Implementation, Identification and Control of an Efficient Electric Actuator for Humanoid Robots

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

Implementation, Identification and Control of an Efficient Electric Actuator for Humanoid Robots

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 Talos), 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.

The design of such kind of actuators is a widely studied subject. Different technologies are used to address these problems. Electric-based actuation is very desirable because it is simple to implement, hence also more reliable for a given integration effort. A first step is to adapt electric motors to humanoid robotics needs as it is done in [1] for quadruped robot. By increasing the motor diametrer, the nominal speed is lowered while the nominal torque is improved, allowing to reduce the reduction ratio and so having a more backdrivable system. Another route is to improve the design of the gear box, as done in [2] where authors chose to improve strain-wave gears technology to build a torque-controlled robot. However, despite the improvement, strain-wave gear implies many drawbacks: unsufficient torque limit, sensitivity to impact, lack of efficiency and of transparence. Adding a passive element at the gear output releases a part of the limitations: it protects the gear from impact. It also makes a part of the actuation transparent, while an encoder may be used to directly (after calibration) measure the torque applied on the joint side. However, the passive element makes the actuation more difficult to control and intrinsically lowers the possible control bandwidth, which is not desirable for achieving fast-dynamics movements. Variable-stiffness actuators as in [3] makes it possible to dynamically stiffen the robot when high dynamics is needed, but then boiling down to the same limits than rigid electric actuation. On a quite different route, hydraulic technology is promising to conceive robotics actuators allowing good power to weight ratio and shock absorption [4], [5], although the implementation of the complete robot becomes more challenging.

In this paper, we present the complete implementation

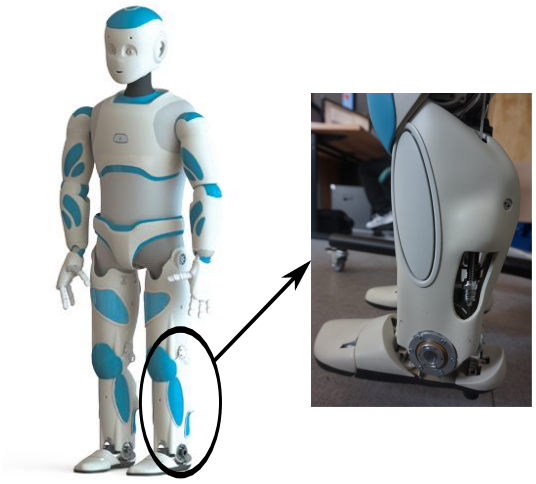

Fig. 1. The leg of the robot Romeo are designed based on a screw-nut-cable actuator, which are shock-proof and low-friction, but induce a flexibility due to the cable.

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

(design, modeling, identification and control) of a screw-nut-cable compact actuator based on [6]. 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) [7], 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.

The actuator offers reduction ratio up to 150 while keeping compact design. It has a high tolerance to shocks and impacts and offers a high transparency, making it reliable to estimate output torques from motor currents. 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 [8]. 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 encoders attached to each side of elasticity). We show that measures of torques/forces can be obtained for quasi-null additional cost. The flexible element behaves like a series-elastic actuator (SEA) [9]. It must be taken into account in the actuator control loop to avoid instability. However, the flexibility is an order of magnitude smaller than on typical SEA.

The contributions of the paper are as follows. We report the implementation of the concept gear [6] in Section II and present an original Lagrangian model of the actuator. Based on this model, we propose in Section III a model-predictive controller (MPC) based on differential dynamic programming (DDP) [10] 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 one of the actuator of Romeo mounted in a work-bench. We report in Section IV the identification of the parameters of the Lagrangian model, the results of controlling the real actuator to track joint references and the study in simulation of the torque bandwidth compared to state-of-the-art actuators with similar ratio.

II. Model of the actuator

We recall here the main principles of the actuator [6], 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 [6]. We recall here the general mechanism of the actuator 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 pull

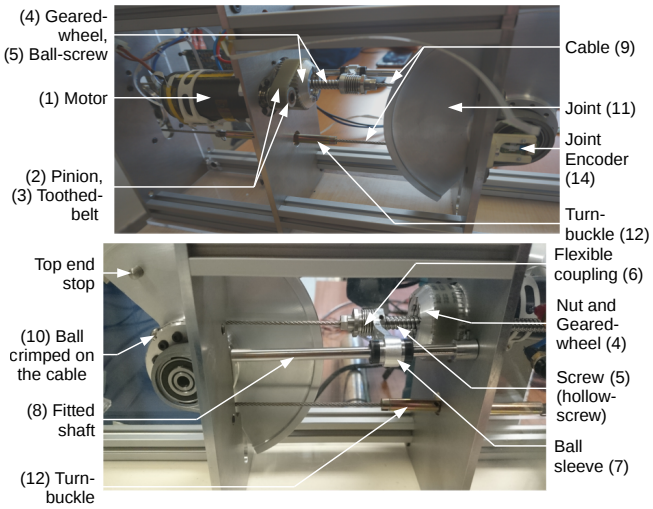

Fig. 2. Right (top) and left (bottom) views of the actuator

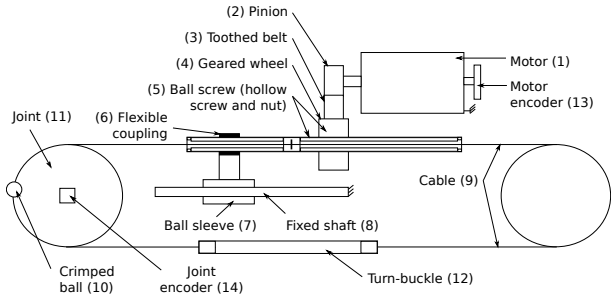

Fig. 3. Schema of the actuator mounted on the workbench

the output joint in the two rotation directions. 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). The screw is the main component allowing the trade-off between an high reduction ratio (of about 100) and a high reversibility. It also 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)(#7) between the screw and a fixed shaft (#8). Note that this part is not introducing the elasticity we try to manage in this paper.

The cable (#9) is the main part of the system introducing the flexibility we deal with in this paper. The forward part of the cable 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

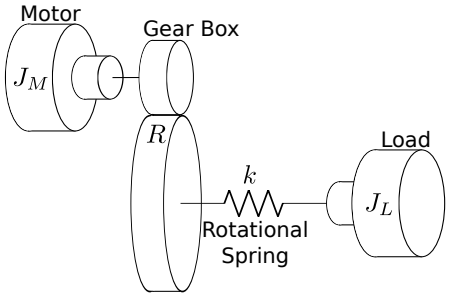

Fig. 4. System considered for the actuator modeling

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	$60.3 \times 10^{-3}\ Nm/A$
μ	motor efficiency	0.78

TABLE I

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

to the joint (#11) in order to apply some load. This set-up limits the output load to only one direction of the joint. This has no negative consequence for our experimental protocol in comparison with the real robot.

To measure the angle positions, two absolute magnetic encoders are mounted on each side of the gear. One is fixed behind the motor on its main shaft (#13), the other is placed on the joint (#14), after the transmission chain. This layout measures the ratio and the deformation on the transmission and makes the model parameters theoretically observable.

Even though the flexible coupling should increase the transparency of the system, it seems to create constraints on the screw by preventing it to oscillate normally. Moreover, contrary to [6], the space around the cable (attached in the middle of the hollow screw with a crimped sleeve) is not sufficient to let avoid the cable to slide on the screw. Any minor misalignment the creates efforts impeding the slight oscillations of the screw and introduce non-linear and cyclic friction (as detailed in Section IV-A).

B. Lagrangian Model

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

- **[A1]** The DC motor driving the mechanism is considered as a perfect source of torque.
- **[A2]** Flexibilities are concentrated into a single linear rotational spring with constant stiffness.

Assumption [A1] is advisable 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). Assumption [A2] 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 in 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.

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

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

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).

Because several mechanical phenomena were difficult to model (see Section IV-A), we make the choice to keep a simple model. In practice, the good transparency of the gear makes it possible to properly estimate the forces applied on the actuator. We can then rely on feedback more than on feedforward, being given that the control law (hence the model) is sufficiently fast to be evaluated. The rational is the higher the control frequency, the lower error between the real system state and its estimation and the more accurate the control.

III. CONTROL

As reported in previous section, the actuator behaves like a SEA transmission, with higher stiffness than typical SEA implementation. It is not relevant to ignore it, however 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 start by a brief overview of possible control approaches that justifies our implementation based on differential dynamic programming (DDP).

A. Brief control state of the art

We are interesting in setting up a controller either of the output joint position (position controller), or of the relative position of input and output (spring deflection, i.e. force controller). In both cases, the major difficulty is the tuning of the control gains: 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. The analysis of the Eigen modes of the actuator models to achieve desired convergence, stability and precision [7], [11] is difficult due to the model 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.

A method has been proposed in [12] to control SEA using H_∞. It relies on automatic controller tuning from identification data. However the results that the controller is quite sensitive to identification errors, which makes it difficult to generalize. Alternatively, optimal control is very versatile (i.e. the controller can be quickly adapted to track either position or force references) and is quite resilient in practice to errors in the model [13], [14]. It is also straightforward to generalize it to other SEA/VSA mechanisms, like pneumatic muscles [15]. Optimal controller can also handle constraints like torque or position limits [10]. Optimal control has been used to control both joint position and stiffness of an approximate linear model (LQR) [13]. Dedicated approximations using polynomials have then been used to fit the computation capabilities of the control board.

We rather propose here a two-stage approach to combine the versatility of the nonlinear optimal control problem with the efficiency needed to solve it on the control board. The nonlinear problem is solved at medium frequency by the central CPU. The optimal control, along the corresponding Ricatti gains are then sent to the control board where the optimal control is updated from sensor measurements at high frequency. To keep the nonlinear solver simple while enforcing joint constraints, we implemented a box-DDP [10], running at 1kHz. The control board then applies a PID corrector (using Ricatti gains extracted from the DDP) at 5kHz.

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 [16].

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 obviously 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

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

next state by $V' \equiv V(i+1)$. We can now expose:

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

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:

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

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:

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

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:

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

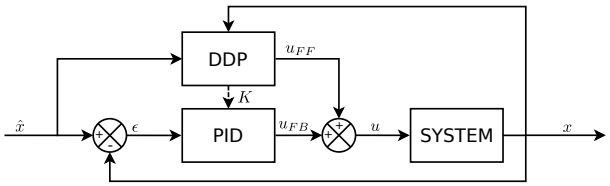

Fig. 5. General architecture of the proposed DDP-based MPC controller. The DDP running of CPU computes the optimal (feedforward) control u_{FF} and the Ricatti gains K at 1Khz. The control board (PID) then adds a feedback term u_{FB} at 5Khz.

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:

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 [10], 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.

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

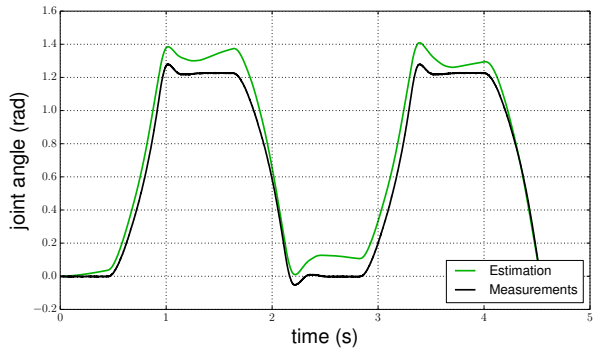

Fig. 6. Comparison between model and real system response to a same control input (motor current)

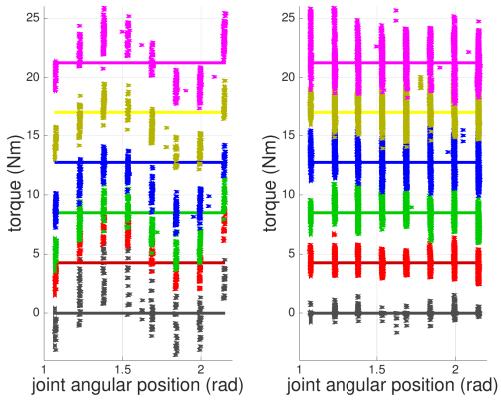

Fig. 7. Output joint torque estimation: **(left)** using only the two joint encoders measuring the spring deflection **(right)** using the current measures and the full actuator model. The estimation from encoders is biased by the friction in the hardware. The current measure leads to a quite good torque estimation although noisy. Both measures are satisfactory given the absence of a direct torque sensor, and are complementary.

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

Off-line estimation: All measurements (joint position, motor position, motor current, motor supply voltage ...) are collected while controlling the actuator with a simple controller (PID with low gains). The estimation of the model parameters is done using MATLAB®. The result is shown in Fig. 6, by comparing simulated and hardware response to a same open-loop control. Thanks to the transparency of the actuator, the model is easily identified. The prediction in simulation properly fits with the real trajectory. Although the parameters are better estimated than on other types of transmission, the identification is not perfect. From the captured data, we identified that this comes from several defects in the implementation of the actuator (ball-screw being to much constrained by the flexible coupling, cable being not free enough at the mounting with the ball-screw, elasticity being different in the two directions due to the inequal length of the two cables). It would be possible to model these effects, hence to obtain a better prediction. However, it would also make the controller more complex and more costly. We rather believe that it would be easier to correct this effect by a more careful implementation of the actuator.

On-line estimation: The actuator is not equipped with direct torque sensor. However, two indirect measurements are available. We have two encoders on each side of the flexibility, and can then use the model to estimate the output torque. Thanks to the actuator transparency, we can also use the measured motor current to estimate the output torque. To validate both measurements, we took measurements points for different joint positions in a static state with different known masses attached to the actuator output. The mass being static, the output torque is known and can be compared to the estimation using either the encoders or the current sensor. The result is displayed in Fig. 7. Both estimations are accurate. They also are complementary: the estimation from the encoders is biased by friction; the estimation from current is more noisy. Merging both estimations in a proper estimator would lead to an accurate estimation able to compete with a direct torque measurement (without the price, implementation issue and fragility of an actual torque sensor).

In conclusion, the transparency of the actuator leads to accurate model estimation, both for (off-line) calibration and (on-line) torque estimation.

B. Experiments - position control

Simulation: We validate first the position controller in simulation, using the model and the two-stage MPC presented above. The MPC uses the true (identified) model for prediction, while the simulation is integrated using a biased model (parameters randomly modified of 50% – for convenience we only plot results for a single biased model). Control frequencies are the same than on the real system.

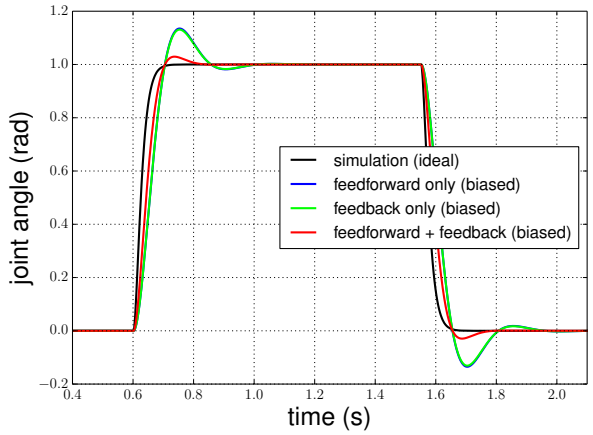

Fig. 8. Simulation for ideal and biased model with feedforward and/or feedback (similar trajectories are obtained when a single term is active).

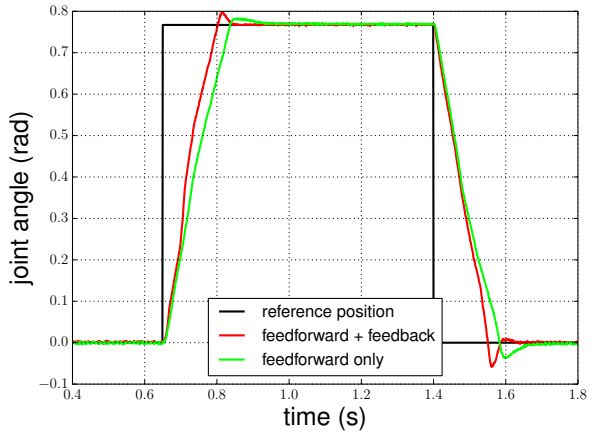

Fig. 9. Hardware experiment: position tracking with and without feedforward (feedback must be alsways active on hardware).

Fig. 8 displays the effect of the feedback and feedforward terms for a step input. The reference is accurately tracked. The steady state is quickly reached because the actuator is quite stiff. By mixing feedback and feedfoward, the MPC offers a good robustness to modeling errors.

Hardware experiments: The same experiments have been made on the real system. Results are displayed on Fig. 9 and 10. The reference is properly tracked, with the steady state being reached with a time similar to the ideal case. As in simulation, mixing feedforward and feedback helps to obtain a better behavior. On the hardware, the modeling errors are more significant than with the simulated models (due to nonmodeled effects as already mentioned, like periodic friction, etc). The box-DDP is useful in this case to prevent current overshoot in the motor (e.g. between 0.8s and 1.5s). Finally, we show in Fig. 11 the results of tracking a realistic trajectory, taken from a walking movement generated with a pattern generator (trajectory of the knee of robot HRP-2 during 15cm stair climbing). The trajectory is very dynamic. It results in less than 1% of error.

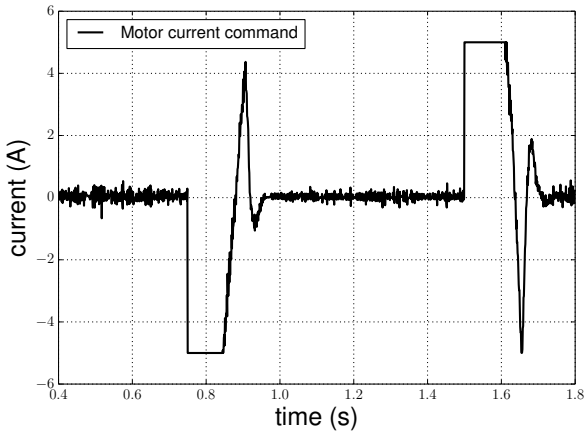

Fig. 10. Hardware experiment: motor current when tracking a step position.

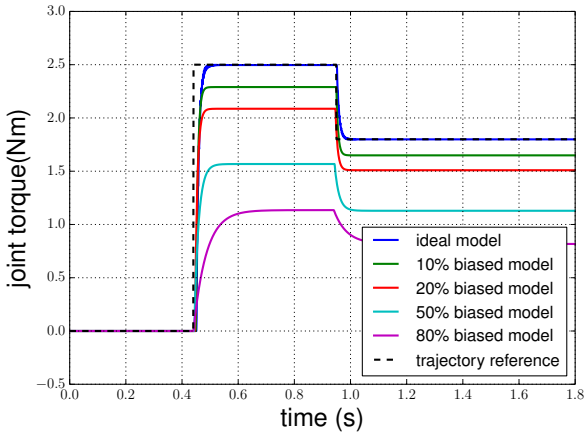

Fig. 12. Torque control of the actuator in simulation for ideal and biased models with several levels of bias

C. Experiments - torque control

We also validate the torque controller, in simulation only. Results are shown in Fig. 12. We again observe the response of the controller to a step input. As before, the analysis is achieved while disturbing the parameters of the model used in simulation while keeping the same MPC model. With a perfect model, the steady state is perfectly reached. The time to steady state is shorter than with the position controller, as expected. When bias is added, we keep a similar behavior but the reference is not perfectly reached any more. As we do not have a direct (nonbiased) estimation of output torque, this would not be possible. However, we also see that the behavior remains good despite the bias and that improving the estimation of the model parameters (in particular the stiffness) on-line based on any external measurement of the output torque would be quite easy.

V. Conclusion

In this paper, we have presented the complete implementation of a new compact, high-gear and transparent actuator, very suitable for mobile robots, in particular in the context

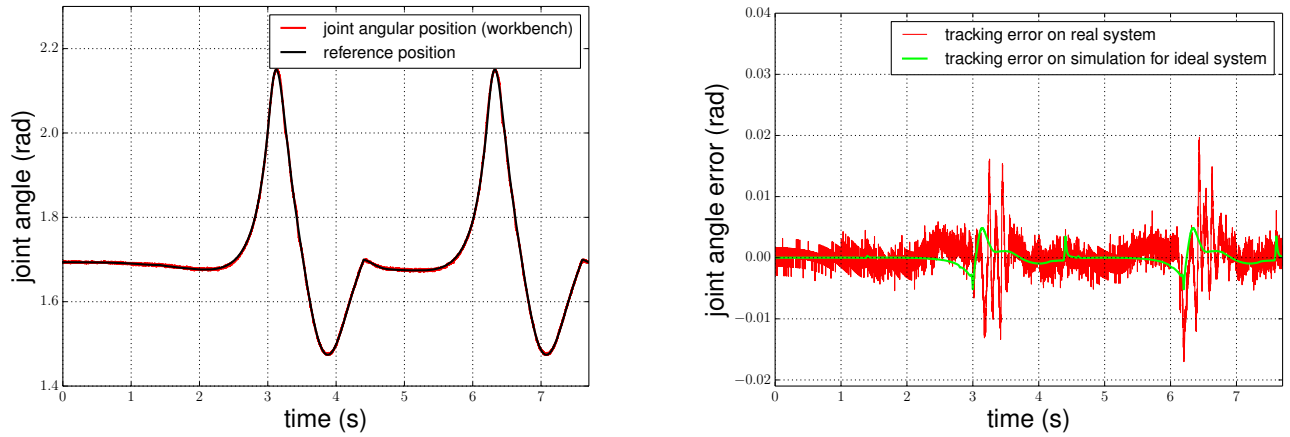

Fig. 11. Tracking a dynamic position trajectory. **(left)** Reference and actual trajectories **(right)** Tracking error. Less than 1% error in average was observed.

of locomotion, with identification and an original two-stage control scheme. 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.

The main result of this study is that the actuator with our control scheme offers very good property, which makes it very suitable to replace strain-wave gears in electric actuation of humanoid robots. In particular, it offers full backdrivability (hence more efficiency, less dangerousness and more chock resistance) and accurate estimation of the output joint torque without direct measurement (i.e. no force sensor needed).

While the proposed control architecture is very suitable for the screw-nut-cable actuator, it is also appropriate for other kind of flexible actuators such as SEA at large, variable stiffness actuators [13] or Mckibben pneumatic actuators [15]. 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.

REFERENCES

[1] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *IEEE Transactions on Robotics*, 2017.

[2] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid *et al.*, "Overview of the torque-controlled humanoid robot toro," in *IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids)*. IEEE, 2014, pp. 916–923.

[3] S. Wolf and G. Hirzinger, "A new variable stiffness design: Matching requirements of the next robot generation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1741–1746.

[4] 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.

[5] S. Alfayad, F. B. Ouezdou, F. Namoun, and G. Gheng, "High performance integrated electro-hydraulic actuator for robotics–part i: Principle, prototype design and first experiments," *Sensors and Actuators A: Physical*, vol. 169, no. 1, pp. 115–123, 2011.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] N. Paine, J. S. Mehling, J. Holley, N. A. Radford, G. Johnson, C.-L. Fok, and L. Sentis, "Actuator control for the nasa-jsc valkyrie humanoid robot: A decoupled dynamics approach for torque control of series elastic robots," *Journal of Field Robotics*, vol. 32, no. 3, pp. 378–396, 2015.

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

[13] 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.

[14] 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.

[15] 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.

[16] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Advances in Neural Information Processing Systems 20*, 2008, pp. 1465–1472.