Learning Optimal Gait Parameters and Impedance Profiles for Legged Locomotion

Elco Heijmink[1,2], Andreea Radulescu[2], Brahayam Ponton[3], Victor Barasuol[2], Darwin G. Caldwell[2] and Claudio Semini[2]

***Abstract*— The successful execution of complex modern robotic tasks often relies on the correct tuning of a large number of parameters. In this paper we present a methodology for improving the performance of a trotting gait by learning the *gait parameters*, *impedance profile* and *the gains of the control architecture*. We show results on a set of terrains, for various speeds using a realistic simulation of a hydraulically actuated system. Our method achieves a reduction in the gait's mechanical energy consumption during locomotion of up to 26%. The simulation results are validated in experimental trials on the hardware system.**

I. Introduction

Legged robots can perform complicated tasks, such as running, balancing and locomotion over challenging terrains, where they offer a clear advantage over wheeled platforms. Yet, in real-world scenarios, biological systems easily outperform legged robots. One of the main reasons is that selecting the "best" gait is not trivial, as it requires the generation and execution of complex behaviours. The success of these behaviours often relies on tuning a large number of parameters, which can be a lengthy empirical procedure, requiring an expert user. In the absence of such an expert the task becomes significantly more difficult and, sometimes, even impossible.

To obtain locomotion policies, multiple learning algorithms, such as Hill Climbing, Amoeba [1], Genetic [2] and Policy Gradient algorithms [3] were used on a commercially available quadruped (Sony's AIBO) and comparisons were made based on factors such as learning time, final policies and performance in the presence of noise [4]. The comparison showed that in this kind of task the Policy Gradient algorithm is superior on all factors. More recent examples of policy learning based algorithm can be found in Policy Learning by Weighting Exploration [5], Relative Entropy Policy Search [6] and The Policy Improvements with Path Integrals (PI^2) [7]. These policy search methods inspired by expectation-maximisation are expected to be able to learn a high dimensional policy effectively [8].

The PI^2 algorithm is employed in [9], [10], where the basic idea is to use a well understood dynamical system (from an analytic point of view) and modulate it such that it achieves a desired attractor behaviour. Dynamic movement primitives are used to encode the dynamics (i.e., a special case of parametrised policies). The PI^2 algorithm is applied in this context for optimal planning and/or gain scheduling. Since this method does not require matrix inversion or gradient learning rates, the update equations are simple and numerically stable. It also scales very well to high dimensionality problems. In [11] the PI^2 algorithm is applied not only for planning, but also to optimise variable gain feedback controllers. This makes it a good candidate for this work, with the aim of creating a practical method to automatically tune the locomotion parameters, without the need for expert knowledge.

Using compliance can give robotic systems the ability to increase their gait stability and energy efficiency. This can be implemented by using compliant elements (*passive compliance*) [12], [13], or by emulating their behaviour using software (*active compliance*) [14], [15]. While the latter approach can modulate the impedance in an on-line fashion, for a theoretically infinite range and speed, it is unable to store energy or exploit the natural dynamics. Defining an optimal impedance profile for a legged robot is not trivial and the complexity of the task scales with that of the design. Reinforcement learning provides us with a feasible strategy for determining such a profile, alongside other parameters of interest, avoiding the common practice requirement to hand tune these values.

Learning algorithms can require a large number of trials over a long time to find the optimal desired behaviour. A successful locomotion strategy will have to be able to handle a wide variety of terrain scenarios, which implies learning new parameters when changes in the operating conditions render the old values sub-optimal. Hence, the learning time becomes a critical factor in the ability to find an optimal strategy for large numbers of tasks and scenarios.

In this work, we improve on the methodology published in [16], which focused on improving a trotting gait [17] by modulating the end-effectors' impedance. We improve the performance of the trotting gait for our quadrupedal system, replacing the manual gain tuning with our learning procedure. The main contributions of our work are: We focus on a previously unexplored application of the approach, by setting the secondary goal (after achieving the desired travel speed) to improve energy efficiency. The procedure delivers the learned: *gait parameters* (i.e., step height, step length, stride frequency, duty factor), *feet impedance profile* in Cartesian coordinates, and *gains of the control architecture* (i.e., trunk controller). We augment the abilities of the

[1]BioMechanical Engineering, Delft University of Technology, Delft, The Netherlands E.Heijmink@student.tudelft.nl

[2] Dynamic Legged Systems, Istituto Italiano di Tecnologia, Genova, Italy. firstname.lastname@iit.it

[3] Max-Planck-Institut für Intelligente Systeme, Tübingen, Germany. brahayam.ponton@tuebingen.mpg.de

procedure by significantly improving the learning time (from over 90 minutes to under 10 minutes required for learning a new task). Our method achieves a reduction in the gait's mechanical energy consumption of up to 26%, compared to the hand tuned values provided by an expert user (as traditionally employed on the platform). Despite using only active impedance modulation, we noticed that part of this improvement is due to the learned impedance profile. We present results obtained in a realistic stimulation of the hydraulic, torque controlled quadruped HyQ [18] for a set of varying terrains and desired speeds. These simulation results are validated in experimental trials.

Adapting parameters based on the particularities of the task and environment was previously used for a biped [19] system, but the approach uses separate methods to individually adjust parameters of the running cycle. Similarly, in [20] the gait of a quadruped is adapted using a set of reflexes, each triggered by a specific process. In our work we tackle *automated learning* and *simultaneous tuning of the desired parameters*, based on proprioceptive information. The recent study in [21] adjusts a similar set of parameters for autonomous locomotion, but in a non-learning fashion and the approach relies on additional visual information.

The rest of the paper is structured as follows. In Section II we present details of our implementation, by introducing the controller framework used and the relevant details of the PI^2 algorithm in the context of our investigation. We also detail the cost function encoding the desired behaviour and discuss the steps taken to improve the speed of the learning procedure. Section III gives a brief overview of the hardware used and the experimental setup. In Section IV we present the results of our investigation in the simulation environment, while Section V shows the performance obtained on the hardware system. We follow up in Section VI with a brief summary and discussion of our results. Finally, in Section VI we present ideas for future research.

II. Technical Approach

We aim to improve the locomotion energy efficiency of the HyQ quadruped by learning the gait parameters, a variable impedance strategy and the gains of the controller architecture for a walking trot (at a desired velocity), while maintaining stability. The impedance profile in Cartesian space is mapped into the feedback gains of the PD torque controller of the joints, as discussed in [16].

A. The Reactive Controller Framework

The underlying gait is generated using our Reactive Controller Framework (RCF) [17], designed for robust legged locomotion. The RCF consists of two main modules: a motion generation and a motion control block. The former uses a network of nonlinear oscillators, one for each foot. The outputs of the oscillators are passed through a nonlinear filter, which results in an elliptic trajectory during the swing phase of each foot. In the stance phase, once the foot makes contact with the ground, the ellipses are cut (i.e., follow a straight trajectory). The oscillators are encoded in the form of gait parameters (i.e., step height, step length, stride frequency, duty factor). Using inverse kinematics, these ellipse trajectories are mapped into desired joint space trajectories. A trajectory tracking controller, incorporating a linear feedback loop, is then used to track these desired trajectories:

$$\tau = InvDyn(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) + K_P\mathbf{S}(\mathbf{q}_d - \mathbf{q}) + K_D\mathbf{S}(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \quad (1)$$

where $\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d$ are the desired joint positions, velocities and accelerations, $\mathbf{q}, \dot{\mathbf{q}}$ are the current positions and velocities, τ the generalised force vector and K_P, K_D the feedback gains of the impedance controller. Additionally, a trunk controller maintains the attitude of the robot and compensates for deviations in the roll and pitch angles of the trunk. The stabilisation forces from the trunk controller are mapped into joint torques, independently from the joint trajectories.

In the context of the RCF architecture described above three sets of *learning variables* are improved: i) *gait parameters* (i.e., step height, step length, stride frequency, duty factor), ii) *the end effectors' (i.e., feet) impedance profiles* (these are subsequently used to compute the feedback gains of the PD torque controller of the joints), and iii) *the feedback gains of the trunk controller* (for the pitch and roll angles). The impedance profile is encoded as a mixture module of von Mises basis functions [22], which can model periodic signals. The number of the basis functions was selected empirically as 10, based on the approach in [16].

B. PI^2 algorithm

As mentioned previously, we use the PI^2 learning algorithm in our study. At each episode, the method evaluates Q rollouts of the *learning variables* (listed above). These values are obtained by injecting zero-mean Gaussian noise into the current policy. We collect data on the trotting behaviour generated by these parameters (approx. 5 to 15*sec*), which is evaluated using the following cost function:

$$R = \phi + \sum_{i=0}^{P} r_i, \quad (2)$$

where (ϕ) is the terminal cost term and (r_i) is the running cost term (defined in Section II-C) at each time step $i \in [0, P]$. In the update rule, the contributions of each rollout set is scaled by their performance with respect to the cost [7].

In our implementation, at each iteration, the policy is updated using the current ($Q = 8$ rollouts). This requires a minimum of 8 rollouts before the first policy update can be applied, after which only two new rollouts are performed in between policy updates. Thus, we reuse the previous best 6 rollouts for the next policy update, which results in faster learning, due to the reduction in the data collection time, at each iteration.

C. Cost function design

The cost function (2) needs to encode the desired behaviour, with the weights of each term reflecting their importance. The running cost r is defined as:

$$r = r_s + r_e + r_{lim} + r_\tau + r_{ft}, \quad (3)$$

with:

$$r_s = w_s \left| 1 - \frac{v}{v^*} \right|, \; r_e = w_e \frac{\sum_{j=1}^{N} |\omega_j \tau_j|}{m \, v_*}, \quad (4)$$

$$r_{lim} = w_{lim} \sum_{j=1}^{N} f(\theta_j, \theta_{lim_j}), \quad (5)$$

$$r_\tau = w_\tau \sum_{j=1}^{N} \left\| 1 - \frac{\tau_{max_j} - \|\tau_j\|}{\tau_{max_j}} \right\|, \quad (6)$$

where the subscript i is omitted in each term for simplicity.

Here, r_s is the speed tracking error, calculated as the average of all instantaneous velocity tracking errors, with v and v^* representing the actual and desired forward velocities of the system, respectively. This simultaneously penalises the average forward velocity tracking errors and its variations. The energy efficiency cost r_e is measured as the mechanical Cost of Transport (CoT) [23], with ω the joint angular velocity and τ the joint torques, which are summed over the joints ($N = 12$) and m is the mass of the robot.

The cost in the joint limits r_{lim} uses a barrier function f (7), which tracks the distance between the current joint angle (θ_j) and the joint limit (θ_{lim_j}):

$$f(\theta_j, \theta_{lim_j}) = \exp(-(u(\theta_{lim_j} - \theta_j))^2), \quad (7)$$

where u is a scaling factor that defines the threshold value at which the barrier starts growing exponentially (e.g., $u = 14$ for 10°). The term r_τ penalises high torques and prevents them from exceeding the maximum allowed values τ_{max}.

The RCF contains reactive modules, such as the step-up reflex [24], which demand minimal foot tracking errors. Although these modules are not used in the learning, the result should be compatible with their operation. Therefore an extra cost r_{ft}, for the tracking of the end effectors (i.e., feet) ($M = 4$) is added:

$$r_{ft} = w_{ft} \sum_{k=1}^{M} \left| \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} - \begin{bmatrix} x_k^* \\ y_k^* \\ z_k^* \end{bmatrix} \right|, \quad (8)$$

where $[x_k, y_k, z_k]^T$ and $\left[x_k^*, y_k^*, z_k^*\right]^T$ represent the actual and respectively desired positions of the end-effectors in the base frame[1]. The terms w_s, w_e, w_{lim}, w_τ and w_{ft} denote the corresponding weights of each term of the running cost function. The values used in our experiments are $w_s = 0.0375$, $w_e = 0.0001$, $w_{lim} = 0.25$, $w_\tau = 0.0025$ and $w_{ft} = 0.25$.

The final cost in (2) is defined as: $\phi = \phi_{tm} + \phi_{fail}$, where ϕ_{tm} is a measure of the motion of the trunk and ϕ_{fail} is a Boolean variable which applies a large penalty cost in the case when the policy causes the robot to slip and fall. The motion of the trunk is calculated as:

$$\phi_{tm} = w_{tm} \mathrm{Var}(|p_{CoP} - p_{CoM}|), \quad (9)$$

where w_{tm} is the associated weight and p_{CoP} and p_{CoM} represent the positions of the robot's Centre of Pressure (CoP) and the Centre of Mass (CoM), respectively. Hence, (9) is based on the variation of the trunk motion estimator that will be presented in Section II-D. In our experiments we used a value of $w_{tm} = 1$. The values of the cost function weights were obtained empirically by starting with the primary objectives and gradually augmenting the cost with all the required terms. To account for the discrepancy between the ranges of individual non-weighted terms (e.g., magnitude of 1 for joint angles vs. one of 10^4 in the CoT) the weights were scaled accordingly and normalised.

[1]This frame's axes and origin are aligned to that of the robot base.

D. Improving learning speed

As mentioned previously, one of the important components in learning is the time to learn. Reducing the required exploration episodes is a crucial step towards achieving an autonomous adaptive strategy that could be employed on-line on the hardware system. As the PI^2 computation time is negligible, this duration represents the exploration time, during which the robot trots with real time factor of 1, collecting behaviour data (i.e., during rollouts).

To reduce the evaluation time of the rollouts, the cost function is constructed based on stable values. This is achieved in three ways. Firstly, instead of using the noisy direct measurement of the trunk motion (i.e. the roll and pitch angles), a stable estimator is used. This trunk motion estimator is chosen as the distance between the CoP and the CoM. This value shows a strong correlation with the peaks of the roll and pitch of the trunk, but is more steady. Secondly, we discard the data collected at each iteration while the robot was still accelerating from zero towards the desired limit cycle. Finally, we remove the effect of small variations in the cost function by ignoring values below thresholds on: the foot tracking ($0.04\,m$), the speed tracking error (5%), the energy consumption ($70\,J/m$), and the trunk motion ($0.01\,m$).

As described in Section II-B, the algorithm generates rollouts by injecting Gaussian noise into the current learning policy. Unfeasible values for the *learning variables* (e.g., negative duty factor) are excluded by imposing an admissible region for the sampling. The rollouts with high costs will be obviously rejected after each iteration. As the trotting motion is periodic, after a few locomotion cycles, a reliable estimate can be calculated for the cost function of the current rollout. This allows us to discard early on any rollout that will result in a high cost. In this work, a rollout is aborted if the estimated cost after five cycles is more than 150% of the cost of the current policy.

To reduce the number of rollouts needed for the cost function convergence, the number of learning variables has been kept to a minimum. This was achieved mainly by learning the impedance only for the swing phase including the touchdown and liftoff, while using a constant impedance value for the stance phase. This is possible since the main advantages of the impedance optimisation arise from the effect of modulation during swing time (as described in Section IV-C). Additionally, the number of learning variables is reduced using critical damping. Hence, the damping profiles of the end effectors (i.e., feet) are indirectly improved, without increasing the number of learning variables.

III. Experimental setup

To evaluate the method proposed in this work, we use the 80 kg HyQ quadrupedal robotic system. The platform is fully torque controlled, with 12 hydraulically actuated joints. Each limb has 3 actuators: HAA (hip abduction/adduction), HFE (hip flexion/extension) and KFE (knee flexion/extension). The corresponding kinematic ranges are 95°, 120°, 120°, respectively. The completely stiff structure allows high bandwidths, while the hydraulic actuation gives a good power to weight ratio and the ability to deal with high impact forces. Hence, the system can perform highly dynamic locomotion tasks on different kinds of terrain. The platform constitutes a perfect candidate to demonstrate how a stiff robot can benefit from active impedance modulation to increase its energy efficiency, without compromising the performance.

The proposed learning algorithm was tested in simulation on two environments: i) flat terrain and ii) rough terrain with cobblestones (as depicted in Fig. 1), with the desired forward velocity varying between 0.2 m/s and 0.8 m/s. The methodology is evaluated on the learning time, the improvement in energy efficiency.

Fig. 1: The HyQ robot trotting on cobblestones in simulation.

The learning time is defined as the time until the total cost function converges (Section II-D). The energy efficiency will be measured using the CoT (Section II-C). We validate the results in terms of disturbance rejection using the Gait Sensitivity Norm (GSN) [25]. The GSN measures the influence of a disturbance on the current and future cycles. It shows a good correlation with the real disturbance rejection in limit cycle walkers and can be calculated from the observed behaviour. This makes the metric suitable for our scenario.

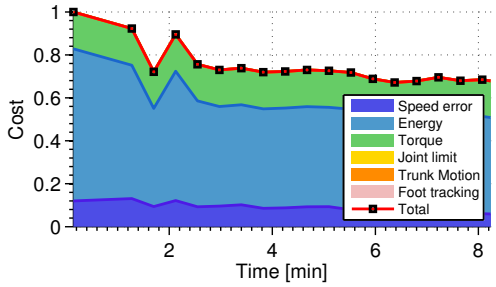
The behaviour obtained after learning is compared to the performance of a hand tuned strategy, as currently employed on the platform, in order to highlight the improvements. Traditionally, for each individual task, the expert user would be required to empirically find a new set of values. In practice, a feasible set of values could be applied to a wider range of tasks (i.e., for a desired velocity within a given range, on flat terrain, the same set of parameters would allow for a stable gait). Evidently, the performance of this generalised set would be worse than that of individually found values, for each respective task. In order to demonstrate the robustness of our approach we perform several learning trials for each task, starting from randomised set of values (including sets that would result in both stable and unstable gaits).

IV. Simulation Results

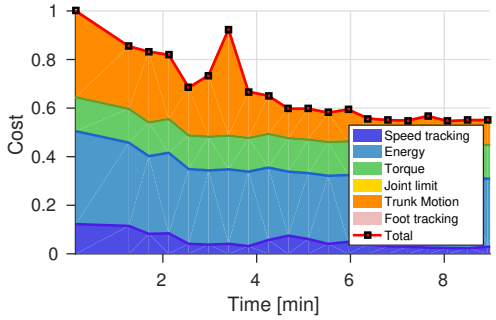
In this section we present the results obtained in simulation on the set of terrains and speeds used. We follow with a brief discussion, before proceeding to the hardware experiments section. In our setup, we employ a realistic model of the system with the Gazebo simulator environment [26], which is able to accurately represent real world scenarios.

A. Flat terrain

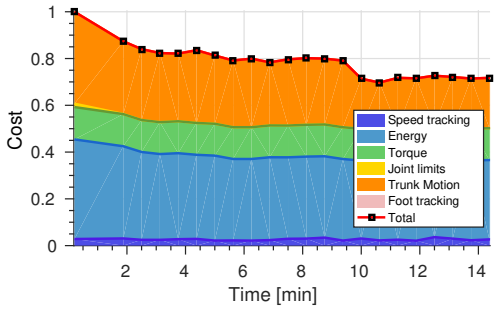
In the flat terrain there are no surface height variations, thus a reduced number of cycles is needed to calculate a reliable cost function. Hence, a rollout time of 7 *sec* is used. In this scenario the cost function converges relatively fast and shows a smooth learning curve, on average (Fig. 2). We note that the joint limits are not reached and the end feet tracking error is below the accuracy threshold, consequently their associated costs are zero. Additionally, at low speed (0.2 m/s) the trunk motion is below the selected threshold, resulting in no corresponding penalty in the cost (Fig. 2 (a)).



(a) Flat terrain. Trotting at 0.2 m/s.



(b) Flat terrain. Trotting at 0.5 m/s.



(c) Flat terrain. Trotting at 0.8 m/s.

Fig. 2: Evolution of the cost during learning in the flat terrain scenario, at low (0.2 m/s), intermediate (0.5 m/s) and high (0.8 m/s) speed, respectively.

Compared to the baseline behaviour (i.e., based on the hand tuned parameters used previously) we report an energy efficiency improvement of 10%, 12% and 26% at 0.8 m/s, 0.5 m/s and 0.2 m/s, respectively. The values of the *gait parameters* obtained are presented in Table I.

Terrain type	Forward velocity	Step Frequency	Step Height	Step Length	Duty Factor
Flat	$0.2m/s$	$1.429Hz$	$0.073m$	$0.083m$	0.55
Flat	$0.5m/s$	$1.72Hz$	$0.066m$	$0.1752m$	0.55
Flat	$0.8m/s$	$1.783Hz$	$0.067m$	$0.268m$	0.55
Cobblestones	$0.3m/s$	$1.50Hz$	$0.077m$	$0.127m$	0.55
Cobblestones	$0.5m/s$	$1.63Hz$	$0.079m$	$0.181m$	0.55

TABLE I: Learned gait parameters.

Fig. 3 shows the impedance profile in the flat terrain scenario (rose lines for 0.2 m/s, blue lines for 0.5 m/s and green lines for 0.8 m/s speed, respectively). The profile is plotted against the normalised duration of the swing phase (which represents 45% of the step cycle[2]). We show the evolution of the values during learning from the start (black), after 5 (dotted, coloured lines) and 10 (thin, coloured lines) learning episodes and the final values (thick, coloured lines).

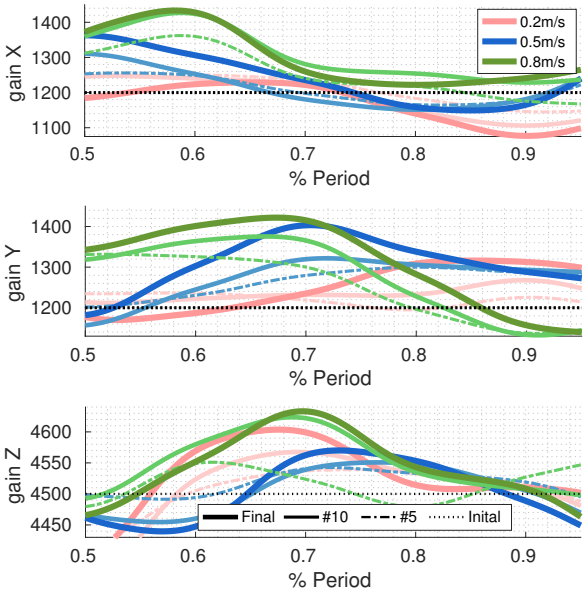


Fig. 3: The impedance profile in the flat terrain scenario (rose lines for 0.2 m/s, blue lines for 0.5 m/s and green lines for 0.8 m/s speed, respectively).

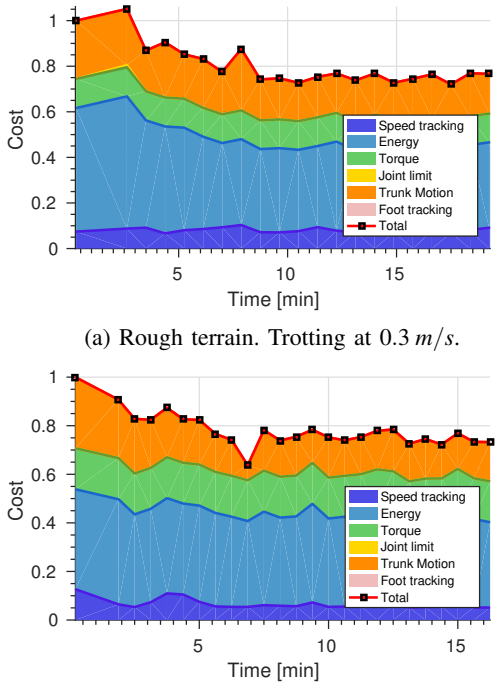
B. Cobblestone terrain

In the cobblestone terrain there are stochastic variations (i.e., the terrain height varies between 0 and 0.03 m). Therefore, a larger number of cycles is needed to compute a reliable average of the disturbances occurring during a locomotion cycle. Hence, a rollout time of 15 *sec*, chosen empirically, is used.

[2]Determined by the *duty factor*, which is one of the *gait parameters*.

The obtained gait parameters are presented in Table I. Compared with the results in the flat terrain we observe an increase in the step height and a decrease in the step frequency. They can be both attributed to the solution trying to prevent the robot from stumbling and reducing the number of height variations acting on the system.

In this scenario the cost function shows variations, even after 9 *min*, since the frequency and amplitude of the disturbances vary every rollout (Fig. 4). However, a significant improvement in the cost is still obtained and the convergence threshold can be adjusted to accommodate this particularity of the scenario ($i.e., 10^{-2}$). As before, the joint and torque limits are not reached and their associated costs are zero. We obtained an energy efficiency improvement of 14% and 21% at high (0.5 m/s) and low (0.3 m/s) speed, respectively.



(a) Rough terrain. Trotting at 0.3 m/s.

(b) Rough terrain. Trotting at 0.5 m/s.

Fig. 4: Evolution of the cost during learning on the cobblestone terrain, at low (0.3 m/s) and high (0.5 m/s) speed, respectively.

Fig. 5 shows the impedance profile in this scenario (orange lines for 0.3 m/s and blue lines for 0.5 m/s speed, respectively). As before, we plot the profile against the normalised duration of the swing phase and the evolution of the values during learning from the start (black), after 5 (dotted, coloured lines) and 10 (thin, coloured lines) learning episodes, and the final values (thick, coloured lines).

C. Discussion

We investigate the disturbance rejection performance, calculated using the GSN, as described in Section III. When comparing the different terrain results, we notice a 15% higher performance in the rough terrain, independent of the forward velocity. We compare the performances of the

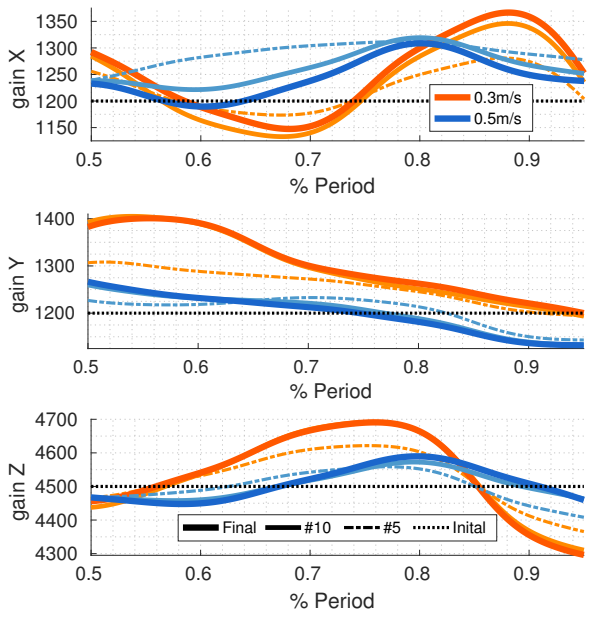


Fig. 5: The impedance profile in the rough terrain scenario (orange lines for 0.3 m/s and blue lines for 0.5 m/s speed, respectively).

initial hand tuned values and the learned ones, on rough terrain. The expert (hand)tuned parameters show an optimum disturbance rejection at $0.5m/s$ (where it slightly outperforms our values). This performance decreases while moving away from this velocity (at $0.3m/s$ our learned values outperform it by 12%). The learned parameters show a similar performance in each terrain type, independent of the velocity. This is expected, as the learning is not taking into consideration any disturbances other than those encountered during exploration.

We summarise the improvements obtained in Table II. The learning time varies between 5 and 10 min. As mentioned previously, the PI^2 computation time is negligible and the learning time is dominated by the exploration time, during which the robot trots with real time factor of 1, collecting behaviour data. This is a significant improvement compared to a similar procedure presented in [16] which requires at least 90 minutes. This improvement was achieved by implementing the modifications discussed in Section II-D.

Terrain type	Forward velocity	Improvement CoT	Learning time
Flat	$0.2m/s$	26%	6*min*
Flat	$0.5m/s$	12%	6*min*
Flat	$0.8m/s$	10%	9*min*
Rough	$0.3m/s$	21%	10*min*
Rough	$0.5m/s$	14%	8*min*

TABLE II: Overview of the Cost of Transport (CoT) improvement obtained in each scenario and the corresponding learning time.

To get an overview of the contributions of the three learning variables groups to the improvement of the performance we ran an initial study on flat terrain with a speed of $0.5m/s$ (Fig. 6). We performed a set of experiments where just one group of the variables was used as the *learning variables* and the resultant performance was recorded. We found that 68% to 75% of the energy efficiency improvement comes from learning the gait parameters and the controller gains, with 24% to 31% coming from the variable impedance strategy.

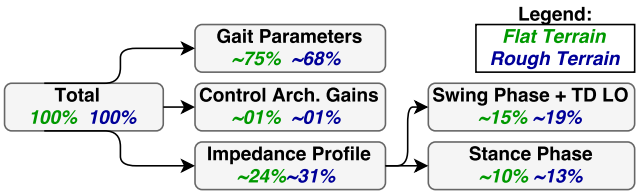


Fig. 6: An overview of our study into the individual contribution of the groups of *learning variables* to the overall improvement.

The analysis indicates that learning the full cycle leads up to a doubling of the computation time. On the other hand, over 80% of the overall energy efficiency improvement can be achieved just by learning the impedance during the swing phase including the touchdown (TD) and liftoff (LO).

Since we use active impedance modulation, which cannot store energy, the contribution of the impedance to the energy performance improvement can be explained by the solution exploiting other advantages of compliance (e.g., reducing the trunk motion). The RCF's trunk controller corrects unwanted trunk motions by generating stabilisation torques. Thus, reducing the need for these forces results in energy savings.

As the PI^2 method is stochastic, we conduct a brief study into how sensitive the solution is to the initial values and the irregularity of the exploration noise. We use the flat terrain with desired speed of $0.5m/s$ as the benchmark task. We construct two distinct scenarios, in which we repeat the learning procedure 10 times, monitoring the convergence of the cost.

In the first scenario, we repeat the learning task, always starting from the same set of *decision parameters* (i.e., the same hand tuned values). Fig. 7 depicts the evolution of the cost as the average (black line) and variance (blue) computer over the 10 instances of learning. The cost converges on average slower than in the experiment reported in Fig. 2(a), but those results fall within the observed variance.

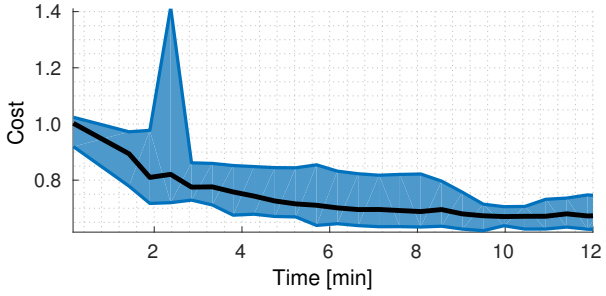


Fig. 7: Evolution of the cost (benchmark task: flat terrain trot at $0.5m/s$). Mean (black) and variance (blue) computed over 10 learning instances (same initialisation of the *decision parameters*).

In the second scenario, we start each learning from a random set of values for the *decision parameters*. These values are sampled from within the respective admissible

regions. We note than although each initial value is feasible, some combinations of parameters can result in completely inadequate gaits.

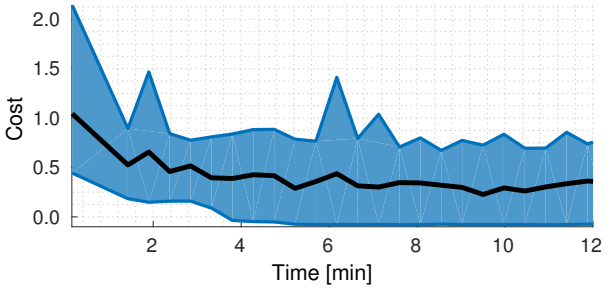


Fig. 8: Evolution of the cost (benchmark task: flat terrain trot at $0.5m/s$). Mean (black) and variance (blue) computed over 10 learning instances (random initialisation of the *decision parameters*).

Fig. 8 depicts the evolution of the cost function in time as the average (black line) and variance (blue) computer over the 10 instances of learning. We note that the evolution of the cost is much slower compared to the behaviour reported in Section IV-A. This suggests that a larger exploration noise might be required in some of the cases, which can be easily addressed by using a methodology with sampling noise adaptation (e.g., the PI^2 extension proposed in [27]).

V. Hardware Experiments

Following the results obtained in the simulation environment, we conducted validation experiments on the hardware platform. In the work presented in this section we focused on a flat terrain trot for a speed of $0.2m/s$ We evaluate the performances of: i) the default RCF values (as tuned by the expert user) and ii) the final values obtained at the end of our learning procedure. Table III show a brief comparison of the performances of the the two solutions in both simulation and hardware.

	$\mathbf{r_s}$	$\mathbf{r_e}$	$\mathbf{r_\tau}$	$\mathbf{r_{lim}}$	$\mathbf{r_{ft}}$	$\theta_{\mathbf{tm}}$
Std RCF (Sim)	0.013	0.09	0.017	0.0	0.0	0.00
Std RCF (Hw)	0.037	0.34	0.017	0.0	0.0041	0.24
Learned (Sim)	0.003	0.07	0.017	0.0	0.0	0.00
Learned (Hw)	0.027	0.21	0.016	0.0	0.0023	0.13

TABLE III: Overview of the performance of the standard RCF and the learned values in both simulation and hardware for the flat terrain trot at $0.2\,m/s$.

The hardware results are consistent with the simulation in terms of joint and torque limits (the small cost values relate to penalties applied to larger torques, but a limit violation would result in much higher ones, as expressed in (4)). The error in the tracking of the feet trajectories and the trunk velocity are higher in the hardware, explained by the higher values in the trunk motion. This additional trunk movements can be attributed to slight discrepancies between the robot model and/or state estimation used [28] in simulation, and the real hardware realities. Consequentially, we observe an overall energy efficiency improvement of 26% and 38%, after learning in simulation and hardware, respectively.

Fig. 9 shows the CoT, feet tracking error and CoM velocity tracking error for the gaits obtained with: i) the standard RCF values and ii) the learned values in the hardware experiments. The values are displayed as average (thick lines) and variance (shaded surfaces) over one cycle of locomotion. The gait obtained after learning is characterised by a similar mean tracking performance, but with a reduced variance (particularly in the velocity tracking behaviour). An overall decrease in the total energy consumption is observed, as expected from the values reported in Table III.

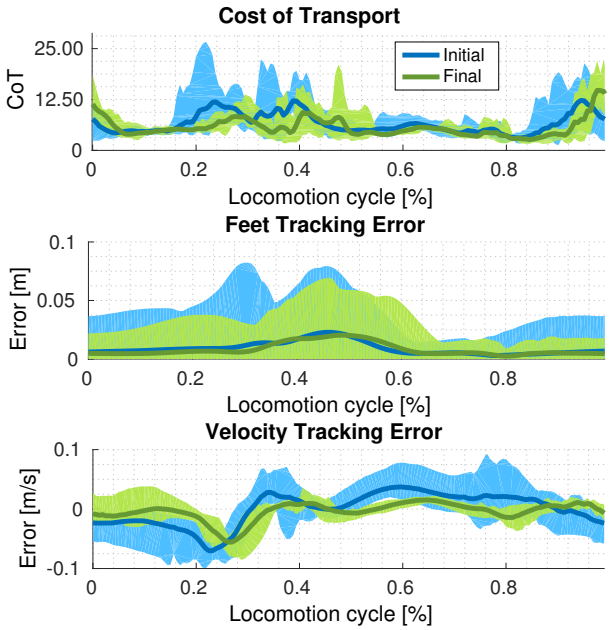


Fig. 9: Performance on the hardware system: CoT (top), feet tracking error (middle) and CoM speed tracking error (bottom) for: i) the standard RCF parameters (blue) and ii) gait with the learned values (green). The behaviour is displayed over one locomotion cycle as: mean (thick line) with variance (shaded surfaces).

These experimental results indicate that the simulation is a good representation of the system and more importantly, the values obtained during the learning were able to improve the performance of the real hardware. A video showcasing the results in both simulation and hardware is available in the additional material.

VI. Conclusions & Future work

We presented a methodology for improving the performance of a locomotion gait by learning a wide set of parameters. Previously, most of these parameters were hand tuned by an expert user (the procedure taking sometimes several hours). We focus on improving the performance of a trotting gait by learning the *gait parameters*, *impedance profile* and *the gains of the trunk controller*. We tackle a previously unexplored aspect, by choosing the secondary goal (after achieving the desired travel speed) of improving energy efficiency (measured using the CoT). We show that

this can be achieved by software based active impedance control, despite no ability to store energy.

The method proved effective on various terrains and speeds. By using a dedicated cost function and limiting the number of optimisation parameters, we significantly improve the learning time (from over 90 to just under 10 minutes), compared to previous approaches [16]. The results obtained in simulation are validated in hardware experiments, with comparable performances. The focus of our investigation goes beyond that of [16] by investigating multiple scenarios and validating our experiments on the HyQ platform.

As mentioned in Section II-D, we learn the impedance profile only for the swing phase, including the touchdown and liftoff. We justify this simplification by the decrease in learning time, weighted against the relatively small benefit of a full impedance profile optimisation. In future work we would like to investigate scenarios where full impedance profile learning might be required and/or alternative approaches (e.g., learning the best constant value required in the stance phase). Since the former has been shown to significantly increase the required learning time, new strategies to overcome these difficulties will have to be devised.

In our investigation we start the learning task from a feasible set of parameters, resulting in a viable gait. We note in Section IV-C that random initialisation might require sampling noise adaptation. Extensions to the standard PI^2, such as [27] can provide this capability.

Additionally, by integrating perception into the learning, the initial values can be pre-set to the terrain. Thus, one could benefit from a predefined policy library, where terrain characteristics are coupled with stored values of the *learning variables*. Lastly, extending the set of experiments to faster movements and various terrains (including the cobblestone scenario) is the focus of future work.

ACKNOWLEDGEMENT

This work was supported by Istituto Italiano di Tecnologia (IIT), with additional funding from the European Union's 7^{th} Framework Programme for research,technological development and demonstration under grant agreement no. 601116, as part of the ECHORD++ (The European Coordination Hub for Open Robotics Development) project under the experiment called *HyQ-REAL*.

REFERENCES

[1] W. H. Press, *Numerical recipes in Pascal: the art of scientific computing*. Cambridge University Press, 1989, vol. 1.

[2] M. Melanie, "An introduction to genetic algorithms," *Cambridge, Massachusetts London, England, Fifth printing*, vol. 3, 1999.

[3] C. Igel, "Policy gradient methods for reinforcement learning with function approximation," 2005.

[4] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *AAAI*, vol. 4, 2004, pp. 611–616.

[5] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, 2009, pp. 849–856.

[6] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search." in *AAAI*. Atlanta, 2010, pp. 1607–1612.

[7] E. Theodorou, J. Buchli, and S. Schaal, "Learning policy improvements with path integrals." in *AISTATS*, 2010, pp. 828–835.

[8] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[9] E. Theodorou, F. Stulp, J. Buchli, and S. Schaal, "An iterative path integral stochastic optimal control approach for learning robotic tasks," in *World Congress*, vol. 18, no. 1, 2011, pp. 11 594–11 601.

[10] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: a path integral approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[11] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[12] J. D. Karssen and M. Wisse, "Running with improved disturbance rejection by using non-linear leg springs," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1585–1595, 2011.

[13] B. Vanderborght, B. Verrelst, R. Van Ham, M. Van Damme, P. Beyl, and D. Lefeber, "Development of a compliance controller to reduce energy consumption for bipedal robots," *Autonomous Robots*, vol. 24, no. 4, pp. 419–434, 2008.

[14] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald *et al.*, "The *KUKA-DLR* lightweight robot arm-a new reference platform for robotics research and manufacturing," in *41st International Symposium on Robotics and 6th German Conference on Robotics*, 2010, pp. 1–8.

[15] B. Vanderborght, A. Albu-Schäffer, A. Bicchi, E. Burdet, D. G. Caldwell, R. Carloni, M. Catalano, O. Eiberger, W. Friedl, G. Ganesh *et al.*, "Variable impedance actuators: A review," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1601–1614, 2013.

[16] B. Pontón, F. Farshidian, and J. Buchli, "Learning compliant locomotion on a quadruped robot," in *IROS Workshop (Ed.), Compliant manipulation: Challenges in learning and control*, 2014.

[17] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2554–2561.

[18] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ–a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[19] J. K. Hodgins and M. Raibert, "Adjusting step length for rough terrain locomotion," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 289–298, 1991.

[20] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *The International Journal of Robotics Research*, vol. 22, no. 3-4, pp. 187–202, 2003.

[21] M. Bjelonic, T. Homberger, N. Kottege, P. Borges, M. Chli, and P. Beckerle, "Autonomous navigation of hexapod robots with vision-based controller adaptation," 2017.

[22] K. V. Mardia and P. E. Jupp, *Directional statistics*. John Wiley & Sons, 2009, vol. 494.

[23] R. M. Alexander, *Principles of animal locomotion*. Princeton University Press, 2003.

[24] M. Focchi, V. Barasuol, I. Havoutis, J. Buchli, C. Semini, and D. G. Caldwell, "Local reflex generation for obstacle negotiation in quadrupedal locomotion," in *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, 2013, pp. 1–8.

[25] D. G. Hobbelen and M. Wisse, "A disturbance rejection measure for limit cycle walkers: The gait sensitivity norm," *IEEE Transactions on robotics*, vol. 23, no. 6, pp. 1213–1224, 2007.

[26] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[27] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," *arXiv preprint arXiv:1206.4621*, 2012.

[28] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic contact estimation and impact detection for state estimation of quadruped robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1023–1030, April 2017.