al. [27] proposed a method for developing pattern generators useful for a biped 13-DOF robot walking on inclined surfaces. They solved the unconstrained optimization problem with a hybrid evolutionary algorithm with a lower layer that generates trajectory points (from a cubic spline) using evolutionary programming (EP), which are added in interpolated point sets, and those set are in time evolved using a genetic algorithm (GA), selecting the successful sets that minimize energy consumed by the robot. Due to that each point evaluation depends on the best individual in the GA layer, and also second layer individuals a sets of elements of the first layer, the algorithm has a co-evolutionary scheme.

In the practical implementation made, the dynamic control is performed by PD controllers.

Reil and Husbands [55] used an evolutionary algorithm (EA) to optimize the parameters of a fully connected recurrent neural network. The model of each neuron is a dynamic model that attenuates its answer in stimulation absence, and depends on a set of weights, a bias factor and a dynamic time constant, with a dynamics similar to the model exposed previously and presented in [10]. The three parameters types for the totality of network neurons are represented linearly in the individuals genotype used in the evolutionary algorithm. The fitness function used favors movement that maximizes displacement in frontal direction and penalizes movement in vertical direction. The authors manage with such simple evaluation rule to generate satisfactory patterns of march without any sensory feedback. Later, they add a sensory entrance of auditive type and, leaving the recurrent connections fixed, they evolve the recurrent neurons connection weights with the sensory input. This way, they attain that the robot navigate towards a source of sound emission. The neural controller generates reference points that are taken to forces in the motors by a PD torque control. An addition to the efficiency of the algorithm is the premature abandonment of marches that move strongly in vertical direction. They have achieved a relative low success rate in evolutions (close to 20%) and propose switching to a more traditional CPG method.

Miyashita et al. [45] recalled the conceptualization of previous authors like Taga, emphasizing the importance of the oscillating behavior in the nervous system in form of CPGs. The proposed controller this way is based on oscillating artificial neurons, or oscillators, which are connected mutually in pairs to obtain a periodic behavior, in a form similar to the Amari-Hopfield model presented previously. It is shown the evolution of neural network structure using genetic programming (GP), in terms of the interconnections between oscillators, supposing the internal structure of the oscillators fixed (i.e. the parameters a, b, c, d an τ). The GP implementation for solving the problem is made representing the local network from each oscillator like an S-expression, doing a co-evolution where a sub-population for each oscillator is evolved, but the evaluation of the aptitude function as well as the application of genetic operators are applied over an oscillator group, each one of a different population. Results of walks generated with this method are displayed, until a maximum of 10 stable steps. This work, spite its relative success, has a very large computational overhead and gives a very small set of successful neural oscillators. Nonetheless, its approach is very illustrative, and somehow resembles the spinal control proposed in [20].

Paul [52] argued that, unlike the controllers that use a CPG whose connections between oscillators in left and right sides are connected, it is possible to obtain a

satisfactory biped walking with two decoupled generators. Two cases area treated: the first one decouples control and sensor system, having each an independent CPG, without any connection to each other and with entrances of sensors strictly located in his partial side; while the second one decouples connections but allows common entrances to both global position sensors. Both controllers are evolved in parallel with genetic algorithms (GA) to three morphological parameters. It is shown that for the first case successful biped walking is achieved in most of cases but there are problems to follow a straight line. In the second, it is shown that the CPGs decoupling with global entrances is an equally satisfactory technique that the entirely coupled case. This way, Paul gives a highly valuable approach, in which a lot simpler technique achieves similar results to those more complex traditionally used. Paul therefore hypothesizes that lower limb control in walking is inherently reactive and sensory-motor coordination based, and CPGs are used only to alter gait conditions acting only in the trunk and arms.

In the another work [53], Paul aimed to continue the work oriented to controller simplification for the biped walking based on neural networks, this time showing that independent simple networks for controlling each leg can generate a stable walk, using only forward connections (i. e. a feed-forward neural network) without hidden layer. The used networks only receive as entrance the contact with their respective leg and a common bias signal for both controller. Of numerous configurations evolved with genetic algorithms, only two obtained the satisfactory long walk, but in a remarkable form, both indefinitely turned out to be stable. It is used, like in previous works of Paul, co-evolution of some few morphological parameters. This is probably the simpler solution to this problem proposed to the time.

Nakanishi et al. [48] made a less traditional approach to the generation of patterns for biped walking with some similarities to the CPGs. The idea consists of designing oscillators represented as systems of nonlinear phase coupled differential equations, in such form that the fundamental element of dynamics is not the oscillation frequency of each element, but its phase relations in order to obtain a desired movement pattern. A local weighted regression (LWR) is proposed as a training scheme to adapt elements phase, as well as dynamic system global oscillation frequency. Additionally, it is applied the concept of phase reset at the moment of heel strike. It is shown that the phase reset principle is advantageous when disturbances in real robot walking appear. Also, it is argued that the phase controller is simpler to train than CPGs. They also affirm the presented controller superiority in front of a controller by finite automata. It is remarkable that this work, first, abstracts the concept behind CPGs and, second, employs the already biologically supported phase reset. A minor flaw of this method is its

high sensibility to initial conditions.

Komatsu and Usul [36] showed the control for different biped locomotion types using Hybrid Central Pattern Generators (H-CPG). The total action of the H-CPG proposed is determined by the sum of the individual actions of each one of its components: A neural oscillator that generates the rhythmical patterns in form of torques, a support force controller that applies the Jacobian matrix to map forces from Cartesian space to joint space, and a position controller that implements a PD control to maintain legs as vertically as possible. The vertical and horizontal movements are separately processed by the force controller. They shown that proposed method allows to vary form slow walking to rapid walking, and also walking and running in modified environments, particularly in slopes. Satisfactory results were obtained in simulations and real robot implementation, and a high movement versatility was attained by complementing traditional CPG techniques with force and posture control. The system complexity in not so high and is indeed feasible.

Computational intelligence methods used are:

- Oscillatory Neural Networks: As nuclear elements of CPGs.
- Evolutionary Optimization: For optimization and training of CPGs
- Rhythmical Dynamic Systems: An alternative to oscillatory neural networks for building CPGs.

2.5.2 Trajectory Tracking Methods

Methods included in this category largely vary in implementation, but its main methodology consists of generating a kinematic pattern or succession, in a way that the joint following of it yields a successful gait pattern. Although there are applied computational intelligence methods in generating the trajectory, this process is typically followed by a simple control method, such as a PID control. Developments under this approach are relatively recent and have strong foundation on evolutionary computation methods.

The work of Capi et al. [12] gave and approach to obtaining a stable biped walk with an optimal power consumption for a biped robot of prismatic joints. The control system is based on the principle of Zero Moment Point (ZMP). The control loop for center of gravity (of PD type) receives a reference of the center of gravity location of the robot, and generates a control signal that receives the ZMP control loop, along with the current ZMP location. In the same way, the ZMP control loop processes that information and outputs actuation values for the rotational joint in the feet and the prismatic joint in the thigh. A genetic algorithm

with real codification is used (equivalent to an evolutionary strategy) to look for the reference trajectory that generates the minimum energy consumption. The results with the optimization for minimum torque applied and optimization for constant height of center of gravity are compared. The problem of optimization with restrictions is turned to a problem without restrictions to which penalties in the aptitude function are added when the conditions of restriction are failed to fulfill. This way, Capi et. al present a gait trajectory generation method for an unusual robot configuration, and also give a forward step to optimization of energy consumption using evolutionary computation.

Later, Yamasaki et al. [73] applied the evolution by genetic algorithms (GA) to design a controller for a biped walker with low torque consumption. For it they proposed the optimization of the gait sequences generated in two phases. The gait sequences in terms of speeds are described like a set of sinusoidal functions with amplitude and phases as parameters, dependent on the angular positions of both legs. The parameters described are optimized along with the functions global angular speed. A binary representation is used, applying cross and mutation operators. The algorithm is divided in two phases, first with a fitness function proportional to total distance walked by the robot during the simulation time (or before falling), and the second one proportional to the walked distance and inversely proportional to the used energy. Therefore, a sequential optimization is made in order to, first, obtain suitable gait trajectories, and second, minimize energy consumed. Nevertheless, the best trajectory obtained attained less than 2 second of walking before falling, and this should be regarded only as a partial success.

Garder and Howin [25] recently applied a hybrid method of genetic algorithms (GA) and hill climbing to evolve walking patterns for a biped robot with pneumatic actuation. In the problem faced by them, the movement of the robot is restricted to sagital plane, as to it is of concern only the robot frontal advance. The genotypical codification used is binary, and the complete sequence is represented by an individual, divided in three mechanism positions with pause times between them. The chosen method consists of leaving the durations fixed and using a traditional genetic algorithm to evolve the three positions that characterize the sequence of movement by 8 generations, and later to make an optimization by hill climbing, leaving fixed the bits of position, and generation after generation optimizing from the most significant bits of pause times to less significant ones. Satisfactory locomotion in the form of synchronous jumps is obtained. The short run time of the algorithm allows to implement it in real time in the robot. The result obtained where facilitated by the low problem dimensionality and relative smoothness of fitness landscape, but the method can fail on more complex

problems.

Contemporarily Yanase and Iba [75] presented an implementation of an evolutionary algorithm with user interaction to determine sequences of movement, as well as other implementations with specific functions of aptitude to reach certain objectives. In the first case, the optimization of a sequence of movement by interactive evolutionary computation appears (IEC), for which a set of alternatives for each picture of movement is generated (keyframes), being these evaluated by the user graphically and, applying evolutionary operators in successive iterations along with the evaluation mechanism, each one of the pictures that are to form the sequence is obtained. Later they show how specially designed fitness functions can be used, altogether with a dynamic simulator, to optimize the movement to seat or kicking a ball. Also, they make a brief reference to the implementation of an multiobjective genetic algorithm (MOGA) for the solution of such problems. This is a novel approach in the fitness dynamical user evaluation employed and also in the alternative objective based approach to evolution.

The application of computational intelligence methods in this approach is focused on:

- Evolutionary Computation: For optimization of trajectories under a performance criterion (such as energy consumption, stability or specific motion objectives).

2.5.3 Dynamic Walking Control

These are probably the most complex methods among those based on computational intelligence. They aim to obtain dynamic stable walking by a control based on various techniques. Among them are included neural networks, fuzzy systems and evolutionary algorithms.

Some concepts used in this section have been already presented, but it is useful to detail recurrent neural networks, which are an important and promising technique. A recurrent neural network is usually modeled as a fully connected network without layered structure [28], in which the output can be expressed in vectorial notation as a linear combination of neuron states:

$$y_{rnn}(n) = Cx_{rnn}$$

And neuron states in $n+1$ are a function of recurrent potentials and input potentials in n, each of them multiplied by weight matrices.

$$x_{rnn}(n+1) = \varphi(W_a x_{rnn}(n) + W_b u_{rnn}(n))$$

The function $\varphi()$ is a diagonal activation function, usually of sigmoid shape. This way, a recurrent neural network is a dynamical system determined by its weight matrices W_a and W_a, its activation function $\varphi()$, and the linear combination of states used as output C. The relevant capability so obtained is that recurrent neural networks not only are a nonlinear mapping from inputs into outputs, but they present memory and dynamical response to inputs, which can give birth to emergent behaviors.

Next, some works in the field of dynamically stable walking are presented, beginning with the preliminary work of Magdalena and Monasterio-Huelin using fuzzy systems.

The work of Magdalena and Monasterio-Huelin [42] dealt with the evolution by genetic algorithms of a fuzzy logic controller and presents its application to locomotion patterns generation in a biped walker. Particularly, a genetic algorithm with binary codification is used to evolve the fuzzy rules as much as the ranks of normalization for each one of the variables involved, although all relevant information is codified to knowledge base, including the number of fuzzy sets by input and output variable and the definition of membership function functions. It is applied crossing between fuzzy rules, rule mutation at bit level and normalization rank mutation. The exposed methodology is used to design a fuzzy controller for a biped walker, which considers the different walking phases. A set of successful biped walks is obtained as result, and also there are obtained rules with better performance than the ones initially provided to the system.

The work of Bergener et al. [9] described an architecture that allows generating patterns of behavior as much as to dynamically control the execution of each task. Unlike other evaluated works, the architecture is applied to an anthropomorphic robot that is not biped and whose similarity with the human morphology is observed mainly in its arm, and therefore the problem faced do not include biped walking. The displayed architecture uses neural fields that map from sensor space to actuator space using principles of dynamic systems whose behavior adapts to conditions of the surroundings through bifurcation phenomena which respond to the so-called instanced dynamics (an elementary behavior parametrized by behavior variables in a specific environment). Also, to generate complex behaviors it is used a competitive dynamic system as arbiter mechanism, which includes a set of parameters that describes the logical and temporal requirements.

In the work of Capi citeCapi01Application the methodology to optimize the biped walk using genetic algorithms followed a procedure very similar to the one shown in [12]. It is observed as difference the application to a robot of rotational joints with 5 degrees of freedom (5-DOF), as well as obtaining both stable biped walking and stable ascent of stairs. Additionally, the data sets generated are used

with the genetic algorithm to train a radial-basis neural network (RBFNN) with an hidden layer, of such form that locomotion patterns could be generated in real time.

In the novel work of Paul and Bongard [54] it is proposed a coupled evolution of morphology and control, since they emphasized the relative little depth with which the morphological evolution (and in general the morphological configuration) for obtaining gait patterns has been studied. The methodology applied consists of evolving simultaneously the morphology, in terms of the distribution of discrete mass blocks in a fixed joint configuration, and the control, in terms of the weights of a recurrent neural network. They made several experiments, varying the mass proportion that can be redistributed, with the purpose of evaluating the morphological modifications in micro, meso and macro scales.

Juang [34] presented a method of trajectories generation using a feed-forward neural network as a nonlinear mapping between the present cinematic state and the control signal. It is based on the principle of periodic of the human walking, looking for to train the network of such form that diminishes the difference between the wished state and the end state, applying recurrent averaging learning in which the experiment is repeated consecutively initiating and finalizing in states determined by the average of the previous beginnings and ending, so that robustness in the system is obtained. The network is dynamically trained using back-propagation trough time.

Juang [33] also made an application of a neural control system for biped walking in slopes. The control system consists of: a neural controller incarnated by a feed-forward neural network that generates torque signals in the actuators, a neural estimator that also consist of a feed-forward neural network, and a third network with the same topology that adds a compensation control signal when there is an inclination in the surface. The method of training used is delayed back-propagation. The first step made is the identification of the system with the neural estimator. Later, the neural controller trains so that it follows a preassigned trajectory on flat surfaces, using the estimator to propagate the return error towards the control network. Finally, the two previous networks are left fixed and the inclination compensator, coupled with the control system, is trained using the estimator to propagate the error backwards. The obtained trajectory tracking, even in slopes, is satisfactory.

Wu et al. [71] studied the problem of inverted pendulum control excited in the base with two rotational degrees of freedom and non-acted movement in the base. There is a relation of such problem with the control of the trunk in biped walking, where the trunk is modeled as an inverted pendulum with such configuration. In order to obtain its objective, they use a set of feed-forward neural networks with

a hidden layer, which generate different functions that are connected in a simple closed loop configuration. An additional neural network is used to inversely model the system of such form that it is not required to measure the base position and an estimate is used instead. The controllers are pre-trained offline but its adjustment is dynamically made on-line providing adaptability to the controller. It is shown that the proposed control system evolves better than novel systems designed by control theory methodologies and in addition it does not require a model of the system nor a direct measurement of the position of the base.

Previously, on the same sense of previous work made by them, Wu et al. [72] studied the problem of inverted pendulum control with angular excitation in the base and free base translation in the three-dimensional space. They used then four neural networks to inversely model the system. Also a feed-forward control with one neural network is used, and its output is feed back to the circuit. The inverse model is used to train the controller network.

Zhou [78] developed a learning agent with fuzzy and reinforcement learning (GAFLR). He had previously conceptualized several versions of agent GAFLR and its application the control of the biped long walk of a robot. The basic characteristics of the agent are: 1, Parts of a fuzzy knowledge base designed by an expert; 2. It is updated dynamically (on-line) using as reinforcement signal some parameter calculated with the fuzzyfication of system state measurements and internal estimations of future reinforcement signals; 3. The estimation of fuzzy reinforcement signals is made by a neuro-fuzzy network of 5 layers that is trained by reinforcement using a temporal technique of difference (temporal differences); 4. The actions are suggested by a neuro-fuzzy network of 5 layers trained with a genetic algorithm whose genotypic representation consists of the fuzzy rules that will be used for train the network directly; 5. A stochastic modifier of actions takes the composed reinforcement signal as the suggested action and generates the output. A fast convergence is exhibited towards the successful walk of the robot.

Zhou [79] developed also a simpler version of GAFLR agent, but without learning by genetic algorithms. There are shown the previous conceptualization and several versions of FLR agent and its application to robot biped walking control. Basic agent characteristics are shared with GAFLR version having discounted the training with GAs (the training by temporary differences is conserved). The operation of the stochastic action modifier module (SAM) is exposed better as an actions generator with normal distribution with average in FLR agent output, and standard deviation given by the reinforcement signal of previous iteration.

Park [51] applied the conceptualization of gait viewed from the zero moment point (ZMP), looking for, instead of generating movements for each one of the

joints of the biped walker, dynamically generating the trajectories of the ZMP according to the hip joints state and the leg in balance, using a fuzzy logic system for trunk trajectory generation. Thus, it generates a natural movement and with reduced hip displacements. In order to calculate the movements of the joints once found the location of the ZMP, the inverse kinematics of the mechanism are used and a control by computed torque is applied. Leg trajectories are input to the trajectory generation system.

The hybrid algorithm proposed by Liu et al. [41] implements a controller for the double support phase in biped walking divided in three components. First it uses a CMAC neural network for which the fuzzy sets are generated, identified as generalization sets of the network, with which the input variables are mapped to a generalization space, obtaining the output as a linear combination between the representation generalized for the set of entrances and the matrix of weights of the CMAC neural fuzzy network. Later a hybrid position/force model is derived for the system including restrictions, with which later a model for the hybrid system is found applying a control scheme that includes the use of the neural fuzzy network. It was implemented an H_∞ optimization parameter for the joint model, which is optimized. Finally a robust hybrid controller was obtained which uses the neural fuzzy networks for the control of the inverse system and control of variable structure. Therefore, a robust method for desired trajectory tacking in double support is obtained. The study of techniques of switching system theory is suggested for, connecting them with the proposed control technique, integrally controlling the several gait phases.

Yamasaki, Nomura and Sato [74] emphasized the importance of dynamical phase modification (phase reset) for gaining stability in the biped walk, which is to be applied in the central patterns generator (CPG) implemented as a neural network, due to biological motivations. For it they approach two types of mechanical systems: first, a double pendulum, and second a 5-links biped walker, for which they develop the dynamic equations. Also they develop the dynamics of the neural controller in general terms and show the results in phase space when phase reset is applied and when it is not. Two effects are verified: 1. The phase reset in a suitable magnitude can relocate the system within the attractor stability basin after it has been put under a disturbance, obtaining stability; and 2. It can be useful to diminish convergence times to stable state.

Hülse et al. [31], aimed to show that minimum and robust control structures can be developed using algorithm ENS3 presented by them, of evolution of neuromodules made up of neurons with feedback connections , i. e. recurrent neural networks. To such networks an evolutionary algorithm is applied, where evolutionary operators are applied in each iteration: reproduction, variation (a type

of mutation), evaluation and selection. In that process changes in the number of hidden neurons, their number of connections and the weights of such connections, are made. Once reached a satisfactory performance, there are reduced the number of neuro-modules elements, introducing costs by neuron and by connection, to urge the minimalism of controllers. The application of the complete procedure is shown in three examples: the expansion of a mobile robot for obstacle evasion to luminance tropism, the morphological and control evolution for a biped walker robot with minimum actuation, and the co-evolution of neuro-controllers for a ring of gravitational impulsion with five actuator arms. An additional initial example is shown which corresponds to robot evolution for obstacles evasion and serves as guide for the controller development methodology according to the proposed algorithm.

Jha et al. [32] made the controller design for stairs ascension of a biped robot modeled in 2D. They use two fuzzy logic controllers to conserve robot stability according to dynamic stability margin criterion based on zero moment point (ZMP). The first controller is in charge to maintain stability during single support phase and the second to maintain it in double support phase. The controllers are, in a first approach, manually designed, and later the obtained controllers are optimized using genetic algorithms (GA), and in a third approach they are designed entirely by the AG. The low run times of the optimization suggest their possible application in real time, but it is necessary to implement ZMP control compensation when the controller does not manage to conserve the stability.

The approach taken by Kurz and Stergiou [40] to biped walking emphasizes the chaotic properties, already stood out by other authors, proposing them like a beneficial characteristic for control. They indicate that the capacity of a chaotic system to present equilibria with different periodic characteristics, added to the possibility of changing from one to another applying a small located actuation, facilitates the system robust control in presence of disturbances. They implemented a neural network whose inputs correspond to positions and speeds of initial states for the 8 previous periods of gait and its output is the hip actuation applied as control. The concluded that biological control of human walk can respond to similar phenomena, considering a more complex hierarchic structure in the neural network control.

Sabourin and Bruneau [56] applied a control based on trajectory tracking generated by a CMAC neural network to control the biped walking. Initially they calculate a set of rules or phases, of active and passive type, which generate partial actuation in different joints according to the geometric configuration in which is the biped walker robot. Later they trained a CMAC network so that it generalizes the trajectories needed to generate the walk, particularly for swing leg. The knee

of supporting leg is blocked and its angle is determined by a high level control that, receiving a speed as parameter, modifies the angle to obtain a gait with the desired speed. The tracking of trajectories generated by high level control (support leg) and CMAC network (swing leg) is implemented with a PI control. It is shown the method robustness putting it under disturbances in floor surface, applying loads and sliding.

Computational intelligence methods mainly used are:

- Recurrent Neural Networks: For dynamic control and, in a novel approach, for a dynamical systems perspective of planning.
- Fuzzy Systems: For dynamic control.
- Hybrid Methods: Control, occasional correction of movements and learning.
- Evolutionary Algorithms: For optimization of planning and control schemes and as an alternative to learning methods.

2.5.4 Static Walking Control

The static walking refers to the achievement of locomotion by conserving static stability along all the path. It causes that a characteristic of locomotion patterns derived from this method is the very low velocities required to make negligible the inertial forces that can be generated, in such a way that they can be classified as quasi-static walking methods. This methods have been displaced by most elaborate dynamic methods, but they still have a research value in studying the stationary phases and posture of bipeds and some specific movement patterns.

Kun and Miller [38] studied again the biped walking problem, but now in static balance. Unlike [37], in this work they aimed for biped locomotion with quasi-static motion and, therefore, the stability criterion required that the projection of the CoM was at any moment within the supporting polygon, thus being able indefinitely to remain in any position. The trajectory generation, like in their previous work, is made with CMAC neural networks, starting off of pre-calculated but adaptable trajectories. In this case a network for foot elevation, one for forward movement and another one for the lateral motion are used. As the kinematics model used is an approximation, other two CMAC networks are used to make corrections to gait patterns such that they compensate model inconsistencies. The generator accompanied with a low level control that is in charge to map the generated posture to joints space, to make the trajectory tracking, to hold the perpendicular position to the ground in the double support phase and to make a lateral and frontal reactive control. A satisfactory gait in a 10-DOF robot with

a speed of 2,2 passages per minute was obtained. Is remarkable the similarity of this method with that used in [37] despite of the dynamic orientation of the latter.

Miyakoshi et al. [44] studied the problem of open loop stabilization in periodic movements using neural network oscillators. They showed the satisfactory application to a juggling problem, and also to the biped walking with static balance (stepping). In this last one, it is used a pattern generator with an oscillator for the frontal plane and two coupled oscillators for the sagital plane, as well as a PD position control with inhibition from the sagital plane oscillators. Also they applied the method to biped walking in dynamic balance but obtain only some steps, since the method becomes unstable. They propose to deepen the study of the open loop control of the dynamic biped walking. This work exposes a divergence from the mainstream of walking control methods in that it emphasizes the role of open loop control and argues to its favor.

Wolff and Nordin [68] presented the evolution with evolutionary strategies (ES) of a control sequence to obtain biped walking with static stability for a robot. The evaluation of the fitness function is made directly in the physical system using a camera to measure direction and an infrared sensor to measure range. The evolution is made using a steady state algorithm which eliminates some of the individuals of an iteration, and manually makes the search in a zone defined by the Euclidean distance to a certain gene in the generated sequence. It is also shown that the evolved controllers performed better than the designed ones. The main contribution of this work is the physical fitness function evaluation which is seldom found in works on the field.

In another work Wolff and Nordin [69] they tried to generate a pseudo-dynamic walking, evolving points that are statically stable in the double support phase and interpolating in the balance phase. For it they make a evolution using evolutionary programming (EP) and representing the controller as a series of instructions with basic arithmetic operators and trigonometric operators, all of them represented like chains of integer numbers. The sequences thus evolved later are transferred to the robot, which can make fine adjustments or recovery of failures in a evolution second phase. The robotic platform used was the same one used in [68]. This last method is a hybrid method between dynamic and static walking control, in which an notable on-line failure recovery strategy is implemented, but the general characteristics are still in disadvantage with dynamic walking control methods.

The two mayor computational intelligence methods used in static walking are:

- Neural Networks: To compensate and map trajectories.
- Genetic Algorithms: To optimize motion sequences.

3 Neural Fields as Control and Planning Systems

3.1 Introduction

Artificial life aims to devise and study those emergent phenomena that give complex attributes to the living beings, like self-organization, cooperation, self-reproduction and adaptation, among others [7]. It focuses on the generation of behavior from a biological, bottom-up perspective that relies on evolution, development and learning [21].

Following a pure artificial life approach to evolutionary robotics [49], it is expected that complex behaviors of simulated agents emerge by local interactions of elements. These interactions form a complex dynamical system which is the generator of behavior and is capable of some form of adaptation or evolution. Examples of such dynamical systems are recurrent neural networks [31] and feed-forward neural networks [59].

One of the tasks in evolutionary robotics is the emergence of motion control and planning capabilities of biped walking agents.

In biped robotics, the methods based on computational intelligence for planning and control have shown to be able to achieve static stability [38], dynamical stability [48, 36], achieve simple control structures [31], and tolerate perturbations [34]. Nonetheless, those properties have not been extended to an integrated architecture of planning and control capable of following goals (for details see the previous chapter).

Here, there are compared two control schemes based on neural networks in order to observe their advantages and disadvantages as planning and control architectures and their suitability as controllers given their structure. The first one, uses a simple model of recurrent connections between neurons without additional restrictions, i.e. a recurrent neural network. The second one, based on neural fields, has a deeper biological basis, applies more restrictions and extends the discrete model to a continuous one, following the method of planning and control by means of neural fields [9]. The neural field model, as noted in the article by Bergener et al., has the potential to address goal-based planning problems, so we

are here interested on its capability to solve dynamic control problems.

The control problem set for testing the architectures is the stability problem on a inverted pendulum, so that the controller ability to perform dynamic control on an unstable plant can be evaluated. This is a first attempt (to the authors' knowledge) to evaluate neural fields for solving a control problem for an unstable plant, and specifically for solving the inverted pendulum control problem. However, there is a wide range of methods that have been applied to the inverted pendulum problem both based on computational intelligence (e.g. [2, 5, 46, 77]) and based on control theory techniques (e.g. [35, 30, 13]).

In this chapter we aim to propose a control planning system or architecture based on neural fields which is suitable to control a relatively complex system. We test it over the stability problem on a typical inverted-pendulum and compare it against a more traditional recurrent neural network controller. First, we present the neural fields model, some variations of it which will be useful for its evolution and some of the properties that arise from it. Next, we study its applications to control and compare it with the recurrent neural network control scheme. We briefly compare its properties with traditional control schemes, and then we test it with the inverted-pendulum problem. Finally, we show how evolution algorithms are applied to the neural field, and discuss the results obtained. The contents of this chapter were published, in two parts, in the Proceedings of the International Joint Conference of Neural Networks (IJCNN) 2009 and the Genetic and Evolutionary Computation Conference (GECCO) 2009 ([22] and [23]).

3.2 Neural Fields for Control and Planning

For the purpose of control and planning we need some particular requirements on the neural fields.

The first one is to have a preprocessing over the input obtained from the sensors, so that there is a closed loop where the representation of inputs has an appropriate form. This mechanism alone (a particular form for the inputs) has shown to be enough for the robot ARNOLD to navigate in the plane with obstacles [9].

The second one is to be able to modify the connection kernel so that it can be suitable to our control problem. In order to do that, we will consider that the connection kernel $w(y)$ is a symmetric function (i.e. $w(y) = w(-y)$), that also is a 2-power Lebesgue integrable function so that it also belong to L^2. It can be shown that, with that definition, a sum of an arbitrary number of kernel functions will also be a kernel function. This way, we have a inner-product defined by the Lebesgue measure:

$$\langle f, g \rangle_{L^2} = \int_{\mathbb{R}} f \cdot g d\mu \tag{3-1}$$

The defined space, with its measure, conforms a Hilbert space, and therefore is complete and metrizable. It also gives a notion of sum, and scalar product:

$$(f + g)(x) = f(x) + g(x) \tag{3-2}$$
$$(\lambda f)(x) = \lambda f(x) \tag{3-3}$$

Those properties will be used shortly, when arises the problem of kernel evolution.

The third one consists of its suitability to simulation. This is not an inherent restriction for it to be physically (or biologically) plausible, but to be implementable on a computer. We will take a discrete form of the equation 2-2:

$$\tau \dot{u}_i = -u_i + \sum_{x_j \in B_p(x_i)} w\left(x_i, x_j\right) f\left(u_j\right) + S(x_i, t) \tag{3-4}$$

Where we replace the integral for a sum over the point included inside a finite neighborhood (ball) around x_i with radius p. The time is considered continuous, and the computation of the dynamical system behavior is evaluated with a Runge-Kutta method. We denote $u_i = u(x_i, t)$. It should be noted that the previous equation can be applied to the n-dimensional case without modification.

3.3 Control Architecture

The control architecture built based on the neural fields has three basic elements.

The first one, is a sensor, which reads the states from the plant and also their derivatives (computed from the dynamical equation of the plant). In particular, the sensor used for the neural field controller is based on the angular acceleration of the pendulum pole, loosely resembling the vestibular system on the inner ear.

The second one is the input layer, which consists of a simple neural field without natural dynamics, where the spatial codification of the sensed values is made. For the problem at hand, we use a finite one-dimensional neural field, where a sensed input with value zero maps to the center point of the field.

The third one is the processing layer, which has a more typical neural field which has inner dynamics given by the eq. 3-4, where the fields taken into the sum are the input neural field, and the processing neural field. This way, besides

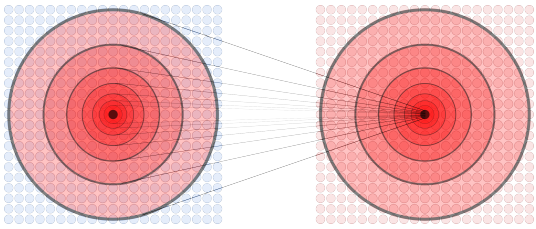


Figura 3-1: Neural fields for stability control

its natural dynamics, the processing layer receives the inputs from the input field filtered by the kernel operator.

The parametrization of the controller is performed by varying the kernel operator. For the hand-tuned case, the kernel operator used is a Wizard Hat Function with the expression:

$$w(x_i, x_j) = ke^{-(x_i - x_j)^2/\delta^2} - H_0 \tag{3-5}$$

Where the diverse parameters work as vertical (k) and horizontal (δ) scaling, and as vertical offset (H_0).

The kernel operator for the evolutionary-tuned case is evaluated with the expression:

$$w(x_i, x_j) = \texttt{kernelArray}[|x_i - x_j|] \tag{3-6}$$

Where `kernelArray` is the array of parameters modified by the evolutionary algorithm. The kernel value is evaluated by accessing the array at position $|x_i - x_j|$.

The additional term on the eq. 3-4 $S(x_i, t)$ is used only as the uniform and static resting potential, that is $S(x_i, t) = -r_p$. The firing rate function $f(u_i)$ is simulated as a simple Heaviside function.

The output is processed taking the position with highest activation on the processing filtered by another wizard-hat function, and decoding it to a value.

The figure **3-1** illustrates the input and output layers (in the 2-dimensional case for generality) and the participation on the potential of a single element in the processing (or main) layer from the elements in the same layer and in the input layer.

3.4 Evolution of Neural Fields

3.4.1 Neural Field Controller Architecture

The architecture used for the neural field controller uses a structure similar to that of multilayer perceptrons, i.e. an input layer, a hidden layer and an output layer. The hidden layer has the properties so far presented in the neural field model. The input and output layers are also modeled as a population of neurons but without inner dynamics. Nonetheless, it is used a kernel for the connection from the input layer to the hidden layer, as well for the connection from the hidden layer to the output layer. The input layer is used as a buffer where sensory inputs are placed before they are processed by the hidden layer. The output layer is used so that it can be applied some form of post-processing to the output of the hidden layer without changing the inner dynamics of the neural field.

3.4.2 Evolutionary Algorithm Structure and Parameters

For the evolution process it is used a simple evolutionary algorithm as shown in the preliminaries, with random elimination of individuals inversely proportional with its fitness.

The evolution parameters are the connection kernels between the input layer and the hidden layer, and between the hidden layer and the output layer. The recurrent connections of the hidden layer with itself are left fixed, in the form of a wizard hat function.

The connection kernels are considered isotropic and homogeneous along the field, so that they can be described as symmetric one-dimensional arrays of values.

3.4.3 Genotypic Representation and Evolution Operators

Each connection kernel can be represented as an array of N values from $w(0)$ to $w(p)$ with homogeneous spacing, using their symmetry. This way, for an equal boundary radius for all the kernels, and a 2-layered architecture, there are $2N$ real values in the genotype. As can be seen, the number of evolution parameters does not have a direct relation with the simulation size of the neural fields (the number of discrete points used), in contradistinction with recurrent neural networks, where the number of parameters depends on the number n of neurons with a polynomial order $O(n^2)$. Nonetheless, here is taken a more general approach and the boundary radius is set equal to n, so that there are $O(n)$ parameters.

The evolution operations used in both steps are:

- Parametric mutation of input array: Gaussian modification of real codified array values, which varies the connection kernel between input layer and processing layer.
- Parametric mutation of recurrence array: Gaussian modification of real codified array values, which varies the recurrent connection kernel of the processing layer.
- Selection: Calculates population fitness, selects with elitism and culling (5% of both) couples of parents for generating new off-springs, calculates the fitness function for both off-springs.

The mutation operators implemented apply the sum operator defined in eq. 3-2 but there was not implemented a crossover operator that used the scalar operator as it was not deemed necessary, but can be easily implemented for another application if it is considered useful.

3.4.4 Fitness Function

The fitness function is selected in such a way that the stability controller mainly has the goal to reduce inclination. It was tuned experimentally to attain a convergence velocity suitable for the experiment. This has in mind a notion of sequential evolution of, first, the capability to attain equilibrium, and later, the capability to perturb the equilibrium controller in such a way that a planned trajectory can be followed or a reference can be tracked. Here we are interested only on the stability problem.

The fitness function for the stability controller is:

$$F_1 = 100 - \frac{100}{(\pi^4 + 2)T_{total}} \sum_t \left(\theta(t)^4 + \frac{|x(t)|}{10} \right) \tag{3-7}$$

This fitness function aims to minimize the orientation error, but also has as a minor second goal to minimize the total horizontal displacement.

While the above expression was used to get the fitness value, the actual fitness function includes running a simulation instance of the control problem with a neural field controller grown from the two kernel arrays.

3.5 Experimental Framework

The model used consists of an approach to biped walking based on a inverted pendulum (car-and-pole) system in which the pendulum equilibrium is looked

for. Nonetheless, supposing that the pendulum mass represents the body center of mass, it is proposed that is reasonable to expect a system with its sole function being to stabilize the body. This way, the navigation system has as purpose to carefully perturb the first controller in such a way that the stabilizing controller moves the car to the desired position. Here we are particularly interested only on the stability problem and controller.

3.5.1 Dynamic Model

The dynamic model used, in mathematical terms, is expressed in the two equations:

$$\ddot{x} = \frac{F + ml\dot{\theta}^2 \sin\theta - mg\cos\theta\sin\theta}{M + m\sin^2\theta} \tag{3-8}$$

$$\ddot{\theta} = \frac{(M+m)g\sin\theta - F\cos\theta - ml\dot{\theta}^2\sin\theta\cos\theta}{l(M + m\sin^2\theta)} + \frac{\tau}{ml^2} \tag{3-9}$$

This model consists of four state variables and a high non-linearity as it departs from equilibrium points. It is worth noting that the wanted equilibrium point is in fact unstable.

The output from the stability controller maps to the lateral force F. The angular actuator with value τ is left to a value of zero, to allow the plant to behave according to its natural dynamics on the angular coordinate.

3.6 A RNN Approach for Comparison

The proposed architecture for the recurrent neural network controller has two expert recurrent networks, whose interaction will achieve positioning and equilibrium as well.

There has been applied a preprocessing stage previous to the input neurons, so that the actual values are not used and instead the inputs are mapped to 3 fuzzy sets. In this way, the stability controller only has 3 inputs, while the positioning controller has 6, corresponding to the same 3 inputs previously described and another 3 due to the fuzzy mapping of the error signal. All neurons are interconnected and the first one of them is selected as output without loss of generality.

3.6.1 Evolutionary Algorithm Structure for the RNN Controller

It is expected, based on the approach of artificial life to evolutionary robotics (Nolfi and Floreano), that the sequential and cooperative evolution of elements with biological similarity leads to an specialization in the process of stabilization and positioning (despite the antagonistic individual goals of each controller because of the interest of the positioning controller to maximize also the global performance).

As said, the two steps are executed sequentially, taking the best individual of the first step to collaborate with the individual evolved in the second step.

Aiming to obtain a fixed length representation and limit the problem dimensionality, it is used a model of order Q totally connected. Any network with an order equal or lesser and with total or partial connections can be represented by the proposed model, by the addition of activating/deactivating elements for neurons and connections. Therefore, individual are codified as:

- A bit sequence representing a serialization of an activation matrix A_a of dimension Q-by-Q which activates/deactivates a recurrent connection.
- A sequence of real numbers representing a serialization of matrices W_a and W_b, of dimension Q-by-Q and Q-by-$(m+1)$ respectively.

The C matrix is not evolved because it is chosen arbitrarily only one output (the first neuron).

The evolution operations used in both steps are:

- Parametric mutation of inputs: Gaussian modification of real codified matrix weights, which varies connection weights of inputs.
- Parametric mutation of recurrences: Gaussian modification of real codified matrix weights, which varies connection weights of recurrences.
- Selection: Calculates population fitness, selects with elitism and culling (5% of both) couples of parents for generating new off-springs, calculates the fitness function for both off-springs.

The fitness functions used are the same presented for the neural field controller.

3.7 Experimental Results

Experimentation Details

The sampling time used was 0.025s (for neural networks, neural fields, and visualization) and 10 s tests were performed.

The differential equation system was solved by a fixed-step numerical method, 4th Order Runge-Kutta. The iteration step selected was also $h = 0.025s$ for each test.

Here are shown the results obtained for:

- The evolved RNN controller
- A (non-evolved) neural field controller only with an input layer. It uses directly the input field activation to control the system (that is, it actually does omits the processing field to perform the control).
- The non-evolved neural field controller. An appropriate selection of parameters is applied (made taking into account only the self-stability of the neural fields and the time constants of the plant). It behaves roughly as a proportional state space controller.
- The evolved neural field controller (also with an appropriate selection of parameters).

For all the simulations, the initial angular position was $\theta = \pi/6$, the number of discrete positions used in the simulation of the neural field was 21, and the angular position θ was encoded into the input field with value 1 while the angular velocity ω was coded with value $k_\omega = 0.5$. The neural field time constant was taken with value $\tau = 1/10$s. The filtering (wizard hat) kernel on the actuator had values $k = 1$, $\delta = 2$ and $H_0 = 0$. Also, the maximum control signal energy was equal for all the simulations (and architectures) presented.

For the non-evolved controller, it were used also wizard hat kernels as defined in eq. 3-5. The input kernel had values $k = 2$, $\delta = 2$ and $H_0 = 0.1$. The processing kernel had values $k = 0.3$, $\delta = 2$ and $H_0 = 0.1$. Those values were derived mostly by trial-and-error.

For the evolved controller, there were used parametrized kernels feed by the kernel arrays on the genome of the evolutionary algorithm. The Gaussian mutation operators were initialized with a standard deviation of 0.3. Its application rate is evolved itself by the evolutionary algorithm used, which adjusts the operators rates on-line (Hybrid Adaptive Evolutionary Algorithm).

Results

RNN Controller The first experiment is performed using the recurrent neural network controller without evolution. Results are presented in the figure **3-2**. The figure shows the natural dynamics of the system when the controller is randomly parametrized. It can be perceived the need for the parametrization made

by the evolutionary algorithm on the recurrent neural network controller, since the inverted pendulum is an unstable system around the origin, and a random controller can not stabilize it. Red dots represent the pole mass location and blue dots represent the cart position.

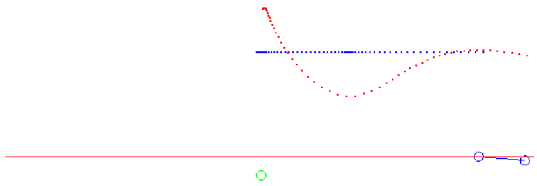


Figura 3-2: System dynamics with an untrained RNN controller. The first figure shows the pendulum trace, and the second shows the pendulum at $t = 3.5$s.

The second experiment shows the behavior of the inverted pendulum once the RNN controller has been evolved. Results are presented in figure **3-3**. The adaptation made by the evolutionary algorithm, as is evident, has an important positive effect on the controller.

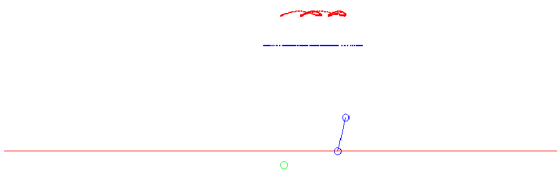


Figura 3-3: System dynamics with a trained RNN controller. The first figure shows the pendulum trace and the second the pendulum at $t = 3.5$s.

RNN Controller and Neural Field Controller Comparison The next three experiments were performed with: an evolved RNN controller architecture, a (non-evolved) neural field architecture using the processing field as output, and a (non-evolved) neural field architecture using the input field as output (effectively omitting the processing field in the feedback loop). In the remaining, the neural architectures will be simply called '(non-evolved) input field architecture' (the architecture omitting the processing field) and 'processing field architecture' (the non-evolved architecture using the processing field in the feedback loop).

The qualitative behavior of the non-evolved processing field is shown in the figure **3-4**. It can be seen that, when an initial angular perturbation is small, the neural field is able to control the stability without evolution.

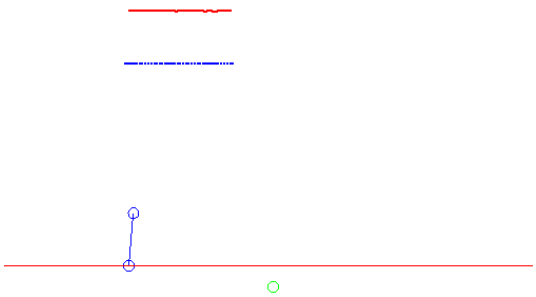


Figura 3-4: System dynamics with a non-trained Neural Field Controller.

Results of these three experiments are presented comparatively in figure **3-5**.

In the upper part of figure **3-5**, the behavior of the angular position state variable (θ) is shown. This is the central variable of the control task, since its minimization would mean also a minimization of the error signal (viewed from a classical control theory perspective). Both the recurrent neural network architecture and the input field architecture (marked as 'input field') show an oscillating behavior with wide movements around the reference value ($\theta = 0$). Of them, the RNN architecture has the fastest response, but also the widest oscillations. On the contrary, the input field architecture shows diminishing oscillations with increasing time.

On the other hand, the processing field architecture (marked as 'processing field') presents the best performance, staying close to the reference (with an error $|e_\theta| < 0.05$) from $t = 1.0s$ onwards.

The middle and bottom parts of figure **3-5** show the state variables ω (angular velocity) and v (linear velocity). The angular velocity plot reinforces the perceptions given by the angular position plot. It also shows that the responses of the RNN controller and the input field architecture are smoother than the response of the processing field architecture.

The linear velocity plot evidences that all controllers deviate significantly from the initial linear position in the experiments. This is expected, since the focus was on balancing of the inverted pendulum, and not on its linear positioning.

While the results shown by both neural field architectures are not ideal (none of them seems to achieve a zero error signal), their results are certainly better than expected. This is notable, considering that the RNN was evolved to solve the task at hand, while the neural field architectures were manually parametrized, and there were not applied any adaptation or evolution schemes to their parametrization.

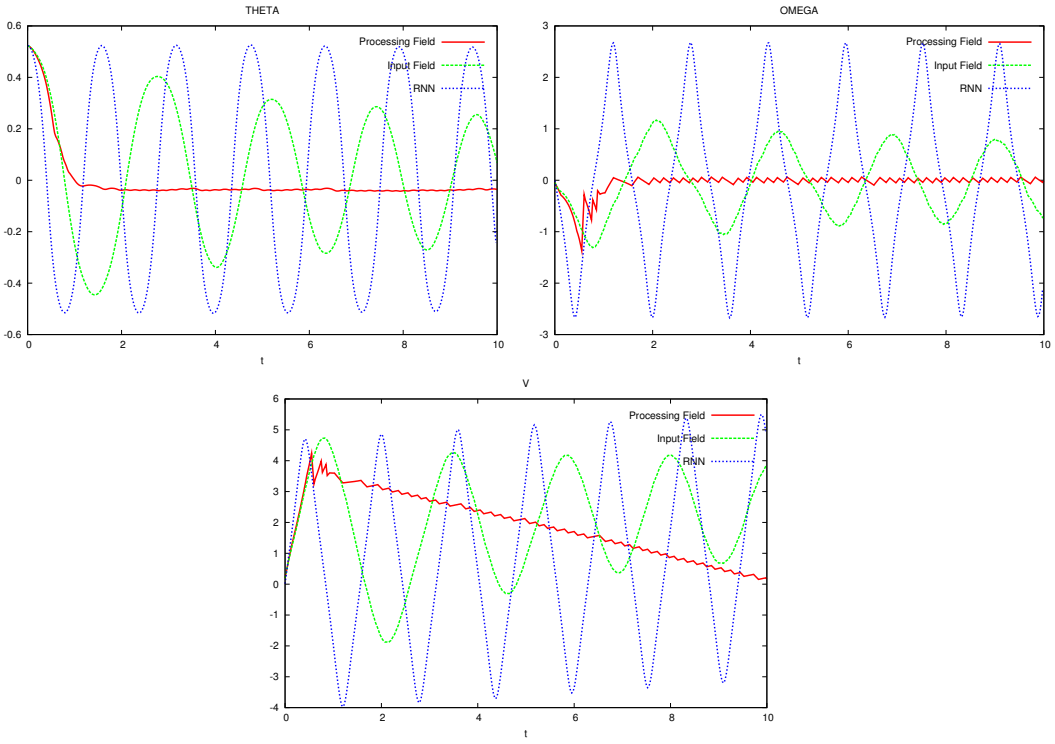


Figura 3-5: Pendulum simulations with neural controllers between $t = 0$s and $t = 10$. The three controllers shown are: a RNN controller, an input field controller and a processing field controller. In the top, the angular position is plotted (θ). In the middle, the angular velocity is plotted (ω). In the bottom, the linear velocity (v) is plotted.

The recurrent neural network controller is expressive enough to solve the problem at hand, but the number of parameters to configure (or in this case to evolve) is of a quadratic order in relation to the number of nodes (or neurons). This was not a particular problem for the evolutionary algorithm used, but it can be a limitation if an adaptation scheme is not applied. Furthermore, the evolutionary parametrization applied was not able to outperform the manual parametrization of the neural field architectures.

On the other hand, the neural field controller is a bit more complex (in its implementation) and its simulation more costly (up to 10 times slower than the recurrent neural network), but has some notable advantages. The first one is its ability to self-compensate or, equivalently, the stability of its natural dynamics, which is attained after the setup of few parameters. The second one is its suitability to the problem at hand, being able to solve it with a good performance. Although there was a need for parameter configuration, evolution was not required because the number of parameters to setup is small: basically three parameters of the kernel function and the resting potential of the field equation — a number of parameters of constant order in relation to the number of nodes (discrete elements on the neural field).

Evolved and Non-Evolved Neural Field Controller Comparison A last experiment is performed using the evolved neural field controller architecture, for the same problem and experimental setup. Its qualitative behavior is shown in figure **3-6**.

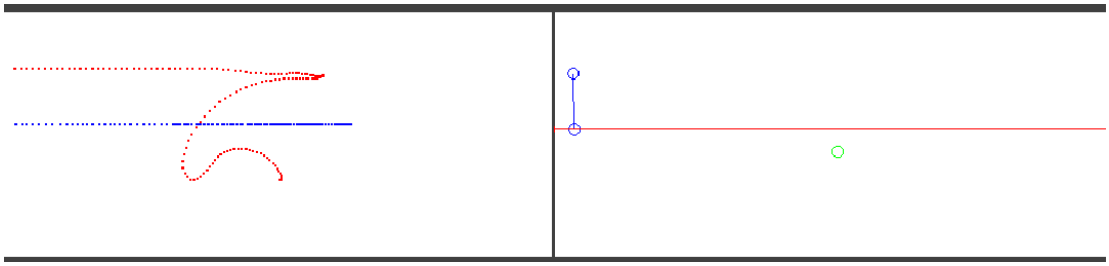


Figura 3-6: System dynamics with an evolved Neural Field Controller, at time $t = 4$. Left: trace of the pendulum (cart in blue and pole tip in red). Right: snapshot of the animation at time t.

The direct controller (non-evolved neural input field architecture) can by itself attain stability for the initial value $\theta = \pi/6$ but the performance is poor. While not shown, it has problems stabilizing the pendulum with an initial value of $\theta = \pi/3$ or higher.

The hand-tuned controller (non-evolved processing field controller) is able to control the stability without evolution with a better performance than the presented by the direct controller. It can stabilize in the 10s time those instances with an initial value of $\theta = \pi/3$ or higher. Nonetheless, it causes big displacements and tends to keep a small but persistent orientation error.

Finally, the evolved controller has the best performance of the three, performs a fast stabilization of the pendulum even with an initial value of $\theta = \pi$ (worst-case scenario for the initial orientation), which is shown in figure **3-9**. It is the only one that stays for long periods of time on the reference orientation and it causes the least displacements. Despite its parametrization being carried out by evolution, its strategy can be understood by looking at processing field activation values, and this controller seems to apply short burst of switching maximum values.

The three figures **3-7**, **3-8** and **3-9** show the input fields, the processing fields and the state variables evolution (respectively) for the three controllers.

In the figure **3-8**, of processing layer activations, the horizontal (x) axis represents the position of each element on the one-dimensional processing layer population, and the oblique axis (t) represents the time elapsed. The presence of the processing field in the feedback loop appears to act as an integrator-like control element. On the contrary, the sole utilization of the input field to spatially code the inputs appears to act as a proportional control element, causing the highly oscillating behavior in the input field architecture. Consequently, the nonlinear coupling between both actions, integral-like and proportional, seems to produce the behavior presented by the processing field architecture. Furthermore, the activation of the evolved neural field is quite chaotic, but shows a preferential activation on the extremal points, which could be interpreted as a preference for application of maximum control values (may be as an approximation of a bang-bang controller).

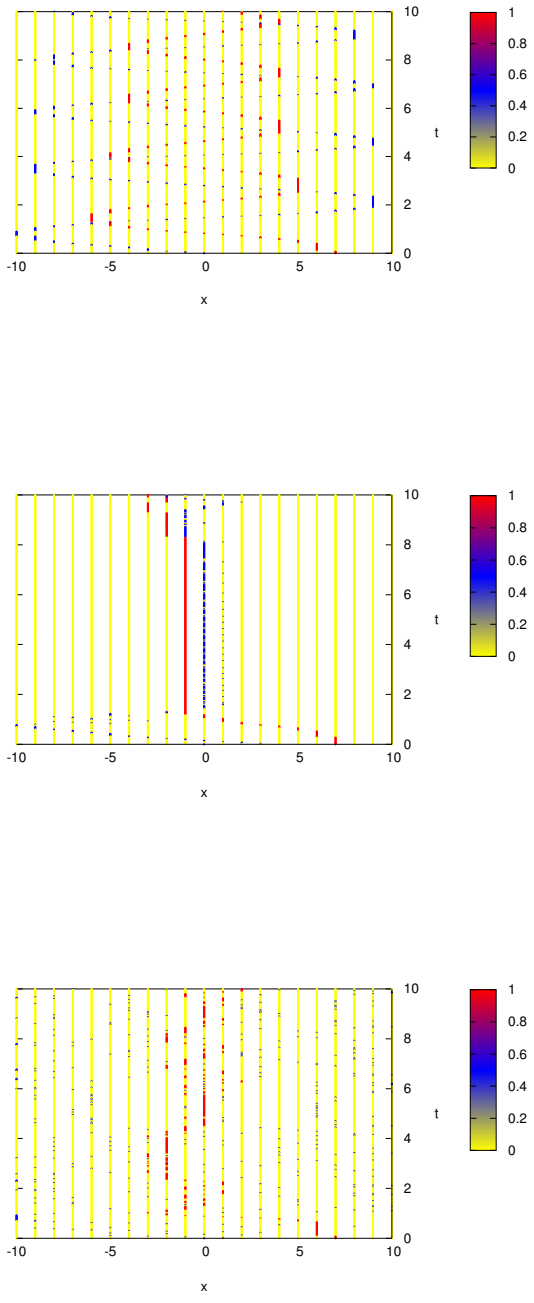


Figura 3-7: Input layer activation for simulations between $t = 0$s and $t = 10$s with steps $h = 1/40$s and positions between $x_{min} = -10$ and $x_{max} = 10$. First: direct (without processing layer) controller. Second: non-evolved controller. Third: evolved controller.

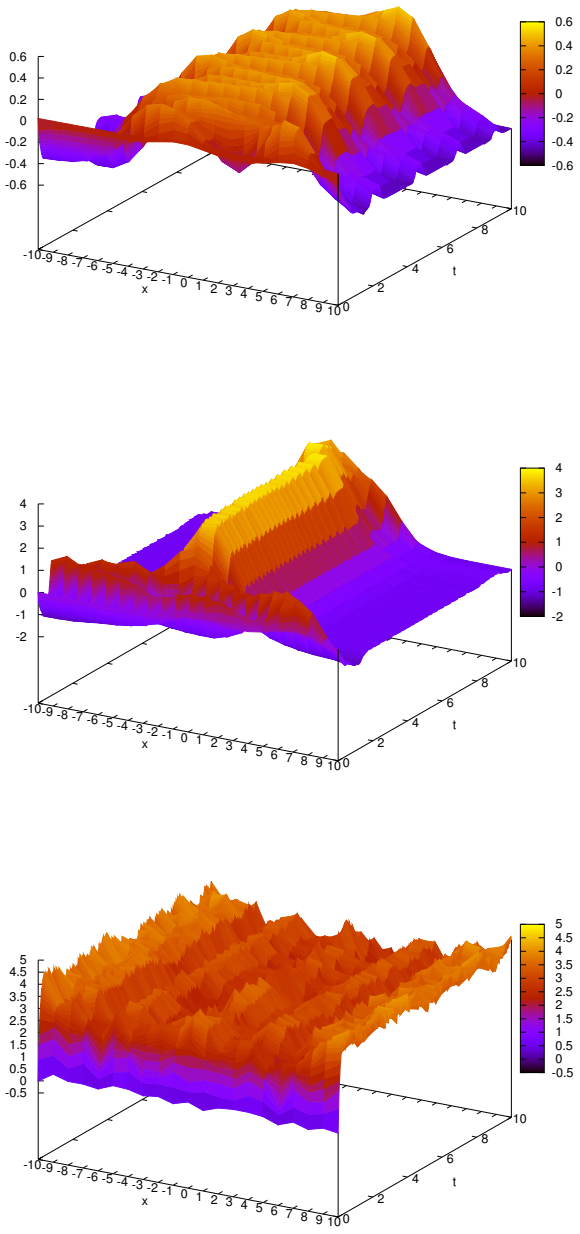


Figura 3-8: Processing layer activation for simulations between $t = 0$s and $t = 10$s with steps $h = 1/40$s and positions between $x_{min} = -10$ and $x_{max} = 10$. First: direct (output from input layer) controller, illustrative of field dynamics excluded of the control action. Second: non-evolved controller. Third: evolved controller.

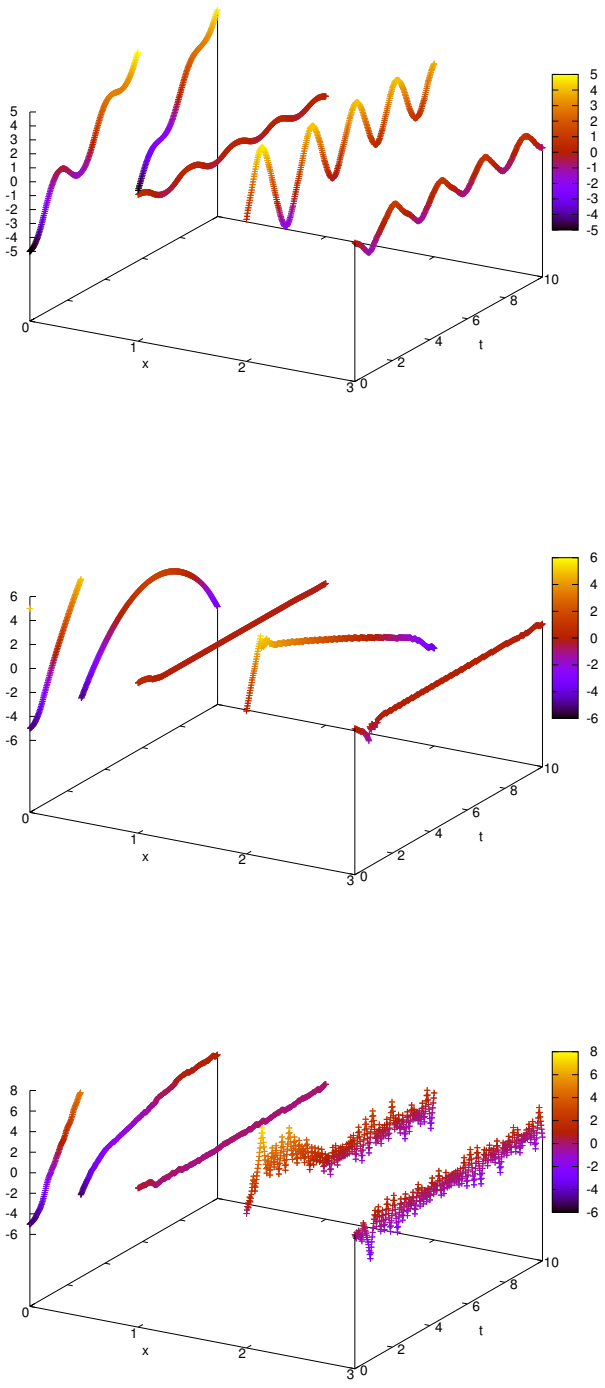


Figura 3-9: Pendulum simulation states with several neural field controllers between $t = 0$s and $t = 10$. Each state is coded with a position (so that the pendulum can be simulated as a field on its own) this way: x:0, θ:1, $\dot{x}$:2, $\dot{\theta}$:3. It can be seen the little variation in the angular position (θ). The discontinuity in x is present because position wraps in the simulation so that $x = 5$ is equivalent to $x = -5$. First: direct (without processing layer) controller. Second: non-evolved controller. Third: evolved controller.

3.8 Discussion

The results obtained from this chapter can be summarized in a short analysis.

While the recurrent neural network controller is expressive enough to solve the problem at hand, the number of parameters to configure (or in this case to evolve) is of a quadratic order in relation to the number of nodes (or neurons). This was not a particular problem for the evolutionary algorithm used, but limits its potential scalability. Furthermore, while it is expressive enough, it does not hold an internal representation that resembles in a meaningful way the problem of the inverted pendulum and there are no reasons to expect something different for a more complex biped model.

On the other hand, the neural field controller is a bit more complex and its simulation more costly, but has some notable advantages. The first one is its ability to self-compensate or, equivalently, the stability of its natural dynamics, which is attained after the setup of few parameters. The second one is it suitability to the problem at hand, being able to solve it with a acceptable degree of performance for low and mid perturbations without evolution.

Although there was a need for parameter configuration to attain a good performance, evolution was not strictly required because the small number of parameters to setup: basically three parameters of the kernel function and the resting potential of the field equation, a number of parameters of constant order in relation to the number of nodes (discrete potentials on the neural field). Nonetheless, the evolved controller performed better than the other two, is more general because eliminates the need for manual conscious parametrization and allows the designer to specify more precisely the performance measure desired.

It is worth noting that the neural field has a spatial representation which allows interpretation of field potentials (as shown in fig. **3-8**). Indeed, its geometrical representation allows for a small effort to go from an state-space like controller to a neural field controller. The interpretation or understanding of recurrent neural networks tends to be difficult and even more with increasing states involved. This makes the neural field controller not as black-box as the a recurrent neural network controller. This holds even for the evolved neural field controller.

More complex control problems could benefit from the strategy of parametrization of neural field controllers by evolution here developed. Therefore, it is left for future work the task of integration of multiple goals in different populations and the design of an arbitration scheme compatible with the architecture.

While the cart-and-pole biped walking model is useful as a first approach to the problem of locomotion, in order to develop a more realistic controller of biped walking, a model closer to human biped walking will be explored in the next

chapter, and also some new controllers will be presented.

4 An Optimal Controller for Biped Walking Using Neural Fields

4.1 Introduction

This chapter introduces the Simplest Biped Walking problem, which will be used onwards as the reference control problem. Approximations to optimal linear state-feedback controllers for the Simplest Biped Walking problem are found using numerical methods (genetic algorithms), for each sub-region on a delimited state-space region. A general architecture for the neural field controllers is presented, and an structured strategy to implement non-linear (sliding-mode-like) state-space controllers using the neural field-like controller architecture is shown. Finally, an specific controller for biped walking is developed and tested using the collected data, its behavior is shown using a Poincaré section for the controlled biped dynamics, and an stability analysis is provided.

4.2 The Simplest Biped Walking Model

The Simplest Walking Model, presented by Garcia et al. [24], is an attempt to create the simplest model that is capable of mimicking bipedal gait. It provides a reference model to study the phenomena that allows walking in two dimensions, and it has been further studied in terms of stability [58] [19], swing-leg control [67], energy consumption in active walking [39], walking on stairs [57], and walking with an upper body [66].

Here is used the model of Garcia et al., as modified by Wisse et al. to allow control torques.

The 2D Biped Walking (yet to be simplified) model assumes a biped with a hip connected to two feet through rigid legs with no knees. It has a point mass M located at the hip, and two point masses m located at the feet, with ratio $\beta = m/M$. This model has a swing phase, where there is a leg in contact with the ground (stance leg), and another one in pendular motion (swing leg). The collision of the swinging foot with the ground at heel-strike is plastic (no-slip,

no-bounce), and causes its velocity to jump to zero. Also, the double support is instantaneous, so only one leg is in contact with the ground at any time. For simplicity, it is assumed that the swing leg is allowed to pass through the floor surface an be below floor level once on each step, and only its second crossing will be detected as a collision (otherwise the foot-scuffing problems inevitable for a walker with straight legs would appear).

The equations of motion for the swing phase have the form $T = H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_g(q)$, with $q = [\theta\ \phi]^T$, where:

$$H = \begin{bmatrix} 1 + 2\beta(1-\cos\phi) & -\beta(1-\cos\phi) \\ \beta(1-\cos\phi) & -\beta \end{bmatrix}$$

$$C = \begin{bmatrix} 2\beta\dot{\phi}\sin\phi & -\beta\dot{\phi}\sin\phi \\ \beta\dot{\theta}\sin\phi & 0 \end{bmatrix}$$

$$\tau_g = \begin{bmatrix} (\beta g/l)[\sin(\theta-\phi-\gamma) - \sin(\theta-\gamma)] - (g/l)\sin(\theta-\gamma) \\ (\beta g/l)\sin(\theta-\phi-\gamma) \end{bmatrix}$$

$$T = \begin{bmatrix} T_\theta \\ T_\phi \end{bmatrix}$$

The Simplest Biped Walking model is achieved simplifying the previous model, by assuming that the point masses located at the feet are infinitesimal in comparison to the point mass at the hip (i.e. when $\beta \to 0$). In the first row β is set to 0, and in the second row we divide by β, effectively isolating the stance leg acceleration $\ddot{\theta}$ from the swing leg angle ϕ. Therefore:

$$T = H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_g(q) \tag{4-1a}$$

$$H = \begin{bmatrix} 1 & 0 \\ 1-\cos\phi & -1 \end{bmatrix} \tag{4-1b}$$

$$C = \begin{bmatrix} 0 & 0 \\ \dot{\theta}\sin\phi & 0 \end{bmatrix} \tag{4-1c}$$

$$\tau_g = \begin{bmatrix} -(g/l)\sin(\theta-\gamma) \\ (g/l)\sin(\theta-\phi-\gamma) \end{bmatrix} \tag{4-1d}$$

Furthermore, we now define the action torque vector as:

$$T = \begin{bmatrix} T_\theta \\ T'_\phi \end{bmatrix} \tag{4-1e}$$

Where we scale the torque on ϕ: $\beta T'_\phi = T_\phi$. Also, it is required a transition mapping for the state vector before and after collision. As shown by Garcia et al., there is a phase reduction in the collision, from four dimensions to two dimensions, which is seen as a rank reduction in the mapping equation:

$$\begin{bmatrix} \theta \\ \phi \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix}^+ = \begin{bmatrix} -1 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & \cos 2\theta & 0 & 0 \\ 0 & (1-\cos 2\theta)\cos 2\theta & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \phi \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix}^- \tag{4-2}$$

To complete the model, it is introduced the heel-strike event condition (also from Garcia et. al), that provides a geometric condition in terms of the state vector, that has to be satisfied for the swing foot to cross the ramp surface:

$$\phi^+ - 2\theta^- = 0 \tag{4-3}$$

Given that there is a model dimension reduction on each heel-strike, each step is uniquely determined by the initial state after heel-strike $v^+ = \left[\begin{smallmatrix} \theta^+ \\ \dot{\theta}^+ \end{smallmatrix}\right]$. The stability of a given controller can be analyzed through the evolution of the Poincaré section v_n^+ after the n-th heel-strike. This is analogous to analyzing the stability of the 'stride function' $v_{n+1} = S(v_n)$ in terms of McGeer [43]. In general, stability in this chapter is understood in the sense of Lyapunov, where the stride function is taken as a discrete system. As there are some v values for which the stride function $S(v)$ has no output, it is assumed that stability is not attained when, for a given v_0, there is a n for which $v_n = S^n(v_0)$ is not defined.

4.3 Searching for Optimal Linear State-Feedback Controllers

As shown by Schwab et al. [58], while there is an attractor for the Simplest Biped Walking model on small slopes, its basin of attraction covers a minute portion of the initial configurations on phase space. Wisse et al. propose a controller for the swing leg that substantially widens the basin of attraction [67], that may be equally implemented as a spring-damper mechanical configuration (aided by some switching mechanism when the swing leg becomes the stance leg), or as a state-feedback control. This controller is based on the intuition that a rimless wheel attains asymptotically stable steady 2D motions, given the proper ground slope and angle between legs [15]. Also, it has the advantage that, for sufficiently

small mass ratio β, the energy cost of the required control action is negligible. Such control strategy can be stated as:

$$T'_\phi = k_\phi(\phi_r - \phi) + \sqrt{k_\phi}\dot{\phi} \tag{4-4}$$

Where the reference swing leg angle is set to a convenient value $\phi_r = 0.3$.

The aforementioned control strategy provides a wider basin of attraction as k_ϕ grows. Therefore, for each initial configuration, there is a minimum k_ϕ that attains stable walking [67].

As a first step to extending the static control policy of Wisse et al., in this section there will be found a set of minimal k_{phi} values that attain stability for each initial configuration, with a given discretrization of the configuration space.

For the purpose of comparison, the system with $\gamma = 0.004$, and $M = g = l = 1$ will be explored. The initial configuration space will have the bounds $\theta_0 = [0, 0.4]$ and $\dot{\theta}_0 = [-0.4, 0]$, with step size $\delta v = 0.025$, for a 17x17 nodes grid.

For each node in the grid, a search with an evolutionary algorithm is performed, using k_ϕ as the genotype. The actual evolutionary algorithm implementation used is HAEA (see [26]), as implemented in the library JML by Gomez. The fitness function evaluation implies running a simulation of the model with the state-feedback controller using the k_ϕ given, for a fixed time interval. The simulation may abort before the fixed time, if state variable (not the Poincaré section) $\theta(t)$ leaves the interval $[-\pi/2, \pi/2]$, and the lowest fitness value is assigned. Otherwise, the fitness value is k_ϕ^2.

The resulting mapping $k_\phi = M(\theta_0, \dot{\theta}_0)$ is shown in the figure **4-1** for $\gamma = 0.004$. As can be seen, the mapping is monotonically increasing both as θ_0 increases and as $\dot{\theta}_0$ decreases.

A displacement on the values found $k_\phi = M(v_n + \left[\begin{smallmatrix} 0.5h \\ -0.5h \end{smallmatrix}\right])$ (where h is the mapping grid step) in the mapping is applied, with the purpose to guarantee stability in of the controller, when using interpolation for the k_ϕ values. This can be done, given the monotonically increasing nature of the mapping.

Before applying this mapping to an extended control policy for the Simplest Biped Walking model implemented with a neural fields-like structure, an architecture for neural field controllers is required.

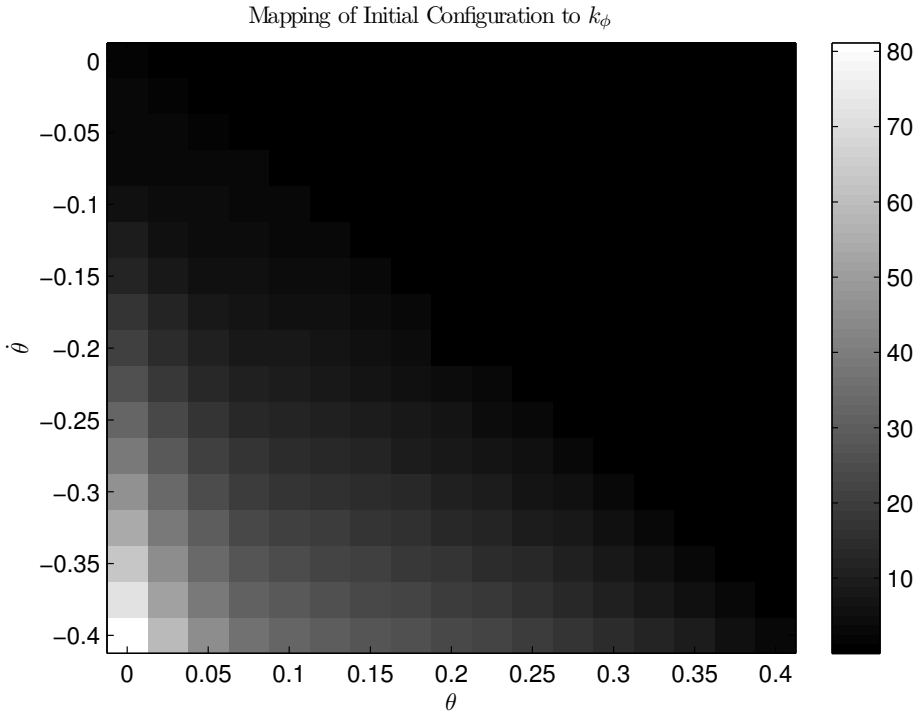


Figura 4-1: Search result for the mapping $k_\phi = M(\theta_0, \dot{\theta}_0)$.

4.4 A Neural Field Controller Architecture

In previous chapters it has been shown how a neural field can be used as a controller for an unstable system. Neural fields have performed good enough (compared to the RNN approach) when applied to a simple control problem: a cart-and-pole (inverted pendulum) system. The particular configuration tested was composed by:

- Two input variables: angular position and angular speed.
- One control goal: to minimize the angular position.
- One control action: lateral force at the pendulum base.

The corresponding controller architecture was devised as a two layer neural field. Each layer of the field is actually a neural population modeled after a subset of the real line: $x|x \in (-1, 1)$. Those layers have the following roles:

- A first layer, without inner dynamics (i.e. not governed by a differential equation) which merely maps the input variables to activation potentials on a field, using the vector coding technique.

- A second layer, with prototypical dynamics of neural fields. The kernel functions (for connections between layers 1 and 2, and for recurrent connections on layer 2) were used as the tuning/evolution parameters from which the control behavior emerged. The actual output was obtained by finding a weighted average of the activation potentials and mapping the resulting centroid to a single value (an inverse process to vector coding).

To generalize this approach, here is introduced a simple architecture of neural field controllers to help the understanding of commonalities and variations of the several controller structures developed next. It is not meant as a global framework for control with neural fields, as other architectures may be fruitfully developed. It is instead, an specific model for the description of controllers based on layered neural fields, able to describe the controller developed in this chapter. The controller architecture that will be used here onwards is described with an emphasis in how information is processed in the controller.

The neural field controller architecture will be generally structured as follows. It will have three layers:

Input: Vector-coded representation of relevant input values (e.g. a subset or linear combination of controlled system state-variables).

Representation: Inner state representation of the controller architecture. Could be interpreted as the probability of being on each state $p(s = s_x)$ from the viewpoint of the controller.

Action: Representation of the preferred actions of the controller. May be interpreted as vector-coded (when using a complete neuron population), or directly coded (when using a single node for each controlled variable).

As a strongly layered architecture, the connections are only allowed from contiguous layers, in one direction. Thus, elements in one layer cannot receive inputs from other elements except those in the immediately preceding layer (with *action layer* receiving from *representation layer*, and *representation layer* receiving from *input layer*).

Each of these layers may or may not have a dynamic behavior, and furthermore, if it does, may have the form of a neural field, in terms of structure and behavior (i.e. governed by the dynamic equations of neural fields).

For those layers that do have a dynamic behavior (as is the case of the representation layer in this chapter, as will be shown next), the derivative of the layer potential at each position $\dot{u}_i(x)$ is fed with the output of function from the

previous layer activation potential $f_{i,i-1}(u_{i-1})$ (and may have also a recurrent component, function of u_i).

Those layers that do not have a dynamic behavior, have their input directly defined as a function of the previous layer activation potential $u_i = f_{i,i-1}(u_{i-1})$. They also can be thought of as a reference-tracking dynamical system with zero reference position error, which is fast enough for its transient behavior to be ignored. The input layer may be a particular case of this, where there is no previous layer, and its activation potential is a function of the controlled system state-variables $u_1 = f_1(X)$.

4.5 An Sliding-mode Neural Field Controller

4.5.1 Controller Structure

The layers, depicted in figure **4-2** are further described below:

Input Layer This layer is used as a spacial representation of a function of the inputs, without a dynamic behavior (it does not add state-variables to the bundled system-controller differential equation). The values stored in this layer act only as a buffer to feed the inputs to the representation layer, and correspond in each iteration to a mapping of the system state-variables. It should be noted that this layer is not governed by a differential equation and therefore does not follow Amari's definition of a neural field.

While the mapping function can be general, here it will be simplified. As a result, the input layer is a direct, vector-coded, representation of the relevant state of the controlled system after heel-strike. More specifically, for the input layer population ($i = 1$):

$$u_1(x,t) = f_I(x, v_n^+) \tag{4-5a}$$

$$f_1(x, v_n^+) = \begin{cases} \text{const.} & \text{if v.c}\,(v_n^+) = x. \\ 0 & \text{otherwise.} \end{cases} \tag{4-5b}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in [0,1] \times [0,1] \tag{4-5c}$$

$$v_n^+ = \begin{bmatrix} \theta_n^+ \\ \dot{\theta_n}^+ \end{bmatrix} \in [0, 0.4] \times [-0.4, 0] \tag{4-5d}$$

Note that the activation potential of the input layer $u_1(x,t)$ is a function of the reduced system state-vector v_n^+ for the n-th heel-strike (the last occurred at time t), and is constant up to the next heel-strike.

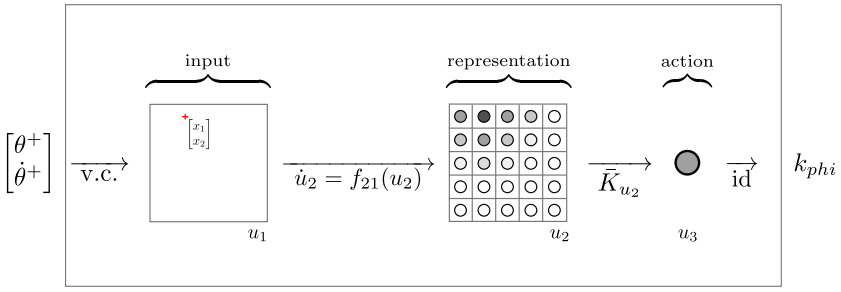


Figura 4-2: Neural field controller architecture for Chapter 3.

Representation Layer The neural population in this layer behaves according to a differential equation where the input if functions of the activation potential of the input layer. Therefore, the population in this layer resembles the definition given by Amari for neural fields [1], but with an specific structure for the interconnection between input layer and representation layer populations, and also, a minor role of recurrence.

The differential equation for the potential at position $x \in \Omega_2 = [0,1] \times [0,1]$ in the representation layer ($i = 2$) population is:

$$\tau_2 \dot{u}_2(x,t) = -u_2(x,t) + \int_{x' \in \Omega_2} w_{2,2}\left(s_{2,2}(x,x')\right) \psi\left(u_2(x',t)\right) dx' + f_{2,1}(x,t) \quad \text{(4-6a)}$$

$$f_{2,1}(x,t) = \int_{x' \in \Omega_1} w_{2,1}\left(s_{2,1}(x,x')\right) \psi\left(u_1(x',t)\right) dx' \quad \text{(4-6b)}$$

Where as in previous chapters τ_i is a time constant for the population i, u_i is the activation potential of elements in the population, $w_{j,i}$ is the connection kernel for connections from population i towards population j, which is a function of $s_{j,i}$, the distance between two positions (on the same population j, or in different populations i and j), $Omega_i$ is the set of positions x in the layer i, and ψ is the activation function (here a Heaviside function).

The connection kernel $w_{j,i}$ could take any form, but the form of a Mexican-hat function will be used.

Action Layer Similarly as it was done between the input layer, the action layer is non-differential in nature (so it also does not add state variables to the bundled system-controller differential equation). Also, it contains only one node, and its activation corresponds to the controller output, k_ϕ, the proportional controller constant defined in section 4.2. The mapping between the representation layer and the action layer ($i = 3$) is evaluated using a weighting matrix, calculated as:

$$u_3(t) = \frac{\sum_{x'} W_{3,2}(x') u_2(x', t)}{\sum_{x'} u_2(x', t)} \tag{4-7}$$

Where $W_{3,2}(x')$ is the estimated control action if the current state where located in the cell with center at position x'. The actual values for $W_{3,2}$ are assigned to the resulting $M(\theta_0, \dot{\theta}_0)$ of the search performed in section 4.3, which was then assigned directly to k_ϕ.

4.5.2 Insights on the Controller Behavior

The actual controller that is presented in this chapter may be understood as a biologically implementation of a Sliding-mode controller, and works (in general terms) selecting a control policy for each step, using as input the Poincaré section $v_n^+ = \left[\begin{smallmatrix}\theta_n^+ \\ \dot{\theta}_n^+\end{smallmatrix}\right]$. The mapping $k_\phi = M([\theta_n^+ \dot{\theta}_n^+]^T$ is embedded in the structure of the neural field. The control policy provided as output by the neural field is the parameter k_ϕ, used as constant for the locally linear critically-damped PD controller. An interpretation of the neural field controller structure is detailed next.

The input layer has a two-dimensional (non-dynamic) neural population, with one dimension for θ_n^+ and another for $\dot{\theta_n^+}$, the elements in the Poincaré section v_n^+ (just after heel-strike), but rescaled to be in the set $[0, 1] \times [0, 1]$. For these populations, the function f in the equation $u_1(x, t) = f(v_n^+)$ (shown in the previous subsection) performs a vector coding, i.e. an specific value of the input variable v_n^+ causes an activation $u_1(x)$ with amplitude 1 at some position x, and an activation of 0 at any other position. For implementation purposes, note that this value does not need discretization, as the input layer is not explicitly modeled, given that it does not add state variables to the bundled system-controller differential equation.

The representation layer also a neural population, where the position x resides in the same two-dimensional space of the input layer population. An activation of an element of the input layer population (e.g the one corresponding to v_n^+) causes an activation bump that reaches its maximum at some position in the representation layer, and decreases as the nodes are farther away. This configuration gives the representation population an structure that resembles the Poincaré section space. The implementation has 17x17 elements, an equal number than that of the mapping found in the section 4.3, but it should be noted that the actual points are different, given that in the position grid of the representation and input layers each node is located at the center of each cell, but in the section 4.3 were located at the cell boundaries.

The output layer has also one population, actually with only one element, which activation provides the output value k_ϕ. Each element in the representation layer population contributes with a value given by the product of its activation potential and the optimal k_ϕ value for that position in the approximated Poincaré section in the representation layer. The total output is calculated as the average of $W_{3,2}(x')$ wighted by the activation of the representation layer $u_2(x')$. If $W_{3,2}(x')$ is an approximation to the optimal control action for state x', and the fraction $u_2(x',t)/\sum_{x'} u_2(x',t)$ is an approximation to the probability of being in the state x' at the current time t, then this average provides an approximated expected value for the optimal control action $\mathrm{E}_{u_2}\{k_\phi\}$.

4.6 Results and Discussion

The experimental results are shown in the following figures. For comparison purposes, the State-feedback (SF) control strategy proposed by Wisse et al. is implemented, and compared to the Optimized Sliding-mode (OSM) controller developed using search on section 4.3, and to the Sliding-mode(-like) Neural Field (SMNF) controller proposed in the previous section. The experiment shown is run for $t = [0, 100]$ with the initial configuration $v_n = \left[\begin{smallmatrix} 0.02 \\ -0.39 \end{smallmatrix}\right]$, which is near the most unstable point in the region studied.

While the SF controller keeps constant the $k_\phi = 100$ value in order to attain stability, both the OSM and SMNF controller proposed recalculate the k_ϕ (at least) on each step using the current Poincaré section and the embedded mapping $k_\phi = M(v_n^+)$, directly for the OSM, and indirectly for the SMNF.

The output values of the OSM take the form of a Sliding-mode controller in the following sense:

- A subset of Poincaré section at heel-strike defines the section of state-space where the system "slides" along.
- At each heel-strike the parameter k_ϕ is re-evaluated, effectively changing the structure of the controller in a discontinuous fashion.
- The matrix of control values $M(v_n^+)$ is built so that the system always stays in the subset of the Poincaré section, and furthermore, approaches the sliding surface in finite time.
- Once reaches the sliding surface (fixed point in the Poincaré section), the system stays there.

On the other hand, the output function of the SMNF uses a weighted average (analogous to a centroid policy). It interpolates between the values on the 17x17 mapping grid, but also has a dynamic estimation of the system state, implemented in the representation layer. The dynamics of the representation layer could cause the loss of stability, as it may lag or otherwise underestimate the required k_ϕ. Nonetheless, as is shown in the figures in this section, the SMNF (with the kernel function parameters used) approaches quite closely the behavior in the long term of the OSM controller, but with a soft and continuous change in the control parameter k_ϕ.

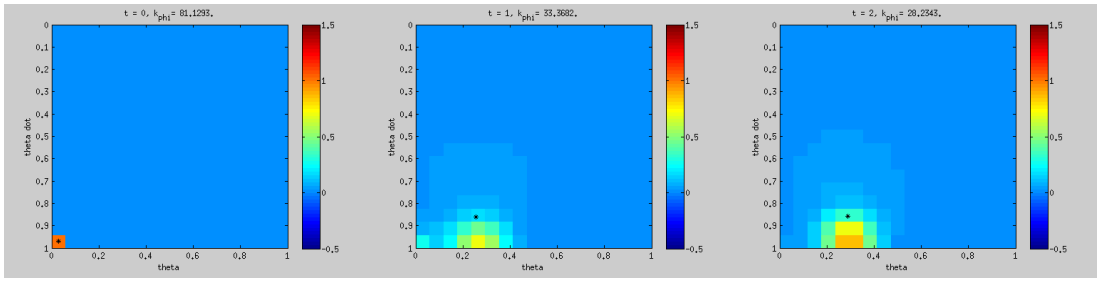
Figure **4-3** shows a sequence of captures, at increasingly longer intervals, of the activation potential of the representation layer neural field population. In it can be observed that the dynamics over the neural field converge as the system approaches the attractor in the Poincaré section.

The overall performance of the SMNF controller is better than the SF in terms of energy consumption and actuator strain, progressively diminishing its k_ϕ value and thus the cumulative control action (see the figure **4-4** where the k_ϕ and $\int \tau_\phi^2 \, dt$ values vs time are plotted). The SF controller converges faster to its fixed point (as would be expected for a greater control action), but even in steady-state the control action stays almost unaltered. On the other hand, the OSM controller has an slightly higher cumulative control action than the SMNF controller, but it provides a monotonic decrease in k_ϕ across time, behaving more properly as an Sliding-mode controller.

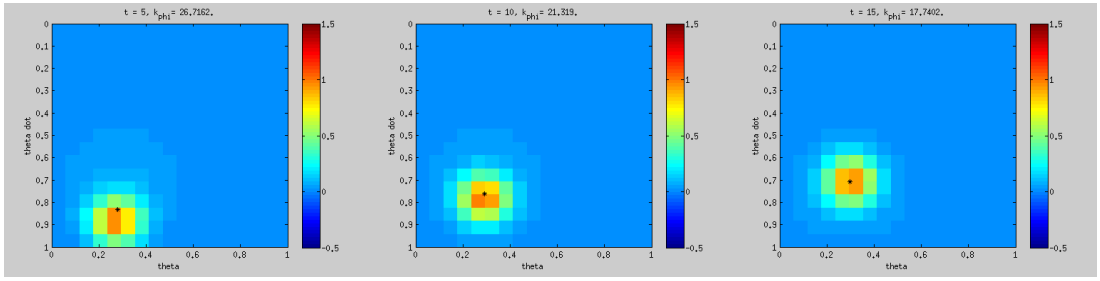
The Poincaré Sections (see figure **4-5**) of the biped using both the SMNF controller and the OSM controller show how the system moves from a configuration that requires higher k_ϕ values to configurations that require lower ones, and that fact is exploited by the controllers. The SF controller also moves to configuration that require lower k_ϕ values (even closer to the natural attraction basin of the biped), but that fact remains unused. This is the main factor of improvement in the controllers proposed in this chapter compared to the solution of Wisse et al.

The time simulation (see figure **4-6**) shows how the SMNF changes the control policy (as parametrized by k_ϕ) softly enough to provide a qualitatively natural gait for the biped. It should be noted that sudden changes of behavior are common in Sliding-mode controllers, but that is mitigated in this case by three facts: 1) The neural field applies an interpolation using the centroid of activation to calculate the output k_ϕ. 2) The neural field has natural dynamics qualitatively equal to a low-band filter. 3) The change in the k_ϕ occur at a non-linear point in the biped dynamics that anyway would cause a jump in its state.

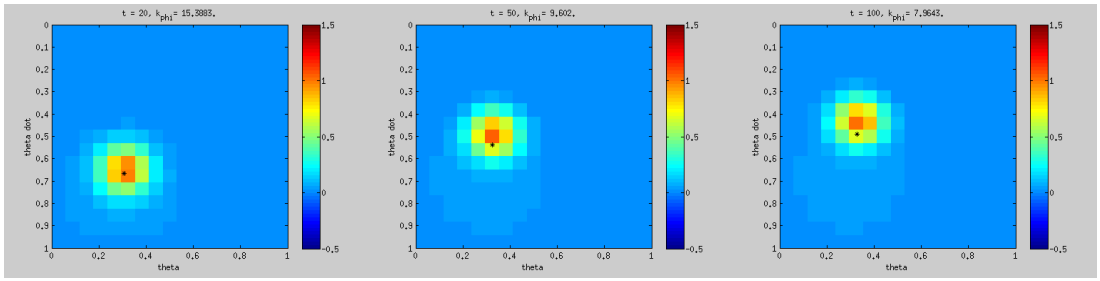
In conclusion, in this chapter we presented two controllers which improve the controller of Wisse et al. over the Simplest Biped Walking (SBW) model, using



(a) $t = 0$ (s) (b) $t = 1$ (s) (c) $t = 2$ (s)

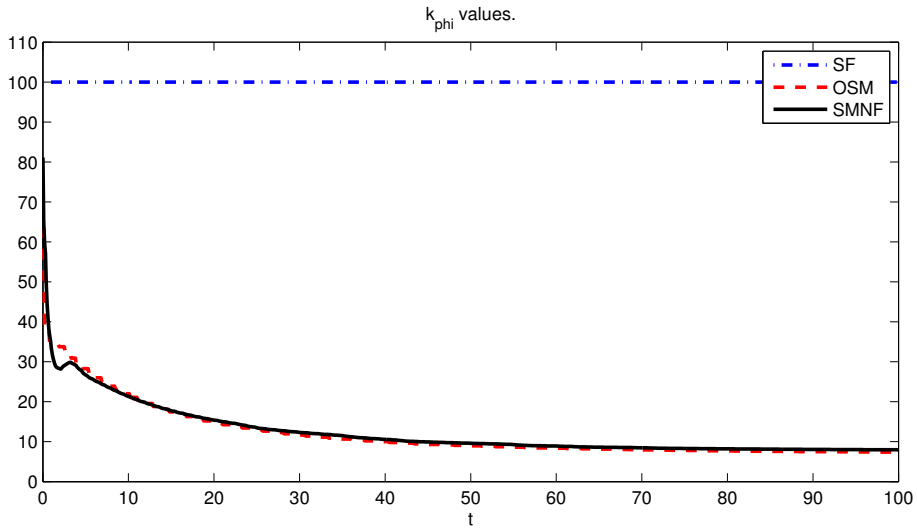


(d) $t = 5$ (s) (e) $t = 10$ (s) (f) $t = 15$ (s)

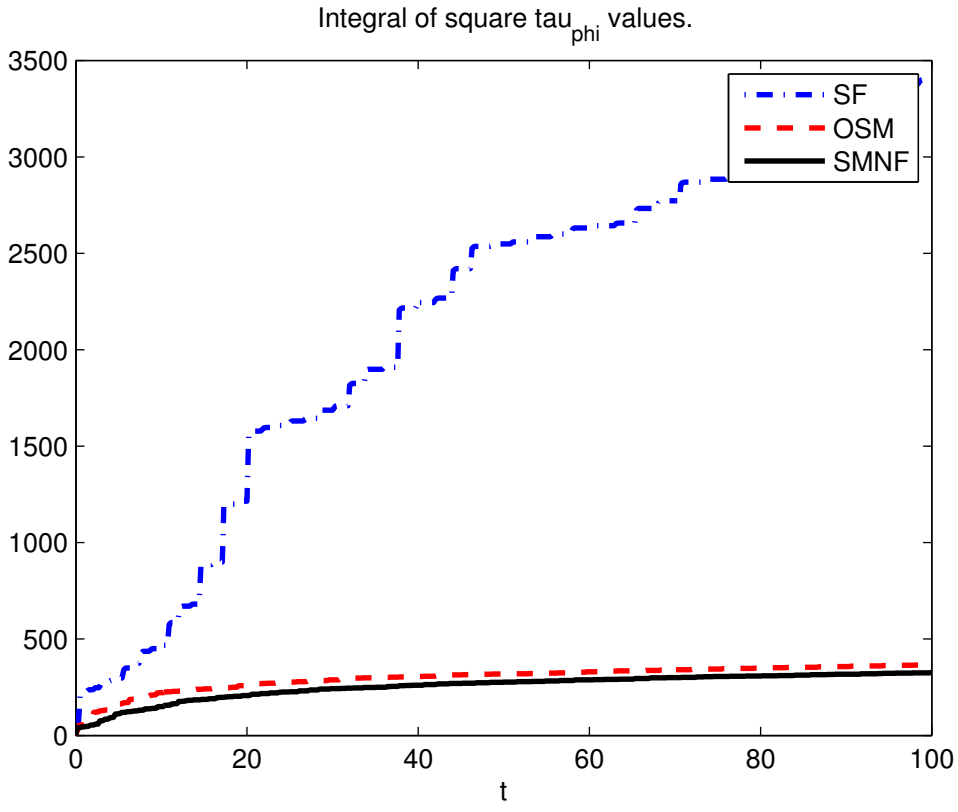


(g) $t = 20$ (s) (h) $t = 50$ (s) (i) $t = 100$ (s)

Figura 4-3: Activation potentials for the representation layer vs. time, for $t \in 0, 1, 2, 5, 10, 15, 20, 50, 100$ (s), for the SMNF controller. Centroid marked with a black asterisk (*).



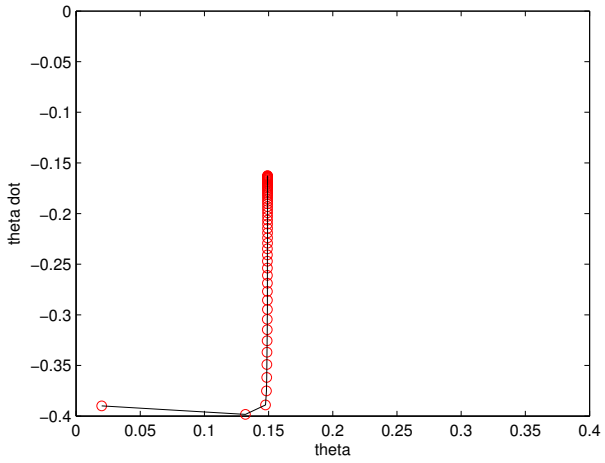
(a) k_ϕ parameter values for the controllers.



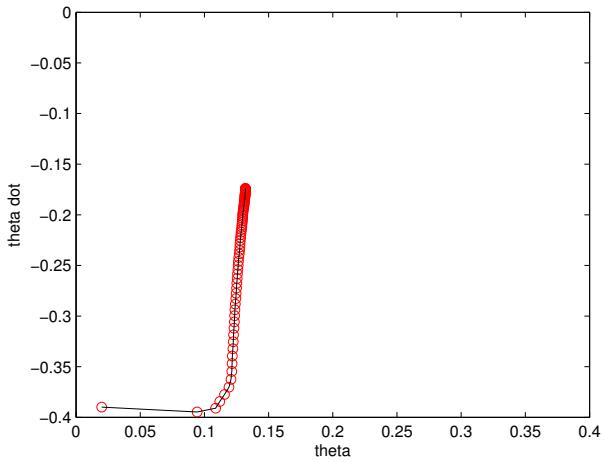
(b) Integral of squared control action ($\int \tau_\phi^2 \, dt$) for the controllers.

Figura 4-4: Controller action vs. time, for $t \in [0, 100]$ (s), for the three controllers presented in this chapter: Constant k_ϕ for SF controller, step-wise adaptation of k_ϕ for OSM controller, and continuous variation of k_ϕ for SMNF controller.

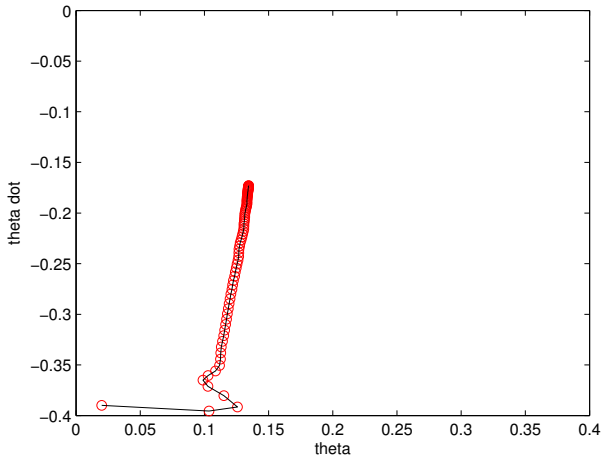
a mapping from regions of the Poincaré Section just after heel-strike $v_n = \left[\begin{smallmatrix} \theta_n^+ \\ \dot{\theta}_n^+ \end{smallmatrix}\right]$ to k_ϕ values, in a manner similar to Sliding-mode controllers. Those controllers, while being active, approach more closely Passive Dynamic Walking (PDW) by diminishing the cumulative control action. One of the two controllers proposed is implemented using a control architecture based on neural fields, which extends and formalizes the structure of the controller for the inverted pendulum shown in the previous chapter.



(a) SF Controller

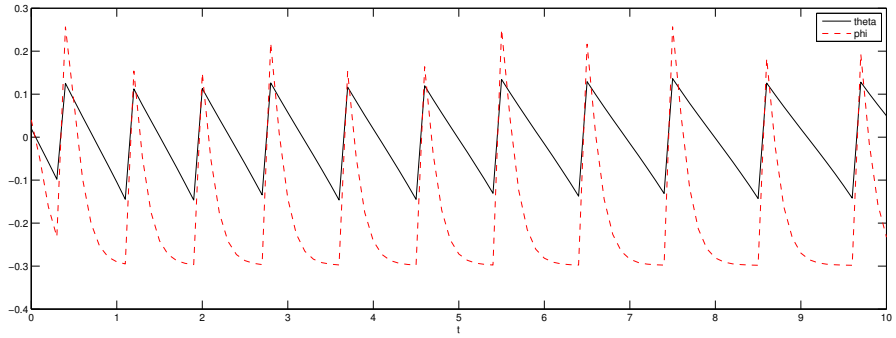


(b) OSM Controller

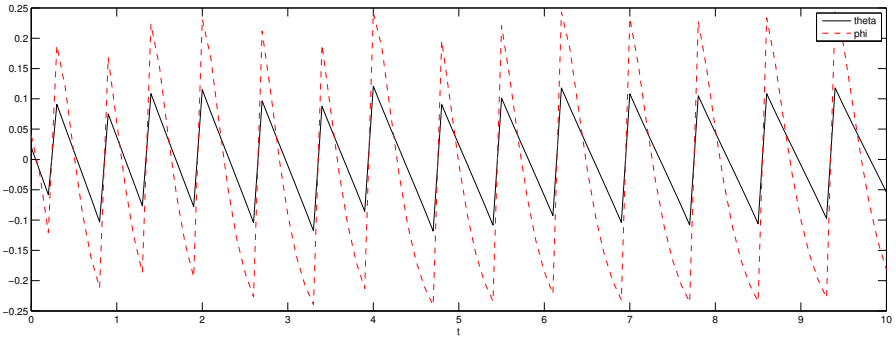


(c) SMNF Controller

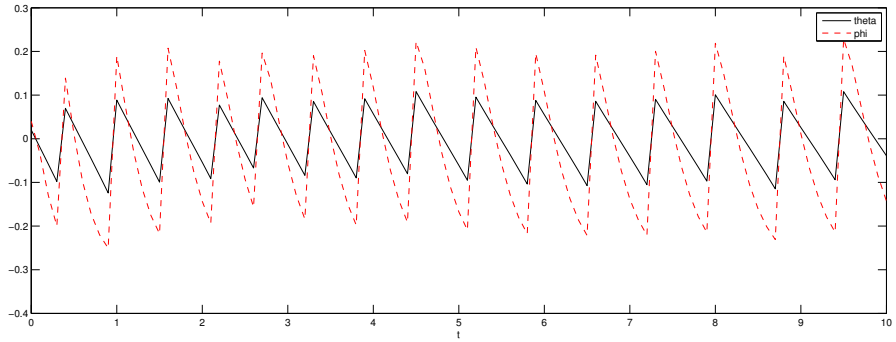
Figura 4-5: Poincaré Section for the reduced-dimensionality system after heel-strike, for $t \in [0, 100]$ (s), for each controller evaluated.



(a) SF Controller



(b) OSM Controller



(c) SMNF Controller

Figura 4-6: System evolution (state variables after heel-strike vs. time) for $t \in [0, 10]$ (s), for each controller evaluated.

5 Conclusions

5.1 Main Contributions

The goal-oriented control architecture using neural fields proposed here, is the main contribution of this work. It shows that neural fields may be used as a useful component to build motion controllers, taking into account that they are also biologically plausible.

An additional but important contribution, related to the goal of minimization of global energy consumption, is the optimization by means of evolutionary algorithms and neural fields of the previous work on pseudo-Passive Dynamic Walking controllers (see [67]) for the Simplest Biped Walking model (as proposed in [24]). With this contribution, this work advances towards what could be the most valuable goal on biped walking control.

5.1.1 Neural Field Controller for the Inverted Pendulum

In chapter 3, which was published in two parts in the Proceedings of IJCNN 2009 and GECCO 2009 ([22] and [23]), it was developed a neural field controller that is able to solve the stability problem for the inverted pendulum (cart-and-pole). Both a neural field controller manually tuned, and a neural field controller parametrized using evolutionary algorithms were presented. Also, two additional controllers were presented for comparison: a controller using evolved recurrent neural networks, and the non-evolved neural field controller omitting the processing layer for output.

It could be seen that, while the recurrent neural network controller is expressive enough to solve the problem at hand, the number of parameters to configure is of a quadratic order in relation to the number of neurons.

On the other hand, the neural field controller was found to be more complex and its simulation more costly, but had some notable advantages: The stability of its natural dynamics, and its suitability to the problem at hand, being able to solve it with a acceptable degree of performance for low and mid perturbations, even without evolution.

The evolved controller performed better than the other two, is more general

because eliminates the need for manual conscious parametrization and allows the designer to specify more precisely the performance measure desired. Also it allows interpretation of field potentials. and its geometrical representation allows for a small switch between a state-space like controller to a neural field controller. The interpretation or understanding of recurrent neural networks tends to be difficult and even more with increasing states involved. This makes the neural field controller not as black-box as the a recurrent neural network controller.

5.1.2 Neural Field Controller for SBW

While the cart-and-pole biped walking model is useful as a first approach to the problem of locomotion, in order to develop a more realistic controller of biped walking, a model closer to human biped walking was explored in chapter 4, and also some new controllers were presented.

An extended neural field control and planning architecture was developed and applied to the stability problem for the Simplest Biped Walking model. Also it was shown how the control architecture may be used at a planning level by changing the control policy to be applied, and also how it is able to minimize the global energy consumption. The neural field control architecture was compared to the linear controller proposed by Wisse et al. [67], and to the optimized but not biologically inspired Sliding-mode controller proposed in this work.

Those controllers proposed in chapter 4, while being active, approach more closely Passive Dynamic Walking (PDW) than previous works, by diminishing the cumulative control action. One of the two controllers proposed was implemented using a control architecture based on neural fields, which extends and formalizes the structure of the controller for the inverted pendulum shown in the previous chapter. Also, a parametrization using evolutionary algorithms (HAEA by Gomez, see [26]) was performed on the mappings between the representation layer and the action layer of the neural field controller proposed (also used directly in the other controller presented).

It is important to note that the control strategies presented in chapter 4, would also work, mostly unaltered, for any other system were there is a controller for which exists a parameter so that the attraction basin widens as that parameter is increased.

While the State feedback (SF) controller of Wisse et al. keeps constant the $k_\phi = 100$ value in order to attain stability, both the optimized Sliding-mode controller (OSM) and Sliding-mode-like neural field (SMNF) controller proposed were shown to recalculate the k_ϕ (at least) on each step using the current Poincaré section and the embedded mapping $k_\phi = M(v_n^+)$, directly for the OSM, and

indirectly for the SMNF.

It was shown that the output values of the OSM take the form of a Sliding-mode controller in the following sense:

- A subset of Poincaré section at heel-strike defines the section of state-space where the system "slides" along.
- At each heel-strike the parameter k_ϕ is re-evaluated, effectively changing the structure of the controller in a discontinuous fashion.
- The matrix of control values $M(v_n^+)$ is built so that the system always stays in the subset of the Poincaré section, and furthermore, approaches the sliding surface in finite time.
- Once reaches the sliding surface (fixed point in the Poincaré section), the system stays there.

Also, it was shown that given the output function the SMNF uses, and the dynamic estimation of the system state implemented in the representation layer, could cause the loss of stability, as it may lag or otherwise underestimate the required k_ϕ. Nonetheless, the SMNF approaches quite closely to the behavior in the long term of the OSM controller, but with a soft and continuous change in the control parameter k_ϕ.

The overall performance of the SMNF controller was better than the SF in terms of energy consumption and actuator strain, progressively diminishing its k_ϕ value and thus the cumulative control action. The SF controller converges faster to its fixed point, but even in steady-state the control action stays almost unaltered. On the other hand, the OSM controller has an slightly higher cumulative control action than the SMNF controller, but it provides a monotonic decrease in k_ϕ across time, behaving more properly as an Sliding-mode controller.

The Poincaré Sections of the biped using both the SMNF controller and the OSM controller did show how the system moves from a configuration that requires higher k_ϕ values to configurations that require lower ones, and that fact is exploited by the controllers.

The time simulation did show how the SMNF changes the control policy (as parametrized by k_ϕ) softly enough to provide a qualitatively natural gait for the biped. It should be noted that sudden changes of behavior are common in Sliding-mode controllers, but that is mitigated in this case by three facts: 1) The neural field applies an interpolation using the centroid of activation to calculate the output k_ϕ. 2) The neural field has natural dynamics qualitatively equal to a low-band filter. 3) The change in the k_ϕ occur at a non-linear point in the biped dynamics that anyway would cause a jump in its state.

5.2 Future Work

More complex control problems could benefit from the strategy of parametrization of neural field controllers by evolution here developed. This way, it is left for future work the task of integration of multiple goals in different populations and the design of an arbitration scheme compatible with the architecture.

Also, the neural field controller presented in chapter 4 may be extended using reinforcement learning, specially a form of Q-learning, to efficiently identify the optimal action-values (mapping between reinforcement and action layers). It could be done using biologically plausible techniques, as shown by Strösslin et al. in [61].

Finally, further application of the extended neural field controller architecture, used at a policy modification level, could be explored to further asses its performance in a wider variety of problems.

Bibliography

[1] Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, **Volume 27, Number 2**, 77–87.

[2] Anderson, C. (1989). Learning to control an inverted pendulum using neural networks. *Control Systems Magazine, IEEE*, **9**(3), 31–37.

[3] Asami, S. (1994). Robots in japan: present and future. *Robotics & Automation Magazine*, **1**(2), 22–222–26.

[4] Azevedo, C., Poignet, P., and Espiau, B. (2004). Artificial locomotion control: From human to robots. *Robotics and Autonomous Systems*, **47**(4), 203–223.

[5] Bardossy, A. and Duckstein, L. (1995). *Fuzzy Rule-Based Modeling with Applications to Geophysical, Biological, and Engineering Systems*. CRC Press, Inc., Boca Raton, FL, USA.

[6] Basdogan, C. and Amirouche, F. M. (1996). Nonlinear dynamics of human locomotion: from the perspective of dynamical systems theory. In *American Society of Mechanical Engineers, Petroleum Division (Publication) PD*, volume 77, pages 163–168. ASME.

[7] Bedau, M. (1992). Philosophical aspects of artificial life.

[8] Benbrahim, H. and Franklin, J. (1997). Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, **22**(3-4), 283–302.

[9] Bergener, T., Bruckhoff, C., Dahm, P., Janssen, H., Joublin, F., Menzner, R., Steinhage, A., and Von Seelen, W. (1999). Complex behavior by means of dynamical systems for an anthropomorphic robot. *Neural Networks*, **12**(7-8), 1087–1099.

[10] Cao, M. and Kawamura, A. (1998). A design scheme of neural oscillatory networks by hierarchical evolutionary calculation for generation of humanoid biped walking patterns. *Advanced Robotics*, **12**(7-8), 697–710.

[11] Capek, K. (1973). *R.U.R.* Pocket Books, New York.

[12] Capi, G., Kaneko, S., Mitobe, K., Barolli, L., and Nasu, Y. (2002). Optimal trajectory generation for a prismatic joint biped robot using genetic algorithms. *Rob Autom Syst*, **38**(2), 119–128.

[13] Chang, W.-D., Hwang, R.-C., and Hsie, J.-G. (2002). A self-tuning pid control for a class of nonlinear systems based on the lyapunov approach. *Journal of Process Control*, **12**(2), 233–242.

[14] Cheng, M.-Y. and Lin, C.-S. (2000). Dynamic biped robot locomotion on less structured surfaces. *Robotica*, **18**(2), 163–170.

[15] Coleman, M. J., Chatterjee, A., and Ruina, A. (1997). Motions of a rimless spoked wheel: a simple three-dimensional system with impacts. *Dynamics and Stability of Systems*, **12**(3), 139–159.

[16] Coombes, S. (2005). Waves, bumps, and patterns in neural field theories. *Biological Cybernetics*, **Volume 93, Number 2**, 91–108.

[17] Craig, J. (1989). *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA.

[18] Dario, P., Guglielmelli, E., and Laschi, C. (2001). Humanoids and personal robots: Design and experiments. *Journal of Robotic Systems*, **18**(12), 673–690.

[19] Das, S. L. and Chatterjee, A. (2002). An alternative stability analysis technique for the simplest walker. *Nonlinear Dynamics*, **28**, 273–284. 10.1023/A:1015685325992.

[20] Duysens, J., Van de Crommert, H., Smits-Engelsman, B., and Van der Helm, F. (2002). A walking robot called human: Lessons to be learned from neural control of locomotion. *Journal of Biomechanics*, **35**(4), 447–453.

[21] Dyer, M. G. (1994). Toward synthesizing artificial neural networks that exhibit cooperative intelligent behavior: some open issues in artificial life. *Artif. Life*, **1**(1-2), 111–134.

[22] Figueredo, J. and Gomez, J. (2009). Applying neural fields to the stability problem of an inverted pendulum as a simple biped walking model. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 820–827. IEEE.

[23] Figueredo, J. and Gómez, J. (2009). Evolved neural fields applied to the stability problem of a simple biped walking model. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1775–1776. ACM.

[24] Garcia, M., Chatterjee, A., Ruina, A., and Coleman, M. (1998). The simplest walking model: Stability, complexity, and scaling. *Journal of Biomechanical Engineering*, **120**(2), 281–286.

[25] Garder, L. M. and Hovin, M. E. (2006). Robot gaits evolved by combining genetic algorithms and binary hill climbing. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1165–1170, New York, NY, USA. ACM Press.

[26] Gomez, J. (2004). Self adaptation of operator rates in evolutionary algorithms. In K. Deb, editor, *Genetic and Evolutionary Computation GECCO '04*, volume 3102 of *Lecture Notes in Computer Science*, pages 1162–1173. Springer Berlin / Heidelberg. 10.1007/978-3-540-24854-5-113.

[27] Hasegawa, Y., Arakawa, T., and Fukuda, T. (2000). Trajectory generation for biped locomotion robot. *Mechatronics*, **10**(1-2), 67–89.

[28] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[29] Hermini, H., Rosa?rio, J., and Cassemiro, E. (2001). Proposal of modeling, simulation and implementation of robotics leg prosthesis. In *Annual Reports of the Research Reactor Institute, Kyoto University*, volume 2, pages 1415–1418.

[30] Huang, S.-J. and Huang, C.-L. (2000). Control of an inverted pendulum using grey prediction model. *Industry Applications, IEEE Transactions on*, **36**(2), 452–458.

[31] Huelse, M., Wischmann, S., and Pasemann, F. (2004). Structure and function of evolved neuro-controllers for autonomous robots. *Connection Science*, **16**(4), 249–266.

[32] Jha, R., Singh, B., and Pratihar, D. (2005). On-line stable gait generation of a two-legged robot using a genetic-fuzzy system. *Robotics and Autonomous Systems*, **53**(1), 15–35.

[33] Juang, J.-G. (2001). Intelligent trajectory control using recurrent averaging learning. *Applied Artificial Intelligence*, **15**(3), 277–296.

[34] Juang, J.-G. (2002). Intelligent locomotion control on sloping surfaces. *Information Sciences*, **147**(1-4), 229–243.

[35] Kajiwara, H., Apkarian, P., and Gahinet, P. (1999). Lpv techniques for control of an inverted pendulum. *Control Systems Magazine, IEEE*, **19**(1), 44–54.

[36] Komatsu, T. and Usui, M. (2005). Dynamic walking and running of a bipedal robot using hybrid central pattern generator method. In *IEEE International Conference on Mechatronics and Automation, ICMA 2005*, pages 987–992.

[37] Kun, A. and Miller, W. (1996). Adaptive dynamic balance of a biped robot using neural networks.

[38] Kun, A. and Miller, W. (1997). Adaptive static balance of a biped robot using neural networks.

[39] Kuo, A. D. (2002). Energetics of actively powered locomotion using the simplest walking model. *Journal of Biomechanical Engineering*, **124**(1), 113–120.

[40] Kurz, M. and Stergiou, N. (2005). An artificial neural network that utilizes hip joint actuations to control bifurcations and chaos in a passive dynamic bipedal walking model. *Biological Cybernetics*, **93**(3), 213–221.

[41] Liu, Z., Li, C., and Xu, W. (2003). Hybrid control of biped robots in the double-support phase via h ? approach and fuzzy neural networks. *IEE Proceedings: Control Theory and Applications*, **150**(4), 347–354.

[42] Magdalena, L. and Monasterio-Huelin, F. (1997). A fuzzy logic controller with learning through the evolution of its knowledge base. *Int J Approximate Reasoning*, **16**(3-4 SPEC. ISS.), 335–358.

[43] McGeer, T. (1990). Passive walking with knees. In *Robotics and Automation, 1990. Proceedings.*, volume 3, pages 1640–1645.

[44] Miyakoshi, S., Taga, G., and Kuniyoshi, Y. (2000). Stabilization of periodic motions.

[45] Miyashita, K., Ok, S., and Hase, K. (2003). Evolutionary generation of human-like bipedal locomotion. *Mechatronics*, **13**(8-9 SPEC.), 791–807.

[46] Moriarty, D. E. and Mikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Mach. Learn.*, **22**(1-3), 11–32.

[47] Nakada, K., Asai, T., and Amemiya, Y. (2003). An analog neural oscillator circuit for locomotion controller in quadruped walking robot. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 983–988.

[48] Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., and Kawato, M. (2004). A framework for learning biped locomotion with dynamical movement primitives. In *2004 4th IEEE-RAS International Conference on Humanoid Robots*, volume 2, pages 925–940.

[49] Nolfi, S. and Floreano, D. (2004). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Book.

[50] Orin, D., McGhee, R., Vukobratovic, M., and Hartoch, G. (1979). Kinematic and kinetic analysis of open-chain linkages utilizing newton-euler methods. *Mathematical Biosciences*, **43**, 107–130.

[51] Park, J. (2003). Fuzzy-logic zero-moment-point trajectory generation for reduced trunk motions of biped robots. *Fuzzy Sets Syst*, **134**(1), 189–203.

[52] Paul, C. (2003). Bilateral decoupling in the neural control of biped locomotion.

[53] Paul, C. (2004). Sensorimotor control of biped locomotion based on contact information.

[54] Paul, C. and Bongard, J. (2001). The road less travelled: Morphology in the optimization of biped robot locomotion.

[55] Reil, T. and Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Trans Evol Comput*, **6**(2), 159–168.

[56] Sabourin, C. and Bruneau, O. (2005). Robustness of the dynamic walk of a biped robot subjected to disturbing external forces by using cmac neural networks. *Rob Autom Syst*, **51**(2-3), 81–89.

[57] Safa, A. T., Saadat, M. G., and Naraghi, M. (2007). Passive dynamic of the simplest walking model: Replacing ramps with stairs. *Mechanism and Machine Theory*, **42**(10), 1314 – 1325.

[58] Schwab, A. L. and Wisse, M. (2001). Basin of attraction of the simplest walking model. In *International Conference on Noise and Vibration*, volume 21363, pages 2186–208.

[59] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.*, **10**(2), 99–127.

[60] Stepanenko, Y. and Vukobratovic, M. (1976). Dynamics of articulated open-chain active mechanisms. *Mathematical Biosciences*, **28**, 137–170.

[61] Strösslin, T. and Gerstner, W. (2003). Reinforcement learning in continuous state and action space. In *ARTIFICIAL NEURAL NETWORKS - ICANN*.

[62] Taga, G., Yamaguchi, Y., and Shimizu, H. (1991). Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biol Cybern*, **65**(3), 147–159.

[63] Vaughan, C. (2003). Theories of bipedal walking: An odyssey. *Journal of Biomechanics*, **36**(4), 513–523.

[64] Venkataraman, S. (1997). A simple legged locomotion gait model. *Rob Autom Syst*, **22**(1), 75–85.

[65] Wilson, H. and Cowan, J. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophys J*, **January; 12(1)**, 1–24.

[66] Wisse, M., Schwab, A. L., and van der Helm, F. C. T. (2004). Passive dynamic walking model with upper body. *Robotica*, **22**(06), 681–688.

[67] Wisse, M., Schwab, A. L., Linde, R. Q. V. D., and Helm, F. C. T. V. D. (2005). How to keep from falling forward: elementary swing leg action for passive dynamic walkers. *IEEE Transactions on Robotics*, **21**(3), 393–401.

[68] Wolff, K. and Nordin, P. (2001). Evolution of efficient gait with autonomous biped robot using visual feedback.

[69] Wolff, K. and Nordin, P. (2003). Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming.

[70] Woo, S.-Y., Abramowitch, S., Kilger, R., and Liang, R. (2006). Biomechanics of knee ligaments: Injury, healing, and repair. *Journal of Biomechanics*, **39**(1), 1–20.

[71] Wu, Q., Sepehri, N., and He, S. (2002a). Neural-based control and stability analysis of a class of nonlinear systems: Base-excited inverted pendulums. *Journal of Intelligent and Fuzzy Systems*, **12**(2), 119–131.

[72] Wu, Q., Sepehri, N., and He, S. (2002b). Neural inverse modeling and control of a base-excited inverted pendulum. *Engineering Applications of Artificial Intelligence*, **15**(3-4), 261–272.

[73] Yamasaki, F., Endo, K., Asada, M., and Kitano, H. (2003a). Control design principle of a low-cost humanoid system using a genetic algorithm. *Adv Rob*, **17**(8), 779–790.

[74] Yamasaki, T., Nomura, T., and Sato, S. (2003b). Possible functional roles of phase resetting during walking. *Biological Cybernetics*, **88**(6), 468–496.

[75] Yanase, T. and Iba, H. (2006). Evolutionary motion design for humanoid robots. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1825–1832, New York, NY, USA. ACM Press.

[76] Yonemoto, K., Kato, I., and Shima, K. (1985). Technology forecast on industrial robots in japan. In -, volume 1, pages 51–58. Japan Industrial Robot Assoc.

[77] Zhang, Y., Jiang, D., and Wang, J. (2002). A recurrent neural network for solving sylvester equation with time-varying coefficients. *Neural Networks, IEEE Transactions on*, **13**(5), 1053–1063.

[78] Zhou, C. (2002). Robot learning with ga-based fuzzy reinforcement learning agents. *Inf Sci*, **145**(1-2), 45–68.

[79] Zhou, C. and Meng, Q. (2003). Dynamic balance of a biped robot using fuzzy reinforcement learning agents. *Fuzzy Sets and Systems*, **134**(1), 169–187.