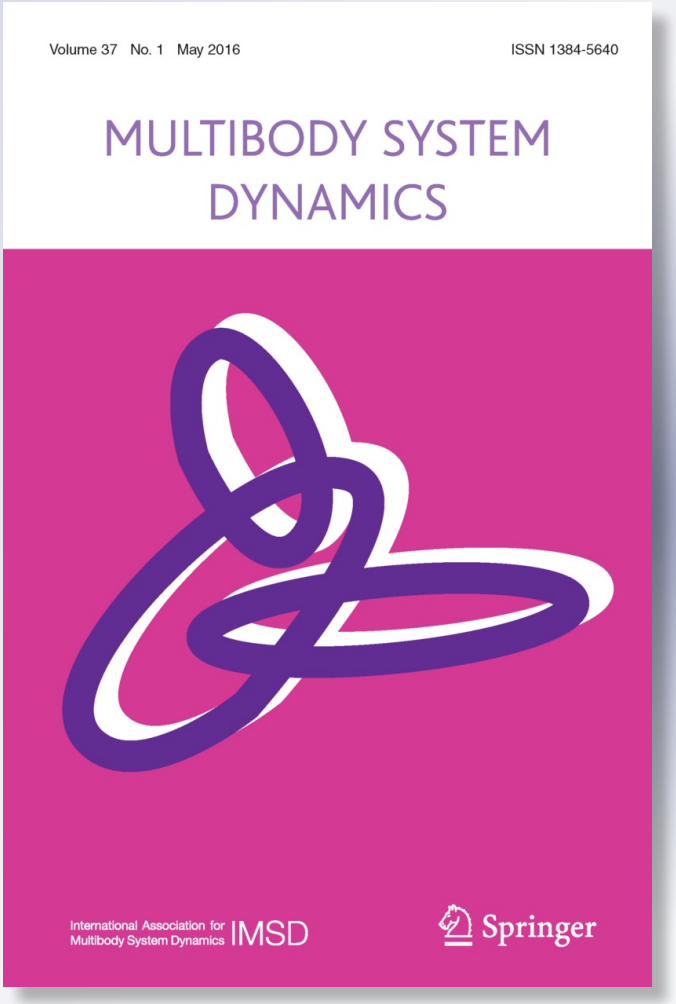
# *Feedforward dynamics for the control of articulated multi-limb robots*

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

**Multibody System Dynamics**

ISSN 1384-5640
Volume 37
Number 1

Multibody Syst Dyn (2016) 37:49-68
DOI 10.1007/s11044-016-9511-1




Springer

Multibody Syst Dyn (2016) 37:49–68
DOI 10.1007/s11044-016-9511-1

# Feedforward dynamics for the control of articulated multi-limb robots

**Abhinandan Jain[1] · Calvin Kuo[1] · Ivan Sinkarenko[1]**

Received: 27 February 2015 / Accepted: 19 February 2016 / Published online: 16 March 2016


**Abstract** We describe a general approach for using linearizing feedforward control inputs for large degree of freedom (dof) multi-limb robots operating in scenarios involving motion and force constraints, and under-actuated degrees of freedom 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.

**Keywords** Robotics · Dynamics · Control

## 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 degrees of freedom, mobility and manipulation control challenges for such robotic scenarios include: the highly nonlinear nature of the system dynamics; the under-actuated nature of the system (i.e., not all degrees of freedom are actuated); motion constraints on the system; the time varying nature of the constraints (e.g., 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

✉ A. Jain
jain@jpl.nasa.gov

[1] Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109, USA

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. Righetti et al. [10] describe an elegant extension of this feedforward control approach for mobile legged robots. The alternative whole body control approach [12] decomposes the control problem based on prioritized objectives using projected operational space techniques [7]. 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 of 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 [10], 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 [10] focuses on legged systems with under-actuated degrees of freedom 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 degrees of freedom which can arise from the robot degrees of freedom as well as the task object degrees of freedom. Moreover, we do not require passive degree of freedom 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 [10], 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 degrees of freedom 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 in 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 degrees of freedom, 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, gyroscopic and 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, e.g., the $\mathcal{J}_e^* \mathfrak{f}_{ext}$ terms from a known end-effector spatial force $\mathfrak{f}_{ext}$ where $\mathcal{J}_e$ denotes the end-effector Jacobian.[1] The general idea is that $\mathcal{C}(\theta, \dot{\theta})$ includes all the explicitly known, state-dependent terms that appear in the equations of motion.

[1] The "$*$" superscript denotes matrix transpose.

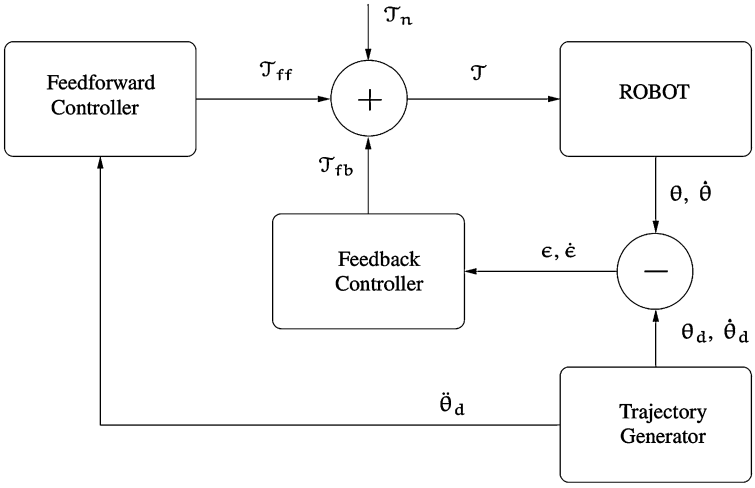



**Fig. 1** Block diagram illustrating the use of feedforward and feedback controllers for robot motion control

The motion and tracking control problem for robots requires the design of 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 nonlinear dynamical systems such as in Eq. (1). Linear feedback control techniques perform poorly due to the widely varying nonlinear dynamics of the system across the configuration space.

The *computed torque* method for robot control provides a solution to this problem [2]. As illustrated in Fig. 1, the basic idea is to use feedforward control in conjunction with feedback control to handle the nonlinearities across the large configuration space. The feedforward term's role is 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 degrees of freedom are actuated, the computed torque approach uses a feedforward torque $\mathcal{T}_{ff}$ of the form

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

The $\mathrm{TINV}(\mathrm{x},\ \dot{\theta}, \theta)$ function represents the standard inverse dynamics computation of generalized forces for a vector $\mathrm{x}$ of joint accelerations for a tree topology system. For brevity we will use $\mathrm{TINV}(\mathrm{x})$ instead of $\mathrm{TINV}(\mathrm{x},\ \dot{\theta}, \theta)$ with the current $(\dot{\theta}, \theta)$ state values being implied for the missing arguments. Low-cost Newton–Euler recursive algorithms to carry out this inverse dynamics computation are well known [9]. In addition, including any prescribed motion degrees of freedom into the formulation are straightforward. With tracking error $\epsilon \triangleq \theta_d - \theta$, a feedback term of the form $\mathcal{T}_{fb} = \mathcal{M}(\theta)[\mathrm{K}_v\dot{\epsilon} + \mathrm{K}_p\epsilon]$ is used to obtain the overall actuator torque of the form

$$\begin{aligned}\mathcal{T} &\triangleq \mathcal{T}_{ff} + \mathcal{T}_{fb} + \mathcal{T}_n = \mathcal{M}(\theta)[\ddot{\theta}_d + \mathrm{K}_v\dot{\epsilon} + \mathrm{K}_p\epsilon] + \mathcal{C}(\theta,\ \dot{\theta}) + \mathcal{T}_n \\ &= \mathrm{TINV}(\ddot{\theta}_d + \mathrm{K}_v\dot{\epsilon} + \mathrm{K}_p\epsilon) + \mathcal{T}_n.\end{aligned} \tag{3}$$

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

$$\ddot{e} + K_v \dot{e} + K_p e = -\mathcal{M}^{-1}(\theta)\mathcal{T}_n. \tag{4}$$

These error dynamics represent a linear, time-invariant system 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 of a perfect model and perfect sensing without noise, we have $e \equiv 0$, and the feedforward force $\mathcal{T}_{ff}$ by itself generates the desired motion for the robotic system. Thus in theory, feedforward control by itself can do the job and feedback control is not required. In practice, however, feedforward control is used in conjunction with feedback control, with the latter's role being to regulate residual errors that arise from modeling and sensing imperfections that are present in reality. A priori knowledge and on-line estimators are also often 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 lead to the exact desired generalized motion and forces for the ideal case of perfect knowledge of the model and system state. Keeping this and our focus on the feedforward term 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 degrees of freedom, and finally the general case of systems with both motion constraints and passive degrees of freedom.

### 2.1 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:[2]

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

where $n_c$ denotes the dimension of the constraints. Here $G(\theta, t) \in \mathcal{R}^{n_c \times \mathcal{N}}$ denotes the full row rank constraint matrix that defines the kinematic constraints of the form

$$G(\theta, t)\, \dot{\theta} = \mathcal{U}(t) \tag{6}$$

on the generalized velocity coordinates. Time differentiating this leads to the acceleration level constraint equation

$$G(\theta, t)\ddot{\theta} = \acute{\mathcal{U}}(t) \quad \text{where} \quad \acute{\mathcal{U}} \triangleq \dot{\mathcal{U}}(t) - \dot{G}\, \dot{\theta} \in \mathcal{R}^{n_c}. \tag{7}$$

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

The $-G^*(\theta, t)\lambda$ term in Eq. (5) represents the (implicitly defined) internal generalized constraint forces arising from the constraints.

Only desired accelerations $\ddot{\theta}$ that satisfy Eq. (7) are feasible for the constrained system, and we refer to such accelerations as being *consistent* with the constraints. For feedforward control, we need to compute the $\mathcal{T}$ generalized forces that can be applied to generate desired accelerations to drive the system along the desired trajectory. The following lemma provides the required expression.

**Lemma 1** *Desired generalized accelerations* $\ddot{\theta}_d$ *that are consistent with the constraints can be achieved by generalized forces of the form*

$$\mathcal{T} \triangleq \text{TINV}(\ddot{\theta}_d) + G^*\chi \overset{(2)}{=} \mathcal{M}(\theta)\ddot{\theta}_d + \mathcal{C}(\theta, \dot{\theta}) + G^*\chi \tag{8}$$

*for arbitrary* $\chi \in \mathcal{R}^{n_c}$. *Moreover, the Lagrange multiplier* $\lambda = -\chi$.

*Proof* Substituting Eq. (8) into the top rows of Eq. (5) leads to

$$\mathcal{M}(\theta)(\ddot{\theta} - \ddot{\theta}_d) = G^*(\lambda + \chi).$$

Choosing $\lambda = -\chi$ together with the invertibility of $\mathcal{M}$ implies that $\ddot{\theta} = \ddot{\theta}_d$, i.e., the $\mathcal{T}$ defined by Eq. (8) induces the desired accelerations in the system. The bottom row of Eq. (5) is satisfied because of the assumption that $\ddot{\theta}_d$ are consistent generalized accelerations. This lemma provides the expression needed for computing the feedforward generalized force needed to achieve desired system motion consistent with the constraints. With $\chi = 0$, even $\mathcal{T} = \text{TINV}(\ddot{\theta}_d)$ satisfies Eq. (5) with $\lambda = 0$. The Lagrange multipliers $\lambda$ serve as a free parameter for the feedforward $\mathcal{T}$—in that all choices for $\lambda$ lead to the $\ddot{\theta}_d$ desired accelerations. In other words, the specific choice of $\lambda$ has no affect on the motion of the system. This allows us to choose the $\lambda$ value to control the internal *squeeze* forces within the system. This decoupling allows us to choose $\text{TINV}(\ddot{\theta}_d)$ part of the feedforward control to generate the desired motion, and to choose $\lambda$ to meet additional system objectives such as the load-balancing of generalized forces across the joints.

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

$$\mathcal{T} = \text{TINV}(\ddot{\theta}_d) + G^*\lambda \tag{9}$$

with $\lambda$ either zero, or chosen to optimize some function of the internal forces. The loss of motion degrees of freedom from the constraints is compensated by the ability to control the same number of degrees of freedom in the force domain through a choice of $\lambda$. A possible choice for $\lambda$ is one that minimizes the weighted norm $\mathcal{T}^*W\mathcal{T}$ for some symmetric positive definite $W$ weighting matrix. For this, it is easy to verify that the minimum norm is obtained for

$$\lambda = \left[\left(G^*WG\right)^{-1}G\right]\text{TINV}(\ddot{\theta}_d). \qquad \square$$

### 2.1.1 Partitioned control coordinates

An important requirement for using Eq. (9) to generate the feedforward accelerations is that the desired $\ddot{\theta}$ 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

$$G\ddot{\theta} = [G_r,\ G_d] \begin{bmatrix} \ddot{\theta}_r \\ \ddot{\theta}_d \end{bmatrix} = \acute{\mathcal{U}} \quad \Longrightarrow \quad \ddot{\theta}_d = G_d^{-1}[\acute{\mathcal{U}} - G_r \ddot{\theta}_r]. \tag{10}$$

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. (10) is used to parametrize the subspace of generalized accelerations that are consistent with the constraints. Thus, we can use $\ddot{\theta}_r$ as the minimal, independent generalized acceleration coordinates for the system. Using Eq. (10), we have

$$\ddot{\theta} = \underline{\ddot{\theta}}_q + X\ddot{\theta}_r \quad \text{where} \quad \underline{\ddot{\theta}}_q \triangleq \begin{bmatrix} 0 \\ G_d^{-1}\acute{\mathcal{U}} \end{bmatrix} \quad \text{and} \quad X \triangleq \begin{bmatrix} I \\ -G_d^{-1}G_r \end{bmatrix} \in \mathcal{R}^{\mathcal{N} \times \mathcal{N}-n_c}. \tag{11}$$

Equation (11) gives us a way to recover the full generalized acceleration $\ddot{\theta}$ vector given the independent generalized acceleration vector $\ddot{\theta}_r$. Note that $GX = \mathbf{0}$.

### 2.2 Feedforward with passive dofs

For many robotic systems, not all degrees of freedom 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 for whom the torso degrees of freedom are passive. Passive degrees of freedom may also arise from the task object and environment, such as doors, door handles, etc. The characteristic of passive degrees of freedom 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 degrees of freedom by $n_p$, we partition the degrees of freedom into *active* dofs $\theta_a \in \mathcal{R}^{\mathcal{N}-n_p}$, and *passive* degrees of freedom $\theta_p \in \mathcal{R}^{n_p}$. Equation (1) can be rewritten in the following partitioned form based on the active and passive degrees of freedom:

$$\begin{pmatrix} \mathcal{M}_{aa} & \mathcal{M}_{ap} \\ \mathcal{M}_{ap}^* & \mathcal{M}_{pp} \end{pmatrix} \begin{bmatrix} \ddot{\theta}_a \\ \ddot{\theta}_p \end{bmatrix} + \begin{bmatrix} \mathcal{C}_a \\ \mathcal{C}_p \end{bmatrix} = \begin{bmatrix} \mathcal{T}_a \\ \mathcal{T}_p \end{bmatrix}. \tag{12}$$

The (unactuated) passive generalized forces $\mathcal{T}_p$ are a known function of the system state $(\theta,\ \hat{\theta})$. The passive degrees of freedom represent constraints in the generalized force space, and are a dual to the kinematic constraints discussed earlier. $\mathcal{T}_a$ represents the feedforward term for the under-actuated system since actuators can be commanded to only set this subset of $\mathcal{T}$. The form of Eq. (12) is not convenient to determine $\mathcal{T}_a$ because of its implicit form, i.e., it contains a mix of the known and unknown quantities on both the left and right hand sides of the equation. The following lemma provides an alternative expression where all the unknown quantities are on the left hand side and can be used to evaluate $\mathcal{T}_a$.

**Lemma 2** *Given a system with no closure constraints, a passive system with $\mathcal{T}_p$ passive generalized forces, desired $\ddot{\theta}_a$ active generalized acceleration can be obtained by applying the $\mathcal{T}_a$ defined by the following expression*:

$$\begin{bmatrix} \mathcal{T}_a \\ \ddot{\theta}_p \end{bmatrix} = \begin{pmatrix} \mathcal{S}_{aa} & \mathcal{S}_{ap} \\ -\mathcal{S}_{ap}^* & \mathcal{S}_{pp} \end{pmatrix} \begin{bmatrix} \ddot{\theta}_a \\ \mathcal{T}_p \end{bmatrix} + \begin{bmatrix} \mathcal{C}_a - \mathcal{S}_{ap}\mathcal{C}_p \\ -\mathcal{S}_{pp}\mathcal{C}_p \end{bmatrix} \tag{13a}$$

*where*

$$\mathcal{S}_{aa} \triangleq \mathcal{M}_{aa} - \mathcal{M}_{ap}\mathcal{M}_{pp}^{-1}\mathcal{M}_{ap}^{*},\ \mathcal{S}_{ap} \triangleq \mathcal{M}_{ap}\mathcal{M}_{pp}^{-1},\ \mathcal{S}_{pp} \triangleq \mathcal{M}_{pp}^{-1}. \tag{13b}$$

$\ddot{\theta}_p$ *denotes the generalized accelerations that are induced at the passive degrees of freedom.*

*Proof* Equation (13a), (13b) is obtained by a straightforward matrix transformation of Eq. (12) [5, 6]. For the desired $\ddot{\theta}_a$ generalized acceleration at the active hinges, Eq. (13a) provides us with a way to evaluate the necessary feedforward $\mathcal{T}_a$ active hinge forces as well as 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 while induced nonzero $\ddot{\theta}_p$ passive accelerations are a (potentially undesirable) side-effect 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. (13b) are discussed in [5, 6]. In the next section we will see that the presence of constraints can often allow us to even control the passive accelerations. □

### 2.3 Feedforward with both constraints and passive dofs

More generally, robotic systems have both motion constraints as well as passive degrees of freedom. Prominent examples of such systems are once again wheeled and legged mobile robots. For these systems, the torso and chassis degrees of freedom are passive, while their motion is constrained by the contact between the wheels/feet and the ground. Since the system has both passive degrees of freedom as well as constraints, the techniques described in Sects. 2.1 and 2.2 cannot be used directly. For a desired $\ddot{\theta}$ (consistent with the motion constraints), we partition the feedforward expression in Eq. (9) between the passive and active degrees of freedom as follows:

$$G^{*} = \begin{bmatrix} G_a^{*} \\ G_p^{*} \end{bmatrix} \lambda \quad \text{and} \quad \mathcal{T}^{f}(\ddot{\theta}) = \begin{bmatrix} \mathcal{T}_a^{f}(\ddot{\theta}) \\ \mathcal{T}_p^{f}(\ddot{\theta}) \end{bmatrix} \triangleq \text{TINV}(\ddot{\theta}). \tag{14}$$

The $G_a^{*} \in \mathcal{R}^{(\mathcal{N}-n_p)\times n_c}$ and $G_p^{*} \in \mathcal{R}^{n_p \times n_c}$ matrices represent a partitioning of $G^{*}$ based on the active and passive degrees of freedom in the system.

The presence of constraints in a passive system turns out to be a mixed blessing. While it is clear from our earlier discussion that arbitrary desired motions are not possible due to the presence of passive degrees of freedom, it turns out consistent generalized accelerations are often achievable due to the presence of the constraints. This is the subject of the following lemma.

**Lemma 3** *The active generalized forces* $\mathcal{T}_a$ *needed to achieve desired consistent generalized accelerations* $\ddot{\theta}$ *is given by*

$$\mathcal{T}_a = \mathcal{T}_a^{f}(\ddot{\theta}) - G_a^{*}\lambda \tag{15}$$

*where* $\lambda$ *is a solution for the following equation*:

$$G_p^{*}\lambda = \mathcal{T}_p - \mathcal{T}_p^{f}(\ddot{\theta}). \tag{16}$$

*Proof* From Eq. (9), the general solution for the feedforward term for a constrained system in partitioned form is given by

$$\begin{bmatrix} \mathcal{T}_a \\ \mathcal{T}_p \end{bmatrix} = \begin{bmatrix} \mathcal{T}_a^f(\ddot{\theta}) \\ \mathcal{T}_p^f(\ddot{\theta}) \end{bmatrix} - \begin{bmatrix} G_a^* \\ G_p^* \end{bmatrix} \lambda. \tag{17}$$

The problem with using Eq. (17) is that $\mathcal{T}_p$ is defined a priori for the passive hinges and cannot be changed. The solution is to use the lower row as shown in Eq. (16) to solve for $\lambda$ that is consistent with the passive degrees of freedom. Using this solution in the upper row (as shown in Eq. (15)) provides the desired active generalized forces $\mathcal{T}_a$. Using this lemma, $\mathcal{T}^f$ can be readily evaluated given the consistent $\ddot{\theta}$ desired accelerations.

A solution for Eq. (16) will exist when there are enough constraints, i.e., when $n_c \geqslant n_p$ and $G_p^*$ has full row rank. Assuming a solution for $\lambda$ 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. (14). 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 degrees of freedom, motion constraints can generate constraint forces that help to fill in for the missing actuation forces for the passive degrees of freedom. In effect, the motion constraints provide a way to introduce actuators that are otherwise missing for the passive degrees of freedom.

When there are not enough constraints, i.e., $n_c < n_p$, a solution for Eq. (16) is not guaranteed. In this case not all accelerations consistent with the constraints are achievable. For this case we have

$$\begin{aligned} \mathcal{T} &\overset{(5)}{=} \mathcal{M}(\theta)(\underline{\ddot{\theta}}_q + X\ddot{\theta}_r) + \mathcal{C}(\theta, \dot{\theta}) - G^*(\theta, t)\lambda \\ &\overset{(2),(14)}{=} \mathcal{M}(\theta)X\ddot{\theta}_r + \mathcal{T}^f(\underline{\ddot{\theta}}_q) - G^*\lambda. \end{aligned} \tag{18}$$

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

$$\begin{aligned} \mathcal{T}_p &\overset{(12),(14)}{=} \left[\mathcal{M}_{ap}^*, \mathcal{M}_{pp}\right] X\ddot{\theta}_r + \mathcal{T}_p^f(\underline{\ddot{\theta}}_q) - G_p^*\lambda \\ \implies & \left(\left[\mathcal{M}_{ap}^*, \mathcal{M}_{pp}\right] X, -G_p^*\right) \begin{bmatrix} \ddot{\theta}_r \\ \lambda \end{bmatrix} = \mathcal{T}_p - \mathcal{T}_p^f(\underline{\ddot{\theta}}_q). \end{aligned} \tag{19}$$

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. (11), followed by using Eq. (15) to evaluate the $\mathcal{T}_a$ feedforward term. In summary, the steps involved 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. (11) 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. (12), extract the $\mathcal{T}_p^f$ sub-vector from this based on the passive degrees of freedom.
3. If full row rank conditions hold, solve $G_p^*\lambda = \mathcal{T}_p - \mathcal{T}_p^f$ from Eq. (16) for $\lambda$. Else, use Eq. (19) to solve for $(\ddot{\theta}_r, \lambda)$. When there are multiple solutions, pick the solution that optimizes other task objectives.

4. Evaluate $\mathcal{T}_a = \mathcal{T}_a^f - G_a^* \lambda$ from Eq. (15) 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 degrees of freedom primarily effect steps 3 and 4. While the passive degrees of freedom are defined by the physics of the robot and the scenario, the independent generalized accelerations degrees of freedom are chosen based on motion planning convenience. There is no requirement that they be the same or have any overlap. Equation (16) defines the connection between them since $\mathcal{T}_p^f$ is determined by the passive degrees of freedom, while G is determined by the constraints. When there are no passive degrees of freedom, $n_p = 0$, $\lambda$ can be arbitrary. When there are no constraints, $n_c = 0$, and only $\ddot{\theta}$ satisfying Eq. (19) can be exactly met. When there are neither constraints nor passive degrees of freedom, we reduce to the standard computed torque feedforward term. Each of the steps is low cost. The cost of the optional $\lambda$ 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 are defined via a constraint matrix $\mathcal{Q} \in \mathcal{R}^{n_c \times 6n_b}$ such that $\mathcal{Q}\mathcal{V}_b = \mathcal{U}(t)$. With $\mathcal{J}_b \in \mathcal{R}^{6n_b \times \mathcal{N}}$ denoting the Jacobian matrix for these nodes

$$\begin{aligned} &\mathcal{V}_b = \mathcal{J}_b\,\dot{\theta} \quad \Rightarrow \quad \mathcal{Q}\mathcal{J}_b\,\dot{\theta} = \mathcal{U}(t) \quad \Rightarrow \quad G = \mathcal{Q}\mathcal{J}_b \quad \text{and} \\ &G_p = \mathcal{Q}\mathcal{J}_p \quad \text{where } \mathcal{J}_b = [\mathcal{J}_a,\ \mathcal{J}_p]. \end{aligned} \tag{20}$$

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 degrees of freedom. □

### 2.4 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. While unilateral constraints do behave like bilateral constraints when they are active, 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 [10], we use the strategy of treating the unilateral constraints as bilateral constraints for the feedforward evaluation, but add a follow up check to verify that the feedforward solution satisfies the conditions necessary for the unilateral constraints to be active. For contact constraints, the condition for them to be active typically requires the solution contact forces to lie within the friction cone to satisfy the no slip condition. While expedient, a pitfall of this approach is that satisfaction of such consistency requirements cannot be guaranteed. We refer the reader to [10] for a more detailed discussion of this topic and useful techniques for helping meet the consistency requirement. An expensive, but rigorous alternative option is to append the inequality conditions for the unilateral constraints to Eq. (16), and use the more expensive quadratic programming like techniques [11] 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 that 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, ensuring end-effector forces do not damage the task object, using grasps to apply sufficient force to avoid object slippage, and avoiding overloading and saturation of actuators.

Let $\mathfrak{m}$ denote the number of arms grasping the task object. For simplicity we focus on the dual-arm/rigid grasp case, i.e., $\mathfrak{m} = 2$, 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.

### 3.1 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 degrees of freedom, 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 degrees of freedom consist of the joint degrees of freedom for each of the arms together with the 6 passive degrees of freedom for the task object. Thus for the two arm and rigid grasp case

$$\mathfrak{n}_\mathfrak{p} = 6, \quad \mathfrak{n}_\mathfrak{b} = 4, \quad \mathfrak{n}_\mathfrak{c} = 6\mathfrak{n}_\mathfrak{b}/2 = 12 \quad \text{and} \quad \mathcal{T}_\mathfrak{p} = \mathbf{0}. \tag{21}$$

$\mathcal{J}_\mathfrak{b}$ and $\mathcal{Q}$ are given by

$$\mathcal{J}_\mathfrak{b} = \left(\begin{array}{cc|c} \mathcal{J}_\mathfrak{r} & 0 & 0 \\ 0 & \mathcal{J}_\mathfrak{l} & 0 \\ 0 & 0 & \mathcal{J}_{\mathfrak{tr}} \\ 0 & 0 & \mathcal{J}_{\mathfrak{tl}} \end{array}\right) \in \mathcal{R}^{6\mathfrak{n}_\mathfrak{b} \times \mathcal{N}}, \qquad \mathcal{Q} = \begin{pmatrix} \mathrm{I} & 0 & -\mathrm{I} & 0 \\ 0 & \mathrm{I} & 0 & -\mathrm{I} \end{pmatrix} \in \mathcal{R}^{\mathfrak{n}_\mathfrak{c} \times 6\mathfrak{n}_\mathfrak{b}}. \tag{22}$$

$\mathcal{J}_\mathfrak{r}$ and $\mathcal{J}_\mathfrak{l}$ denote the end-effector Jacobians for the right and left arms, respectively, and $\mathcal{J}_{\mathfrak{tr}}$ and $\mathcal{J}_{\mathfrak{tl}}$ denote the task object Jacobians for the right and left attachment points, respectively. For a rigid task object, $\mathcal{J}_{\mathfrak{tr}} = \phi_\mathfrak{r}^*$ and $\mathcal{J}_{\mathfrak{tl}} = \phi_\mathfrak{l}^*$, where the $\phi_\mathfrak{r}$ and $\phi_\mathfrak{l}$ $6 \times 6$ matrices denote the rigid body transformation matrices from the right and left attachment points to the task object's reference frame. The $\mathcal{J}_\mathfrak{p} \in \mathcal{R}^{6\mathfrak{n}_\mathfrak{b} \times \mathfrak{n}_\mathfrak{p}}$ passive Jacobian from Eq. (20) corresponds to the block column for the passive degrees of freedom in the right part of the $\mathcal{J}_\mathfrak{b}$ matrix in Eq. (22). Note that we never need to compute the full $\mathcal{J}_\mathfrak{b}$ matrix, just the much smaller $\mathcal{J}_\mathfrak{p}$ matrix. Since $\mathfrak{n}_\mathfrak{c} \geqslant \mathfrak{n}_\mathfrak{p}$ is satisfied, Eq. (15) holds, and we obtain the feedforward condition

$$\mathcal{T}_\mathfrak{p}^\mathfrak{f} - \mathcal{T}_\mathfrak{p} \overset{(16),(21)}{=} -\mathcal{J}_\mathfrak{p}^* \mathcal{Q}^* \lambda \overset{(22)}{=} -[\mathcal{J}_{\mathfrak{tr}}, \quad \mathcal{J}_{\mathfrak{tl}}]\lambda. \tag{23}$$

Note that $\mathcal{T}_\mathfrak{p}^\mathfrak{f}$ represents the D'Alembert spatial force needed at the task object's reference frame to move it along the desired motion trajectory. The $\lambda$ solutions to Eq. (23) are the end-effector forces that can provide this D'Alembert force, and can be used to evaluate the $\mathcal{T}_\mathfrak{a}$ active feedforward forces using Eq. (15).

### 3.2 Static case

For single or multiple arms statically holding a heavy object, the feedforward term reduces to the joint torques needed to compensate for the gravitational load from the arms and 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 (e.g., to avoid damaging the object, preventing slippage), or avoiding actuator saturation can be selected.

### 3.3 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 nonzero 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, namely *explicit* and *implicit*, are available for extending our approach to this situation and are described below.

#### *3.3.1 Explicit constraints*

The explicit option treats the task object as a floating object and appends the task object constraints to the overall list of constraints. This option is more general and explicitly tracks the task constraints. The explicit option may be preferable when the task object constraint is a unilateral constraint, (e.g., when polishing, a positive force is required to maintain contact) since it is easier to enforce such an inequality condition when solving for $\lambda$.

For the explicit option, both $n_b$ and $n_c$ increase due to additional task object constraints, 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. (22).

#### *3.3.2 Implicit constraints*

The implicit option treats the task object constraint as a hinge and the task object as a branched articulated system. For this option, $n_b$ and $n_c$ remain unchanged, but $n_p$ decreases to the number of degrees of freedom available to the task object from its "constraint" hinge. In Eq. (22), 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. Due to its lower dimensional impact, this approach is preferable to the explicit option, but can be used only if the task object has a branched structure.

For both explicit and implicit options, the left half of Eq. (23) continues to hold, and solutions to it can be used to compute the $\mathcal{T}_a$ feedforward values.

### 3.4 Articulated task object

As a further generalization, consider the case where the task object has internal articulation degrees of freedom. 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 degrees of freedom in the task object.

As long as the condition $n_c \geqslant n_p$ continues to hold, we can use Eq. (23) to determine the feedforward term 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. For this case, the achievable motions are governed by the more restrictive and complex condition in Eq. (19)—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 degrees of freedom 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 so that there is no slippage or loss of contact.

### 4.1 Point contact

Assuming *point contact* between the feet and the ground, we have

$$n_p = 6, \quad n_b = m, \quad n_c = 3n_b \quad \text{and} \quad \mathcal{T}_p = \mathbf{0}. \tag{24}$$

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

$$\begin{aligned} \mathcal{J}_b &= \left(\begin{array}{cccc|c} \mathcal{J}_1 & 0 & \cdots & \cdots & \mathcal{J}_{b1} \\ 0 & \mathcal{J}_2 & \cdots & \cdots & \mathcal{J}_{b2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathcal{J}_l & \mathcal{J}_{bm} \end{array}\right), \\ \mathcal{Q} &= \begin{pmatrix} [0_3,\ I_3] & 0 & \cdots & 0 \\ 0 & [0_3,\ I_3] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & [0_3,\ I_3] \end{pmatrix} \in \mathcal{R}^{n_c \times 6n_b} \end{aligned} \tag{25}$$

$\mathcal{J}_i$ denotes the Jacobian for the $i$th leg to its foot, and $\mathcal{J}_{bi}$ denotes the Jacobian from the torso degrees of freedom to 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. (15) to be guaranteed to exist.

### 4.2 Area contact

For biped robots, 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. (25) 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 degrees of freedom which are used for motion planning. The kinematic constraints are used to compute the dependent leg degree of freedom accelerations that effect the robot's

posture. The $\lambda$ 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 $\lambda$ 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 $\lambda$ solution chosen to manage the ground contact forces and the loads on the leg actuators.

### 4.3 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 pure twist condition on the COP point $C$ takes the form

$$N(B) + \tilde{l}(C,B)F(B) = \begin{bmatrix} 0 \\ 0 \\ r \end{bmatrix} \tag{26}$$

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 $r$ is the moment on the torso about the normal. The 0 values on the right side for the $x$ and $y$ components of the torso moment correspond to the pure twist requirement. 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

$$N_x(B) = -l_z F_y(B) + l_y F_z(B) \quad \text{and} \quad N_y(B) = l_z F_x(B) - l_x F_z(B) \tag{27}$$

for $l_x$ and $l_y$. 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 for legged robots is to use an inverted pendulum model as a basis for the $\mathcal{T}_p^f \approx M(B)\ddot{\theta}(B) + \mathcal{C}(\theta, 0)$ 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 just the torso. The advantage of this approximation is that the torso's spatial inertia is constant and the full 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. Thus only the torso's D'Alembert force and gravitational forces are included. These assumptions are entirely optional and only used to reduce the computational burden.

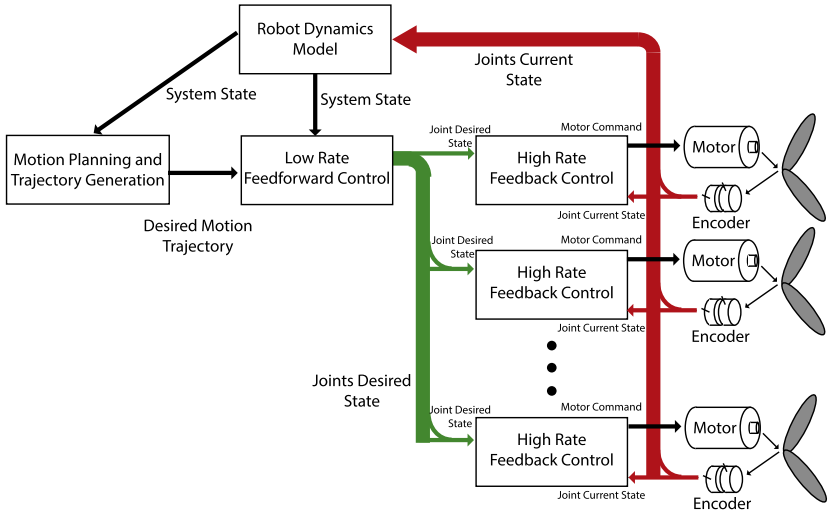



**Fig. 2** Block diagram of the control architecture utilizing feedforward control

Note that our feedforward approach is valid even when the ground plane is not level, since the pure twist requirement in Eq. (27) 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 degrees of freedom and arms. During manipulation, any additional constraints on the arms can be handled as described in Sect. 3 and used to augment the constraint Jacobian and the other matrices. Again, generalizing to multiple robots performing a task in coordination is straightforward.

### 4.4 Wheeled robots

The feedforward terms for wheeled platforms with arms can also be evaluated by our change, especially where the mobility and manipulation degrees of freedom 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 $\mathcal{Q}$ in Eq. (25). 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 shown in Fig. 2 to illustrate the use of our generalized feedforward control in simulation. The control architecture contains modules responsible for motion planning and trajectory generation, feedforward control, and feedback control. The task scenario defines the passive degrees of freedom and constraints on the robot. This information is used by the motion planner to design trajectories for all the robot joints that are consistent with the constraints for the task. This module is also responsible for defining the passive degrees of freedom and active constraints during the execution of the trajectory since these can change during the task execution.

While the motion planning's algorithms are tailored to generate motion trajectories specific to each scenario and activity type, our feedforward module is generic and remains unchanged across widely varying tasks. The feedforward module only requires the current state of the robot, the active constraints, the passive degrees of freedom, and desired joint accelerations at any time instant to compute the feedforward generalized force needed for all the *active* joints. The feedforward control computation uses the full dynamics model of the system for the generalized feedforward compensation techniques described in this paper.

While the feedforward control module takes into account the system level state and desired motion of the robot system, the feedback control module consists of a bank of decoupled, low-level proportional derivative feedback controllers for each of the actuated joints. The gains for each of these joint controllers are set as needed for tracking the desired trajectory for the joint.

We demonstrate the versatility of our feedforward formulation by using this control architecture to execute a variety of robot tasks in simulation. The robot tasks we chose are from the Defense Advanced Research Projects Agency Robotics Challenge (DRC) competition and 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. Not only do these tasks require the control of challenging robot and task dynamics, but also the execution of a variety of different activities including legged mobility, single and multi-arm manipulation, combined mobility and manipulation, and manipulation of constrained objects.

The robot used for simulating the execution of these tasks is based on the Jet Propulsion Laboratory's (JPL) RoboSimian robot [4] shown in Fig. 3. This robot has four 7 degree of freedom limbs, with each limb having a hand with three fingers. Overall the robot has 70 degrees of freedom. The dexterous limbs are designed to be used as legs for walking, or as manipulators as needed by the task.

The following sections provide the description of the use of the constraints and passive degree of freedom models used by the feedforward module for each of these simulated tasks. While the limb joints have actuators, the 6 torso degrees of freedom are unactuated and represent passive degrees of freedom that are taken into account by the feedforward module for each of the tasks. The generalized forces associated with the torso's passive degrees of freedom are zero. As we will see, some of the tasks introduce additional passive degrees of freedom that are also handled by the feedforward module.

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 [5]. While unilateral ground contact constraints for the stance legs were modeled as bilateral constraints for the feedforward control computation, the simulation model made no such assumption and treated them as unilateral constraints. Despite these substantial differences between the feedforward and simulation models, the feedforward approach successfully executed all of these tasks in simulation.

### 5.1 Driving task

The first task (a) we describe is the vehicle driving task illustrated in Fig. 3. This task requires the robot to turn the steering wheel of a vehicle while seated. For the feedforward computation, we treat the robot torso as being fixed to the seat via a constraint. The limbs below the vehicle's dash are free to move and perform actions such as gas pedal depression. The limbs above the dash alternate between free space motion to approach and grasp the steering wheel, and constrained motion when turning the steering wheel. The steering wheel

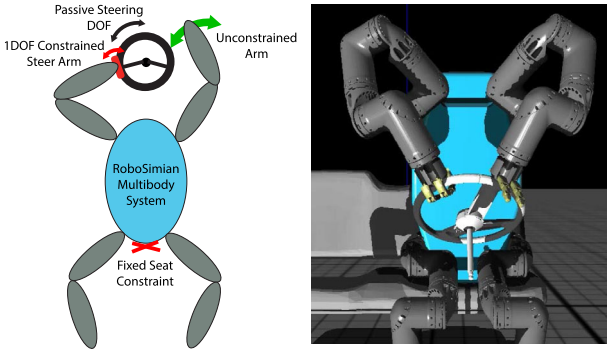



**Fig. 3** The passive degrees of freedom and constraints for the driving task

is modeled as a passive single degree of freedom joint that can rotate about its axis with a nonzero resistance force that needs to be applied to turn it. This is an example of a task that introduces passive degrees of freedom in addition to the 6 passive degrees of freedom from the robot's torso. Moreover, the generalized force associated with the steering wheel passive degree of freedom is nonzero.

The motion planning module generates the required joint trajectories for the limbs for the free space motion to approach the steering wheel. Once a limb grasps the steering wheel, the limb is modeled as having a constraint between the limb and the steering rim that allows only 1 degree of freedom rotation of the hand about the rim. The input to the motion planning module is the desired motion trajectory of the steering wheel, which is used to compute the desired motion of the joints in the limb grasping the rim. The passive degrees of freedom and constraints are different between the approach to grasp phase, and the steering wheel turn phases. The motion planning module monitors and updates the passive degrees of freedom and constraints data during the task execution so that the feedforward module uses the correct information at each time instant.

### 5.2 Debris clearing task

The next simulated task (b) is the debris clearing task. This task consists of the robot moving a heavy object in free space using one of its limbs while the other 3 limbs provide support for the robot. For feedforward computation, the robot's three stance limbs are constrained to the ground with revolute joints where the single degree of freedom is the rotation about an axis normal to the ground plane. The manipulator limb is free to move in space with the debris object fixed to the end effector. This task requires knowledge of the debris' mass in order to properly compensate for its load during free space motion.

Task (f) requires using a saw to cut a hole in the wall. Its feedforward computation is similar to the debris clearing task in that a known force needs to be applied by the free arm end effector on the saw. The hose mating task (h) is also similar to the debris clearing task where the hose replaces the debris in the manipulation arm for feedforward control purposes.

### 5.3 Ladder climbing task

The ladder climbing task (c) shown in Fig. 4 requires the robot to climb a ladder with rungs. This task is a mobility task where at least 3 of the robot's limbs are constrained to a fixed object (such as the ladder in this case) for stability. The supporting limbs are said to be in stance, while the remaining limb, that can move in free space, is referred as the swing limb.

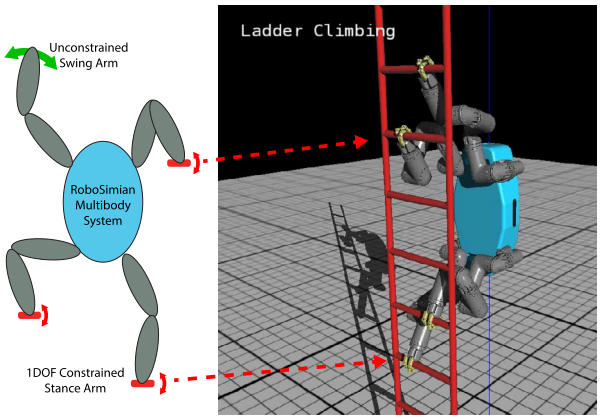



**Fig. 4** The passive degrees of freedom and constraints for the ladder climbing task

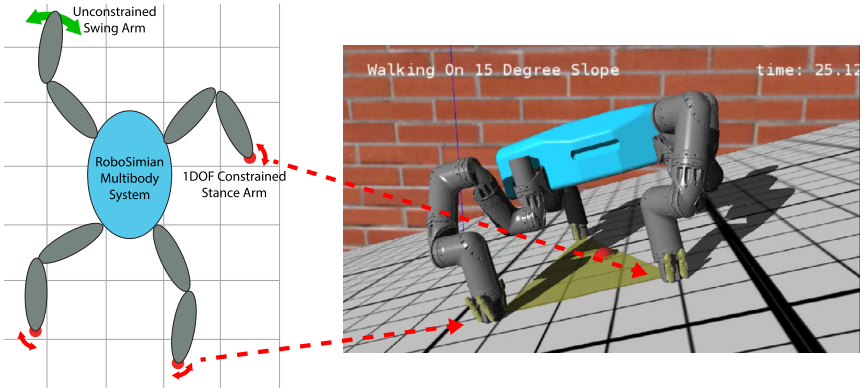



**Fig. 5** The passive degrees of freedom and constraints for the walking task.

The limbs alternate between stance and swing states in order to propel the robot. For the ladder task feedforward computation, we represent constraints between the robot and ladder as revolute joint constraints where the single degree of freedom is rotation about the ladder rung axis. As the robot climbs the ladder, the limbs in sequence release their rung and move in free space to another rung on the ladder. As in the case of the driving task, the constraints change during the execution of the climbing task.

### 5.4 Walking task

Task (d) requires the robot to walk across terrain. This mobility task works in much the same way as ladder climbing and is illustrated in Fig. 5. For feedforward control, the most important difference between the two tasks is the way the stance limb constraints are represented. Both use revolute joint constraints, however, while the ladder constraint allows rotation about the rung axis, the walking constraint allows rotation about the axis normal to the ground plane. Otherwise, like the ladder climb, walking is achieved by having limbs alternately release their ground constraints and move in free space to the next limb location.

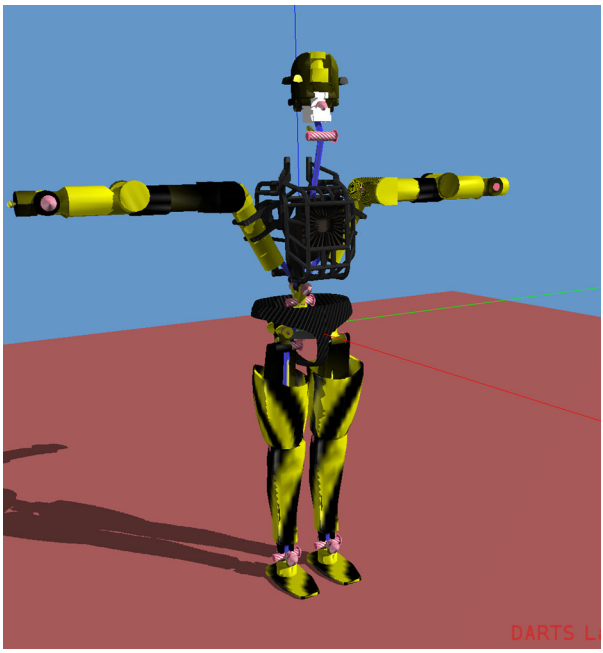



**Fig. 6** The Atlas biped robot

### 5.5 Door opening task

Task (e) requires opening a door. For the feedforward control the door revolute degree of freedom represents an additional passive degree of freedom. For feedforward control, the robot is constrained to the ground with three stance limbs in the same manner as the debris removal task. However, the remaining limb is not free but instead is fixed to the door handle, thus forming a closed chain with the rest of the robot and ground. The motion trajectory of the door hinge is used to determine the constrained motion of this limb.

### 5.6 Valve turning task

The valve turning task (g) is another manipulation task that can be done using a single arm similar to opening a door. However, to demonstrate the capabilities of the feedforward scheme for coordinated control, we use two limbs instead of just one to turn the valve—especially since a significant torque is required to overcome the valve's resistance. For the feedforward model, two stance limbs are constrained to the ground in a similar fashion as for opening the door. The valve's axis of rotation represents an additional passive degree of freedom. The two manipulation limbs grasping the valve rim contribute two additional constraints. The two manipulation limbs form a closed chain with the valve and the robot torso and thus a coordinated effort from both arms is required to turn the valve. The motion trajectory of the valve angle is used to generate the constrained joint trajectories for the manipulation limbs.

### 5.7 Bipedal walking

As a final note, we also used the generalized feedforward control architecture to simulate walking for a Boston Dynamics 36-dof Atlas biped robot [8] shown in Fig. 6. 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 leg 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 ground contact constraints were defined as fixed, area constraints instead of revolute constraints in order to account for the larger surface area of the Boston Dynamics Atlas feet.

## 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 degrees of freedom. The feedforward evaluation procedure includes at its heart the well known computed torque feedforward evaluation for unconstrained systems, and provides a general purpose approach to include the motion constraints to define admissible motions, and to select feedforward solutions that are compatible with the passive degrees of freedom. 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 versatile and general, but has reduced computational cost for embedded control use. The computational benefits arise from the exploitation of the low cost inverse dynamics algorithms for $\mathrm{TINV}(x, \dot{\theta}, \theta)$ for the feedforward computations, and avoiding the need for more expensive quantities such as the mass matrix and the operational space matrix found in other approaches. 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 degrees of freedom 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.

**Acknowledgements** 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.[3] The authors declare that they have no conflict of interest.

## References

1. Dynamics and Real-Time Simulation (DARTS) Lab (2014). http://dartslab.jpl.nasa.gov
2. Craig, J.J.: Introduction to Robotics. Addison-Wesley, Pub. Co., Reading (1986)
3. Goswami, A.: Postural stability of biped robots and the foot rotation indicator (FRI) point. Int. J. Rehabil. Res. **18**(6), 523–533 (1999)
4. Hebert, P., Bajracharya, M., Ma, J., Hudson, N., Aydemir, A., Reid, J., Bergh, C., Borders, J., Frost, M., Hagman, M., Leichty, J., Backes, P., Kennedy, B., Karplus, P., Satzinger, B., Byl, K., Shankar, K., Burdick, J.: Mobile manipulation and mobility as manipulation-design and algorithms of RoboSimian. J. Field Robot. **32**(2), 255–274 (2015). http://doi.wiley.com/10.1002/rob.21566. doi:10.1002/rob.21566

[3] 

5. Jain, A.: Robot and Multibody Dynamics: Analysis and Algorithms. Springer, Berlin (2011). doi:10.1007/978-1-4419-7267-5. http://www.springerlink.com/content/978-1-4419-7266-8/contents/
6. Jain, A., Rodriguez, G.: An analysis of the kinematics and dynamics of underactuated manipulators. IEEE Trans. Robot. Autom. **9**, 411–422 (1993). doi:10.1109/70.246052. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=246052
7. Khatib, O.: A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE J. Robot. Autom. **RA–3**(1), 43–53 (1987)
8. Kuindersma, S., Deits, R., Andr, M.F., Dai, H., Permenter, F., Pat, K., Russ, M.: Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. Auton. Robots (2015). http://dx.doi.org/10.1007/s10514-015-9479-3. doi:10.1007/s10514-015-9479-3
9. Luh, J.Y.S., Walker, M.W., Paul, R.P.: On-line computational scheme for mechanical manipulators. J. Dyn. Syst. Meas. Control **102**(2), 69–76 (1980)
10. Righetti, L., Buchli, J., Mistry, M., Kalakrishnan, M., Schaal, S.: Optimal distribution of contact forces with inverse dynamics control. Int. J. Robot. Res. **32**(3), 280–298 (2013). http://ijr.sagepub.com/cgi/doi/10.1177/0278364912469821. doi:10.1177/0278364912469821
11. Saab, L., Ramos, O.E., Mansard, N., Sou, P., Fourquet, Jy.: Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. IEEE Trans. Robot. **29**(2), 346–362 (2013)
12. Sentis, L., Petersen, J., Philippsen, R.: Implementation and stability analysis of prioritized whole-body compliant controllers on a wheeled humanoid robot in uneven terrains. Auton. Robots **35**(4), 301–319 (2013). doi:10.1007/s10514-013-9358-8