Optimally Energy-Efficient Bipedal Gaits on Varying Slopes

Nicolas Champagne-Williamson (nac52)

Abstract— **The purpose of my research is to discover optimally efficient control trajectories for the Ranger robot over a range of slopes and to use these to improve the robots robustness while walking. Optimal control solutions are found by the SNOPT optimization package in Matlab for slopes of -10 to 10 degrees. The estimator uses the inertial measurement unit and robot kinematics to calculate the approximate ground slope in the world frame. The trajectories are discretized and approximated using a finite state machine paradigm with parameters set according to derived functions of the measured slope. However, due to physical limitations of the robot, this technique was unable to allow the robot to walk up even a modest 2 degree slope.**

I. BACKGROUND

Walking robots are awesome! From a scientific perspective, their artificial gaits can help researchers investigate the dynamics of walking and control. From an engineering perspective, robots with legs may be able to travel in places inaccessible to wheeled robots, such as up stairs or through rubble during search and rescue operations. These motivations have encouraged the creation of innumerable examples of robot walking, and yet these robots continue to perform below the level required to play useful roles in society. For example, Cornell's Ranger robot has reliably walked over half a marathon on one charge... around Barton's indoor track. Give it a slope of more than 1 degree to climb and it simply can't do it. There is one scary section of track where Ranger consistently falls over, dubbed "Mount Barton", and we take care to always avoid this section of track, even though in reality the slope there is so slight that most runners don't even realize it exists!

Currently, the most common way to quantify stability of bipedal walking is with the Zero-Moment Point (ZMP) criteria which, if met, ensure dynamic stability. However, the gaits generated with ZMP satisfaction are often slow and awkward, far from the smooth and controlled walking of animals and humans, not to mention use about 20 times the energy. Additionally, while ZMP is sufficient for stable walking, it is not necessary; humans actually violate ZMP criteria during toe push-off.[3] The Ranger robot took a different approach, focusing on passive dynamics principles to create energy efficient and smooth motions. Though it is nominally powered (and must be for level ground or uphill, to both fight the losses due to gravity, friction, and collisions), much of the gait is unpowered and left to freely swing. [2]

The general goal of my work is to make Ranger more stable over a wide range of slopes. Optimal control trajectories (motor torque profiles) were found offline using dynamics equations of the robot, structural and periodic constraints, and the SNOPT (Sparse Nonlinear Optimization) package in Matlab. A simulation of the robot, developed by fellow research Pranav Bhounsule, uses these current trajectories to see how the robot would walk given those controls. The simulator is thought to be very accurate and has been shown to match well against empirical walking data on level ground. Target optimal hip velocities during the leg swing phase and ankle positions during pushoff were taken, and functions of slope were fitted against them. These results were discretized into a state machine for use on the robot, with the state variables as a function of slope. Using an inertial measurement unit, the robot accurately senses the current slope of the ground with respect to the world frame and sets its optimal trajectory accordingly.

A. The Cornell Ranger Robot

The Ranger Robot, which our lab lovingly calls a 4-legged biped, actually consists of two pairs of legs moving together (see Figure 1, page 2). This creates the illusion of biped walking with side to side stability, but no fore-aft stability. There are four actuators total: 1 to swing the hip, 2 to flip the pairs of ankles up and down (giving foot clearance), and 1 to steer the robot by twisting the inner pair of legs. Each motor is controlled by a separate ARM7 microcontroller built onto what we call a satellite board. These boards run a 2khz scheduler to control the motor and read sensor information, and communicate with each other as well

N. Champagne-Wiliamson is a student Computer Science, Information Science and Cognitive Studies, Cornell University, NY 14853, USA nac52@cornell.edu

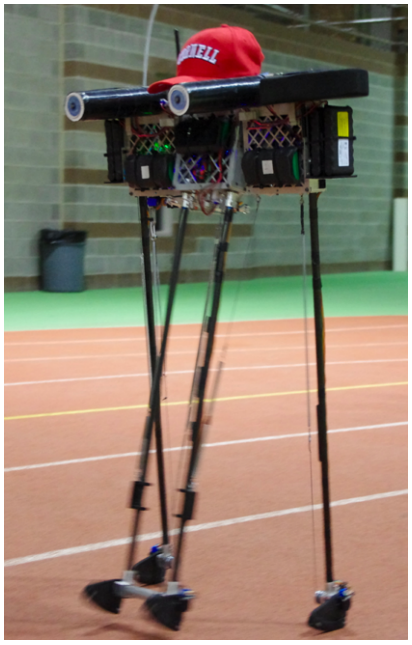

Fig. 1. Cornell's Ranger Robot, built by students in Prof. Andy Ruina's Biorobotics and Locomotion Laboratory. *Photo credit: Nicolas Champagne-Williamson*

as the central ARM9 controller - the main brain - over a Controller Area Network (CAN) bus. The main brain runs a 500hz scheduler running a finite state-machine, and sends sensor readings to the monitoring laptop computer over Bluetooth. The main sensors on Ranger are optical motor encoders, magnetic encoders at the joints, foot-contact sensors, an Inertial Measurement Unit (IMU), motor current sensors (for motor control), RF receiver for steering commands, buttons for changing overall state (walking, calibrate, standby), and safety limit switches for joint angle limits. The general gait of Ranger is split into 4 parts: Double Stance (DS), Pushoff, Single Stance (SS), and Heelstrike.

1) Single Stance (SS): During single stance, one foot is on the ground, and the other leg is swinging through the air with its foot flipped up to give ground clearance. The main control during this phase is hip torque at the beginning of the swing to get the leg to swing at the desired speed, depending on current robot velocity.

2) Pushoff: During pushoff, the back ankle motors cause the feet to flip down and give a toe pushoff. When optimizing control for energy efficiency it turns out that pushoff of the back foot occurs slightly before the impact of the front foot during heelstrike, in what is called a pre-push. Note that the feet pushing off don't actually cause the feet to be lifted off the ground just yet - otherwise the robot would be running!

3) Heelstrike: After pushoff, the swing leg flips down its feet, and it collides with the ground causing a heelstrike. Heelstrike is detected by a deformation sensor in the foot. There is often a swing leg retraction before this phase.

4) Double Stance (DS): After heelstrike, both feet are on the ground at the same time. This phase is short but necessary in walking, because otherwise we would be running, and also because it is the only time we know the robot's state without the IMU data, and so we can reset the IMU to get rid of drift in readings. At the end of this phase the back feet flip up, giving ground clearance, and we enter SS.

II. APPROACH

A. Gait Optimization

The SNOPT package solves large-scale nonlinear constraint optimizations. The way it generally works is by using the Sequential Quadratic Programming (SQP) method, processing QP subproblems in search directions which minimize a linearly constrained quadratic Lagrangian by utilizing the computed Jacobian and Hessian of the function matrix. As input to the program we provide 1) the objective function, 2) auxiliary functions to be constrained, 3) a set of linear and nonlinear constraints on those functions, and 4) the function parameters to tune. The optimizer operates over one simulated step of the robot.

The objective function we try to minimize is a measure of walk efficiency called the specific energy of transport, which is essentially how far the robot can walk for a given amount of energy per weight of the robot. The equation for cost of transport is:

$$c_t = \frac{energy}{weight * distance} \tag{1}$$

$$c_t = \frac{\int_0^t (\sum motorpower)dt + overheads * t_{step}}{m * g * steplength} \tag{2}$$

Auxiliary functions describe the robot's walk, and are given by a dynamics model of Ranger which very closely approximates the physical robot.

Constraints on the system include periodic constraints, such as matching velocity at the beginning and the end of the step, and structural constraints, such as that the foot height can only be positive (above the ground) and that the hip should be above the feet, etc. Lack of structural constraints allows the optimizer to produce some amusing results, such as finding that the robot walks upside-down, or flies through the air. A problem I personally ran into initially occurred because

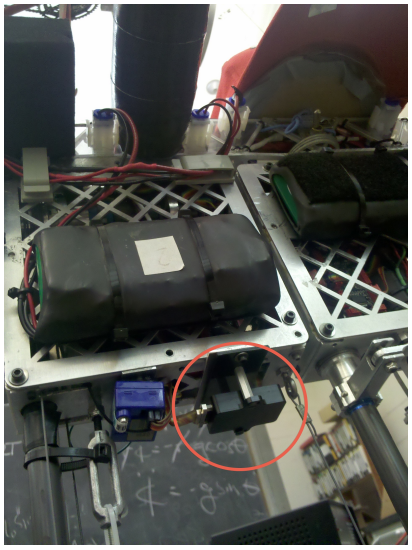

Fig. 2. The Inertial Measurement Unit (IMU, circled in red) is used to estimate the ground slope.

the foot of the robot is round, and is represented in the simulator as a circle. However, there were no constraints on what part of the circle the foot is, and so animated solutions clearly show the robot making use of 'invisible' portions of its foot to gain extra height!

For the optimization parameters, we split the motor current control into piecewise linear functions, and use the points describing that function. Both the hip and ankle trajectories use 8 control points during SS/swing phase, and 4 during DS. A loop runs the optimization for each desired slope and outputs the optimized parameters. For the initial parameter values (the 'best guess') we use the results of the previous optimization. For example, at 5.5 degrees, the results for the 5 degree case are used and the optimization start its search from that state. The code takes over an hour to run one optimization: 15 minutes to calculate function derivatives and an hour or more doing search iterations - even with compiled C code instead of pure Matlab.

B. Slope Sensing

Ranger possesses an Inertial Measurement Unit (IMU) that gives both euler angles and angular rates at 200Hz. The IMU is mounted on the starboard side of Ranger, coaxial to the outer set of legs. Our convention is that 'roll' is what is normally called 'pitch', because that is the axis around which a bike's wheel rolls. The IMU is useful because it gives us the angle of the outer leg with respect to the world frame. However, this knowledge is not enough to estimate ground slope. Upon heelstrike, we know that both feet are on the

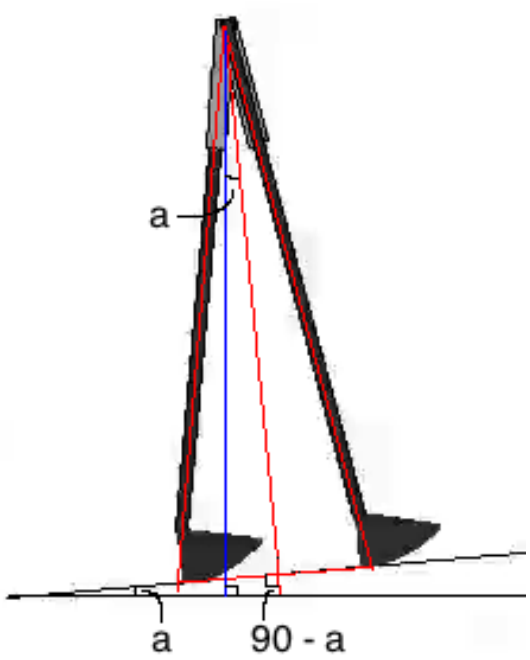

Fig. 3. The robot at heelstrike. The IMU roll angle of the outer leg is given by the blue line, and the estimated angle of the outer leg assuming horizontal ground is given by the red line.

ground. Using this info and the joint angles of the motors (ankle and hip), the robot makes its own estimate of the angle of the outer legs with the assumption that it is on level ground (0 degree slope). The difference between this estimation and the IMU roll data (plus a small offset - the IMU is not perfectly square to the robot[1]) is the local ground slope (see figure 3). This information is vital in setting the optimal control trajectory.

C. Finite State Machine Approximation

However, we can't simply run the generated open-loop current trajectory because the robot would not be able to react to any disturbances during the step. The solution is to discretize the gait into a finite state machine that approximates the original function, or what the original trajectory was trying to accomplish. For example, instead of controlling for current, the hip swing tries to achieve a target leg velocity or hip angular rate. Similarly, instead of an ankle pushoff current, a target ankle position is used instead. From the data that SNOPT and the post-processing simulator

[1]The offset was found by keeping the robot exactly perpendicular using an electronic level, and then recording the IMU roll output

gave me, I created plots of both the required ankle target angle and the hip swing velocity as functions of slope.

III. RESULTS AND DISCUSSION

A. Gait Optimization

The optimizations found valid solutions for all slopes from -10 to 10 degrees. In simulation the robot found ways to power up 10 degree slopes and waltz down -10 degree slopes without tripping. It is likely that the simulation could be pushed even further, but the optimizations took a long time and I was more interested in implementing the changes on the robot.

1) Downhill: In walking downhill we can see some interesting changes to gait. One of the main problems for this robot walking down steep slopes is that it must rid itself of excess energy and slow down, otherwise it will speed up until it can't move its legs fast enough to catch itself, and falls over. If we look at the ankle currents in figure 7 in the appendix (the motor current approximates), we can see that as downhill slope increases the pushoff disappears and then reverses. The effect of this can be clearly seen in animations, which show the ankle trying to stop the center of mass from going over the stance leg, and so slow down the robot.

If we look at the hip currents going downhill in figure 8 (again, as an approximation of hip torque), we see that the swing torque decreases - without a large loss to angular rate - and a strange spike at the very end of the swing before heelstrike. The current going to the motor is in the opposite direction of the forward swing, and so it seems (and animations confirm) that the robot is retracting its legs quickly to strike the ground. I posit that the purpose of this jerk is to create large ground collisions with the foot, and therefore incur large collisional energy losses. This helps to remove some of the energy gained by going downhill, and aids the robot at slowing down. Additionally, step times become slower and step lengths become longer, further aiding the robot to slow down.

2) Uphill: In uphill walking the robot has to fight gravity and maintain positive velocity. Looking at ankle current (figure 7) demonstrates that as slope increases, pushoff power and angle both increase as well. The step lengths decrease and step times are shorter. All of these serve to put energy into the system by lifting the back leg higher to help get the center of mass over the stance foot and keep the robot moving forward.

The hip trajectories at higher uphill slopes are no surprise either. There is a clear increase in swing torque/current, which serves to keep the angular rate of the hip at around the same speed, increasing it slightly with slope.

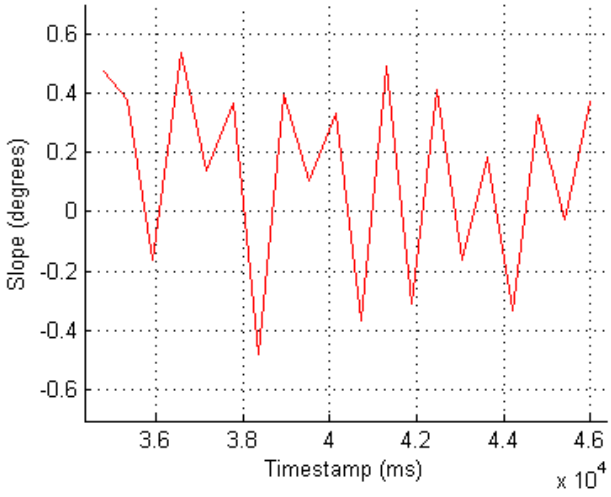

Fig. 4. Measurement of slope while walking on level ground. Actual slope varied between ± 0.5 degrees. $\bar{x} = 0.12, s = 0.33, range = 1.01$

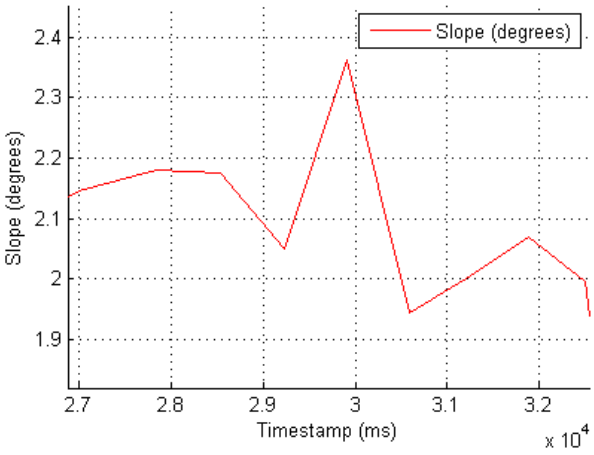

Fig. 5. Measurement of slope while walking on a slight uphill. Actual slope went from 2.4, to 2.6, then back down to 2.4 degrees. $\bar{x} = 2.14$

B. Slope Sensing

The slope sensing capabilities of the robot were adequate for responding to large disturbances (see figures 4, 5, 6). It is difficult to tell whether the deviations in slope on a 0.5 degree scale were even errors, because no floor is perfectly flat and I didn't follow along the floor with a level the whole route to find precise actual slope measurements. The data has a standard deviation of approximately 0.3 (see figure 4) which is

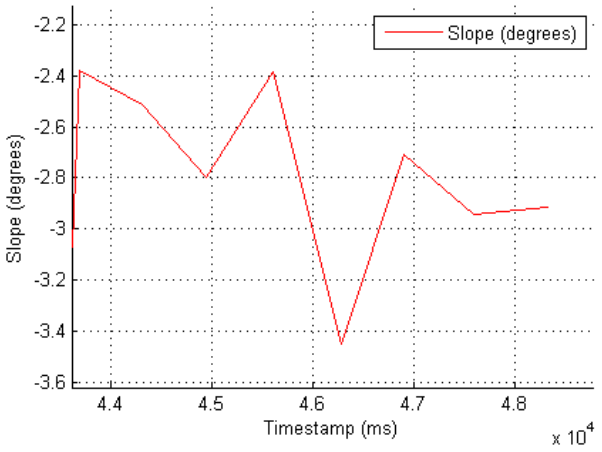

Fig. 6. Measurement of slope while walking on a slight downhill. Actual slope went from -2.4, to -2.6, then back down to -2.4 degrees. $\bar{x} = -2.81$

enough accuracy for setting state machine parameters and allowing for reactions to larger and more consistent slope disturbances. The measurement also doesn't have to be the control itself is mildly stable, and so small deviations from optimal policies keep the robot walking - just not with optimal energy efficiency.

C. State Machine

The ankle pushoff data had a linear relationship with slope, and so it was approximated with the line:

$$ankleangle(rad) = 0.26 + 0.045 * slope(degrees) \quad (3)$$

The hip swing velocity appeared to be quadratically correlated to slope, and so quadratic regression was used to generate the following curve:

$$hiprate(rad/s) = 1.9 - 0.054 * slope + 0.032 * slope^2 \quad (4)$$

These functions were utilized at every heelstrike to set the parameters of the next step.

1) Falling Down: Unfortunately there is little success to report with actually implementing this control on the physical robot. The equations were added to the control and set the target hip velocity and ankle pushoff angle at every heelstrike according to equations 3 and 4 above. Testing of the system showed the expected normal gait on level ground. However, the robot failed to walk up even a 2 degree slope. Checking the state machine parameters it was using showed the theoretically correct values given the slope measurements the robot was sensing.

So what was going wrong? It turns out that while we tell the robot to flip its foot down by 0.35 radians, it can't physically do that with the motors it has and it's current limits. This shows a discrepancy between the simulation and the physical robot. Possible sources of the discrepancy are from the changes recently made to Ranger to allow it to do it's record-breaking 40+ mile walk, including more batteries and extra weight. The 6-amp ankle motor limit is simply unable to lift the robot anymore, and without this boost to height and leg angle, is unable to walk even seemingly mild 2 degree slopes. The increased weight also makes it so that the same movements have to get a much higher center of mass over the stance leg. Even with hacks such as "At pushoff, flip down as much as possible", "After swinging the hip, hold the leg out" and fast initial velocities, the robot still doesn't get its center of mass over the hump. It seems as though it is impossible for Ranger to climb a 2 degree slope in his current form.

IV. CONCLUSION AND FUTURE WORK

A. Conclusion

Using optimal trajectory control is an interesting area of research. It 'lightens the load' on control schemes such as PID or other methods by keeping closer to the optimal solution based on external sensor feedback. In this paper I have shown that it is theoretically possible to find gaits that optimize energy efficiency and manage to walk up and down some pretty steep slopes. Additionally, we now have an accurate algorithm for measuring slope while the robot is walking, by combining IMU data with robot kinematics. However, the physical limitations of the robot suggest that this system is not physically realizable for this purpose. But this does not mean we should ignore optimal trajectory control as a whole.

B. Future Work

Future work will need to investigate if perhaps rewriting the entire state machine would allow us to more closely approximate the output of the gait optimizations. Additionally, the simulation constants and equations will need to be changed to mirror the new upgraded Ranger before any output can be trusted and used to generate controls. Finally, I would like to try this approach with other facets of Ranger's control (or perhaps other robots!) to keep him walking more efficiently over a wider-range of conditions.

V. ACKNOWLEDGMENTS

I'd like to thank Andy Ruina, Professor of Theoretical and Applied Mechanics at Cornell University and my research advisor; Ashutosh Saxena, Professor of Computer Science/Mechanical Engineering at Cornell University; Pranav Bhounsule & Anoop Grewal, both wonderful Ph.D. students in the Biorobotics and Locomotion Lab; Jason Cortell, lab manager of the Biorobotics and Locomotion Lab.

REFERENCES

[1] Bhounsule, Pranav. Extensive discussions and adaptations of previous work.

[2] Collins, S., Ruina, A., Tedrake, R., Wisse, M. Efficient Bipedal Robots Based on Passive-Dynamic Walkers, *Science*, 307, pp1082-1085. February 2005.

[3] Pratt, J., Tedrake, R. Velocity-Based Stability Margins for Fast Bipedal Walking, *Fast Motions in Biomechanics and Robotics*, Vol. 340, pp299-324. 2006.

VI. APPENDIX

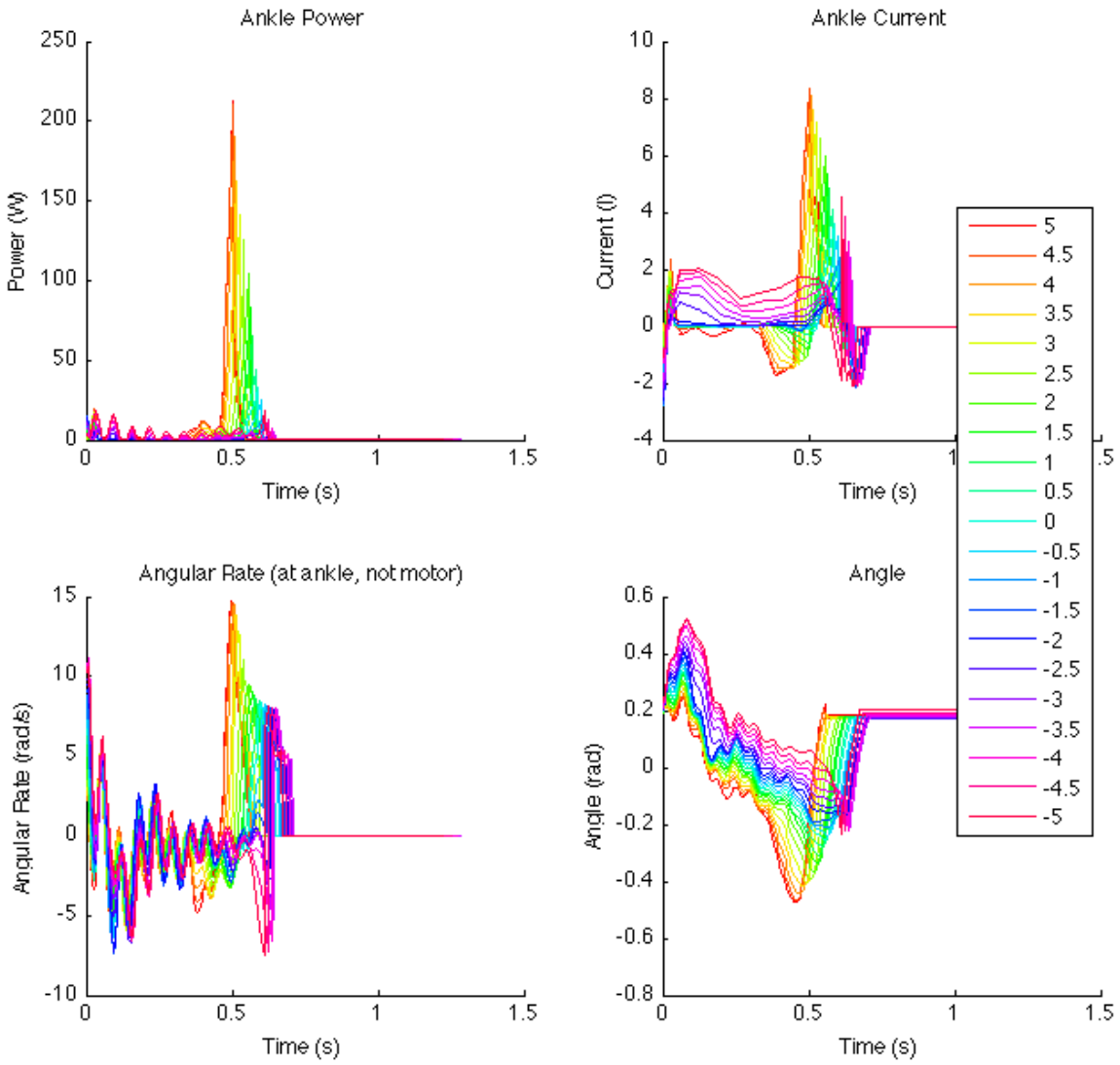

Fig. 7. Optimal current/power, position, and velocity profiles for slopes from 5 degrees uphill to 5 degrees downhill (negative value).

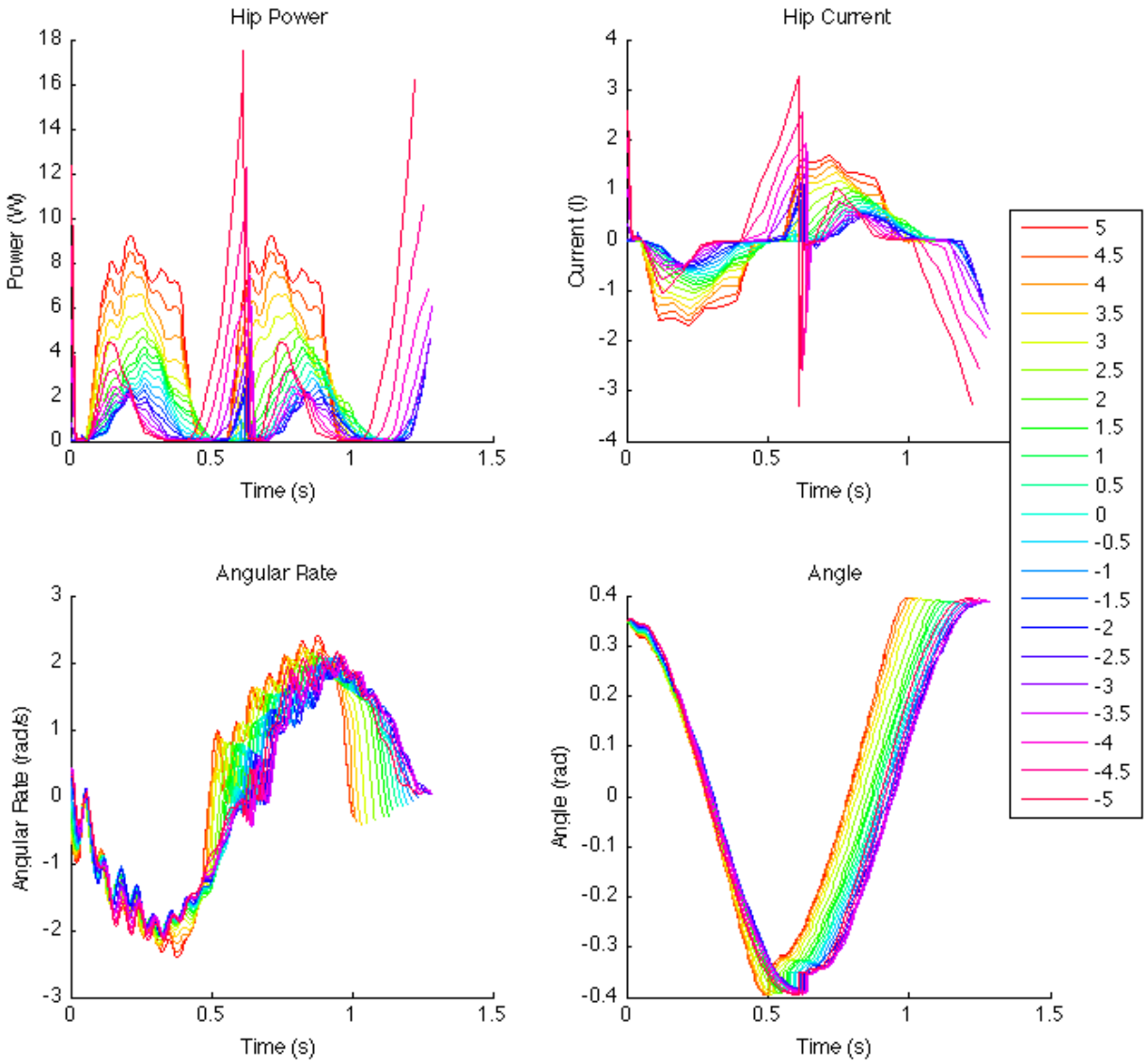

Fig. 8. Optimal current/power, position, and velocity profiles for slopes from 5 degrees uphill to 5 degrees downhill (negative value).