Hierarchical Inverse Dynamics for Control of Lower body of Sarcos Robot

Akshara Rai
May 7, 2015

Abstract

Controlling complex humanoid robots can involve a lot of competing paradigms that can be simplified by breaking down the control process into a hierarchy of tasks. For this project, we implemented one such controller, as introduced in [1], that uses a user-defined hierarchy of tasks to perform a required task. We tested this on tasks such as squatting and balancing on flat ground under disturbances. The controller was seen to follow the desired trajectories well. However, when testing on stepping tasks, we found the simulator incapable of handling model inconsistencies. We show that hierarchical approaches such as these are capable of handling conflicting tasks in a hierarchy by maintaining a strict ranking between constraints.

1 Introduction

Controlling humanoid robots can be a cumbersome process due to a multiple of factors involved in the overall behavior, for example, controlling center of mass (COM) behavior as well as individual joints, while balancing and planning footsteps. Some of these objectives can be competing, for example following a COM trajectory while minimizing torques applied, in which case an exact solution cannot be found. A common way to handle this is to add weights to the cost functions related to the different tasks and one expects the tasks with higher costs to be satisfied as compared to conflicting tasks. But this process requires tuning of weights to ensure that the task is satisfied under all conditions and still does not necessarily ensure that one task is definitely preferred over another. We use a hierarchical quadratic program (QP) solver, as introduced in [1], that allows a user to define a hierarchy of tasks. Tasks lower in the hierarchy are only satisfied if the ones higher up are not violated. This way we can define a clear ranking of task importance for a particular example.

We implement this controller on the lower body of a simulated model of the Sarcos robot, shown in Figure 1. The robot has two legs, each with 7 degrees of freedom (DOF) and a 3DOF hip. It has an under-actuated base with 6 DOF which makes the control hard. We have perfect information of the joint angles, velocities and accelerations on the robot, as well as torques applied. We also have 6-axis foot sensors in the two feet that can return us ground reaction forces when in contact with the ground.
Figure 1: The lower body of the Sarcos robot in simulation was used for the experiments in this project. The robot has two legs, each with 7 degrees of freedom (DOF) and a 3DOF hip. It has an underactuated base with 6 DOF.

Our experiments include balancing with external disturbances, squatting and turning from side to side. We tried to implement a stepping controller, but couldn’t succeed at the attempt and the method and reasons for failure are discussed.

In the coming sections, we will describe the hierarchical QP inverse dynamics solver, a centroidal momentum controller that helps in balancing experiments and our hierarchies for different tasks. In the end, we discuss our results on different experiments.

2 Hierarchical Inverse Dynamics

The dynamics equation of a floating base robot can be written as:

$$M(q)\ddot{q} + C(q, \dot{q}) = S^T \tau + J_c^T \lambda$$

where $q = [q_j^T, x]$ is the configuration space of the robot and the floating base, $q_j$ being the joint angles and $x$ the floating base. $M(q)$ is the inertia matrix, $C(q, \dot{q})$ is the combination of all the forces acting on the robot, like gravitational and friction, etc. $S = I_{n \times 6n \times 6c}$ is the under-actuation matrix which makes the torques acting at the base 0. $J_c$ is the contact Jacobian and $\lambda$ are the contact forces. The dimension of the matrices in this equation depends on the number of contacts. If in double support, that is both feet on the ground, $x$ is 12 dimensional. If in single support, the dimension of $x$ changes to 6.

2.1 Defining Constraints

The dynamics equation of a robot is a constraint that has to be satisfied at all points of time for the robot state to be physically consistent. Other than
this, we have other assumptions, like the torques acting on the contact points are 0. This means that the contact point is not accelerating, and needs to be stationary. Other stability constraints can be keeping the center of pressure within the support polygon of the robot, etc. If there is a reference trajectory that the robot COM has to follow, this becomes a new constraint.

To solve this problem of constraint optimization, the objectives need to be expressed in terms of \((\ddot{q}, \lambda, \tau)\). As a result, all the above mentioned constraints are expressed in these terms as follows:

\[
\ddot{x}_{\text{ref}} = J_s \dot{q}
\]

and

\[
\dddot{x}_{\text{ref}} = \dot{J}_s \dot{q} + J_s \ddot{q}
\]

Other constraints are:

\[
\tau_{\min} < \tau < \tau_{\max}
\]

and

\[
\ddot{q}_{\min} < \ddot{q} < \ddot{q}_{\max}
\]

### 2.2 Full Body Controllers as Quadratic Programs

A typical quadratic program can be defined as:

\[
\min_{X} X^T G X + g^T X
\]

subject to

\[
AX + a \leq 0
\]

\[
BX + b = 0
\]

For solving the problem of inverse dynamics using QPs, we can represent \(X\) as \([\ddot{q}, \tau, \lambda]\). Then all the above mentioned constraints can be added as soft or hard constraints. If the constraint is added as a part of the cost function then the constraint is a soft constraint. If added as a hard constraint, the optimizer has to satisfy the constraint as best as it can, even if the cost is no longer minimized.

The above described QP can also be written as below:

\[
\min_{X,v,w} \|v\|^2 + \|w\|^2
\]

subject to

\[
V(AX + a) \leq v
\]

\[
W(BX + b) = w
\]

Here \(v\) and \(w\) are slack variables that are being minimized at the same time to minimize violation of constraints. \(V, W\) are weighting matrices that enable to relatively weigh different constraints in the task.

For a hierarchical QP however, the formulation is different. A user defines a desired hierarchy of tasks which are solved in a sequential order. Given a solution for \((X^r_r, v^r_r)\) a QP of the priority \(r\), all remaining solutions \(X\) are expressed by the equations

\[
X = X^r_r + Z_r u_{r+1}
\]

\[
V_r(A_r X + a) - v^r_r \leq 0,
\]

\[
\vdots
\]

\[
V_1(A_1 X + a_1) - v^1_1 \leq 0
\]
where $Z_r$ is the mapping into the null-space of all previous equalities $W_1B_1, \cdots W_rB_r$ and $u_r$ is a variable that parametrizes that nullspace. We perform an SVD to find the matrix $Z$ at each QP hierarchical step. Now imposing these constraints, the lower level QP becomes

$$
\min_{u_{r+1}, v_{r+1}} \|v_{r+1}\|^2 + \|W_{r+1}(B_{r+1}(g_r^* + Z_ru_{r+1}) + b_{r+1})\|^2
$$

subject to

$$
V_{r+1}(A_{r+1}X + a) - v_{r+1} \leq 0, \\
V_r(A_rX + a) - v_r^* \leq 0, \\
\vdots \\
V_1(A_1X + a_1) - v_1^* \leq 0
$$

This allows us to solve for new solution to constraints at step $r + 1$ while making sure that the previous constraints are not violated.

### 2.3 Centroidal Momentum Dynamics

In balancing experiments, we found that not modelling angular momentum dynamics makes the model very susceptible to external disturbances. So we added a centroidal momentum dynamics controller as introduced in [2], which creates a linear mapping from joint velocities to COM angular momentum.

$$
\dot{h} = H_G\ddot{q} + \dot{H}_G\dot{q} = F_{net}
$$

With this controller that helps control the angular as well as the linear momentum of the COM, we found our robot much more robust to external disturbances.

### 3 Results

In this section, we describe the results we obtained for the experiments we conducted using the described methods. We conducted balancing, squatting and stepping experiments.

#### 3.1 Balancing Experiments

For balancing, the desired trajectory of COM is a just a static point, with 0 desired linear and angular momentum.

We gave the robot 200N pushes from its back for 0.2s. In the video [www.andrew.cmu.edu/user/arai/balance_2.5x.mp4](www.andrew.cmu.edu/user/arai/balance_2.5x.mp4), the robot successfully recovers from two pushes.

The hierarchy of tasks for this experiment are as below:

1. Rank 1 : Dynamic Equations
2. Rank 2 :
   • Torque limits
   • Joint limits
   • COP inside support polygon

3. Rank 3 :
   • Ground reaction forces in friction cones
   • Momentum control of COM

4. Rank 4 : Reference posture of the robot

3.2 Squatting Experiments
For squatting, the trajectory of COM is defined by moving the COM first down and then up. To follow the trajectory of COM, the robot needs to generate squatting movement and standing-up movement. The video for squatting and standing-up can be found here: www.andrew.cmu.edu/user/arai/crouch_3x.mp4.

The hierarchy of tasks for this experiment are as below:

1. Rank 1 : Dynamic Equations
2. Rank 2 :
   • Torque limits
   • Joint limits
   • COP inside support polygon
3. Rank 3 :
   • PD Control on COM
   • Ground reaction forces inside friction cone
4. Rank 4 :
5. Reference posture of the robot

3.3 Stepping Experiments
We tried to implement stepping by bringing down the ground reaction forces (GRFs) on one of the feet to 0 and then giving the foot a trajectory. However, as soon as the went to low values, the dynamics model of the system had to be switched from double support to single support. This switching of models is important for the simulator as well as the QP as it determines the number of variables in the optimization, as well as the model used for integrating the physics in the simulator. We found our simulator to be very sensitive to this switching of models. It would crash often, making it very difficult to debug the issue. If it wouldn’t crash, it was extremely difficult to balance on one foot and the robot would fall. We need to debug this situation to make a successful stepping controller, which can then be extended to a walking controller. A failed video can be found here: http://www.andrew.cmu.edu/user/arai/step_fail.wmv.
4 Discussion

In this work, we used hierarchical inverse dynamics to follow target behaviours like balancing, squatting and stepping. Hierarchical QP lets up explicitly determine the importance of different constraints that vary from task to task. This helps us satisfy conflicting constraints as best as we can. For balancing experiments, we used a centroidal momentum controller to allow for large external disturbances of up to $200\text{N}$ for $0.2\text{s}$. However, when trying to perform stepping tasks, we found our simulator very sensitive to model inconsistencies. Switching models instantly can cause numerical instabilities in our simulator, which makes it hard to perform stepping and walking tasks. Future work is to sort out model inconsistency issues in the simulator and conduct walking experiments on the full robot.

References
