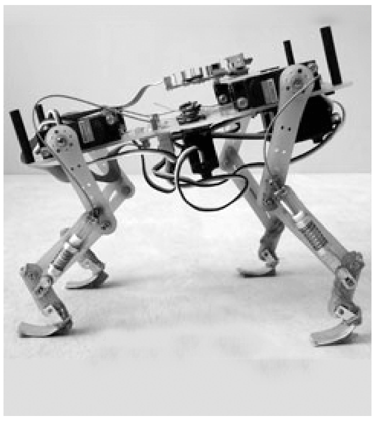
École Polytechnique Fédérale de Lausanne
Projet de Master
2010
Section de Microtechnique
TULEU Alexandre

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

BIOLOGICALLY INSPIRED ROBOTICS GROUP (BIRG)

Master Project Report

Improvement of the Cheetah Locomotion Control.



Professor : Auke Jan Ijspeert
Supervisor : Alexander Sproewitz
Date : January 29, 2010 *Edited version*

Abstract

This project relates to the locomotion of *Cheetah* , a small and light robot, who features tri-segmented, biologically inspired, compliant legs. It is a continuation of former BIRG students projects, who brought the conception of two simulated model and two prototypes.

This project focuses on the simulation aspect, where a Central Pattern Generator (CPG) is used to abstract the control of the locomotion, and evolutionary algorithms are used to perform *offline learning* with a model of the robot in the WEBOTS simulator.

At first, the model of the robot has been improved to be more accurate toward the prototypes, particularly on the level of the asymmetric compliant behavior of the leg. Afterwards the control software has been carried out with the use of new Optimization and CPG framework developed at BIRG . Then an implementation of locomotion self-stabilization principles, observed in quadrupedal locomotion, has been introduced inside a Central Pattern Generator previously designed by Ludovic RIGHETTI. Finally, the extensive study of a learned trot gait has been completed, measuring the benefit of the implementation of such behaviors.

Contents

Introduction

In recent years, there are more and more designed robot, which are biomomimetics copies of animals, like for example the Sony Aibo entertainment robot. But bioinspired robotics is not just making robot that look like animals, it is understanding the underlying principle of the animal, and adapt them to machine [WHI$^+$00].

Cheetah is a robot that try to follow this idea : it features compliant, pantographic leg, that are designed from observations in small mammals [WHI$^+$00, FB06]. In addition, previous work on this robot [Rie08, Kvi09], used Central Pattern Generator (CPG) to control its locomotion : they are a model of how animals produce rhythmic neural activities to control their movements. CPG abstract the command of the whole dynamic, with a few parameters. One approach is to use optimization algorithms to learn the optimal value of these parameters, to perform locomotion.

The same approach is taken in this project. After a in-depth presentation of *Cheetah* and locomotion behaviors in nature, an improvement of the simulated model of *Cheetah* is presented. Then an extension of an existing CPG is proposed. Finally the study of a learned trot gait with the work flow is carried out.

Part 1

Presentations and Motivations

Legged locomotion is a well spread mechanism in nature. However, due to the complex mechanisms it implies, (complex movement of limbs, complex bodies dynamics, non continuous contact with the ground, ...), it is also one of the most challenging problem in robotics. A way to outcome these difficulties, as stated by [WHI$^+$00] is to take inspiration from the nature, as the behavior found in animal, results from evolution process of thousands of generations.

Cheetah is an implementation of this idea. Thus, to understand its conception choice, locomotion behavior found in quadruped mammals will be introduced before the presentation of the actual robot and the previous work about this platform. Finally the goal of this project will be introduce.

1.1 Locomotion behavior in Nature for inspiration

As it names let it guess, *Cheetah* takes inspiration from the mammals. After a brief presentation of behavior and "blueprints" found in mammals, it will present how locomotion control is abstracted with Central Pattern Generator (CPG).

1.1.1 Similarities in locomotion between quadruped mammals.

Study of the animal locomotion behavior is a growing field in biomechanics [Ale84, FSS$^+$02, FB06, WBH$^+$02, Sch05]. As stated by [WHI$^+$00], the extension of the biomechanics to the animal yield to the extraction of principles that could extend the engineer toolbox. As the common ancestor, in an evolution point of view, of large quadruped and biped mammals like horse, dog and humans are the small mammals, the study should focus in first place on the later.

[FB06, WHI$^+$00, WBH$^+$02], compares and extract several similarities between several species, that are useful to understand why animal are so efficient in locomotion.

- The limb of mammals are almost Tri-segmented shaped (see figure 1.1), and have the same forelimb and hindlimb configuration in term of functionality [FB06]. However the serial segments are no longer homologous.
- The progression is mainly due to the retraction - caudal displacement of the limb - of the most proximal segment i.e., the scapula for forelimb and the femur for the hindlimb, [FB06].
- Two segments (scapula/lower arm for forelimb and femur/foot) operate almost parallel during retraction of the limb, as in a pantograph mechanism, [FB06, WHI$^+$00].
- High limb compliance is a general principle for small mammals, and this help the self-stabilization of the limb, i.e., the stable locomotion of the animal, in presence of external disturbance without the necessity of a sensory feedback, [FB06], More details are given in the section 1.1.2
- In asymmetrical running gait, like galop, canter, or bound, the animals extensively use the bending of their spinal cord, and thus the back of the spine become the main propulsive segment for the hindlimb [FB06, WHI$^+$00].

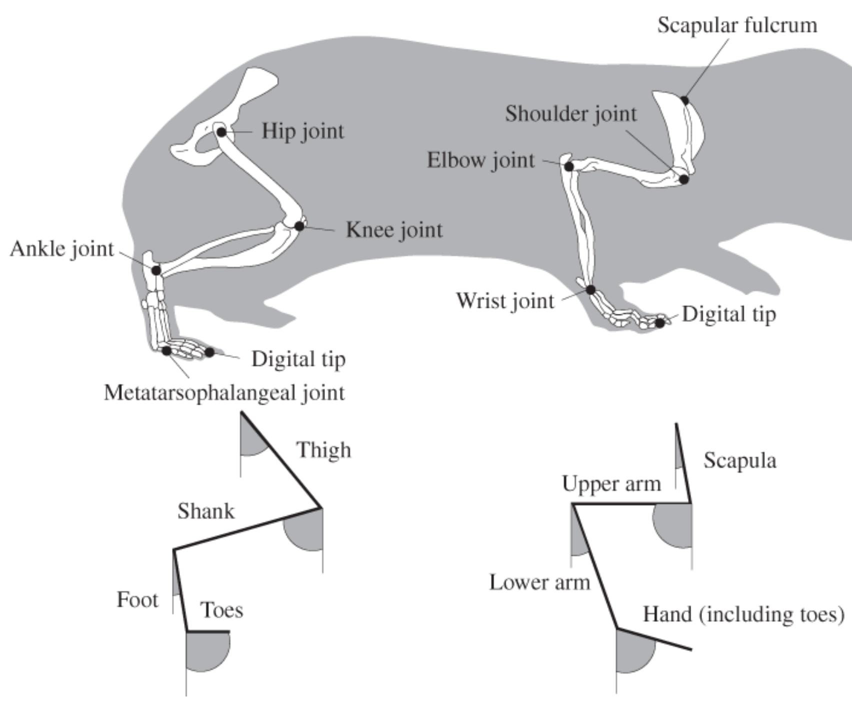


Figure 1.1 – The tri-segmented limb abstraction for small mammals. If the most distal segment are omitted, the limbs are segmented in three parts : (a) for the forelimbs : scapula, humerus and lower arm, (b) for the hindlimbs : femur, shank and foot . Taken from [FB06].

- One can observes in certain animals, a connection between the wrist and precedent segment. This connection is made by elastic tendons, which also gives compliant foot/hand.

One of the most important point before, is the self-stabilization capacity of the leg in animals. Indeed, if the robot dynamics, passively goes into a limit cycle, it simplifies its control effort [IP04]. Therefore one might be interested in other self-stabilization behaviors.

1.1.2 Leg retraction as a Self-Stabilization leg control strategy

As stated previously, the self-stabilization of the locomotion seems to be a general strategy among the animals. At least three self-stabilization principle has been observed :

1. leg stiffening : animals can modulates the leg stiffness to perform different gait, and also stabilize it [DK09]. However it is too difficult to implement such a mechanical behavior for a light robot, so this possibility has not been investigated further.

2. leg extension : it has been shown for small birds, that at each stride, the leg is automatically fully extended if the contact on the ground has not been made at the expected touchdown. This behavior limit the changes of height of the Center of Mass (CoM), under ground height disturbance like step up and step down, which help the self-stabilization of the animal [DVB09]. However as this is not a well known behavior for quadruped locomotion, it had not been investigated further.

3. leg retraction : that state the animal start the retraction of the limb a certain time before the touchdown. Then the foot touches the ground with a non-zero horizontal velocity [SGH03].

To bring out this last behavior [SGH03] uses a spring mass model, where the angle of attack is controled. This model although it is conservative, is asymptotically stable, and predict human data at moderate running speed. However it was not stable for low speed, and a high accuracy of the angle of attack is needed to achieve stability.

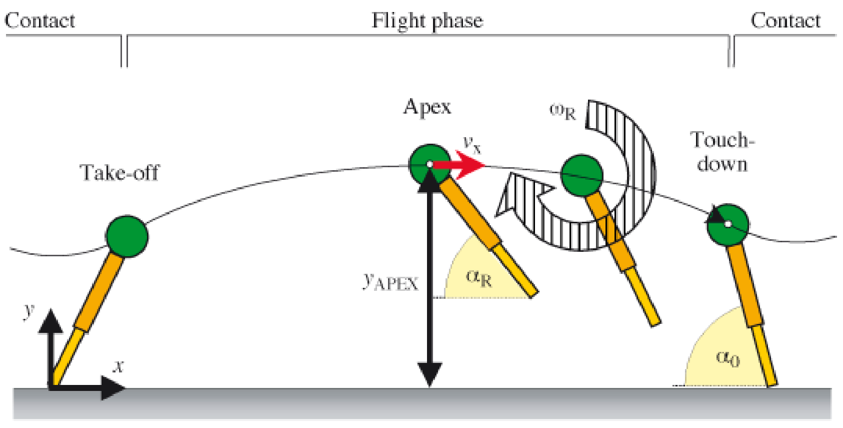


Figure 1.2 – Leg retraction principle for [SGH03]. The angle at APEX and the rotational velocity are controlled, which provides an increase of the basin of attraction of the global system according to [SGH03].

[SGH03] extend this model : instead of controlling the angle of attack at each touchdown, controls the angle at the APEX - point where the height of the Center of Mass (CoM) is maximal - and the rotational velocity of the leg, once the APEX is passed (see figure 1.2) . Thus it showed that the bigger this speed is, the more it increase the size of the basin of attraction of the system. Moreover this model is also able to find limit cycle for low speed running gait.

1.1.3 Central Pattern Generators : a model for animal locomotion control

Central Pattern Generators are now a well diffused assumptions in biologically fields, as stated in [Ijs08], that movement in animals are centrally generated through block of neurons, that generate rhythmic pattern of neural activity, without receiving rhythmic input.

They are a big field in biologically research, and it has been established several neurobiological models, at different levels. All these models are good tools for describing the movements, and there is more and more research over the past few year, that try to use this model to control the locomotion of robots. [Ijs08] propose several reasons for preferring the use of a CPG instead of classical methods :

1. The purpose of CPG models is to exhibit limit cycle behavior, therefore they are more robust to perturbation.

2. CPG models have only few parameters to drive speed, direction or even the type of gait. Therefore, at higher level, the command is simplified, as command just have to provide only a small number of variable.

3. CPG are ideally suited for sensory feedback, that able us to control finely the mutual interaction between the CPG and the body dynamics.

4. CPG are a really good substrate for learning and optimization algorithm.

They are several model of CPG that are well suited for robotics control from standard to modular robot, like phase oscillator [IC07, SMM$^+$08] or Hopf modified oscillators [RI08, Wie08]. It is the last one that is used in *Cheetah* . Note, as an analogy with biological study, the oscillators are also named neurons, as in biological fields, these oscillators are just a mathematical macroscopic model of how a group of neurons acts.

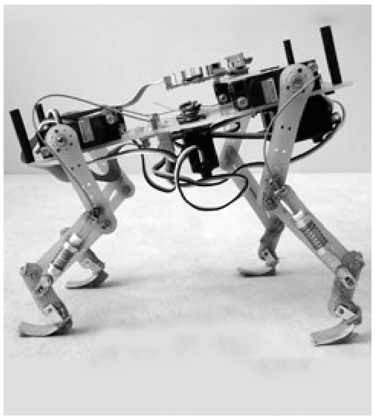


Figure 1.3 – First prototype of the *Cheetah* robot.

1.2 *Cheetah* a compliant leg robot

1.2.1 Global presentation of the *Cheetah*

[WHI$^+$00] propose several principles for the design of bioinspired quadruped robots. These principles has been partially applied by [Rut08] in the design of the *Cheetah* robot. The first prototype is then a small and light robot, who has a size similar to a small cat. It features tri-segmented pantographic leg, which are compliant, and actuated by two Degree of Freedom (DoF), in the hip and in the knee. In agreements with [WHI$^+$00], all its actuators are located on the main part, the trunk, and the actuation of the knee is made through Bowden cables (section 1.2.2).

1.2.2 Asymmetric compliance leg design

The legs of *Cheetah* are tri-segmented; They are designed with a pantographic mechanism (see figure 1.4) such as the femur (l_3) and the foot (l_1) are always parallel. The kinematic may be abstracted by the leg segments $(\lambda_1, \lambda_2, \lambda_3)$. [Rut08] proposed the following relative length (1.1) for one leg, and a total leg length of 150 mm for the forelimb and 180 mm for the hindlimb.

$$\begin{aligned} \lambda_3 &= \quad 0.39 \\ \lambda_2 &= \quad 0.45 \\ \lambda_1 &= \quad 0.16 \\ \lambda_p &= \quad 12mm \end{aligned} \tag{1.1}$$

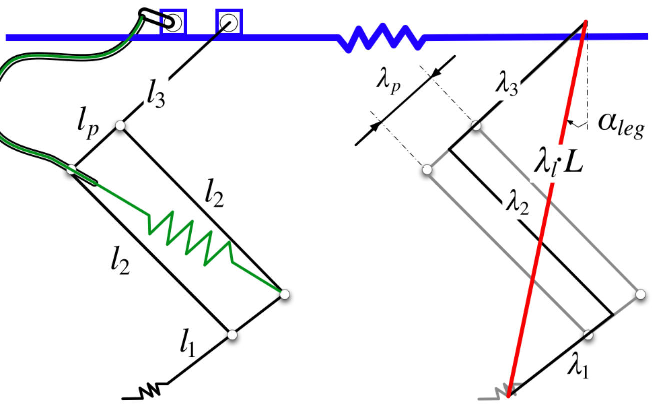
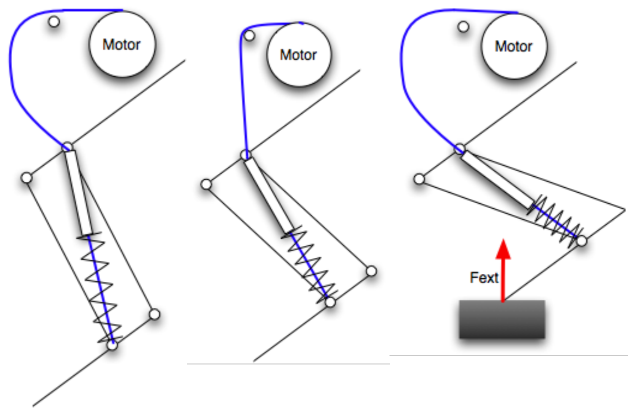


Figure 1.4 – Schematics of the Cheetah robot, from [Rut08]



(a) Resting position **(b)** Actuated **(c)** Under external forces

Figure 1.5 – Passive functioning of the compliant mechanism of the pantographic leg of the *Cheetah* . Here, for drawing simplification, the actual Bowden cable is replaced by a simple wire.

The compliance of the leg is within the knee mechanism. As shown in figure 1.5, the motor is pulling a Bowden cable that tends to reduce the diagonal of the parallelogram of the tibia segment, and then contract the leg. One can sees three operation modes :

1. If there is no action of the servo, and no action on the distal segment, then the spring extends the leg at its maximum (figure 1.5a).
2. If the motor pulls the wire, then it contract actively the leg (figure 1.5b).
3. In the presence of external interaction (if the robot stands on the ground), then the wire is loosened, disconnecting the motor from the system. The equilibrium is obtained by the spring, counter-acting the gravity force from the weight of the robot.

[Rut08] state that this leg can be abstracted by a single compliant leg, with the use of the following equation for the leg length and angle :

$$l_\lambda = L\lambda_l = \sqrt{\lambda_2^2 \sin^2\varphi + (\lambda_1 + \lambda_3 - \lambda_2 \cos(\varphi))^2} \quad (1.2)$$

$$cos(\varphi_3 - \alpha_{leg}) = \frac{\lambda_1 + \lambda_3 - \lambda_2 . cos(\varphi)}{\lambda_l} \quad (1.3)$$

1.2.3 Previous simulation model of *Cheetah*

In order to study the locomotion of the *Cheetah* , a WEBOTS model of the robot had been made by Yvan BOURQUIN and Martin RIESS [Rie08] (see figure 1.6a). The purpose was to use this model and implement a Central Pattern Generator designed by Ludovic RIGHETTI (see [RI08] and section 3.1), thus a quantitative study of a couple of gait (walk and trot) have been done. The approach used by [Rie08] was to use a mechanical model really close to the reality. However the model was complex, causing large computation time. Thus it prevents the use of optimization algorithm, which large amount of trial for the optimization of one gait.

Then another study had been made by Ivan KVIATKEVITCH [Kvi09] (see figure 1.6b), with a more abstracted model of the leg. This new model did not perform sufficiently, so it had been decided to use a more reality-close one. However [Kvi09] pointed out some flaw in the WEBOTS simulation like the bad quality of contact modelization and the need to filter the sensory feedback information

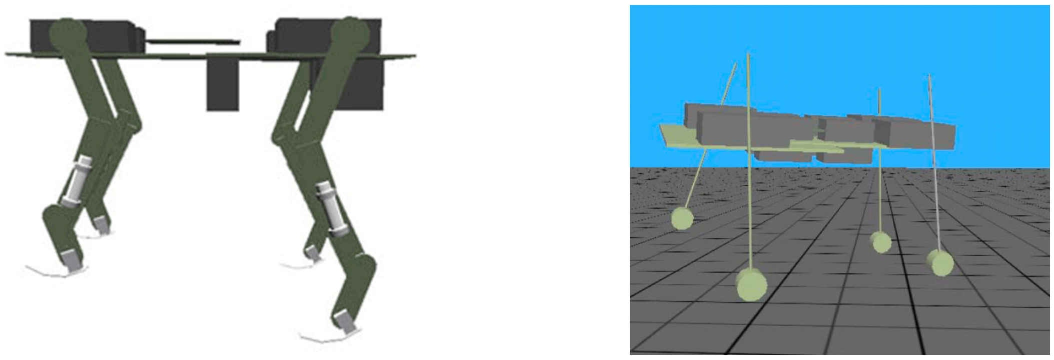

(a) Realistic model by Martin RIESS **(b)** Abstracted Model by Ivan KVIATKEVITCH

Figure 1.6 – The two previous models of *Cheetah* in WEBOTS

1.3 Goals of the project

1. Recover and update the previous work on the cheetah platform :
   - Choose among the previous WEBOTS model made by [Rie08] and [Kvi09].
   - Create new foot with just a straight segment, connected with a rotational spring (stiffness is to determine).
   - optionally :
     - Make the geometry of the model as a parameter for optimization.
     - Add a new DoF for each legs (hips).
     - Add a spine to the robot. Look in the literature for possible spines.
   - Assure the model is sufficiently coherent with the reality, by making him carrying weight, drop it on the ground, and some naive gaits.
2. Prepare the optimization process by using the Optimization framework made by Jesse VAN DEN KIEBOOM. It should be extended for using any optimization algorithm wanted, or type of Central Pattern Generator (CPG).
3. Use the CPG, as described by Ludovic RIGHETTI [RI08]. The foot trajectory should be generated, using two principles found in animal legged locomotion. This could be done by either inverse kinematics or integrated in the CPG. The leg retraction behavior should also be carried out and tested .
4. Make extensive research on :
   - search for a good fitness function that does take into account control of speed, gait and direction changes (with the emphasis on speed and robustness).
   - the stability of the walk, by examining some criteria as Zero moment Point or Pointcaré return map. This information could be also use to increase the basin of attraction of the locomotion, by using it in the CPG.
   - study the robustness of cheetah under terrain changes of slope (up, down, tilt right and left), and changes of friction.
   - the optimal leg geometry (optional).

Part 2

Modelization of *Cheetah* in WEBOTS

The first task to this project was to choose between Martin RIESS or Ivan KVIATKEVITCH model. As the introduction of Jesse Optimization framework helped to distribute evolutionary algorithms over cluster of computer, speed is not the first priority. Therefore the more realistic Model has been chosen.

At first some minor changes had been done on the model :

1. The legs were redesigned to have both a functional length of 160 mm (see section 1.2.2).
2. The legs have now the same cranial orientation (first segment femur/scapula is oriented toward the head). Previously the forelimb had their femur caudally oriented.
3. The density of the materials has been updated.
4. The weight of the servo-motor has been updated due to a change of type between the first and second prototype of *Cheetah* .

However the biggest changes were maid for the Knee actuator. Indeed the original model were biased. The system described in 1.2.2 was indeed modelized by a linear motor with a P Controller. Thus the asymmetrical behavior was obtained by the fact that in one direction the maximum force available is different from the other. This is not the case in reality, as the motors see a non-constant non-linear charge. Thus a better modelization of the servo-motor had been developed.

2.1 Knee Articulation Modelization

The *Cheetah* robot knee actuator present an asymmetry (section 1.2.2) and then cannot be straight forward modelized in WEBOTS using only WEBOTS primitives. Therefore a mathematical model of the mechanism must be created, who features a more precise model of the servo-motor, whose characteristic had been guessed.

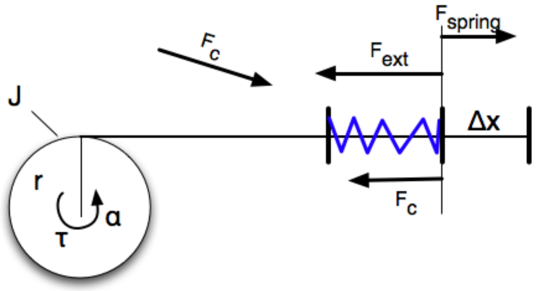


Figure 2.1 – Mechanism of the knee actuator.

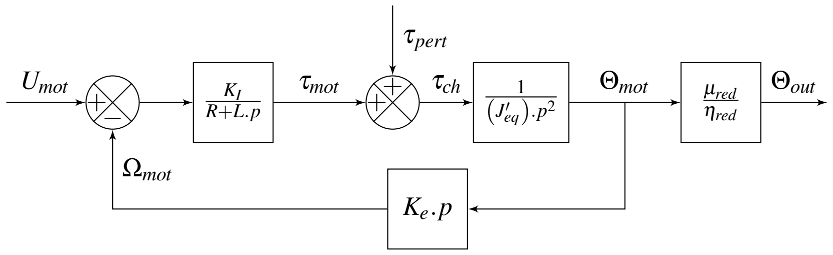


Figure 2.2 – Model of the servo-motor.

The knee mechanism is show in figure 2.1. He is comprised of a servo-motor (α, τ) that drive a wheel, on which is fixed a cable with a spring. The movement of the cable Δx_c can also be different from Δx. One can found the following relation :

$$\Delta x_c = \alpha.R \tag{2.1}$$

where τ_c is the couple from the cable tension, and R the ratio of the gear :

$$\tau_c = r.F_c \tag{2.2}$$

and where F_c is the cable tension. For the servo-motor, the following model is used as described by figure 2.2. The following notation is used :

- K_I and K_E respectively the torque and speed constant of the DC motor.
- R and L respectively the intern resistance and inductance of the DC motor
- J_{eq} the equivalent inertia of the motor axis with its charge.
- μ_{red} and ν_{red}, respectively the efficiency and the ratio of the gear.

It results the following 2nd order equation that drives the motor :

$$J_{eq}R\ddot{\alpha}(t) + K_E K_I \dot{\alpha}(t) = \frac{\mu_{red}}{\eta_{red}} K_I U(t) - \frac{\mu_{red}^2}{\eta_{red}^2} R\tau_c(t) \tag{2.3}$$

Finally, it is assumed that the motor is controlled by a PID controller (however, one must notice that the constructor could have settled a more sophisticated controller) :

$$U(t) = k_p\left(\alpha_{des}(t) - \alpha(t)\right) + k_d\left(\dot{\alpha}_{des}(t) - \dot{\alpha}(t)\right) + k_i \int_0^t \left(\alpha_{des}(t) - \alpha(t)\right) dt \tag{2.4}$$

2.1.1 Modelization of the cable

Their are two ways to modelize the cable : by a spring or a rigid body.

2.1.1.a Asymmetric Spring Model

The cable tension is defined by using the equation of a spring with a constant $k_c >> k_s$. Then the value of F_c is given by equation (2.5) :

$$F_c = \begin{cases} k_c\left(\Delta x_c - \Delta x\right) & \text{,if } \Delta x_c \geq \Delta x \\ 0 & \text{, else} \end{cases} \tag{2.5}$$

2.1.1.b Asymmetric rigid body

In the case of a pure rigid cable, then the value of F_c can be computed as the solution of an optimization problem :

$$\min_{F_c}(F_c)\ ,\ with\ :\begin{cases}\alpha r \leq & x\\ \dot{\alpha} r \leq & \dot{x}\,,\, if \qquad \alpha r = x\\ & \qquad equation(2.3)\\ F_c \geq & 0\end{cases} \tag{2.6}$$

Equation (2.3) show us that F_c is a decreasing function of $(\alpha,\dot{\alpha},\ddot{\alpha})$. One can compute F_c using the algorithm 1.

Algorithm 1 Computation of the tension of the rigid Bowden cable

```
1: procedure COMPUTETENSIONINRIGIDCASE($x(t)$,$\dot{x}(t)$, $\alpha(t)$, $\dot{\alpha}(t)$, $\ddot{\alpha}(t)$)
2:     if $\alpha(t).r \geq x(t)$ then                                   ▷ We violate our first boundary condition
3:         $\alpha_N = \frac{x(t)}{r}$                           ▷ We decrease $\alpha$ in order to respect our first condition
4:         $\dot{\alpha}_N = \frac{\alpha_N - \alpha(t)}{dt} + \dot{\alpha}(t)$              ▷ We compute the corresponding speed
5:         if $\dot{\alpha}_N > \dot{x}$ then                           ▷ We don't respect the second condition
6:             $\dot{a}lpha_{NN} = \frac{\dot{x}}{r}$                ▷ decrease speed in order to respect second condition
7:             $\ddot{\alpha}_{NN} = \frac{\dot{\alpha}_{NN} - \dot{\alpha}}{dt} + \ddot{\alpha}$                 ▷ Compute corresponding acceleration
8:             $\alpha_{NN} = (\dot{\alpha}_{NN} - \dot{\alpha}).dt + \alpha$   ▷ Compute corresponding position, we have $\alpha_{NN} < \alpha_N$,
   so first condition is still valid
9:                         ▷ Now as we have the "should be" state, we compute $F_c$ using equation (2.4)
10:            $U = k_p(\alpha_{des} - \alpha_{NN}) + k_d.(\dot{\alpha}_{des} - \dot{\alpha}_{NN}) + k_i.\int_0^t(\alpha_{des}(t) - \alpha_{NN}(t))\,dt$
11:            return max $\left(\frac{\eta_{red}^2}{\mu_{red}^2.R}\left(\frac{\mu_{red}}{\eta_{red}}.K_I.U - J_{eq}.R\ddot{\alpha}_{NN} - K_E.K_I\dot{\alpha}_{NN}\right), 0\right)$  ▷ We make sure we
   have the last condition full-filled
12:        end if
13:        $\ddot{\alpha}_N = \frac{\dot{\alpha}_N - \dot{\alpha}}{dt} + \ddot{\alpha}$
14:                        ▷ Now as we have the "should be" state, we compute $F_c$ using equation (2.4)
15:        $U = k_p(\alpha_{des} - \alpha_N) + k_d.(\dot{\alpha}_{des} - \dot{\alpha}_N) + k_i.\int_0^t(\alpha_{des}(t) - \alpha_N(t))\,dt$
16:        return max $\left(\frac{\eta_{red}^2}{\mu_{red}^2.R}\left(\frac{\mu_{red}}{\eta_{red}}.K_I.U - J_{eq}.R\ddot{\alpha}_N - K_E.K_I\dot{\alpha}_N\right), 0\right)$   ▷ We make sure we have
   the last condition full-filled
17:    end if
18:    return 0                                                  ▷ here the cable is loosened
19: end procedure
```

2.1.2 Servo Specifications

This new mathematical model is based on lot of the mechanical characteristics of the servo-motor, like the gear ratio and efficiency, motor inner characteristics ... However these characteristics are not given for almost all servo-motor, and particularly for the one used by *Cheetah* (DYNAMIXEL RX-28). Therefore these parameters have to be estimated from the few characteristic given by the constructor (see table 2.1) : as the gear ratio of the servo is given, one can estimate some of the characteristics of the motor.

As the mark and series of the motor, the motor with the closest characteristics can be found in the constructor documentation [Mot], (table 2.2).

For the gear, it seems that DYNAMIXEL use a custom one. So the value of a Maxon gear that could be fitted in the servo-motor is taken. Its specifications are given by table 2.3.

Type	Constructor Data	Estimated value for the motor
Gear ratio	1:193	-
Weight	$72\,\text{g}$	$< 72\,\text{g}$
Applied Voltage	$12\,\text{V}$	$12\,\text{V}$
Stall Torque	$2{,}78\,\text{Nm}$	$\frac{2{,}781}{193} = 14{,}04\,\text{mNm}$
Maximum velocity with no load	$59{,}9\,\text{rpm}$	$59{,}9 * 193 = 11560\,\text{rpm}$
Starting Current	$1{,}2\,\text{A}$	$1{,}15\,\text{A}$
Mark and series of the motor	Maxon ReMax Series	Maxon ReMax Series

Table 2.1 – Dynamixel RX-28 constructor specifications, from [Rob], and estimated value for its DC motor. The weight and starting current are smaller, as one must take into account the weight of the additional gear and packaging, and the current consumption of the electronic command.

Characteristic	Value
Nominal Voltage	12 V
No load Speed	11500 rpm
Stall torque	$14{,}4$ mN.m
Starting Current	$1{,}45$ A
Internal resistance	8 Ω
Torque Constant	$9{,}32e^{-3}$ N.m.A^{-}1
Speed Constant	$9{,}92e^{-3}$ V.s.rad^{-}1
Weight	26 g
Rotor Inertia	$0{,}868$ g.cm^2

Table 2.2 – Specification of the Maxon ReMax 21 48 97, from [Mot]

Characteristic	Value
Ratio	1 : 199
Inertia	$8e^{-3}$ g.cm^2
Efficiency	$0{,}53$
Weight	30 g

Table 2.3 – Guessed specification for the gear, from [Mot]

2.1.2.a Tuning of the PID parameter

Once the characteristic are chosen, the last step is to find the good parameter for the PID constant. For that purpose the Ziegler-Nichols closed loop method [Wik] is applied on the model of one motor. Ziegler-Nichols is applied on the servo-motor alone because the end-user of the servo-motor doesn't have access to the tuning of the controller.

The ultimate gain obtened is $K_U = 1870$ with oscillations of period $T_U = 72\,\text{ms}$. Therefore the value of the PID are given in table 2.4 for the PID.

k_p	k_d	k_i
$K_U.0{,}6 = 1123$	$k_p\frac{T_U}{8} = 10{,}09$	$\frac{T_U}{2k_p} = 0{,}025$

Table 2.4 – PID parameter identified by Ziegler-Nichols closed loop method

2.1.3 Implementation in WEBOTS

2.1.3.a Implementation using a force driven linear servo-motor

The following system has been firstly implemented on WEBOTS , using a linear WEBOTS actuator to model the global system composed of the servo, the spring (with mechanical stops) and the cable. This servo is directly commanded by the force computed by the new model.The two cable model described in section 2.1.1 are implemented, but only the spring one currently stable. The following algorithm is used to update the state of the system :

1. Get the value x_t from WEBOTS , and compute $\dot{x}_t$
2. Retrieve previously computed α_t, $\dot{\alpha}_t$, $\ddot{\alpha}_t$.
3. Compute F_c^t using one of the two model described in section 2.1.1.
4. Compute F_{out}, the actual force to add in WEBOTS , by adding it the spring force and a damping force.
5. Derive the WEBOTS linear servo-motor with the computed force F_{out}
6. Update α_{t+1}, $\dot{\alpha}_{t+1}$, $\ddot{\alpha}_{t+1}$ using equation (2.3) and a Runge-Kutta 4 method (instead of a Euler one to avoid computation instability).

In order to prevent some instability in the model, the following limitations had been added :

- The order given to the servo are prefiltered, not to exceed maximum speed the servo can give ($V_{max} = 6,27$ rad.s^{-1}). It is done by a low-pass filter (see algorithm 2)
- The cable tension is limited to an amount $Fc^{max} = 45\,\mathrm{N}$ with the spring cable model.
- A damping value of $60\,\mathrm{Nm}^{-1}\mathrm{s}$ has been added on the spring. This value had been empirically computed to reduce oscillation of the spring.

Algorithm 2 Low pass filter on command.

```
procedure LOWPASSFILTERONCOMMAND($\alpha_{des}^t, \alpha_{des}^{t-1}$)
    if $\left|\frac{\alpha_{des}^t-\alpha_{des}^{t-1}}{dt}\right| > V_{max}$ then
        return $\frac{\frac{\alpha_{des}^t-\alpha_{des}^{t-1}}{dt}}{\left|\frac{\alpha_{des}^t-\alpha_{des}^{t-1}}{dt}\right|}.V_{max} + \alpha_{des}^{t-1}$
    else
        return $\alpha_{des}^t$
    end if
end procedure
```

2.1.3.b Comportment under extreme conditions

A minimalist simulation has been computed, where the four knee actuators had a command that was over the speed limit the servo can admit, in order to see the model consistency. The results are made of a movie of the simulation (`http://akay.fr/data/CheetahHardCondition.avi`) and the output measured from WEBOTS (see figure 2.3)

This figure shows the following results :

- The servo-motor doesn't see any load, because desired and real angular position concord. Indeed all the charge is taken by the springy cable.
- There is an expected mismatch between desired and observed contraction in absence of disturbance, because of the use of a spring cable.
- The cable force is noisy, as oscillations of high frequencies appear, but it is asymptotically stable.

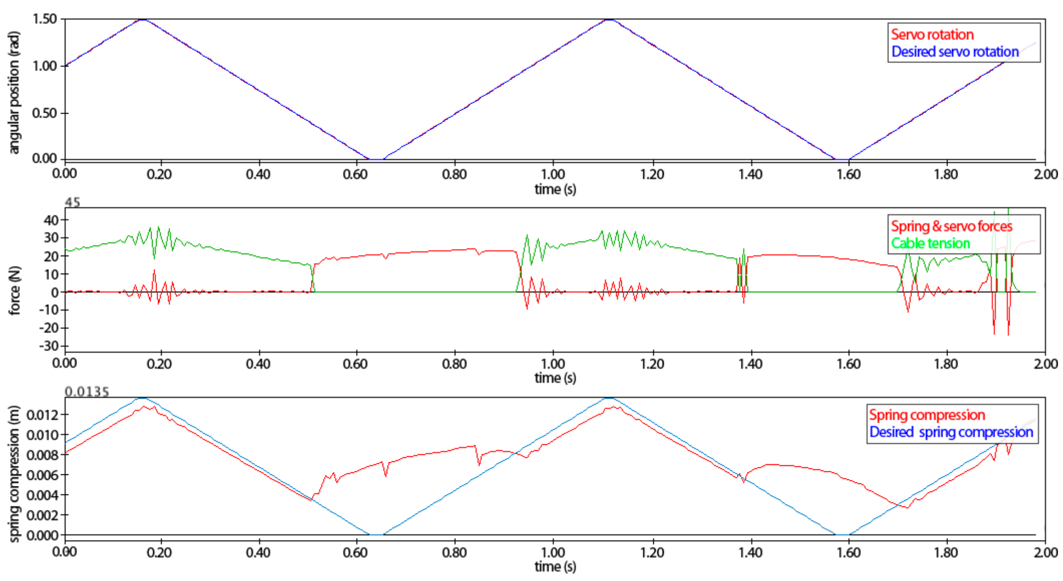


Figure 2.3 – Output of a knee actuator (with a springy cable) at the limit of its capacities. One can sees that : (a) the desired servo position match the computed output. This is less due to the well tuning of PID parameters, than the fact that the cable (because it's a spring) is taking all the load (b) the spring movement is noisy, with high frequency oscillations. However, its seems asymptotically stable. (c) the final contraction of the spring is always lower than the desired one, in absence of disturbance. It is not surprising due to the springy cable. However the compliance of the leg appears when the leg touches the ground.

However this last point brings up some problems : These high oscillations make the model overcome overriding mechanical stop constraint. Under a high disturbance the tibia segments could go over their end positions and become crossed each other. Such a problem appears because these constraints are too soft. Open Dynamics Engine (ODE), the physics engine upon which WEBOTS is based, permits the tuning of the softness of these constraints through the tuning of the Constraint Force Mixing (CFM)/Error Reduction Parameter (ERP) parameters.

An empirical tuning of these parameters was performed, in order to avoid such a situation. However this brought another problem as it will be discussed in section 2.1.4.

2.1.3.c Discussion and further improvement

Although this method works to implement the symmetric behavior of the leg, some further improvements are needed :

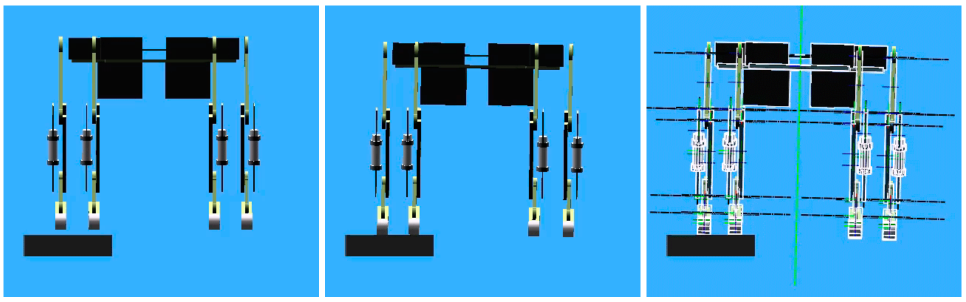
- The first annoying thing, is that the servo doesn't see any charge, so a workaround must be found.
- The hard cable model is not working, because, in opposition with the spring mode, it sends all the load of the mechanism to the servo-motor, causing too high torques on the servo-motor axis, and then causing numerical explosion of the model.
- Under high constraint, the WEBOTS joints of the mechanism are oscillating around their axis, due to the high constraints of the mechanism. One way to deal with is to tune more finely the ERP/CFM parameters for these particular joints, but it is not an easy job to do in WEBOTS , because this must be done inside a physics plug-in (WEBOTS only let use manage ERP and CFM on a global perspective).
- The spring constant is limited to an amount of $3000\,\mathrm{N.m^{-1}}$, to preserve consistancy of the mechanism simulation. Although the prototype had lower value ($\approx 2300\,\mathrm{N.m^{-1}}$) this is a low value compared to the one Martin RIESS used ($5000\,\mathrm{N.m^{-1}}$)

One way to deal with all of these problems would be to reduce the time step of the simulation. There is not so many scopes for this, because the current time step is already small (4 ms). One way too deal with the first two problems alone, is to only decrease the time step of the integration of the model, by repeating step 3,4 and 6 of the algorithm (section 2.1.3.a) several times for each simulation step. However this possibility has not been yet tested.

2.1.4 ODE unrealistic behavior with the *Cheetah* knee mechanism

2.1.4.a Problem description

As described by figure 2.4, the model of the *Cheetah* was subjected to unrealistic behavior : the complete robot was not able to fall on the side, or more precisely, takes several minute to roll of a few degrees.



(a) $t = 0\,s$ **(b)** $t = 30\,s$ **(c)** $t = 60\,s$

Figure 2.4 – ODE physics simulation unrealistic behavior. Although *Cheetah* should fall on the side in a short time, it takes minute to do so.

Yvan BOURQUIN, who design in first place the model under WEBOTS , observed this behavior, by increasing the CFM parameter. However :

- A good tradeoff between the resolution of this problem and the stability of the new knee actuator model was not found.
- It is pretty hard for a human observer, to determine the correctness of the simulation, without a pretty good analytical model of the system, or real data to compare with the numerical model, And at the time of this project, none of both was available, and then it is hard to find the correct value for the CFM.

Therefore, it might be useful to present the meaning in ODE of these parameters.

2.1.4.b Presentation of the ERP/CFM parameters

Open Dynamics Engine is the physics engine used by WEBOTS [Cyb], and like almost physics engine, it is based on the integration of the 2nd Newton law for rigid bodies [ODE]. The contact forces and joint constraint forces (forces that maintain the restriction of motions between two bodies) are expressed by ODE - and other - as a solution of a linear optimization problem under constraint, in a similar way than for the rigid Bowden cable constraint in section 2.1.1.b.

A choice made by ODE, is the possibility to transgress this constraint, in order to gain some computation time. Other physics engines that use hard constraint resolution like ARBORIS [ISI], are much slower in comparison.

There are two parameters that drive how the constraints are managed [ODE] :

- Error Reduction Parameter : it is a value between 0 and 1. it drives the forces that are added to bodies,in order to align correctly, those which joint position have drifted apart due to numerical instability. 0 mean that the bodies can drift apart (no extra forces will be applied), and 1 means that the solver will try to correct the numerical drift in one single step. A value between 0.2 and 0.8 is advised by [ODE].

- Constraint Force Mixing : it is a value > 0 that indicates how much the solver can violate a constraint, by in some way, reducing proportionally to the CFM value the forces resulting from a given constraint.

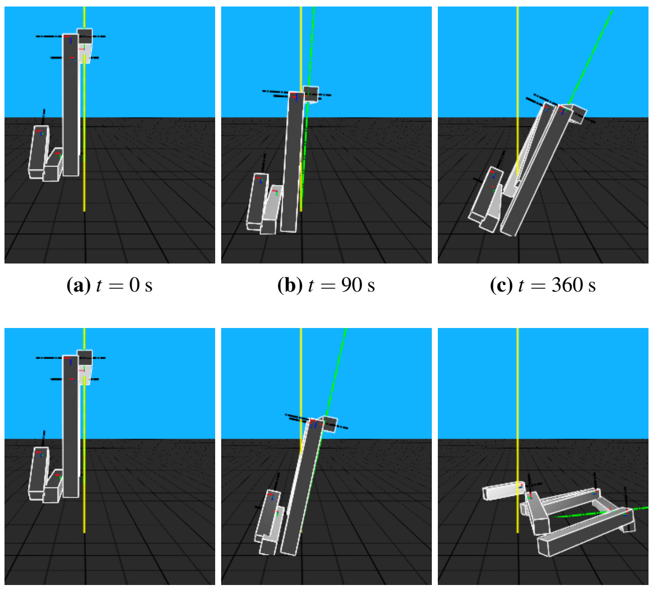
Then, in addition to reduce the computation time, these parameters could be used for the experienced user to modelize soft constraints. But a bad tuning could also lead to instabilities.

Therefore one can imagine, the fact that the increase of CFM solves the unrealistic behavior, is because the system is over-constrained. However it is a bad approach to release these superfluous constraints by increasing CFM :

- It denotes more a problem of conception of the ODE model, and the fault should rather be searched here.
- Under big external disturbances, the use of a high CFM also lead to unrealistic behavior of the legs (section 2.1.3.b)
- Closed loop must be carefully performed. If there is some initial misplacement of the bodies axis, it will cause unrealistic torque and force to be added, to maintain the constraints. And as the goal is to modify the model programmatically, usually it may have such errors.

Therefore, in this project it will be better to find out the constraint, that cause these unrealistic behavior.

2.1.4.c Finding the design problem of the model


(a) $t = 0$ s **(b)** $t = 90$ s **(c)** $t = 360$ s

(d) $t = 0$ s **(e)** $t = 0.5$ s **(f)** $t = 1$ s

Figure 2.5 – Test of two closed loop mechanism similar to the *Cheetah* 's leg mechanism. With only one closed loop (no closure between the piston and the ankle), the simulation seems to be physically realistic.

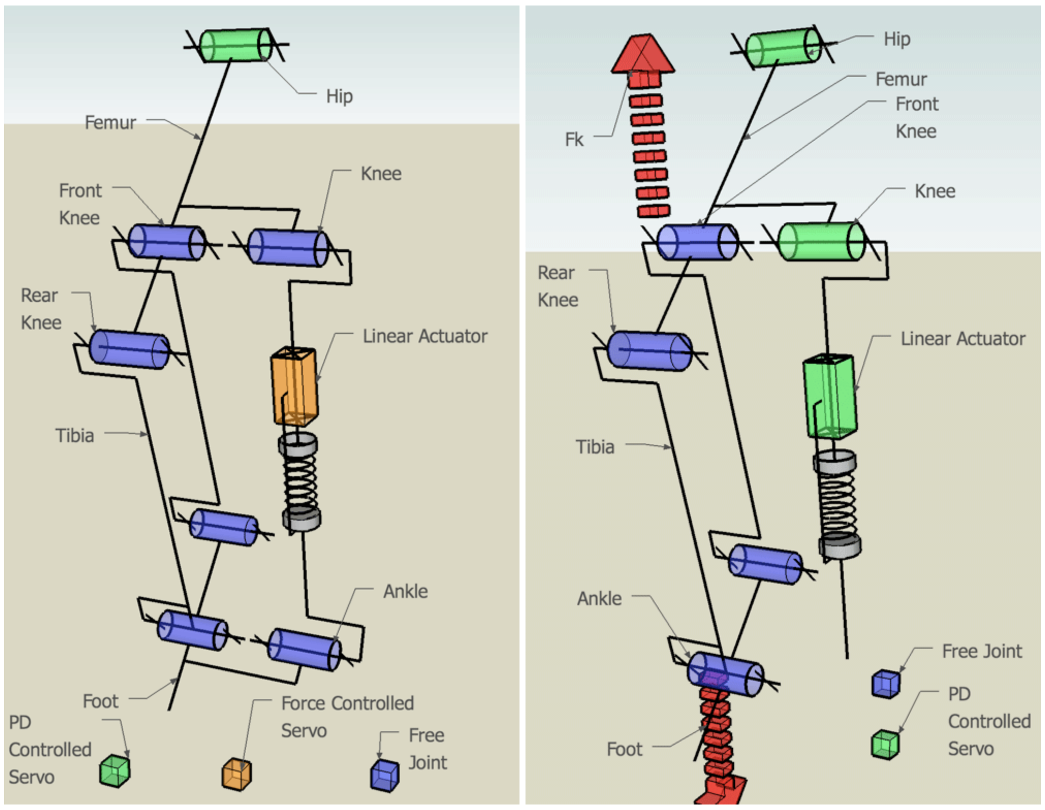
WEBOTS comes with several samples of a closed loop mechanism, and several come with more than two loops. All simulations seemed realistic, and used a standard CFM parameter. However None used closed loop that recovers each other, i.e. where several joints of the two loops have the same axis.

So a small model was made in WEBOTS , that is exactly the same mechanism of the *Cheetah* leg but with easily computable dimensions (a square mechanism, angle of 90 and 45 degrees) to avoid bad closure of the loop. All joints were tuned to be passives, to reduce the error of a bad command, and the loop closure was made by the same code than for the *Cheetah* legs. It leads to the following results : in the presence of a double intersected loop closure, the same bug appear in huge proportions.

It seems to be a restriction of ODE not to be able to solve such mechanisms. But as the resolution of this problem is part of the ODE internal, the easier solution is to open one of the two loop of the mechanism.

2.1.4.d Resolution of the design error

As the model of the knee actuator directly give us the output force of the system (composed of the spring, the Bowden cable, and the servo-motor), it is easiest to remove the closed loop between the "muscle" part and the ankle joint (see figure 2.6). Then, to drive the whole mechanism, the physics plug-in is used, to tell ODE to apply directly the computed force from the knee model, between the femur and foot bodies, aligned on the diagonal of the mechanism (figure 2.6b).



(a) 2 loop **(b)** 1 loop

Figure 2.6 – Old and new mechanical model of the leg in WEBOTS . To avoid the unrealistic behavior, the second closed loop is opened, and the forces computed with the model are directly applied to mechanical bodies in the physics plug-in. The other part are conserved and are driven by servo to follow their real position, to keep the accuracy of the *Cheetah* limb inertia.

However the knee model needs the contraction of the spring as input (section 2.1.3). For this purpose, a direct kinematic model of the leg is used. This model is also used to drive the WEBOTS servo-motors of the "muscle" part, in order to follow the position these bodies should have, in the case where the 2nd loop is closed.

One can notice that these parts aren't anymore useful for the modelization of the knee, and are only kept to maintain the accuracy of the leg inertia computation. If one want to neglect this inertia, these parts should be removed from the model.

2.2 Using the BIOROB Software framework.

The second task of the project was, to include the BIOROB [1] Software framework made by Jesse VAN DEN KIEBOOM [vdK09], to control, and perform offline learning of the Cheetah. What is presented here is just a brief overview, for a more in depth presentation please refer to the corresponding documentation .

After the presentation of the two main tool of this framework, an overview of the whole controller will be done.

2.2.1 LIBCPG-NETWORK an Object Oriented simulator of dynamical system.

The first software is the LIBCPG . As seen in section 1.1.3, CPG are often referred as dynamical system, and more precisely, as a network of coupled oscillators.

The LIBCPG has been conceived like a general, Object Oriented simulator for dynamical system, and is not limited to the only study of CPG : the user defines objects that contains properties, that are mathematical expression that can reference other properties of the same object. Data can be transmitted between object with the use of link. Some property can be marked as "integrated", in order to establish differential equation of 1st order for this property. Then by combining different object property and link, one can simulate any dynamical system with an Euler integration principle.

The library permits the definition of such dynamical system, in several way :

1. Directly with the C API.
2. Through the definition of an XML file that defines the dynamical system. It is then loaded and executed in a program with the C API.
3. With the use of a GUI, that help the creation and debugging of the XML file.

This library features also a way to make controller that easily drive a WEBOTS model with a CPG, as a property of an object could be automatically linked to a command value of a Servo in WEBOTS . However as in this case, the output values doesn't drive directly WEBOTS servomotor, but complex model are used that seems hard to code using the LIBCPG , one couldn't use this feature. But for this purpose, two tools are proposed (section 2.3.2), and a extension of the file job description (section 2.2.3), to facilitate the creation of generic controllers.

2.2.2 Optimization framework

As stated by its name, the purpose of this framework is to solve optimization task. but moreover, it is designed to distribute all the computation in parallel on several computer. Indeed this framework only implement *evolutionary algorithms*. All of these algorithms have in common to maintain a set of solutions where their fitness are estimated at each iteration. Therefore the parallelization of the algorithm is straight forward, as these estimations are independent from each other. When performing *offline learning* of a robot, this lead to huge decrease of the computation time : the fitness computation of one individual is time consuming as it is often the estimation of the behavior of a simulated robot for a certain duration.

This framework gives all the tools necessary to perform optimization on several computers. Moreover, a cluster of 45 computers is maintained at BIOROB , in addition of a token server to share fairly the available computational power over several end users. Each components is dedicated to one particular task :

[1] new name of BIRG

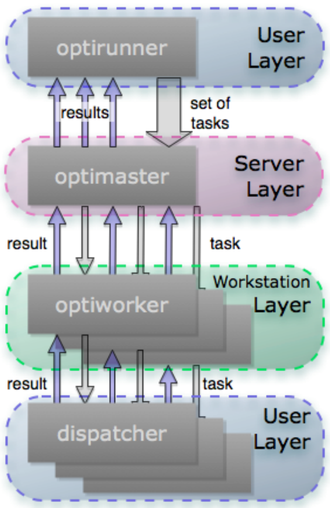


Figure 2.7 – Overview of the optimization framework. End user had to step in the optirunner program through an XML job definition file, and in writing the right dispatcher program for its application.

- The OPTIRUNNER program, which is used by the end user to launch optimization task, defined by an XML job file.
- The OPTIMASTER program, which is run on a server. Each OPTIRUNNER that are performing optimization are connected to it, and send it tasks. Then the OPTIMASTER distribute them to available workstation.
- Each workstation are running OPTIWORKER programs. They are connected to the OPTIMASTER and receive one task description at a time. Then OPTIWORKER prepare the environment and launches the DISPATCHER .
- A DISPATCHER program is responsible of the computation of a task. In evolutionary algorithms, it is the fitness computation of an individual. Then it send back the result to the OPTIWORKER , who will transmit it to the OPTIMASTER and finally the later will give back the result to the OPTIRUNNER .

The end user had to :

- manage the OPTIRUNNER program, by providing the job definition file, that define the type of optimization algorithm and it settings, the parameters to optimize with their boundaries, and which dispatcher to use.
- provide the right dispatcher that will perform a given task.

Here a WEBOTS DISPATCHER is used, that load simulation of *Cheetah* . The robot is driven by a CPG as described in section 3.1 and implemented with the LIBCPG . However if a small changes is made to the optimization process, like the parameter to explore, the WEBOTS controller might also need some small changes which is cumbersome. So an extension of the job configuration file structure is proposed.

2.2.3 Extension of the job file

The goal of this extension, is to made the controller aware of which optimization parameter and settings are linked to which CPG parameter.

For this purpose the definition of the parameter node of the job description file is extended [vdK09] from :

```
<parameter name="xxxxx" boundaries="yyyyyyy"/>
```

to :

```
<parameter name="parameter_name" boundaries="yyyyyyy">
  <group_of_cpg_parameter>
    <item object="object_name" property="property_name"/>
    <item object="object_name" property="property_name"/>
     ...
    <item object="object_name" property="property_name"/>
  </group_of_cpg_parameter>
</parameter>
```

where `object_name` and `property_name` designate the property of one object, the parameter `parameter_name` is linked to.

The possibility to link LIBCPG properties to setting is also added by adding the `<settings>` node after the `<parameters>` one :

```
<job>
  <optimizer>
    <parameters>
      <parameter/>
       ...
    </parameters>
    <settings>
      <setting name="setting_name">
        <group_of_cpg_property/>
      </setting>
       ...
    </settings>
  </optimizer>
  <disatcher>
   <setting/>
     ...
  </dispatcher>
</job>
```

and then each `<setting>` defined under `<dispatcher>` whose name is `setting_name` will be linked to the corresponding LIBCPG properties.

2.3 Software organization

2.3.1 General

The software that controls the optimization of the *Cheetah* relies on three main parts, a WEBOTS controller a physics plug-in, and the WORLDMAKER executable.

For the WEBOTS controller the goal was to develop a generic controller for the *Cheetah* , where everything is controlled by configuration files. With this approach, several optimizations could be performed in the same time. This would not be possible, or easily manageable, if the executable needs to be recompiled, at each change of parameter.

The role of the WORLDMAKER is to programmatically change the geometric configuration of the *Cheetah* .

The whole software was coded in C++, because it is an industry standard Object Oriented language. For complex task like GUI, the QT library has been used, which is an open source library under LGPL license. However the whole framework can be at runtime independent from QT and can be used without it (with the absence of a GUI, for example during optimization). However all the Makefile generation is handled by the QMAKE utility which is part of the QT development tool.

2.3.2 *Cheetah* Controller

The controller can be launched in two mode :

- `OPTIMIZATION` mode : it act as a back end for a `<webots::Dipatcher>` . It can perform different types of optimization through the use of `<OptimizationLoop>` .
- `RUN` mode, when compiled with QT libraries : it launches a GUI that controls for example the `<CpgNetwork>` properties, or monitors some values of the robot.

The launch of the controller need some arguments :

- `-j/-job` (mandatory) : path to the XML job configuration file
- `-c/-cpg` (mandatory) : path to the XML cpg configuration file
- `-o/-optimization` : launches the controller in optimization mode (will try to get a request from a dispatcher, if none will quit)
- `-r/-redirect` : will redirect `stdout` (standard output) on a temporary unique file
- `-n/-none` : will launch the controller in `VOID` mode, all actuator will stay at their zero position.

While performing `OPTIMIZATION` mode, the controller can use the additional dispatcher settings :

- "**`Simulation::Duration`**" : who in most cases defines the duration of the estimation of one individual. It depends of the `<OptimizationLoop>` used (section 2.3.2.a).
- "**`ValidityTest`**" which is a string of the form :

  ```
  Test1:Args11:Args12:....|Test2:Arg21:Arg22:...|.....
  ```

  where `"TestX"` designate a `<ValidityTest>` (section 2.3.2.b), and `"ArgsXY"` its arguments. The number of arguments depend on the «> ValidityTest>
- "**`OptimizationType`**" which designate the `<OptimizationLoop>` to use.

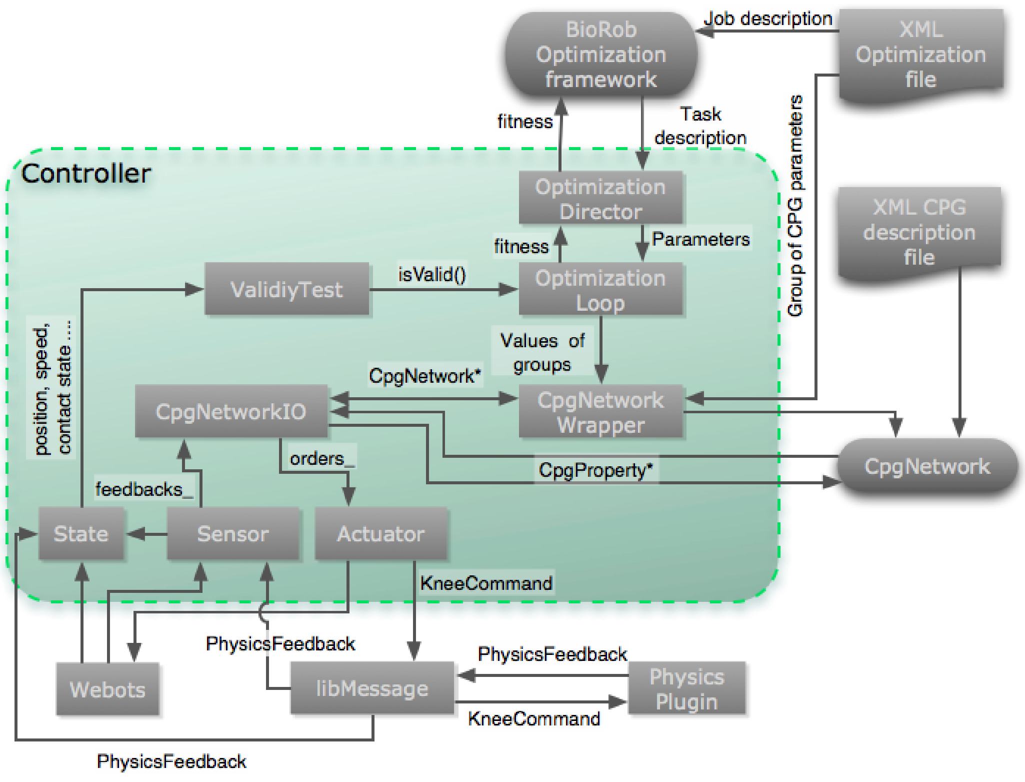


Figure 2.8 – General organization of the controller.

2.3.2.a `<OptimizationLoop>` and `<OptimizationDirector>` classes

An `<OptimizationLoop>` is responsible for providing the loop of a certain type of optimization, and for defining what will be the response of the task. The `<OptimizationDirector>` take charge of getting the task description from the dispatcher, and of selecting the right OptimizationLoop.

The `<OptimizationLoop>` also check the validity of the *Cheetah* `<State>`, through the `<ValidityTester>`. If the state become invalid, the simulation is stopped, and the corresponding fitness is returned without any further computation.

Currently two `<OptimizationLoop>` are implemented :

- `<DistanceLoop>` : the simulation will last for "**`Simulation::Duration`**" time. then the traveled distance along the x direction (clamped to a minimum of zero) is returned. The corresponding value of "**`OptimizationType`**" is `"distance"`, and its the default value.
- `<UniformSpeedLoop>` : the simulation is run several time, with a user defined parameter being changed. Then the weighted mean (weight are also defined by the user) of the fitnesses is returned. The corresponding value of "**`OptimizationType`**" is `"uniform"`.

2.3.2.b `<ValidityTester>` class

The `<ValidityTester>` tests if the `<State>` of the Cheetah is valid through the use of `<ValidityTest>`. The enabling/disabling and configuration of `<ValidityTest>` is done with the "**`ValidityTest`**" setting. There are several ValidityTest implemented know :

- `<SpeedTest>` : tests if the speed over the last second of the cheetah is too high. Its setting name is `"Speed"` and it takes one numerical arguments.
- `<DistanceTest>` : tests if the distance traveled by the robot is too high. Its setting name is `"Distance"` and it takes one numerical arguments.

- `<HeightTest>` : tests if the geometric center of *Cheetah* is outside defined bounds. Its setting name is `"Height"` and it takes two numerical arguments. First argument is the upper bound and second the lower.
- `<RenversedTest>` : tests if the robot had a pitch or roll rotation of more than $\frac{\pi}{2}$ from its starting position. Its setting name is `"Renversed"` and it takes no argument.
- `<ContactTest>` : test if the trunk of the *Cheetah* is touching the ground. Its setting name is `"Contact"` and it takes no argument.

2.3.2.c **`<Actuator>` and `<Sensor>`**

These two classes are respectively abstraction of the actuator output and sensor input of *Cheetah* . `<Actuator>` permits the implementation of more complex model then P controlled `<webots::Servo>` . There are several `<Actuator>` implemented :

- `<SimpleActuator>` is the same as a `<webots::Servo>` .
- `<FreeActuator>` free joint. Used to measure the corresponding joint angle.
- `<KneeActuator>` modelize the knee actuator as in section 2.1.
- `<ToeActuator>` modelize the toe actuator as a passive spring.

  There is two implemented `<Sensor>` classes. Both of them use basic signal processing on the WEBOTS data. The signal is filtered by a low pass filter, and hysteresis threshold. Then they both return value that are either 0.0 (no contact) or 1.0 (contact).

  1. `<TouchSensor>` that use a standard `<webots::TouchSensor>` as input for the signal.
  2. `<TouchSensor3D>` that use direct measurement of the Ground Reaction Forces (GRFs) from the Physics plug-in.

2.3.2.d **`<CpgNetworkIO>` and `<CpgNetworkWrapper>`**

The `<CpgNetworkIO>` class aims to manage the input and output of the `<CpgNetwork>` . Then it gives the orders to the `<Actuator>` and receives feedback from the `<sensor>` .

The `<CpgNetworkWrapper>` is responsible for extracting the parameters of the job file that are associated with `<CpgNetwork>` properties. Then it give an interface to easily access these properties.

2.3.3 Physics Plug-in

The physics plug-in is a dynamic library that is responsible for :

- Closing the mechanical loop of the leg at the ODE world initialization.
- Apply the forces computed from the model (section 2) and given by `<KneeActuator>` .
- extracting, at the end of each step, the GRFs, and send them back to `<TouchSensor3D>` .
- detecting if the trunk of the *Cheetah* is in contact with the ground and send back the information to the Controller
- Display visual hints, like the knee forces, and the Ground reaction forces.
- Enable and disable a `<Deambulator>` , that restrict the motion of the *Cheetah* to the sagital plane.

For the communication with the controller, the previously developed library LIBMESSAGE was used [Tul09, May09].

2.3.4 WorldMaker

The WORLDMAKER is responsible for the automatic generation of WEBOTS model of the *Cheetah* with the use of the "**`worldBuilderPath`**" . This option of optimization feature permits to test different leg configuration.

The WORLDMAKER can be launched from the command line with the `-i/-interactif` option. Then it will ask for the value that define the leg configuration.

If no option are specified, it will follow the expected comportment of WORLFBUILDER : it will

- read the standard input for a `<taskDescription>` ,
- generate the world file in a unique-named, temporary file,
- send back the location of this file on standard output.

In this mode, the job description file must also provide the additional dispatcher settings :

- "**`Cheetah::Fore::LegDescription`**" , a string that describe the geometry of the fore-limbs , which is of the form `"Length|FemurCoeff|TibiaCoeff|FootCoeff|DeltaTibia"` , where :
  - `"Length"` is the total length of the leg.
  - `"FemurCoeff"` is the λ_3 parameter of figure 1.4.
  - `"TibiaCoeff"` is the λ_2 parameter of figure 1.4.
  - `"FootCoeff"` is the λ_1 parameter of figure 1.4.
  - `"DeltaTibia"` is the λ_p parameter of figure 1.4.
- "**`Cheetah::Hind::LegDescription`**" the same parameters as the previous one, but for hindlimbs.
- "**`Cheetah::Controller::Arguments`**" , which is the standard arguments to pass to the *Cheetah* controller.

The WORLDMAKER has been made upon the LIBWBT , a library who add be designed for the writting of WEBOTS world file (.wbt).

Part 3

Implementation of the locomotion control

3.1 Central Pattern Generator defintion and extension

For the control of the *Cheetah* a Central Pattern Generator, previously designed by Ludovic Righetti[RI08] has been used. This CPG is able to generate up to four gait found in quadrupedal (walk, trot, pace and bound), and features sensory feedback, that synchronizes the command with the state of the foot (stance and swing phases). After a presentation of this CPG, a modification is proposed to introduce the leg retraction strategy.

3.1.1 A Central Pattern Generator for quadrupedal walking

3.1.1.a General presentation

The CPG proposed by [RI08], is based on a modified Hopf oscillator. There are four neurons which have a state (x_i, y_i)

$$\begin{aligned} \dot{x}_i &= \alpha(\mu - r^2)x_i - \omega_i y_i && (3.1) \\ \dot{y}_i &= \beta(\mu - r^2)y_i + \omega_i x_i + \sum_{j \neq i} k_{ij} y_j && (3.2) \\ \omega_i &= \frac{\omega_{st}}{1+e^{by}} + \frac{\omega_{sw}}{1+e^{-by}} && (3.3) \end{aligned}$$

According to [RI08], this oscillator exhibit a limit cycle (see figure 3.1) , which is the circle of radius $\sqrt{\mu}$ in the phase space (x, y). [RI08] shows that the choose of the appropriate coupling matrix $(k_{i,j})$ for four oscillators, leads them to phase lock. The resulting difference of phase between the oscillator depends on symmetries between block of the chosen matrix (see figure 3.2 as an example). [RI08] also gives 4 matrices for each of the following gait : walk, trot pace and bound.

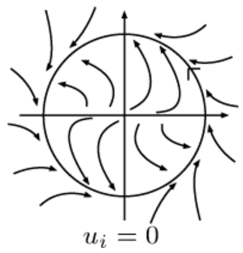
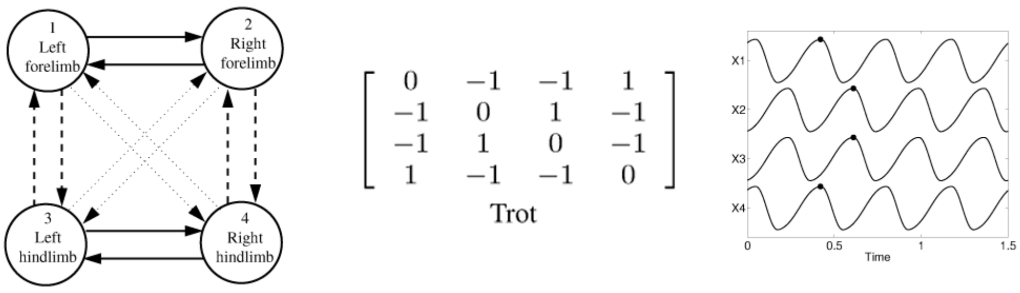


Figure 3.1 – Representation of the attraction field of one oscillator. from [RI08]



$$\begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

Figure 3.2 – Trot coupling for the CPG. taken from [RI08]

One nice property of this oscillator, is that the phase velocity ω_i is not constant, but is "smoothly" separated into two phases :

- the stance phase for $y < 0$. In this case the ω_{st} term is prevailing in (3.3).
- the swing phase for $y > 0$. In this case the ω_{sw} term is prevailing in (3.3).

However this introduces some drawback. One can remark experimentally, that even the average of the two angular frequencies ω_{st} and ω_{sw} is kept constant, it doesn't set the frequency of the final system. Indeed the relation between angular frequency and the frequency f of the limit cycle is not straight forward : if the system (3.1-3.3) is transformed in polar coordinates (r, θ), $T = \frac{1}{f}$ becomes the result of the following integral :

$$\frac{1}{f} = T = \int_0^{2\pi} \frac{1}{\frac{\omega_{st}}{1+e^{b\sqrt{\mu}\sin(\theta)}} + \frac{\omega_{sw}}{1+e^{-b\sqrt{\mu}\sin(\theta)}}} d\theta \tag{3.4}$$

Which is not easily computable.
This have two consequences :

- the angular frequencies should be the same for each of our oscillators : oscillator with different frequency will converge slower (or even not converge) to a global limit cycle. The final frequency is also not easily predictable.
- It is hard to relate the duty factor expected for one leg, to the value of the angular frequency.

An approximation of the integral (3.4) is proposed, in order to relate the duty factor to the angular frequencies and frequency.

3.1.1.b Relations between frequency, angular frequencies and duty ratio

The idea behind the equation (3.3) is to use the hyperbolic tangent to "smoothly" switch between the two angular frequencies. Then for high values of b, the switch is almost instantaneous between the two values. In the other case for small values of b, the value of ω is almost constant.

For this two cases, the expressions of the angular frequencies depending on the main frequency of the oscillator and its duty ratio is presented :

- Case $b >> 1$ In this case, the integral (3.4) is divided in two parts :

$$\begin{aligned} T &= \int_0^{\pi} \underbrace{\frac{1}{\frac{\omega_{st}}{1+e^{b\sqrt{\mu}\sin(\theta)}} + \frac{\omega_{sw}}{1+e^{-b\sqrt{\mu}\sin(\theta)}}}}_{=\frac{1}{\omega_{sw}}} d\theta + \int_{\pi}^{2\pi} \underbrace{\frac{1}{\frac{\omega_{st}}{1+e^{b\sqrt{\mu}\sin(\theta)}} + \frac{\omega_{sw}}{1+e^{-b\sqrt{\mu}\sin(\theta)}}}}_{=\frac{1}{\omega_{st}}} d\theta \\ &= \frac{\pi}{\omega_{sw}} + \frac{\pi}{\omega_{st}} \\ T &= \pi \frac{\omega_{st} + \omega_{sw}}{\omega_{st}\,\omega_{sw}} \end{aligned} \tag{3.5}$$

One can also identify in (3.5) the half period corresponding to stance phase $T_{st} = \frac{\pi}{\omega_{st}}$ and the half period corresponding to the swing phase $T_{sw} = \frac{\pi}{\omega_{sw}}$. Therefore the definition of the duty ratio d becomes :

$$d = \frac{T_{st}}{T} = \frac{\omega_{sw}}{\omega_{sw} + \omega_{st}} \tag{3.6}$$

This bring up the expressions of the angular frequencies depending on f and d :

$$\omega_{st} = \frac{\pi}{d} f \tag{3.7}$$

$$\omega_{sw} = \frac{\pi}{1-d} f \tag{3.8}$$

- case $b <<< 1$ In this case there is almost any change between the two phases and the angular frequency is $\frac{\omega_{st}+\omega_{sw}}{2}$, so the integral (3.4) becomes :

$$T = \frac{2\pi}{\frac{\omega_{st}+\omega_{sw}}{2}} = \frac{4\pi}{\omega_{st} + \omega_{sw}} \tag{3.9}$$

But here the definition of the duty ratio have less meaning because there is no change of the angular frequency, then the real duty ratio in this limit case is constant to 0.5. Therefore a parameter called duty ratio with the same definition (3.6), is kept, that bring up the following expression for the angular frequencies :

$$\omega_{st} = 4\pi.(1-d).f \tag{3.10}$$

$$\omega_{sw} = 4\pi.d.f \tag{3.11}$$

- Validity of the two cases Using numerical integration, it has been proved that that for $b = 1$ the second case is within 10% of relative error, and for $b = 100$ the first case is within 5% of relative error, in the case $f = 1$Hz and $\mu = 1$.

3.1.1.c Use of sensory feedback

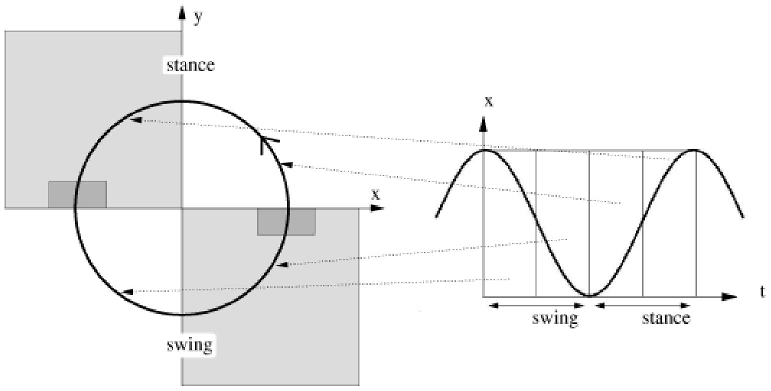


Figure 3.3 – Enabling region for the sensory feedback. Fast transition should only take place in light gray regions, Stop transition only in dark grey ones. taken from [RI08]

[RI08] propose a way to add sensory feedback to his CPG. As seen before, the phase state is separated in two phases : the stance ($y < 0$) and the swing ($y > 0$) phase. The idea is to match up this two phase with to their definition upon the state of the foot (foot touches the ground during the stance phase, foot is on the air during the swing phase).

Therefore the following events could happen :

- Have a fast transition between the two phases of the oscillator:
  - while the oscillator is in the stance phase, and the foot leaves the ground.
  - while the oscillator is in the swing phase, and the foot touches the ground.
- Stop the transition of the oscillator :
  - from stance to swing phase, while there is still a sufficient weight supported by the foot.
  - from swing to stance phase, while the foot still doesn't touch the ground.

For this purpose [RI08] propose to add the following control input u_i to equation (3.2) :

$$u_i = \begin{cases} -\operatorname{sign}(y_i)F & \text{for fast transition} \\ -\omega_i x_i - \sum_{j\neq i} k_{i,j} y_j & \text{for stop transition} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The sensory feedback should either not be enabled for every time. [RI08] propose to activate the sensory feedback only for subspace of the phase space (see figure 3.3)

3.1.1.d Use with *Cheetah*

For controlling the *Cheetah* , this CPG is used. However, with some trial test it appears that the global dynamical system converge much faster with the same μ for the four oscillators. Therefore, the possibility to have different value for the front and hindlimb is important, the following values are added for the command.

$$\varphi_i^{hip} = a_i * x_i + h_i \tag{3.13}$$

$$\dot{a}_i = k_a \left(a_i - a_i^{des}\right) \tag{3.14}$$

$$\dot{h}_i = k_h \left(h_i - h_i^{des}\right) \tag{3.15}$$

Where a_i is the the amplitude of the oscillator and h_i its offset. The two last equation provide only smooth changes of these parameters, the user should only change the values a_i^{des} and h_i^{des}

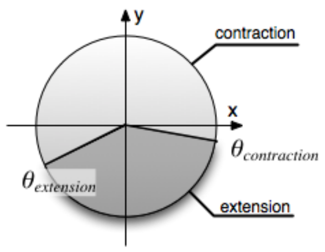


Figure 3.4 – Threshold for extension/contraction of the knee joint.$\Theta_{extension}$ and $\Theta_{contraction}$ defines the two regions where the leg should be either contracted ($\varphi_i^{knee} = \lambda_i > 0$) or extended ($\varphi_i^{knee} = 0$).

There are two active joints per leg. As stated before, the most proximal joint (here the hip joint) must brought the most considerable contribution to the locomotion [FB06]. So φ_i becomes output as the command of the hip joint. The knee joint, according to the principle of [WHI$^+$00], as the most active distal joint must only drives the foot clearance. Therefore the joint must only jump from the state "extended" ($\varphi_i^{knee} = 0$) in the stance phase to the state "contracted" ($\varphi_i^{knee} = \lambda_i$) in the swing phase. This behavior is parameterized by two thresholds (figure 3.4) :

$$\varphi_i^{knee} = \begin{cases} 0 & \text{if } \Theta_{extension} \leq \theta_i < \Theta_{contraction} \\ \lambda_i & \text{else} \end{cases} \tag{3.16}$$

Here the choice of extending at maximum the leg at each gait cycle has been made. The height of the CoM can easily be controlled, by adding a new parameter ν_i, and so not extend the leg as it maximum ($\varphi_i^{knee} = 0$. Once again, the following equations are introduce, to provide only smooth changes of the two level parameters :

$$\begin{aligned} \dot{\lambda}_i &= k_\lambda \left(\lambda_i - \lambda_i^{des}\right) && (3.17) \\ \dot{\nu}_i &= k_\nu \left(\nu_i - \nu_i^{des}\right) && (3.18) \end{aligned}$$

3.1.2 Addition of the leg retraction principle

The next goal is to implement the leg retraction principle into the CPG. More formally, we want that at touchdown, the gesture of retraction has already begun, i.e. : that both $x < 1$ and $\dot{x} > 0$. Ideally we also want to parameterize this amount of retraction, by for example specifying $\dot{x}(t_{touchdown})$. In [SGH03], this parameter is the constant rotational speed of the leg.

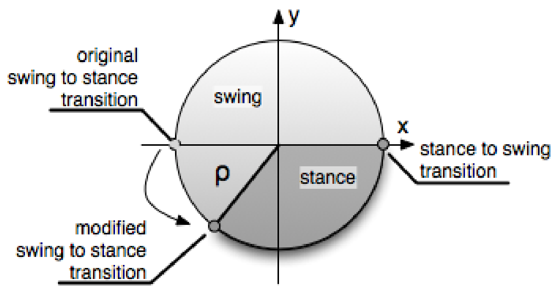

y
original swing to stance transition
swing
x
stance to swing transition
ρ
stance
modified swing to stance transition

Figure 3.5 – Implementation of the leg retraction principle. The swing to stance transition point should be moved of an angle $\rho \geq 0$.

However, implementing the leg retraction principle this way is rather difficult. In [SGH03] the angle (to the vertical) of the leg at APEX is precisely controlled, and the rotational speed is constant. The first predicate is difficult to implement in a robot, and the second one isn't reachable within the servo limits. *Cheetah* is intended to be really fast, at the limit of the servo specification. So it couldn't be expected, the servo not to have a delay to reach some specific speed.

Further more, the convergence properties of this CPG are rather complicated, and defining new limit cycle would be difficult. Then it may be simpler, in a first test, just to transform the output of the oscillators. Then the new behavior won't interfere with the properties of the dynamical system. The trajectories generated by the new output should also be changed, as little as possible, from the original one. Therefore the limit cycle (unit circle) must not be changed.

In order to fulfill the first requirements, one can have the idea to move along the unit circle the transition point between the swing and stance phase from an angle $\rho \geq 0$ (see figure 3.5). Therefore, the angular rate of the hip at touch down will not be null anymore, but will be $sin(\rho)$. Finally in order to fulfill the last requirements, a transformation of the plan, in polar coordinates could be proposed :

$$\begin{aligned} \mathbf{R}^+ \times [-\pi, \pi] \rightarrow & \quad \mathbf{R}^+ \times [-\pi, \pi] \\ \begin{bmatrix} r \\ \theta \end{bmatrix} \mapsto & \quad \begin{bmatrix} r \\ f_\rho(\theta) \end{bmatrix} \end{aligned} \qquad (3.19)$$

provided that f_ρ is a bijection of $[-\pi, \pi]$ and that f_ρ is continuous, the unit circle will be left unchanged. The following properties for f_ρ are also required :

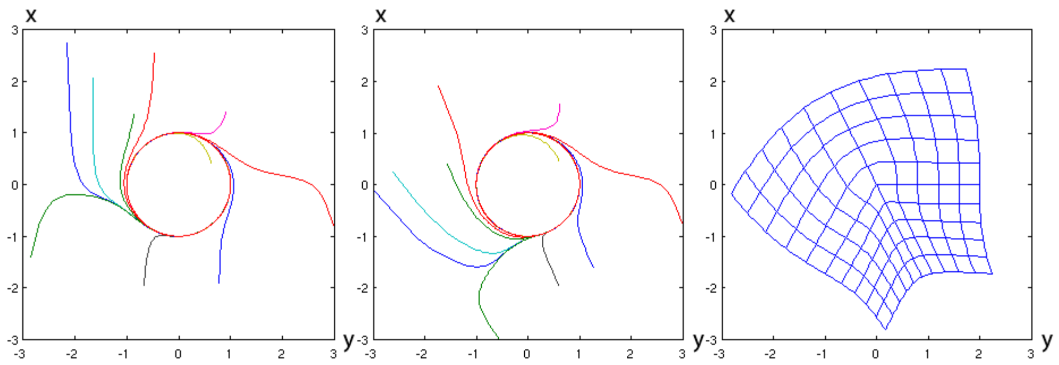
$$\begin{aligned} f_\rho(0) &= 0 && (3.20)\\ f_\rho(-\pi) &= \rho-\pi && (3.21)\\ f_\rho(\pi) &= \rho+\pi && (3.22)\\ f'_\rho(0) &= 1 && (3.23)\\ f'_\rho(-\pi) &= f'_\rho(\pi) && (3.24) \end{aligned}$$

- Equation (3.20) avoids any displacement of the stance to swing transition point.
- Equations (3.21) and (3.22) move the stance to swing transition point and are a necessary condition, f_ρ to be a bijection.
- Equations (3.23) and (3.24) insure a smoothness of the solution

One can found a unique 4th order polynom that satisfies the conditions (3.20-3.24) :

$$f_\rho(\theta) = -\rho.\left(\frac{\theta}{\pi}\right)^4 + 2.\rho.\left(\frac{\theta}{\pi}\right)^2 + \theta \qquad (3.25)$$

The result of one of these transformations is seen in figure 3.6.



(a) Original trajectories **(b)** Images of the trajectories **(c)** Transform of a uniform mesh

Figure 3.6 – Transformation of a few trajectories with the function f_ρ, for $\rho = \frac{\pi}{4}$.

One should also notice that all the conditions are not met, f_ρ to be a bijection. However if $\rho \in \left[0, \frac{\pi}{2}\right]$, this is always the case.

3.2 Definition of the *offline learning* process.

In order to perform offline learning of *Cheetah* gaits, one have to choose, an optimization method, a fitness function, and the parameter space to explore.

As the former directly mostly influence the speed of converge of the learning, and its capacity to learn optimal gait, the latters have an impact on the quality of the obtained gaits.

3.2.1 Particle Swarm Optimization

For the optimization, a Particle Swarm Optimization (PSO) algorithm in its canonical form, given by [CK02], is applied. PSO operates on a set of particle, using swarm/flocking behavior. Each particle remembers it best fitness position p_b. They also have a set of neighbors, and knows the best position for all the particles in this neighborhood p_n. Then at each iteration the position of the particle is updated, in order to oscillate between its current position, p_b and p_n. [CK02] gives then the correct update rules for the position p_i that insure convergence of the algorithm to (at least) a local minima :

$$\begin{aligned} v_i^{t+1} &= \chi\left(v_i^t + R_1\varphi_1\left(p_b - p_i^t\right) + R_2\varphi_2\left(p_n - p_i^t\right)\right) && (3.26)\\ p_i^{t+1} &= p_i^t + v_i^{t+1} && (3.27)\\ \varphi &= \varphi_1 + \varphi_2 > 4 && (3.28)\\ \chi &= \frac{2\kappa}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|} && (3.29)\\ \kappa &\in [0,1] && (3.30) \end{aligned}$$

It is commonly chosen $\varphi = 4.1$ and $\chi = 0.72984$

One last two parameters to choose are the number of particle, and the number of iteration. The algorithm scales linearly against this parameter, so a tradeoff between efficiency (avoiding to fall in a local minima), and speed of convergence must be found.

- The bigger the number of particles, the lower is the probability to fall into a local extrema, as more parts of the optimization space are explored by particles.
- The bigger the number of iterations the finest the determination of the extrema is.

However, with the use of distributed computation on N computers, one should notice that we can at least give a value of N for the number of particle. Indeed the particles positions can only be updated when all the other particles fitnesses are up-to-date. Then if we have a population of $N-I$ computers, I computers will never have simulation to perform. Moreover, computation of a fitness could be quicker for particular particles. Hence it is wise to use a number of particles bigger than the number of available computers,in order not to waste computation power.

3.2.2 Fitness Evaluation definition.

For the first trials, a simple fitness has been chosen : the mean speed along the front direction, over a time T_s. To avoid the benefit of starting point instability, and to insure that a limit cycle of the robot gait is measured, T_s is chosen to cover several gait cycle.

$$F_1 = \frac{1}{T_s}.\left(p(T_s) - p(0)\right).\begin{bmatrix}1\\0\\0\end{bmatrix}^T \qquad (3.31)$$

where $p(t)$ is the position of the robot at time t

However first experiments with this fitness function gave bad result. Convergence was rather slow, and learned gait used some strange strategies, like laying the back part of the *Cheetah* on the ground, letting the hindlimb drag on the floor, and using only the frontlimbs to perform locomotion.

In order to avoid these bad strategies, a number of validity condition of the gait have been settled :

- The height of the *Cheetah* must be in correct bound, in order to avoid fall.
- The orientation of the cheetah must rotate of an angle of more than $\frac{\pi}{2}$ in the front or sagital plane, in order to avoid flip on the side or the front.
- The speed must not exceed a maxima, to avoid the use of the numerical instability of the physics engine.

Two methods can be used when such a condition is invalidated, to compute the fitness of the particle position:

- *Zero* invalidation mode : give a fitness of 0.
- *Clamp* invalidation mode : give a value of $\frac{1}{T_s}.(p(t_i)-p(0)).\begin{bmatrix}1\\0\\0\end{bmatrix}^T$ with $p(t_i)$ is the time where the first of the condition has been invalidated.

The two methods methods are compared in part 4.

Although this fitness function is simple, it is known to give over optimized gaits, where a slight changes of parameter lead to an instable gait.

One can also use the specific resistance $\varepsilon(\dot{p})$ of the gait [TPPB00], a increasingly assumed measure of the energy efficiency. The associated fitness is :

$$F_2(p) = \frac{1}{\varepsilon(\dot{p})} = \frac{MgF_1(p)}{P} \tag{3.32}$$

Where M is the mass of the robot, g the gravitational acceleration, and P the power consumption of the robot.

3.2.3 Choice of the Open Parameters

The last things is to determine the opened parameters for the optimization. They are described in table 3.1. They result of the following choice :

- A clear distinction has been made between forelimb and hindlimb, has the repartition of the weight is not equilibrated in the sagital plane. Therefore the system needs the possibility to correct this disequilibrium.
- However following the discussion of section 3.1.1.b, the duty ratio is the same for the four limbs, in order to kept a rapid convergence of the CPG.
- All parameters, with the exception of the frequency, are left open.
- All boundaries are kept as big as possible, in a first trial.

Name	**Applies to**	**Boundaries**
Hip::duty	(d_1, d_2, d_3, d_4)	$[0.05, 0.95]$
Fore::Hip::amplitude	(a_1, a_2)	$[0.0, 2.0]$
Fore::Hip::offset	(h_1, h_2)	$[-1.3, 0.9]$
Fore::Hip::refraction	(ρ_1, ρ_2)	$\left[0.0, \frac{\pi}{2}\right]$
Hind::Hip::amplitude	(a_3, a_4)	$[0.0, 2.0]$
Hind::Hip::offset	(h_3, h_4)	$[-1.3, 0.9]$
Hind::Hip::refraction	(ρ_3, ρ_4)	$\left[0.0, \frac{\pi}{2}\right]$
Fore::Knee::amplitude	(λ_1, λ_2)	$\left[0.0, \frac{\pi}{2}\right]$
Fore::Knee::ThExtension	$\left(\Theta_1^{extension}, \Theta_2^{extension}\right)$	$\left[\frac{\pi}{2}, \frac{3\pi}{2}\right]$
Fore::Knee::ThContraction	$\left(\Theta_1^{contraction}, \Theta_2^{contraction}\right)$	$\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$
Hind::Knee::amplitude	(λ_3, λ_4)	$\left[0.0, \frac{\pi}{2}\right]$
Hind::Knee::ThExtension	$\left(\Theta_3^{extension}, \Theta_4^{extension}\right)$	$\left[\frac{\pi}{2}, \frac{3\pi}{2}\right]$
Hind::Knee::ThContraction	$\left(\Theta_3^{contraction}, \Theta_4^{contraction}\right)$	$\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$

Table 3.1 – Open parameters of the optimization

Part 4

Execution of the learning process

Using the experimental setup described in part 3, the learning of a Trot gait has been tried several time, in order to find a good basis to test the influence of the proposed improvement in section 1.3. In this perspective, it had been chosen, not to include sensory feedback, and to use value for the spring constant of the leg that are close to the one utilized for the previous prototypes. However, as a direct observation of experiment made from the prototype, it has been chosen to fix the length of the fore and hind limbs to be the same.

After some feedback on the optimizer performance, a study of the learned gait is presented. For the data presented in this section the notations are depicted in figure 4.1. The study also concentrates on the sagital plane.

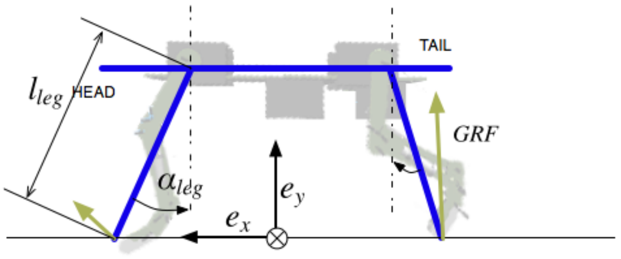


Figure 4.1 – Notations Convention for data. e_x is directed to the front, e_y to to the top, α_{leg} is positive when the leg is oriented toward the head.

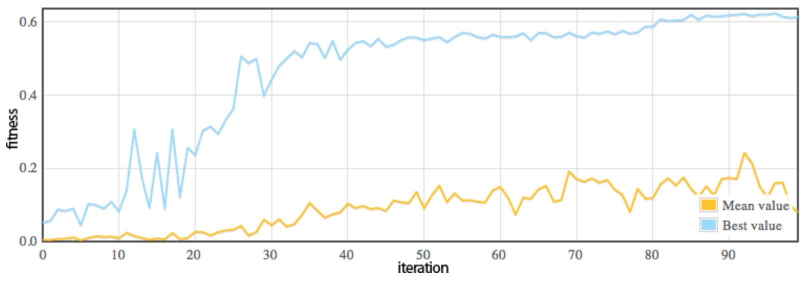
4.1 PSO parameter tuning

First trials have shown that the PSO parameters should be tuned correctly, to assure convergence to a global maximum of the fitness. Indeed, with badly tuned parameters, unoptimized gait were obtained one out of three run.

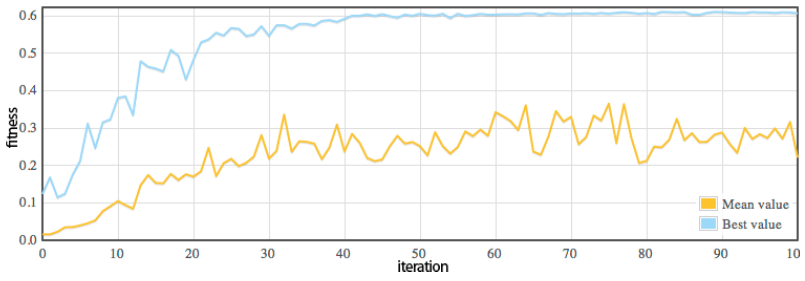
In every optimization it seems that a high number of particle is recommended (60 particle for 45 computers), and this doesn't decrease the speed of the algorithm too much, as explained in section 3.2.1. It also seems that in most of the cases, every optimization has converged after 100 iterations (see figure 4.2). However the right choice for the invalidation mode (section 3.2.2) can be crucial, and a way to increase the exploration of the space has been tested.

4.1.1 *Clamp* and *Zero* invalidation mode comparison

At first sight, it seems that it should be better to use the *Zero* invalidation mode whenever the robot falls in an invalid state. But after some trials, the learned gaits often used tricks like, let the back of the robot lying on the ground, the hindlimb limping on the ground, inactive, and using only the forelimb for propulsion.



(a) ZERO



(b) CLAMP

Figure 4.2 – Comparison of the fitness landscape for two different optimizations, one that use a zero fitness when invalidated, the second a clamped value (see section 3.2.2). Convergence in the second case is almost twice faster.

In order to avoid such situations, more invalidation criterion has been included, and it resulted, that the optimization had more difficulty to converge to a global minima. However if *Clamp* invalidation mode is used, this is less the case (see figure 4.2), and the latter is almost twice faster to converge. One explanation may arise by the fact that the mechanical system of the *Cheetah* is not stable. Therefore at the beginning, most of the particle won't generate gait that will even have a few stride period in a limit cycle. This assumption is supported by the fact that in *zero* invalidation mode, mean fitness of the particle is almost zero.

Then if all the fitness of these particles are settled to zero, the algorithm will concentrate all its particles to the non zero-place, and there is lot of chances that they are around local minima rather than a global one.

On the other hand if a particle as a small but non-zero value for the particle falling in a invalid state, it conserve some information on the gait generated, and the algorithm will have more chance to explore neighborhood of this region. And has the jump between a good and bad region is really small in comparison to the boundaries size, one can assume that it avoids the PSO to pass on a local minima.

4.1.2 Use of a decreasing constriction to increase the exploration of the parameter space

As it has been explained before, it is easy for the optimizer to converge prematurely in a local maxima. One can avoid this problem by using a derivative of the PSO called Adaptive Diversity Particle Swarm Optimization (ADPSO). This algorithm introduces collision detection, dispersion of the particles, and collision reaction forces. It is a solution for resolving some flaw of the PSO, such as premature convergence [vdK09, MS06].

However during the project, the implementation of this algorithm within the BioRob software wasn't tested, and this testing cannot be performed within the time window. But to avoid premature convergence, a trick had been tested. As stated in [CK02], if the constriction parameter χ is kept inside $\left[0, \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}\right]$, the algorithm will converge to at least a local minima (see section 3.2.1). Then one can try to put a bigger starting value for χ and decrease it to a correct one, during the

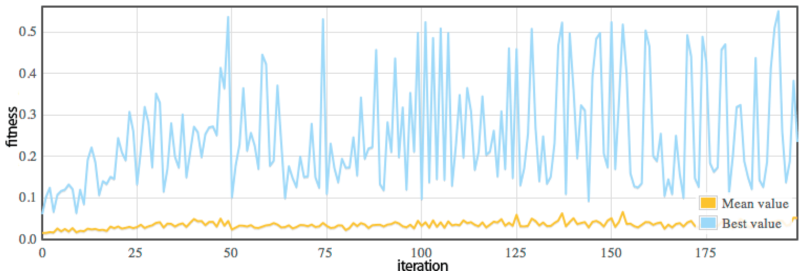


Figure 4.3 – Fitness for each iteration of the optimization using a decreasing constriction. Here there is any convergence (however the best fitness is near the global minima, see figure 4.2).

optimization process. Therefore, at the beginning, the algorithm will tend to have high velocity particles, that will cover large region of the search space, and it may reduce the chances to fall into a local minima.

This method had been tried using the following update rule for the constriction :

$$\chi_{start} = 1.2 \tag{4.1}$$

$$\chi_{end} = 0.6 \tag{4.2}$$

$$\chi_{i+1} = \chi_i + a * (\chi_{end} - \chi_i) \tag{4.3}$$

$$a = \frac{6}{numberIteration} \tag{4.4}$$

This rule insures that independently of the number of iterations, χ_{start} and χ_{end}, the value of χ_i will be very high for the first half of the optimization and around χ_{end} for the second half.

However the first trial with this trick hasn't been successful, as depicted by figure 4.3. Indeed, it doesn't converge at all. One can assume that the velocities of the particles increase too much and explode (it is to avoid this behavior that Clerk use the rule (3.29)), and then, the reducing of the constriction parameters is not sufficient to go back to a stable state.

However, further work should not be done under this direction, because this setup was a quick workaround to avoid premature convergence, and may not even work at all, even with a better tuning of χ_{start} and χ_{end}. Testing and performing ADPSO seems more promising.

4.2 Study of a learned trot gait without sensory feedback

4.2.1 Dynamical analysis

The first learned gait is showed in figure 4.4. Its mean speed is $0.608\,\mathrm{m.s^{-1}}$, which represent more than 3 body length of the robot. From the video one can observe that the gait has a high variation of the CoM height. Finally, the trunk is constantly pitched backward, therefore the hindlimbs :

- are highly folded during the gait,
- lacks foot clearance, because the hind feet touch the ground during swing phase
- their amplitudes are greater than needed, as for the second half of the retraction movement, the feet don't touch the ground anymore.

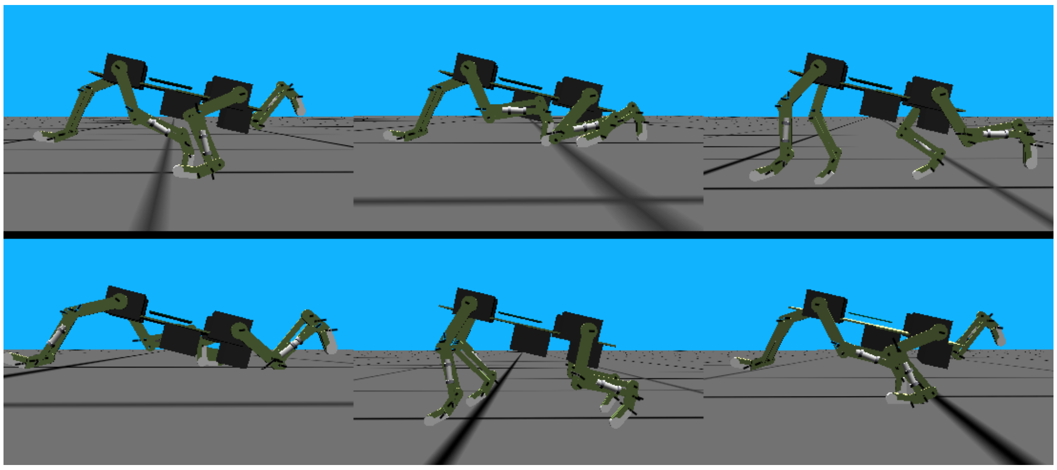


Figure 4.4 – Screen-shots of the learned gait for one gait cycle, Each screen-shot is taken 200 *ms*. The gait has high variation of the CoM, and forelimb doesn't support so much weight. Video available at `http://www.akay.fr/data/CheetahFirstTrotWithoutSensory.avi`.

The height of the CoM (approximated by the CoM of the trunk alone) for one gait cycle is shown in figure 4.5. Despite it has high variations (about 60% of its maximal value), it has a once repeated shape, that is twice the frequency of the gait periodic. Indeed, it is expected, as the CPG generate a symmetric trot.

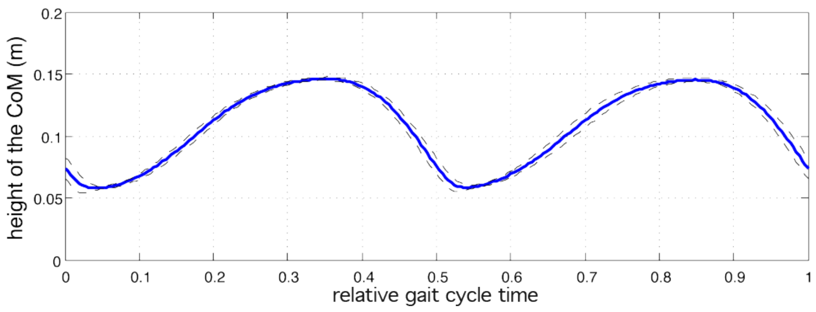


Figure 4.5 – Mean trajectory of the CoM (h_{CoM}) measured over 25 stride period. Dashed lines represent $< h_{CoM} > \pm Var(h_{CoM})$.

The same remark goes for the speed of the CoM (figure 4.6). This figure also shows that *Cheetah* has a "breaked" gait : at each touchdown, the speed of the CoM shrinks abruptly to a zero value, or even goes in the opposite direction, before reaching again its maximal speed.

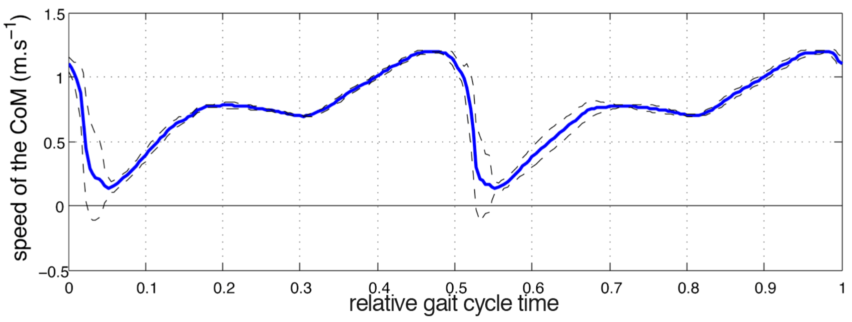


Figure 4.6 – Mean Speed of the CoM (h_{CoM}) measured over 25 stride period. The robot is breaking as the speed reaches zero, or even become inverted.

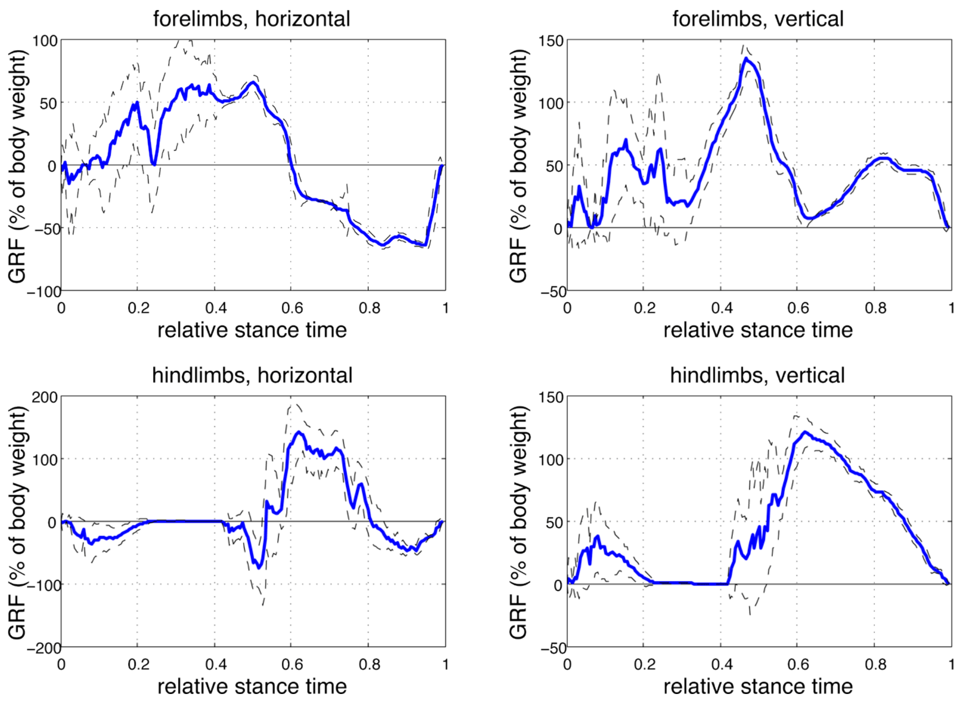


Figure 4.7 – Ground Reaction Forces measured over 25 stride period. They are normalized according to body weight of the *Cheetah* , and positive value indicates forces acting in the direction of propulsion. For the hindlimbs, the measure of the stance phase is distorted as there is a contact of the feet during swing phase (which is still represented on the graphics).

Figure 4.7 shows that the GRFs are pretty high (often more than 100% of the body weight). The total vertical GRF for the four limbs has a mean value of $\approx$ 122% of the body weight. This figure also exhibit the fact that the hind feet touch the ground during the swing phase (the small possitive value of vertical GRF at the beginning of the "stance phase"). The horizontal GRF for both fore and hindlimb are really elevated. Moreover for the hindlimbs, for $t \approx 60\,\%$ of the stance phase, the ratio is 1, which is almost the limit of the friction coefficient.

These high GRF values could be explained, as the robot is "breaking", it must at each stance phase, recover all its speed. One could also compare the GRFs pattern with the one for the *Ochotena rufescens* (Pika) (figure 4.8), were the GRFs of the forelimb are greater then for the hindlimb. [RQF04] reminds, as a generality for mammals, that they are different roles for hindlimbs, that provide power for forward locomotion and forelimbs, that provide braking and searching movements. This is supported by figure 4.8, where forelimb GRF is negative.

Here it is quite the opposite, although the horizontal forelimb GRF does get negative at the second half of the stance phase, in the beginning of the stance phase, forelimbs are the main propul-

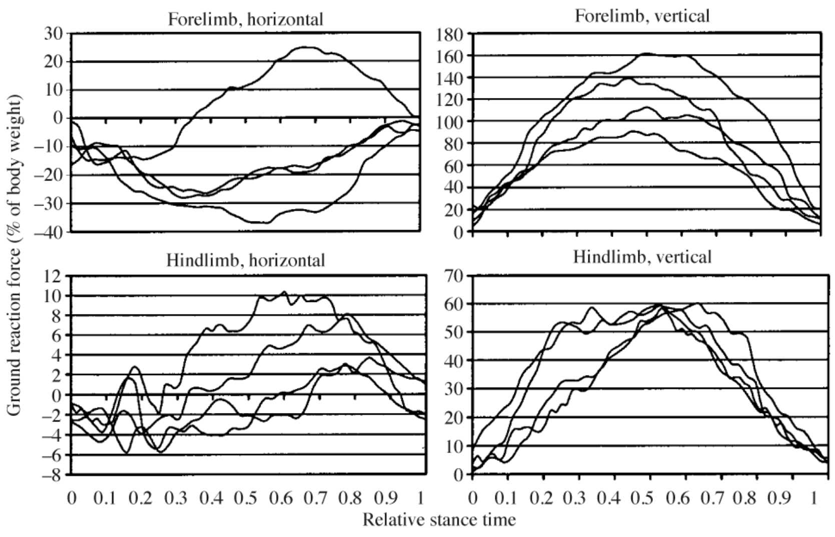


Figure 4.8 – GRFs measured in *Ochotena rufescens*, taken from [WBH+02]. It shows the role difference between braking forelimb and propulsive hindlimb.

sive limbs. Another thing described in figure 4.9, is that both the forelimbs and hindlimbs produces positive torques at the end of the stance phase, while the angle is still decreasing, and then the hip joints are taking energy from the system.

For the forelimb this is expected, because the GRF is negative, but for the hindlimb, they are not. So one can assume that the servo-motor is taking energy from the knee actuator, which is stored due to the leg compliance.

This show that we don't have an efficient locomotion, as it is expected for an efficient gait, to furnish energy during the whole stance phase. But here there is time period where both the hindlimb and forelimb is taking energy away, and then it may be one of the cause of the "breaked" gait.

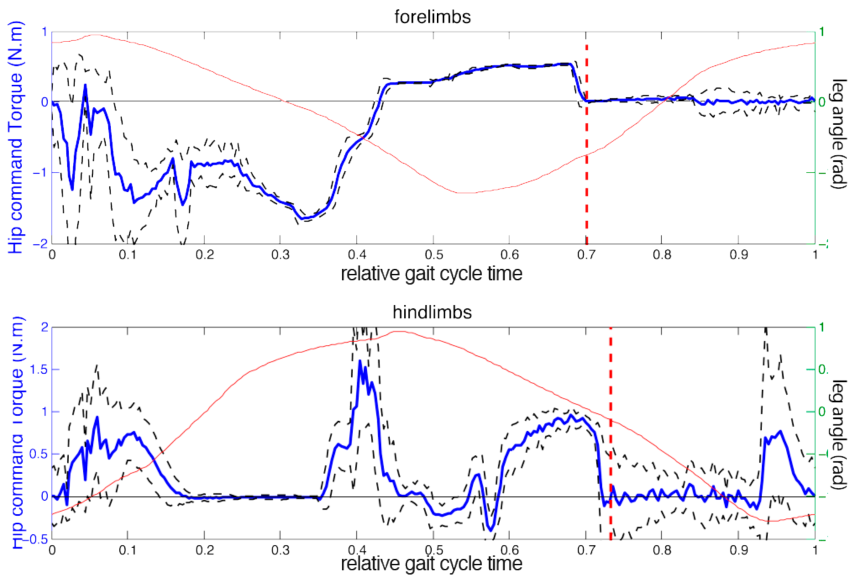


Figure 4.9 – Torque pattern in the hip joint, measured over 25 stride period. Red dashed line mark the transition between stance (left) and swing (right) phase. For the second half of the stance phase, one can observe that the joint is taking energy instead of giving it.

One can also assume that all of these patterns could be explained by the fact that the offset values of the command, put the forelimb highly backward (figure 4.10). So the *Cheetah* is in an

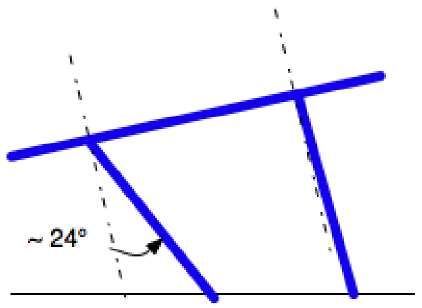


Figure 4.10 – Leg offset for the learned gait (forelimb -0.426 rad and hindlimbs 0.02 rad.

unstable position. Then the stable gaits are the one with high amplitudes, as the forelimbs need to be putted forward to avoid a fall (which will result in a low fitness for the controller).

Due to this high amplitude, the forelimb have high velocities and slap the ground (great forelimb force and hip torque-figure 4.9). They don't have a braking effect, and are now propulsive. The low angle of attack make the CoM to move more upward than forward.

This low angle of attack may be caused by the over-folding of the hindlimb. Moreover it causes a lack of foot clearance, that brakes the gait while the hind feet touches the ground during swing phase.

Therefore the learned gait seems unnatural and is not mechanically efficient. But it might be interesting to know how much one can change or adapt the parameter of the control, while still having a similar gait.

4.2.2 Proposition of a method to find the less influential parameters

It may be useful to know what are the parameters that have the less influence on the learned gait. The term less influence should be taken in the sense that if one of the parameter is changed, it will not decrease the performance of the resulting gait. We propose here an orginal method to guess this parameter, and discuss its correctness.

One can remenber, that the use of an evolutionnary algorithm, give us a lot of positions in the parameter space, from which we know the fitness. This information could be used with statistical methods to answer this question.

One of the simpliest method, Principal Component Analysis (PCA) [Bil08], could give a guess of these parameters. This methods search to characterize a set of data. It involves to compute the mean and covariance matrix of the data, and diagonalize it. Then the eigenvectors associated with the highest eigenvalues (called the principal component) are examinated.

Applied to this problem, PCA could be performed on the best particles over all the iteration, selected by a certain threshold. Then a small displacement from the mean along the direction of the principal component will not decrease the performance of the controller, as there is a high probability that the resulting particle have a fitness superior to the threshold.

However, this method may not give us the less influential parameter, but instead show that such a parameter doesn't exist. The less influential parameter(s) could be "guessed" if :

- there is one eigenvalue, or a set of eigenvalues that dominate(s) the other.
- the associated eigenvector is aligned with one particular parameter, i.e. is almost a canonical basis vector.

If the first condition isn't fulfilled, then all parameters will have the same influence. If the the second isn't fulfilled, it's a weighted combination of parameters one have to modify to keep the performance of our controller inside the threshold.

Finally, the eigenvalue give us another information : it indicates how much the parameter or combinaison of one can be modified without degrading the performance of the controller.

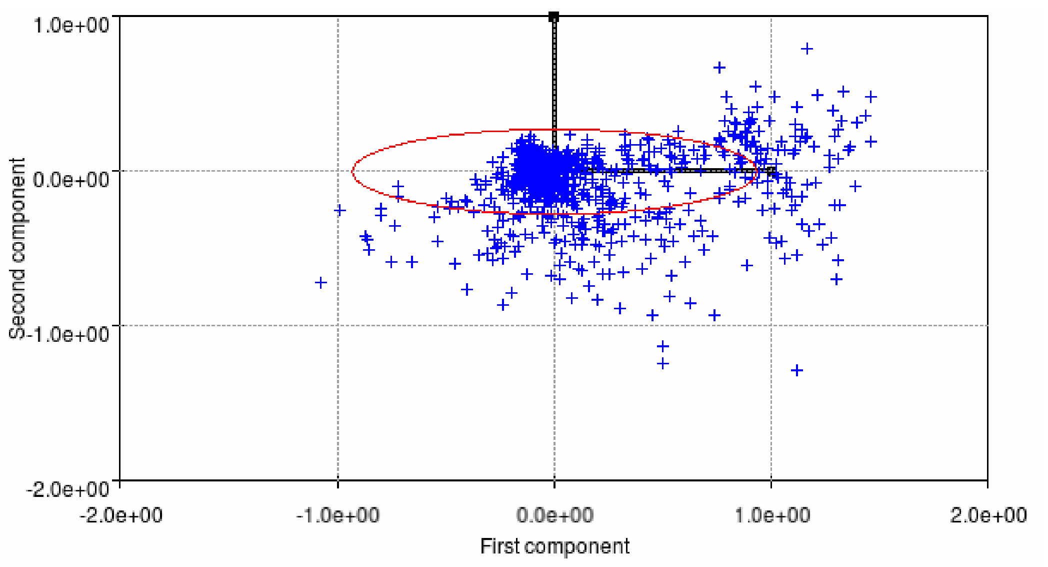


Figure 4.11 – Principal Components Analysis of the particle best positions. The position of the particles that have a fitness bigger than $\frac{4}{5}max_{fitness}$ are represented, projected on the two principal components. The ellipsoid represent the approximation made by PCA, of the data set by a gaussian.

With the trot gait the following method has been used for a threshold of $\frac{4}{5}max_{fitness}$. The first eigenvalues is a bit higher ($7.46e^{-2}$) than the other ($\geq 2.19e^{-2}$) (see table 4.1). And its associated vector is almost directed along one direction: the parameter Fore::Knee::ThExtension.

This result isn't surprising. Indeed figure 4.4 shows that the forelimb touches the ground, after being fully extended. So if we change the threshold of the time we extend the leg, this will have no effect, for cases where the leg is still extended at the foot touchdown.

Finally, one must point out, this approach may be biased. PCA is a *learning machine* algorithm used to extract the stochastic characteristic of a set of data. But here our set of data is defined by "the position of the parameter space, explored by the PSO, whose fitness exceed a certain threshold". Somehow, some information from the PSO computation are also learned. One can make the assumption, that a PSO that has converged for a long time, is almost equivalent than random trial in a neighborhood of the extrema, as the particles oscillate in this neighborhood. But such strong assumption must be verified, which is beyond the scope of this project.

Eigenvalues
$3.09e^{-4}$
$6.45e^{-4}$
$7.37e^{-4}$
$1.21e^{-3}$
$1.34e^{-3}$
$1.93e^{-3}$
$3.60e^{-3}$
$4.62e^{-3}$
$5.32e^{-3}$
$7.39e^{-3}$
$1.49e^{-2}$
$2.19e^{-2}$
$7.46e^{-2}$

Parameter	Eigenvector
Hips::Duty	$8.17e^{-2}$
Fore::Hip::Amplitude	$-1.06e^{-1}$
Fore::Hip::Offset	$-7.11e^{-2}$
Fore::Hip::Retraction	$1.14e^{-1}$
Hind::Hip::Amplitude	$-3.82e^{-2}$
Hind::Hip::Offset	$2.36e^{-2}$
Hind::Hip::Retraction	$1.40e^{-2}$
Fore::Knee::Amplitude	$1.01e^{-1}$
Fore::Knee::ThExtension	$-3.30e^{-2}$
Fore::Knee::ThContraction	$-9.39e^{-1}$
Hind::Knee::Amplitude	$1.44e^{-1}$
Hind::Knee::ThExtension	$-1.78e^{-1}$
Hind::Knee::ThContraction	$1.23e^{-1}$

Table 4.1 – Eigenvalues of the PCA, and first component.

4.2.3 Insight for improving the gaits

Has it has been stated before, the first gait obtained with the workflow, didn't perform well. One of the main reason, is that the hindlimb are too much folded, as they support most of the weight of the robot. In comparison with quadrupedal mammal, this is the opposite, where forelimb support more weight. Then one should move the CoM to lay more on the front than the back of the *Cheetah* . In addition, the hind leg length, and its spring knee fitness could be increased.

Moreover, it has been shown, that the hindlimb uses too much amplitude, and lack foot clearance. Adding some sensory feedback could help to deal with, because :

- controller that use too much amplitude, will see their amplitude reduced by fast transitions.
- if the foot lacks clearance, when it would touch the ground, will enable a fast transition, and then will start he stance phase.

Finally, changing the fitness function to the inverse of the specific resistance (section 3.2.2) may also help to obtain energy efficient gait, and to avoid the "breaked" gait.

Conclusion

The main goal of this project was to improve the locomotion control of *Cheetah* , in the same philosophy, that it has been previously done : by taking inspiration from nature. Not by imitating it, but by understanding the underlying principles in animal locomotion, and by adapting it to robotics.

In this perspective, some behaviors and principles currently investigated in biomechanics were presented, and a method to add the leg retraction principle to a Central Pattern Generator designed by Ludovic RIGHETTI has been proposed. The idea behind this addition is not to interfere with the dynamical properties of the oscillator, and to transform no more than needed their output.

But prior to test this controller for *Cheetah* , an update of previous WEBOTS models was needed. A new model for the knee actuator was proposed, and successfully tested. However it still needs further improvements, as the maximum stiffness admitted by the model might be too small.

The introduction of this new model, has revealed a problem of conception of the previous model, due to some limitation of Open Dynamics Engine. This problem was solved, by removing one of the two closed kinematic loops of the pantographic mechanism of the leg, and by simulating the removed part by directly adding the force computed by our own model into ODE.

Then the software controller has been developed, based upon the BIOROB Optimization and CPG framework. The software has been made with the idea to control almost all parameters by configuration files, allowing the computation of different optimizations at the same time.

Furthermore the comparison between two way to reject the unstable gait on the level of the fitness function was performed.The first of the method sets a value of zero for unstable gait, and the second takes the distance traveled until the instability has been discovered as fitness. The comparison showed that the second one is more efficient.

A method which modulates the constriction value of the Particle Swarm Optimization, in order to increase the exploration of the parameter space was also proposed. However its is not stable, and if a better exploration of the space is needed, then it might be more useful to check out the Adaptive Diversity Particle Swarm Optimization algorithm.

Finally these developments were too much time consuming, and we were only able to perform and extensively study the learning of one single gait. The study shows that this gait was both not energy efficient, and not comparable to gait performed by small mammals. The simplicity of the control of this gait, that did not include sensory feedback, prevent us to test the impact of the leg retraction principle on locomotion behaviors of the *Cheetah* .

One solution to increase *Cheetah* performance, may arise from the mass redistribution, change of the leg geometry and fitness. But the use of a fitness based on the energy expenditure, like the specific resistance, should be tested. Furthermore, addition of bioinspired mechanism, like a spinal cord, or mechanically driven toes, should also be tested, because they are strategy used by mammals, in order to improve respectively the stride length in running gait, and the energetic efficiency of the gait.

Lausanne, the January 29th 2010
TULEU Alexandre

Acknowledgments

I want to deeply acknowledge :

- My supervisor, Alexander SPROËWITZ, for all the time he spent for me, even in the last minutes, when I thought that everything was impossible.
- Jesse VAN DEN KIEBOOM, who spent lot of time helping me to understand and set up the Optimization framework. Its advises on WEBOTS and ODE were also really helpful.
- Alessandro CRESPI, for its kindness and availability, when I had to deal with informatic issues.
- Yvan Bourquin, who has been a great help, specially when I needed support for WEBOTS .
- Professor Auke Jan IJSPEERT, who supervised me at high level, and for his kindness when I had some trouble with deadlines.

Bibliography

[Ale84] R. McN. Alexander. The Gaits of Bipedal and Quadrupedal Animals. *The International Journal of Robotics Research*, 3(2):49–59, 1984.

[Bil08] Aude Billard. Applied machine learning. Lecture Notes, 2008.

[CK02] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. 6(1):58–73, Feb. 2002.

[Cyb] CyberBotics. Webots, fast prototyping and simulation of robots. `http://www.cyberbotics.com`.

[DK09] J. C. Dean and A. D. Kuo. Elastic coupling of limb joints enables faster bipedal walking. *Journal of The Royal Society Interface*, 6(35):561–573, June 2009.

[DVB09] Monica A Daley, Alexandra Voloshina, and Andrew A Biewener. The role of intrinsic muscle mechanics in the neuromuscular control of stable running in the guinea fowl. *J Physiol*, 587(Pt 11):2693–2707, Jun 2009.

[FB06] M.S. Fischer and R. Blickhan. The tri-segmented limbs of therian mammals: kinematics, dynamics, and self-stabilization - a review. *J Exp Zool*, 305A(11):935–952, 2006.

[FSS$^+$02] Martin S Fischer, Nadja Schilling, Manuela Schmidt, Dieter Haarhaus, and Hartmut Witte. Basic limb kinematics of small therian mammals. *J Exp Biol*, 205(Pt 9):1315–1338, May 2002.

[IC07] A.J. Ijspeert and A. Crespi. Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. pages 262–268, 2007.

[Ijs08] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

[IP04] F. Iida and R. Pfeifer. "Cheap" rapid locomotion of a quadruped robot: Self-stabilization of bounding gait. In F. Groen et al., editor, *Intelligent Autonomous Systems 8*, pages 642–649. IOS Press, 2004.

[ISI] CEA-List & ISIR. Arboris un simulateur de chaîne de solide polyarticulées. `https://vizir.robot.jussieu.fr/trac/arboris/`.

[Kvi09] Ivan Kviatkevitch. Locomotion exploiting body dynamics on the cheetah robot. Technical report, Biologically Inspired Robotic Group, 2009.

[May09] Mikaël Mayer. Roombot modules - kinematics considerations for moving optimizations. Semester project, Biologically Inspired Robotic Group, 2009.

[Med] website MedecineNet.com. Definition of anatomic orientation terms. `http://www.medterms.com/script/main/art.asp?articlekey=9210`.

[Mot] Maxon Motors. Maxon online catalog. `http://shop.maxonmotor.com/ishop/app?language=en&country=INT`.

[MS06] Christopher K. Monson and Kevin D. Seppi. Adaptive diversity in pso. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 59–66, New York, NY, USA, 2006. ACM.

[ODE] ODE develloping team, `http://opende.sourceforge.net/wiki/index.php/Manual`. *Open Dynamic Engine Users's Manual*, 1.10 edition.

[RI08] L. Righetti and A. J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, pages 819–824, 19–23 May 2008.

[Rie08] Martin Riess. Development and test of a model for the cheetah robot. Technical report, Biologically inspired Robotic Group, 2008.

[Rob] Robotis, `http://www.robotis.com`. *Dynamixel RX-28 Users's Manual*, 1.10 edition.

[RQF04] Roy E. Ritzmann, Roger D. Quinn, and Martin S. Fischer. Convergent evolution and locomotion through complex terrain by insects, vertebrates and robots. *Arthropod Structure and Development*, 33(3):361 – 379, 2004. Arthropod Locomotion Systems: from Biological Materials and Systems to Robotics.

[Rut08] Simon Rutishauser. Cheetah: compliant quadruped robot. Technical report, Biologically Inspired Robotic Group, 2008.

[Sch05] Nadja Schilling. Ontogenetic development of locomotion in small mammals–a kinematic study. *J Exp Biol*, 208(Pt 21):4013–4034, Nov 2005.

[SGGB02] Andre Seyfarth, Hartmut Geyer, Michael Günther, and Reinhard Blickhan. A movement criterion for running. *Journal of Biomechanics*, 35(5):649 – 655, 2002.

[SGH03] André Seyfarth, Hartmut Geyer, and Hugh Herr. Swing-leg retraction: a simple control model for stable running. *J Exp Biol*, 206(Pt 15):2547–2555, Aug 2003.

[SMM$^+$08] A. Sproewitz, R. Moeckel, J. Maye, M. Asadpour, and A.J. Ijspeert. Locomotion in modular robots based on central pattern generators. In *AMAM 2008 abstracts*, pages 86–87, June 2008.

[TPPB00] S. Talebi, I. Poulakakis, E. Papadopoulos, and M. Buehler. Quadruped robot running with a bounding gate. In *Proc. 7 th Int. Symp. on Experimental Robotic (ISER'00*, pages 281–289. Springer-Verlag, 2000.

[Tul09] Alexandre Tuleu. ROOMBOTS central pattern generators, symmetries and online learning. Semester project, Biologically Inspired Robotic Group, 2009.

[vdK09] Jesse van den Kieboom. Biorob software documentation. `http://birg2.epfl.ch/~jvanden/docs/`, 2009.

[WBH$^+$02] Hartmut Witte, Jutta Biltzinger, Rémi Hackert, Nadja Schilling, Manuela Schmidt, Christian Reich, and Martin S Fischer. Torque patterns of the limbs of small therian mammals during locomotion on flat ground. *J Exp Biol*, 205(Pt 9):1339–1353, May 2002.

[WHI$^+$00] H. Witte, R. Hackert, J. Ilg, J. Biltzinger, N. Schilling, F. Biedermann, M. Jergas, H. Preuschoft, and M. S. Fisher. Quadrupedal mammals as paragons for walking machines. In *International Symposium on Adaptive Motion of Animals and Machines, Montreal, Canada, August 8-12, 2000*, 2000.

[Wie08] Sandra Wieser. Locomotion in Modular Robotics : Roombot Module. Semester project, Biologically Inspired Robotic Group, 2008.

[Wik] Wikipedia. The ziegler-nichols method, wikipedia entry. `http://en.wikipedia.org/wiki/Ziegler\T1\textendashNichols_method`.

Glossary

caudal directed toward the tail or hind part, [Med].. 3

cranial directed toward the head or front part, [Med].. 3

distal Further from the beginning (spine), as opposed to proximal, [Med].. 3, 4

protraction the cranial displacement of the distal end of each segment of a limb [FSS^{+}02].. 3

proximal Toward the beginning (spine), as opposed to distal, [Med].. 3

retraction the caudal displacement of the distal end of each segment of a limb, [FSS^{+}02].. 3

stance phase Phase during a stride period where the foot touches the ground.. 3

swing phase Phase during a stride period where the foot doesn't touche the ground.. 3

Acronyms

CFM	Constraint Force Mixing. 14–16
ERP	Error Reduction Parameter. 14, 15
ODE	Open Dynamics Engine. 14–17, 23, 43
ADPSO	Adaptive Diversity Particle Swarm Optimization. 35, 36, 43
CoM	Center of Mass. 5, 29, 37, 38, 40, 42
CPG	Central Pattern Generator. 1–3, 5, 7, 8, 18, 19, 25, 43
DoF	Degree of Freedom. 6, 8
GRF	Ground Reaction Force. 23, 38, 39
PCA	Principal Component Analysis. 40, 41
PSO	Particle Swarm Optimization. 31, 35, 41, 43

List of Figures

List of Tables