Differential Dynamical Programming to control a cable driven actuator for the humanoid robot Romeo

Florent Forget, Kevin Giraud-Esclasse, Rodolphe Gelin, Nicolas Mansard, Olivier Stasse

Differential Dynamical Programming to control a cable driven actuator for the humanoid robot Romeo.

Florent Forget[1] and Kevin Giraud-Esclasse[1] and Rodolphe Gelin[2] and Nicolas Mansard[1] and Olivier Stasse[1]

***Abstract*— Autonomous robots such as legged robots and mobile manipulators imply new challenges in the design and the control of their actuators. In particular, it is desirable that the actuators are back-drivable, efficient (low friction) and compact. In this paper, we report the complete implementation of an advanced actuator based on screw, nut and cable. This actuator has been chosen for the humanoid robot Romeo. A similar model of the actuator has been used to control the humanoid robot Valkyrie. We expose the design of this actuator and present its Lagrangian model. The actuator being flexible, we propose a two-layer optimal control solver based on Differential Dynamical Programming. The actuator design, model identification and control is validated on a full actuator mounted in a work bench. The results show that this type of actuation is very suitable for legged robots and is a good candidate to replace strain wave gears.**

I. Introduction

Mobile robots, such as legged robots and humanoid robots, imply new challenges in the design of their actuation system. In this context, it is very important that the robot is able to feel the force that it exerts on its environment. In the same time, the actuator must be light-weight and compact. Direct-drive actuation is then not an option. On many electric-powered humanoid robot, strain wave gears (e.g. Harmonic Drive gear) are used for their compactness. However, if back-drivable, strain wave gears have a poor transparency, i.e. the torque exerted at the joint level (output) is poorly correlated to the torque at the motor level (input), and the output torque is difficult to estimate from the motor current. If an accurate joint-torque estimation is needed, a joint torque sensor must be added to the robot design, which increases the total design cost and the actuation flexibility. Moreover, strain-wave gears are sensitive to impacts, which tend to damage the gear. Their maximum torques are also limited, in particular when impacts have to be expected. For the design of full-size humanoid robots (i.e. size similar to Shaft, NASA Valkyrie, PAL Thalos), strain-wave gears are clearly one of the main limiting factor of the design. On the other hand, most of alternative gears are either not compact enough, or with insufficient reduction ratio.

In this paper, we present the complete implementation (design, modeling, identification and control) of a screw-nut-cable compact actuator based on [1]. This actuator has been used to design the legs of the humanoid robot Romeo (see Fig. 1). A similar model of the actuator has been used to control the humanoid robot NASA R5 (Valkyrie), although the control strategy built upon it is different from ours. In the context of the new NASA challenge with this humanoid robot, the work presented in this paper is very relevant.

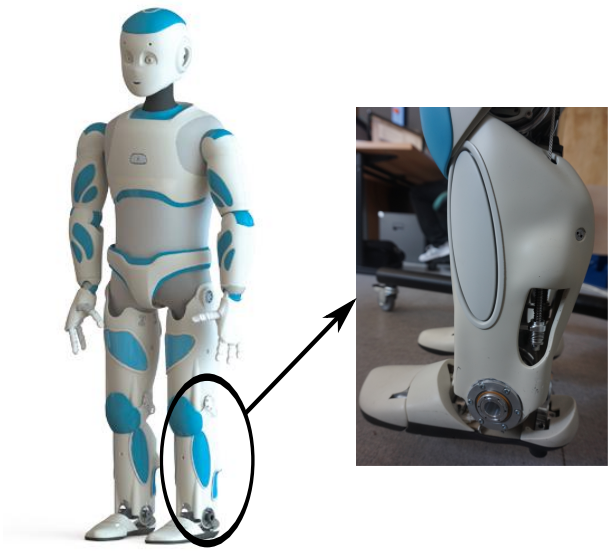

Fig. 1. The leg of the robot Romeo are designed based on a screw-nut-cable actuator, which are low-friction, shock-proof and low-friction, but induce a flexibility due to the cable.

This actuator can offer reduction ratio up to 150 while keeping compact design. It is not sensitive to shocks and impacts and offers a high transparency, making it reliable to estimate output torques from current inputs. It also induces flexibility coming from the cable connecting the screw to the joint output. Adding elasticity into the actuation smoothes the contact with the environment, which prevent rebound and in certain case sliding effects [2]. The flexible element in this particular gear can also be exploited to directly measuring the output torques, by equipping it with sensor able to measure the spring deflection (e.g. angle coders attached to either side of elasticity). We show that good measures of torques/forces can be obtained for quasi-null additional cost. On the other hand, the flexible element behaves like a series-elastic actuator (SEA) [3]. It must be taken into account in the actuator control loop to avoid instability, in particular when accurately tracking reference position.

The contributions of the paper are as follows. We report the implementation of the concept gear [1] in Section II and present an original Lagrangian model of the actuator.

[1] CNRS - LAAS, Toulouse, France, ostasse@laas.fr,florent.forget@laas.fr
[2] SoftBank Robotics Europe, Paris, France, gelin@aldebaran-robotics.com

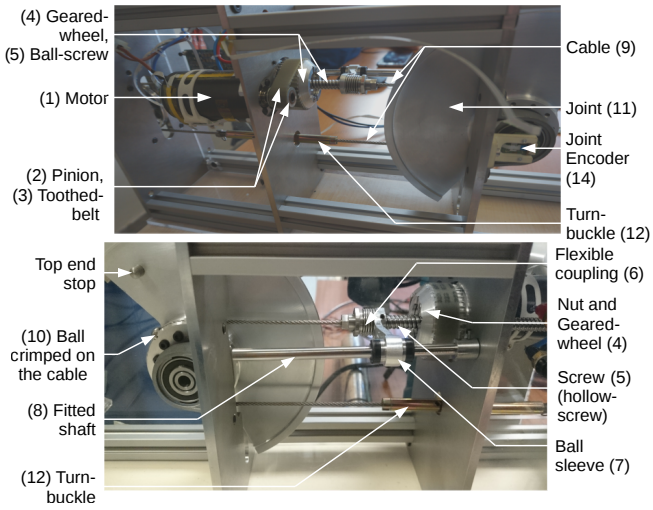

Fig. 2. Right (top) and left (bottom) views of the actuator

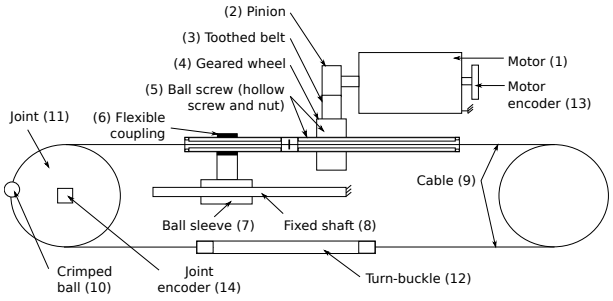

Fig. 3. Schema of the actuator mounted on the workbench

Based on this model, we propose in Section III a model-predictive controller (MPC) based on differential dynamic programming (DDP) [4] able to cope with the actuator flexibility with only few parameters left to the designer to tune. The optimal controller can be set up to either implement a position controller (i.e. by tracking the output position) or a force controller (i.e. by tracking a reference spring deflection). We implemented the proposed approach in an actuator similar to those in the legs of Romeo but mounted in a work-bench. We report in Section IV the identification of the parameters of the Lagrangian model, and the results of controlling the actuator to track joint references.

II. Model of the actuator

We recall here the main principles of the actuator [1], present the design of the actuator used in the experiment and propose an original Lagrangian model upon which our controller is built.

A. Mechanical description

The original design of the actuator has been proposed in [1]. We recall here the general mechanism of the actuator along with the specific implementation that we used in the result section. The actuator is composed of an electrical motor attached to a ball screw which is guided along a fixed axis but can freely rotate inside the nut. The output of the screw is connected to two cables which can either pull or push on the output joint. Our particular actuator is mounted in a workbench used for identification and control validation. The same actuator equips 10 degrees of freedom of the legs of the medium-size humanoid robot Romeo. Two pictures of the actuator with legends are shown in Fig. 2. A schema of the actuator is shown in Fig. 3.

The motor (referenced as (#1) in Fig. 2) is fixed on the base, a pinion (#2) is mounted on its shaft. The pinion leads a toothed belt (#3) to a geared wheel (#4). This part is fixed to the nut of the ball screw (#5). This one is the main component allowing the trade-off between an high reduction ratio (of about 100) and a high reversibility. It is composed of two parts : firstly, guided by bearings with respect to the base, the nut is rotating attached to the previous geared wheel (#4). Secondly the screw is translating (#5) in the same direction of the motor axis. This placement increases the compactness of the system.

To avoid the screw rotation around its main axis and to enforce its motion to be a translation, an additional part is flexibly coupled (#6) with the screw. This part makes the link between the screw (#5) and a ball sleeve (#7) mounted on a fixed shaft (#8). This shaft, whose main axis is in the same direction as the screw translation, and the ball sleeve (#7) are not directly transmitting efforts. These parts are marked in Fig. 2. They are leading the ball screw motion. The flexible coupling mentioned before should allow the screw to slightly oscillate around the two non-axial axis. This increases the reversibility of the system by reducing the friction which can appear due to hyperstaticity. However, the lightly not-respected parallelism between the shaft (#8) and the cable, appended to the not so soft flexible coupling, seems to create constraints on the screw by preventing it to oscillate as it should do. Moreover, contrary to [1], even thought the cable is attached in the middle of the screw (hollow screw) with a crimped sleeve, struts are mounted throughout the screw to maintain the sleeve. The space around the cable is not sufficient to let the cable contact-free with the screw. In this condition, a minor misalignment can create efforts impeding the slight oscillations of the screw and induce non-linear and cyclic friction.

The forward part of the cable (#9) is linked to the joint with a crimped ball (#10) placed in the spherical imprint of the joint (#11). The cable then goes to the turn-buckle (#12). The backward part of the cable is also going to the turn-buckle by the way of a pulley. The turn-buckle is used to fix and pre-load the cable. To keep the workbench simple to use, a rope is attached to the joint (#11) in order to apply some load. This choice limits the output load to only one direction of the joint, which has no negative consequence for our experimental protocol. On the humanoid robot, the load is rigidly driven by the pulley and can move in both directions.

To measure the angle positions, two absolute magnetic encoders are mounted on the test bench. One is fixed behind the motor on its main shaft, marked as (#13) on Fig. 3, the other is placed on the joint (#14), after all the transmission

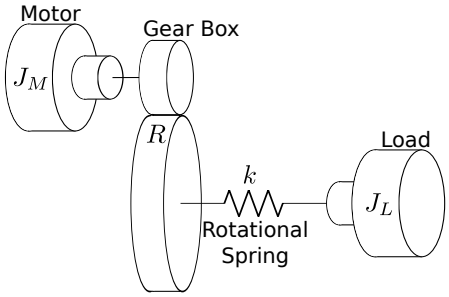

Fig. 4. System considered for the actuator modeling

Symbol	Physic meaning	Identified value
J_M	motor inertia	$1.38 \times 10^{-5}\ kg.m^2$
J_L	load inertia	$8.5 \times 10^{-4}\ kg.m^2$
θ_M	motor angular position	N.A
q	joint angular position	N.A
k	equivalent rotational spring stiffness	$588\ Nm/rad$
R	transmission ratio	96.1
τ_M	motor torque	N.A
τ_{ext}	external torque (environment)	N.A
d_M	vicious friction at motor level	$0.003\ Nm/(rad/s)$
d_L	vicious friction at load level	$0.278\ Nm/(ras/s)$
Cf	dry friction at motor level	$0.1\ Nm$
K_T	motor current to torque ratio	$60.3 \times 10^{-3}\ Nm/A$

TABLE I

VARIABLES AND PARAMETERS USED IN THE MODEL AND CORRESPONDING VALUES ESTIMATED ON OUR SYSTEM

chain. This layout theoretically allows to identify the ratio and the deformation on the transmission (i.e. makes the model parameters observable).

B. Lagrangian Model

With respect to the mechanical description given before, we use the following hypotheses to construct the system model:

[H1] The DC motor driving the mechanism is considered as a perfect source of torque.

[H2] Flexibilities are concentrated into a single linear rotational spring with constant stiffness.

Hypothesis [H1] sounds as we consider the torque directly at the motor side (i.e. before any reduction device) while the motor is current controlled (with current sensor and modern power electronics). Hypothesis [H2] is reasonable as the stiffness of the cable is several order of magnitude lower than the stiffness of any other element in the transmission.

The actuator then boils down to a two-masses system attached by a spring, as illustrated Fig. 4. The first mass is driven by a DC motor coupled with the gearbox. The second mass is interacting with the environment. Friction arises in opposition to the motion of both masses.

Using Lagrangian formalism, we expose the following model describing classical series elastic actuator. We empirically decided during the identification experiments to consider vicious friction on both motor and load side and dry friction on motor side.

$$J_M\ddot{\theta}_M = K_T i_M - \frac{k}{R}(\frac{\theta_M}{R} - q) - d_M\dot{\theta}_M - C_f sign(\dot{\theta}_M) \quad (1a)$$

$$J_L\ddot{q} = \tau_{ext} + k(\frac{\theta_M}{R} - q) - d_L\dot{q} \quad (1b)$$

where the notations are given in Tab I. In order to keep the equations smooth (as needed for the controller), we used a smooth version of the $sign$ function (i.e. a hyperbolic tangent).

III. CONTROL

As reported in previous section, the actuator behaves like a SEA transmission. However, its stiffness is high. Even if it is not relevant to ignore it, it is possible to handle it directly at the joint level, while neglecting it at the level of the robot whole body. In this section, we present the control solution that we implemented to track the joint reference, specified either as reference position or as reference torque. Our objective is then to compute the joint references from a whole-body optimization scheme, that will be accurately tracked by the joint controller running at high frequency.

The controller we report below is composed of a two-stage architecture: a first stage, running on CPU at 1kHz, computes the optimal trajectory (feedforward) and the optimal feedback gains in a model-predictive-control (MPC) style. The second layer, running on the micro-controller at higher frequency (5kHz or higher) uses the feedforward and the feedback gains to compute the reference current sent to the motor.

We first start by a brief overview of possible control approaches that justifies our implementation based on DDP.

A. Brief control state of the art

Considering the implementation of a position controller on our actuator, the major difficulty is the gain tuning. In fact high gains on position loop are necessary for good precision but may make the system unstable due to the flexibility. Several control schema have been developed to address this problem. Using an automatic control approach, we could have analyzed the eigen modes of the actuator models and consequently tuned a correction scheme to achieve desired convergence, stability and precision [5]. However, the model contains non-linear terms. Moreover, the same controller must be deployed on several variations of the same actuator implemented on the legs of Romeo. A more automatic method is desirable. On a second hand, H_∞ synthesis methods allow to automatically conceive a efficient controller for a given system. A method has been proposed in [6] to control Series Elastic Actuators as variable stiffness actuators. Finally, solving an optimal control problem makes it possible to generate a gain scheduling over a trajectory or a command sequence to achieve a given behavior [7], [8]. Contrary to the previous kernel of methods, optimal control is very versatile: a first controller (e.g. position-based) can be quickly adapted to a new version (e.g. force-based). It would also be easy to adapt on other flexible actuators with a different mechanical principle but where the two-stage implementation is desirable, like on pneumatic muscles [9].

It is also quite easy to handle constraints like command limitations or joint limits [4].

We need an optimal control solver that is lightweight and easy to implement within the low-level control architecture of the robot. Here, we do not need to deal with particularly complex constraints but only with some box constraints on the control or the state. However, the optimal-control solver must run in a predictive (receding-horizon) way so that a new optimal control trajectory can be recomputed at every new sensor measurement. We decided to implement our solver based on DDP, that we quickly recall below, before presenting our two-stage control architecture.

B. Differential dynamic programming

DDP is an optimal control scheme with a "single shooting" strategy, that is able to efficiently cope with the sparsity of the underlying numerical system but has the drawbacks of being quite unable to handle complex constraints. This trade-off is very suitable to our problem. We recall here the basis of DDP. This recall is needed for understanding how we designed both layers of our control architecture. More detailed information about this optimal control solver can be found in [10].

Consider a generic mechanical system described by its (discrete) dynamic equation:

$$\mathbf{x_{i+1}} = f_i(\mathbf{x_i}, \mathbf{u_i}) \tag{2}$$

Where $\mathbf{x_i}$ represents the current state of the actuator (position, speed, torque ...) and $\mathbf{u_i}$ is the input command (current, torque, voltage ...). This model may or not be linear and time varying. We expose the cost we want to minimize on a given horizon T.

$$J(\mathbf{U}|\mathbf{x_0}) = \sum_{i=0}^{T-1} c_i(\mathbf{x_i}, \mathbf{u_i}) + c_T(\mathbf{x_T}) \tag{3}$$

with $X = \{\mathbf{x_0}, \mathbf{x_1}, ..., \mathbf{x_T}$ and $U = \{\mathbf{u_0}, \mathbf{u_1}, ..., \mathbf{u_{T-1}}\}$ respectively the state and control sequence over horizon T and $\mathbf{x_0}$ the initial state of the system (typically estimated from sensors). It is to be noticed that knowing $\mathbf{x_0}$ and $\mathbf{U}$ is enough to know the state of the system at each moment of the horizon because of the relation (2). So the optimal control problem consists in finding the correct $\mathbf{U}$ minimizing the cost for a given $\mathbf{x_0}$ initial state.

We introduce then the cost-to-go function to be

$$J_i(\mathbf{U_i}|\mathbf{x_i}) = \sum_{j=i}^{T-1} c_j(\mathbf{x_j}, \mathbf{u_j}) + c_T(\mathbf{x_T}) \tag{4}$$

with $\mathbf{x_j}$ integrated from $\mathbf{x_i}$. The optimum cost-to-go is named the value function V:

$$V(\mathbf{x_0}, i) = \min_{\mathbf{U_i}} J_i(\mathbf{x_0}, \mathbf{U_i}) \tag{5}$$

We evidently have that $V(\mathbf{x_0}, T) = c_T(\mathbf{x_T})$. The "Belman" dynamic-programming principle teaches us that minimizing the cost by choosing the correct control sequence can be reduced to the backward minimization of a single control input. This principle gives us the Bellman equation :

$$V(\mathbf{x_i}, i) = \min_u [c_i(\mathbf{x_i}, \mathbf{u_i}) + V(f(\mathbf{x_i}, \mathbf{u_i}), i+1)] \tag{6}$$

The DDP solver computes the optimal control sequence $\mathbf{U}$ by solving equation (6) backwardly in time. For this purpose let Q be the variation of $c(\mathbf{x}, \mathbf{u}) + V(f(\mathbf{x}, \mathbf{u}), i+1))$ around the $i-th$ state and command. We have :

$$\begin{aligned} Q(\delta\mathbf{x}, \delta\mathbf{u}) \equiv\ & c(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}) - c(\mathbf{x}, \mathbf{u}) \\ & + V(f(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}), i+1)) \\ & - V(f(\mathbf{x}, \mathbf{u}), i+1)) \end{aligned} \tag{7}$$

By taking the second order approximation of (7) we obtain:

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix} \tag{8}$$

For readability we will use subscript notation to denote partial derivative (e.g. $f_x = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}$). We also denote the next state by $V' \equiv V(i+1)$. We can now expose:

$$Q_x = c_x + f_x^T V_x' \tag{9}$$
$$Q_u = c_u + f_u^T V_x' \tag{10}$$
$$Q_{xx} = c_{xx} + f_x^T V_{xx}' f_x + V_x' f_{xx} \tag{11}$$
$$Q_{uu} = c_{uu} + f_u^T V_{xx}' f_u + V_x' f_{uu} \tag{12}$$
$$Q_{ux} = c_{ux} + f_u^T V_{xx}' f_x + V_x' f_{ux} \tag{13}$$

Where the last term of the last three equations represents the contraction of a tensor with a vector. Minimizing (7) with respect to $\delta\mathbf{u}$ gives us:

$$\delta\mathbf{u}^* = arg\min_{\delta\mathbf{u}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta\mathbf{x}) \tag{14}$$

Showing up two terms:

- a feedforward term: $\mathbf{k} = -Q_{uu}^{-1} Q_u$
- a feedback term : $K = -Q_{uu}^{-1} Q_{ux}$

Using back this result into (7), we obtain a quadratic approximation of the value function at $i-th$ instant:

$$\Delta V(i) = -\frac{1}{2} Q_u Q_{uu}^{-1} Q_u \tag{15}$$
$$V_x(i) = Q_x - Q_u Q_{uu}^{-1} Q_{ux} \tag{16}$$
$$V_{xx}(i) = Q_{xx} - Q_{ux} Q_{uu}^{-1} Q_{ux} \tag{17}$$

Computing all term from (9) to (17) for $i = N-1$ down to $i = 0$ is called the backward phase. We then need to calculate the change induced on the state sequence by the modification on the command sequence. This is the forward phase, detailed below:

$$\hat{\mathbf{x}}(0) = \mathbf{x}(0) \tag{18}$$
$$\hat{\mathbf{u}}(i) = \mathbf{u}(i) + \mathbf{k}(i) + K(i)(\hat{\mathbf{x}}(i) - \mathbf{x}(i)) \tag{19}$$
$$\hat{\mathbf{x}}(i+1) = f_i(\hat{\mathbf{x}}(i), \hat{\mathbf{u}}(i)) \tag{20}$$

The solver iterates on these two phases until convergence of the result (minimal changes on $\mathbf{U}$). One can find the algorithm detailed in a pseudo-code on algorithm 1.

In order to ensure good convergence and to add some specificities to the algorithm, we decided also to implement some other features:

Algorithm 1 Differential Dynamic Programming Solver

1: $\{initialisation:\}$
2: $\mathbf{U} \leftarrow$random command sequence
3: $\mathbf{X} \leftarrow$init($\mathbf{x_0}$, $\mathbf{U}$)
4: **repeat**
5: $V(T) \leftarrow c_T(\mathbf{x_T})$
6: $V_x(T) \leftarrow c_{Tx}(\mathbf{x_T})$
7: $V_{xx}(T) \leftarrow c_{Txx}(\mathbf{x_T})$
8: **for** i=T-1 down to 0 **do**
9: $Q_x, Q_u, Q_{xx}, Q_u, Q_{ux} \leftarrow$ see equations (9) to (13)
10: $\mathbf{k} \leftarrow -Q_{uu}^{-1} Q_u$
11: $K \leftarrow -Q_{uu}^{-1} Q_{ux}$
12: $\Delta V, V_x, V_{xx} \leftarrow$ see equations (15) to (17)
13: **end for**
14: $\hat{\mathbf{x}}(0) = \mathbf{x}(0)$
15: **for** i=0 to T-1 **do**
16: $\hat{\mathbf{u}}(i) = \mathbf{u}(i) + \mathbf{k}(i) + K(i)(\hat{\mathbf{x}}(i) - \mathbf{x}(i))$
17: $\hat{\mathbf{x}}(i+1) = f_i(\hat{\mathbf{x}}(i), \hat{\mathbf{u}}(i))$
18: **end for**
19: **until** convergence

Line search: DDP being a type of Newton descent, line search allows the algorithm to adapt the step length so that convergence is faster.

Regularization: DDP implies the inversion of Q_{uu} matrix which in certain cases may not be invertible. Regularization makes the matrix invertible if it was not in the first place and it integrates this modification into the whole computation.

Control limitation: Introduced in [4], the control limited DDP is an extension of the DDP where it is possible to add bound constraints on the command input vector. In practice this feature is possible by solving a box QP problem.

C. Two-stage control architecture

The outputs of the DDP solver are the optimal trajectories in both control U and state X spaces, along with the optimal feedback gains along this trajectory K. Our objective is to use at best these information to feedback as frequently as possible on the sensor measurements.

For that, we implement the DDP as a model-predictive (receding-horizon) control scheme. At any instant, we maintain a valid (possibly suboptimal) control trajectory U^*. As soon as the DDP performed one valid step of the nonlinear search loop (line #4 of Alg. 1), the solver candidate trajectory U is used to update U^*. The receding horizon of the DDP is then shifted, while the initial state of this new horizon is updated to the latest state estimation. The dynamic system considered in our DDP (1) is low-dimension and thus leads to short computation timings. It is easy to implement such solver to obtain 1kHz control frequency. However, it is difficult to implement the DDP solver directly on the actuator micro-controller, but rather on the central robot CPU board. The frequency is then limited by the communication bandwidth to upload the sensor measurements and download the control references.

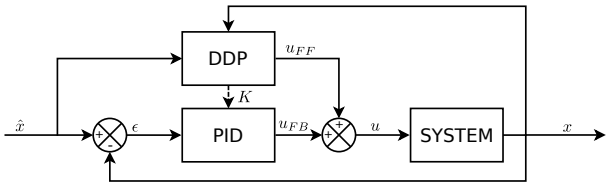

Fig. 5. General architecture of the proposed DDP-based MPC controller

On the other hand, the micro-controller of the actuator is able to update the motor control at much higher frequency (e.g. 5kHz). This higher frequency enables us to take advantage of the optimal feedback gains computed by the DDP solver. On the micro-controller, we then maintain an optimal control (feedforward) trajectory and the corresponding optimal feedback gains along this trajectory. At each control cycle of the micro-controller, the state is estimated from previous sensor measurements. The control (reference motor current i^*) is then computed as the sum of the feedforward (optimal control u^*) and feedback (optimal gains K^*):

$$i^*(t) = u^*(t) + K^*(\mathbf{x}^*(t) - \hat{\mathbf{x}}(t))$$

where x^* is the latest optimal trajectory in the state space computed by the DDP and $\hat{x}$ is the estimated state.

Fig. 5 shows the general architecture of the proposed controller. The DDP controller runs at 1kHz on CPU and produces the feedforward control u_{FF}. The feedback controller runs at 5kHz on micro-controller, estimates the state and produces the feedback control u_{FB} from the optimal gains K. The communication bus between micro-controller and CPU board carries the estimated state at 5kHz and the optimal feedforward and gains at 1kHz. Both feedforward and feedback are finally summed and used to servo the motor current.

IV. Simulation and experiments

A. Actuator parameters estimation

Parameters estimation was done using matlab with temporal measurements. Data have been collected by running joint position control experiments with a simple PID with stable gains. All measurements (joint position, motor position, motor current, motor supply voltage ...) are then collected to achieve the parameters estimation. It appears several unmodeled phenomena leading to bad results of the estimation (see Fig. 6. These unmodeled phenomena create non-linear and position-dependant friction which are hard to model. Among these potential phenomena we can list:

- Ball-screw being to much constrained by the flexible coupling
- The cable being not free enough at the mounting with the ball-screw.
- It is possible to load the system with a mass but it acts only in one direction which hides part of the dynamics.
- The cable is compound of two parts of different length. Namely the elasticity of the system is function of the sens of the efforts.

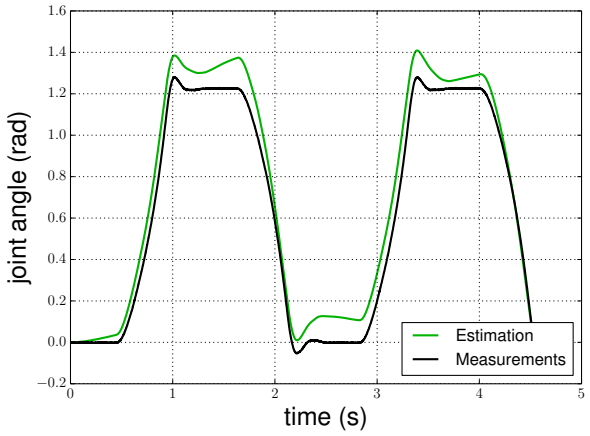

Fig. 6. Comparison between model and real system response to a same control input (motor current)

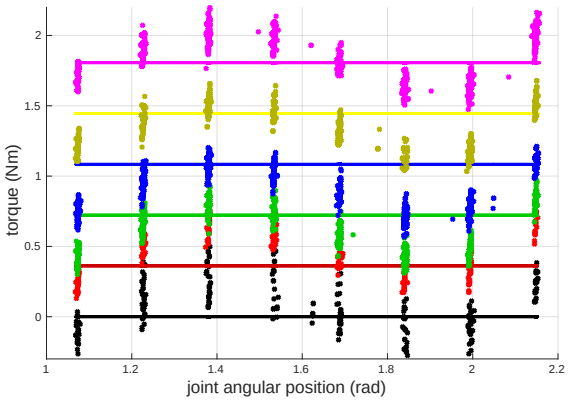

Fig. 7. Torque estimation based on angular position measurements for different external torque applied

All these specificities are more detailled in Part II.

As it was introduced in Part I, SEA are suitable for external torque estimation when equipped with angular sensor on both sides of elasticity. We present on Fig. 7 an experiment to show this particularity. For this experiment, we attached a mass to the output of the actuator as a known external torque. We take measurements points for different joint position in a static state with different masses. We then reconstruct this torque by computing the spring deflection given by the angular sensor measurements. As shown in Fig 7 these measurements allow us to have an approximation of the external torque. This experiment exposes a cyclic non-linearity on the torque estimation depending on the joint position. It is due to the same phenomena as above and more detailed in Part II.

B. Experiments - position control

1) Simulation: Experiments were first conducted on simulation using Romeo's actuator model. It was decided to compute the response of the system with noisy parameters in order to observe the solver robustness. To make the model biased, we added a random error with a maximum of 50%

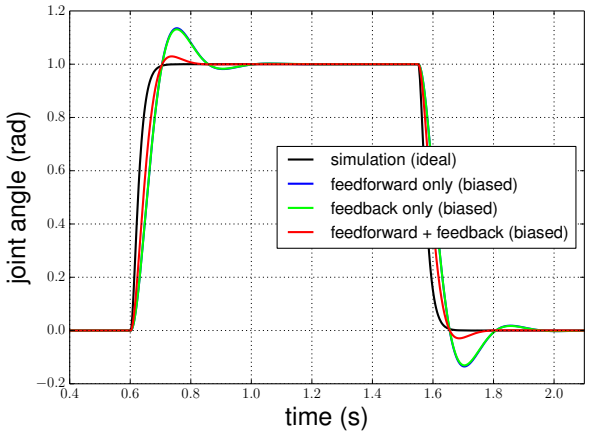

Fig. 8. Control of Romeo Actuator in simulation for ideal (black) and biased model with Feedforward only (blue) feedback only (green) and both feedforward and feedback term (red) (red and blue curve are approximately the same)

of its value to each parameter. For convenience we only plot results for a single biased model which represents one of the worth case we have seen. We also wanted to observe the effect of each term in the control. For this purpose, plot are showing system evolution for architecture depicted in Fig. 5, for a feedback-free, a feedforward-free and finally the full architecture.

Results are shown in Fig. 8. One can observe the robustness of the control to errors on parameters estimation. It can also be noticed that the feedback term has an effect on the system response time.

2) Romeo Actuator: The same experiments have been made on the real system (see Fig. 9). As the parameters estimation wasn't very precise, the observed behavior is different from what we expected with the model. Nevertheless, our architecture succeed in controlling the system with good stability and response time. Fig. 10 shows the input command generated by the DDP solver. One can notice that around 0.8s and 1.5s the control input (motor current) is limited to (-5A, 5A) thanks to Control Limited Differential Dynamic Programming approach [4].

3) Trajectory tracking: We decided to test our approach on the tracking of a joint trajectory (see Fig. 11). For this purpose, we choose the joint trajectory of the knee of HRP2 humanoid robot during a stairs climbing. We compared the tracking in simulation for both ideal and biased system and then on the real workbench. Due to the likeliness of the curves (1% error), we decided to plot for each case the difference between the reference and the considered curve.

C. Experiments - torque control

We also adapted our architecture to control the actuator joint torque. Results of the simulation are shown in Fig. 12. We can see that with the biased model, the solver is not able to reach the desired torque. This is because, contrary to the position, torque estimation relies directly on parameters value (especially on the spring stiffness, transmission ratio

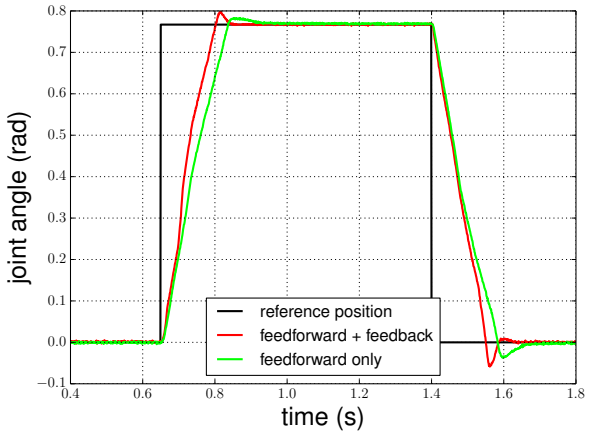

Fig. 9. Joint angle during experiment on real system with feedforward only (green) and feedforward+feedback term (red)

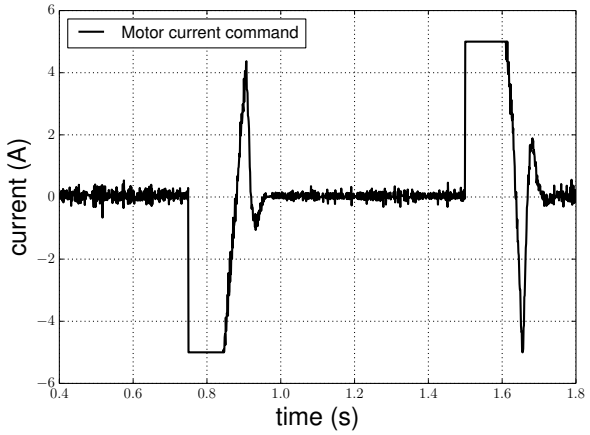

Fig. 10. Command (motor current) during experiment on real system

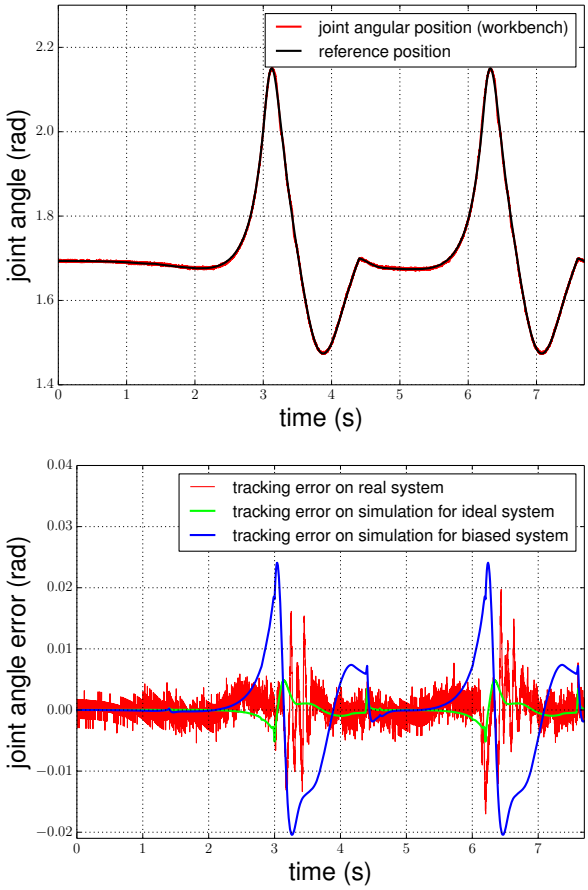

Fig. 11. Position trajectory tracking in a scenario of climbing stairs for knee joint. First plot shows the reference position trajectory, then the different errors with respect to this reference

and load vicious friction factor). Practically, an error on one of these parameters generates directly an error on the torque estimation.

V. Conclusion

In this paper, we have presented the complete implementation of a new compact and transparent actuator, very suitable for mobile robots, in particular in the context of locomotion. We have presented the Lagrangian model of the actuator and build atop of it a two-stage optimal controller able to accurately track either output position or torque references. The optimal controller is lightweight, easy to implement, and can compute a feedback at 5kHz on a micro-controller board. We have shown in the experiment results that both layers are needed to efficiently control the actuator: either feedforward or feedback alone are not able to perform as efficiently. Moreover, we experimentally showed the capabilities of the actuator in term of transparency (i.e. estimating output torques from motor current), and the adequacy of the model to capture the complexity of the actuator.

While the proposed control architecture is very suitable for the screw-nut-cable actuator, it is also appropriate for

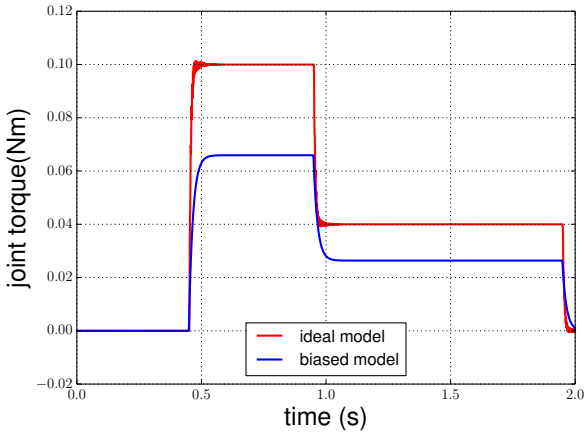

Fig. 12. Torque control of the actuator in simulation for both ideal (red) and biased (green) model

other kind of flexible actuators such as SEA at large, variable stiffness actuators [8], [7] or Mckibben pneumatic actuators [9], [11]. The MPC scheme can be easily adapted to another dynamic model or another cost function. Our objective is now to adapt the controller to the whole body of the robot and to use it to control complex humanoid movements.

ACKNOWLEDGMENT

This work was partially supported by the PSPC project Romeo 2.

REFERENCES

[1] P. Garrec, "Design of an anthropomorphic upper limb exoskeleton actuated by ball-screws and cables," *Bulletin of the Academy of Sciences of the Ussr-Physical Series*, vol. 72, no. 2, p. 23, 2010.

[2] J. Lee, W. Choi, D. Kanoulas, R. Subburaman, D. G. Caldwell, and N. G. Tsagarakis, "An active compliant impact protection system for humanoids: Application to walk-man hands," in *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2016, pp. 778–785.

[3] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 1, 1995, pp. 399–406.

[4] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 1168–1175.

[5] J. S. Mehling, "Impedance control approaches for series elastic actuators," Ph.D. dissertation, Rice University, 2015.

[6] N. Abroug and E. Laroche, "Transforming series elastic actuators into variable stiffness actuators thanks to structured h inf control," in *European Control Conference (ECC)*, 2015, pp. 734–740.

[7] I. Sardellitti, G. A. Medrano-Cerda, N. Tsagarakis, A. Jafari, and D. G. Caldwell, "Gain scheduling control for a class of variable stiffness actuators based on lever mechanisms," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 791–798, 2013.

[8] P. Geoffroy, O. Bordron, N. Mansard, M. Raison, O. Stasse, and T. Bretl, "A two-stage suboptimal approximation for variable compliance and torque control," in *Control Conference (ECC), 2014 European*, 2014, pp. 1151–1157.

[9] G. K. H. S. L. Das, B. Tondu, F. Forget, J. Manhes, O. Stasse, and P. Soueres, "Controlling a multi-joint arm actuated by pneumatic muscles with quasi-ddp optimal control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 521–528.

[10] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., 2008, pp. 1465–1472.

[11] G. K. H. S. L. Das, B. Tondu, F. Forget, J. Manhes, O. Stasse, and P. Soueres, "Performing explosive motions using a multi-joint arm actuated by pneumatic muscles with quasi-ddp optimal control," in *Control Applications (CCA), 2016 IEEE Conference on*, 2016, pp. 1104–1110.