2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Extending Riemmanian Motion Policies to a Class of Underactuated Wheeled-Inverted-Pendulum Robots

Bruce Wingo[1], Ching-An Cheng[1], Muhammad Murtaza[1], Munzir Zafar[1], and Seth Hutchinson[1]

***Abstract*— Riemannian Motion Policies (RMPs) have recently been introduced as a way to specify second-order motion policies defined on robot task spaces. RMP-based approaches have the advantage of being more general than traditional approaches based on operational space control; for example, the generalized task inertia in an RMP can be fully state-dependent, which is particularly effective in designing collision avoidance bahaviors. But until now RMPs have been applied only to fully actuated systems, i.e. systems for which each degree of freedom (DoF) can be directly actuated by a control input. In this paper, we present a method that extends the RMP formalism to a class of underacutated systems whose dynamics are amenable to a decomposition into a fully-actuated subsystem and a residual dynamics. We show the efficacy of the approach by constructing a suitable decomposition for a Wheeled-Inverted-Pendulum (WIP) humanoid robot and applying our method to derive motion policies for combined locomotion and manipulation tasks. Simulation results are presented for a 7-DoF system with one degree of underactuation.**

I. INTRODUCTION

Recently a new family of robots, with both dynamic locomotion capability and high degrees-of-freedom (DoF) manipulator arms, are growing in prominence [1], [2], [3]. These robots combine progresses made in manipulation with serial robots [4], [5] and locomotion with inherently unstable dynamics (such as biped and Wheel-Inverted-Pendulum (WIP) robots) [6], [7], [8], offering a promising hybrid platform to achieve multi-objective task specifications.

In this paper, we present a motion planning and control framework for a class of underactuated WIP humanoids, leveraging the task prioritization of Whole-Body-Control (WBC) [9] and the flexibility of Riemannian Motion Policies (RMPs) [10], [11]. These two techniques have been developed separately for locomotion and manipulation tasks in the past. Here we show how to combine these two ideas to complement each other, so that multi-objective manipulation tasks can be successfully performed on WIP humanoids while maintaining dynamic stability for locomotion.

In the WBC framework, different objectives are framed in their respective task spaces and combined using null space projections according to a task-prioty hierarchy. A task with high priority is satisfied by implementing other tasks with lower priority within its null space. Robot dynamics and other physical constraints, such as contacts, can be incorporated by enforcing them as the top-level task. However, WBC suffers from algorithmic singularities when a large number of tasks are present, as successive null space projections is often ill-conditioned [12], [13].

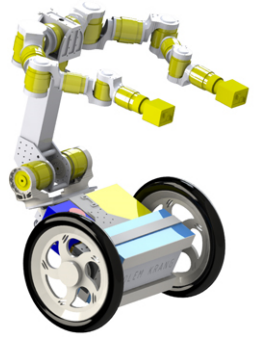

Fig. 1. Golem-KRANG.

RMPs [10] and the associated computational framework RMPflow [11] provide an alternative for multi-objective task planning and policy fusion. Individual tasks in RMPflow are described in their respective task spaces similar to WBC, but modeled in terms of RMPs (second-order differential equations and the associated Riemmanian metrics).[1] For each task RMP, the second-order differential equation defines the motion policy for the task, and the task space is treated as a manifold with an (non-)Euclidean metric capturing properties of the aforementioned motion policy. Intuitively one can view RMPflow as an extension of operational space control [4] to using full-state dependent inertia matrices, which encompass a richer class of task behaviors that depends on both a robot's configuration and velocity.

RMPflow [11] provides a recursive scheme for combining RMPs associated with different tasks into a single control policy for the entire robot. This is done by using a differential geometric operation, **pullback**, which brings RMPs from task manifolds to the configuration space (another manifold). Issues with the null space projections in WBC are no longer present in this approach, even when the number of tasks is large. When task policies are generated from a special subset of second-order differential equations, called Geometric Dynamic Systems (GDSs)[2], the combined policy after **pullback** is proved to be Lyapunov-stable [11], [13].

While RMPs and RMPflow have been successfully implemented on serial manipulators and multi-robot systems [11], [13], it is unclear how they can be applied to robots with

[1]Bruce Wingo, Ching-An Cheng, Muhammad Murtaza, Munzir Zafar and Seth Hutchinson are with the Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA, 30332, USA. bwingo@gatech.edu, cacheng@gatech.edu, mamurtaza@gatech.edu, mzafar7@gatech.edu, seth@gatech.edu

[1]Technically, an RMP is defined by a second-order differential equation and its interia, the latter of which in most cases is the same as the Riemmanian metric; sec [11] and Section II-A.

[2]A non-Euclidean generalization of spring-mass-damper systems.

underactuated dynamics or non-holonomic constraints. Current limitations of this framework stem from the implicit assumption that every DoF of the robot can be directly controlled [11]. However, this is hardly ever the case outside of the serial manipulator arms. RMPs and RMPflow have yet to be implemented on underactuated systems, like the robot used in this work, KRANG [12] shown in Fig. 1 (which has a WIP base with two serial manipulator arms attached on top, and one degree of underactuation).

In this work, we propose a variant of RMPflow capable of executing RMPs on a class of WIP humanoids while maintaining dynamic stability. Our design is based on a hierarchical control and dynamics separation scheme. The main idea is to split the dynamics into two parts: the dynamics of purely actuated DoF and the residual dynamics that describes the relationship between actuated DoF and unactuated DoF. A high-level controller based on the residual dynamics is first designed to guarantee the stability during locomotion by generating a Center-of-Mass (CoM) trajectory using planning. Then a low-level RMP-based controller for the fully actuated subsystem is used to track the desired CoM trajectory as the top-priority task. Other remaining manipulation tasks are specified as RMPs and combined using the **pullback** operation of RMPflow into a single policy on the configuration space. This pullback policy is finally realized in the null space of the top-priority CoM trajectory tracking task. Overall, our scheme guarantees the dynamic stability of the locomotion by using the task prioritization idea in WBC, and uses RMPflow with the remaining DoF in an attempt to achieve multiple manipulation tasks without the algorithmic singularity due to successive projections in WBC. In simulations, we show the efficacy of our approach by designing motion policies for KRANG in Fig. 1 to solve combined locomotion and manipulation tasks.

II. BACKGROUND

We review the essence of RMPs and RMPflow, as they will be used to define and combine the task-space policies in our framework. Further details can be found in [11], [13]

A. RMPs and RMPflow

Riemannian Motion Policies (RMPs) were first proposed by Ratliff et al. in [10] as a language to describe motion policies defined on general task spaces. Its main idea is to treat the task space as a manifold and model motion as geodesics (paths with the shortest distance). Using this correspondence, one can naturally think of motion policies as generators of these geodesics, and then controls the desired behavior through designing the manifold's Riemannian metric; e.g., for a properly selected metric, geodesics can be used to realize curved trajectories. As such, these motion policies are suitably named *Riemannian* Motion Policies [10].

Specifically, for a manifold $\mathcal{M}$ with a coordinate map x, a motion policy a is defined as the controller of the acceleration $\ddot{x} = a(x,\dot{x})$, which takes the state $(x,\dot{x})$ as input and outputs the desired acceleration $a(x,\dot{x})$. An RMP $(a,M)^{\mathcal{M}}$ pairs a motion policy a with an importance weight M, which is a symmetric positive-definite matrix function. For most applications[3], the matrix function M is the coordinate representation of the Riemannian metric of $\mathcal{M}$. In [11], the expression $(a,M)^{\mathcal{M}}$ is referred to as the canonical form of the RMP, and the associated natural form is written as $[f,M]^{\mathcal{M}}$, where $f \coloneqq Ma$ is the (virtual) control force.

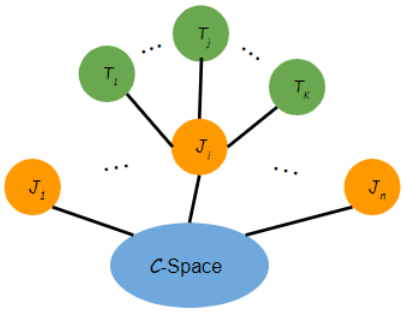

Fig. 2. A sample RMP tree for an n-DoF robot.

Representing motion policies as differential geometric objects allows one to transform them between manifolds naturally. For example, consider the configuration space of a robot, $\mathcal{C}$. Suppose that $\mathcal{C}$ is an n-dimensional smooth differentiable manifold and has a global chart $(\mathcal{C},q)$, where $q : \mathcal{C} \to \mathbb{R}^n$. Then motion policies defined on task manifolds can then be mapped to motion policies on $\mathcal{C}$ through standard differential geometric tools, such as pullback, as they just correspond to curves on manifolds.

RMPflow [11] is a computational framework that formalizes this concept to consistently combine task-space RMPs into a single motion policy on $\mathcal{C}$. To accomplish this, RMPflow deploys a tree-structure computational graph, RMP-tree. A typical RMP-tree for an n-joint robot is visualized in Fig. 2. In an RMP-tree, each node represents an RMP and its state on a manifold, and each edge corresponds to operations that transforms RMPs and states between nodes. In particular, the root corresponds to the configuration space, and the leaf nodes correspond to the task spaces. These edge operations are termed RMP-algebra, and consist of three transformations: **pushforward**, **pullback**, and **resolve**.

B. RMP-algebra

Let us use the sample RMP-tree in Fig. 2 to show how these operations propagate information across the RMP-tree and generate a policy on the configuration space $\mathcal{C}$ that combines the RMPs defined on the task spaces. This is done in three steps for every sampled time instance:

1) calls of **pushforward** from the root to the leaves
2) calls of **pullback** from the leaves to the root
3) a single call **resolve** at the root

For illustration, consider the node $\mathcal{J}_i$ in Fig. 2. Suppose $\mathcal{J}_i$ describes an RMP $[f,M]^{\mathcal{M}}$ and state $(x,\dot{x})$. In Fig. 2, $\mathcal{J}_i$ has K child nodes $\mathcal{T}_1,\ldots,\mathcal{T}_K$, and its parent node is the configuration space $\mathcal{C}$. For each child node $\mathcal{T}_j$, we suppose it corresponds to an RMP $[f_j,M_j]^{\mathcal{T}_j}$ and its state $(y_j,\dot{y}_j)$.

[3][11] shows when the metric starts to depend on velocity, M is equal to the metric plus a curvature modification (cf. Section II-C).

pushforward is the operation that updates state information in child nodes using their parent node's state. Suppose these two coordinates of $\mathcal{J}_i$ and $\mathcal{T}_j$ are related through a smooth map ψ as $y_j = \psi_j(x)$. Then given $(x,\dot{x})$, **pushforward** updates the child node with $(y_j,\dot{y}_j) = (\psi_j(x), J_j(x)\dot{x})$, where $J_j(x)$ is the Jacobian of $\psi_j(x)$, which is a linear operator that maps tangent vectors form $\mathcal{J}_i$ to $\mathcal{T}_j$.

On the other hand, **pullback** is an operation that back propagates RMPs from child nodes to their parent node. For $\mathcal{J}_i$, its RMP $[f,M]^{\mathcal{M}}$ is updated as follows:

$$M = \sum_{j=1}^{K} J_j^\top M_j J_j \qquad \textit{and} \qquad f = \sum_{j=1}^{K} J_j^\top (f_j - M_j \dot{J}_j \dot{x}) \quad (1)$$

Therefore, in the policy generation process described above, the recursive calls of **pullback** at the end will give an RMP at the root of the configuration space $\mathcal{C}$ as $[f_q, M_q]^{\mathcal{C}}$.

The final component of RMP-algebra, **resolve**, transforms this RMP from its natural form to the canonical form to generate the motion policy for robot control. Given the combined policy $[f_q, M_q]^{\mathcal{C}}$, it computes $(\ddot{q}_q, M_q)^{\mathcal{C}}$, where $\ddot{q}_q = M_q^{-1} f_q$. This vector $\ddot{q}_q$ represents the desired acceleration that can consistently trades off different task-space RMPs with respect to their metric information (encoded as the abstract inertia). In the following, we will also use $\ddot{q}_{RMP} = \ddot{q}_q$ as an alias to emphasize the desired acceleration on $\mathcal{C}$ computed by RMPflow.

C. Geometric-Dynamical-Systems (GDSs)

Geometric Dynamical Systems (GDSs) is a family of second-order dynamics commonly used to specify RMPs on task manifolds. A GDS on an m-dimensional manifold $\mathcal{M}$ with a global chart $(\mathcal{M}, x)$ is defined by a symmetric positive-definite *metric* $G(x,\dot{x}) \in \mathbb{R}_+^{m\times m}$, a positive-definite damping matrix $B(x,\dot{x}) \in \mathbb{R}_+^{m\times m}$, and a lower-bounded potential energy function $\Phi(x) \in \mathbb{R}$:

$$(G(x,\dot{x}) + \Xi_G(x,\dot{x}))\ddot{x} + \xi_G(x,\dot{x}) = -\nabla_x \Phi(x) - B(x,\dot{x})\dot{x} \quad (2)$$

where $\Xi_G(x,\dot{x})$, and $\xi_G(x,\dot{x})$ are curvature terms defined as

$$\begin{aligned} \Xi_G(x,\dot{x}) &= \frac{1}{2}\sum_{i=1}^{m} \dot{x}_i \partial_{\dot{x}} g_i(x,\dot{x}) \\ \xi_G(x,\dot{x}) &= \overset{x}{G}(x,\dot{x})\dot{x} - \frac{1}{2}\nabla_x(\dot{x}^\top G(x,\dot{x})\dot{x}) \end{aligned} \quad (3)$$

with x_i being the i^{th} element of x, $g_i(x,\dot{x})$ being the i^{th} column of $G(x,\dot{x})$, and $\overset{x}{G}(x,\dot{x}) = [\partial_x g_i(x,\dot{x})]_{i=1}^m$ being a matrix constructed by horizontal concatenation of column vectors $\partial_x g_i(x,\dot{x})$. Borrowing analogy from dynamics, one can define a virtual *inertia* matrix $M(x,\dot{x}) := G(x,\dot{x}) + \Xi_G(x,\dot{x})$. This *inertia* matrix corresponds to weight matrix in the RMP formulation of the GDS (i.e. $(a,M)^{\mathcal{M}}$ with a specified by (2) and M given by G as described above).

It can be shown that RMPflow is stable in the sense of Lyapunov, when all leaf-node RMPs are specified in the form of GDSs [11]. In our experiments, the end-effector position and orientation attractor RMPs were implemented as GDSs. We will discuss the details in section V.

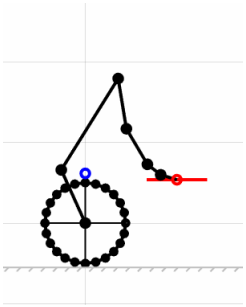

Fig. 3. A WIP humanoid with a 5-DoF arm mounted on a 2-DoF torso with 1-DoF locomotion which is underactuated. The blue circle denotes the CoM location of the robot and the red horizontal line at the end-effector indicates the orientation of the end-effector.

III. OVERCOMING UNDER-ACTUATION THROUGH DYNAMICS SEPARATION

RMPflow provides a systematic framework for designing control policies that can achieve multiple task objectives. However, one of its fundamental limitations is the assumption of *full* actuation. This can be seen from the example described in Section II-B, which requires the robot to realize the final acceleration $\ddot{q}_{RMP}$ in the configuration space returned by the last **resolve** operation call. While this is feasible, e.g., for robot manipulators, it is impossible for underactuated systems (i.e. systems with number of actuators less than the dimension of the configuration space), including the WIP humanoids of interest here,

In this work, we overcome this challenge by identifying a fully actuated subsystem based on the dynamics properties of WIP humanoids. We show that, under mild assumptions, these robots, despite being underactuated, can still be controlled by the RMPflow framework through this subsystem. This insight allows us to directly work with this recent advancement in policy fusion to design complex behaviors for WIP humanoids.

A. Dynamics

For a general WIP humanoid with n joints and d-DoF locomotion shown in Fig. 3, its dynamics can be written as

$$A \begin{bmatrix} \ddot{x} \\ \ddot{q} \end{bmatrix} + h = B\Gamma \quad (4)$$

where $x \in \mathbb{R}^d$ and $q \in \mathbb{R}^n$ are the coordinates of the center of mass (COM) and the joints, respectively, $\Gamma = [\tau_1, \ldots, \tau_n]^\top \in \mathbb{R}^n$ denotes the actuator torques, $A \in \mathbb{R}^{(n+d)\times(n+d)}$ is the physical inertia matrix (which is symmetric positive definite), $h \in \mathbb{R}^{(n+d)\times 1}$ describes the combined Coriolis and gravity effects, and $B \in \mathbb{R}^{(n+d)\times n}$ is the actuation matrix. The system overall has $n+d$ DoF, but the number of control inputs is only n. Therefore the system is underactuated.

In this paper, we concern especially the set of WIP robots with an actuated base joint sharing the same control torque as the wheel base. For these robots, When the platform of the robot is attached to the ground (i.e. the locomotion coordinate x simplifies into the horizontal component that is parallel to the ground), the above equation (4) admits certain structure. For illustration purpose, we write the planar dynamics of

this class of robots as below (more details can be found in [12]). In this case, we have $d = 1$ and $x \in \mathbb{R}$, which is the horizon coordinate (the only DoF left in locomotion after the simplification to the sagittal plane and the non-floating assumption). Then the dynamics in (4) can be written as

$$\begin{bmatrix} A_{xx} & A_{xq} \\ A_{qx} & A_{qq} \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{q} \end{bmatrix} + \begin{bmatrix} h_x \\ h_q \end{bmatrix} = \begin{bmatrix} \frac{-1}{R} e_1^\top \\ I \end{bmatrix} \Gamma \tag{5}$$

where $I \in \mathbb{R}^{n \times n}$ denotes the identity matrix, $e_1 \in \mathbb{R}^{n \times 1}$ denotes the canonical vector (which is 1 in the first coordinate and zero elsewhere), and $R \in \mathbb{R}_+$ is the wheel radius. We recall that, because A is the physical inertia matrix, it satisfies $A_{xx} > 0$, $A_{qq} \succ 0$, and $A_{qx} = A_{xq}^\top$.

The main structure introduced in (5) compared with (4) is the specific form of B, which shows one degree of under-actuation. Again, this is because the horizontal movement is dictated solely by the motors installed on the wheels (here τ_1 results from the net torque of the left and right wheels). A similar structure shows up in the full 3D dynamics under the non-floating assumption [12].

B. Identification of Fully Actuated Subsystem

We leverage the structure in (5) to identify a fully actuated subsystem, on which we can implement RMPflow. The main idea is based on a simple observation. Suppose we multiply (5) on the left with a non-singular matrix

$$P = \begin{bmatrix} I & 0 \\ -A_{qx}A_{xx}^{-1} & I \end{bmatrix} \tag{6}$$

We can write (5) equivalently as

$$\begin{bmatrix} A_{xx} & A_{xq} \\ 0 & A_{qq} - A_{qx}A_{xx}^{-1}A_{xq} \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{q} \end{bmatrix} + \begin{bmatrix} h_x \\ h_{q|x} \end{bmatrix} = \begin{bmatrix} \frac{-1}{R} e_1^\top \\ I + \frac{1}{R} A_{qx}A_{xx}^{-1} e_1^\top \end{bmatrix} \Gamma$$

where $h_{q|x} = h_q - A_{qx}A_{xx}^{-1}h_x$. This step is similar to Gauss elimination. The second row above describes how the torques in Γ influence the joint acceleration $\ddot{q}$

$$A_{q|x}\ddot{q} + h_{q|x} = \mathscr{B}\Gamma \tag{7}$$

where $A_{q|x} = A_{qq} - A_{qx}A_{xx}^{-1}A_{xq}$ and $\mathscr{B} = I + \frac{1}{R}A_{qx}A_{xx}^{-1}e_1^\top$. Notice that because $R > 0$, $\mathscr{B}$ is full rank; also because $A_{q|x}$ is the Schur compliment of A_{xx} in the positive definite matrix A, $A_{q|x}$ is positive definite (which can be viewed as a projected inertia) [14]. In other words, (7) shows that the subsystem of $\ddot{q}$ in WIP humanoids under the non-floating assumption is actually fully actuated; we can treat this subsystem just like the dynamics of any other fully actuated robot dynamics such as the usual robot manipulator.

We remark that the above trick works mainly because of structure of the actuation matrix B given in (5). Our approach can be viewed as a simple abstraction of the work by Zafar et al. in [12], where the dynamics separation of the KRANG robot are studied but with much involved algebraic manipulation.

Thus, to control WIP humanoids, we can view (7) as the fully-actuated dynamics required in RMPflow, and then view the locomotion coordinate x as one of the task spaces, which can be treated in the same way as other task spaces. With this change of perspective, we can, for example, design a multi-task controller for WIP humanoids by first specifying RMPs on the task spaces (including the desired locomotion behavior) and pulling back these task RMPs to obtain the desired acceleration for $\ddot{q}$ (as we showed in Section II-B), from which we can then compute the necessary torque by (7). In the next section, we detail this procedure. For convenience, we will write (7) in a different way as

$$\mathscr{A}\ddot{q} + \mathfrak{h} = \Gamma \tag{8}$$

where $\mathscr{A} = \mathscr{B}^{-1}A_{q|x}$ and $\mathfrak{h} = \mathscr{B}^{-1}h_{q|x}$.

IV. WHOLE-BODY-CONTROL SYSTEM

We adopt the dynamics separation principle above to design our WBC system. In short, our system is based on a combination of null-space control and RMPflow, where the high-level control aims to maintain the balance through properly planing the CoM trajectory of the robot. The low-level control tracks this CoM trajectory as the top priority task and realizes the remaining multiple objectives of the robot through RMPflow in the null space.

The merit of this system design is two-fold. First, it ensures the robot always stays upward; if we were to use purely RMPflow, this could be difficult to achieve unless we carefully design the importance weights (i.e. the abstract inertia) in RMPs, because RMPflow in essence is performing a soft fusion. Second, our scheme has only one level of hierarchy, so it is less sensitive to algorithmic singularity [14] which could easily happen if we were to implement all the tasks of the robot following the classic hierarchical control approach [15].

A. High-Level Controller

We use planning to design a CoM trajectory so that the robot can track this desired motion trajectory of CoM while maintaining an upright posture. In implementation, our system replan the CoM trajectory given new information of the robot's state.

We realize this planning idea by Differential Dynamic Programing (DDP) with a simplified dynamics model, which is set as the inverted pendulum approximation of the full system (i.e. motion of the upper joints is ignored in planning). Using a low-dimensional approximate model yields a simpler planning problem than that of the full system, so the computation becomes more efficient (DDP scales cubically with the dimension of the state space), which is especially critical to online replanning.

We recall that an inverted pendulum is an under-actuated system, and the system is balanced if the CoM deviates from the vertical line with an angle θ, within some small threshold. It is critical that for KRANG, and the class of WIP robots it represents, θ is completely determined by all the actuated DoF, i.e. the q. In this high-level planing problem, the goal is to track a desired horizontal motion trajectory $X^{des} = [x^{des}, \dot{x}^{des}]^\top$ by specifying $\ddot{\theta}$ as the control input. The output of the high-level planner is an acceleration policy for the CoM angle θ, which we denote as $\ddot{\theta}_{COM}$.

Once $\ddot{\theta}_{COM}$ is computed through DDP, we can find the necessary torque Γ to realize $\ddot{\theta}_{COM}$. Specifically, we use the the Jacobian matrix J_θ, a function of only q, to relates joint velocities $\dot{q}$ and the angular velocity $\dot{\theta}$. We emphasize again that J_θ exists because θ is solely a function of q.

With J_θ, we can realize $\ddot{\theta}_{COM}$ in the standard way as follows. Let us first choose a torque, say $\Gamma_{\theta_{COM}}$, to generate the acceleration $\ddot{\theta}_{COM}$. This can be achieved by inspecting the relationship between torques and $\ddot{\theta}$,

$$(\ddot{\theta} - \dot{J}_\theta \dot{q}) + J_\theta \mathscr{A}^{-1} \mathfrak{h} = J_\theta \mathscr{A}^{-1} \Gamma_{\theta_{COM}} \tag{9}$$

which can be obtained by the basic relationship $\ddot{\theta} = J_\theta \ddot{q} + \dot{J}_\theta \dot{q}$ and (8). We want to find $\Gamma_{\theta_{COM}}$ such that $\ddot{\theta} = \ddot{\theta}_{COM}$. One can easily verify that a particular solution is given as

$$\Gamma_{\theta_{COM}} = J_\theta^\top (\Lambda_{\theta_{COM}} (\ddot{\theta}_{COM} - \dot{J}_\theta \dot{q}) + \bar{J}_\theta h) \tag{10}$$

where $\Lambda_\theta = (J_\theta \mathscr{A}^{-1} J_\theta^\top)^{-1}$ and $\bar{J}_\theta = (\Lambda_\theta J_\theta \mathscr{A}^{-1})^\top$ is a pseudo-inverse of J_θ.

Setting $\Gamma = \Gamma_{\theta_{COM}}$ in (10) and applying it to the system in (8) would render $\ddot{\theta}_{COM}$. But since we also want to achieve the other objectives of the robot, we do not just adopt $\Gamma = \Gamma_{\theta_{COM}}$ but consider instead adding a null space component, let

$$\Gamma = \Gamma_{\theta_{COM}} + N_\theta^\top \Gamma_0 \tag{11}$$

where Γ_0 in a vector that will be chosen in the next section to realize the remaining task objectives, and

$$N_\theta = I - \bar{J}_\theta J_\theta \tag{12}$$

is a null-space projection matrix. One can verify that $J_\theta N_\theta = 0$ and applying the command in (11) will still gives $\ddot{\theta} = \ddot{\theta}_{COM}$. In particular, we note that since we are going to set Γ_0 properly to realize other tasks, the choice of particular solution $\Gamma_{\theta_{COM}}$ in (10) becomes of no importance. In other words, one can also choose other solutions that realizes $\ddot{\theta}_{COM}$ to replace $\Gamma_{\theta_{COM}}$ in (11), and the final torque command Γ in (11) will stay the same (as Γ_0 will be modified accordingly).

B. Low-Level Controller

We now describe how to choose to the null space command Γ_0 for the other objectives. This is done in two steps: First, we compute the desired acceleration $\ddot{q}_{RMP}$, which is done by the standard RMPflow routine by pulling back task RMPs through using RMP-algebra. Second, we set Γ_0 so that $\ddot{q}_{RMP}$ can be realized as close as possible. Note that we may not be able to realize $\ddot{q}_{RMP}$ exactly, because after the high-level controller is imposed, the effective dynamical system changes from (8) into

$$\mathscr{A} \ddot{q} + \mathfrak{h} = \Gamma_{\theta_{COM}} + N_\theta^\top \Gamma_0 \tag{13}$$

and $N_\theta^\top$ by definition is not full-rank (it is a projection). Thus, $\ddot{q}_{RMP}$ must be approximated. But in which sense?

To this end, we leverage the *metric* information given in RMPflow. Recall, when $\ddot{q}_{RMP}$ is computed by **resolve**, a virtual inertia matrix M_q defined on the joint space is also computed. This matrix M_q, as shown in [11], acts as a directional importance weight that trades off different motion policies collected by **pullback**. Borrowing this idea, we use M_q as an importance weight to select Γ_0. Namely, we set Γ_0 as a solution to the weighted least-squares problem,

$$\begin{aligned} \min_{\Gamma'} \quad & \|\ddot{q} - \ddot{q}_{RMP}\|^2_{M_\mathbf{q}} \\ s.t. \quad & \ddot{q} = \mathscr{A}^{-1}(\Gamma_{\theta_{COM}} - \mathfrak{h} + N_\theta^\top \Gamma') \end{aligned} \tag{14}$$

where $\Gamma_{\theta_{COM}}$ is computed from (10). We note that, because N_θ is singular, the solution to this problem is not unique. Nonetheless the value $N_\theta^\top \Gamma_0$ is uniquely determined. In practice, we can choose Γ_0 as the minimal norm solution to the above problem, which can be computed in closed-form in terms of Moore-Penrose pseudo-inverse. Because of the parameterization in (11), the final policy will always achieve the desired CoM balancing and horizontal motion behavior, while the role of Γ_0 is to realize other control tasks using the remaining DoF.

V. IMPLEMENTATION AND RESULTS

We verify the proposed control framework in Section IV in MATLAB simulation on a simplified[4] 2-D KRANG model (Fig. 3) that has a 2-DoF torso mounted on a rigid wheel and a 5-DoF serial manipulator arm. This system has one degree of locomotion, which is underactuated as described in Section III-A. The high-level controller generates the CoM angle trajectory using the MPC-DDP optimizer implemented in [16] for KRANG. The low-level controller is generated by RMPflow as discussed in Section II-C. Two task RMPs based on GDSs are implemented to maintain desired end-effector position and orientation. Details on the attractor implementation can be found in Appendix-A.

The simulation task is for the robot to move forward 5 meters while maintaining the end-effector position (with respect to the body frame), and orientation in the global frame (holding the end-effector at 90 degress measured from the horizontal axis). A sequence of frozen frames of the 2-D simulation is shown in Fig. 4. As a verification of controller stability, the robot's CoM position and orientation along with the CoM linear velocity and angular velocity are also plotted in Fig. 5. The convergence behavior shown in Fig. 5 indicates that the overall controller is indeed stable. However, from Fig. 4, it is clear that there are some overshoot present in end-effector position tracking. This is because the parameters for the attractor GDSs were only roughly tuned by hand. Further fine tuning can improve the task performance.

VI. CONCLUSION

In this work, we extend the range of applicability of RMPflow to the set of WIP-humanoids whose base joints are actuated by the induced wheel torque. This unique feature allows us to perform dynamics separation to identify a fully actuated body subsystem dynamics. A hierarchical controller was then developed for this subsystem, capable of both balancing the CoM during locomotion and tracking joint acceleration policies generated by RMPflow. To apply our

[4]The full 2-D KRANG model has a pair of 7-DoF arms mounted on a 3-DoF torso, and 1 degrees of underactuation.

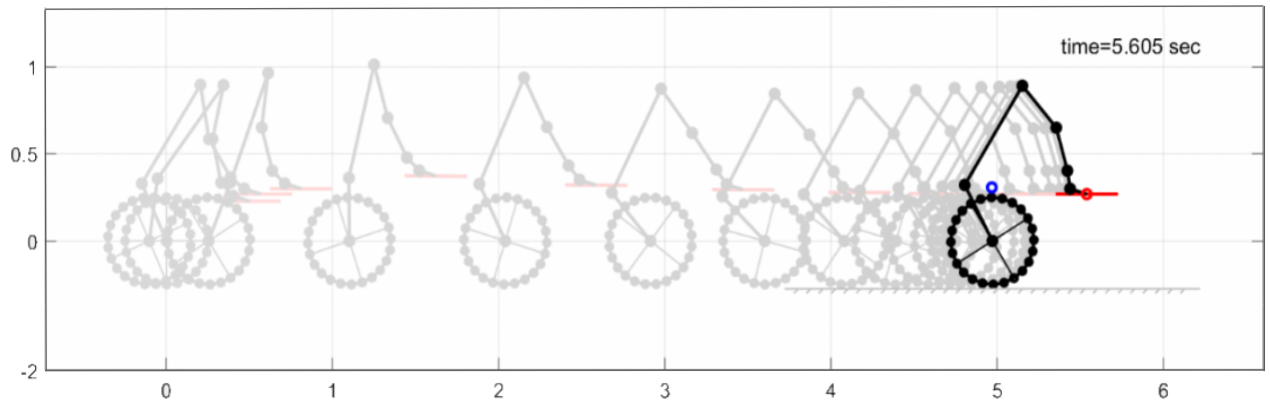

Fig. 4. Snap shots of the 2-D simulation, axis unit in meters.

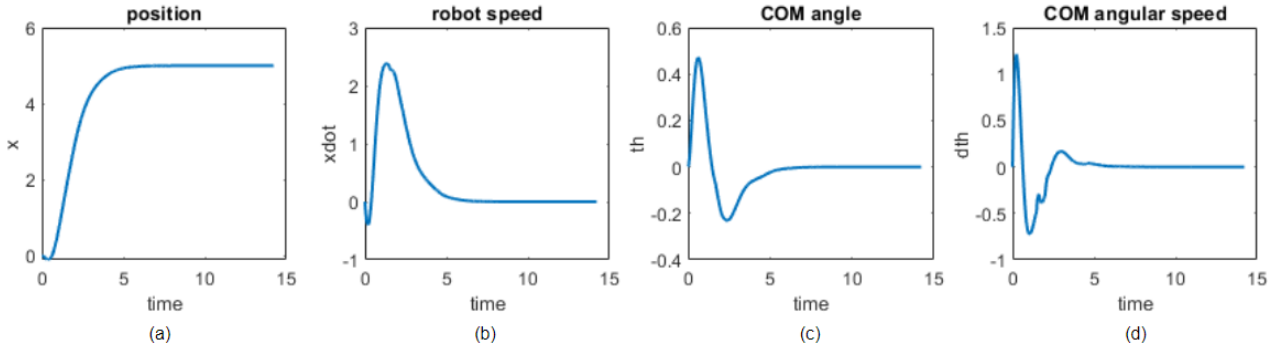

Fig. 5. (a) CoM horizontal position plot, time in seconds [sec] and x in meters [m], (b) CoM linear velocity plot, time in [sec] and xdot in [m/sec], (c) CoM angle with respect to vertical axis, time in [sec] and th in radiance [rad], (d) CoM angular velocity, time in [sec] and dth in [rad/sec].

proposed framework on another robot, it is essential to perform dynamics separation, as described in section III, which may or may not be possible depending on the specific dynamics of that robot.

APPENDIX

A. *End-effector position and orientation attractor RMPs*

Attractor RMP for end-effector position control is straightforward. Define the task-space coordinate $x = x_{current} - x_{target}$, where $x_{current}$ and x_{target} are the current and the target end-effector position, respectively. We chose attractor RMP candidates in the form

$$\ddot{x}^{des} = -\nabla_x \tilde{\Phi}(x) - \tilde{B}(x,\dot{x})\dot{x} - M(x)^{-1}\xi_M(x,\dot{x}) \qquad (15)$$

[11] shows that with an appropriate choice of M, the attractor RMPs of this form can be written as GDSs (2).

In our experiments, an η-scaled softmax over $\|x\|$ and $-\|x\|$ is chosen as the potential energy function. $\tilde{\Phi}(x) = \|x\| + \frac{1}{\eta} log(1 + e^{-2\eta\|x\|})$ for some positive η. Its gradient can be derived as $\nabla_x \tilde{\Phi}(x) = (\frac{1-e^{-2\eta\|x\|}}{1+e^{-2\eta\|x\|}}) \frac{x}{\|x\|}$. For this choice of potential, [11] also shows one of its compatible metric is a isotropic metric in the form $M_{uni} = w(x)I$, with weight function $w(x) = e^{-\frac{\|x\|^2}{2\sigma^2}} w_u + (1 - e^{-\frac{\|x\|^2}{2\sigma^2}}) w_l$, for some $\sigma \in \mathbb{R}$, and $0 \leq w_l \leq w_u < \infty$. The curvature induced by this metric is given by $\xi_M(x,\dot{x})$, which satisfies $-M_{uni}^{-1}\xi_M(x,\dot{x}) = \frac{1}{2}\|x\|^2(I - 2\frac{\dot{x}}{\|\dot{x}\|}\frac{\dot{x}^\top}{\|\dot{x}\|})\nabla_x log(w(x))$. Finally we define the damping matrix $\tilde{B} = \varepsilon I$, for some $\varepsilon \in \mathbb{R}_+$.

The orientation attractors are implemented in the same way as the position attractors, except for the task space definition. For 2-D orientation, a single angle measurement is sufficient to characterize the orientation of the end-effector. Thus, for the 2-D MATLAB simulation, we used the task-space coordinate $x = \alpha_{current} - \alpha_{target}$, where $\alpha_{current}$ and α_{target} are current and target orientation angles, respectively. Task space definition for 3-D orientation is more involved, but similar idea can be realized through quaternions.

REFERENCES

[1] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," *IEEE International Conference on Robotics and Automation*, 2009.

[2] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, 2016.

[3] S. R. Kuindersma, E. Hannigan, D. Ruiken, and R. A. Grupen, "Dexterous mobility with the ubot-5 mobile manipulator," *International Conference on Advanced Robotics*, 2009.

[4] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, 1987.

[5] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs," *Robotics: Science and Systems*, 2016.
[6] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," *IEEE International Conference on Robotics and Automation*, 2016.
[7] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, 2003.
[8] F. Grasser, A. D'arrigo, S. Colombi, and A. C. Rufer, "JOE: a mobile, inverted pendulum," *IEEE Transactions on Industrial Electronics*, 2002.
[9] L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems," Ph.D. dissertation, Stanford university USA, 2007.
[10] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018.
[11] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "RMPflow: A computational graph for automatic motion policy generation," *Workshop on the Algorithmic Foundations of Robotics*, 2018.
[12] M. Zafar and H. I. Christensen, "Whole body control of a wheeled inverted pendulum humanoid," *IEEE-RAS 16th International Conference on Humanoid Robots*, 2016.
[13] A. Li, C.-A. Cheng, B. Boots, and M. Egerstedt, "Stable, concurrent controller composition for multi-objective robotic tasks," *IEEE Conference on Decision and Control*, 2019.
[14] D. S. Bernstein, *Matrix mathematics: theory, facts, and formulas*. Princeton University Press, 2009.
[15] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, 2005.
[16] M. Zafar, S. Hutchinson, and E. A. Theodorou, "Hierarchical optimization for whole-body control of wheeled inverted pendulum humanoids," *IEEE International Conference on Robotics and Automation*, 2019.