The 3rd Joint International Conference on Multibody System Dynamics
The 7th Asian Conference on Multibody Dynamics
June 30-July 3, 2014, BEXCO, Busan, Korea

Dynamics Compensation for the Control of Articulated Multi-Limb Robots

Abhinandan Jain*, **Calvin Kuo***, **Ivan Sinkarenko***

* Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Drive, Pasadena, CA 91109, USA
Abhi.Jain@jpl.nasa.gov, calvink@stanford.edu, ivan.sinkarenko@gmail.com

ABSTRACT

We describe a general approach for using linearizing feedforward control terms for large degree of freedom (dof) multi-limb robots operating in scenarios involving motion and force constraints and under-actuated dofs arising from the task and the environment. Our solution is general and has low computational cost needed for real-time control loops. It supports the tuning of the feedforward term to meet multiple task objectives. Being structure-based, it is able to easily accommodate changes in motion and force constraints that often occur in robotics scenarios.

1 INTRODUCTION

Mobility and manipulation of multi-limb robots, such as humanoids and legged robots requires the coordinated control of multiple coupled degrees of freedom (dof) in the system. Example scenarios include robots walking on an uneven surface, climbing a ladder or using a tool. Beyond the large number of dofs, mobility and manipulation control challenges for such robotic scenarios include: the highly non-linear nature of the system dynamics; the under-actuated nature of the system (i.e. not all dofs are actuated); motion constraints on the system; the time varying nature of the constraints (eg. leg contact with the ground); and the need to meet multiple control objectives.

Techniques for handling such control problems include the combined use of feedforward and feedback control to remove nonlinearities and obtain uniform control performance across the configuration space. The computed torque method for unconstrained manipulators is a well known example of such a technique [2]. Feedforward terms are used to generate actuator commands that exactly meet the control objectives assuming perfect sensing and control. The feedback terms correct for deviations that arise from imperfections in sensing and control. The use of feedforward compensation improves precision, reduces the demands on the feedback controller and improves its robustness and performance despite the large dynamic range in the nature of the tasks. Reference [7] describes an elegant extension of this feedforward control approach for mobile legged robots. The alternative whole body control approach [9] decomposes the control problem based on prioritized objectives using projected operational space techniques. The strength of this approach is that it poses the control problem directly in task space, but it can be computationally demanding.

Our motivation for this research is to develop a general purpose control paradigm that handles the complex interactions across a broad variety of multi-arm manipulation, legged/wheel mobility and combination thereof robotic tasks potentially involving articulated and constrained task objects. With this objective in mind, we present a general formulation of the feedforward control approach that meets these goals. While we are close in spirit to [7], our approach differs in some simple, but crucial respects that make it more broadly applicable, as well as less complex and lower cost. The technique in reference [7] focuses on legged systems with under-actuated dofs for the robot's torso, uses joint selection matrices which narrows down its applicability, and adds complexity by eliminating the constraint contact forces from the dynamics model. In contrast, our approach introduces the more general notion of passive dofs which can arise from the robot dofs well as the task object dofs. Moreover, we do not require passive dof torques to be zero, and instead only require them to be known functions of the robot state. These basic changes allow our formulation to cover a very large family of combined manipulation and mobility activities. Also, we avoid the generalized inverse steps needed in [7], and derive a direct, simpler and lower-cost feedforward control formulation that is very general. Our integrated robot/task/environment perspective decouples the impact of the constraints on the permissible motion from the way that the passive dofs affect the feedforward solution.

The resulting space of feedforward solutions allows us to further refine the solution to meet secondary task objectives.

We demonstrate our new general purpose control technique for scenarios consisting of legged robots performing a variety of mobility and manipulation tasks such as the opening of valves, climbing ladders and walking across slopes in simulation.

2 Feedforward Inverse Dynamics for Control

We begin by reviewing the use of feedforward control for the motion control of an unconstrained branched topology manipulator. Using $\mathcal{N}$ to denote the number of dofs, the equations of motion for such a robotic system can be expressed as

$$\mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta,\ \dot{\theta}) = \mathcal{T} \tag{1}$$

where the configuration dependent, symmetric matrix $\mathcal{M}(\theta) \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$ is the *mass matrix* of the system, $\mathcal{C}(\theta,\ \dot{\theta}) \in \mathcal{R}^{\mathcal{N}}$ includes the velocity dependent Coriolis and gyroscopic forces as well as the gravitational forces, and $\mathcal{T} \in \mathcal{R}^{\mathcal{N}}$ denotes the applied generalized forces. The mass matrix is positive-definite and invertible for branched systems. We generalize the definition of the $\mathcal{C}(\theta,\ \dot{\theta})$ to also include all *explicitly* known generalized force contributions such as $\mathcal{J}_e^* \mathfrak{f}_{ext}$ terms from known end-effector spatial force $\mathfrak{f}_{ext}$ where $\mathcal{J}_e$ denotes the end-effector Jacobian. The general idea is that $\mathcal{C}(\theta,\ \dot{\theta})$ includes all the explicitly known and state-dependent terms that appear in the equations of motion.

The motion and tracking control problem for robots requires feedback control that will drive the robot along a desired motion trajectory while meeting requirements on position and trajectory tracking error, disturbance rejection etc. While a substantial body of techniques for doing this is available for linear time-invariant dynamical systems, they do not directly apply to non-linear dynamical systems such as in Eq. 1. Linear feedback control techniques perform poorly due to the widely varying characteristics of the system across the configuration space.

The *computed torque* method for robot control provides a solution to this problem [2]. The basic idea is to use a feedforward control term to linearize the system dynamics, so that linear control theory techniques can once again be used to design the feedback controller. Assuming that all the dofs are actuated, the computed torque approach uses a feedforward torque of the form

$$\mathcal{T}_{ff} = \text{TINV}(\ddot{\theta}_d,\ \dot{\theta}, \theta) \quad \text{where} \quad \text{TINV}(x) \triangleq \mathcal{M}(\theta)x + \mathcal{C}(\theta,\ \dot{\theta}) \tag{2}$$

The $\text{TINV}(x,\ \dot{\theta}, \theta)$ function represents the standard inverse dynamics computation of generalized forces for a vector x of joint accelerations for a tree topology system. For brevity we will use $\text{TINV}(x)$ instead of $\text{TINV}(x,\ \dot{\theta}, \theta)$ with the current $(\ \dot{\theta}, \theta)$ state values implied for the missing arguments. Low-cost Newton-Euler recursive algorithms to carry out this inverse dynamics computation are well known [6]. Including in any prescribed motion dofs is straightforward as well. With tracking error $\epsilon \triangleq \theta_d - \theta$, a feedback term of the form $\mathcal{T}_{fb} = \mathcal{M}(\theta)\left[K_v\dot{\epsilon} + K_p\epsilon\right]$ is used to apply an overall actuator torque of the form

$$\begin{aligned}\mathcal{T} &\triangleq \mathcal{T}_{ff} + \mathcal{T}_{fb} + \mathcal{T}_n = \mathcal{M}(\theta)\left[\ddot{\theta}_d + K_v\dot{\epsilon} + K_p\epsilon\right] + \mathcal{C}(\theta,\ \dot{\theta}) + \mathcal{T}_n \\ &= \text{TINV}(\ddot{\theta}_d + K_v\dot{\epsilon} + K_p\epsilon) + \mathcal{T}_n\end{aligned} \tag{3}$$

Here $\mathcal{T}_n$ denotes a noise/disturbance term. When used in Eq. 1 this generalized torque leads to the following error dynamics equation:

$$\ddot{\epsilon} + K_v\dot{\epsilon} + K_p\epsilon = -\mathcal{M}^{-1}\mathcal{T}_n \tag{4}$$

These error dynamics represent a linear, time-invariant system, whose stability is easy to analyze, and for which the K_v and K_p gain terms can be easily chosen to meet the desired control objectives across the full configuration space. While the use of such a model based feedforward torque computation is far more expensive than just basic PID control, it offers robust and consistent performance across the configuration space despite the highly nonlinear nature of the underlying dynamics.

It is noteworthy that in the ideal case with perfect model and sensing and no noise, the feedforward force $\mathcal{T}_{ff}$ generates the desired motion for the robotic system with $\epsilon \equiv 0$. This motivates the use of the feedforward

term, and the the feedback term to handle modeling and sensing errors that occur in reality. A-priori knowledge and on-line estimators are typically used to refine and improve the accuracy of the model parameters. Note that the feedforward term reduces to the familiar gravity compensation term for the static case.

As observed above, the feedforward term can be regarded as the actuation generalized force that will exactly lead to the desired generalized motion and forces in the idealized scenario of perfect knowledge of the model and system state. With this in mind, for simplicity of notation, we will from now on simply use $\ddot{\theta}$ for the desired acceleration $\ddot{\theta}_d$, and $\mathcal{T}$ for $\mathcal{T}_{ff}$.

In the following sections we look at the feedforward problem, first for robotic systems subject to kinematic motion constraints, then for under-actuated systems with passive dofs, and finally the general case of systems with both motion constraints and passive dofs.

Feedforward with kinematic constraints: The dynamics of a robotic system subject to kinematic constraints can be obtained by modifying the unconstrained system dynamics in Eq. 1 to include the effect of the kinematic constraints via *Lagrange multipliers*, $\lambda \in \mathcal{R}^{n_c}$, as follows[1]

$$\begin{pmatrix} \mathcal{M} & G^* \\ G & \mathbf{0} \end{pmatrix} \begin{bmatrix} \ddot{\theta} \\ -\lambda \end{bmatrix} = \begin{bmatrix} \mathcal{T} - \mathcal{C} \\ \acute{\mathcal{U}} \end{bmatrix} \tag{5}$$

Here $G(\theta, t) \in \mathcal{R}^{n_c \times \mathcal{N}}$ denotes the full row rank constraint matrix that defines the kinematic constraints on the generalized velocity coordinates via $G(\theta, t)\,\dot{\theta} = \mathcal{U}(t)$, with the acceleration level constraint equation given by $G(\theta, t)\ddot{\theta} = \acute{\mathcal{U}}(t)$ where $\acute{\mathcal{U}} \triangleq \dot{\mathcal{U}}(t) - \dot{G}\,\dot{\theta} \in \mathcal{R}^{n_c}$. The $-G^*(\theta, t)\lambda$ term represents the (implicitly defined) internal generalized constraint forces arising from the constraints.

For a $\ddot{\theta}$ that is consistent with the constraints, it is easy to verify that $\mathcal{T} \triangleq \text{TINV}(\ddot{\theta})$ satisfies Eq. 5 with $\lambda = 0$. Indeed $\mathcal{T} \triangleq \text{TINV}(\ddot{\theta}) + G^* x$, for arbitrary x, is also a solution with $\lambda = -x$! Thus the Lagrange multipliers λ serve as a free parameter for the feedforward $\mathcal{T}$, and have no affect on the motion of the system, and can be used to control the internal *squeeze* forces within the system. Thus, λ can be chosen to meet additional objectives such as load-balancing of forces across the joints.

Thus, the feedforward strategy for such kinematically constrained systems is to choose a desired generalized acceleration $\ddot{\theta}$ that is consistent with the constraints and compute the feedforward generalized forces as

$$\mathcal{T} = \text{TINV}(\ddot{\theta}) + G^*\lambda \tag{6}$$

with λ either zero, or chosen to optimize some function of the internal forces. Thus the loss of motion dofs from the constraints is compensated by the ability to control the same number of dofs in the force domain.

An important requirement for using Eq. 6 to generate the feedforward accelerations is that the $\ddot{\theta}$ be consistent with the constraints. Due to the presence of the constraints, the motion of the system is restricted, and only some of the components of $\ddot{\theta}$ are independent. We select a subset as independent coordinates, $\ddot{\theta}_r \in \mathcal{R}^{\mathcal{N}-n_c}$, and the remainder set as dependent coordinates, $\ddot{\theta}_d \in \mathcal{R}^{n_c}$, so that

$$G\ddot{\theta} = [G_r,\ G_d] \begin{bmatrix} \ddot{\theta}_r \\ \ddot{\theta}_d \end{bmatrix} = \acute{\mathcal{U}} \quad \Longrightarrow \quad \ddot{\theta}_d = G_d^{-1}[\acute{\mathcal{U}} - G_r\ddot{\theta}_r] \tag{7}$$

The partitioning above is chosen such that the $G_d \in \mathcal{R}^{n_c \times n_c}$ sub-matrix of $G(\theta, t)$ is invertible. Such a choice is always possible due to the full row rank assumption for G. The $(\mathcal{N} - n_c)$ dimensional $\ddot{\theta}_r$ vector in Eq. 7 parametrizes the subspace of generalized accelerations that is consistent with the constraints. Thus, we can think of $\ddot{\theta}_r$ as the minimal, independent generalized acceleration coordinates for the system. Eq. 7 provides a way to obtain the full and consistent $\ddot{\theta}$ generalized acceleration coordinates from $\ddot{\theta}_r$ via

$$\ddot{\theta} = \underline{\ddot{\theta}}_q + X\ddot{\theta}_r \quad \text{where} \quad \underline{\ddot{\theta}}_q \triangleq \begin{bmatrix} 0 \\ G_d^{-1}\acute{\mathcal{U}} \end{bmatrix} \quad \text{and} \quad X \triangleq \begin{bmatrix} I \\ -G_d^{-1}G_r \end{bmatrix} \in \mathcal{R}^{\mathcal{N} \times \mathcal{N} - n_c} \tag{8}$$

[1] For a matrix A, the A^* notation denotes its matrix transpose.

Eq. 8 gives us a way to recover the full generalized acceleration $\ddot{\theta}$ vector given the independent generalized acceleration vector $\ddot{\theta}_r$. Note that $GX = \mathbf{0}$.

Feedforward with passive dofs: For many robotic systems, not all dofs are actuated, i.e. some of the dofs are passive. Such systems are referred to as under-actuated systems, with examples including mobile wheeled and legged robots. Passive dofs may also arise from the task object and environment, such as doors, door handles etc. The characteristic of passive dofs is that their generalized force values cannot be commanded, but are instead a known function of the system state. While passive generalized forces are often zero, this is not a requirement. Denoting the number of passive dofs by n_p, we partition the dofs into *active* dofs $\theta_a \in \mathcal{R}^{N-n_p}$, and *passive* dofs $\theta_p \in \mathcal{R}^{n_p}$. Eq. 1 takes the partitioned form

$$\begin{pmatrix} M_{aa} & M_{ap} \\ M_{ap}^* & M_{pp} \end{pmatrix} \begin{bmatrix} \ddot{\theta}_a \\ \ddot{\theta}_p \end{bmatrix} + \begin{bmatrix} C_a \\ C_p \end{bmatrix} = \begin{bmatrix} \mathcal{T}_a \\ \mathcal{T}_p \end{bmatrix} \tag{9}$$

The (unactuated) passive generalized forces $\mathcal{T}_p$ are a known function of the system state $(\theta,\ \dot{\theta})$. The passive dofs represent constraints in the generalized force space, and are a dual to the kinematic constraints discussed earlier. The form of Eq. 9 is not convenient because of its implicit nature, since it contains a mix of the known and unknown quantities on both the left and right hand sides of the equation. A transformation of Eq. 9 that expresses the unknown quantities in terms of the known quantities is as follows:

$$\begin{bmatrix} \mathcal{T}_a \\ \ddot{\theta}_p \end{bmatrix} = \begin{pmatrix} S_{aa} & S_{ap} \\ -S_{ap}^* & S_{pp} \end{pmatrix} \begin{bmatrix} \ddot{\theta}_a \\ \mathcal{T}_p \end{bmatrix} + \begin{bmatrix} C_a - S_{ap} C_p \\ -S_{pp} C_p \end{bmatrix} \tag{10a}$$

$$\text{where} \quad S_{aa} \triangleq M_{aa} - M_{ap} M_{pp}^{-1} M_{ap}^*, \quad S_{ap} \triangleq M_{ap} M_{pp}^{-1}, \quad S_{pp} \triangleq M_{pp}^{-1} \tag{10b}$$

Feedforward terms only apply to $\mathcal{T}_a$ since actuators can be commanded to only set this subset of $\mathcal{T}$. For a desired $\ddot{\theta}_a$ generalized acceleration at the active hinges, Eq. 10a provides us with a way to evaluate the necessary feedforward $\mathcal{T}_a$ active hinge forces, and the passive hinge accelerations $\ddot{\theta}_p$ induced by this motion. Thus arbitrary values for only the $\ddot{\theta}_a$ active generalized acceleration subset of the full $\ddot{\theta}$ generalized acceleration are achievable for such passive systems. Techniques to manage (and perhaps minimize) the induced $\ddot{\theta}_p$ passive generalized accelerations, and to evaluate them efficiently without having to explicitly compute the sub-matrices in Eq. 10b are discussed in references [4, 5].

Feedforward with constraints and passive dofs: More generally, robotic systems have both motion constraints as well as passive dofs. Prominent examples of such systems are once again wheeled and legged mobile robots. For these systems, the torso and chassis dofs are passive, while their motion is constrained by the contact between the wheels/feet and the ground. For a desired $\ddot{\theta}$ (consistent with the motion constraints), we partition the feedforward expression in Eq. 6 between the passive and active dofs as follows:

$$\begin{bmatrix} \mathcal{T}_a \\ \mathcal{T}_p \end{bmatrix} = \begin{bmatrix} \mathcal{T}_a^f \\ \mathcal{T}_p^f \end{bmatrix} - \begin{bmatrix} G_a^* \\ G_p^* \end{bmatrix} \lambda \quad \text{where} \quad \mathcal{T}^f(\ddot{\theta}) \quad = \begin{bmatrix} \mathcal{T}_a^f(\ddot{\theta}) \\ \mathcal{T}_p^f(\ddot{\theta}) \end{bmatrix} \triangleq \text{TINV}(\ddot{\theta}) \tag{11}$$

The $G_a^* \in \mathcal{R}^{(N-n_p)\times n_c}$ and $G_p^* \in \mathcal{R}^{n_p \times n_c}$ matrices represent a partitioning of G^* based on the active and passive dofs in the system. $\mathcal{T}^f$ can be readily evaluated given the $\ddot{\theta}$ desired acceleration. Since $\mathcal{T}_p$ is known, we can solve for λ using the passive half of the equation:

$$G_p^* \lambda = \mathcal{T}_p - \mathcal{T}_p^f \tag{12}$$

A solution for Eq. 12 will exist if $n_c \geqslant n_p$ and G_p^* has full row rank. Assuming a solution for λ exists, the $\mathcal{T}_a$ feedforward term to meet the full desired $\ddot{\theta}$ motion accelerations in the presence of the $\mathcal{T}_p$ passive generalized forces can be computed using the active half of Eq. 11:

$$\mathcal{T}_a = \mathcal{T}_a^f - G_a^* \lambda \tag{13}$$

In contrast with the case of unconstrained robotic systems with passive dofs, the presence of constraints allows us to attain the full desired generalized accelerations including the passive generalized accelerations. Thus in the presence of passive dofs, motion constraints allow the use of the constraint forces to help to fill

in for the missing actuation forces for the passive dofs. In effect, the motion constraints provide a way to introduce actuators that are otherwise missing for the passive dofs.

When $n_c < n_p$, a solution for Eq. 12 is not guaranteed. In this case not all accelerations consistent with the constraints are achievable. For this case we have

$$\mathcal{T} \overset{5}{=} \mathcal{M}(\theta)(\underline{\ddot{\theta}}_q + X\ddot{\theta}_r) + \mathcal{C}(\theta, \dot{\theta}) - G^*(\theta, t)\lambda \overset{2,11}{=} \mathcal{M}(\theta)X\ddot{\theta}_r + \mathcal{T}^f(\underline{\ddot{\theta}}_q) - G^*\lambda \tag{14}$$

The passive dof rows give us the following condition on the feasible $(\ddot{\theta}_r, \lambda)$ values:

$$\mathcal{T}_p \overset{9,11}{=} [\mathcal{M}^*_{ap}, \mathcal{M}_{pp}]X\ddot{\theta}_r + \mathcal{T}^f_p(\underline{\ddot{\theta}}_q) - G^*_p\lambda \implies \left([\mathcal{M}^*_{ap}, \mathcal{M}_{pp}]X, -G^*_p\right)\begin{bmatrix}\ddot{\theta}_r \\ \lambda\end{bmatrix} = \mathcal{T}_p - \mathcal{T}^f_p(\underline{\ddot{\theta}}_q) \tag{15}$$

The $(\ddot{\theta}_r, \lambda)$ solutions to this equation define the feasible motions and internal forces. These solutions can be used to define the viable $\ddot{\theta}$ using Eq. 8, followed by using Eq. 13 to evaluate the $\mathcal{T}_a$ feedforward term. In summary, the steps involve in evaluating the $\mathcal{T}_a$ generalized feedforward term are

1. Compute desired generalized accelerations that are consistent with the motion constraints. This can be done by planning in the $\ddot{\theta}_r$ independent generalized accelerations space, and using Eq. 8 to obtain the full set of generalized accelerations $\ddot{\theta}$.
2. Compute the $\mathcal{T}^f(\ddot{\theta}) = \text{TINV}(\ddot{\theta})$ free generalized forces. This is the familiar unconstrained computed torque computation. As shown in Eq. 9, extract the $\mathcal{T}^f_p$ sub-vector from this based on the passive dofs.
3. If full row rank conditions hold, solve $G^*_p\lambda = \mathcal{T}_p - \mathcal{T}^f_p$ from Eq. 12 for λ. Else, use Eq. 15 to solve for $(\ddot{\theta}_r, \lambda)$. When there are multiple solutions, pick the solution that optimizes other task objectives.
4. Evaluate $\mathcal{T}_a = \mathcal{T}^f_a - G^*_a\lambda$ from Eq. 13 to obtain the feedforward active generalized forces.

The constraints only effect the desired consistent $\ddot{\theta}$ generalized acceleration choice in step (1), while the passive dofs primarily effect steps (3) and (4). While the passive dofs are defined by the physics of the robot and the scenario, the independent generalized accelerations dofs are chosen based on motion planning convenience. There is no requirement that they be the same or have any overlap. Eq. 12 defines the connection between them since $\mathcal{T}^f_p$ is determined by the passive dofs, while G is determined by the constraints. When there are no passive dofs, $n_p = 0$, λ can be arbitrary. When there are no constraints, $n_c = 0$, and only $\ddot{\theta}$ satisfying Eq. 15 can be exactly met. When there are neither constraints nor passive dofs, we reduce to the standard computed torque feedforward term. Each of the steps is low cost. The cost of the optional λ optimization in step (3) however depends on the criteria and technique employed.

An important special case that often occurs is when the kinematic constraints arise from *loop constraints*, i.e constraints on the spatial velocities of locations (e.g. end-effectors, feet) on the robotic system. Let us assume that there are n_b such *loop closure* nodes, with $\mathcal{V}_b \in \mathcal{R}^{6n_b}$ denoting the stacked vector of spatial velocities of these nodes. The constraints on these nodal spatial velocities is defined via a constraint matrix $\mathcal{Q} \in \mathcal{R}^{n_c \times 6n_b}$ such that $\mathcal{Q}\mathcal{V}_b = \mathfrak{U}(t)$. With $\mathcal{J}_b \in \mathcal{R}^{6n_b \times \mathcal{N}}$ denoting the Jacobian matrix for these nodes

$$\mathcal{V}_b = \mathcal{J}_b\,\dot{\theta} \;\Rightarrow\; \mathcal{Q}\mathcal{J}_b\,\dot{\theta} = \mathfrak{U}(t) \;\Rightarrow\; G = \mathcal{Q}\mathcal{J}_b \quad \text{and} \quad G_p = \mathcal{Q}\mathcal{J}_p, \quad \text{where} \quad \mathcal{J}_b = [\mathcal{J}_a, \mathcal{J}_p] \tag{16}$$

Thus G and G_p have a special structure for the important special case of loop constraints. In the above $\mathcal{J}_a \in \mathcal{R}^{6n_b \times (\mathcal{N} - n_p)}$ and $\mathcal{J}_p \in \mathcal{R}^{6n_b \times n_p}$ represent a partitioning of the columns of the constraints node's Jacobian matrix $\mathcal{J}_b$ in accordance with the active and passive dofs.

Unilateral constraints: Our feedforward control formulation thus far has assumed that all constraints are bilateral, i.e. equality constraints. There are many scenarios however when some of the constraints are unilateral, i.e. inequality constraints. Examples include contact constraints for legged robots, hands grasping a task object etc. Active unilateral constraints can indeed be treated as bilateral constraints. However, unlike bilateral constraints, unilateral constraints can become inactive, and can even disappear (e.g. when contact is broken). For simplifying the feedforward computation, as in reference [7], we treat the unilateral constraints as bilateral constraints for the feedforward evaluation, with the consistency requirement that the

feedforward solution satisfy conditions necessary for the unilateral constraints to be active. For contact constraints, this typically requires the solution contact forces to lie within the friction cone to satisfy the no slip condition. Satisfaction of such consistency requirements cannot be guaranteed. We refer the reader to [7] for a more detailed discussion of this topic and useful techniques for helping meet the consistency requirement. The expensive, but rigorous alternative option is to append the inequality conditions for the unilateral constraints to Eq. 12, and use quadratic programming like techniques [8] to find, and even optimize the solution to meet performance objectives.

Having developed our feedforward control formulation for the general case, we now describe several robotics application scenarios to illustrate its use.

3 Multi-arm manipulation

This scenario focuses on multi-arm manipulation, such as for multiple limbs or fingers manipulating, carrying or grasping a task object. The primary objective is to move the task object in a desired trajectory, while meeting sub-objectives such as equitably balancing the load across the arms, or ensuring end-effector forces do not damage the task object, or grasp closure constraints to apply sufficient force to avoid slippage, or to avoid overloading and saturating actuators. Let m denote the number of arms grasping the task object. For simplicity we focus on the dual-arm/rigid grasp case, i.e. $m = 2$, though the approach easily generalizes to the multi-arm and non-rigid grasp case. The approach we describe also easily simplifies to the case of a single arm, e.g. scenarios involving a single arm using a drill, or pushing a lever.

Rigid, unconstrained task object: We begin with the case where the task object is a rigid object, and the only constraints on it are from the arm end-effectors that are moving it in free space. For this system, the grasped rigid body has 6 passive dofs, and the loop constraints are from the attachment/grasp points on the task object. The constraint nodes are the end-effector nodes for each of the arms, and the attachment nodes on the grasped body. The system dofs consist of the joint dofs for each of the arms together with the 6 passive dofs for the task object. Thus for the two arm and rigid grasp case

$$n_p = 6, \quad n_b = 4, \quad n_c = 6n_b/2 = 12 \quad \text{and} \quad \mathcal{T}_p = \mathbf{0} \tag{17}$$

$\mathcal{J}_b$ and $\mathcal{Q}$ are given by

$$\mathcal{J}_b = \left(\begin{array}{cc|c} \mathcal{J}_r & 0 & 0 \\ 0 & \mathcal{J}_l & 0 \\ 0 & 0 & \mathcal{J}_{tr} \\ 0 & 0 & \mathcal{J}_{tl} \end{array}\right) \in \mathcal{R}^{6n_b \times \mathcal{N}}, \quad \mathcal{Q} = \begin{pmatrix} I & 0 & -I & 0 \\ 0 & I & 0 & -I \end{pmatrix} \in \mathcal{R}^{n_c \times 6n_b} \tag{18}$$

$\mathcal{J}_r$ and $\mathcal{J}_l$ denote the end-effector Jacobians for the right and left arms respectively, and $\mathcal{J}_{tr}$ and $\mathcal{J}_{tl}$ denote the task object Jacobians for the right and left attachment points respectively. For a rigid task object, $\mathcal{J}_{tr} = \phi_r^*$ and $\mathcal{J}_{tl} = \phi_l^*$, where the ϕ_r and ϕ_l 6×6 matrices denote the rigid body transformation matrices from the right and left attachment points to the task object's reference frame. The $\mathcal{J}_p \in \mathcal{R}^{6n_b \times n_p}$ passive Jacobian corresponds to the block column for the passive dofs in the right part of the $\mathcal{J}_b$ matrix in Eq. 18. Note that we never need to compute the full $\mathcal{J}_b$ matrix, just the much smaller $\mathcal{J}_p$ matrix. Since $n_c \geqslant n_p$ is satisfied, Eq. 13 holds, and we obtain the feedforward condition

$$\mathcal{T}_p^f - \mathcal{T}_p \overset{12,17}{=} -\mathcal{J}_p^* \mathcal{Q}^* \lambda \overset{18}{=} -[\mathcal{J}_{tr},\, \mathcal{J}_{tl}]\lambda \tag{19}$$

Note that $\mathcal{T}_p^f$ represents the D'Alembert spatial force needed at the task object's reference frame to move it along the desired motion trajectory. The λ solutions to Eq. 19 are the end-effector forces that can provide this D'Alembert force, and can be used to evaluate the $\mathcal{T}_a$ active feedforward forces using Eq. 13.

For a single arm statically holding a heavy object, the feedforward term reduces to the additional joint torques needed to compensate for the gravitational load from the task object. When there are multiple solutions, a solution to best meet secondary requirements such as load balancing across the arms, or on the end-effector forces (eg. to avoid damaging the object, preventing slippage), or avoiding actuator saturation can be selected.

Constrained rigid task object: Now let us consider the more general case where there are additional bilateral constraints on the task object. Examples of such scenarios include: two-handed polishing of a surface, two handed opening of a circular valve or a spring-loaded door, turning the steering wheel of a vehicle. $\mathcal{T}_p$ may be non-zero in some of these instances, such as when there are passive forces from a spring-loaded door, or a resistance torque that needs to be overcome when turning a valve or steering wheel.

Two options are available for extending our approach to this situation: (a) treat the task object as a floating object and append the task constraints to the system level constraints, or (b) treat the task object constraint as a hinge and the task object as a branched articulated system. Option (a) is more general, but keeps the task constraints explicit, while (b) eliminates the constraint and uses minimal number of dofs. Though preferable, option (b) can be used only if the task object has a branched structure. Option (a) may be preferable when the task object constraint is a unilateral constraint, (eg. when polishing, a positive force is required to maintain contact) since it is easier to enforce such an inequality condition when solving for λ with this option. For option (a), both n_b and n_c increase, and additional rows need to be added to the constraint Jacobian $\mathcal{J}_b$ and additional columns and rows need to added to $\mathcal{Q}$ in Eq. 18 for the task object constraints. For option (b), n_b and n_c remain unchanged, but n_p decreases to the number of dofs available to the task object from its "constraint" hinge. In Eq. 18, the form of $\mathcal{J}_b$ remains unchanged except for a reduction in the number of columns from the decrease in n_p, while $\mathcal{Q}$ remains unchanged. For either option, the left half of Eq. 19 continues to hold and solutions to it can be used to compute the $\mathcal{T}_a$ feedforward values.

Articulated task object: As a further generalization, consider the case where the task object has internal articulation dofs. Examples of such scenarios include: the use of a tool such as a two handle trimmer, or pushing a wheelbarrow along the ground. Our approach continues to apply, with the main change being that n_p increases due to the additional passive dofs in the task object. As long as the condition $n_c \geqslant n_p$ continues to hold, we can use Eq. 19 for the feedforward term - one that will even manage the internal posture of the task object. However, when $n_c < n_p$, we may not be able to meet all the motion objectives. The achievable motions are governed by the more restrictive and complex condition in Eq. 15 which the motion planner needs to take into account when planning feasible motions.

4 Legged robots

We now look at feedforward computation for legged systems. We assume that the robot consists of a torso with m legs. The 6 torso dofs are passive. The feet in contact with the ground provide support for the robot. As discussed earlier, for feedforward computation we treat the unilateral contact constraints for the feet as bilateral constraints. Clearly this assumption is valid only if the ground contact forces at the feet lie within the friction cone to avoid slippage or loss of contact. Assuming *point contact* between the feet and the ground, we have

$$n_p = 6, \quad n_b = m, \quad n_c = 3n_b \quad \text{and} \quad \mathcal{T}_p = \mathbf{0} \tag{20}$$

The constraint nodes Jacobian $\mathcal{J}_b \in \mathcal{R}^{n_c \times 6n_b}$ and $\mathcal{Q}$ are

$$\mathcal{J}_b = \left(\begin{array}{cccc|c} \mathcal{J}_1 & 0 & \cdots & \cdots & \mathcal{J}_{b1} \\ 0 & \mathcal{J}_2 & \cdots & \cdots & \mathcal{J}_{b2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathcal{J}_l & \mathcal{J}_{bm} \end{array}\right), \quad \mathcal{Q} = \begin{pmatrix} [0_3,\ I_3] & 0 & \cdots & 0 \\ 0 & [0_3,\ I_3] & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & [0_3,\ I_3] \end{pmatrix} \in \mathcal{R}^{n_c \times 6n_b} \tag{21}$$

$\mathcal{J}_i$ denotes the Jacobian for the i^{th} leg to its foot, and $\mathcal{J}_{bi}$ denotes the Jacobian from the torso dofs to the the i^{th} leg's foot. Once again $\mathcal{J}_p$ is defined by the right block column of $\mathcal{J}_b$. While $n_c \geqslant n_p$ is satisfied for $m \geqslant 2$, i.e. for systems with two or more legs in contact with the ground, the reality is that the $\mathcal{J}_p^* \mathcal{Q}^*$ matrix does not have full row rank for $m = 2$, and three or more supporting legs are required to be in contact with the ground for it to have full row rank and for solutions to Eq. 13 to be guaranteed to exist.

Biped robots typically have large feet, and the foot/ground contact represents an *area contact*. The only change for this case is that the $[0_3,\ I_3]$ elements of $\mathcal{Q}$ in Eq. 21 need to be replaced with I_6, and $n_c = 6n_b$.

The convention for legged robots is to include the torso's (passive) dofs within the independent dofs which are used for motion planning. The kinematic constraints are used to compute the dependent leg dof accel-

erations that effect the robot's posture. The λ solution vector contains the interaction force between the foot and the ground for each supporting leg. Since the ground contact constraint is in reality a unilateral constraint, for consistency with the bilateral assumption, it is necessary that the λ solution be such that the contact constraints be active (i.e. the normal force components be positive), and that there be no slippage (i.e. the contact forces lie within each foot's friction cone). Additional secondary objectives, such as load balancing, can be met by further refining the solution choice. Overall, the planned generalized acceleration $\ddot{\theta}$ can be used to select the value of $\mathcal{T}_p^f$, and the λ solution chosen to manage the ground contact forces and the loads on the leg actuators.

Center of Pressure: For legged systems, walking involves the making and breaking of contact between the feet and the ground. Since the support legs are constantly changing, the sequencing and timing of leg lift-off has to be done with care to avoid destabilizing the robot and keep it from falling. The notions of center of pressure (COP) and zero moment point (ZMP) are useful for this purpose [3]. The COP is defined as the point in the ground plane such that the torque moment on the torso is a pure twist about the local normal and the pair of tipping components are zero. For stability, motion is planned so that the COP lies within the support region defined by the feet in contact with the ground. Since $\mathcal{T}_p^f \in \mathcal{R}^6$ defines the spatial force on the torso about its reference frame B, the COP C is defined by the condition

$$N(B) + \tilde{l}(C,B)F(B) = [0,\ 0,\ x]^* \tag{22}$$

where $N(B)$ and $F(B)$ denote the moment and force 3-vector components of $\mathcal{T}_p^f$, $l(C,B) \in \mathcal{R}^3$ denotes the vector from C to B and x is some scalar. The above equation has rank 2, and its top 2 rows can be used to solve for the x and y components of $l(C,B)$ which locate the COP on the ground plane, while the z component represents the known height of B above the ground plane. Thus we need to solve

$$N_x(B) = -l_z F_y(B) + l_y F_z(B) \quad \text{and} \quad N_y(B) = l_z F_x(B) - l_x F_z(B) \tag{23}$$

The requirement that the COP lie within the support area places restrictions on the permissible $\mathcal{T}_p^f$, which in turn restricts the achievable independent $\ddot{\theta}$ generalized accelerations.

While the full equations of motion are needed to compute $\mathcal{T}_p^f$ for the desired $\ddot{\theta}$, a common simplifying assumption is to use an inverted pendulum model as a basis for the $\mathcal{T}_p^f \approx M(B)\ddot{\theta}(B)$ approximation, where $M(B) \in \mathcal{R}^{6\times6}$ and $\ddot{\theta}(B) \in \mathcal{R}^6$ denote the spatial inertia and the desired spatial acceleration for the torso. The advantage of this approximation is that the torso's spatial inertia is constant and the unconstrained inverse dynamics computation is avoided. This approximation in essence assumes that the contribution of the legs to the system spatial inertia and to the torso inertial forces is negligible and that Coriolis and gyroscopic force contributions can also be neglected. These assumptions are entirely optional and only used to reduce the computational burden.

Note that this approach is valid even when the ground plane is not level, since the pure twist requirement in Eq. 23 is about the local normal. The primary impact of this is that the gravitational forces contribution has to be rotated into the local ground plane frame. This approach remains valid even when the torso has additional dofs and arms. During manipulation, any additional constraints on the arms can be handled as described in Section 3 and used to augment the constraint Jacobian and the other matrices. Again, generalizing to multiple robots performing a task in coordination is straightforward.

Wheeled Robots: The feedforward terms for wheeled platforms with arms can also be evaluated by our approach especially where the mobility and manipulation dofs need to be coordinated during task execution. The primary difference from the legged instance is that the constraint between wheels and the ground differs from that between feet and the ground, and requires a wheel constraint version of Q in Eq. 21. Also, the COP method is not relevant here since foot placement is not an issue for wheeled robots.

5 Simulation Examples

We have implemented a control architecture based on our generalized feedforward control formulation. We demonstrate its versatility by simulating a variety of tasks from the Defense Advanced Research Projects Agency Robotics Challenge (DRC) competition for the Jet Propulsion Laboratory's (JPL) 70-dof, 4-limb

RoboSimian robot. The DRC tasks consisted of: (a) vehicle driving, (b) clearing debris, (c) climbing a ladder, (d) traversing uneven terrain, (e) opening a door, (f) making a hole in a wall, (g) opening a valve, and (h) mating a hose. The control modules automatically generate the requisite constraint Jacobian matrix and feedforward terms based on the constraints, passive dofs and robot configuration. Even though these change significantly from task to task and even within a task, the control modules are able to perform this broad variety of tasks without any additional customization. The DRC tasks can be grouped into three categories as described below.

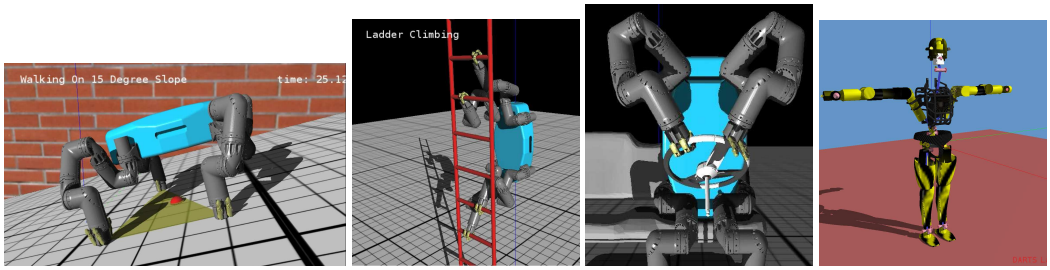

Figure 1. The feedforward control technique has been used to simulate mobility and manipulation control tasks for the RoboSimian and Atlas multi-limb, legged robots for tasks including walking, climbing a ladder and turning a steering wheel.

The first category consists of single arm manipulation tasks with some legs providing support. This category encompasses the debris, door, hose, and wall tasks. These tasks are characterized by having three support limbs and a manipulation limb. The formulation for these tasks is relatively straight forward as the ground constraints remain fixed for the task duration. Thus, the constraint Jacobian simply handles the ground constraints and the manipulation limb motion is unconstrained.

The second category tasks involve multi-limb manipulation of a constrained task object. This includes the vehicle and valve tasks. In both scenarios, the valve and steering wheel task objects are constrained to the environment through a passive rotational joint with non-zero resistance torque. Both manipulation limbs have constraints between their hands and the task object. Thus, the constraint Jacobian, in addition to including terms for the support limb constraints, also include the constraint between manipulation limbs and the task object.

The final task category consists of the legged mobility task (the uneven terrain traverse and ladder scenarios) in which the limbs are used for walking. The main characteristic of this task is that the limbs themselves are constantly switching between ground contact support and free swing motion, and thus the constraint Jacobian must be updated after each such transition. We used a quadruped walking gait consisting of a repeating sequence of transitions from quadruped support, to tripod support during leg swing, and then back to quadruped support. In the quadruped state, the constraint Jacobian includes ground contact constraint terms for all the limbs, while in the tripod support state, the constraint Jacobian only includes constraint terms for the three support legs as the fourth leg is in free motion. In the case of ladder climbing, the bilateral constraints between the hands and the ladder rungs only allow rotational motion about the rung axis.

As a final note, we also used our generalized feedforward control architecture to simulate walking for a Boston Dynamics 36-dof Atlas biped robot. Biped walking for the Atlas consists of a repeating sequence of transitions from dual support, to single leg support during leg swing phase, and back to dual support. As in the RoboSimian case, the constraint Jacobian must be updated at each such transition to handle the limbs coming in and out of ground contact. Unlike the RoboSimian case, the unilateral ground contact constraints were modeled as area contacts instead of point contacts.

While the unilateral constraints were modeled as bilateral constraints for the feedforward control computation, the simulation model made no such assumption and treated them as unilateral constraints. The dynamics model as well as the closed-loop control modules, were simulated using JPL's DARTS [1] minimal

coordinates, recursive dynamics simulation software that is based on the spatial operator algebra methodology [4]. Figure 1 shows screen shots from some of the simulations.

6 Conclusions

In this research we describe a general feedforward control framework that applies to a broad class of robotic mobility and manipulation scenarios, where robots can be subject to motion constraints and under-actuated dofs. The feedforward evaluation procedure includes at its heart the well known computed torque feedforward evaluation for unconstrained systems, but includes steps to include the motion constraints to define admissible motions, and to select feedforward solutions that are compatible with the passive dofs. We derive conditions for the existence of feedforward solutions that meet the control objectives. Options to tune the solution to meet secondary objectives are also provided. Our approach is not only general, but has reduced computational cost for embedded control use. Multiple representative scenarios involving multi-arm manipulation and mobile robots are used to illustrate the application of the feedforward procedure.

We demonstrate in simulation how the feedforward control architecture can be used to handle a large variety of mobility and manipulation tasks. The different scenarios only require the appropriate modeling of the constraints and passive dofs and avoid the need for custom control schemes for each task. Besides allowing the control framework to easily handle a variety of tasks, the structure-based nature allows it to accommodate variability within the tasks. Such generality is important since task objectives and constraints are ever changing during task execution, and a framework for accommodating such changes is essential.

ACKNOWLEDGMENTS

The research described in this paper was performed at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration.[2]

REFERENCES

[1] Dynamics and Real-Time Simulation (DARTS) Lab. http://dartslab.jpl.nasa.gov/, 2014.

[2] CRAIG, J. J. Introduction to Robotics. Addison-Wesley, Pub. Co., Reading, MA, 1986.

[3] GOSWAMI, A. Postural stability of biped robots and the foot rotation indicator (FRI) point. The International Journal of Robotics Research 18, 6 (1999), 523–533.

[4] JAIN, A. Robot and Multibody Dynamics: Analysis and Algorithms. Springer, 2011.

[5] JAIN, A., AND RODRIGUEZ, G. An analysis of the kinematics and dynamics of underactuated manipulators. IEEE Transactions on Robotics and Automation 9 (1993), 411–422.

[6] LUH, J. Y. S., WALKER, M. W., AND PAUL, R. P. On-line Computational Scheme for Mechanical Manipulators. ASME Journal of Dynamic Systems, Measurement, and Control 102, 2 (June 1980), 69–76.

[7] RIGHETTI, L., BUCHLI, J., MISTRY, M., KALAKRISHNAN, M., AND SCHAAL, S. Optimal distribution of contact forces with inverse dynamics control. The International Journal of Robotics Research 32, 3 (2013), 280–298.

[8] SAAB, L., RAMOS, O. E., MANSARD, N., SOU, P., AND FOURQUET, J.-Y. Dynamic Whole-Body Motion Generation under Rigid Contacts and other Unilateral Constraints. IEEE Transactions on Robotics 29, 2 (2013), 346–362.

[9] SENTIS, L., PETERSEN, J., AND PHILIPPSEN, R. Implementation and stability analysis of prioritized whole-body compliant controllers on a wheeled humanoid robot in uneven terrains. Autonomous Robots 35, 4 (Aug. 2013), 301–319.