Compliant and Adaptive Control of a Planar Monopod Hopper in Rough Terrain

Salman Faraji, Soha Pouya, Rico Moeckel, Auke Jan Ijspeert
Biorobotics Laboratory, Institute of Bioengineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

***Abstract*—In this paper, a method is proposed for controlling a hopping monopod. It takes dynamics of the robot into account to have better nominal tracking of desired trajectories and more compliant environmental interactions at the same time. We have incorporated also natural dynamics of the robot into the system by using off-line gaits extracted from optimizations on energy. The main control loop consists of the projected inverse dynamics that generates actuator torques given desired trajectories and also a feedback loop designed and tuned specifically for the structure of the robot. A trajectory generator uses known optimal trajectories together with some stabilizing control laws that modify these trajectories to have better robustness in different situations. The average speed of the robot is also regulated by means of a self-organizing controller. We apply soft transitions in trajectories from phase to phase to avoid sharp actuator input profiles. Our method is successfully tested on a monopod hopper robot in simulation. It can handle slightly rough or sloped terrains while maintaining a given average speed. Simulation results suggest that our method is a promising candidate to control a real robot under construction.**

I. INTRODUCTION

Although wheeled robots are widely used for different applications, legged ones have also their own advantages in locomotion. They can potentially perform better on rough terrains and complex environments. These robots are mostly inspired from animals, but with simpler structures. However, the complexity of structure, considerable number of actuators and degrees of freedom have made the control problem a great challenge for engineers. There are various intelligent approaches developed that control the robot for different tasks, but not so general to handle every kind of terrain or obstacle potentially existing in a complex environment. Control of a robot may require different tasks such as balance control, locomotion control, compliant interaction with environment and self collision avoidance. In addition to these general tasks, problem constraints are also important: for instance, actuator saturation and joint limits. These are constraints on inputs or states of the system and should be considered when designing a controller. Each method therefore has its own advantages and disadvantages regarding task properties.

In this study, we aim to control a monopod robot. Different approaches are described in literature to handle above mentioned tasks. Virtual model control considers virtual components applying external forces to the robot at different points which are then translated into actuator forces/torques to perform the desired task [23]. Operational space control [19][20] is similar in the sense that it also applies these forces externally, but produces them in a different manner. In humanoid robotics context, people use Zero Moment Point (ZMP) [21] approach to maintain stability of the robot. However, these approaches mostly rely on kinematic model of the robot and do not include any information about dynamics which helps to improve nominal tracking and optimality of the robot motion. These requirements are important for a legged robot as it can harm itself by having stiff interactions with environment or falling over. Among existing model-based control approaches, we use projected inverse dynamics method proposed by Aghili [14], because it relies less on mass matrix measurement. Recent works [3][4] also try to generate actuator profiles that minimize contact forces, meaning more compliant environmental reactions. Knowing the dynamics of the robot, we can calculate actuator variables such that they induce energy-optimal actuator inputs.

Pre-optimized gaits that include dynamics of the robot [2] are used as the basis for trajectory generation. Other approaches generate parametric trajectories (for example [8][9]). Generally, transitions of trajectories in legged robots are inevitable due to hybrid states. Parametric trajectories have the benefit of being tunable to preserve continuity in transitions, but they are not optimized for the natural dynamics of the system. [10], [11] and [13] use Splines specifically for the same purpose, but they also lack optimality. In [12], the concept of gait transition is used for AIBO using CPG controllers which is not applicable in our case, since it does not tackle with hybrid states and trajectory errors. Therefore, a challenging part of our study would be handling transitions in the off-line solutions.

The proposed method in this paper is different from previous works as it imitates energy-optimized periodic solutions. It takes advantage from a new adaptive phase transition method and introduces few control laws in flight phase to stabilize an under-actuated robot. Acceptable tracking of desired trajectories is expected while consuming the same levels of energy compared to the optimal solution. We want it to be robust against discretizations in computation, roughness in the terrain, sloped terrains and also potential instability of the optimal solution. It should also be capable of maintaining a given average speed over successive hops. In the next section, we first introduce the dynamics of our robot. We describe the structure and principles of the controller. Underlying mathematics are also included together with a short study

978-1-4673-5643-5/13/$31.00

on under-actuated behaviour of this robot. In the last section, we will then show the effect of different controller blocks step by step. The performance of the robot regarding tracking, robustness and also versatility will be shown quantitatively.

II. METHODS

In this section, we will first introduce the dynamics of the robot and then formalize the mathematical background behind our novel control approach.

A. System Dynamics

Among various legged robot structures, a monopod one is interesting to begin testing locomotion algorithms due to its simplicity. The dimensionality of the system is low and also states to be controlled, either continuous or discrete are not so complicated. The challenge in control of these robots is the fact that they do not have static stability. However, hybrid states play an important role to keep the robot hopping continuously and thus, to maintain dynamic stability. We begin with applying our algorithm to a 2D monopod platform in order to find key elements and strategies required for controlling a legged robot. We will apply it to more complex legged robots in future. The real robot is now under construction in LocoMorph project at BioRob and will be used in future to test this method. It has in fact 2 legs working identically and also being attached to a rotating boom with large enough radius. A leg of this robot consists of a prismatic joint (acting as a knee to change the leg's length) and a rotary hip joint both with series elastic components (spring and damper). Inspired by the previously presented leg models [1] and [22], the trunk-leg dynamics is modelled with three inertial components: one presenting the trunk dynamics and two presenting the leg dynamics (upper one as the thigh and lower one as the shank). In total, there are 3 degrees of freedom for translation and attitude of the base and 2 degrees for joints fully describing the state of the robot. All these variables are depicted on Fig.1.

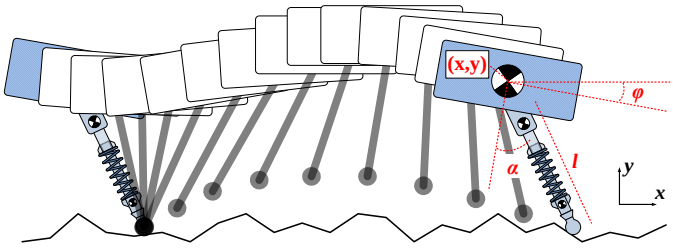

Fig. 1: The schematic of the leg model, its generalized coordinates and the sequence of hopping. The rough terrain used in simulation experiments is a spatially sampled Gaussian sequence.

For this robot, hybrid states consist of flight and stance phases. In the flight phase, there is no contact generating a force that helps to control the robot. Also, we have no actuation on the centre of mass location and orientation as it has a ballistic motion. Whereas in the stance phase, we assume two translational constraints imposed by the contact point resulting in an under-actuated system.

$$\dot{x}_{foot} = 0, \dot{y}_{foot} = 0 \tag{1}$$

Where x_{foot} and y_{foot} indicate the position of the foot. The rigid body dynamics formulation for this system implies:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} + \mathbf{J}_{\mathbf{C}}^{\mathbf{T}}(\mathbf{q})\boldsymbol{\lambda} \tag{2}$$

With $n = 2$ as the number of degrees of freedom. Variables are defined as:

- $\mathbf{q} = [x, y, \phi, \alpha, l]'$: x, y and ϕ are the floating base position and orientation while l and α are the leg and hip joint positions.
- $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n+3 \times n+3}$: the floating base inertia matrix
- $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n+3}$: the floating base centripetal, Coriolis and gravity forces
- $\boldsymbol{\tau} = [0_{1\times 3}\ \tau_1\ \tau_2]'$: joint torques/forces
- $\mathbf{J_C} \in \mathbb{R}^{k \times n+3}$: the Jacobian of k linearly independent constraints
- $\boldsymbol{\lambda} \in \mathbb{R}^k$: the vector of k linearly independent constraint forces

We simulate the robot dynamics with an extended version of [1] in MATLAB which uses orthogonal decomposition in driving the dynamics. Within the simulator, this dynamical system is described by five continuous variables of $\mathbf{q}$ and their derivatives together with a discrete variable corresponding to the hybrid states. So, $\boldsymbol{\lambda}$ and $\mathbf{J_C}(\mathbf{q})$ depend on the hybrid state, being 0 in flight phase or introducing $k = 2$ linearly independent constraints in stance phase. All the variables are normalized in order to make simulations independent of SI dimensions. But the ratios are held the same as the real robot under construction. To give an intuition, the total mass of the robot is 2.423 kg and the leg has a normal length of 29 cm. Regarding actuator limitations, we consider maximum possible torque/force to be produced (7 N.m and 242 N for hip joint and knee respectively). This is verified off-line after the simulation.

B. Control Architecture

As discussed before, we want to incorporate the state of the art projected inverse dynamics in our method. It brings better environmental interactions and more accurate nominal tracking while relying less on sensitive measurements of mass matrix. In [3] and [4], this method is used to control a humanoid robot in simulation and also the LittleDog robot, on a sloped terrain. They deal with optimality by minimizing torque commands and also minimizing ground reaction forces. They use simple sinusoid profiles for joints and also a ZMP-based method to keep the humanoid robot stable. However these sinusoid profiles are not guaranteed to be optimal for locomotion. Mistry [15] also uses operational space trajectories to guide a manipulator. The purpose is to understand better how under-actuation and constraints interact with the dynamics of the task. His method can realize desired trajectories at the end effector, while minimizing the kinetic energy of the articulated arm. Despite being optimal, this method is not yet applied to a floating based robot. Our proposed control approach in contrast is based on optimal gaits found off-line. Based on the work presented by [1], this work [2] has extracted some off-line open-loop periodic solutions for controlling this robot on flat terrain. These solutions include both the gait

initialization parameters and periodic actuation profiles which result in a periodic solution of this non-linear hybrid system (so called gait). These off-line solutions are optimized either for forward velocity or cost of transportation (COT), resulting in open-loop unstable behaviour. Having these solutions at hand, one may think of using simple feedback loops to control the system. However, this approach requires a lot of effort to properly design the feedback due to non-linearity and hybrid behaviour of the system. In addition to all advantages enumerated, to avoid high-gain position control and therefore stiff performance, inverse dynamics which acts as a feed-forward block will help.

For a constrained system, the manifold of unconstrained movements lies in a $\mathbb{R}^{n+3-k}$ dimensional space, determined by constraint Jacobian at each moment. For our 2D robot, as $k < 3$, the system is under-actuated, meaning that it has more DOFs than the number of actuators. Our robot has hybrid states and is basically different from [14], but it is worth seeing why the controller of Aghili cannot work in stance phase. He deduces the controllability condition of the system under his quite similar proposed control law as (for the proof please see [14]):

$$controllability \Leftrightarrow \mathscr{N} \cap \mathscr{B}^{\perp} \subseteq \mathscr{N}^{\perp} \cap \mathscr{B}^{\perp} \tag{3}$$

Where:

- $\mathscr{N}$: free space (or null space of constraints)
- $\mathscr{N}^{\perp}$: constrained space
- $\mathscr{B}$: actuated space : $\{l, \alpha\}$
- $\mathscr{B}^{\perp}$: passive space : $\{x, y, \phi\}$

$\mathscr{N}$ and $\mathscr{N}^{\perp}$ are calculated on-line based on constraint Jacobian $\mathbf{J_C}$. As this matrix is 2×5 and non-zero in stance phase, $\mathscr{N}$ should be a 3 or 4 dimensional subspace. It cannot have 5 dimensions since $\mathbf{J_C}$ is non-zero, and consequently $\mathscr{N}^{\perp}$ is non-empty. The condition for (3) to hold is therefore $\mathscr{N} \cap \mathscr{B}^{\perp} = \varnothing$. Clearly, we cannot find $\mathbf{dq} = \{dx, dy, d\phi, 0, 0\} \in \mathscr{B}^{\perp}$ that induces no movement on foot. Physically, this means to lock the two joints and since $\mathbf{dq}$ is a movement of a rigid body, it includes the foot too. Therefore, (3) can never be satisfied in stance phase. Worse is the flight phase where $\mathscr{N}^{\perp} = \varnothing$ and thus $\mathscr{N} = \mathbb{R}^n$. We should have $\mathscr{B}^{\perp} \subseteq \varnothing$ which is not the case for the floating-base systems.

The way we deal with this problem is to take advantage of hybrid states like [6]. We design a separate controller for each hybrid state and they cooperate to stabilize the robot. In the rest of this section, we will introduce the features of each controller.

1) Stance controller: We will develop this controller in three steps.

a) Feed-forward inverse dynamics as a core: Considering Eq.(2) again, the main problem is the coupling between actuator forces, contact forces and also dynamics of the robot. There exist other methods that depend on either measurement or estimation of contact forces from previous step. The former suffers from noise and delays in sensor reading and filtering. The later is an approximation and requires fast enough control loops; however it can become unstable in adverse situations.

A recent work [3] decouples these forces by projecting the main equation of motion onto null space of actuators. By doing so, it obtains two equations where contact forces do not appear in one. This equation is therefore used for calculating required forces to be applied by actuators knowing desired accelerations. The only sensor required is thus contact detection to determine the discrete state (in addition to encoders and torque sensors). Being in stance phase implies that the position of the foot should be on the ground and its velocity should be zero. The acceleration should be consistent with these conditions too. So we have:

$$\begin{aligned} \mathbf{J_c}\dot{\mathbf{q}} = \dot{\mathbf{x}}_\mathbf{c} = \mathbf{0} \\ \mathbf{J_c}\ddot{\mathbf{q}} = -\dot{\mathbf{J}}_\mathbf{c}\dot{\mathbf{q}} \end{aligned} \tag{4}$$

which introduces two translational constraints in 2D.

The inverse dynamics algorithm calculates QR decomposition of $\mathbf{J_C^T}$ which is done quite fast with modern libraries:

$$\mathbf{J_C^T} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} \tag{5}$$

$\mathbf{Q}$ is an orthogonal matrix ($\mathbf{QQ^T} = \mathbf{Q^TQ} = \mathbf{I}$), and $\mathbf{R}$ is an upper triangle matrix of rank k. Then it projects (2) onto null-space of actuators by multiplying with $\mathbf{Q^T}$. The result is decomposition of the rigid-body dynamics into two independent equations:

$$\begin{aligned} \mathbf{S_cQ^T}(\mathbf{M\ddot{q}} + \mathbf{h}) &= \mathbf{S_cQ^TS^T}\boldsymbol{\tau} + \mathbf{R}\boldsymbol{\lambda} \\ \mathbf{S_uQ^T}(\mathbf{M\ddot{q}} + \mathbf{h}) &= \mathbf{S_uQ^TS^T}\boldsymbol{\tau} \end{aligned} \tag{6}$$

Where $\mathbf{S_c}$ and $\mathbf{S_u}$ are matrices used to select the top and lower portions of the full equation (there are 5 equations for our robot).

$$\begin{aligned} \mathbf{S_c} &= [\mathbf{I}_{k\times k}\mathbf{0}_{k\times n+3-k}] \\ \mathbf{S_u} &= [\mathbf{0}_{n+3-k\times k}\mathbf{I}_{n+3-k\times n+3-k}] \end{aligned} \tag{7}$$

The main point here is that contact forces are not appearing in the second equation. This allows to calculate actuator torques without knowing contact forces.

Pre-optimized trajectories are used to obtain desired acceleration. These optimal trajectories are discrete samples of continuous time-variables of the system. An interpolation is therefore needed to calculate desired values at an arbitrary moment. We use the Spline method to generate smooth interpolations, however we will replace it by linear interpolation to be computationally less expensive while the system is tested to remain stable. Having calculated desired trajectories (here only accelerations), we use the known Moore-Penrose pseudo-inverse method shown by $()^+$ which calculates joint torques that induce optimal actuator command (refer to [3]).

$$\boldsymbol{\tau} = (\mathbf{S_uQ^TS^T})^+\mathbf{S_uQ^T}[\mathbf{M\ddot{q}_{des}} + \mathbf{h}] \tag{8}$$

This is the main equation used inside our feed forward block (refer to Fig.2). It requires only a sensor of discrete phase detection. The aim is to eliminate deteriorating effects of sensor uncertainties, noises and delays in feed-forward torque

calculations. However, due to existence of uncertainties in modelling parameters and also un-modelled dynamics of the robot, a feedback term is necessary to compensate for deviations from desired trajectory. Note that thanks to normalization done inside the simulator for this model, we do not need to deal with inhomogeneity of variables as addressed in [14] and [3].

b) Feedback: In general, when the system is not under-actuated, a feed-forward controller may work together with a feedback law. This loop is used to sustain the desired trajectories. The stance controller is supposed to do its job well as there is no control in the ballistic motion of the next phase. The controller here is of PD type with the control law defined as:

$$\ddot{\mathbf{q}}_{\mathbf{fb}} = \mathbf{K_p}(\mathbf{q_{des}} - \mathbf{q}) + \mathbf{K_d}(\dot{\mathbf{q}}_{\mathbf{des}} - \dot{\mathbf{q}}) \tag{9}$$

Where $\mathbf{K_p}, \mathbf{K_d} \in \mathbb{R}^{\mathbf{2\times 5}}$. $\ddot{\mathbf{q}}_{\mathbf{fb}}$ is then passed through projected inverse dynamics, modifying (8) as:

$$\boldsymbol{\tau} = (\mathbf{S_u Q^T S^T})^{+} \mathbf{S_u Q^T}[\mathbf{M}(\ddot{\mathbf{q}}_{\mathbf{des}} + \ddot{\mathbf{q}}_{\mathbf{fb}}) + \mathbf{h}] \tag{10}$$

There are different scenarios to configure these blocks. However, projecting the acceleration in (10) leads to more stable and smoother actuator profiles. Aghili [14] uses this projection as it yields simpler error dynamics which could be stabilized using enough feedback gains. The properties of this method is also investigated in this book [5].

By now, the controller has two blocks which are supposed to sustain our optimal trajectories. We will show the good performance of these blocks in results section. However, these blocks do not stabilize the robot hopping due to numerical deviations and moreover, they may not work robustly on more complex terrains like sloped or rough ones. So one may think of modifying desired trajectories in order to obtain more robustness and stability with the potential cost of getting far from optimality.

c) Desired trajectories: Uncertainties in modelling, noises in sensor readings and also under-actuated nature of the robot may cause it to deviate from the off-line trajectories. Phase transitions could also take place at different moments compared to the off-line solution. We basically need to solve these problems adaptively while generating desired trajectories.

There could be two kinds of discontinuities. First, due to touch down strike a jump in variable derivatives and also actuator profiles is expected which actually exists in our off-line solution. Second, as we design feedback gains separately for each phase, the same error in both phases may lead to different actuator inputs generated by feedbacks. Our strategy will therefore minimize this effect. Other methods like [8], [9], [10] and [11] change parameters of their polynomials or splines to preserve continuity which could not be applied here, as we use off-line non-parametric optimal trajectories.

At the end of each phase, we assume that feedback controllers have made a specific set of variables in $\mathbf{e}$ and $\dot{\mathbf{e}}$ nearly zero. Let's call these sets $\mathscr{Y}_S \in \mathbb{R}^{n+3-k}$ and $\mathscr{Y}_F \in \mathbb{R}^n$ for stance and flight phases respectively. $\mathscr{Y}_S$ and $\mathscr{Y}_F$ are relating to non-zero elements of feedback matrices. This means that the feedback is then assumed not to generate a considerable force. Note that due to under-actuation, we cannot control for all variables. By design, we may not have $\mathscr{Y}_F = \mathscr{Y}_S^{'}$ or $\mathscr{Y}_F^{'} = \mathscr{Y}_S$ where $\mathscr{Y}_S^{'}$ and $\mathscr{Y}_F^{'}$ are complements of $\mathscr{Y}_S$ and $\mathscr{Y}_F$. This means that the flight controller for example may deal with a variable which was not controlled before. If we let the desired trajectories of $\mathscr{Y}_F \cap \mathscr{Y}_S^{'}$ be still different from their online variables (right after phase transition moment), the error may not be zero any more. Thus the feedback generates a compensating force, leading to discontinuity of force-torque profiles.

After a transition, in the next phase, we smoothly bring the system back to the optimal one. To this end, we damp each (online) variable at the beginning of a new phase while trying to smoothly achieve the (off-line) desired value at the end. This simply means a transition in the form of weighted average between these values. The constraint is continuity of the variable and its first derivative in the beginning of the new phase (ensuring $\mathbf{e}$ and $\dot{\mathbf{e}}$ to be zero) and also having desired off-line values at the end. So, a weighting of the form:

$$\mathbf{q_{des}}(t) = a(t-t_0) \times \tilde{\mathbf{q}}(t) + b(t-t_0) \times \mathbf{q_{opt}}(t) \tag{11}$$

is used which requires these conditions on a and b:

$$\begin{aligned} &a(0) = 1, a(T) = 0, b(0) = 0, b(T) = 1 \\ &\dot{a}(0) = 0, \dot{a}(T) = 0, \dot{b}(0) = 0, \dot{b}(T) = 0 \end{aligned} \tag{12}$$

where T is the expected period of the new phase (from optimal solution), t_0 is indicating the phase transition moment detected by sensors, $\tilde{\mathbf{q}}(t)$ is determined by online variables and $\mathbf{q_{opt}}$ is the optimal solution. We calculate also the two first derivatives of $\mathbf{q_{des}}(t)$ in the same way as required in feedback and feed-forward blocks. It means that we use the same weighting for both of them like (11). Using these soft transitions, neither $\mathbf{K_p}$ nor $\mathbf{K_d}$ will produce impulse-like outputs, while there could be discontinuities in the desired trajectories. In (12), $\tilde{\mathbf{q}}(t)$ will be defined:

$$\tilde{\mathbf{q}}(t) = \dot{\mathbf{q}}(t_0) \times (t-t_0) + \mathbf{q}(t_0) \text{ for } t > t_0 \tag{13}$$

as a linear interpolation of variables after phase transition. Among various weighting choices, exponential weightings of the form:

$$a(t) = e^{-(t/\tau_1)^2}, b(t) = 1 - e^{-(t/\tau_2)^2} \tag{14}$$

are selected because of their tunable damping nature. We choose small τ_1 and τ_2 to reach the desired values at T.

The trajectory modifier restarts the desired trajectory right after transition moment. For many reasons, the past phase may not have the nominal duration. If it exceeds the nominal one, the modifier continues to use trajectories from next phase. Other methods could also be used such as linear, spline or cubic interpolation of previous phase (outside the nominal interval) to continue trajectories. However, using the system's trajectories is more natural. After transition detection however,

the modifier restarts desired trajectory from nominal phase transition point in recorded trajectories. Fig.4 visualizes this.

So far we have introduced the mechanism for soft transitions. This trajectory modifier implicitly has an intrinsic feedback mechanism that performs adaptive trajectory modification. However, in case of more complex terrains like sloped or rough ones it may fail, since the optimal solution is found specifically for flat terrain. Remark that the major problem in stance phase was under-actuation. In next part, we will explain the design of controller for flight phase which collaborates with this stance controller to keep the robot stable.

2) Flight controller: This controller will be developed in two steps. In contrast to stance phase as discussed before, there is no control over base position and attitude of the robot. Therefore, we just use a feedback block and modify the desired trajectories to make the system stable.

a) Feedback: In contrast to previous part, our feedback term is directly applied to actuator inputs here.

$$\mathbf{F_{fb}} = \mathbf{K_p}(\mathbf{q_{des}} - \mathbf{q}) + \mathbf{K_d}(\dot{\mathbf{q}}_{\mathbf{des}} - \dot{\mathbf{q}}) \tag{15}$$

Where again, $\mathbf{K_p}, \mathbf{K_d} \in \mathbb{R}^{\mathbf{2\times 5}}$. The process of designing this feedback is the same as before. The important task is to follow desired trajectories determined in the next Step. We do not use high gains as they will increase energy consumption, lead to stiff behaviour and therefore less robustness.

b) Desired trajectories: Regarding under-actuation problems, we stabilize the robot benefiting from hybrid states. In each phase, a distinct controller has to do its best to prepare the robot for the next phase. Thus, we are looking for simple laws that make the system stable primarily and also more robust on complex terrains. Taking inspiration from the previous works, the following strategies are designed to control trajectories in operational space:

- There are different ways to shape the foot trajectory in the swing phase like [17]. We generally decrease leg's length fast and extend it gradually again to have an arc-shaped foot trajectory. This will prevent the foot from hitting potentially available obstacles in front of it right after lift off. The desired arc-shaped trajectory is induced using different time constants τ_1 and τ_2 (in (14)).
- Adjust the attack angle of the leg in proportion to horizontal speed to improve stability by proper exchange of energy between flight and stance phase. This law was in fact used first by Raibert [6] in 1984, resulting an impressive stability on a real robot.

We will show in section III that these laws can increase manoeuvrability of the system, i.e. working robustly on flat and rough terrains (and sloped ones). In general, we can use more laws like amplifying y trajectories or setting ϕ to zero. But we prefer to keep it as simple and generic as possible. This will require less tuning and thus more straightforward design of controller. One may also desire small variations of base attitude during hopping due to restrictions of instruments installed on the base. To this end, setting ϕ_{des} to zero at the end of stance phase will make sense. In our robot, the base mass is 1.75 kg whereas the mass of the leg is 0.623 kg. Considering the length of the leg, these two parts have comparable inertias. Therefore in flight phase, the swing of the leg will cause the base to rotate considerably. It means that setting ϕ_{des} to zero may not help reduce variations.

These two steps together stabilize the robot. However, depending on the choice of attack angle coefficient, the steady state average speed of the robot will not remain the same as the optimal solution. Larger coefficients may stabilize the robot better, but result in smaller average speeds.

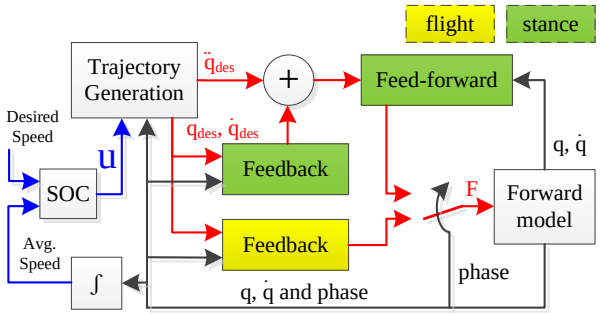

Fig. 2: We have different controllers for each phase, shown with green and yellow colors. In stance phase, the feedback contribution is passed through projected inverse dynamics while in flight phase, feedback directly produces actuator inputs. The trajectory modifier makes smooth transitions and applies additional rules in flight phase.

3) Speed regulator: Till now, there was no control over the average forward speed of the robot. After stabilizing the robot to hop continuously without falling over, one may think of a regulator that will adjust the forward speed of the robot. By introducing such high level controller, some of the computationally expensive methods such as spline interpolation may not be necessary any more (refer to Sec.III).

To adjust the speed of the robot, we have to incorporate a high level mechanism that helps the robot approach a given desired average speed. To this end, we use a Fuzzy Self Organizing Controller (SOC) [7]. Our SOC controller principally has two look-up tables. Based on measured e (error in average speed of a cycle) and $\dot{e}$, the first look-up table generates a proper Δu as delta input of the plant. This table is designed such that the state trajectory will follow a spiral shape to origin in plane of $e - \dot{e}$. The second constant table will update (modify) the first one based on performance of controller according to the desired spiral shape. The advantage of such controllers is their adaptability to the model of the system which may change on-line. The Δu generated by this controller will be integrated and the resulting input u will be used in the system as:

$$\alpha_{opt}(t) \quad = \quad \mu(\dot{x}(t) - \psi u) - \phi_{opt}(t) \tag{16}$$

which resembles the Raibert's law [6]. ψ is set to zero in case of disabling the regulator. After each cycle, the high level controller will calculate the average speed and tick once. We use Fuzzy numerical calculations in our high level controller to have better smoothness compared to a simple look-up table controller.

The overall control architecture proposed in this work is shown in Fig.2. Note that since in flight phase, there is no control over center of mass, we have disabled feed-forward controller and all the job is done by feedback.

III. RESULTS AND DISCUSSIONS

In this section we will perform simulation experiments to show how different blocks are performing in the controller. In these experiments, we will test our control approach for an open-loop unstable periodic solution of the system optimized for energy. Instability is due to the $\dot{x}$ state whose Eigenvalue is bigger than one (Eigenvalues are calculated based on Floquet theory and using Poincaré map calculation [1]).

First we investigate the effect of different preliminary blocks (before using any trajectory modification) in tracking performance. The result is shown in Fig.3, concerning scenarios 1,2 and 3 in Table.I. This table summarizes the features of our control architecture and the scenarios used to verify the performance.

In scenario 1 as discussed before, because of the optimality-stability trade-off, our open-loop solution is intrinsically unstable (due to eigenvalue of bigger than 1 for $\dot{x}$ in Poincaré map, refer to [2]). This is obviously observable in Fig.3. Unstable eigenvalues together with discretizations are deviating the robot from its desired trajectory. Like many control systems, the open-loop behaviour is not needed to be stable (otherwise, it could be too conservative behaviour) and we rely on the closed-loop control strategies to stabilize the system.

In scenario 2, recorded trajectories which are replicated for multiple periods are used in feed forward controller to generate desired actuator variables. In this case, robot's response has considerably precise tracking of the desired value. However, this tracking is good only for one period. Note the small mismatch existing at the end of the first period (around $t = 2$) for α and ϕ. So if we extend the simulation time, discretization errors accumulate and since calculations are highly dependent on the discrete states of robot, after 2-3 periods, the robot falls over. The open-loop solution can continue its periodic behaviour until it gets perturbed due to numerical problems. Thus we do not expect stability of this response.

In scenario 3, a simple feedback is integrated to the system. It tries to keep the system as close to desired trajectories as possible. However, it fails to fully stabilize the system and the robot falls over after few steps. Note that if we cancel the feed-forward, a feedback of at least 10 times larger gains was tested to do the same job which causes more stiffness of the system. There could be more complex feedback design methods like Linear-Quadratic-Gaussian (LQG) as referenced in [18] rather than our easier and earlier procedure. It does the same thing but with taking an objective into account. We look to apply such methods is in future.

So in scenario 4, to have a stabilized controller we integrated the pre-described trajectory modifier to the system according to configuration of Fig.2. Transitions are done with less actuator input jumps and the laws in flight control have made the robot hopping robustly on a flat terrain. But depending on

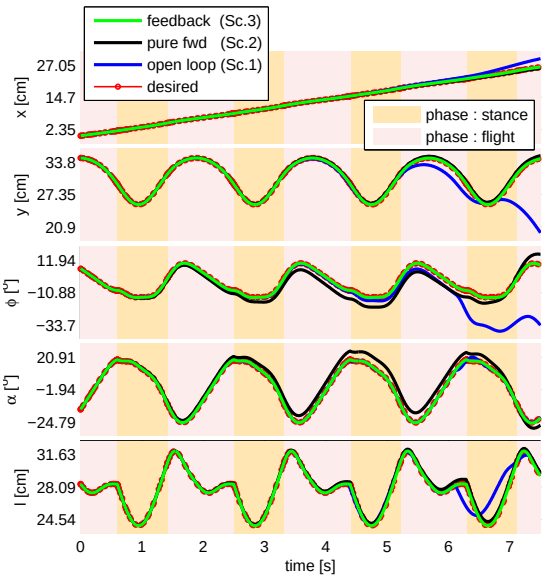

Fig. 3: (Scenarios 1,2 and 3 of Table.I) The open loop time-trajectories of the system, outputs when feed forward block generates desired torques and the outputs after integration of feedback. Note that the feedback improves trajectory following, but it can not stabilize the system forever (we consider 150s of simulation). It also results in stiff and sensitive performance.

attack angle coefficient μ of (16), it attains average speeds different than the nominal one (the larger the μ, the smaller the average speed). The robot could hop continuously on

1- A rough terrain (Fig.1) with $\sigma = 4.06cm$ (Assuming $z \in N(0, \sigma^2)$ is the terrain height, sampled every 2.9 cm, note that the normal length of the leg is 29 cm.
2- Slopes up to 20°.

The response of the system in scenario 4 is similar to the next scenario (Fig.4) in shape of trajectories. What we expected in this stage from the robot was to show robustness on flat terrain and also rough terrains as it is necessary for integration of the high level controller.

To reach arbitrary desired speeds, we use a SOC high level controller in scenario 5. This controller changes attack angle as in Eq.(16) to reach the desired speed. The time-response of the system after integration of the regulator is shown on Fig.4.

- On the top plot of Fig.4 showing energy, our controller consumes more energy mostly in flight phase due to Raibert's law and desired arc shape trajectory. But in stance, using our state of the art projected inverse dynamics method, we can work nearly optimally, imitating the off-line optimal solution. This plot shows that our approach is promising for those who are looking to make their robot energy efficient.
- Remember the discussion on the inertias of base and leg. Generally we expect a constant $\dot{\phi}$ during the flight phase because of ballistic behaviour. However, you can see great change of base attitude resulted from actuation of hip joint to reach the desired attack angle in Fig.4. This actuator in fact consumes more energy than the prismatic one.

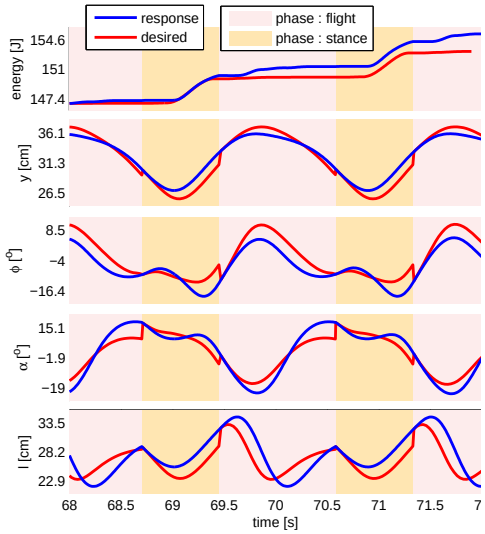

Fig. 4: Steady state response of scenario 5 in Table.I. Soft transitions are observed from phase to phase and also desired trajectories are modified in flight phase so as to stabilize the system. SOC is responsible for reaching the desired average speed which is the same as optimal solution here. Note the top plot showing energy (COT). Mostly in flight phase, our controller consumes more energy compared to the nominal profile. This extra energy is the cost we pay to stabilize the system by changing the nominal off-line solution.

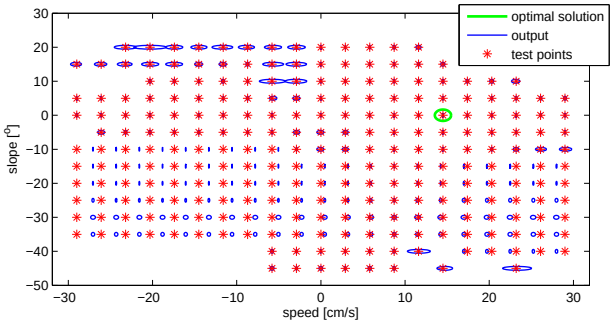

Fig. 5: SOC performance for various (average) speeds over different slopes. Red stars show test points where the robot is stable, blue circles show the performance and the green one shows the off-line solution. In case of horizontal red-blue location mismatch, the controller has a steady state error. The horizontal width of a circle shows the variance of steady state average speed, measured in the second half of a 150s simulation hop by hop, when the controller is stabilized enough. The vertical width shows nothing. The controller can perfectly reach average speeds around the optimal solution, but needs retuning to perform reliably in other regions.

Our closed loop controller (in scenario 5) consumes about 4.21 J of energy per cycle at the speed of nearly 14.5 cm/s, compared to the optimal one which is 3.16 J. The results of testing the high level controller over different slopes and given different desired speeds is shown in Fig.5. The controller can perfectly reach average speeds around the optimal solution. Feedback gains need retuning to perform reliably in other regions.

The general trend in first scenarios is to use computationally heavy methods to sustain the off-line solution found by optimizations. But we use simpler basic methods in more complex controllers with motivation to implement them on a real robot which works on-line. It remains stable if we interpolate the optimal solution linearly instead of using Spline method which is computationally expensive. Our controller has intrinsically more compliant environmental interactions due to integration of inverse dynamics. The robot is hopping robustly on different terrains in simulations while being nearly optimal as it imitates an off-line periodic solution optimized for energy. See http://biorob.epfl.ch/page-80131.html for movies of all scenarios.

IV. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper, a method is proposed for the control of legged robots. This method takes dynamics of the robot into account to have better nominal tracking of the desired trajectories and more compliant environmental interactions at the same time. We have incorporated also natural dynamics of the robot into the system by using off-line calculated gaits extracted from optimizations on energy. The main control loop in this method consists of a feed-forward block that generates actuator torques given desired trajectories and also a feedback block designed and tuned specifically for the structure of the robot. A trajectory generator uses known optimal trajectories together with some control laws that essentially modify these trajectories to have robustness on different terrains.

The platform of the interest in this work is the well-known planar monopod hopper modelled and studied in previous works to study leg dynamics and control. The specific problem with this system is two-folded; on one hand this is an under-actuated system in stance and has flight phase where there is no contact with the ground to control the robot through the contact forces. We simulated our method on such robot with two actuators and five degrees of freedom. On this robot, we could also integrate a high level controller that made the robot capable of performing well on sloped or considerably rough terrains.

Overall, our method shows good tracking performance on our monopod robot. There are two mechanisms that help the robot having more compliant interactions with environment. First, it uses pre-optimized trajectories that contain natural dynamics of the robot. Second, having a feed-forward term in the loop helps to rely less on stiff reactions of the PD controller. The same performance is obtained with at least 10 times smaller gains. The core feed-forward method also takes advantage from a pseudo-inverse method that basically calculates actuator torques inducing optimal contact forces. Different steps are considered in designing the controller. However, to make the approach real time and implementable on a real robot, using simpler basic methods in terms of computation seem to have negligible effect on the performance of robot, especially stability and optimality.

B. Future Works

In developing and tuning process of this method, it will be more straightforward if better methods are used for feed-back design. In addition, we will verify this method on the

TABLE I: Comparison of different scenarios used in experiments. First block shows different methods and segments used in the controller of a scenario. The second block shows the performance of that controller, as tested in simulation experiments.

Scenario	1	2	3	4	5
Feature	open-loop	pure feed-forward	+feedback	+traj. modifier(speed≈3.2 cm/s)	+regulator(speed≈14.5 cm/s)
trajectory interpolation	-	spline	spline	linear	linear
stance phase controller	-	fwd	fwd+fb	fwd+fb	fwd+fb
flight phase controller	-	fwd	fwd+fb	fb	fb
soft transitions	-	no	no	yes	yes
Raibert's law	no	no	no	yes	yes
arc shaped foot trajectory	no	no	no	yes	yes
stability	no	no	no	yes	yes
rough terrain (max σ)	-	-	-	4.06cm	2.32cm
sloped terrain	-	-	-	-45° to 20°	-35° to 15°
following a desired speed	no	no	no	no	yes

real monopod robot as well. The next step is applying this algorithm to a biped and then a quadruped. In such robots, the sequence of phase changes could differ from nominal trajectories, which makes the control problem sophisticated. However they have less problems with under-actuation and it is easier to formulate the problem.

V. ACKNOWLEDGMENTS

This work has received funding from the EPFL and from the European Community's Seventh Framework Programme FP7/2007-2013 - Future Emerging Technologies, Embodied Intelligence, under the grant agreements no. 231688 (Locomorph).

REFERENCES

[1] C. David Remy, Optimal Exploitation of Natural Dynamics in Legged Locomotion, PHD thesis, Swiss Federal Institute of Technology Zurich, (2011).

[2] S. Pouya, M. Khodabakhsh, R. Moeckel and A. J. Ijspeert, How Morphological and Control Parameters Affect the Locomotion Performance of a 2-DOF Single Leg Design, Technical report, Locomorph project annual review meeting, pp. 75-92, March 2012.

[3] M. Mistry, J. Buchli and S. Schaal, Inverse Dynamics Control of Floating Base Systems Using Orthogonal Decomposition, IEEE International Conference on Robotics and Automation, pp. 3406-3412, May 2010.

[4] L. Righetti, J. Buchli, M. Mistry and S. Schaal, Control of legged robots with optimal distribution of contact forces, IEEE-RAS International Conference on Humanoid Robots, October 2011.

[5] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, Robotics Modelling, Planning and Control, Springer, August 2008.

[6] M. H. Raibert, H. Benjamin Brown Jr and M. Chepponis, Experiments in Balance with a 3D One-Legged Hopping Machine, The I. J. of Robotics Research, vol. 3, no. 2, pp. 75-92, June 1984.

[7] J. Jantzen, The Self-Organising Fuzzy Controller, Technical report, Technical University of Denmark, Aug 1998.

[8] X. Wang, T. Lu and P. Zhang, Study on State Transition Method Applied to Motion Planning for a Humanoid Robot, International Journal of Advanced Robotic Systems, vol. 5, no. 2, pp. 145-150, 2008.

[9] E. Cuevas, D. Zaldívar and R. Rojas, Walking trajectory control of a biped robot, Technical report, Freie Universität Berlin, November 2004.

[10] Z. Tang, C. Zhou and Z. Sun, Trajectory Planning for Smooth Transition of a Biped Robot, Proceedings of the IEEE International Conference on Robotics and Automation, September 2003.

[11] A. A. Thant and K. K. Aye, Application of Cubic Spline Interpolation to Walking Patterns of Biped Robot, World Academy of Science, Engineering and Technology, 2009.

[12] C. P. Santos, V. Matos, Gait transition and modulation in a quadruped robot: A brainstem-like modulation approach, Robotics and Autonomous Systems, vol. 59, pp. 620-634, September 2011.

[13] J. Bruce, S. Lenser and M. Veloso, Fast parametric transitions for smooth quadrupedal motion, In Proceeding of RoboCup 2001: The Fifth RoboCup Competitions and Conferences, pp. 293-298, 2001.

[14] F. Aghili, A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: Applications to control and simulation, IEEE Transactions on Robotics, vol. 21, no. 5, pp. 834–849, October 2005.

[15] M. Mistry and L. Righetti, Operational Space Control of Constrained and Underactuated Systems, Proceedings of Robotics: Science and Systems, June 2011.

[16] C. D. Remy, K. Buffinton and R. Siegwart, A MATLAB Framework for Efficient Gait Creation, International Conference on Intelligent Robots and Systems - IROS , pp. 190-196, 2011.

[17] Y. Sakakibara, K. Kan, Y. Hosoda, M. Hattori, M. Fujie, Foot Trajectory for a Quadruped Walking Machine, International Workshop on Intelligent Robots and Systems - IROS, pp. 315-322, 1990.

[18] M. Hutter, M. Hoepflinger, C. Gehring, M. Bloesch, C. D. Remy, R. Siegwart, Hybrid Operational Space Control for Compliant Legged Systems, Proceedings of Robotics: Science and Systems, July 2012.

[19] O. Khatib, A united approach to motion and force control of robot manipulators: The operational space formulation, The Int. J. of Robotics Research, vol. 3, no. 1, pp. 43-53, 1987.

[20] L. Sentis, J. Park, O. Khatib, Compliant Control of Multicontact and Center-of-Mass Behaviors in Humanoid Robots, IEEE Transactions On Robotics, vol. 26, no. 3, pp. 483-501, June 2010.

[21] M. Vukobratović and B. Borovac, Zero-moment point thirty five years of its life, Int. J. Humanoid Robot, vol. 1, no. 1, pp. 157–173, 2004.

[22] K.D. Mombaur, R.W. Longman, H.G. Bock, J.P. Schloder, Stable one-legged hopping without feedback and with a point foot, Proceedings of IEEE Int. Conf. on Robotics and Automation ICRA, vol. 4, pp. 3978-3983. 2002.

[23] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, G. Pratt, Virtual Model Control: An Intuitive Approach for Bipedal Locomotion, The Int. J. of Robotics Research, vol. 20, no. 2, pp. 129-143, February 2001.