Project Report in Legged Robots

Developing a Walking Controller for a Three-link 2D Biped

MICRO-507

NIEDERHAUSER Thibault
SIEVERING Ivan-Daniel
STEFANINI Niccolò

École Polytechnique Fédérale de Lausanne
Autumn 2019

Contents

1. Introduction

This project was conducted in the the frame of the course MICRO-507 *Legged Robots* at EPFL. The aim is to design, implement and analyse the walking control of a simple biped. The considered biped is 2-dimensional and made out of three links (two legs and a torso), each of them having its own punctual mass (see Figure 1.1).

The biped, its kinematics, its dynamics and its control are modeled, simulated and animated in a *Matlab* environment.

In this project, three different controllers are implemented: two classical control methods (PD-controller and Virtual Force Control) and one learning method (Deep Reinforcement Learning).

In a first time, this report explains how the kinematics and dynamics of the biped are designed and how the controller are implemented and tuned so that the biped can walk. Then, in a second time, the three controller's performance are analysed regarding different criteria (speed, robustness against perturbations, cost of transport, etc.). Eventually a synthetic comparison of the three methods is made.

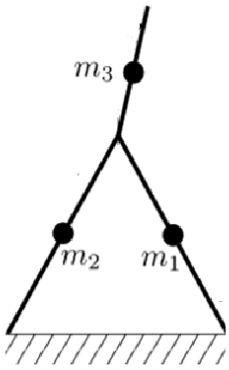


Figure 1.1: 2D Three-link biped

2. Methods

2.1 Model

The model of the biped has be done in three main parts : generating the kinematic, generating the dynamic and implementing the impact map.
First the kinematics was generated. i.e. the position and velocity of the three masses m_1, m_2 and m_3, but also the position and velocity of some important points like the swing foot (shorten in this report as swf), the hip and the top of the torso. The stance leg is the origin of the system. The following pictures show the different points of interest on the biped.

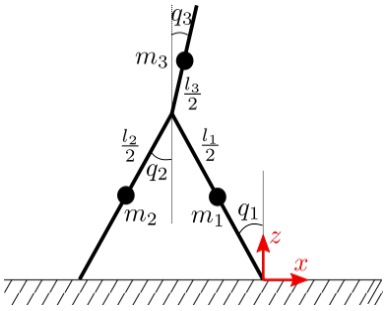


Figure 2.1: Position of the masses on the biped

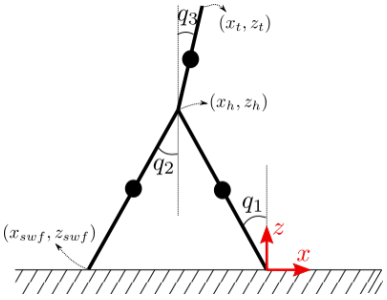


Figure 2.2: Position of the important points on the biped

The dynamic of the biped is computed using the Lagrange method and the equation of motion :

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu \tag{2.1}$$

Where q represent the different joint angles and u is the control input. M is the mass matrix, C the Coriolis matrix, G the gravity matrix and B the control matrix (it describes which actuator apply on which angle, and in which direction). The control torques are applied between the torso and the stance leg and between the torso and the swing leg. More explicitly :

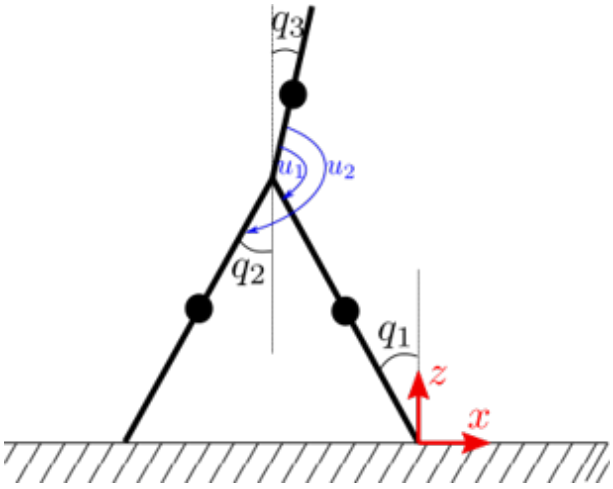


Figure 2.3: Position of the control torques on the biped

To complete the model the switch between the stance leg and the swing leg was implemented when the swing foot collides with the ground, so that the biped can walk. Note that for simplicity the switch is instantaneous. To implement this an impact map (relation between the angles and velocities before and after the impact) was implemented. One can represent it as follows :

$$\Delta(q^-, \dot{q}^-) = (q^+, \dot{q}^+) = (\Delta_q(q^-), \Delta_{\dot{q}}(q^-, \dot{q}^-)) \tag{2.2}$$

For Δ_q the impact map is trivial as only some indices are switched.
But for $\Delta_{\dot{q}}$ the computation is more complex as velocities are not the same before and after the impact. Conservation of angular momentum ([3]) was used:

$$A_- \dot{q}^- = H_- = H_+ = A_+ \dot{q}^+ \Rightarrow \dot{q}^+ = A^- 1_+ * A_- \dot{q}^- \tag{2.3}$$

A script that gives us the evolution of the biped over the time was programmed. It works with *ode45* (an differential equation solver). The architecture is the following :

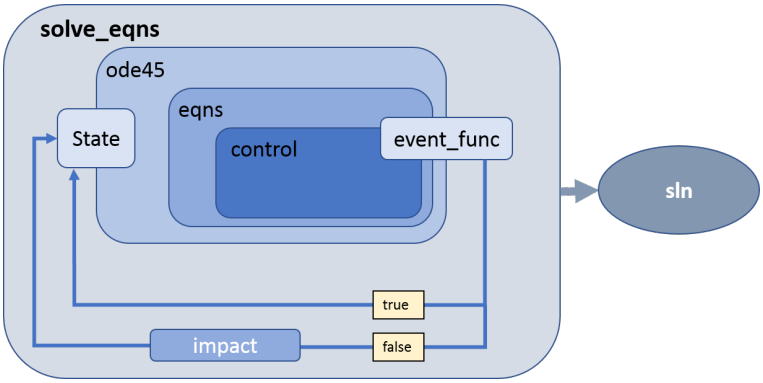


Figure 2.4: Architecture of the computation of the displacement of the biped

ode45 is called at a high frequency and computes the new position and velocities (state) in function of the control that is applied to it (that depends on the previous state). If event function detects that there is

a collision with the ground the impact map updates the positions. When the desired time of simulation is reached, the function returns a structure (sln) which contains how the biped evolved during the simulation time.

2.2 Controllers

The three-link 2D biped robot have two actuators that apply torques between the torso and the stance leg (u_1), and the torso and the swing leg (u_2) respectively. Therefore two controllers need to be designed in order to make the three-link 2D biped walk. Note that since the robot has three degrees of freedom but only two actuators, the system is under-actuated. Hence, one needs to control combination of variables rather than every variable independently.
In our implementation, two classical control methods and one learning method were applied and tested:

- PD Controller
- Virtual Model Control
- Deep Reinforcement Learning (DDPG)

PD Controller

The first control method that is applied to the biped is a PD controller. This was chosen as a first method, because it is simple to design and implement: it gives a good first insight in the challenges of biped control without adding much complexity. Note that the derivative term is important in this case because it allows for a smoother control, reducing the overshoot and oscillations that could cause the robot to fall. On the other hand, it is unnecessary to add an integrative term to the PD controller, because the time of each step of the robot is too short to reach a steady state and therefore a potential steady-state error is not problematic. Moreover, an integrative term would increase the overshoot and degrade the stability.
The first controller, u_1, is dedicated to stabilizing the torso whereas the second controller u_2, controls the spread angle (angle between the two legs, combination of two angles q_1 and q_2). The control laws are the following:

$$u_1 = K_{p1} * \epsilon_{torso} + K_{d1} * \epsilon_{dtorso} \tag{2.4}$$

$$u_2 = K_{p2} * \epsilon_{spread} + K_{d2} * \epsilon_{dspread} \tag{2.5}$$

Where K_p and K_d are the parameters of the controller and ϵ represents the errors (the difference between the measured value and our target value):

$$\epsilon_{torso} = q_3 - qt_{des} \qquad \epsilon_{spread} = (-q_2 + q_1) - spread_{des}$$

$$\epsilon_{dtorso} = 0 - dq_3 \qquad \epsilon_{dspread} = 0 - dq_2 + dq_1$$

The equations show that $u1$ affects q_1 and $q3$ but only depends on $q1$ and u_2 affects q_2 and $q3$ but depends on $q1$ and $q2$. Consequently, no controller has both $q1$ as input and output, therefore $q1$ can be considered as the free variable of the under-actuated system. Note that the output of the control function is saturated beyond 30 Nm and -30 Nm.

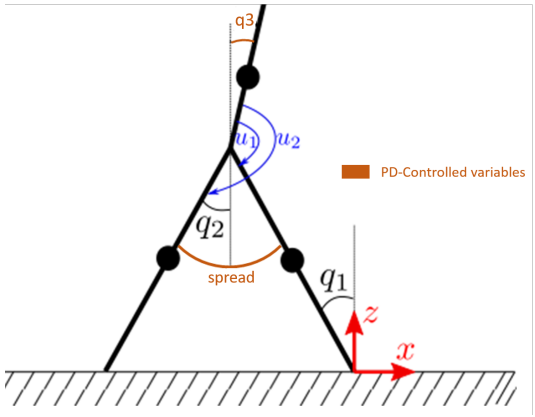


Figure 2.5: PD controller

Virtual Model Control

Virtual Model Control is a control that consists in modeling virtual forces. The virtual forces are applied to the robot by converting them into joint torques using the Jacobian of the system. This method is suitable for the control of biped robots [1], allows for a more intuitive control design and is robust against external perturbations.
The virtual forces that we decided to implement are the following :

$$F = \begin{pmatrix} C_1 * (dx_{hip} - v_{target}) \\ k_1 * (x_{des} - x_{swf}) + C_2 * (-dx_{swf}) \\ k_2 * sin(q_2) \\ -k_3 * (q_3 - qt_{des}) - C_3 * dq_3 \end{pmatrix} \begin{matrix} (f_1) \\ (f_2) \\ (f_3) \\ (f_4) \end{matrix}$$

- f_1 is a force applied to the hip. It is a damper that tries to give the desired speed to the hip. This force makes the biped move forward.
- f_2 is the combination spring and a damper in parallel that applies a force to the swing foot to move to x_{des}, the desired position of the next step. Without this force, the swing leg would not swing forward.
- f_3 is a spring that raises the swing foot when it is behind the hip and pull it down when it is in front of the hip.This force prevents the robot from stumbling.
- $f4$ is a spring and a damper in parallel that is applied on q_3. This force ensures a human-like posture by maintaining the torso at a desired angle.

Note that there is no virtual force that directly affects $q1$, hence $q1$ is the free variable of the under-actuated system.
In order to convert the virtual forces into joint forces, the Jacobian of the system is required. It is known that $x = J^T\theta$, where x corresponds to the the application points of the virtual forces and θ is a vector of the joint angles. In this case:

$$\begin{pmatrix} x_{hip} \\ x_{swf} \\ z_{swf} \\ q_{torso} \end{pmatrix} = J^T \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$$

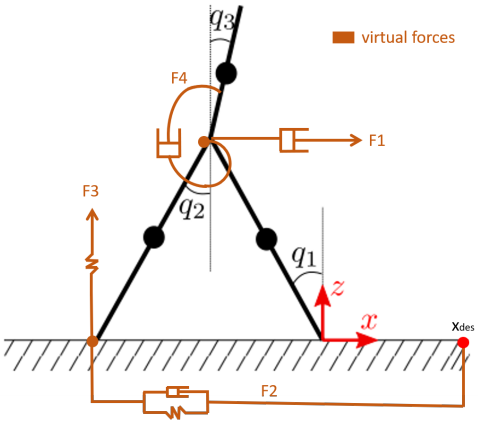


Figure 2.6: Virtual Model Control

Furthermore, the relations between x and θ are the following:

$$\begin{pmatrix} x_{hip} \\ x_{swf} \\ z_{swf} \\ q_{torso} \end{pmatrix} = \begin{pmatrix} f_1(\theta) \\ f_2(\theta) \\ f_3(\theta) \\ f_4(\theta) \end{pmatrix} = \begin{pmatrix} l1 \sin q_1 \\ l1 \sin q_1 - l1 \sin q_2 \\ l1 \cos q_1 - l1 \cos q_2 \\ q_3 \end{pmatrix}$$

And the Jacobian can be computed as follows:

$$J = \begin{pmatrix} \frac{f1}{dq_1} & \frac{f1}{dq_2} & \frac{f1}{dq_3} \\ \frac{f2}{dq_1} & \frac{f2}{dq_2} & \frac{f2}{dq_3} \\ \frac{f3}{dq_1} & \frac{f3}{dq_2} & \frac{f3}{dq_3} \\ \frac{f4}{dq_1} & \frac{f4}{dq_2} & \frac{f4}{dq_3} \end{pmatrix} = \begin{pmatrix} l_1 cos(q_1) & 0 & 0 \\ l_1 cos(q_1) & -l_1 cos(q_2) & 0 \\ -l_1 sin(q_1) & l_1 sin(q_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Now one can compute the control input by applying the transformation :

$$u = B^T * J^T * F$$

where B is a control matrix that maps the joint torques to the controlled torques $u1$ and $u2$.
As before the output of the control function is saturated beyond 30 Nm and -30 Nm.

DDPG

DDPG, aka Deep Deterministic Policy Gradient, is a Policy Gradient algorithm built upon two deep neural networks with the objectif to train an Actor (the first one) capable of maximizing a reward anticipated by the Critic (the second). Together, Actor and Critic compose the Agent.

Proposed in 2014 by Deepmind, and used by Google in 2016 achieving astonishing end-to-end control results[5], DDPG provides an interesting and challenging tool to make the biped learn to walk.

The principle of end-to-end Reinforcement Learning in control, and more specifically for our case, is to create an Agent capable of generating actions from the observation of the state maximizing a reward function which is then calculated on the results of the action on the state. This means that training a DDPG algorithm doesn't

require a precise comprehension of the system kinematic or dynamic (just the size of states and observations, provided a good simulator) but requires a well crafted reward function and two proper networks capable of learning what is required to achieve good results. Also, a lot of time and patience is not optional when dealing with those algorithms.

The training process was run on two medium level PC lasting from 5 to 15 hours. Two combinations of NN were used to learn the model to which we will refer as *smallmodel* and *bigmodel*, both following the architecture in fig 2.7 for the Actor and Critic networks.

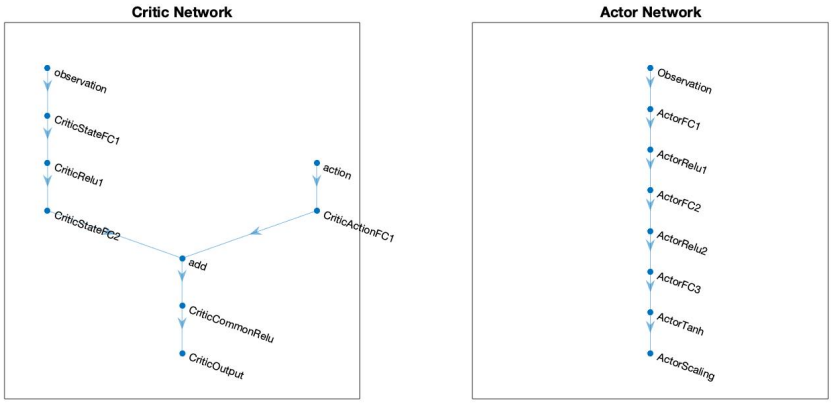


Figure 2.7: Architecture for ddpg agent

As of the *smallmodel* the observation is assumed 6x1, corresponding to the state, for the *bigmodel* the observation is 8x1 corresponding to the state and two more free variables for possible speed tracking purposes. The action is a 2x1 vector.

In the Critic Action and Observation follow their own path of fully connected layers until they are added together, and pass inside a final layer adding them together. Obviously the observation path is longer because the observation, having a bigger dimension, is more complex and has more hidden dependencies to be covered.

In the Actor the observation is passed through some FC layers alternated with ReLu, to finally get to a tanh layer which limits the output in the interval of ± 1. A scaling layer is added allowing the maximum output to match the maximum torque possible from the motor.

The dimensions for the two models are detailed in table 2.2:

	Small model	Big model
observation	6	8
CriticStateFc1	16	200
CriticStateFc2	8	100
action	2	2
CriticActionFc1	8	100
ciriticOutput	2	2
ActorFC1	10	300
ActorFC2	5	200
ActorFC3	2	2

The environment also needs a step function which is updating the state and giving a reward. The model of the biped was hence adapted to advance one step at the time and the frequency was reduced to 100 Hz to have

faster simulations.

Notes:

- All the result of the simulations are visible on the drive. The simulation is coded to save one video every 100 episodes to show the progess.
- In the next Subsection we will refer to step as 'simulation step' and not 'robot step' where more steps (usually 3-500) compose an episode.

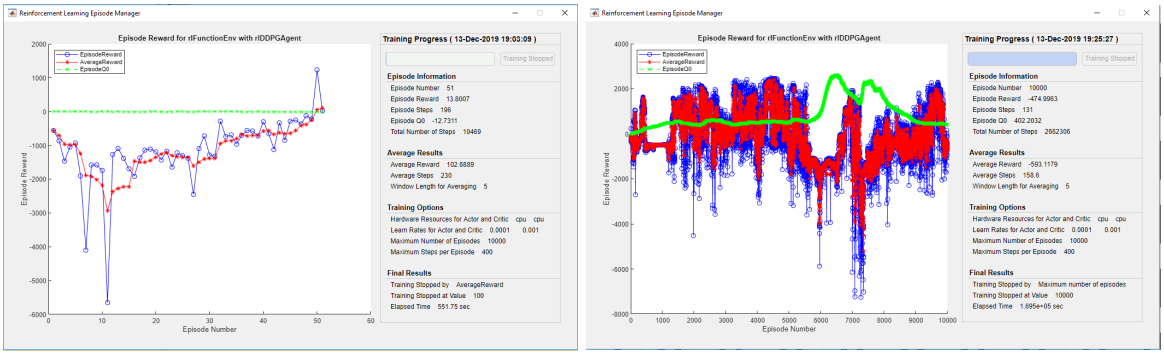
Simulation approach

Underestimating the complexity of the challenge, the first model that was developed was the *smallmodel*, the first simulation used a reward function crafted to make the robot walk directly (combination of speed and z coordinate of the hip), this simulation didn't lead to any result, putting in discussion the validity of the model. The second training was on an easy-to-get reward function (maximization of zhip) to validate the model and the hyper-parameters. After some thousands of simulations the robot started to try to stand with the stance leg straight and spinning torso and swing leg to maintain the position, but even after 10k episodes the robot did not make it to the end. Understanding the robot desists to learn to swing because of the sudden change of leg every times he tries and the following destabilization, a change in the approach was decided, the new approach was to imitate a PD controller that has been developed for an other method.

Here the step function is tricky: for every step that was calculated, the torques from the Actor and from the PD, the reward is the negative of the MSE, the next state is calculated using the torques from the PD, so that the biped is obliged to follow and learn the trajectory of this controller.

While the *Smallmodel* didn't converge properly, the newly developed *Bigmodel* was able to converge after few hundreds of steps.

Since the Actor were now able to swing, the obtained pd mimicking Agent was taken, and a reward function was crafted validating all the order of magnitude of the component by test-case. The Agent was trained for other 10k episodes. It result in fig 2.8.



(a) Early training

(b) Complete training

Figure 2.8: Training results for the last simulation. Here the green line is the expected understanding of the model, the blue is the reward per episode and the red is a mean filter applied on the reward/episode with window of 5 elements

Reward function

As explained *bigmodel* achieved very good reward after a small number of steps, and then explored many solutions all being good for the given reward. The reward that was used is the following :

```
speedReward = 5*min(dx,10);
torsoPenality = 5*(normAngle(q_rew(3) - 0.09))^2;
zPenality =  20*abs(min(z-0.5,0))^2;
actionPenality = 0.002 * sum(Action.^2);
aliveReward = 0.5;

Reward = speedReward - torsoPenality - zPenality - actionPenality + aliveReward;
```

The reward is continuous (given every step) and take into account many parameters.

- The speed is limited to 10m/s to avoid the biped to launch himself getting to a sudden death
- The torso penality is very small, the reference angle is set to be 5 degrees, the result of the previous PD. The biped will learn not to care about this penality in order to maximize the reward
- zPenality try to stabilize the robot
- action penality give a penality on the torques, helping to reduce the CoT
- aliveReward is given to award the robot for being alive, so that the robot wont suicide to maximize is reward

2.3 Optimization

Both the PD Controller and the Virtual Model Control have several variables that need to be tuned to achieve the desired performance. The parameters that are optimized for both controllers are summed up in the following table:

Table 2.1: Parameters to optimise

PD	VMC
K_{p1}, K_{p2}, K_{d1}, K_{d2}, $spread_{des}$, qt_{des}	C_1, C_2, C_3, k_1, k_2, k_3, qt_{des}, x_{des}

The aim of the optimisation is to find parameters that allows the biped to walk at a specified speed. Therefore the cost function needs to take into account a speed target. In addition two further cost contributions are implemented: a cost on the hip height to prevent the robot from falling and a cost on the torso angle to ensure a human-like walking gait:

$$Cost = C_1 * speed_{cost} + C_2 * zhip_{cost} + C_3 * torso_{cost}$$

where C_1, C_2 and C_3 are weight function and $speed_{cost}$, $zhip_{cost}$, $torso_{cost}$ represent cost respectively on the speed of the biped, the position of its hip along z and the position of its torso:

$$speed_{cost} = \sum_{i=1}^{N} [v(s) - v_{des}]^2$$

$$zhip_{cost} = \sum_{i=1}^{N} [zhip(s) - zhip_{des}]^2$$

$$torso_{cost} = \sum_{i=1}^{N} [q_3(s) - qt_{des}]^2$$

where N is the number of steps considered for the optimization, v_{des} is the desired speed, $zhip_{des}$ is the desired height of the hip at each step and qt_{des} is the desired angle of the torso.

The cost function is non-convex and its optimisation is time-consuming since it includes the solving of an ODE. To solve this optimisation problem, different functions from MATLAB Global Optimisation Toolbox were tested.

Table 2.2: Optimisation functions

Optimisation function	**Algorithm**	**Solving Time** (for 30 steps)	**Optimisation Performance**
fmincon()	Interior point (by default)	$\sim 1\,\text{min}$	Only finds local minima.
patternsearch()	Direct search	$10 - 60\,\text{min}$ depending on cost function and start point	Better than fmincon(), but not always satisfying.
fmincon() with multiple startpoints	Multiple local optimisations (one of the start points is specified, the other are randomly chosen)	Few minutes to hours depending on number of startpoints	Satisfying if enough startpoints or specified startpoint near the global minima.
ga()	Genetic algorithm	$\sim 60\,\text{min}$	Since the algorithm is based on stochastic methods, the solutions are not always satisfying. Increasing the number of generation allows better performance but increases the solving time.
particleswarm()	Particle swarm algorithm	$\sim 5\,\text{h}$	Satisfying
surrogateopt()	Surrogate optimisation	$1 - 60\,\text{min}$ depending on optional parameters	Satisfying, but slightly worse results than the particle swarm method. Effective for time-consuming cost functions.

Knowing the performance and time-need of each tested optimization algorithm, the following optimization methods were chosen:

- for the PD-controller a first solution for a speed of $0.5\text{m}\,\text{s}^{-1}$ was found using the particle swarm algorithm. This gives a good solution, but is highly time-consuming. Then, in order to find parameters for other gaits (other desired speeds), the first solution was used as a specified start-point and a multiple start-points local optimisation was performed (5 to 10 different start-points, one of them is specified, the other are randomly chosen). This allows to find satisfying parameters in a reasonable solving time.

- for the VMC however, local optimisation with multiple start-points did not permit to find good enough solutions, which probably mostly due to the fact that the global minima for two different targeted speeds are further away one from another than in the previous case. Since the cost function's solving is time demanding,the best method was to apply surrogate optimisation. By changing the optional parameters of the function *surrogateopt* (maximum number of objective function evaluations, initial points, etc.), it was possible to find a good comprise between the optimization performance and the solving time.

The code architecture used for the optimisation is represent in Figure 2.9. Note that *eqns opti* defines the cost function and *solve eqns* is used to solve the dynamic ODE.

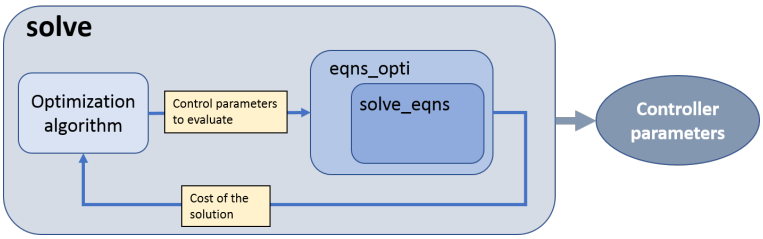


Figure 2.9: Code Architecture for Optimisation

2.4 Perturbation implementation

In order to test the robustness of the controllers, perturbations are applied on the biped. Two different ways to do this are considered:

- adding a constant horizontal force that pulls or pushes the the hip of the robot.
- adding noise on the actuators' torques.

Note that the perturbation force is only applied after a defined number of steps so that the biped has time to enter in a steady gait.

Horizontal force on the hip

Since the perturbation is a virtual force on the hip, the same method as for VMC is used. Hence:

$$u_{noise} = B^T * J^T * F_{noise}$$

where: J $= \begin{pmatrix} l1\cos q_1 & 0 & 0 \end{pmatrix}$, u_{noise} is the perturbation on the actuators (2x1 matrix), B is a matrix that maps the joints' torques to the actuators' torques and F_{noise} is the magnitude of the perturbation force. The perturbation is a backward force if $F_{noise} < 0 \quad N$ and a forward force $F_{noise} > 0 \quad N$.
Note that the force is constantly applied to the hip (no spikes or magnitude change), hence this models the case where the biped would have to pull or to push an object.
Eventually, to test and compare the robustness of the controllers, for every selected gait the magnitude of the force was slightly increased until the biped falls.

Noise on the actuators' torques

The second way to test the robustness of the controllers is to add noise on the torques produced by the actuators. This is a really common situation both motors and captors are getting more precise, but actuators still have complex issues in term in control, that involves a noise on the output of the motor. At every iteration step of the ODE solver *ode45*, White Gaussian Noise and a constant bias are added to the two control torques $u1$ and $u2$.
Two noise parameters can be tuned in order to test the robustness of the gait: the SNR of the White Gaussian Noise and the value of the bias.
This technique would work well in theory and would be very interesting, however adding a random noise to

the torques makes the solving of the motion ODE very hard for the function *ode45*. The computation becomes extremely time-consuming and therefore it has been decided to focus on the constant force method and not to implement the noise on the actuators' torques.

2.5 Analyze methodology

To analyze the different controllers a pool of solutions provided by each one will be presented. They are valid: that means that the biped doesn't fall before 30 steps and that the torques limitation (in term in intensity and spikes) are respected.
In order to compare and evaluate them different main magnitudes will be used :

- Achieved speed $[m/s]$: the speed that the biped achieve in steady state (compare it to the targeted speed)
- Maximum speed, minimum speed and interval speed $[m/s]$: the minimum and maximum speed achieved between all the test and their difference
- Maximum perturbation tolerated :
  - Maximum force $[N]$: Maximum positive force that we can put on the hip
  - Minimum force $[N]$: Maximum negative force that we can put on the hip
- Cost of Transport $[J/m]$: Efficiency in term of energy of the displacement

$$CoT = \frac{\int_0^T max(0, u_1(\dot{q}_1 - \dot{q}_3))\mathrm{d}t + \int_0^T max(0, u_2(\dot{q}_2 - \dot{q}_3))\mathrm{d}t}{x(T) - x(0)} \tag{2.6}$$

3. Results

3.1 Energy losses due to the impact

At each step the biped loses some energy due to the impact with the ground.
Potential energy stays constant because the impact map only changes the indices of the angles during the mapping and the two legs have the same mass, therefore no change can occur.
It is different for kinetic energy. The swing foot has a certain velocity when it touches the ground and it suddenly goes to 0 (it becomes the stance leg). Although the conservation of the momentum gives an impulsion to the new swing foot, this does not imply that the velocity is fully transferred to the other foot. That results in a loss of kinetic energy.

For $q = [\pi/6, -\pi/6, \pi/10]$ and $\dot{q} = [1, 0.2, 0]$ the biped loses **28%** of kinetic energy.
The following graph shows the percentage of kinetic energy losses as function of the angle (in radians) between the two legs (α) :

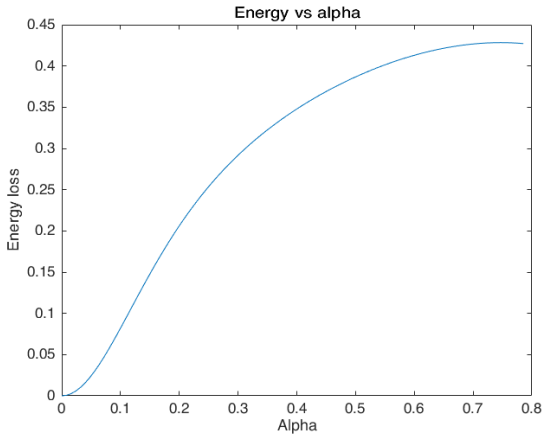


Figure 3.1: Percentage of kinetic energy loss in function of alpha (angle between the legs)

From this plot, it can be concluded that the energy loss increases with alpha, the angle between the legs, meaning that a bigger step length results in a higher energy loss.

3.2 PD controller

This section will present the results of the PD controller. The graphs plotted use the parameters of the highest speed, other plots can be found in files.

Overall performance

Table 3.1: Global performances of the PD controller

Max speed $[m/s]$	0.74
Min speed $[m/s]$	0.38
Mean positive force tolerated $[N]$	1.54
Mean negative force tolerated $[N]$	-17.01
Speed interval $[m/s]$	0.36
Mean CoT $[J/m]$	25.2

Here the mean positive and negative forces tolerated correspond to the average (for every gait) of the minimal and maximal forces that could be applied on the hip of the robot without falling.

The different solutions and theirs parameters

The optimisation on the control parameters was performed for an array of different speeds between $0.2\,\mathrm{m\,s^{-1}}$ and $1.5\,\mathrm{m\,s^{-1}}$. However, as shown in Table 3.1, the PD-controller cannot achieve such low or high speeds. If the optimisation is run with a targeted speed higher than $0.74\,\mathrm{m\,s^{-1}}$ or lower than $0.38\,\mathrm{m\,s^{-1}}$, the biped would either fall or saturate at an achievable speed.
After performing the optimisation for several targeted speeds, a choice of 9 gaits that give a rather stable behaviour were selected and are presented in Table 3.2.

Table 3.2: Parameters found with optimization of the PD controller and their results

Achieved Speed (in steady state) $[m/s]$	$K_{ptorso}[Nm/rad]$	$K_{pspread}[Nm/rad]$	$K_{dtorso}[Nms/rad]$	$K_{dspread}[Nms/rad]$	$qt_{des}[10^{-3}rad]$	$spread_{des}[rad]$	**CoT** $[-]$	**Max perturbations tolerated**	
								Max Force $[N]$	Min Force $[N]$
0.38	60	6.6	1.7	1.4	0	0.72	27.4	1.69	-3.3
0.48	84	21.1	9.2	6.8	0.2	0.96	21.3	4.96	-15.7
0.53	120	21.5	10.4	4.8	8.8	0.79	30.0	0.01	-33.9
0.54	56	23.9	8.3	3.8	1.1	0.62	28.4	4.5	-14.8
0.60	69	26.8	9.3	4.6	24.8	0.65	36.1	0.01	-23.2
0.61	75	18.9	17.8	2.8	6	0.56	22.8	2.53	-22.4
0.69	54	23.7	11.0	3.6	31.7	0.54	20.6	0.11	-22.8
0.73	53	20.5	10.2	3.1	46.3	0.59	21.2	0.01	-16.9
0.74	979	5.7	211.2	0.3	130.1	0.34	19.3	0.01	-0.1

State of the robot over time

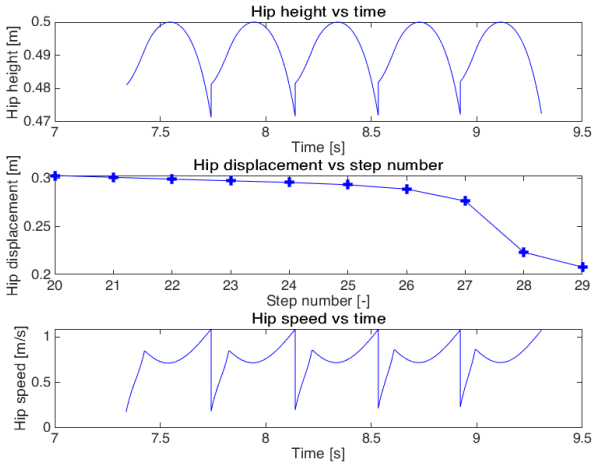


Figure 3.2: Displacement of the robot's hip over time

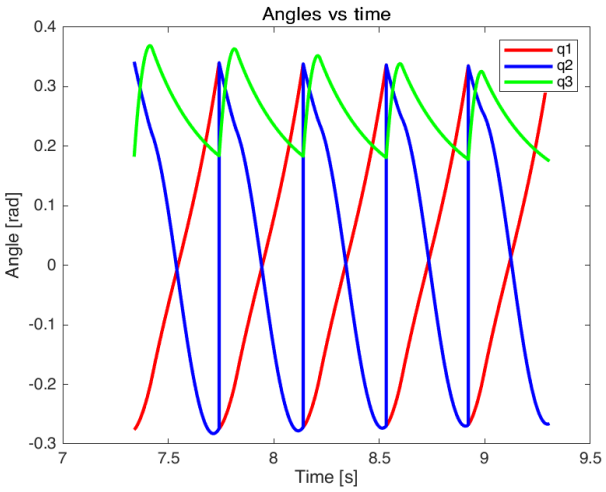


Figure 3.3: Evolution of the joint angles over time

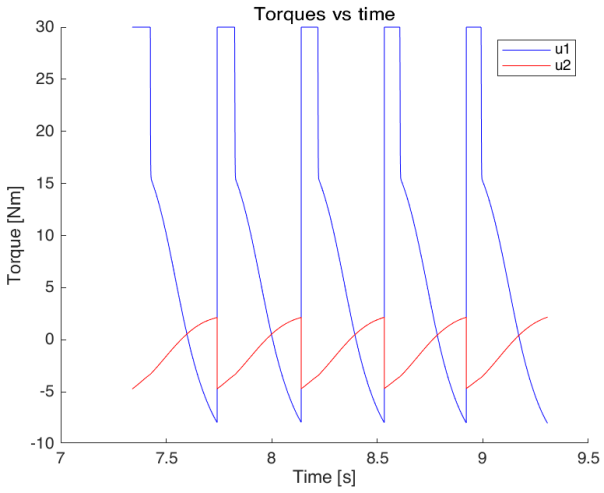


Figure 3.4: Evolution of the actuators' torques over time

Others useful plots

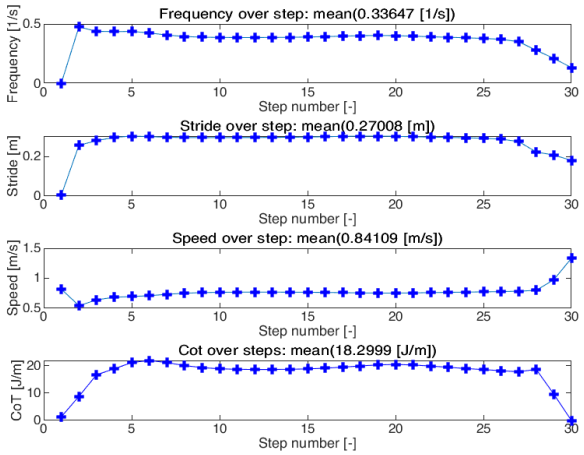


Figure 3.5: Evolution of the frequency, stride, speed and CoT vs the step number

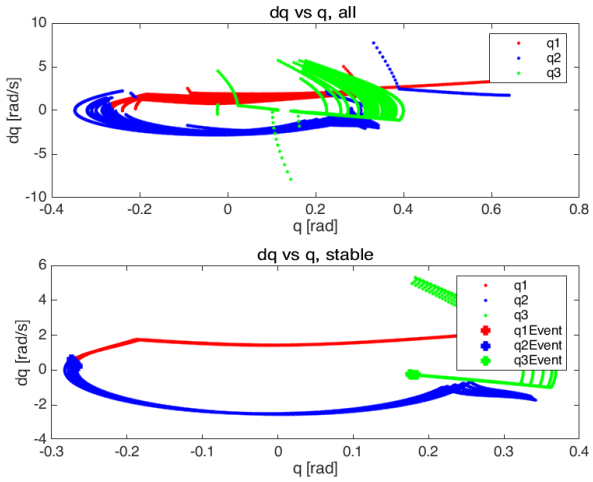


Figure 3.6: Relation between $\dot{q}$ and q (second graph corresponds to the stabilized gait)

3.3 VMC controller

This section will present the results of the VMC controller. The graphs plotted use the parameters of the highest speed, other plots can be found in files.

Overall performance

Table 3.3: Global performances of the VMC controller

Max speed $[m/s]$	1.05
Min speed $[m/s]$	0.36
Mean positive force tolerated $[N]$	24.3
Mean negative force tolerated $[N]$	-23.5
Speed interval $[m/s]$	0.69
Mean CoT $[J/m]$	67.1

Here the mean positive and negative forces tolerated correspond to the average (for every gait) of the minimal and maximal forces that could be applied on the hip of the robot without falling.

The different solutions and their parameters

As for the PD controller (see 3.2), the control parameters were optimized for a selection of speeds that are presented in Table 3.4

Table 3.4: Parameters found with optimization of the VMC controller and their results

Achieved Speed (in steady state) $[m/s]$	$k_1[N/m]$	$k_2[N/m]$	$k_3[N/m]$	$C_1[kg/s]$	$C_2[kg/s]$	$C_3[kg/s]$	qt_{des} $[10^-3 rad]$	$x_{des}[rad]$	$v_{target}[m/s]$	**CoT** $[-]$	**Max perturbations tolerated**	
											Max Force $[N]$	Min Force $[N]$
0.36	7401	6562	4360	907	229	994	0	0.26	0.3	71.5	42	-4
0.49	9439	7930	10000	617	156	998	0	0.25	0.4	99.5	49	-16
0.50	2981	4308	6042	401	218	979	104	0.34	0.5	46.5	52	-34
0.60	7508	7384	5653	768	399	998	191	0.34	0.6	51.3	35	-15
0.64	1805	2760	5407	365	59	831	1	0.33	0.7	89.4	12	-30
0.80	7260	1648	7019	731	268	772	228	0.28	0.8	68.6	23	-23
0.90	7786	4913	10000	824	305	779	189	0.19	0.9	58.0	5	-23
1.00	9779	3390	6614	786	236	686	265	0.17	1	57.1	0.8	-18
1.05	68728	98770	49736	9589	1839	8451	366	0.21	1.1	62.2	0.1	-49

State of the robot over time

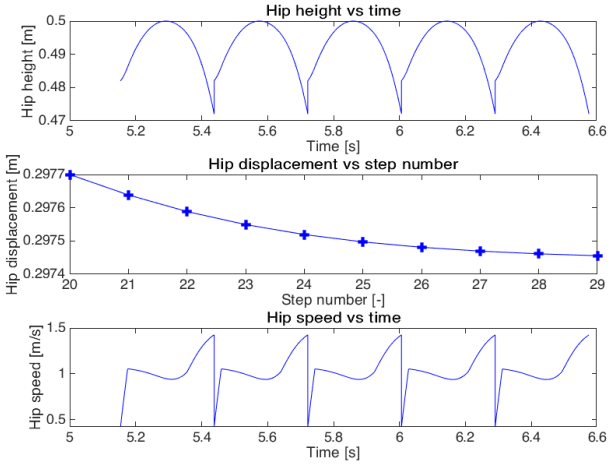


Figure 3.7: Displacement of the robot's hip over time

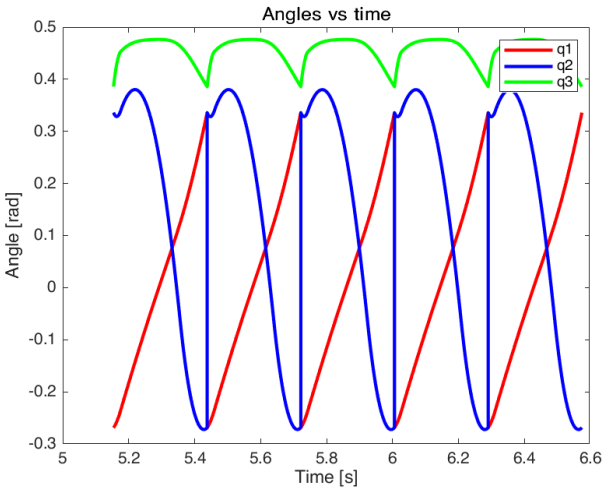


Figure 3.8: Evolution of the joint angles over time

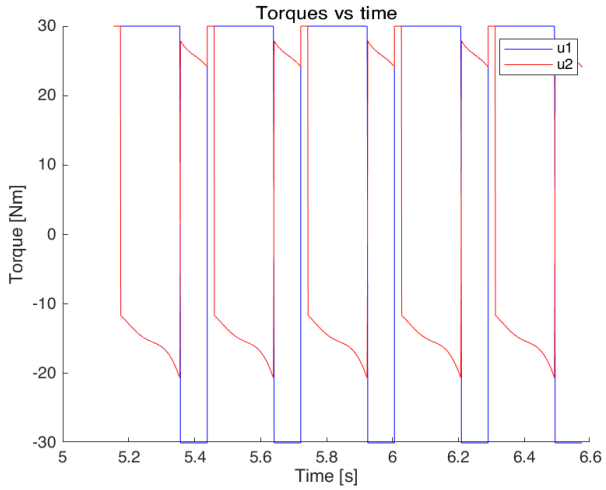


Figure 3.9: Evolution of the actuators' torques over time

Others useful plots

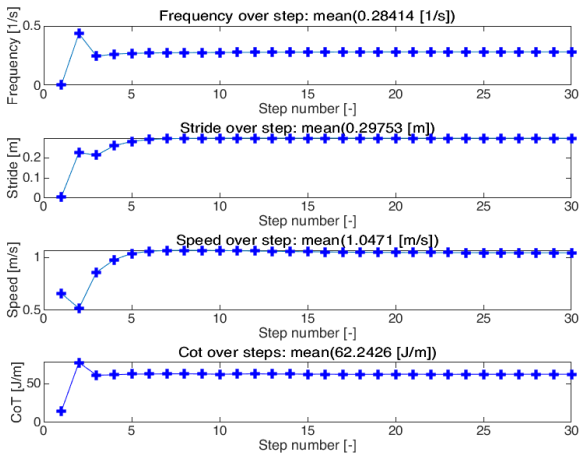


Figure 3.10: Evolution of the frequency, stride, speed and CoT vs the step number

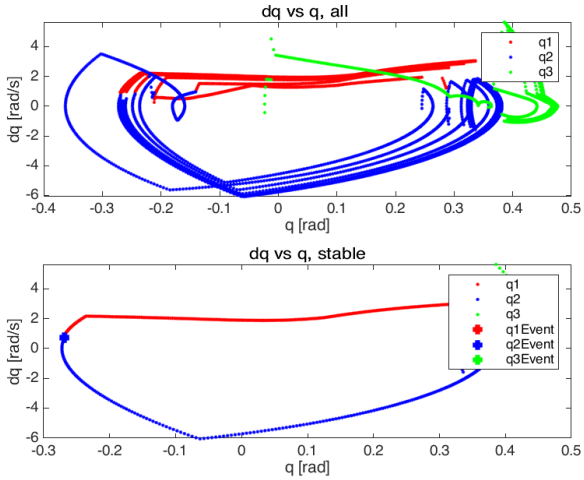


Figure 3.11: Relation between $\dot{q}$ and q (second graph corresponds to the stabilized gait)

3.4 DDPG controller

Three different agents are taken into account for the DDPG method:

- Reward Agent: the agent that achieved the highest reward function
- Fastest Agent: the agent that reached the highest velocity.
- Compromise Agent: an agent that a good comprise between the reward function and the speed

Agents' performance

Here are the performance achieved by the three selected agent:

Table 3.5: Results of the agent

	Reward Agent	Fast Agent	Compromise Agent
Speed $[m/s]$	1.75	1.78	1.76
Max Force Tolerated [N]	21	21	21
Min force tolerated $[N]$	-65	-65	-65
Mean CoT $[J/m]$	109	97	109

3.5 State of the robot over time

The plots shown in this section correspond to the *Reward Agent*.

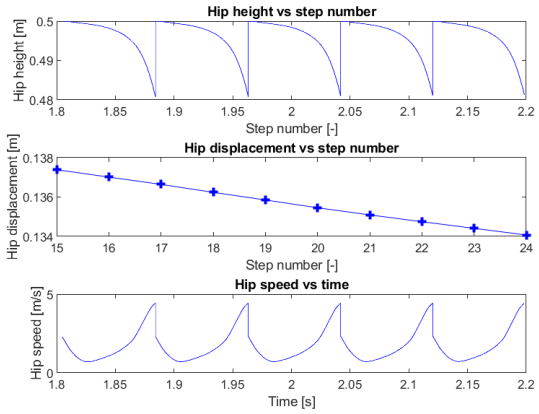


Figure 3.12: Displacement of the robot's hip over time

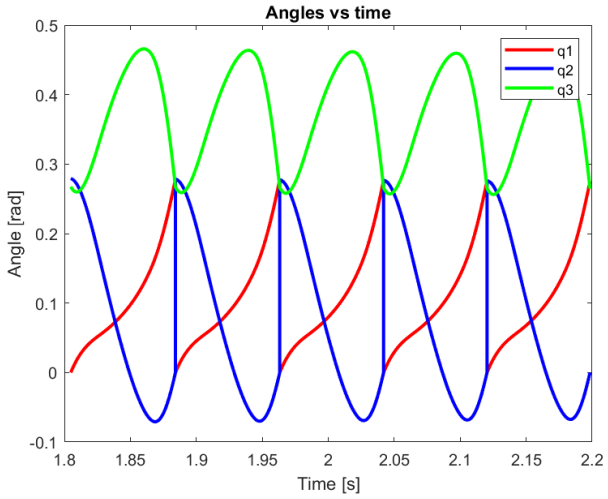


Figure 3.13: Evolution of the joint angles over time

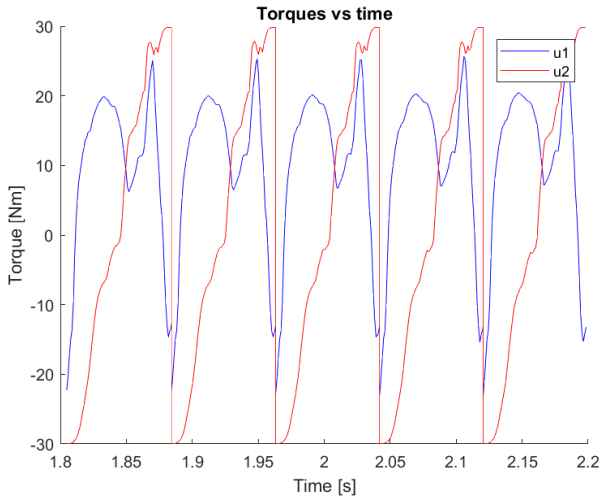


Figure 3.14: Evolution of the torques angles over time

3.6 Other useful plots

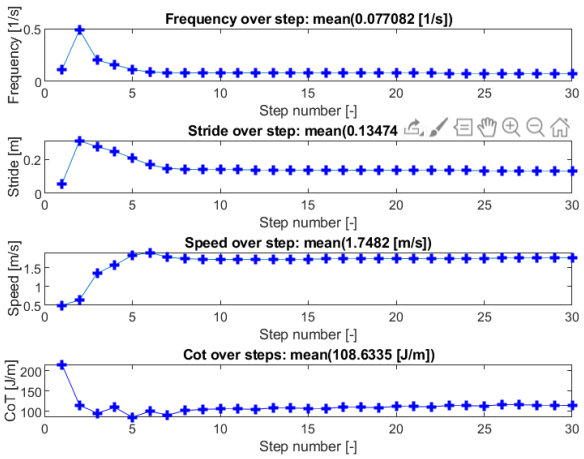


Figure 3.15: Evolution of the frequency, stride and CoT vs step number

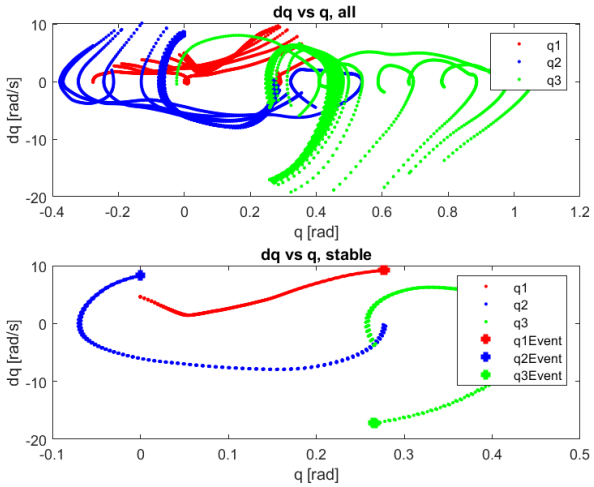


Figure 3.16: Relation between $\dot{q}$ and q

4. Discussion

4.1 Energy loss in the model

The model loses energy at each step of the biped. The energy lost is kinetic energy as potential energy is conserved. Obviously the loss is due to the fact that the velocity of the swing leg is not fully transferred to the former stance leg in spite of the conservation of the momentum. The losses increases when α, the angle between the legs at the impact increases. Our hypothesis to explain this phenomena is the following : the model is better suited to transfer horizontal velocities than vertical velocities. The greater α, the more important the vertical component is. This vertical component is poorly transferred (lost in the elastic shock with ground), which results in a loss of kinetic energy.

4.2 PD controller

The PD-controller has the advantage that it is relatively simple to implement (relies on simple mathematical equations) and that the computational cost is fairly low. However the achieved performance is poor.

Speed

The speed interval achieved by the PD-controller is narrow and the maximal velocity is fairly low. Those poor performances can mainly be explained by the fact that the controller is to simple and naive relatively to the complexity of the full system. Indeed, the PD-equations focus on only two variables (the spread angle and the torso angle.). However, several other parameters that are involved in the achievement of a good gait are omitted, like for example the z-coordinate of the swing foot or the trajectory of the hip during a step.
Furthermore, the naivety of the PD-controller has for consequences that the performances of the biped strongly rely on the optimization (as a matter of fact, it is very difficult, if not impossible, to tune the parameters by hand and find a good walking gait). Since the cost-function is non-convex, time demanding and complex to solve, having a controller that shows a strong dependence to its parameters' optimization has a performance degrading effect. This also explains why the achieved set of speeds do not follow a regular pattern (see Table 3.2) .
Finally, note that the desired speeds can be achieved with acceptable variation of torque (no spikes or brutal change during a step). (Figure 3.4). This is an inherent property of the PD-equations: if there is no sudden change in the state of the biped, the will not be any sudden change in the controller's output.

Stability and Periodicity of the Gait

As shown in Figure 3.2, the hip displacement at every step does not stabilizes even after 30 steps. This mean that the biped does not succeed in reaching a steady state. This effect can also be observed in Figure 3.5, where the plotted parameters (stride length, step frequency, speed and CoT) do not stabilize after 30 steps or

in Figure 3.6, where the joint angles velocities $\dot{q}$ still diverge for the same given angle q after many steps.
The lack of stability of the PD-controller is not very surprising since almost nothing is implemented in this controller to try to reach a stable gait. The only effort put in stability is done through the cost function of the optimization model: there is a cost on the hip height and the torso angle at every step with respect to a fixed and constant targeted hip height and targeted torso angle. Therefore a cyclic behaviour would minimize the cost function. However, as already discussed, the cost function is hard to minimize and the used tools are unable to find a point were it is exactly zero, hence a perfectly cyclic behaviour cannot be reached. Note that the parameters are optimized on 30 steps, therefore, the biped show acceptable performance on the 30 first steps but it falls at the very end due to its lack of stability.
An other interesting point is the fact that the PD-controller is able to reach a more stable gaits for lower velocities (it can walk more than 500 steps for some velocities around $0.5\,\mathrm{m\,s^{-1}}$). Two hypotheses might explain why the analysed gait is less stable :

- The initial state q_0 and dq_0 that is given to the biped was extracted from a gait at $0.5\,\mathrm{m\,s^{-1}}$. So, it makes sens that the gaits around that speed are more stable that others. With an other initial point the biped could possibly reach a steady state for the analysed velocity
- The analysed velocity is the maximal achievable speed, and in order to walk faster the biped needs higher torques and higher balance perturbations. Therefore it is more likely to fall and it is more difficult to reach a stable, cyclic gait.

Robustness Against Perturbation

Due to the naivety of the PD-controller discussed above and the inability of the PD-controller to bring the biped in a steady-state, the the resistance against perturbation is catastrophic (3.1).
Interestingly, the robustness against the force perturbation on the hip is better for backward forces than for forward forces. This is due to the fact that the center of mass of the robot is in front of the hip walking (there is a forward disequilibrium) and therefore a slight forward force is enough to make the biped fall.

Cost of Transport

The CoT of this method is shown in Figure (3.2). It is approximately constant at every step even though, as discussed above, the biped does not reach a true steady state.
A 70kg human walking at $1.7\,\mathrm{m\,s^{-1}}$ has a metabolic rate of $P = 450$ W [2]. If this is value converted into the definition of CoT that is used in this report, this gives: $CoT_{human} = P/v = 450/1.7 = 264.7 \quad J/m$. Hence the CoT achieved by the PD-controller is fairly low compared to humans. This is due to two different factors:

- The torques output of the PD controller only saturates occasionally and is maintained at low values during most of the motion (Figure 3.4), which results in a low energy consumption.
- The model of the biped of its environment is much simpler that real world case of a human walking, therefore many energy loss sources are neglected. As a matter of fact the 2D biped does not need to use energy to stabilize in 3-dimensions, to bend knees or to move arms. Furthermore the metabolic rate considers the energy that is used by the whole human body, so some some energy will be wasted by processes that are not linked to motion (digestion, brain, etc.). Finally, the calculation of the CoT of the biped does not account for energy losses in the actuators (energy efficiency considered to be 100%), whereas the value considered for the human does take into account the energy that is wasted in muscles (the metabolic rate considers the input power and not the output power)[2].

4.3 VMC controller

The VMC controller is intuitive to develop as it is not based on the dynamic of the robot. The simplicity of design allows to develop easily a relatively complex system that takes a diversity of parameters for the control. This results in good and robust performance.

Speed

The interval of speed achieved by the VMC is important (3.3). The different forces and their specific roles enables a variety of working combination and take into account several parameters that are involved in the gait, which leads to satisfying performance.
Moreover the same table indicates that the tracking is good for most of the tested velocities. This is not a big deal for this kind of controller as there is a specific force whose objective is to pull the robot at the selected speed.

Stability and Periodicity of the Gait

The figure 3.11 in the stable gait shows clearly that the biped entered in a steady stable gait. In the lower graph the relations between q and $\dot{q}$ superpose finely for each step, meaning that the gait is cyclic over one step period. The graph of the hip displacement (3.7) show that the gait is stable beyond this period and that the robot doesn't fall at the last steps. The proprieties of this gait can be seen through the periodicity of most of the graphs (3.7 or 3.8).
This stable gait is the result of the combination of the different forces : as each of them is enforcing a goal only the right combination subsists.

Robustness Against Perturbation

The VMC provides good results in term of resistance against perturbations (3.4). This was expected because VMC is a robust method of control and the implementation relies in a variety of stabilizing forces that ensures that it will do the right movement without falling. Combination of spring and damper are known to be effective. Surprisingly, the tolerance against a perturbation force is the same for a negative force or positive one. That may be explained by the fact that this kind of controller has not to fear to the slight imbalance due to the head.

Cost of Transport

The cost of the transport of the VMC is high (3.3). This is the consequence of the method itself : the VMC does not take in consideration the dynamic of the robot for the control. Therefore the VMC will use a lot of costly (in term of energy) movements and won't use the free dynamics of the robot at its advantage. (3.4). For the same reason the torques are subject to high variations (3.9).
A way to obtain a lower CoT using VMC could be to introduce a term in optimization's cost function that penalizes the biped for high CoT.

4.4 DDPG controller

As we can see in Figure. 2.8, the last simulation converged to good rewards while exploring many possibilities. Underestimating the success of the training, the script was set to save every Agent winning more than 100 of

reward. That led to 3570 saved various agents whose analysis is not trivial, since the dimension of the folder is more than 2GB and loading one agent require more than 4 seconds in a *Macbook Pro 2015.*

As said before, the simulation was set up to save videos every 100 episodes. Looking at those videos and at the reward plots to understand what is happening is pretty straight forward.

4.4.1 Videos

The video were saved with a frame rate of 20, while the simulation was running at 100 hz, so the video, during max. 20 seconds, is five time slower than reality. Here are some highlights:

- Even tough the biped is trained from the PD, at the beginning, free of it's exoskeleton and trying to experiment, the biped is not stable.
- from episode 1300 (video 13), the biped start to learn to move using the swing foot properly. It seems that the favorite gaits are simile-passive gaits, were the biped tilts the torso forward and try not to fall by quickly swinging the leg to the vertical positions, that allows very fast gaits.
- From video 19 the biped starts falling, and get negative rewards, in this situation the DDPG goes to a totally new direction. In video 21, it can be observed that the biped is trying to use the motor to balance the swing foot and get to the maximum possible stride opening. This can also be seen in videos 30 and 31. This strategy, even though very interesting, is very expensive in the *ActionPenality*, so the biped abandons it.
- Nice open gaits can be seen in videos 37 and 38, then it is dark for many videos. In the meanwhile the robot learns to strive for survival.
- Video 66 shows how good the DDPG is in experimenting new solutions. How would a limping biped walk?
- 95 and 96 are also worth viewing.

4.4.2 Agents

Getting back on the agents, in the scope of a general analysis, one agent out of ten were tested for surviving more than 100 seconds. 231 agents passed this selection, showing how the results obtained by the *bigmodel* in only for seconds are good in generalization.

Then the agents were classified based on their speed and reward. Figure 4.1

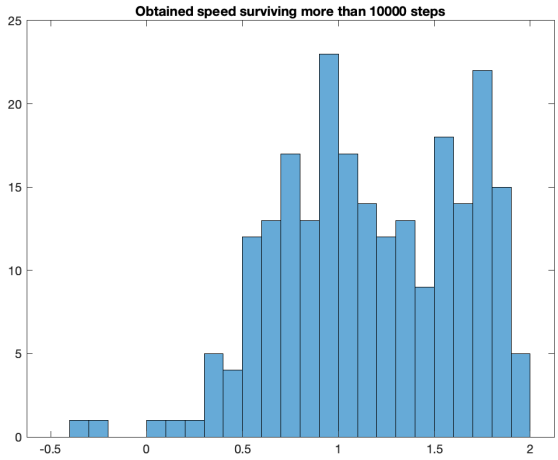


Figure 4.1: Histogram of the supposed stable gaits on 231 selected Agents

And three agents were chosen; maximising the speed, the reward, and the product of the two, as it can be seen in Figure 4.2

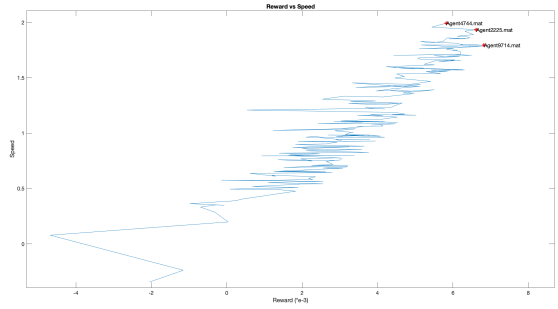


Figure 4.2: Speed vs reward

Speed

As shown in the histogram, the DDPG can achieve almost any speed between 0 and 2, although we suppose that those slow gaits with few reward and speed might be not periodic. The speed of 2 is a really great result overthrowing the traditional controllers analysed before. More will be said in section 4.5. An interesting question is still open: since in the state two free variables were let empty, how can they be used to track a velocity or a trajectory?

Stability and Periodicity of the Gait

Since the method doesn't provide any theoretical convergence, no certitude about the periodicity of the gait can be guarantied. But as seen in the plots (Figure 3.14), Figure 3.13 and Figure 3.16), the systems seems to achieve periodicity.

Robustness Against Perturbation

The three different agent show a really strong resistance against the perturbation that can be applied on them (3.5). This results is not surprising because the training was done with a permanent noise, by way of consequence, the system has become robust.
The maximum positive perturbation is a lot smaller than the negative one. That can be explained by the fact that the three agent move at high velocity and incline their torso to be more efficient. As consequence their stability in the positive direction is lower than the negative one (that first has to slow them, and then make them fall).

Cost of transport

Since the reward function is not optimized for providing a great CoT, the Agents try to maximize other parameters not caring about the ones that are not specified (this is a clear problem often discussed in philosophy of AI). The CoT ends up being pretty high, in spite of *ActionPenality*. Because of the high speed achieved by the system, the result is not comparable to the other methods. An interesting idea for a novel simulation could be to compose the reward function including a strong term taking consideration of the CoT.

4.5 Comparison

The overall results of the different methods are summed up in the following table :

Table 4.1: Comparison of the results for each method

	PD	VMC	DDPG
Speed interval $[m/s]$	0.36	0.69	>1
Max speed $[m/s]$	0.74	1.05	1.78
Max force tolerated (mean) $[N]$	1.54	24.3	21
Min Force tolerated (mean) $[N]$	-17	-23.5	-65
Mean CoT $[J/m]$	25.2	67.1	105

First, let conduct a comparison between the two standard methods (VMC and PD).
VMC shows better overall results : it can achieve desired speeds more accurately and can achieve faster gaits. This had to be expected since VMC is takes into account a complex combination of variables of the robot's motion where the PD controller is more restrained. VMC is able to track speeds where the PD fails: VMC has a virtual force dedicated to pull the biped at a desired speed whereas the PD has to find a complex combination of Kp and Kd in order not to fall and to walk at the same time. Moreover, VMC shows better robustness against perturbations; this is due to the fact that some virtual forces are dedicated to the stabilization of the robot.
However, when considering the CoT and the torques' abrupt changes, the PD yields better results. This is a consequence of the absence of the dynamical model in the VMC design, while the PD design is based on the dynamics. As a result PD method interferes less with the natural dynamics of the biped.
One way to achieve higher speeds for both methods may be to allow a higher angle for the torso in the optimization process. In this implementation the torso angle was maintained between 0 and $\pi/4$.
To conclude, PD design gives worst overall results than VMC design but is more energy efficient. This is represented in the following graph :

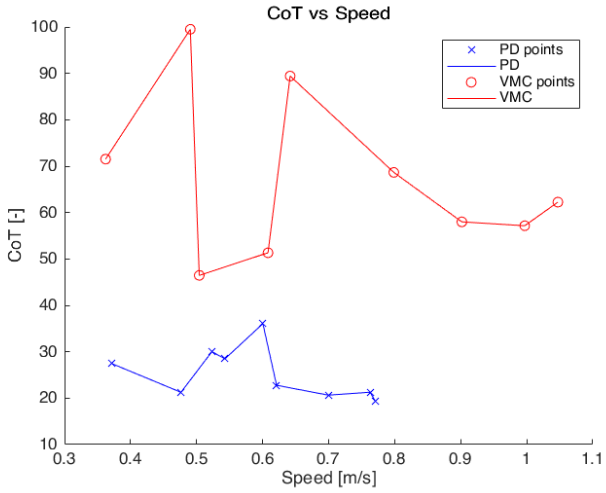


Figure 4.3: Cost of transport vs speed for PD and VMC

The same graphs can be used to find a relation between the CoT and the speed for both controllers. At first sight they don't seem to be an obvious dependencies, but one can speculate. For the PD the CoT has higher values for mean speeds, that may be because in the extreme cases during the optimization process the PD had to take advantage of the dynamic of the robot to be able to walk, and doing so it developed a energetic favorable gait. In the mean values the robot did not need the help of its own dynamics and so the optimizer without taking care of it. For the VMC it is bolder to present an explanation but we suggest that for higher speeds the CoT is lower because the optimization generated a higher angle for the torso to achieve better results, which resulted in a higher energy efficiency. Note that those two explanation are hypothesis and were not verified.

Now let compare the results of the two classical control methods with the DDPG.
First of all, in term of speed performance, one can see that the Deep Reinforcement Method yields much better results: a much higher maximal speed is achieved and the speed interval is larger. This can be explained by the fact that the DDPG controller is not constrained to simple mathematical laws like in classical control, but is free to explore more motion possibilities and torques combinations, until better performances are reached.
Regarding stability and robustness, the DDPG method shows similar results to the VMC method, since both are able to bring the biped in steady state and are robust against perturbations. One can see that the DDPG shows better robustness to backward forces than VMC, this is mainly due to the fact that only high velocities have been selected for the analysis of the DDPG. Therefore, the biped is subject to a strong forward force in order to move at high speed and the backward perturbation forces are overtaken.
Finally, the DDPG has a high cost of transport compared to the other methods. This again can be partly explained by the fact that only the fast agents are selected for the analysis. It is suspected that a such a high velocity requires a high energy consumption.

5. Conclusion

A kinematic and a dynamic model for the biped have been developed. To allow the robot to walk an impact map has been generated.
To control the bidep three different methods of control have been proposed : a PD controller, a VMC controller and a DDPG. The parameters of the PD and the VMC have been found through an optimization process that has been run under different conditions to proposes a diversity of gaits. The results of the different methods are summed up in the next table (same table as in the analyze chapter) :

Table 5.1: Comparison of the results for each method

	PD	VMC	DDPG
Speed interval $[m/s]$	0.36	0.69	> 1
Max speed $[m/s]$	0.74	1.05	1.76
Max force tolerated (mean) $[N]$	1.54	24.3	21
Min Force tolerated (mean) $[N]$	-17	-23.5	-65
Mean CoT $[J/m]$	25.2	67.1	105

As discussed in this paper the VMC produces better results in term of velocity and stability than the PD but implies a higher cost of transport because the dynamic is not considered in this model. The first place in terms of pure performance is owned by the DDPG: the maximum speed generated by this method is incredibly high and the gait is considerably stable and resistant. On the other hand the CoT provided by this solution is important, as we didn't include this parameter in the reward function. One can imagine that with a more complex reward function and with more computational resources to compute a solution, the DDPG could generate even better results. We overall think that the decision of switching from the *smallmodel* to the *bigmodel* allowed us to get good results of which we can be happy. It would be although interesting to add other deep-learning elements in the model to see how the results would change (dropouts? attention in the critic? more actors pooling for the action?), also the new algorithms of UDRL [4] seem very promising.

As students, this work was our first experience with a concrete system to control and optimize. To present this results, we had to investigate and learn, mainly from scratch, several concrete techniques, as optimization and reinforcement learning.

External Files

Link to the videos and results :
https://drive.google.com/drive/folders/1qenjORdCHLljTt2KNs7ebmt23KKPHqHM

List of Figures

List of Tables

Bibliography

[1] Pratt et al, *Virtual Model Control: An intuitive approach for bipedal locomotion*, The International Journal of Robotics Research, Vol. 20, No. 2, 2001

[2] Radhakrishnan, *Locomotion: Dealing with friction*, Proceedings of the National Academy of Sciences. 95 (10): 5448–5455, 1998.

[3] McGeer, Tad. *Stability and control of two-dimensional biped walking*, Center for Systems Science, Simon Fraser University, Burnaby, BC, Canada, Technical Report 1, 1988.

[4] Schmidhuber, *Reinforcement Learning Upside Down: Don't Predict Rewards – Just Map Them to Actions*, 2019, arXiv:1912.02875

[5] Silver et al, *Deterministic Policy Gradient Algorithms*, http://proceedings.mlr.press/v32/silver14.pdf