A Quadratic Program Controller for the ATRIAS Point-Footed Biped*

Laura Hallock[1] and Veronica Lane[2]

***Abstract*— We implement a whole-body dynamic walking controller which uses a convex quadratic program for the ATRIAS point-footed biped robot in simulation. The controller attempts to minimize an objective function while maintaining input, contact, dynamic, and no-slip constraints. We attempt to demonstrate that the Zero Moment Point (ZMP) stability criterion can apply to a point footed biped.**

I. INTRODUCTION & PROBLEM FORMULATION

An agile biped robot capable of walking and running over rough terrain has a wide variety of applications, including search and rescue operations and exploration. A biped robot is underactuated and the robot cannot execute an arbitrary trajectory in all degrees of freedom. Because of this, conventional controls do not work and the dynamics of walking robots pose a difficult challenge.

II. MOTIVATION

A QP Controller uses the ZMP stability criteria. There is debate in the robotics community about whether or not the ZMP stability criteria will work for a point footed-biped, since there is no support polygon. We aim to prove that the ZMP stability criteria is an effective model of a point-footed biped by developing a QP controller for ATRIAS, which will enable it to execute walking trajectories over flat terrain.

For our project, we chose to test the efficacy of a QP controller on the ATRIAS biped, a point-footed robot that is highly articulated enough to merit such a sophisticated control system. We chose ATRIAS due both to its similarities to Atlas — as they are both bipeds with a sophisticated, non-flat-footed gait — and because we had access to the existing URDF files to describe the robot.

III. RESOURCES & EXISTING WORK

A. The ATRIAS Biped

The ATRIAS biped — short for assume the robot is a sphere, a design paradigm used by creator Jonathan Hurst — represents the third iteration, following several monopods, of the ATRIAS series of robots developed at Oregon State University. The series was designed for walking and running over uneven, unpredictable terrain while maintaining a reasonably economy of energy. [CITATION] Although most of the published successes in control systems restrict ATRIAS to operating on a boom, the ATRIAS biped is designed to ultimately walk and run untethered. Under University of Michigan Professor Jessy Grizzle, researchers are currently working toward this goal on MARLO, an iteration of ATRIAS with prosthetic feet. MARLO has successfully taken steps, both in the lab and outdoors, while tethered to a gantry. [CITATION]

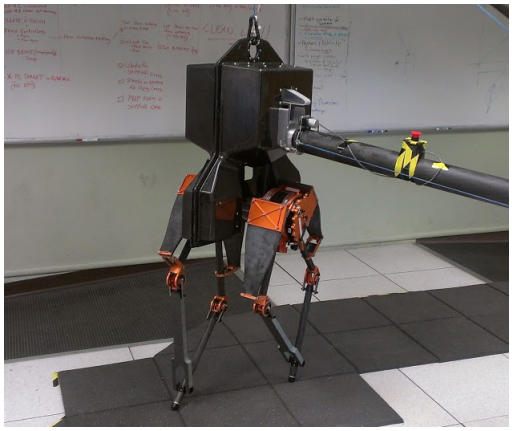

Fig. 1. The ATRIAS biped.

For our project, we made use of several URDF files of ATRIAS created by Professor Grizzle when he visited MIT. As discussed below, we made extensive modifications to these files, including the removal of several linkages and the addition of actuators.

B. Previous Approaches

Researchers at Carnegie Mellon developed a controller for ATRIAS using a spring-mass model (SMM). SMM was chosen to describe ATRIAS's locomotion dynamics because it is a good approximation for the center of mass dynamics of human running. [CITATION]

C. Drake's Atlas Controller

The MIT Robot Locomotion Group (RLG) has developed a whole body motion planner for Atlas, a humanoid robot designed by Boston Dynamics. Instead of using the position of the ZMP as the stability criterion, the planner considers the centroidal dynamics of the robot, which is the way the center of mass and angular momentum change over time. The new criterion holds for a broader group of motions than ZMP, including motions with out of plane contacts and motions with flight phases (i.e. running). It characterizes full-body motions by a single quantity — the robots momenta. The current whole body motion planner has enabled Atlas to run in simulation, but only when the actuator limits of

*This work was completed as a final project for MIT 6.832 Underactuated Robotics, Fall 2014.

[1]Laura Hallock is an undergraduate student of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA lhallock@mit.edu

[2]Veronica Lane is an undergraduate student of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA vmlane@mit.edu

the robot are removed. [CITATION Dai, 2014] RLG also developed a Quadratic Program (QP) controller for Atlas, which computes the optimal torques for each joint at each time step given the COM trajectory produced by the whole body motion planner. [CITATION]

The QP controller is a whole-body controller that uses a convex quadratic program. It solves an optimal control problem while maintaining input, contact, no-slip, and dynamic constraints. As previously stated, maintaining the ZMP ensures dynamic stability. Given the ZMP trajectory, the time-varying LQR yields the optimal cost-to-go for a simplified model. This function is then passed to the quadratic program which solves for all torque inputs for the robot dynamics.

In creating a control system for ATRIAS, we drew heavily on the ATLAS controller and walking planner already implemented in the Drake library. Our specific modifications to the code to make it work with ATRIAS are described below.

IV. ALGORITHMIC APPROACH

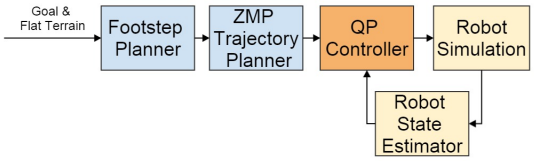

Fig. 2. The algorithmic pipeline. For our purposes in simulation, the state estimator is perfect.

A. Pipeline

We employ the same control pipeline as ATLAS to generate and execute a walking trajectory. We first generate a robot model, goal location and (flat) terrain map. These then become the inputs to the footstep planner, which outputs the (2D?) positions of each foot based on ATRIASs kinematics. This plan is then analyzed by the ZMP trajectory planner, which uses inverse kinematics to calculate the trajectory of the robots zero-moment point. This trajectory is then passed into the QP controller, which calculates the required torques at each of ATRIASs actuators at each time step. This controller is then tested on the robot in simulation to determine whether it allows the robot to successfully execute the planned trajectory.

B. The QP Controller

In calculating the joint torques, we use the QP formulation used to control Atlas in Drake and documented in [CITATION]. At each time step, we solve the quadratic program by minimizing objective function

$$V(\bar{\mathbf{x}}, \bar{\mathbf{u}}, t) + w_{\ddot{\mathbf{q}}} \|\ddot{\mathbf{q}}_{des} - \ddot{\mathbf{q}}\|^2 + \epsilon \sum\nolimits_{ij} \beta_{ij}^2 + \|\eta\|^2 \quad (1)$$

over $\ddot{q}$, β, λ, and η subject to

$$\mathbf{H}_f \ddot{\mathbf{q}} + \mathbf{C}_f = \Phi_f^T \lambda \quad (2)$$

$$\mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} = -\alpha \mathbf{J}\dot{\mathbf{q}} + \eta \quad (3)$$

$$\mathbf{B}_a^{-1}(\mathbf{H}_a \ddot{\mathbf{q}} + \mathbf{C}_a - \Phi_a^T \lambda) \in [\tau_{min}, \tau_{max}] \quad (4)$$

$$\forall_{j=\{1\ldots N_c\}} \lambda_j = \sum_{i=1}^{N_d} \beta_{ij} \mathbf{v}_{ij} \quad (5)$$

$$\forall_{i,j} \beta_{ij} \geq 0 \quad (6)$$

$$\eta \in [\tau_{min}, \tau_{max}] \quad (7)$$

These constraints ensure that the dynamics and input limits (2 and 4), no-slip contact constraints (3), and contact force constraints (5 and 6) are respected. A more complete discussion of this formulation and the symbols used can be found in [CITATION].

C. Testing Efficacy on Point-Footed Robot

In order to work within the existing Drake infrastructure — which requires 3-DOF feet in order to successfully plan footsteps — we began by giving ATRIAS large, square feet. This allowed us to get the pipeline up and running without having to completely rewrite the footstep planning algorithm. Since our ultimate goal is a point-footed robot, we will then systematically shrink these feet until the QP controller ceases to work. Pending this strategys success, we will in future iterations of the project attempt to modify the planner such that these dummy feet become unnecessary.

D. Subgoals and Planned Technical Approach

To test the efficacy of our control approach at each stage in the process, we first planned to modify the ATRIAS URDF file to successfully run in simulation and ensure that all joint limits were reasonable. We then planned to use $runPassive$ to confirm that the robot ragdolls as expected when dropped. Once this was achieved, we planned to first work to use our QP controller to balance the footed ATRIAS, then confirm that we can generate a footstep plan and ZMP trajectory, and test our controller on the resulting trajectory. We then planned explore how small we can make ATRIAS's feet, as stated above.

E. Modifications to Planned Technical Approach

We first attempted to ensure that the URDF for ATRIAS was working by starting ATRIAS from an initial position and simulating it with no torque inputs. If successful, ATRIAS bends its knees and falls to the ground. Unfortunately, even after making significant modifications to the 4Bar model of ATRIAS, the model was unable to run passively. We then decided to switch to a simpler model with the 4Bars removed. We again had to make significant modifications to the URDF, including removing coordinate frame that was included in the URDF, adding actuators at the knees, and removing unnecessary links. Initially, we did not attempt to run a balancing controller on ATRIAS, because the initial model was point footed, and therefore the balancing controller would not work. We instead worked on make Atrias walk, we initially attempted to generate the footstep planner using the lower leg as the foot link, but this caused problems with the footstep planner. To allow both the footstep planner and controller to work, we added spheres to each of the

lower-leg links to represent feet. In the URDF we added two sphere links for each leg, one for both the pitch and roll degrees of freedom. We also added small (less than 3 cm area) triangular collision groups to each foot, because the footstep planner required a support polygon in order work properly. Even after making these modifications, the footstep planner was unable to generate a footstep plan because unlike Atlas, ATRIASs hip doesnt have freedom to rotate in the yaw direction. We fixed this issue by adding another spherical link and joint ot the foot to allow yaw rotation, which allowed the footstep planner to successfully generate a plan. Once the footstep planner was working, we adjusted the gains and parameters of the QP Controller in order to make ATRIAS walk, however, ATRIAS would immediately begin to fall over. In order to isolate the source of the error, we removed the body motion controllers for the feet and torso, and attempted to make ATRIAS walk by adjusting the IKPD gains.

First, we attempted to make ATRIAS fall over rigidly by setting the output of the QP Controller and the input to the Ricatti equation to zero and adjusting the IKPD controller gains. No matter how the gains were adjusted ATRIAS would bend its knees each time it fell. Even after making these adjustments, ATRIAS would still fall over immediately.

We then modified the URDF to add actuators to each of the foot joints, to allow the QP controller to control the roll, pitch, and yaw of the foot. This allowed ATRIAS to balance for a longer period of time, but it still fell over before taking a single step.

Since we could not get the robot to balance with very small triangular support polygons, we modified the contact groups in the URDF so that ATRIAS had large rectangular feet. Now that ATRIAS had feet, we decided to work on making ATRIAS balance, before trying to make it walk. The balancing test drops ATRIAS from a certain height above the ground, and attempts to make ATRIAS balance with its COM at a given location. During this test, we discovered that the URDF had no actuator at the hip, which was causing the balancing controller to fail. After adding the actuator at the hip, the robot still fell over while balancing, but was able to balance without falling over, but the error in COM location was about 15 cm, while less than 2 cm was desired.

Given that the balancing controller worked for other bipeds, including Atlas and Hubo, we concluded that the failure of the balancing controller was due to problems with the model of the robot. We made further modifications to the URDF by removing actuators that did not move anything when viewed in the inspector and fixing the corresponding joints. This yielded slight (2-3cm) improvements in the error. We then removed the yaw degree of freedom in the foot, which resulted in significant performance improvement. The error was approximately 5 cm, as opposed to 15 cm with this degree of freedom.

V. PROJECT STATUS

Due to the time-intensive modifications we had to make to the ATRIAS URDF file, we have not yet been able to create a working QP controller that allows ATRIAS to walk. Currently, we have successfully generated footstep plans and are close to a working balancing controller, but we are currently debugging an issue that prevents both of these processes from successfully operating on the same URDF file. Namely, we can only achieve a reasonable result while balancing when we fix the yaw angle of each of ATRIASs feet. At the same time, the footstep planner explicitly plans for each foots position in roll, pitch, and yaw, and thus requires ATRIAS to have each of these degrees of freedom: otherwise, ATRIAS must bend its knees and apply strange torques to its hip actuators in order to keep the foot pointed forward, causing ATRIAS to infeasibly lower its center of mass and ultimately fall over.

We are currently evaluating our balancing controller with respect to the threshold used by ATLAS: i.e., we consider ATRIAS balanced if it remains within 2 cm of the goal fixed point after 2 s of simulation. While we have not yet been able to remain within this threshold, we have achieved an error of approximately 5 cm. At this point, when ATRIAS is dropped, the torso first leans off to one side, then appears to stabilize very precisely. We hypothesize that by further adjusting the gains to mitigate this oscillation, we can achieve better balancing precision. We are also investigating the possibility that there are still more significant problems with the URDF file that are preventing us from accurately balancing with a fixed foot yaw or from balancing at all with yaw as a DOF.

VI. LESSONS LEARNED

Lessons learned.

VII. FUTURE WORK

Further work will attempt to successfully implement the QP Controller for the ATRIAS biped. In order for ATRIAS walk with feet, it first must be able to balance. As previously stated, allowing the feet to rotate in the yaw direction causes the balancing controller to fail. However, removing this degree of freedom causes the footstep planner to fail. We will modify the simulated ATRIAS model to allow both the balancing controller and the footstep planner to succeed, while still accurately approximating the dynamics of ATRIAS. Once both the balancing controller and footstep planner work, we will adjust the gains and paramaters of the QP controller to make ATRIAS walk over flat terrain.

After ATRIAS successfully walks with feet, we will incrementally decrease the size of the feet until the controller fails. For each iteration, we will adjust gains and controller parameters to make ATRIAS walk over flat terrain. When the size of the support polygon is negligible, the ATRIAS model will approximate a point-footed robot.

VIII. CONCLUSIONS

Ultimately, we were unable to progress as far toward our goal of creating a QP controller for a point-footed biped as we would have liked. By exploring the the system's failure modes, however, we learned a great deal about the intricacies of each stage in the pipeline, from URDF modeling, to

footstep planning, to the QP controller itself. We gained significant intuition for what might cause a controller to fail, and we look forward to continuing this project over IAP.

ACKNOWLEDGMENT

We would like to thank visiting Professor Jessy Grizzle for the ATRIAS URDF files we used in our project, as well as Professor Russ Tedrake and the entire Drake development team. Additionally, we would like to thank Dr. Scott Kuindersma, Andres Valenzuela, and Robin Deits of the MIT Robot Locomotion Group for all their help problem solving and debugging.

REFERENCES

[1] Russ Tedrake, Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2014, http://drake.mit.edu.

[2] S. Kuindersma, F. Permenter, and R. Tedrake, An efficiently solvable quadratic program for stabilizing dynamic locomotion, in Robotics and Automation, 2013. ICRA 13. IEEE International Conference on, 2013.