The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Prioritized Optimization for Task-Space Control

Martin de Lasa and Aaron Hertzmann

Abstract— **We introduce an optimization framework called prioritized optimization control, in which a nested sequence of objectives are optimized so as not to conflict with higher-priority objectives. We focus on the case of quadratic objectives and derive an efficient recursive solver for this case. We show how task-space control can be formulated in this framework, and demonstrate the technique on three sample control problems. The proposed formulation supports acceleration, torque, and bilateral force constraints, while simplifying reasoning about task-space control. This scheme unifies prioritized task-space and optimization-based control. Our method computes control torques for all presented examples in real-time.**

I. Introduction

As robots become more complex, there is increased interest in defining control in terms of high-level tasks. For example, in the context of humanoid robot control, one might wish to strictly maintain balance while attempting to reach a target. Strong evidence is also emerging to suggest that low-dimensional control strategies are exploited by animals to manage complexity during control. Humans have been shown to only correct deviations in motion directly interfering with tasks [1], while high-level similarities has been reported across a wide variety of animals during locomotion [2]. Task-level control decomposition can also make the application of automated policy learning methods tractable [3]. This motivates the need to understand how control can be formulated in terms of high-level objectives.

Task-space control has a long history in robotics, e.g., [4], [5], [6], [7]; for a survey, see [8]. Task-space strategies aim to achieve tasks as closely as possible, without lower-priority tasks interfering with higher-priority tasks, while also resolving ambiguities due to underdetermined tasks. Task-space strategies in the literature can be categorized by whether control is specified in terms of velocity [4], acceleration [5], or force [6]. These methods are based on null-space projection operators. Velocity-based techniques gained early adoption because of their ease of implementation and modest computational requirements. These methods work well when high-gain tracking is desired, but are ill-suited for low-stiffness control [8]. Acceleration and force level approaches are better suited for impedance control, but are more challenging to implement since they require a dynamic model of the system. Force-based methods can also require a potentially complex set of derivatives to be calculated, making their implementation difficult [9]. An alternative approach is to specify all tasks in a constrained optimization framework [10], [11], [12], [13]. This avoids the need for projection operators, but cannot guarantee strict prioritization. In practice, adjusting weighting terms for such methods can be very difficult.

This paper presents a general framework that unifies optimization-based and prioritized task-space control approaches. Our approach is based on the concept of prioritized optimality, in which all constraints are achieved as closely as possible without interfering with higher-priority constraints. The generality of our formulation allows acceleration, torque, and bilateral force constraints to be easily incorporated, making it straightforward to implement and perform task-space control. Due to the recursive nature of our formulation, we derive a compact iterative algorithm able to handle an arbitrary number of user-specified tasks. Since tasks are formulated as constraints, failure to meet a constraint is also easily detectable. Lastly, because we can enforce a prioritization order, our method eliminates the need to tune objective function weights. To demonstrate our approach, several simulations of increasing complexity are presented.

Martin de Lasa and Aaron Hertzmann are with Dept. of Computer Science, University of Toronto, 10 King's College Road, Toronto, ON, CANADA mdelasa|hertzman@dgp.toronto.edu

II. Prioritized Optimization

We now introduce the concept of prioritized optimization. In the next section, we show how it can be applied to task-space control.

A. Problem Statement

In a prioritized optimization problem, we are given a list of objective functions $E_i(\mathbf{x})$ over some variable $\mathbf{x} \in \Re^D$. For example, suppose we are given three objective functions $E_1(\mathbf{x}), E_2(\mathbf{x})$, and $E_3(\mathbf{x})$. There are three steps to define the solution to this problem. Our first goal is to find the minimum of $E_1(\mathbf{x})$. Since tasks are generally underconstrained, $E_1(\mathbf{x})$ will typically have multiple minimizers. Our next goal is to find which among these valid minimizers have minimal $E_2(\mathbf{x})$. Finally, among this new set of minimizers, we want to return the one which minimizes $E_3(\mathbf{x})$. More formally, we can write the problem as the following sequence of optimizations:

$$h_1 = \min_{\mathbf{x}} E_1(\mathbf{x}) \tag{1}$$

$$h_2 = \min_{\mathbf{x}} E_2(\mathbf{x}) \quad \text{subject to } E_1(\mathbf{x}) = h_1 \tag{2}$$

$$h_3 = \min_{\mathbf{x}} E_3(\mathbf{x}) \quad \text{subject to } E_1(\mathbf{x}) = h_1, E_2(\mathbf{x}) = h_2 \tag{3}$$

We then return the value $\mathbf{x}^*$ that solves this final optimization. Figure 1 illustrates a case with two tasks.

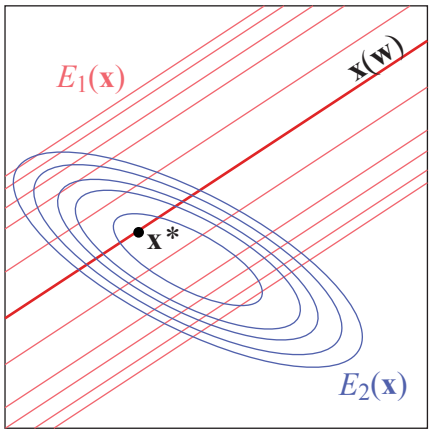

Fig. 1. A pictorial representation of a prioritized optimization problem with 2 objectives. $\mathbf{E}_1$ (in red) constrains the minimizer to lie in the family of solutions $\mathbf{x}(\mathbf{w})$ shown as a dark red line. Selecting among valid solutions, the solver identifies the minimizer in the family of solutions $\mathbf{x}(\mathbf{w})$ producing lowest $\mathbf{E}_2$ (shown in blue).

In the general case of N objectives $E_i(\mathbf{x})$, the problem can be defined recursively as:

$$h_i = \min_{\mathbf{x}} E_i(\mathbf{x}) \quad \text{subject to } E_k(\mathbf{x}) = h_k \ \forall k < i \tag{4}$$

Then, the prioritized optimization problem is to find some $\mathbf{x}_N$ that solves for h_N. In principle, prioritized optimization could be solved by N applications of a general constrained optimization algorithm.

In this paper, we focus on the case where all objectives are positive semidefinite quadratics, and thus can be written as:

$$E_i = ||\mathbf{g}_i(\mathbf{x})||^2 \tag{5}$$

where we define

$$\mathbf{g}_i(\mathbf{x}) = \mathbf{A}_i\mathbf{x} - \mathbf{b}_i \tag{6}$$

for some matrix $\mathbf{A}_i$ and vector $\mathbf{b}_i$. Given N objectives, our goal is to solve for unknowns $\mathbf{x} \in \Re^D$ such that we minimize $||\mathbf{g}_i(\mathbf{x})||^2$, subject to each of the previous objectives.

In this formulation, each energy term can be thought of as a constraint of the form

$$\mathbf{g}_i(\mathbf{x}) = 0 \tag{7}$$

Solving the prioritized optimization problem ensures that this constraint will be satisfied when possible, since, in general, E_i is minimized at zero. When a constraint cannot be satisfied (typically because it conflicts with a higher-level constraint), minimizing E_i will minimize the failure of the constraint. Hence, each $E_i/\mathbf{g}_i$ can be viewed as either a constraint or as an objective.

B. Solution for Three Objectives

To see how to solve this type of optimization problem, we consider the case of three quadratic objectives $E_1(\mathbf{x}), E_2(\mathbf{x})$, and $E_3(\mathbf{x})$. Solving for $\mathbf{x}$ in a naïve manner would involve solving a sequence of quadratically-constrained quadratic programs. However, the space of optimal solutions to a positive semidefinite quadratic objective must be a linear subspace. Specifically, the space of optimal solutions to $E_1 = ||\mathbf{A}_1\mathbf{x} - \mathbf{b}_1||^2$ can be parameterized by a vector $\mathbf{w}_1$ as:

$$\mathbf{x}(\mathbf{w}_1) = \mathbf{C}_1\mathbf{w}_1 + \mathbf{d}_1 \tag{8}$$

where $\mathbf{C}_1 = \text{null}(\mathbf{A}_1)$ is a nullspace basis for $\mathbf{A}_1$, and $\mathbf{d}_1 = \mathbf{A}_1^\dagger\mathbf{b}_1$ is a minimizer of E_1. Any choice of $\mathbf{w}_1$ produces a minimizer of $E_1(\mathbf{x})$. The quantities $\mathbf{C}_1$ and $\mathbf{d}_1$ can be computed using a single Singular Value Decomposition (SVD). Note that their dimensionalities depend on the rank of $\mathbf{A}_1$.

Now we can express the second objective in terms of this parameterization:

$$\begin{aligned}\mathbf{g}_2(\mathbf{w}_1) &= \mathbf{A}_2\mathbf{x}(\mathbf{w}_1) - \mathbf{b}_2 &(9)\\ &= \mathbf{A}_2\mathbf{C}_1\mathbf{w}_1 - (\mathbf{b}_2 - \mathbf{A}_2\mathbf{d}_1) &(10)\\ &= \bar{\mathbf{A}}_2\mathbf{w}_1 - \bar{\mathbf{b}}_2 &(11)\end{aligned}$$

where

$$\begin{aligned}\bar{\mathbf{A}}_2 &= \mathbf{A}_2\mathbf{C}_1 &(12)\\ \bar{\mathbf{b}}_2 &= \mathbf{b}_2 - \mathbf{A}_2\mathbf{d}_1. &(13)\end{aligned}$$

This reparameterizes the second objective function as $E_2(\mathbf{w}_1) = ||\bar{\mathbf{A}}_2\mathbf{w}_1 - \bar{\mathbf{b}}_2||^2$. The solutions to this problem can be described by a smaller subspace, parameterized by a vector $\mathbf{w}_2$ as $\mathbf{w}_1(\mathbf{w}_2) = \mathbf{C}_2\mathbf{w}_2 + \mathbf{d}_2$, where $\mathbf{C}_2 = \text{null}(\bar{\mathbf{A}}_2)$ and $\mathbf{d}_2 = \bar{\mathbf{A}}_2^\dagger\bar{\mathbf{b}}_2$. If we only have two objectives, such as the example shown in Figure 1, the solution to the optimization problem $\mathbf{x}^* = \mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1$.

To account for the third objective, we reparametrize $\mathbf{g}_3$ in terms of $\mathbf{w}_2$, using:

$$\begin{aligned}\mathbf{x}(\mathbf{w}_2) &= \mathbf{x}(\mathbf{w}_1(\mathbf{w}_2)) &(14)\\ &= \mathbf{C}_1(\mathbf{C}_2\mathbf{w}_2 + \mathbf{d}_2) + \mathbf{d}_1 &(15)\\ \mathbf{g}_3(\mathbf{w}_2) &= \mathbf{A}_3\mathbf{x}(\mathbf{w}_2) - \mathbf{b}_3 &(16)\\ &= \bar{\mathbf{A}}_3\mathbf{w}_2 - \bar{\mathbf{b}}_3 &(17)\end{aligned}$$

where

$$\begin{aligned}\bar{\mathbf{A}}_3 &= \mathbf{A}_3\mathbf{C}_1\mathbf{C}_2 &(18)\\ \bar{\mathbf{b}}_3 &= \mathbf{b}_3 - \mathbf{A}_3(\mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1). &(19)\end{aligned}$$

The subspace of optima for this problem is given by $\mathbf{w}_2(\mathbf{w}_3) = \mathbf{C}_3\mathbf{w}_3 + \mathbf{d}_3$, where $\mathbf{C}_3$ and $\mathbf{d}_3$ are defined as above. Then, any value of $\mathbf{w}_3$ gives an optimal solution for the final objective. Substituting in the original space, we have:

$$\begin{aligned}\mathbf{x}(\mathbf{w}_3) &= \mathbf{x}(\mathbf{w}_1(\mathbf{w}_2(\mathbf{w}_3))) &(20)\\ &= \bar{\mathbf{C}}\mathbf{w}_3 + \bar{\mathbf{d}} &(21)\end{aligned}$$

where $\bar{\mathbf{C}} = \mathbf{C}_1\mathbf{C}_2\mathbf{C}_3$ and $\bar{\mathbf{d}} = \mathbf{C}_1\mathbf{C}_2\mathbf{d}_3 + \mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1$. Hence, a solution to the entire prioritized optimization problem is given by $\mathbf{x}^* = \bar{\mathbf{d}}$.

C. General Solution

We can generalize to the case of N objectives as follows. First, we define an initial subspace equivalent to the full space $\mathbf{x}(\mathbf{w}_0) = \mathbf{w}_0$, so that $\mathbf{C}_0 = \mathbf{I}_{D\times D}$ and $\mathbf{d}_0 = \mathbf{0}_D$. Then for all $1 \leq i \leq N$, we have:

$$\bar{\mathbf{A}}_i = \mathbf{A}_i \prod_{j=0}^{i} \mathbf{C}_j \tag{22}$$

$$\bar{\mathbf{b}}_i = \mathbf{b}_i - \mathbf{A}_i \sum_{j=1}^{i-1} \left(\prod_{k=0}^{j-1} \mathbf{C}_k \right) \mathbf{d}_j \tag{23}$$

$$\mathbf{C}_i = \text{null}(\bar{\mathbf{A}}_i) \tag{24}$$

$$\mathbf{d}_i = \bar{\mathbf{A}}_i^{\dagger} \bar{\mathbf{b}}_i \tag{25}$$

$$\bar{\mathbf{d}}_i = \sum_{j=0}^{i-1} \left(\prod_{k=0}^{j-1} \mathbf{C}_k \right) \mathbf{d}_j \tag{26}$$

At each step, the substitution of the subspace is of the form $\mathbf{g}_i(\mathbf{w}_i) = \bar{\mathbf{A}}_i \mathbf{w}_i + \bar{\mathbf{b}}_i$. The solution subspace for this problem is $\mathbf{w}_i(\mathbf{w}_{i-1}) = \mathbf{C}_i \mathbf{w}_i + \mathbf{d}_i$. The substitution of $\mathbf{d}_i$ back into the original space is $\bar{\mathbf{d}}_i$, yielding the final solution to the entire problem $\mathbf{x}^* = \bar{\mathbf{d}}_N$.

D. Efficient Algorithm

Equations (22) - (26) provide a recursive description of the prioritized solver for quadratic constraints. However, they contain a great deal of redundant computation. We can define a more efficient recursive algorithm as follows:

Algorithm 1: Quadratic Prioritized Solver

```
1  C̄ ← I , d̄ ← 0
2  for i = 1 to N do
3  |   Ā_i ← A_i C̄
4  |   b̄_i ← b_i − A_i d̄
5  |   d̄ ← d̄ + C̄ Ā_i^† b̄_i
6  |   if Ā_i is full rank then
7  |   |   return d̄
8  |   end
9  |   C̄ ← C̄ null(Ā_i)
10 end
11 return d̄
```

In this algorithm, at each step $\bar{\mathbf{C}} = \prod_i \mathbf{C}_i$.

If the objective functions use up all available degrees-of-freedom (DOF), line 11 will never be reached. For example, if $\mathbf{A}_N$ is full-rank, then $\bar{\mathbf{A}}_N$ must also be full-rank. We discuss this issue further in the next section.

The nullspace and pseudoinverse computations using SVD require user-specified tolerances. These tolerances impact solution smoothness. We use:

$$\text{tol}_{pinv} = \sqrt{\epsilon} \tag{27}$$

$$\text{tol}_{null} = \max(m, n) \max(\mathbf{s}) \epsilon \tag{28}$$

where m and n are the dimensions of each $\bar{\mathbf{A}}_i$, $\mathbf{s}$ are its eigenvalues, and ϵ is the machine tolerance for the floating point representation used by the implementation.

The exact complexity of the iterative solver described in Algorithm 1 depends on the number of tasks and rank of each task. However, since a SVD is required for each task, a worst-case estimate is $O(D^3)$. This is equivalent to other existing task-space control approaches, which typically require a matrix inverse, or pseudoinverse, to compute projection operators [4], [14].

III. Task-Space Control

We now describe how to apply prioritized optimization to task-space control of dynamic articulated rigid-body simulation. When defining a controller, one must define a set of tasks as constraints in the form $\mathbf{g}_i(\mathbf{x})$ that, at each time-step, are provided to the optimizer to calculate joint torques. We define the vector of free variables as:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau}^T & \ddot{\mathbf{q}}^T \end{bmatrix}^T. \tag{29}$$

In all of our examples, the top-level constraint is the system equations of motion relating joint torques and accelerations:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}. \tag{30}$$

which is written in constraint form as:

$$\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{C} + \mathbf{G} \end{bmatrix}. \tag{31}$$

The quantities $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{G}$ are the joint space inertia matrix, Coriolis/centrifugal and gravitational forces respectively. For brevity, we omit dependence of the quantities on $\mathbf{q}$ and $\dot{\mathbf{q}}$ in the remainder of the paper. We use the Recursive Newton-Euler and Composite Rigid-Body algorithms to calculate these quantities efficiently [15]. Note that this constraint will always be satisfied exactly (i.e., $\mathbf{g}_1(\mathbf{x}) = 0$) because it is the top-level constraint.

We then define additional constraints for the tasks we wish to control. We use acceleration-level constraints for all problems described in this paper. For a given system configuration $\mathbf{q}$, forward kinematics relates generalized system coordinates to task-space quantities according to:

$$\mathbf{y}_i = f_i(\mathbf{q}) \tag{32}$$

where $\mathbf{y}_i$ is any task variable that we wish to control, such as end-effector position or center-of-mass (COM). Differentiating (32) twice we obtain:

$$\dot{\mathbf{y}}_i = \mathbf{J}_i \dot{\mathbf{q}} \tag{33}$$

$$\ddot{\mathbf{y}}_i = \mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}} \tag{34}$$

where $\mathbf{J}_i = \frac{\partial \mathbf{y}_i}{\partial \mathbf{q}}$ is the Jacobian matrix of $\mathbf{y}_i$. We can specify a desired acceleration by linear control as:

$$\ddot{\mathbf{y}}_d^i = \ddot{\mathbf{y}}_r + \mathbf{K}_v(\dot{\mathbf{y}}_r - \dot{\mathbf{y}}_i) + \mathbf{K}_p(\mathbf{y}_r - \mathbf{y}_i). \tag{35}$$

where $\mathbf{K}_v$ and $\mathbf{K}_p$ are task-specific servo gain matrices. $\mathbf{y}_r$, $\dot{\mathbf{y}}_r$, and $\ddot{\mathbf{y}}_r$ are the reference position, velocity, and acceleration values. Then, the constraint for this task is:

$$\mathbf{g}_i(\mathbf{x}) = \ddot{\mathbf{y}}_d^i - \ddot{\mathbf{y}}_i \tag{36}$$

$$= \ddot{\mathbf{y}}_d^i - (\mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}}). \tag{37}$$

Note that the task $\mathbf{g}_i(\mathbf{x})$ will always be satisfied as long as it is achievable without violating the equations of motion.

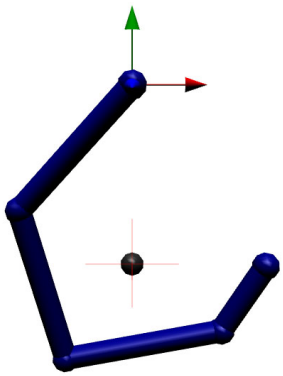

Fig. 2. Planar Four Link Chain Simulation. The chain's COM is shown as a black sphere. The Cartesian location of the commanded trajectory is shown as a red crosshair. Chain movement is restricted to the XY plane.

Lower-level tasks will be satisfied as closely as possible as well.

It is important to define sufficient tasks to completely specify the torque. Although Algorithm 1 returns the minimum norm solution when ambiguities remain (line 11), in practice this solution leads to instability due to insufficient damping. Instead, in all the examples that follow, we define the final task such that it affects all torques and includes some dissipative component. For example, the final task may damp velocities or servo all joints to some default pose. Since the task will be full rank, it will not be exactly satisfied. Instead, we will obtain the solution satisfying the damping task as closely as possible. Based on our experience, this is sufficient to prevent instability.

IV. Examples

We now apply our technique to three control problems: tracking a target with a planar serial chain, tracking a target with an underacted planar chain using partial feedback linearization, and performing squats with a planar biped with bilateral force-ground constraints. All examples are shown in the accompanying video. These examples employ a generalized-coordinate simulator using Featherstone's algorithm [15]. The simulator supports efficient computation of forward and inverse dynamics, Jacobians, and the matrix representation of dynamics needed by the task constraints (37). The simulations range in complexity from 4 to 9 DOFs, with the individual tasks ranging from 8 to 20 dimensions. All examples are integrated and controlled at 1 kHz and run in real time.

A. Fully Actuated Chain

A fully actuated planar serial-chain with four one-DOF revolute joints is first simulated (Figure 2). All links are given identical inertial and geometric properties (length=0.7 m, mass=10 kg, inertia=0.41 kg·m^2). The primary control objective is to servo the chain's COM along a sinusoidal figure-eight reference trajectory:

$$\mathbf{y}_r = \begin{bmatrix} A\sin(t) & A\sin(2t) \end{bmatrix}^T. \tag{38}$$

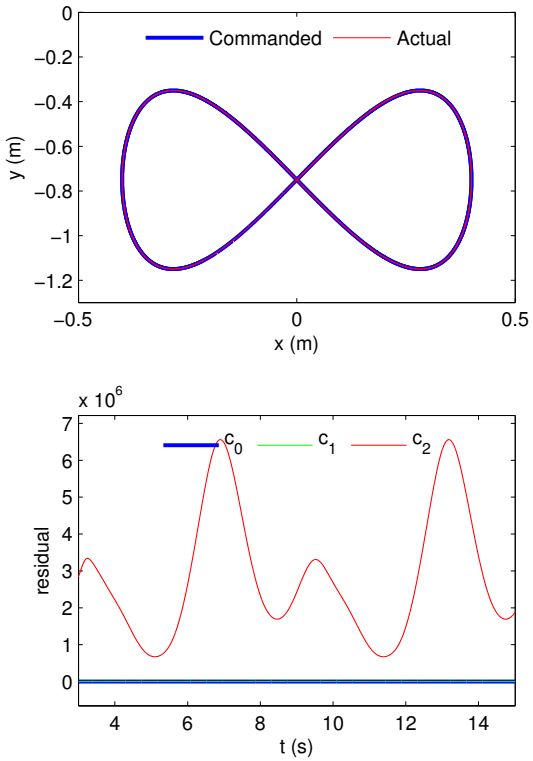

Fig. 3. Planar Four Link Simulation Results. **Top:** COM commanded and actual trajectories, **Bottom:** Prioritize solver rask residuals.

This yields the task objective:

$$\mathbf{g}_2(\mathbf{x}) = \ddot{\mathbf{y}}_d^{com} - (\mathbf{J}_{com}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{com}\dot{\mathbf{q}}) \tag{39}$$

where $\ddot{\mathbf{y}}_{com}$ is calculated using (35). To resolve redundancy in the remaining DOFs, a secondary configuration-space task servos all joints to a reference posture:

$$\mathbf{g}_3(\mathbf{x}) = \ddot{\mathbf{y}}_d^{posture} - (\mathbf{J}_{posture}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{posture}\dot{\mathbf{q}}). \tag{40}$$

Since the second constraint is for a configuration-space task (i.e., $\mathbf{J}_{posture} = \mathbf{I}$, $\dot{\mathbf{J}}_{posture} = 0$), it can be written:

$$\mathbf{g}_3(\mathbf{x}) = \ddot{\mathbf{y}}_d^{posture} - \ddot{\mathbf{q}}. \tag{41}$$

Summarizing all constraints in matrix form we obtain:

$$\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} \end{bmatrix}\mathbf{x} - \begin{bmatrix} \mathbf{C}+\mathbf{G} \end{bmatrix} \tag{42}$$

$$\mathbf{g}_2(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{J}_{com} \end{bmatrix}\mathbf{x} - \begin{bmatrix} \mathbf{v}_{com} \end{bmatrix} \tag{43}$$

$$\mathbf{g}_3(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}\mathbf{x} - \begin{bmatrix} \mathbf{v}_{posture} \end{bmatrix} \tag{44}$$

where $\mathbf{v}_i = \ddot{\mathbf{y}}_d^i - \dot{\mathbf{J}}_i\dot{\mathbf{q}}$. Figure 3 shows commanded and actual task-space coordinates for a 15 second simulation using the described controller. We also see that the first two constraints are exactly satisfied, while there is error in the third task. This is expected, since the last task operates in the mechanism's configuration space and conflicts with the COM servoing task.

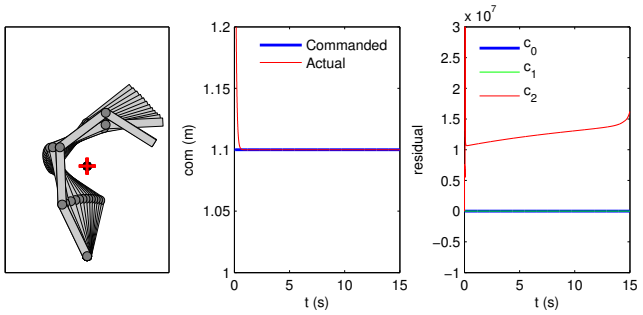

Fig. 4. Underactuated Chain Example. **Left:** Time-series of underactuated chain during COM tracking task. Desired COM height shown as red crosshair. Actual COM height shown as black circle. **Middle:** Commanded (blue) and actual (red) values for COM height. **Right:** Prioritized optimization residuals for provided constraints.

B. Underactuated Chain

A mechanism is said to be underactuated if a control system is unable to produce accelerations in certain DOFs. This includes well-known systems such as Acrobot [16] and all untethered humanoid robots. Underactuation arises when a system has more DOFs than actuators. One strategy used to control such systems is to use a partial feedback linearization (PFL), to couple active and passive DOFs [3], [16]. Due to the generality of our technique, defining a PFL control for an underactuated version of the system requires only minor modifications to the specification. This avoids having to manually decompose the mass matrix into active/passive parts [3].

In this example, we constrain the torque on the first joint of the 4-link serial chain to be passive and perform a setpoint regulation task. We specify the following constraints:

$$\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} \\ \mathbf{S} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{C}+\mathbf{G} \\ 0 \end{bmatrix} \tag{45}$$

$$\mathbf{g}_2(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{J}_{com} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{v}_{com} \end{bmatrix} \tag{46}$$

$$\mathbf{g}_3(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{v}_{posture} \end{bmatrix} \tag{47}$$

where $\mathbf{S} = \begin{bmatrix} \mathbf{I}_{1\times1} & \mathbf{0}_{1\times3} \end{bmatrix}$ is the passive joint selection matrix. We set $\mathbf{K}_p = 0$ in $\mathbf{g}_3$, such that it only applies damping forces. All other quantities and gains are unchanged from section IV-A.

Figure 4 shows results for this simulation. Despite making the base joint's torque passive, the COM height is accurately tracked. As can be seen from the solver residuals, with the exception of the final full-rank task, all task constraints are met.

C. Bilateral Constraints

As a last example and to test how our method scales to more complex problems, we simulate a planar human performing squats (Figure 6). We use a HAT model (Figure 7) that aggregates head, arms, and trunk mass properties into a single link. Character height and weight correspond to a 50th percentile North American male, with individual link mass/geometric properties estimated using data from Winter [17]. Given link mass and geometric properties, inertia is calculated using a thin-rod assumption. Provided COM values are expressed in link local coordinates. Inertia is expressed about each link's COM (cf. Table I).

TABLE I

PLANAR BIPED MASS PROPERTIES

Link	Length $[m]$	Mass $[kg]$	COM (x,y) $[m]$		Inertia $[kg \cdot m^2]$
trunk	0.846	55.47	0.00	0.325	3.670
uleg	0.441	8.18	0.00	-0.191	0.166
lleg	0.443	3.81	0.00	-0.191	0.068
foot	0.274	1.19	0.07	-0.035	0.001
total	1.80	81.82			

We model foot/ground interaction using a set of bilateral force constraints, acting at 2 contact points on each foot. This requires that we include external forces as unknowns

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau}^T & \ddot{\mathbf{q}}^T & \mathbf{f}^T \end{bmatrix}^T \tag{48}$$

and that we augment the equations of motion accordingly. Three other tasks are provided in our example: a task to regulate horizontal and vertical COM position, a task to regulate trunk orientation, and a final configuration-space task servoing all joints to a reference posture. The complete set of constraints is:

$$\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} & \mathbf{J}_c^T \\ \mathbf{0} & \mathbf{J}_c & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{C}+\mathbf{G} \\ \boldsymbol{\beta} - \dot{\mathbf{J}}_c\dot{\mathbf{q}} \\ \mathbf{0} \end{bmatrix} \tag{49}$$

$$\mathbf{g}_2(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{J}_{com} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{v}_{com} \end{bmatrix} \tag{50}$$

$$\mathbf{g}_3(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{J}_{trunk} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{v}_{trunk} \end{bmatrix} \tag{51}$$

$$\mathbf{g}_4(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{v}_{posture} \end{bmatrix}. \tag{52}$$

In the above $\mathbf{J}_c^T = \begin{bmatrix} \mathbf{J}_{c1}^T & \mathbf{J}_{c2}^T & \mathbf{J}_{c3}^T & \mathbf{J}_{c4}^T \end{bmatrix}$ contains the Jacobians for all contacts points and $\mathbf{S} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times6} \end{bmatrix}$ is the passive joint selection matrix for our simulation. We compensate for numerical drift issues arising from expressing contact point position constraints at the acceleration level using Baumgarte stabilization (i.e., $\boldsymbol{\beta} = \mathbf{K}_p^\beta \varepsilon + \mathbf{K}_v^\beta \dot{\varepsilon}$) [18].

Figure 5 summarizes key results for this example. As can be seen, COM position and trunk orientation are accurately tracked. With the exception of the last full-rank task, residuals for all other tasks are within machine tolerance of zero.

V. DISCUSSION

A. Evaluation

An alternative to executing tasks in strict priority order is to combine them in a weighted fashion:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=2}^{N} \alpha_i ||\mathbf{g}_i(\mathbf{x})||^2$$
$$\text{subject to } \mathbf{g}_1(\mathbf{x}). \tag{53}$$

This corresponds to formulating control as a constrained quadratic program (QP), with weights determining each objective's influence.

To evaluate the benefits of our prioritized optimization scheme, we repeat the actuated chain (IV-A) and squatting simulations (IV-C) using a QP-based control. We reuse the

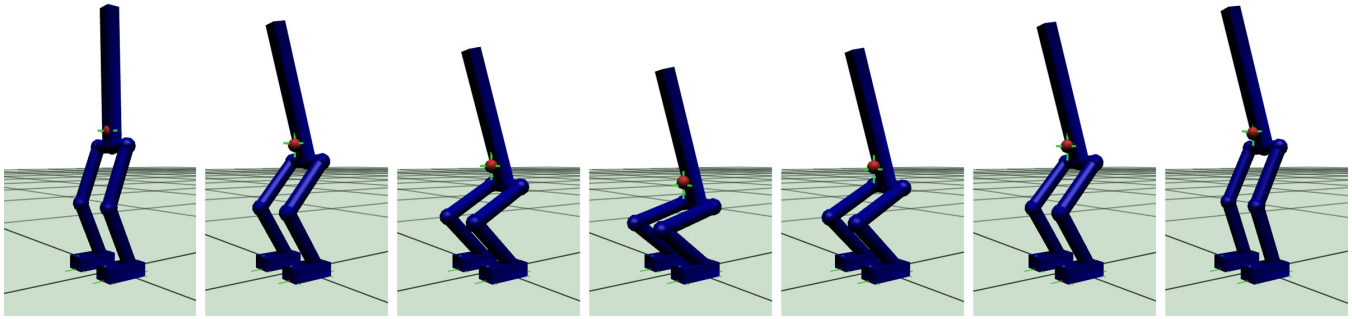

Fig. 6. Seven frames from the planar biped squatting simulation. Actual and desired COM/contact point positions are shown as red spheres and green crosshairs respectively.

same tasks, gains, and setpoints from these examples and combine them as described in (53). Since we wish to ensure that control does not violate the system equations of motion, we enforce the top-level dynamics tasks, $\mathbf{g}_1(\mathbf{x})$, as an equality constraint. For the squatting simulation, we also include an additional regularization task (i.e., $\mathbf{A}_{reg} = \mathbf{I}$, $\mathbf{b}_{reg} = \mathbf{0}$, $\alpha_{reg} = 0.01$) to improve problem conditioning. Our implementation uses MOSEK's (www.mosek.com) interior-point QP solver [19].

For the actuated chain example, we were unable to find a set of weights for the QP formulation that achieved both tasks accurately. Instead, interference was observed between tasks in all attempted simulations. As one task was achieved, its contribution to the objective diminished, causing the optimization to favor the other objective.

For the squatting example, many weight combinations cause the simulation to fail. With sufficient tuning, the QP-based controller results approach those from the prioritized solver. Figure 8 shows QP controller performance for a representative set of weight combinations. The QP-based control implementation runs at approximately half the speed of the prioritized solver.

Our prioritized solver allows a great deal of control design flexibility. Users are free to specify an arbitrary set of task-space constraints, as long as they are linear functions of free variables. In practice, this can result in constraints that are linearly dependent and that interfere with one another. For example, in our squatting simulations contact point Jacobians become ill-conditioned when the foot is flat on the ground. Jacobians can also become rank deficient near mechanical singularities, such as when the knee is fully extended. Because the proposed prioritized solver relies on a SVD-based singularity-robust pseudoinverse, problems with ill-conditioned systems are mitigated. Furthermore, since the pseudoinverse returns the minimum norm solution for a particular subtask, instabilities arising from excessive control torques or forces are avoided. The algorithm also works equally well with single or double precision floating-point numbers. In contrast, the single-precision implementation of the QP-based squatting controller fails, since it is unable to handle the ill-conditioned dynamics constraint.

B. Task Specification Options

In section IV, we presented three simulations of increasing complexity. To handle differences between the simulated models, we augmented the highest priority task to include torque and bilateral force constraints. Although we recommend the formulation presented above, which uses a single constraint to relate problem unknowns, other alternatives exist. For example, the top-level constraint (49) in IV-C can be broken up into several different parts and rewritten as:

$$\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} & \mathbf{J}_c^T \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{C} + \mathbf{G} \end{bmatrix} \quad (54)$$

$$\mathbf{g}_2(\mathbf{x}) = \begin{bmatrix} 0 & \mathbf{J}_c & 0 \end{bmatrix} \mathbf{x} - \begin{bmatrix} \boldsymbol{\beta} - \dot{\mathbf{J}}_c \dot{q} \end{bmatrix} \quad (55)$$

$$\mathbf{g}_3(\mathbf{x}) = \begin{bmatrix} \mathbf{S} & 0 & 0 \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{0} \end{bmatrix}. \quad (56)$$

Another variant is to define the free variables as solely the torques ($\mathbf{x} = \boldsymbol{\tau}$), solve the equations of motion for $\ddot{\mathbf{q}}$, and substitute them into all remaining constraints. Though these alternative representations are mathematically equivalent, we found that they were more cumbersome to work with and did not offer any computational benefits. Nonetheless, this illustrates the design flexibility afforded by the prioritized optimization framework.

C. Comparison

Our approach draws inspiration from two areas: optimization-based control [10], [12], [20] and prioritized task-space control [6], [9], [21], but provides advantages over both.

Optimization based approaches such as those based on QP are versatile and very general. By exploiting off-the-shelf optimizers, they provide a general framework for reasoning about control. These methods typically use a weighted sum of quadratic objectives to specify different aspects of the desired output motion. Though this is straightforward, tuning weights for different objective function elements is non-trivial. This approach also does not allow strict prioritization of task behaviors.

In contrast, our approach allows the same ease of use as methods based on optimization while permitting task prioritization. As shown above, using a single framework, it is straightforward to incorporate a variety of constraint types.

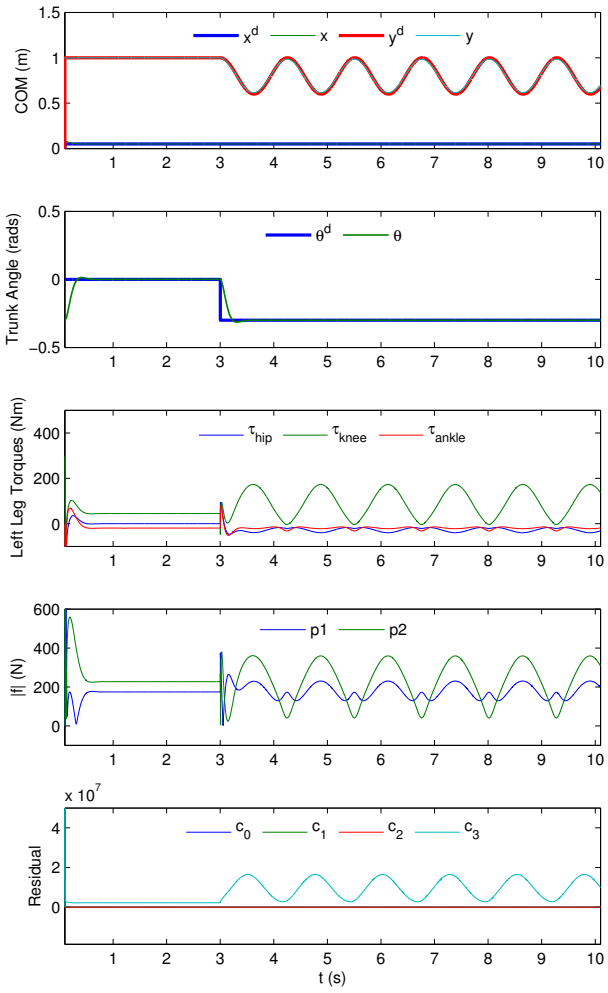

Fig. 5. Planar biped squatting simulation results. Right leg joint torques and contact forces omitted for brevity. From top to bottom, plots show: commanded vs. actual COM position, commanded vs. actual trunk orientation, left leg control torques, magnitude of bilateral contact forces for leg foot (p1 and p2 are the foot contact points), and residuals for all tasks specified to the prioritized solver (i.e., $\mathbf{c}_i = ||\mathbf{A}_i\mathbf{x} - \mathbf{b}_i||$).

Techniques based on prioritized task-space control have a long history in robotics. To truly compensate for system dynamics in control, acceleration or force level approaches are required. Force approaches such Operational Space Control [9] correctly decouple primary and secondary behaviors and ensure minimal interference between tasks. However, these methods require the computation of a complex set of task-specific projection matrices and of their derivatives.

Our approach guarantees minimal interference between tasks and does not require computation of any specialized projection matrices. Instead, we rely on a general solver that requires only task-dependent Jacobians and a dynamic system model in standard block form as input, which can be efficiently computed [22].

VI. Conclusions

We have presented a general framework for reasoning and performing task-space control based on the concept of prioritized optimization. Prioritized optimization combines

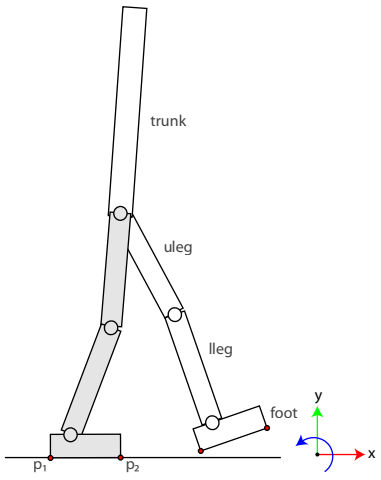

Fig. 7. A 7 link (9 DOF) planar HAT model is used for squatting simulations. Model parameters are taken for a 50th percentile North American male. Our model has a floating base, connecting the character's pelvis to the world via a three DOF planar joint.

the ease of use of convex-optimization methods, with the benefit of ensuring strict task prioritization. For the case of positive semidefinite quadratic objectives, we obtain an efficient recursive algorithm, with real-time performance. Because prioritized optimization can ensure minimal interference between objectives, it is ideally suited for task-space control. This guarantee enables decomposition of complex motions into simpler high-level actions using a straightforward approach. Acceleration, torque, and force constraints can also be included into our formulation easily.

There are several directions we wish to explore for this research. We have begun modifying our approach to handle unilateral constraints. This extension is needed for better modelling of contact forces with the environment. We are also incorporating bounds constraints, a requirement to produce more conservative control strategies. The inclusion of configuration space bound constraints will also ensure the solver yields feasible solutions. Following these additions we plan to test our approach on more interesting systems, such as more detailed humanoid models.

VII. Acknowledgments

We thank Jack Wang and Igor Mordatch for their comments on the manuscript. This work was supported by the Canada Foundation for Innovation, the Canadian Institute for Advanced Research, a Microsoft Research New Faculty Fellowship, the National Sciences and Engineering Research Council of Canada, and the Ontario Ministry of Research and Innovation.

References

[1] E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination.," *Nat Neuroscience*, vol. 5, no. 11, 2002.

[2] R. J. Full and D. E. Koditschek, "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land," *J. of Experimental Biology*, vol. 202, 1999.

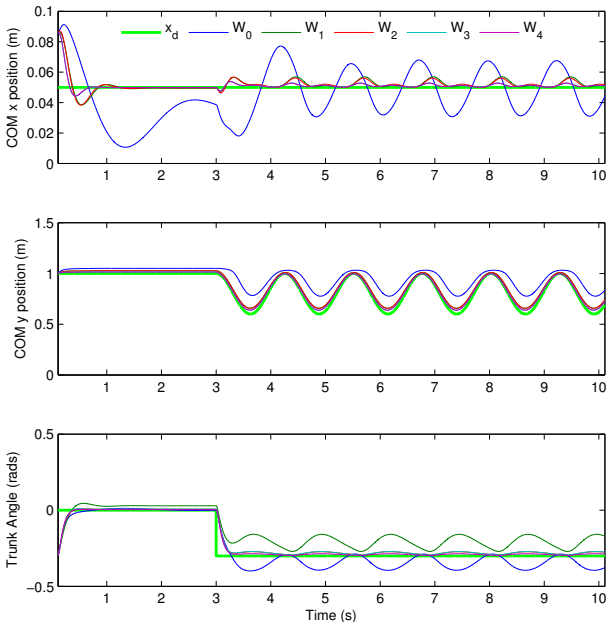

Fig. 8. Planar biped results using non-prioritized QP-based control. Tracking of task-space quantities for five different sets of weights is shown. Tasks weights are: $\mathbf{W}_0 = (10, 10, 0.1)$, $\mathbf{W}_1 = (50, 10, 0.1)$, $\mathbf{W}_2 = (50, 50, 0.1)$, $\mathbf{W}_3 = (100, 50, 0.1)$, $\mathbf{W}_4 = (100, 100, 0.1)$. As weights are increased, QP control output converges to prioritized solver output.

[3] A. Shkolnik and R. Tedrake, "High-dimensional underactuated motion planning via task space control," *IROS '08*, Sept. 2008.

[4] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 7, 1977.

[5] P. Hsu, J. Hauser, and S. Sastry, "Dynamic control of redundant manipulators," *J. of Robotic Systems*, vol. 6, no. 2, 1989.

[6] O. Khatib, "A unified approach to motion and force control of robot manipulators: The operational space formulation," *IEEE J. of Robotics and Automation*, vol. 3, no. 1, 1987.

[7] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *Int. J. of Robotics Research*, vol. 6, no. 2, 1987.

[8] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *Int. J. of Robotics Research*, vol. 27, no. 6, 2008.

[9] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole body dynamic behavior and control of human-like robots," *Int. J. of Humanoid Robotics*, vol. 1, 2004.

[10] Y. Abe, M. da Silva, and J. Popović, "Multiobjective control with frictional contacts," in *SCA '07*, Eurographics Association.

[11] C. Azevedo, P. Poignet, and B. Espiau, "Moving horizon control for biped robots without reference trajectory," in *ICRA '02*.

[12] M. da Silva, Y. Abe, and J. Popovic, "Simulation of human motion data using short-horizon model-predictive control.," *Computer Graphics Forum*, vol. 27, no. 2, 2008.

[13] Y. Fujimoto, S. Obata, and A. Kawamura, "Robust biped walking with active interaction control between foot and ground," *ICRA '98*.

[14] O. Khatib, "Inertial properties in robotics manipulation: An object-level framework," *Int. J. of Robotics Research*, vol. 14, 1995.

[15] R. Featherstone, *Rigid Body Dynamics Algorithms*. 2007.

[16] M. Spong, "Partial feedback linearization of underactuated mechanical systems," *IROS '94*.

[17] D. A. Winter, *Biomechanics and Motor Control of Human Movement*. 2004.

[18] M. Cline and D. Pai, "Post-stabilization for rigid body simulation with contact and constraints," *ICRA '03*.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

[20] A. Hofmann, S. Massaquoi, M. Popovic, and H. Herr, "A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs," *IROS '04*.

[21] Y. Abe and J. Popović, "Interactive animation of dynamic manipulation," in *SCA '06*.

[22] R. Featherstone and D. E. Orin, "Robot dynamics: Equations and algorithms," in *ICRA '00*.