Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.
Planning Natural Locomotion for Articulated Soft Quadrupeds

Mathew Jose Pollayil, Cosimo Della Santina, George Mesesan, Johannes Englsberger, Daniel Seidel, Manolo Garabini, Christian Ott, Antonio Bicchi, and Alin Albu-Schäffer

Abstract—Embedding elastic elements into legged robots through mechanical design enables highly efficient oscillating patterns that resemble natural gaits. However, current trajectory planning techniques miss the opportunity of taking advantage of these natural motions. This work proposes a locomotion planning method that aims to unify traditional trajectory generation with modal oscillations. Our method utilizes task-space linearized modes for generating center of mass trajectories on the sagittal plane. We then use nonlinear optimization to find the gait timings that match these trajectories within the Divergent Component of Motion planning framework. This way, we can robustly translate the mode-aware centroidal motions into joint coordinates. We validate our approach with promising results and insights through experiments on a compliant quadrupedal robot.

I. INTRODUCTION

Compliant legged robots [1]–[3] are being increasingly used as part of the effort to match the mobility of living beings [4]. Exploiting their inherent elastic motions promises more natural movements and efficiency [4], [5]. Certainly, by taking inspiration from muscles, tendons, and other soft tissues, introducing elastic elements in the mechanics can promote energy-saving and high-speed locomotion [6], [7].

This notwithstanding, such trend does not tally with a corresponding attention towards locomotion planning techniques for legged robots with elasticity: in particular, efficient motion planning for compliant walkers remains to date an under-explored topic. Few authors have addressed the problem of finding motions to locomote following the natural dynamics of the robotic system and with minimal reliance on feedback: for instance, see [8]–[12]. Data-driven approaches have also been applied successfully with the aim of learning to store and release elastic energy in the legs [13], [14]. However, all these works have the common limitation of tackling the problem only from a control perspective and never actually attempting to generate physically feasible trajectories that comply with the natural system motions of elastic robots. Perhaps, this is due to the mismatch between the complexity of such systems and the simplification commonly required by motion planning.

Indeed, traditional motion generation strongly reduces system dynamics through template models [15]. This has the advantage of taming the complexity of planning. Examples of templates are the Linear Inverted Pendulum Model (LIPM) [16], or the Spring-Loaded Inverted Pendulum (SLIP) [17] - all of which have been successfully used in locomotion planning. Nevertheless, using templates may deprive planners of necessary information about the mechanical system - such as its compliance, its multi-body dynamics, and its nonlinear natural oscillations. These oscillations can be studied with nonlinear modal theory [18]–[21]. Some preliminary applications to locomotion have been proposed (e.g. [22], [23]). But - at least at their current stage - these mathematical descriptions are too complex and not yet sufficiently understood to be directly applied to general quadrupedal locomotion.

A strategy that sits in between these two extremes may potentially combine the capability of the template-based planners to manage complex systems and the power of modal analysis to exploit the robot’s natural dynamics. This paper aims to provide an example of such an architecture (Fig. 1). We build upon DCM-based planning framework [24] - of which this work represents the first implementation on a quadrupedal robot - so as to optimally select some free parameters for maximally matching the natural dynamics of the robot. Our planner can generate joint trajectories that produce smooth and natural center of mass (CoM) evolutions, which are then implemented by low level feedforward controllers. We validate our approach on the articulated soft quadruped DLR Bert [1] (Fig. 2).

II. OVERALL MAP OF THE PROPOSED FRAMEWORK

We give a general overview of the proposed architecture, as shown in Fig. 1 and summarized by Algorithm 2. In the following sections, the building blocks will be explained more in detail. First, modal oscillations of the linearized
systems around planned stances equilibria are combined to obtain natural CoM trajectories (Sec. III). We then use nonlinear motion optimization to find the optimal gait timings to match the DCM-based CoM trajectory to the natural motion given by the modal oscillations (Sec. V). Such motion is then made physically feasible by using the DCM motion planning framework [25] (Sec. IV) using the optimized timings. The plans returned by our architecture can be implemented with any low level controller. Here, we use feedforward inverse kinematics (IK) augmented with gravity compensation (Sec. VI). Note that the planner’s input - a sequence of stances $\Sigma$ and step heights $\Delta h$ - can be evaluated by standard foot-step planners (e.g. [26], [27]).

III. SIMPLIFIED MODAL EVOLUTION

This section discusses the automatic generation of a CoM evolution composed by interconnecting linear modal oscillations. This part corresponds to “Natural Trajectory Generation” in Fig. 1 and Find Natural Trajectory in Algorithm 2.

A. Modal Oscillations of Floating-Base Robots

Consider the elastic model of a compliantly actuated floating-base robot

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = S^T K (\theta - q_i) + J^T_i(q)\lambda,$$

$$B\dot{\theta} + K(\theta - q_i) = \tau.$$ (1)

Here, $q = [q^T, \theta^T] = [q^T, \theta^T] = [q^T, \theta^T] = [q^T, \theta^T] = [q^T, \theta^T] = [q^T, \theta^T] = [q^T, \theta^T] = [q^T, \theta^T] = [q^T, \theta^T]$ is the vector of generalized coordinates of the robotic system, $q_i \in \mathbb{R}^{n_b}$ is the vector of virtual base coordinates, and $q_i \in \mathbb{R}^{n_b}$ is the vector of link positions of the legs, with $n = n_b + n_j$. We denote by $\theta \in \mathbb{R}^{n_j}$ the vector of motor positions, by $\tau \in \mathbb{R}^{n_j}$ the motor torques vector, and by $\lambda \in \mathbb{R}^{n_c}$ the contact forces vector, where $n_c$ is the number of constraints imposed by contacts.

Assuming $n_c \leq n_j$, according to the size of the contact constraint $J(q, \dot{q}) = 0$, we partition the generalized coordinates $q$ into independent and dependent variables, $q_i \in \mathbb{R}^{n_c}$ and $q_D \in \mathbb{R}^{n_c}$, respectively. It is noteworthy that, with $n_c \leq n_j$, the independent variables $q_i$ contain the base coordinates $q_b$, which can be chosen to be a parametrization of the CoM position $x = (x_1, x_2, x_3)^T$ and orientation without loss of generality.

Subsequently, we use the Lagrange-D’Alembert Method to solve the constraint

$$\begin{bmatrix} J_{c,1} & J_{c,D} \end{bmatrix} \begin{bmatrix} \dot{q}_i \\ \dot{q}_D \end{bmatrix} = 0 \Rightarrow \dot{q}_D = -J^{-1}_{c,D} J_{c,i} \dot{q}_i.$$ (2)

We re-write the generalized coordinates as a function of the independent ones:

$$q = \begin{bmatrix} T_{D,i} \\ t \end{bmatrix} \dot{q},$$

Replacing into (1), we obtain:

$$\ddot{q}_i + T^i \left( J_l^i \right) \dot{q}_i + \phi^i \left( \dot{q}_i \right) = M_i \left( q_{eq} \right) \ddot{q}_i + K_i \left( q_{eq} \right) \Delta \dot{q}_i = 0,$$

where $\Delta \dot{q}_i = q_i - q_{eq}$, the inertia of the independent coordinates is given by $M_i(q) = T^i \left( J_l^i \right)$, and the Hessian of the potential energy gives the effective stiffness

$$K_i(q) = \frac{\partial}{\partial q} \left( T^i \left( J_l^i \theta - q_i \right) \right) \approx T^i \left( K + \frac{\partial\hat{g}}{\partial q} \right) T.$$ (3)

The matrix $K = S^T KS$ is the equivalent stiffness of the generalized coordinates and $T$ is considered quasi-static. The modal oscillations of (3) can be found by studying the generalized eigenvalue problem

$$\lambda \Omega(q_{eq}) u_i = K_i(q_{eq}) u_i,$$ (4)

and are of the form

$$\Delta q_i(t) = u_i A \sin(\sqrt{\lambda_i} t + \psi).$$ (5)

These oscillations are commonly referred to as the eigenmodes of the linearized system (3). In general, there are $n - n_c$ distinct oscillations $\Delta \dot{q}_i(t)$ about $q_{eq}$. They evolve on the subspaces (eigenspaces) generated by $u_i$, which are the eigenvectors of $K^{-\frac{1}{2}} M_i \Omega^{-\frac{1}{2}} K_i^{-\frac{1}{2}}$. The oscillation frequency is the square-root of the associated eigenvalue $\lambda_i = \sqrt{\lambda_i}$. Eq. (5) parametrizes the small natural oscillatory motions of the independent variables $q_i$ of (1), constrained by (2), and close enough to $q_{eq}$. The amplitude $A$ and phase $\psi$ depend on the initial conditions $\Delta \dot{q}_i(0)$ and $\Delta \dot{q}_i(0)$.

Remark 1. With $n_c \leq n_j$, eq. (5) by itself describes the natural oscillations $\Delta x$ of the CoM around the corresponding equilibrium.

B. Natural Motions of the Center of Mass

In this subsection, we propose the method to generate CoM trajectories that are compliant with the natural dynamics utilizing the theory presented in Sec. III-A

Assumption 1. We assume to be provided with a sequence of $n_s$ stances $\Sigma = \{\sigma_1, \ldots, \sigma_{n_s}\}$ that have been planned to move the robot. Here, a stance $\sigma_i = \{c_k|k = 1, \ldots, n_k\}$ is a set of limb contacts [28].
The generation of natural motions of the CoM is as follows: we choose modal oscillations around the stances equilibria and concatenate them in order to move in the planned direction. As mentioned, we plan these motions only in the sagittal plane of the robot. We reserve to lift this simplification in future works. We define the following procedure, which is also described in a compact way in Algorithm 1:

- Compute a succession of \( n_s \) admissible equilibria \( Q_{eq} = \{ q_{eq}^{(1)}, \ldots, q_{eq}^{(n_s)} \} \) of (1) corresponding to the contact positions given by the precomputed stances in \( \Sigma \). That is, we request that a \( q_{eq}^{(i)} \), found by means of IK from a given stance, complies with the static equilibrium condition \( g(q_{eq}^{(i)}) = S^T K(\theta_{eq} - S q_{eq}^{(i)}) + J^T(\theta_{eq} - S q_{eq}^{(i)}) \lambda_{eq} \), while being subject to friction cone constraints on \( \lambda_{eq} \) and to bounds on both \( \theta_{eq} \) and \( \lambda_{eq} \). This problem can be solved effectively using a quadratic program, which can also minimize some cost related to the contact and elastic forces: for instance, \( \frac{1}{2} \lambda_{eq}^T \lambda_{eq} + \frac{1}{2} (\theta_{eq} - S q_{eq})^T K(\theta_{eq} - S q_{eq}) \). This is performed by ComputeAdmissibleEquilibria.

- Next, for each of the equilibria in \( Q_{eq} \), the function ComputeCoMPositions computes forward kinematics and finds the associated CoM positions \( X_{eq} = \{ x_{eq}^{(1)}, \ldots, x_{eq}^{(n_s)} \} \).

- For each element \( x_{eq}^{(i)} \in X_{eq} \), using (4), compute the related eigenspaces \( E_{eq}^{(i)} = \lambda_{1,2}^{(i)}, u_{1,2}^{(i)} \) to get a list \( E_{eq} = \{ E_{eq}^{(1)}, \ldots, E_{eq}^{(n_s)} \} \). This is done by ComputeEigenSpaces.

- Initialize an empty variable for the trajectory \( x^*(t) \) using function EmptyTrajectory.

- Then, iteratively repeat the following for each successive pairs of equilibria \( (x_{eq}^{(i)}, x_{eq}^{(i+1)}) \in X_{eq} \):
  - Find all possible intersection points of the corresponding eigenspaces in \( E_{eq}^{(i)} \) and \( E_{eq}^{(i+1)} \) (FindIntersections).
  - From these intersections, choose a random \( \hat{x} \) through the function ChooseRandom.
  - Then, ComputeOscillation computes the trajectories \( x'_{eq}^{(i)}(t) \) and \( x''_{eq}^{(i)}(t) \), from \( x_{eq}^{(i)} \) to \( \hat{x} \) and from \( \hat{x} \) to \( x_{eq}^{(i+1)} \), considering the appropriate time intervals.

That is, we find a motion \( x'_{eq}^{(i)}(t) \), which follows modal oscillation with eigenfrequency \( \sqrt{\lambda^{(i)}} \) and is along the associated eigenvector \( u_{1,2}^{(i)} \) connecting the equilibrium \( x_{eq}^{(i)} \) to the state \( \hat{x} \), and such that \( \forall t' : x'_{eq}^{(i)}(t') = \hat{x} \) we have \( x'_{eq}^{(i)}(t') = 0 \). Such a motion is given by \( x'_{eq}^{(i)}(t) = x_{eq}^{(i)} + \Delta x'_{eq}^{(i)}(t) \), where \( \Delta x'_{eq}^{(i)}(t) = u_{1,2}^{(i)} A' \sin(\sqrt{\lambda^{(i)}} t) \) with \( A' = ||\hat{x} - x_{eq}^{(i)}|| \). The trajectory \( x'_{eq}^{(i)}(t) \) goes from \( x_{eq}^{(i)} \) to \( \hat{x} \) in the time interval \( [0, \frac{\pi}{2\sqrt{\lambda^{(i)}}}] \). A similar procedure can be followed to find the motion from \( \hat{x} \) to \( x_{eq}^{(i+1)} \) by computing \( x''_{eq}^{(i)}(t) = x_{eq}^{(i+1)} + \Delta x''_{eq}^{(i)}(t) \), where \( \Delta x''_{eq}^{(i)}(t) = u_{1,2}^{(i+1)} A'' \sin(\sqrt{\lambda^{(i+1)}} t) \), \( A'' = ||\hat{x} - x_{eq}^{(i+1)}|| \).

- Finally, append the found \( x'_{eq}^{(i)}(t) \) and \( x''_{eq}^{(i)}(t) \) to the natural one \( x^*(t) \) (AppendTrajectory).

Remark 2. Given the periodicity of gaits, there exists a subset \( \Sigma \) of the stances \( \Sigma \) that contains all information about the chosen gait. Hence, it suffices to apply Algorithm 1 only to \( \Sigma \) and concatenate the resulting \( x^*(t) \) appropriately to find the natural CoM trajectory associated with \( \Sigma \).

An example trajectory, computed using Algorithm 1 and following the natural dynamics, is shown in Fig. 3. Therein, from three equilibria \( x_{eq}^{(1)} \), \( x_{eq}^{(2)} \), and \( x_{eq}^{(3)} \) of a trot gait for

**Algorithm 1** Natural Trajectory of the Center of Mass

**Input:** \( \Sigma \) **Output:** \( x^*(t) \)

1: procedure FIND_NATURAL_TRAJECTORY(\( \Sigma \))
2: \( Q_{eq} \leftarrow \) ComputeAdmissibleEquilibria(\( \Sigma \))
3: \( X_{eq} \leftarrow \) ComputeCoMPositions(\( Q_{eq} \))
4: \( E_{eq} \leftarrow \) ComputeEigenSpaces(\( X_{eq} \))
5: \( x^*(t) \leftarrow \) EmptyTrajectory()
6: for \( x_{eq}^{(i)} \in X_{eq} \) do
7: \( E_{eq}^{(i)} \leftarrow \) GetEigenSpaces(\( E_{eq}, x_{eq}^{(i)} \))
8: \( E_{eq}^{(i+1)} \leftarrow \) GetEigenSpaces(\( E_{eq}, x_{eq}^{(i+1)} \))
9: \( \{ \hat{x}_{int} \} \leftarrow \) FindIntersections(\( E_{eq}^{(i)}, E_{eq}^{(i+1)} \))
10: \( \hat{x} \leftarrow \) ChooseRandom(\( \{ \hat{x}_{int} \} \))
11: \( x'_{eq}^{(i)}(t) \leftarrow \) ComputeOscillation(\( x_{eq}^{(i)}, \hat{x} \))
12: \( x''_{eq}^{(i)}(t) \leftarrow \) ComputeOscillation(\( x_{eq}^{(i)}, x'_{eq}^{(i)} \))
13: \( x^*(t).\)AppendTrajectory(\( x'_{eq}^{(i)}(t) \))
14: \( x^*(t).\)AppendTrajectory(\( x''_{eq}^{(i)}(t) \))
15: return \( x^*(t) \)

6595
the quadruped DLR Bert, we concatenate modal oscillations to get a natural motion of the CoM.

Trajectories found by Algorithm 1, even though compliant with the mechanics of the robot, are not guaranteed to be physically feasible, in the sense of locomotion. This means that they disregard the physical constraint that the line of action of the sum of external forces acting on the CoM must intersect the base of support. Traditional motion planning efficiently deals with such a constraint. For this reason, in Sec. IV we quickly review the three-dimensional DCM planning, which will be used to “follow” CoM references generated by Algorithm 1.

IV. PHYSICAL FEASIBILITY OF COM TRAJECTORIES
We discuss here briefly the Divergent Component of Motion layer of our architecture (“DCM Motion Planning” in Fig. 1 and DCMPlanning in Algorithm 2), only to the extent that it is needed to this paper. It is indeed worth stressing out here that - with the exception of two mappings of the quadruped gait phases to the conventional single support and double support phases of bipeds - this block is not a contribution of this work. It is instead an implementation of the theory laid by [25], [29].

The DCM is a point \( \xi = (\xi_1, \xi_2, \xi_3)^T \) defined as \( \xi = x + i\dot{x}, \) where \( x \) is the center of mass (CoM) position, \( \dot{x} \) its velocity, \( b = \sqrt{\Delta z/g} \) is a constant, \( \Delta z \) the average height of the CoM above the ground and \( g \) the gravitational acceleration. The second order dynamics of the CoM can be rewritten as

\[
\ddot{x} = -\frac{1}{b}(x - \xi), \quad \ddot{\xi} = \frac{1}{b}(\xi - v).
\]  

Here, \( v = (v_1, v_2, v_3)^T \) is the so called Virtual Repellent Point (VRP), Eq. (6) shows that the CoM has a stable first order dynamics that converges to the DCM and the DCM has an unstable first order dynamics that diverges pushed by the VRP. From a planning perspective, the problem of finding feasible CoM trajectories reduces to planning properly \( v. \) Solving (6) with suitable boundary conditions and a feasible VRP trajectory \( v(t) \) yields closed-form feasible \( C^2 \) continuous trajectories of the CoM. This process is briefly summarized below.

1) Trajectory Generation: The motion of the CoM is divided into \( n_\phi \) transition phases. For each phase \( \phi \in \{1, \ldots, n_\phi\}, \) we design VRP trajectories \( v_\phi(t). \) Then, the corresponding \( \xi_\phi(t) \) and \( x_\phi(t) \) are found by integrating (6). The boundary values of each phase are some \( v_{\phi,0}, \xi_{\phi,0} \) and \( x_{\phi,0} \) for the start points and \( v_{\phi,T}, \xi_{\phi,T} \) and \( x_{\phi,T} \) for the end points. Continuity of the whole trajectories is guaranteed by connecting start and end points of successive phases: \( v_{\phi,0} = v_{\phi-1,T}, \quad \xi_{\phi,0} = \xi_{\phi-1,T}, \) and \( x_{\phi,0} = x_{\phi-1,T}. \) These points can be seen as waypoints of the trajectories. The generation of these waypoints is discussed later on.

For each phase, VRP waypoints are smoothly interpolated in time as

\[
v_\phi(t) = (1 - f_\phi(t))v_{\phi,0} + f_\phi(t)v_{\phi,T}, \tag{7}
\]

where, \( t \in [0,T_\phi], \) \( T_\phi \) is the constant phase duration, and the function \( f_\phi(t), \) called temporal interpolation function, is a polynomial with the following properties: (i) \( f_\phi(0) = 0, \) (ii) \( f_\phi(T_\phi) = 1, \) (iii) \( 0 \leq f_\phi(t) < 1, \forall t \in [0, T_\phi]. \) Substituting (7) in (6) and solving, for each phase, with boundary condition \( \xi_{\phi,0} = \xi_{\phi,T}, \) yields

\[
\xi_\phi(t) = \alpha_{\phi,\xi}(t)v_{\phi,0} + \beta_{\phi,\xi}(t)v_{\phi,T} + \gamma_{\phi,\xi}(t)\xi_{\phi,T}, \tag{8}
\]

with \( \alpha_{\phi,\xi}(t), \beta_{\phi,\xi}(t), \) and \( \gamma_{\phi,\xi}(t) \) being nonlinear functions of \( t, b \) and \( T_\phi. \) Similarly, combining (8) and the first equation in (6), and solving for the boundary condition \( x_\phi(0) = x_{\phi,0}, \) yields

\[
x_\phi(t) = \alpha_{\phi,x}(t)v_{\phi,0} + \beta_{\phi,x}(t)v_{\phi,T} + \gamma_{\phi,x}(t)\xi_{\phi,T} + \delta_{\phi,x}(t)x_{\phi,0}. \tag{9}
\]

Analogously to (8), also \( \alpha_{\phi,x}(t), \beta_{\phi,x}(t), \) and \( \delta_{\phi,x}(t) \) are nonlinear functions of \( t, b \) and \( T_\phi. \)

In summary, given the start and end waypoints for each phase, smooth trajectories of VRP, DCM and CoM can be found using (7), (8), and (9), respectively, and concatenated to find the whole trajectories \( v(t), \) \( \xi(t), \) and \( x(t). \)

2) Waypoint Generation: Waypoints are computed from the given sequence of \( n_\sigma = n_\phi + 1 \) stances in \( \Sigma. \) We place VRP waypoints \( V = [v_1 \ldots v_{n_\sigma}]^T \in \mathbb{R}^{n_\sigma \times 3} \) for each \( \sigma_i \in \Sigma. \) Our strategy is to choose the \( v_i \) based on the equilibria \( x_{eq} \) computed for the stances \( \sigma_i \) in Sec. III-B. We set \( v_i \) with the same horizontal position, however we impose the height to be \( \Delta z. \) This is to be consistent with the definition of the time constant \( b. \) Notice that these \( v_i \) are the same waypoints that were previously denoted as start and end points. For instance, for the \( i^\text{th} \) phase \( v_{\phi,0} = v_i \) and \( v_{\phi,T} = v_{i+1}. \) DCM waypoints \( X = [x_1 \ldots x_{n_\sigma}]^T \in \mathbb{R}^{n_\sigma \times 3} \) and CoM waypoints \( X = [x_1 \ldots x_{n_\sigma}]^T \in \mathbb{R}^{n_\sigma \times 3} \) can be found from V by evaluating (8) for \( t = 0, \) and (9) for \( t = T_\phi, \) for all \( n_\phi \) phases, and writing the resulting relations in matrix form. In this way, waypoints can be efficiently found as follows [29]

\[
X = [\Xi C_V \Xi c_\xi \Xi c_x] \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix}, \quad X = [x C_V x c_x x c_z] \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}. \tag{10}
\]

Here, \( \xi_i = \xi_{x_i} = v_{n_{\sigma_i}} \) for stopping the robot and \( x_i = x(0). \) The matrices \( \Xi C_V, \Xi c_\xi, \Xi c_x, \) and \( \Xi c_z \) depend on the coefficients \( \alpha_{\phi,\xi}(0), \beta_{\phi,\xi}(0), \gamma_{\phi,\xi}(0), \alpha_{\phi,x}(T_\phi), \beta_{\phi,x}(T_\phi), \gamma_{\phi,x}(T_\phi), \) and \( \delta_{\phi,x}(T_\phi). \) Please see [29] for more details.

Remark 3. The formulation of DCM-based planning adopted in this paper [24], [29] requires that the average height of the CoM does not deviate significantly from \( \Delta z. \) Hence, oscillations in the vertical direction are not present.

3) Phases Mapping: Here, we propose a simple mapping from quadrupedal to bipedal phases to ease the implementation of the DCM framework on multi-legged robots.

Definition 1 (Quadrupedal Single Support). We define as single support the phases in which the VRP reference is constant. In these phases, one or more legs are swinging.

Definition 2 (Quadrupedal Double Support). We define as double support the phases in which the VRP reference is interpolated between two waypoints. These correspond to stance phases, in which no legs are swinging.
4) Feet Trajectories: The trajectories of the limbs $r_k(t)$, with $k \in \{ LF, RF, LH, RH \}$, can be computed from the stances $\Sigma$ as usually done in the literature, i.e. using cubic or quintic splines. In this work, we use third-order polynomials. Note that in our formulation, planning feet motions only requires the step height $\Delta h$, since the stances in $\Sigma$ already specify the step forward-length and side-length.

V. Motion Optimization for Natural Walking

This section aims to enable the DCM-based CoM trajectory (eq. (9) of Sec. IV) to follow CoM motions, compliant with the natural dynamics, which are generated by Algorithm 1 of Sec. III-B. This part corresponds to “Motion Optimization” in Fig. 1 and MotionOptimization in Algorithm 2.

We notice that the trajectories generated by (9) can be shaped by varying either the phase times $T_\phi$ or the VRP waypoints $V$. Hence, $T_\phi$ and $V$ can be used to set up an optimization to match the $x(t)$, computed using (9), to the natural CoM motion $x^*(t)$ over a number of $p \leq n_\phi$ significant phases.

In this work, we choose to optimize over the phase times since we believe that matching the natural frequencies of the system will already produce desirable results (as confirmed by the experiments in Sec. VI). The possibility of finding online “optimal” VRP adjustments is left to a future work.

We choose as cost function the least-squares difference between the two trajectories:

$$w(x(t), x^*(t)) = ||x(t) - x^*(t)||_2^2.$$ (11)

Thereafter, we set up the nonlinear optimization

$$\min_{T_1, \ldots, T_p} w(x(t, \Sigma, X), x^*(t))$$ s.t. $l_\phi \leq T_\phi \leq u_\phi$ (12)

Here, $l_\phi$ and $u_\phi$ are arbitrary lower and upper bounds on the phase times. Recall that both $\Sigma$ and $X$ are nonlinear functions of the phase times $T_\phi$; moreover, the coefficients $\alpha_{\phi, \xi}$, $\beta_{\phi, \xi}$, $\gamma_{\phi, \xi}$, $\alpha_{\phi, x}$, $\beta_{\phi, x}$, $\gamma_{\phi, x}$, and $\delta_{\phi, x}$ all depend nonlinearly on $T_\phi$. This is the reason why the cost function $w$ is nonlinear with respect to the optimization variables $T_\phi$.

In practice, nonlinear optimization problems, such as (12), can be addressed with state-of-the-art numerical methods, such as branch-and-bound and SQP algorithms [30], [31], or other methods as [32]–[35]. Also note that nonlinear optimization (12) is not performed online in Algorithm 2, thus avoiding heavy computations during online planning. Only the DCM-based trajectory generation, which is proven to be very efficient [24], is carried out online.

We applied optimization (12) using as $x^*$ the trajectory computed in Sec. III-B and shown in Fig. 3. Here, we optimized only over the first two phases (single and double support), which we henceforth call $T_{swing}$ and $T_{stance}$ for the sake of clarity. Assuming a symmetric robot and periodic trot, these two phases fully specify the gait. In line with Remark 3, we used as $\Delta \phi$ the average of the vertical component of $x^*$ (the $x_3$ of Fig. 3b). The resulting DCM-based optimal CoM trajectory $x_d(t)$, which uses the optimal $T_{swing}$ and $T_{stance}$, is shown in Fig. 4. It is noteworthy that the forward component of the natural trajectory is followed as close as possible by the second-order dynamics of the CoM (see Fig. 4a). However, given our particular choice of VRPs, by virtue of Remark 3, Fig. 4b shows that the vertical oscillations are zeroed to the average height.

At this point, having defined all the necessary concepts, we merge our CoM modal evolution method (Sec. III-B) into the DCM planning framework (Sec. IV) using the proposed motion optimization (Sec. V). Thus, we build our complete planning architecture, shown in Algorithm 2.

VI. Experimental Validation

In this section, we validate the proposed approach on the compliant quadruped DLR Bert [1]. The robot weighs 3.2 kg. Its length, width, and height are approximately 0.33 m, 0.21 m, and 0.20 m in zero position. The stiffness of the joint springs is 2.7 Nm/rad.

A. Implementation Details

We purposely do not employ any feedback controller for two reasons: 1) for a fair evaluation of the planned trajectories, and 2) because it is known that feedback actions on compliant robots have a stiffening effect [36]. Hence, we perform offline closed-loop inverse kinematics [37]

$$q_{id} = J_{ip}^{-1}[s_{id} + K_P(s_{id} - s_k)],$$ (13)

Algorithm 2 Natural Motion Planning. $\Sigma$ is a pre-planned sequence of stances and $\Delta h$ the step height. FindSubset finds the subset of stances $\Sigma$ that are periodically repeated in $\Sigma$.

**Input:** $\Sigma$, $\Delta h$ **Output:** $x_d(t)$, $r_{id}(t)$

1: **procedure** NATURAL MOTION PLANNING($\Sigma$)
2: $\Sigma \leftarrow$ FindSubset($\Sigma$)
3: $x^*(t) \leftarrow$ FIND NATURAL TRAJECTORY($\Sigma$)
4: $(T_1, \ldots, T_p) \leftarrow$ MotionOptimization($x^*(t)$)
5: **repeat**
6: $x_d(t) \leftarrow$ DCMPlanning($\Sigma$, $T_1, \ldots, T_p$)
7: $r_{id}(t) \leftarrow$ FeetTrajectories($\Sigma$, $\Delta h$, $T_1, \ldots, T_p$)
8: ForwardPlan($x_d(t)$, $r_{id}(t)$)
9: **until** stopping condition
B. Experiments: Results and Discussion

We performed three experiments with trot gait. Initially, we planned the CoM motion exclusively based on DCM planning without motion optimization. We stress again that, to the best of the authors’ knowledge, this is the first time that this framework is employed for quadrupedal robots. For the sake of comparison, we also let the robot walk with small “non-optimal” timings. Finally, we used our planner (Algorithm 2) to optimize the phase times, which were found to be very close to the eigenfrequencies of the linearizations. The three experiments are shown in the photo sequences of Fig. 5a, 5b, and 5c and in the attached submission video.

In the first experiment (shown in Fig. 5a), we used the following two phase durations: $T_{\text{swing}} = 0.5$ s and $T_{\text{stance}} = 0.4$ s. We noticed that since we do not consider at all the elasticity of the robot, the behaviour is undesirable; the robot bounced back and forth due to the elastic forces and tracking errors. Instead, the second experiment ($T_{\text{swing}} = 0.2$ s and $T_{\text{stance}} = 0.1$ s), although not as bad as the first one, exhibited some small disturbing oscillations. This is probably due to the non-compliance with the system eigenfrequencies. Finally, for the third experiment, we use the optimal phase durations $T_{\text{swing}} = 0.1335$ s and $T_{\text{stance}} = 0.1019$ s output by Algorithm 2. The corresponding natural and optimized trajectories, $x^*(t)$ and $x_d(t)$, were already shown in Fig. 3 and Fig. 4, respectively. The trot was visually impressive and very robust notwithstanding the relative IK and the feedback control. The robot seemed to abide by its intrinsic elasticity. This shows that our planning method complies with the natural dynamics of the system.

Furthermore, we observed that, in the first two experiments, due to the non-conformity to the natural dynamics, the robot movements were considerably more jerky and unsteady when compared to the third experiment (please see the CoM paths, drawn using the video tracking software Kinovea1, in Fig. 5 and watch the attached video submission). Furthermore, notice also that with Algorithm 2 the robot covered more distance than with the DCM planning with non-optimal timings, despite the same total time of 5 s.

VII. Conclusions

In this paper, we addressed the challenge of planning natural locomotion with articulated soft quadrupeds. The proposed architecture generates CoM trajectories on the robot’s sagittal plane using modal oscillations. By means of nonlinear optimization, phase durations are found so that said natural trajectories are matched by the DCM planning framework. We achieved promising - although preliminary - experimental results with the quadruped DLR Bert. Future work will extend our natural trajectory generation beyond the sagittal plane and use VRF adjustments to optimize the trajectories. We will also employ nonlinear modes to better capture the robot’s natural dynamics. Finally, we will conduct a thorough and quantitative evaluation of energy efficiency: this was not included at present due to space constraints.

\footnote{https://www.kinovea.org/}
REFERENCES


