Modeling robot geometries like molecules, application to fast multi-contact posture planning for humanoids

Salman Faraji[1], and Auke Jan Ijspeert[1]

***Abstract*—Traditional joint-space models used to describe equations of motion for humanoid robots offer nice properties linked directly to the way these robots are built. However, from a computational point of view and convergence properties, these models are not the fastest when used in planning optimizations. In this paper, inspired by Cartesian coordinates used to model molecular structures, we propose a new modeling technique for humanoid robots. We represent robot segments by vectors and derive equations of motion for the full body. Using this methodology in a complex task of multi-contact posture planning with minimal joint torques, we set up optimization problems and analyze the performance. We demonstrate that compared to joint-space models that get trapped in local minima, the proposed vector-based model offers much faster computational speed and a suboptimal but unique final solution. The underlying principle lies in reducing the nonlinearity and exploiting the sparsity in the problem structure. Apart from the specific case study of posture optimization, these principles can make the proposed technique a promising candidate for many other optimization-based complex tasks in robotics.**

***Index Terms*—Humanoid Robots, Kinematics, Biologically-Inspired Robots, Task Planning, Gesture, Posture and Facial Expressions, Optimization and Optimal Control**

I. INTRODUCTION

HUMANOID robots normally have many degrees of freedom, enabling them to perform various complex tasks ranging from locomotion to manipulation. For cyclic motions like walking, model-based approaches dominantly rely on simplifications and periodicity analysis techniques to stabilize the motion [1]. In a larger scale, however, especially for locomotion in complex environments, one would require a plan ahead of time. Dynamics of the robot, contact surface configurations, inherent geometric and actuation limitations and finally, the presence of gravity make motion synthesis and control complicated. In a model predictive control paradigm [2], handling large perturbations in an online fashion depends directly on a high planning speed, i.e. the ability to synthesize feasible motions in short time spans. A simple model that captures main effects is favorable since it offers fast computational capabilities. On the other hand, optimality and feasibility analysis require dynamic information and indeed more geometric details. Therefore, a computationally fast yet inclusive kino-dynamic model is paramount in multi-contact motion planning problems.

Apart from motion planning, finding feasible and optimal postures for manipulation and balancing tasks requires a fast geometric solver. One popular approach is to find a time evolution of internal coordinates that converges to the optimal posture. This can be done via inverse kinematics (IK) [3] or inverse-dynamics (ID) formulations [1], [4] which offer compliance as well. In these time-integration controllers, an optimal posture is found naturally, but with certain dynamics. This is because only first or second order equations are solved to find velocities or accelerations, favoring their linear properties. Although optimization problems are solved as fast as $1-2$ *ms* in each control tick, it takes time for the robot to follow trajectories, i.e. to integrate velocities and accelerations. It typically takes $1-2$ *s* to reach the final goal, depending on robot capabilities [3].

An alternative to time-integration approaches is to solve for positions directly in a single control tick. This is more suitable for planning postures ahead of time, e.g. to check feasibility or optimality of a multi-contact posture in new environmental conditions. This is, in fact, the same as solving a nonlinear inverse-kinematics optimization problem [5]. The nature of this single-shot optimization is obviously different from time-integration approaches. However, the role of Hessians and Jacobians in finding descending search directions seems equivalent to finding accelerations and velocities in time-integration approaches.

Planning "optimal" postures with minimal joint torques is, however, far more complex than simple IK problems due to inclusion of contact forces and joint torques (which are complex functions of the geometric configuration and contact forces) as optimization variables. In these tasks, the inherent redundancy in the system is solved by optimizing the overall joint torques while in normal IK methods, a desired position and orientation is considered for the pelvis, torso or the Center of Mass (CoM). The task of optimizing postures is well addressed in the literature, for example in matching a model with recorded human data in OpenSim [6] or analyzing multi-contact human postures in car assembly lines [7]. In static optimal postures, the equations of motion ensure stability of the humanoid while in normal IK methods, the Zero Moment Point (ZMP) or CoM is forced to lie within the support polygon [5].

Reaching real-time performance is important, but yet a

Manuscript received: February, 16, 2017; Accepted July, 16, 2017.

This paper was recommended for publication by Editor Nikos Tsagarakis upon evaluation of the Associate Editor and Reviewers' comments. This work was funded by the WALK-MAN project.

[1]Salman Faraji and Auke Jan Ijspeert are with Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL) salman.faraji@epfl.ch

Digital Object Identifier 10.1109/LRA.2017.2739103

secondary objective. More fundamental problems are convergence properties and nonlinearity of the equations which often affect planning problems. Handling the inherent redundancy, joint limits and of course minimization of joint torques are major critical factors for optimal posture planning. Apart from increased computation time, these issues often lead to local minima which make the final solution very sensitive to the starting point in optimizations. These issues limit the application of optimal posture planning in online control. Besides, sensitivity analysis and convergence properties are rarely discussed or addressed in motion planning and modeling papers.

Our goal is to approach geometric optimization problems differently. Traditional joint space models encapsulate all geometric complexity inside the robot model and lead to very nonlinear equations. In this work, we propose writing mechanical equations in a different way, inspired by molecules. We unroll the compacted inherent complexity of the body and distribute it over the external tasks as well. We demonstrate that formulating kinematic optimization problems becomes much easier in this way while better speed and convergence properties are achieved. We limit our focus on a case study of optimal posture optimization and provide an insightful comparison with joint-space models. A wider range of applications, however, including forwards and inverse dynamics as well as motion planning can be foreseen with this modeling technique.

II. Molecule-inspired modeling

The proposed modeling technique is based on methodologies used in molecular mechanics where atoms are represented by points linked together with certain bonds. A very fundamental task in computational chemistry is to determine geometric shapes or conformations of the molecules. Well-known techniques like Nuclear Magnetic Resonance (NMR) spectroscopy or X-ray crystallography reveal information about the mean position and size of atoms as well as length and type of bonds in different materials. Then, various optimization techniques are used to find exact positions of atoms in 3D space, given certain energy functions that determine the optimal length for the bonds and the angles between them. These optimization problems can have multiple local or global solutions depending on the level of energy in the molecule. Figure.1.A shows an example conformation, a member of NanoPutian series called NanoBalletDancer [8] which is an artificially synthesized anthropomorphic molecule that resembles human. This particular shape is indeed engineered by experts knowing principles of bond geometries. However, a popular molecular mechanics software (Spartan [9] for example) can also find this conformation by minimizing energy levels.

In molecular mechanics, there are two dominant coordinates used to model configurations: natural internal vs. Cartesian coordinates [10]. In the former, keeping atomic distances fixed, one would use bond angles to express 3D coordinates of atoms. This is very similar to traditional joint-space modeling in robotics. In the latter approach, however, each atom is assigned a 3D position constrained to have certain distances from neighboring atoms. These holonomic constraints can also describe

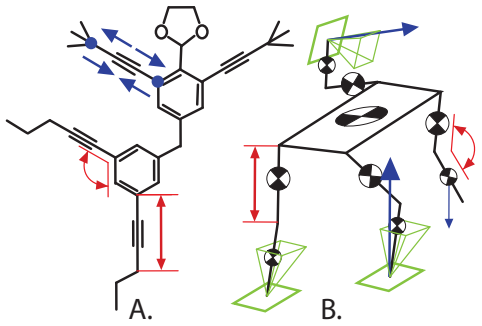

Fig. 1: **A.** NanoBalletDancer, a member of NanoPutian artificial molecules [8] designed with an anthropomorphic shape. **B.** a typical humanoid robot in multi-contact interaction with the environment, assigned to pick up something with the right hand. Geometric similarities between chain structures (shown in red) make 3D formulations interchangeable. Dynamic interactions are, however, very different (shown in blue, attractive and repulsive forces in molecules versus gravitational and contact forces in robots), indicating that equations of motion might have different properties.

certain angular limitations between the bonds, expressed by vector operations [10]. A constrained optimization would then treat these constraints via undetermined Lagrangian multipliers in Verlet time integration [11]. More formally, imagine a set of atoms represented by 3D coordinates r_i, a potential function $U(r^N)$ where $r^n = (r_1, r_2, ..., r_N)$ and a set of constraints:

$$\chi(r_i, r_j) = |r_i - r_j|^2 - b_{ij}^2 = 0 \tag{1}$$

where b_{ij} represents the distance between atoms i and j. Cartesian equations of motion take the form:

$$m_i \ddot{r}_i = -\frac{\partial U(r^N)}{\partial r_i} - \sum_{k \in K_i} \lambda_k \frac{\partial \chi_k}{\partial r_i} \tag{2}$$

where m_i is the mass of atom i and the set K_i represents its neighboring atoms. The Verlet time integration updates positions iteratively by:

$$r_i^{(t+\Delta t)} = \hat{r}_i^{(t+\Delta t)} + \sum_{k \in K_i} \lambda_k \frac{\partial \chi_k}{\partial r_i} \Delta t^2 m_i^{-1} \tag{3}$$

where Δt is the integration time-step and $\hat{r}_i^{(t+\Delta t)}$ is the unconstrained position of atom i. Putting these update rules into (1) yields:

$$\chi^{t+\Delta t}(r_i, r_j) = |r_i^{(t+\Delta t)} - r_j^{(t+\Delta t)}|^2 - b_{ij}^2 = 0 \tag{4}$$

which should be satisfied in the next time step. By solving this nonlinear system of equations, one would obtain λ_k, necessary to correct the unconstrained positions during integration. Verlet integration is very similar to nonlinear optimization in the sense that Lagrange multipliers are helping to satisfy constraints. However, in molecular dynamics, one would assume weak coupling between constraints and use linear approximations (Gauss-Seidel method) of the non-linear system to solve for λ_k iteratively (SHAKE algorithm [11]). Interior point methods are also popular [12] and provide faster convergence rates if all couplings are considered [13]. However, solving full-dimensional system of equations might still be time-consuming, despite their sparse structure.

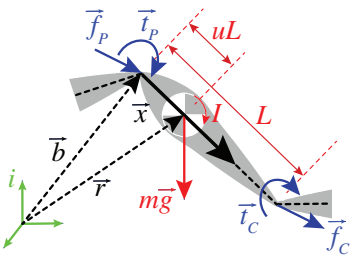

Fig. 2: Notations used to model a segment of the robot by vectors in the inertial frame. The unitary vector $\vec{x}$ represents the segment alignment expressed in the inertial frame as well as the interaction forces and torques (shown in blue). Model-specific parameters are also shown in red.

Even though the weak coupling assumption makes the convergence slow, the intrinsic properties of Cartesian coordinates together with the power of Lagrange multipliers let the atoms move freely to find better solutions, especially in the case of having constraints in the system. In unconstrained conditions, however, natural internal coordinates are more suitable for finding molecular conformations [10]. We adapt this idea and use Cartesian coordinates to solve multi-contact posture optimization problems for humanoid robots where a large number of constraints are involved. We only take inspiration from geometric similarities of molecules and humanoids. Interaction forces and "tasks" have entirely different natures, depicted in Figure.1. Indeed, the nature of atomic and molecular forces is very different from gravity. Therefore, we limit our focus on static conditions in this paper. Exploring dynamic similarities and motions are left for future work.

A. Vector-Based Equation of Motion

Dynamic equations and the notion of potential fields for robots are not similar to molecules due to the different nature of dominant forces. Besides, we are not going to use Verlet integration to solve the posture optimization problem, although it might be possible in an inverse-dynamics paradigm. Our goal is to 1: find an alternative for joint-space coordinates 2: formulate optimization problems and 3: solve them via interior-point methods, aiming at improving the computation time and convergence properties.

Similar to Cartesian coordinate systems used for molecules, we model each segment of the robot body by an approximative vector expressed in the inertial frame (Figure.2), linking the parent and child joints together. This vector freely rotates in 3D space while its length is constrained. The inertial frame Center of Mass (CoM) position for this body segment is:

$$\vec{r} = \vec{b} + (uL)\ \vec{x} \tag{5}$$

where the vector $\vec{b}$ denotes parent joint position, L represents the segment length, u quantifies the relative CoM position across the segment and the unitary vector $\vec{x}$ is the free variable shown in Figure.2. Writing equations of motion for this system in Cartesian coordinates is straightforward:

$$\begin{aligned} \vec{f}_p - \vec{f}_c &= m\ (\ddot{\vec{b}} + uL\ \ddot{\vec{x}} - g) \\ \vec{t}_p - \vec{t}_c &= uL\ (\vec{x} \times \vec{f}_p) + (1-u)L\ (\vec{x} \times \vec{f}_c) + I\ (\vec{x} \times \ddot{\vec{x}}) \end{aligned} \tag{6}$$

where $\times$ stands for cross product, $\vec{f}_p$ and $\vec{t}_p$ are incoming Cartesian forces and torques applied by the parent link, $\vec{f}_c$ and $\vec{t}_c$ are outgoing forces and torques applied to the child link and I is the inertia around any axis orthogonal to $\vec{x}$ (refer to Figure.2). We call (6) constrained equation of motion, because of holonomic constraints on $\vec{x}$:

$$\vec{x}^T\vec{x} = 1, \quad \vec{x}^T\dot{\vec{x}} = 0, \quad \vec{x}^T\ddot{\vec{x}} + \dot{\vec{x}}^T\dot{\vec{x}} = 0 \tag{7}$$

There are two underlying assumptions: the CoM of the segment is aligned with joint positions, and the rotational inertia around $\vec{x}$ is negligible. These assumptions are realistic for most humanoid robots. Otherwise, an auxiliary vector $\vec{y}$ subject to $\vec{y}^T\vec{y} = 1$ and $\vec{x}^T\vec{y} = 0$ should be added, to reconstruct the local frame to account for the asymmetry incurred. The advantage of modeling a robot with vectors lies in the ability to express the tasks (end-effector positions) with a linear combination of segment vectors. Besides, joint angles and torques are all expressed with a quadratic combination of variables. Equations (6) and (7) indeed provide a geometric transformation of the traditional joint-space equations. Therefore, all translational and rotational dynamics are still included. Next, we are going to use these properties and establish a full model for our humanoid robot, consisting of limbs and a torso.

B. Case Study: Humanoid Robot

The spirit of vector-based equations is based on breaking the geometry of the robot into individual vectors with certain quadratic relations. Here, we consider a generic humanoid robot with a torso and four limbs (indexed by j), described by the following variables:

- $\vec{b}$: inertial-frame position of the base (root).
- $\vec{e}_1$, $\vec{e}_2$, $\vec{e}_3$: orthonormal basis for orientation of the torso.
- $\vec{x}_1^j$, $\vec{x}_2^j$: unitary vectors for limb segments.
- $\vec{t}_1^j$, $\vec{t}_2^j$: Cartesian torques in the knee, elbow, hip and shoulder joints.
- $\vec{F}^j$: contact forces constrained in a friction polyhedral with coefficient μ^j.
- $\vec{T}^j$: contact torques. The resulting Center of Pressure (CoP) is limited to a square of size w^j.

All these quantities are shown in Figure.3. A dynamic model indeed includes derivatives of position vectors as well. Note that parameters s_1^j, s_2^j and s_3^j are used to express the hip and shoulder positions, M and $I = [I_x, I_y, I_z]$ represent mass and inertial properties of the torso and finally, each link segment has its own geometric and inertial properties shown in Figure.2. The task parameters are also given by desired contact positions $\vec{P}^j$ and contact surface coordinates $\vec{N}^j$. We skip writing full equations, but principles of (6) are simply applied here to describe the relation between variables. The inter-segment interaction forces (formerly $\vec{f}_p$ and $\vec{f}_c$) are indeed resolved by combining equations together. An important property of such modeling technique is adding vectors together to reach the desired end-effector point $\vec{P}^j$. This linear constraint potentially simplifies geometry optimization tasks, discussed in the next section.

Note that since most of the humanoid robots are constructed with articulated joints, we need to encode joint limits into

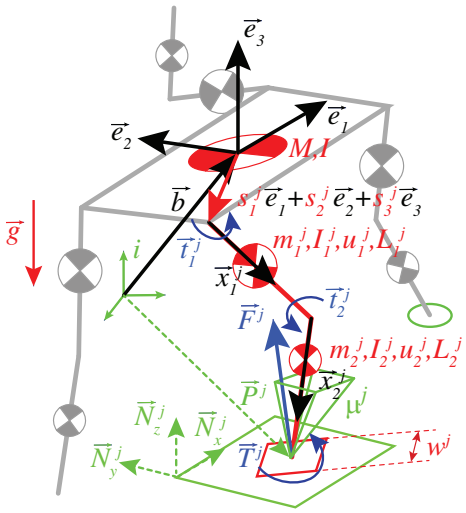

Fig. 3: A humanoid robot can be represented by a set of vectors in Cartesian space; all expressed in the inertial frame. Fixed model parameters are shown in red, given task parameters including desired contact positions, and orientations are shown in green and free dynamic variables (forces and torques) drawn in blue. Black arrows are free position variables (Refer to the text for further information). Green areas denote surfaces that can provide a supporting force and green circles denote Cartesian points to be reached, without establishing a contact.

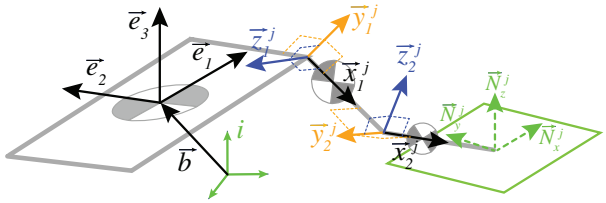

Fig. 4: Reconstructing local frames on each body segment to encode joint-space boundaries of articulated joints. These auxiliary orthonormal frames can be either added as decision variables during the optimization to impose joint limits, or calculated offline to convert from vector-space to joint-space.

the framework as well. To this end, we can introduce auxiliary vectors $\vec{y}^j$ and $\vec{z}^j$ for each segment and reconstruct the orthonormal local frame together with $\vec{x}^j$. Joint-limits can then be directly added by constraining the dot product of these vectors together or with the given contact frame vectors $\vec{N}^j$. A simple demonstration of these local frames is found in Figure.4. We do not necessarily need to add derivatives of these vectors unless explicit bounds on joint velocities are desired. Note that Cartesian joint torques $\vec{\tau}^j$ can also be projected onto these vectors to find real actuator torques if needed.

III. Application: Static Posture Optimization

In this section, we use models developed earlier to formulate a task of posture optimization. We consider point contacts and drop contact torques $\vec{T}^j$ to simplify the system, although adding them is straightforward due to linearity. In this case, CoP and rotational friction constraints are linear functions of contact torques and vertical forces [4]. Besides, to give the robot more freedom, we let it rotate the feet (and the wrists) around the normal axis of the contact surface ($\vec{N}_z^j$ vector in Figure.3). This relaxation makes the system redundant but helps rotate the body and avoid restricting joint limits, aiming at finding more optimal postures. The redundancy can be removed by introducing orientation constraints or other simple policies explained later. We also limit our focus on static postures in this paper but consider six tasks that require very different interactions with the environment. The purpose is to assess the baseline convergence behavior of a minimal setup in symmetric and asymmetric postures as well as in starting from random initial conditions. Joint torque limits can also be added as described earlier, but practically, we found these constraints never activated. We use the model of our robot Coman [4] with an approximate mass of $30kg$, a height of $1m$ and joint torque limits of $40Nm$.

A. Optimization setup

1) Optimization variables: The algorithm has to find position vectors $\vec{b}$, $\vec{e}_1$, $\vec{e}_2$, $\vec{e}_3$, $\vec{x}_1^j$ and $\vec{x}_2^j$, contact forces $\vec{F}^j$ and joint torques $\vec{t}_1^j$ and $\vec{t}_2^j$ for each joint j together with auxiliary vectors $\vec{y}_1^j$, $\vec{y}_2^j$, $\vec{z}_1^j$ and $\vec{z}_2^j$ to reconstruct local frames when joint angle bounds are considered. For the joint-space model in a similar optimization setup, we consider the generalized state vector $q \in \mathbb{R}^{22}$, joint torques $\tau \in \mathbb{R}^{16}$ and contact forces $\vec{F}^j$.

2) Objective function: Consists of two dominant terms: the magnitude of contact forces and joint torques. However, we observed that both modeling approaches (joint-space and vector-based) could hardly converge to a unique solution, starting from random initial conditions. This is mainly due to the nonlinearity of the cross products and redundancy in the system. Therefore, we added an auxiliary linear term to lift the base (root) up as much as possible and to orient the limbs along the sagittal vector of the contact surface ($\vec{N}_x^j$, shown in Figure.3). These terms implicitly reduce joint torques and remove the redundancy mentioned earlier. The objective function for vector-based model is therefore described as:

$$f = Q_\tau(\textstyle\sum_j |\vec{t}_1^{\,j}|^2 + |\vec{t}_2^{\,j}|^2) + Q_F(\textstyle\sum_j |\vec{F}^j|^2) + \\ Q_{lin}(-\vec{\alpha}^T\vec{b} + \beta \textstyle\sum_j (\vec{x}_1^j - \vec{x}_2^j)^T \vec{N}_x^j) \tag{8}$$

where the vector $\vec{\alpha} = \begin{bmatrix}0,0,1\end{bmatrix}^T$ denotes upward direction, $\beta = 0.01$ and Q_τ, Q_F and Q_{lin} are 0 or 1, simply enabling or disabling the terms. The parameter β is used to remove the redundancy, but it can indeed affect the overall posture as well, pushing the knees or elbows away from straight postures. Therefore, we choose a small value to minimize this effect yet removing the redundancy. Later, we provide a further analysis and show that the choice of this parameter is not critical. Besides, adding contact orientations to the tasks relieve the need to have any β in the objective function. Note that the same objective function is also used for the joint-space model:

$$f = Q_\tau|\vec{\tau}|^2 + Q_F(\textstyle\sum_j |\vec{F}^j|^2) + \\ Q_{lin}(-\vec{\alpha}^T\vec{b}(q) + \beta \textstyle\sum_j (\vec{x}_1^j(q) - \vec{x}_2^j(q))^T \vec{N}_x^j) \tag{9}$$

where the base positions and limb vectors are functions of the generalized state vector q.

3) Constraints: A set of equality constraints describe the equations of motion, linking all variables together. In the vector-based model additionally, linear task constraints are added for each limb j:

$$\vec{b} + s_1^j \vec{e}_1 + s_2^j \vec{e}_2 + s_3^j \vec{e}_3 + L_1^j \vec{x}_1^j + L_2^j \vec{x}_2^j - \vec{P}^j = 0 \qquad (10)$$

while end-effector positions are complex functions of the generalized state vector ($\vec{EF}^j(q)$) in the joint-space model:

$$\vec{EF}^j(q) - \vec{P}^j = 0 \qquad (11)$$

Motion equations in the vector-based model take the form:

$$\begin{aligned} \textstyle\sum_j \vec{F}^j &= M_{tot}\vec{g} \\ \textstyle\sum_j \vec{P}^j \times \vec{F}^j &= M_{tot}(\vec{x}_{CoM} \times \vec{g}) \end{aligned} \qquad (12)$$

where M_{tot} is the overall body mass and $\vec{x}_{CoM}$ represents the center of mass position as a linear function of decision variables. Note that for static cases, given that desired task positions $\vec{P}^j$ are fixed, equations in (12) become linear in terms of optimization variables. These equations in the joint-space model appear as:

$$h(q) = \tau + \textstyle\sum_j J^j(q)^T \vec{F}^j \qquad (13)$$

where $h(q)$ denotes gravitational forces and $J^j(q)$ represents the Jacobian of each end-effector position. We also have holonomic constraints of the form (7) in the vector-based model while these constraints are inherent in the joint-space model. Regarding inequality constraints, we mainly have joint position bounds and friction polyhedra in both models. The former constraint takes the form of simple bounds on optimization variables in the joint-space model while it requires reconstruction of local frames in the vector-based model. In both models, we define friction polyhedra as:

$$\begin{aligned} (\vec{F}^j)^T \vec{N}_z^j &\geq 0 \\ \mu^j (\vec{F}^j)^T \vec{N}_z^j &\geq |(\vec{F}^j)^T \vec{N}_x^j| \\ \mu^j (\vec{F}^j)^T \vec{N}_z^j &\geq |(\vec{F}^j)^T \vec{N}_y^j| \end{aligned} \qquad (14)$$

where parameters μ^j are friction coefficients of contact surfaces. We also introduce constraints of the form:

$$\vec{x}_2^{jT} \vec{N}_z^j \leq 0 \qquad (15)$$

in both models to make sure that the limbs stay in front of contact surfaces. We also disable contact forces of floating links simply by multiplying them with constant binary mask parameters, determined by task requirements.

4) Multi-stage optimization: Despite some linear constraints and quadratic terms in the objective, our optimization setup remains non-convex due to the holonomic quadratic equality constraints and joint angle boundaries in the vector-based model. To further simplify the analysis, we consider three variants of optimization: (I) full formulation as described, (II) a version without joint boundaries and (III) a version in which we solve the problem in two stages: once without joint boundaries and then using the solution as a warm starter in a second stage where boundaries are added. We further consider enabling and disabling the torques and will demonstrate that even the simple linear objective can do the job most of the time. These strategies will be applied on both joint-space (equations (9), (11), (13), (14) and (15)) and vector-based (equations (8), (10), (12), (14) and (15)) optimizations respectively.

5) Implementation: All optimizations for both models are implemented using SNOPT [14], with a maximum of 250 iterations, feasibility tolerance of 10^{-6} and accuracy level of 10^{-6}. Singularities are handled by a constant damping of the Hessian matrix implemented in this package. We use a dedicated Matlab code to generate vector-based model equations, implemented in c++. Forces and torques are all normalized by the body weight ($M_{tot}g$) and lengths by the body length in the objective and constraints. We avoided including these terms in the equations for the sake of simplicity. Problem sizes are reported in Table.I

Setup =	joint-space	vector-based
variant I	$N = 50, M = 58$	$N = 93, M = 117$
variant II	$N = 50, M = 58$	$N = 69, M = 77$
variant I w.o. torques	$N = 34, M = 42$	$N = 69, M = 93$
variant II w.o. torques	$N = 34, M = 42$	$N = 45, M = 53$

TABLE I: Size of optimization problems in different configurations. Here N stands for the number of variables and M stands for the total number of equality and inequality constraints. The variant III optimizer runs variant II and then variant I optimizers in the first and second stages respectively. Note also that joint limits are implemented as simple bounds on variables in the joint-space model, not counted here. Disabling linear terms or contact forces in the objective does not influence the problem size.

IV. RESULTS

Using the tools and methods explained, we aim at comparing the two modeling techniques over the complex task of posture optimization where computational aspects play an important role. Our analysis covers convergence behavior, i.e. finding local or global solutions, perturbation analysis, and computation time performance.

A. Global convergence

Starting from random initial points, we applied the three optimization variants to various interesting tasks: symmetric and asymmetric quadruped postures, normal standing, single support, picking an object on the ground and stair climbing with supports from the walls. The results are shown in Figure.5 for both modeling techniques. Although the starting points are far from desired, both models can lead to meaningful postures. Considering joint boundaries (variant I) leads to local minima in both models. Removing these constraints in variant II, however, results in a different behavior. The vector-based model converges to the same solution every time while the joint-space model still gets trapped in local minima. We took advantage of this property in variant III and reintroduced joint-boundaries in a second optimization stage which starts from the solution of the boundary-free optimization. As a result, the vector-based model can now find a unique solution while the joint-space model fails. The optimization problems are not convex, and a formal uniqueness proof for the solution is missing. But over a large number of random starting points (10^3 trials), we found the vector-based model convincingly convergent for the popular humanoid tasks considered. We defined uniqueness

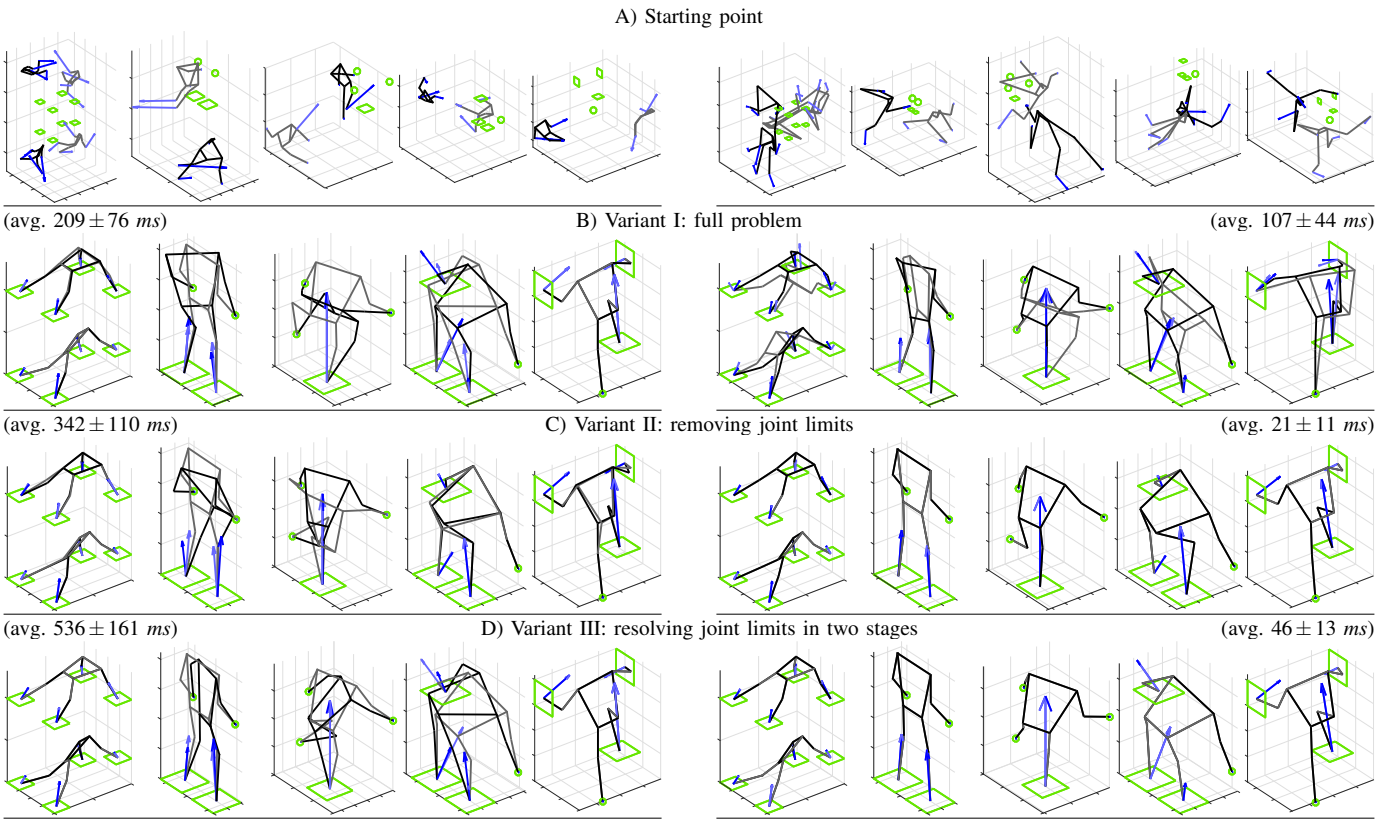

Fig. 5: Global convergence analysis for the joint-space (left) and vector-based (right) models with $Q_\tau = 1$, $Q_{lin} = 1$, $Q_F = 0$ and $\beta = 0.01$ in the objective function. Each row consists of two batches of six typical tasks for humanoid robots: (from left) symmetric and asymmetric quadruped postures, normal standing, single support, picking an object on the ground and stair climbing with supports from the walls. A) a typical random starting point. B) Variant I: considering joint limits in the first run leads to local minima. C) Variant II: removing joint limits still does not solve the problem. D) Variant III: adding joint limits back in the second stage of optimization can not lead to a global optimum too. In these figures, we plot two trials of each optimization to distinguish cases which are globally convergent. This is separately verified over 1000 trials, and the average time spent in the optimizer over all tasks is mentioned on top of batches. The vector-based model performs much faster and is more convergent.

by a threshold (10^{-3}) on the maximum standard deviation of the final solutions obtained (over different dimensions). More accuracy levels can be achieved by tightening the stopping criteria in the optimization.

B. Sensitivity to initial condition

This test could be performed in different ways. Here, we considered a single normal starting point and perturbed it randomly by 10%. Thanks to global convergence, the vector-based formulation behaves in the same way as before while the joint-space formulation again converges to different local minima. Figure.6 demonstrates few examples. Perturbation test can quantify how continuous the model is if used as a basis in a higher level optimization frameworks. Jumping between different local minima would be confusing for higher level planners, and it makes the joint-space formulation less suitable for planning.

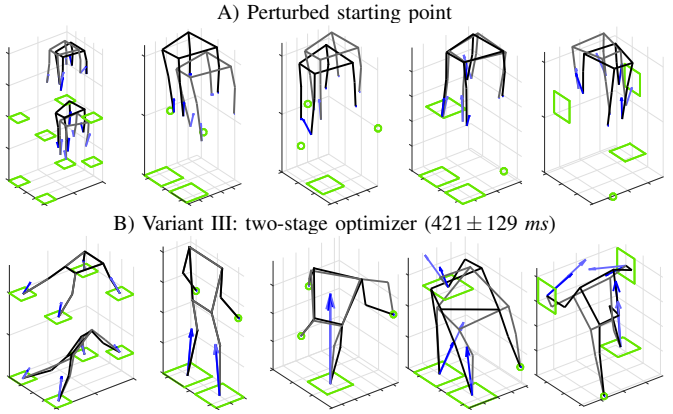

Fig. 6: Sensitivity of the joint-space model to initial conditions. (A) The starting points are only slightly different, (B) but the optimization seems very sensitive, converging to different local minima (two trials per task). Here we used variant III optimization with same parameters as Figure.5. The vector-based model converges to the same solution every time, not interesting to show.

C. Sensitivity to optimization parameters

So far, all optimizations were using both the linear and joint torque terms in the objective function. It becomes interesting to test them separately, since many similar posture optimization methods in literature rely on minimizing joint torques [7], [15]. Enabling contact force terms produce less optimal solutions (Figure.7), since they are in conflict with joint torques. When disabling joint torques however, these terms are necessary to produce optimal contact forces. In this case, the results are less optimal, but still meaningful. Without linear terms however, the vector-based model is not globally convergent anymore

(Figure.7). A variant II optimizer without joint torques and bounds is the fastest and produces convincing postures, useful for robots with large range of joint motions (Figure.7). Remember we chose a small β value to reduce the effect of preferred limb orientation on other tasks which stretch the knees and elbows. In Figure.7, we show how increasing β can flex the knees and elbows. Our choice of $\beta = 0.01$ seems less conservative and thus reasonable. Disabling β however leads to ambiguities (Figure.7) that could be resolved by introducing feet/wrist orientation constraints alternatively.

D. Computation time

In both Figure.5 and Figure.7, we have reported the average timing performance over 10^3 random trials. Adding joint bounds slows down the process in the vector-based model, regardless of joint-torque terms actually. Producing crouched postures seem less expensive while stretched-limb postures seem more costly (Figure.7). It is preferred to keep torque terms for maximum optimality and linear terms to ensure global convergence, even from very different initial conditions. Although the first stage in variant III is globally convergent, we can not still say whether the solution of the second stage is the global optimum of the full problem (variant I). The uniqueness of results is very interesting, however, considering the nonlinear and non-convex properties of multi-contact posture optimization. Our minimal setup (variant II) can reach up to 21 *ms* on average, starting from totally random initial conditions. This could be further improved to 12 *ms* by disabling torque terms and still generating meaningful postures for some tasks. To the best of our knowledge, for such randomly initialized non-linear optimizations, this computation speed was not achieved before. This is due to unrolling complex equations and the convergence properties that our formulation offers.

V. Conclusion

We proposed a fast algorithm that finds unique, feasible and sub-optimal solutions in two stages. The idea of using stages was inspired from the well-known simulated annealing technique [16] where constraints are tightened gradually to better guide the solution towards a feasible point. The result might not be the global minimum of the full problem, but in our case yet meaningful. The vector-based model offers faster speeds than joint-space formulations in our case study, though the core novelty of this paper lies in our molecule-inspired modeling technique that distributes the complexity of the robot to improve the convergence behavior. We break the chain of the robot into multiple segments and link them together with linear constraints. This makes the Jacobian and Hessian matrices simpler in the optimization and speeds up the convergence. The vector-based model has only first and second order polynomial terms while the joint space model has many sine and cosine terms multiplied together. Extensive analysis and comparison demonstrate superior speed and convergence properties in the proposed vector-based modeling technique, despite typically larger problem sizes.

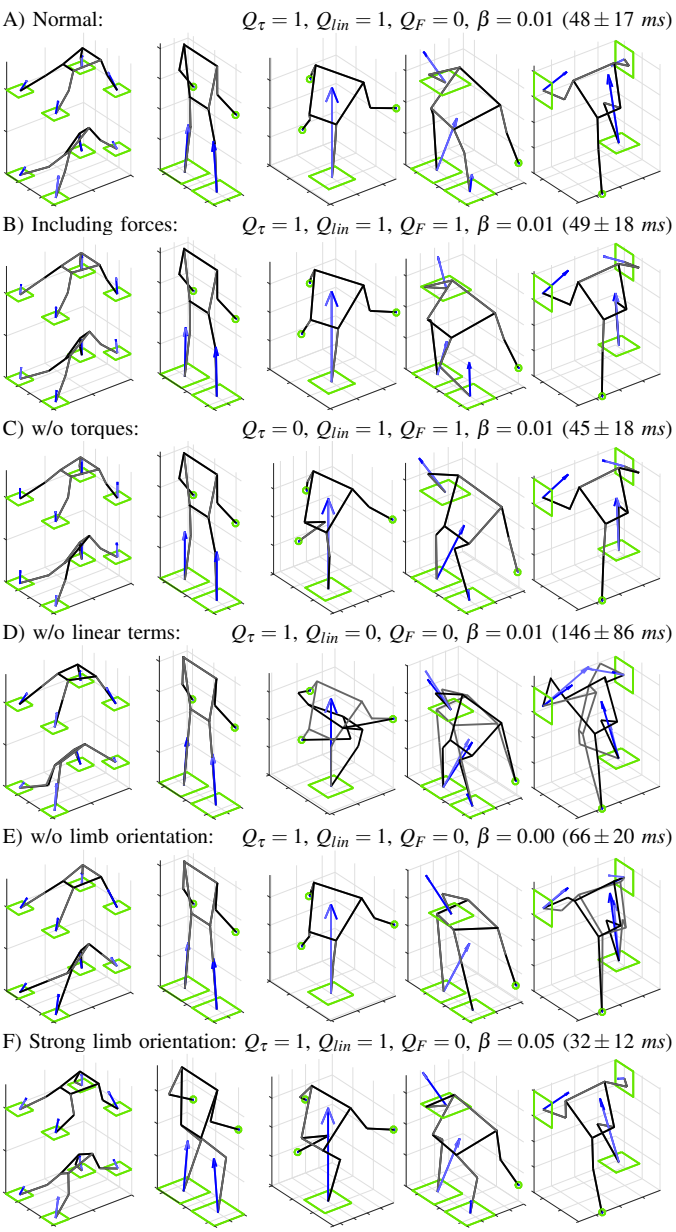

Fig. 7: Sensitivity of vector-based optimization to the choices of objective function parameters. A) Normal choice of parameters similar to Figure.5. B) Including contact forces results in minimal and more uniform contact forces, often increasing joint torques (e.g. object picking task). C) Excluding torque terms leads to slightly crouched postures. Note that contact force terms are needed to avoid arbitrary large forces. Formerly, the torque terms could indirectly optimize contact forces as well. D) Excluding linear terms can still generate optimal postures, but without global convergence properties. E) Disabling β will remove desired limb orientation preference and lead to optimal and singular postures, but it can cause ambiguities in leg/arm orientations (e.g. stair climbing task). F) Too large β values result in crouched postures, but faster convergence.

The proposed modeling technique is fundamentally different from task-space formulations of Khatib [17], although both deal with Cartesian variables. We reconstruct the model via individual vectors that represent body segments while task-space formulations use Jacobians to convert delta-motions from joint-space to task-space and vice versa. The goal of this paper is not to speed up traditional time-integration IK or ID transformations. Those problems are linear with respect to

velocities or accelerations where warm-starting or decomposition techniques [3] can reduce optimization times even below 1 *ms*. Linearizing our vector-based model can indeed lead to such performance as well for online control. In this paper, instead, we directly find final "optimal" postures which involve finding positions, joint torques, and contact forces altogether. This problem is much larger in size and very nonlinear with respect to optimization variables. We also do not have precalculated solutions to speed the process by warm-starting. Despite these limitations and starting from totally random initial conditions, the proposed vector-based approach can find unique final postures (46 *ms*, Figure.5) much faster than time-integration ($1-2$ *s*) [3] or direct nonlinear optimizations in the joint-space ($200-500$ *ms*, Figure.5). Despite recent advances in fast motion planning which typically use abstract variables (CoM, momentum, contacts etc.) [18], we can not provide any performance comparison yet. In this paper, we have not set up any motion planning problem. It remains interesting for future work to use vector-based modeling techniques in motion planning too. Due to better convergence properties, we expect to be able to add more dynamics details without compromising optimization times.

To prioritize different tasks and deal with over-constrained cases, we found soft weighting matrices [19] robust against sensory noises in our torque-based controller using ID formulations [4]. In this work, all end-effector tasks are coded as equality constraints (assuming feasible tasks). It would be interesting to incorporate weighting matrices or hierarchical priorities of time-integration approaches [3], [20] in our nonlinear optimizations. These policies would better handle unsolvable or over-constrained situations. In these cases, it becomes interesting to further analyze optimization times and final errors to make sure the algorithm offers a constant timing like [19] in case of infeasible tasks. Compared to joint-space models, representing the robot by vectors leads to simpler inequality constraints for collision avoidance. Due to a non-convex nature [21], however, these constraints further complicate convergence properties in both joint-space and vector-based optimizations. Since the goal is limited to investigate model properties in this paper, we consider including them in future work.

In future, we consider exploiting our particular problem structure (which consists of few quadratic and many linear terms) to make efficient use of sparsity and further reduce optimization times. Apart from this, we aim at deriving full-body forward and inverse dynamic equations, inspiring from popular techniques in computational chemistry. The proposed vector-based technique can also facilitate modeling of simpler robots like manipulators, bipeds or monopods.

VI. ACKNOWLEDGMENTS

This work was funded by the WALK-MAN project (European Community's 7th Framework Programme: FP7-ICT 611832). The authors would like to thank Mr. Fazel Parsapour for his insightful discussions on the chemistry-related part of this paper.

REFERENCES

[1] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, 2015.

[2] S. Faraji, S. Pouya, and A. Ijspeert, "Robust and agile 3d biped walking with steering capability using a footstep predictive approach," in *Robotics Science and Systems (RSS)*, 2014, pp. 1–9.

[3] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, p. 0278364914521306, 2014.

[4] S. Faraji, L. Colasanto, and A. J. Ijspeert, "Practical considerations in using inverse dynamics on a humanoid robot: torque tracking, sensor fusion and cartesian control laws," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2015, pp. 1619–1626.

[5] S. Faraji and A. J. Ijspeert, "Singularity-tolerant inverse kinematics for bipedal robots: An efficient use of computational power to reduce energy consumption," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1132–1139, 2017.

[6] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "Opensim: open-source software to create and analyze dynamic simulations of movement," *IEEE transactions on biomedical engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.

[7] B. Howard, J. Yang, and B. Ozsoy, "Optimal posture and supporting hand force prediction for common automotive assembly one-handed tasks," *Journal of Mechanisms and Robotics*, vol. 6, no. 2, p. 021009, 2014.

[8] S. H. Chanteau and J. M. Tour, "Synthesis of anthropomorphic molecules: the nanoputians," *The Journal of organic chemistry*, vol. 68, no. 23, pp. 8750–8766, 2003.

[9] W. J. Hehre, *A guide to molecular mechanics and quantum chemical calculations*. Wavefunction Irvine, CA, 2003, vol. 2.

[10] J. Baker, "Techniques for geometry optimization: A comparison of cartesian and natural internal coordinates," *Journal of computational chemistry*, vol. 14, no. 9, pp. 1085–1100, 1993.

[11] M. P. Allen *et al.*, "Introduction to molecular dynamics simulation," *Computational soft matter: from synthetic polymers to proteins*, vol. 23, pp. 1–28, 2004.

[12] H. B. Schlegel, "Geometry optimization," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 1, no. 5, pp. 790–809, 2011.

[13] M. Doyle, "A barrier algorithm for large nonlinear optimization problems," Ph.D. dissertation, stanford university, 2003.

[14] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[15] T. Marler, L. Knake, and R. Johnson, "Optimization-based posture prediction for analysis of box lifting tasks," in *International Conference on Digital Human Modeling*. Springer, 2011, pp. 151–160.

[16] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated Annealing: Theory and Applications*. Springer, 1987, pp. 7–15.

[17] O. Khatib, "A unified approach to motion and force control of robot manipulators: The operational space formulation," *The International Journal of Robotics Research (IJRR)*, vol. 3, no. 1, pp. 43–53, 1987.

[18] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 842–849.

[19] T. Sugihara, "Solvability-unconcerned inverse kinematics by the levenberg–marquardt method," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 984–991, 2011.

[20] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2014, pp. 981–988.

[21] B. Alrifaee, M. G. Mamaghani, and D. Abel, "Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles," in *Intelligent Control (ISIC), IEEE International Symposium on*, 2014, pp. 1583–1588.