Millirobot Simulation

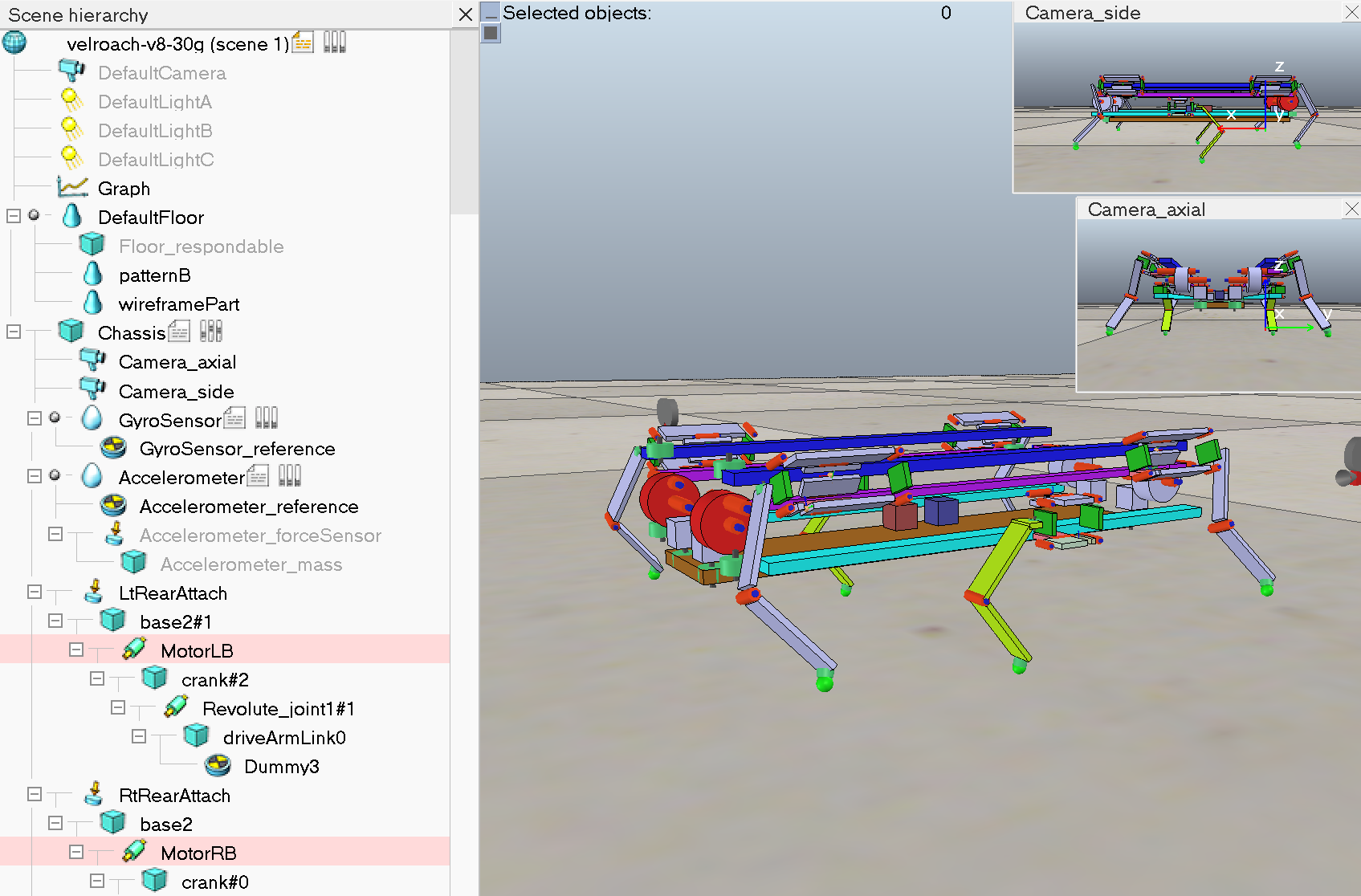
Allen Zhu

Advisor: Prof. Ronald Fearing

May 2017

**Abstract**. As we develop more complex control strategies, especially strategies using machine learning techniques, we may wish to conduct a large volume of preliminary experiments in simulation. In this work, I developed and validated a simulation of the VelociROACH, a 10 cm long hexapod that runs with an alternating tripod gait. I matched parameters affecting robot dynamics including size, mass, and relative leg stiffness . Like the real VelociROACH, the simulation walks with an alternating tripod gait and undergoes roll oscillation modulated turning. However, the simulation is ten times slower than real time and does not currently work at high frequencies (over 5 Hz).

## Model development



**Figure 1**: V-REP scene displaying the VelociROACH model. The user interface can also show camera views, graphs, and the scene hierarchy, which is a complete listing of model components.

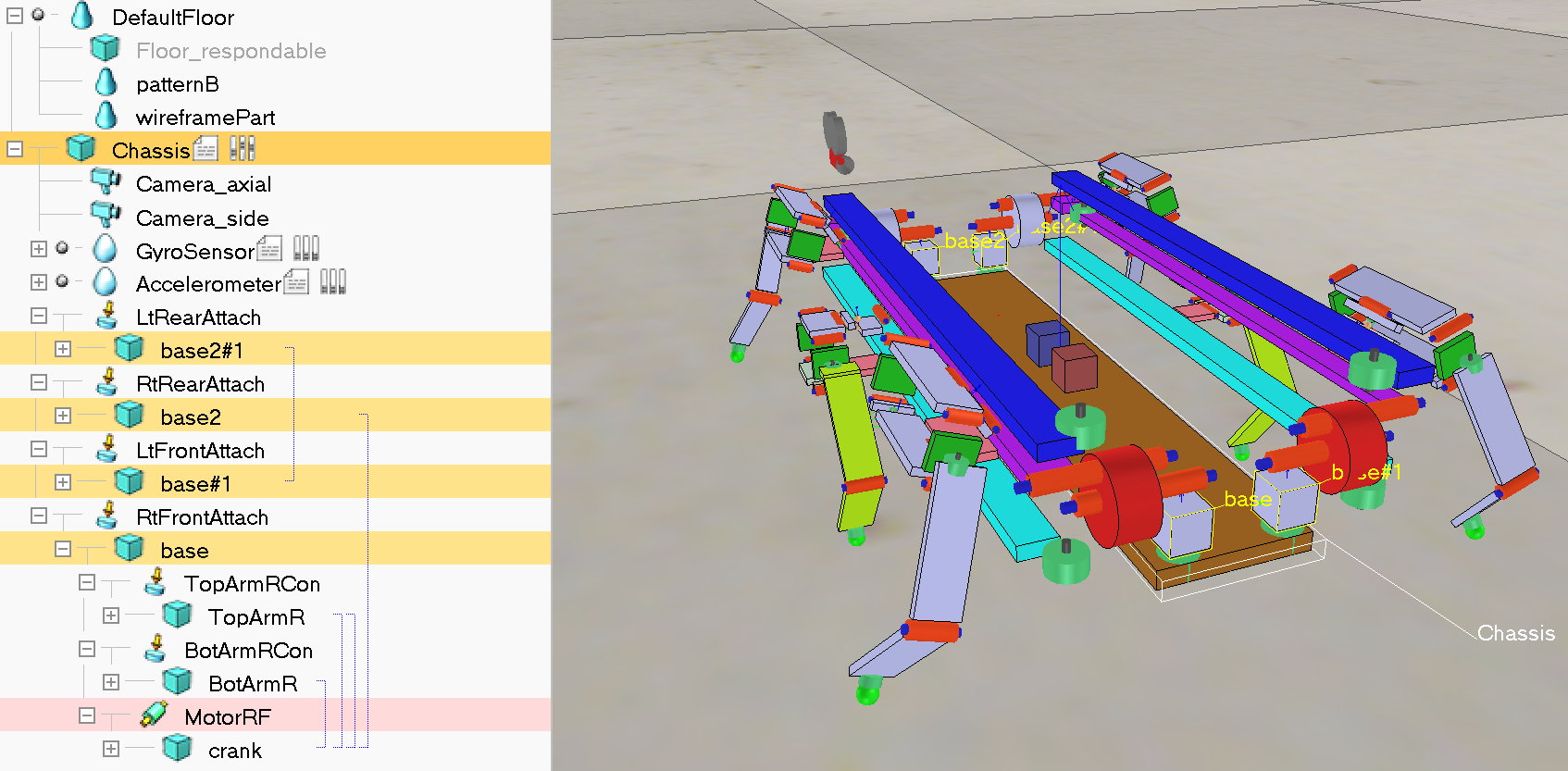
I started with an existing but not yet validated OpenROACH model using the V-REP robotics simulator with the ODE physics engine. V-REP has several desirable features, including a preexisting component library, cross-platform support, ROS integration, and active support forums. V-REP also supports loop closure dummies for VelociROACH’s planarizing four-bar linkages. However, there are some disadvantages. V-REP only supports rigid body dynamics, so our model approximates VelociROACH’s compliant ‘C’ shaped legs using joints in spring-damper mode. V-REP stores models in an undocumented binary file format, so it is difficult to programmatically modify or generate models. Finally, V-REP is not typically used for millirobots, and much of the existing documentation and suggested parameters are geared towards larger masses.

Each side of the model has two motors, such as MotorLB and MotorLF, which are coupled using the same drive arm. Motors are implemented in V-REP as dynamic joints in torque/force mode with motor enabled and control loop enabled. The control mode is PID control, with =0.005 and ==0. Motors have a maximum speed of 1145.9 deg/s. This is a VREP limitation; it is not possible to set a higher motor upper velocity limit value. In particular, setting higher values using the Lua API fails silently. In contrast, maximum torque can be effectively unlimited (20 N\*m).

Six leg joints model the compliance of VelociROACH C-legs. These joints are also implemented as dynamic joints in torque/force mode with motor enabled and control loop enabled. However, the control mode is set to spring-damper mode, with K=0.0005 N and C=1e-6 N\*s.

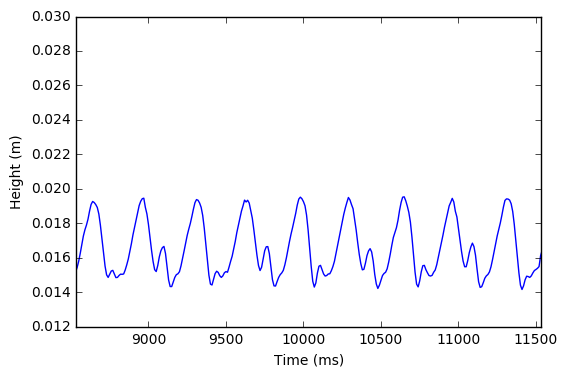
The model was at 10x scale, so the first task was to scale the model to the desired dimensions of 0.1 x 0.02 x 0.0025 m. In V-REP, this can be done by selecting all elements with Ctrl-A and then individually deselecting external elements, such as the floor. I verified that that mass, torque, and rotational inertia scaled proportional to , , and , respectively. It was also necessary to reduce the ODE time step from 1 ms to 0.1 ms; otherwise, the scaled robot would not move. The ODE time step is the time step used for dynamics calculations, so a smaller time step increases numerical stability at the cost of increased simulation time. The ODE time step is distinct from the simulation time step dt, which determines the frequency of simulation passes, i.e. main script execution which includes reading sensors, calculating controls, and writing telemetry. It is also different from simulation passes per frame parameter, which determines the number of simulation passes for one rendering pass.

Next, we needed to increase the mass of the model to match the weight of the VelociROACH, 30 g. Unfortunately, V-REP does not provide any graphical or programmatic way to scale all masses simultaneously, and the model files are stored in an opaque binary format. Increasing only the chassis mass might be a good approximation, but it just results in the entire robot collapsing, no matter how stiff the joints are. The iterative dynamics solver is limited in the mass ratios it can simulate because a very large chassis mass effectively overwhelms the normal force from the supporting leg. Therefore, increasing the mass requires manually editing each shape to ensure that the masses of any two dynamically linked shapes (shapes directly connected with a dynamic joint or force sensor) are within a factor of 10 (Figure 2). It is also possible to connect the two shapes through shapes with intermediate masses, though this adds to the model complexity.

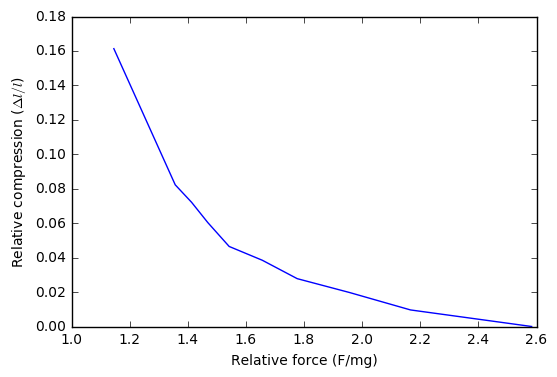


**Figure 2**: Chassis connections. The Chassis is connected to four base cubes (highlighted in yellow), which are in turn connected to motors, legs, and other structural elements of the model. The simulation can be numerically unstable if two dynamically linked shapes have very different masses.

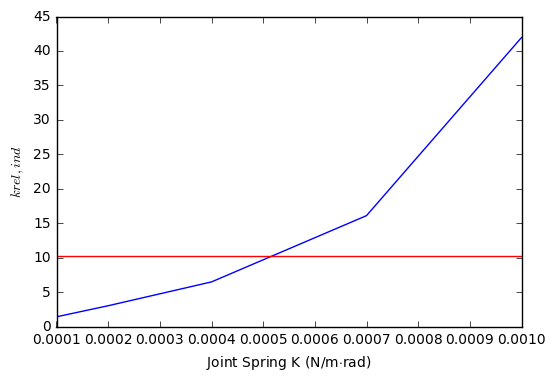
Next, I analyzed VelociROACH gait dynamics using the spring-loaded inverted pendulum (SLIP) model. Holmes et. al. found that “motion is typically restricted to a low-dimensional subspace within a high-dimensional joint space,” so that under a preferred posture, robot dynamics can be modeled as a single inverted pendulum that compresses under ground reaction forces and produces a constant linear response [1]. The simulated VelociROACH height time series is sinusoidal, consistent with the SLIP model (Figure 3). The relative force vs. relative compression relationship is somewhat linear but deviates at extremes because the model uses a rotational joint spring instead of a linear spring (Figure 4). The joint springs are the only source of compressibility in the model, so relative stiffness has a direct linear relationship with joint stiffness (Figure 5). I tuned relative stiffness per leg to 10.3, which matches the VelociROACH and is similar to most animals [2].



**Figure 3**: z motion of the simulated Chassis, which is sinusoidal to a first approximation.



**Figure 4**: Relative force vs. relative compression in the 10x scaled model. Static test with varying robot mass. Note that force and compression are inversely correlated, which is not consistent with the SLIP model.



**Figure 5**: Spring K vs in the 10x scaled model. Individual-leg SLIP stiffness depends largely on leg joint spring constant. Red line indicates the target = 10.

|  |  |  |
| --- | --- | --- |
| Parameter | Real robot | Simulation |
| Size | 0.1 x 0.02 x 0.0025 m (overall) | 0.1 x 0.02 x 0.0025 m (chassis) |
| Mass | 0.03 kg | ~0.028 kg |
| Ground reaction forces | 0.1 N each | ~0.1 N each |
| Motor frequency | 0-45 Hz | 0-3 Hz |
| Motor torque |  | Up to 0.003 N\*m |
| Leg inertia | 30 g\*mm^2 | ~300 g\*mm^2 |
| Center of mass | Higher | 0.015 m |
|  | 10.3 | 10.3 |

**Table 1**: Summary of parameters analyzed.

After ensuring that the size and mass of the model were accurate, I investigated ODE parameters that can be tuned to minimize computation time while maintaining simulation accuracy. ODE can numerically solve the constrained dynamics equations using either an exact simplex method or an iterative approximation method, known as quickStep. The runtime of the exact method is cubic in the number of constraints, whereas quickStep is linear, so for large models quickStep is about 10 times faster [3]. Using quickStep, the main accuracy parameter is the number of quickStep iterations per ODE step, where more iterations increase solution accuracy at the cost of increased simulation time. The Constraint Force Mixing (CFM) parameter improves numerical stability for near-singular systems. In theory, the CFM and the Error Reduction Parameter (ERP) make collisions behave like a spring-damper system, but the spring is sufficiently stiff () that it is indistinguishable from numerical error. Neither parameter had a significant effect on simulation accuracy.

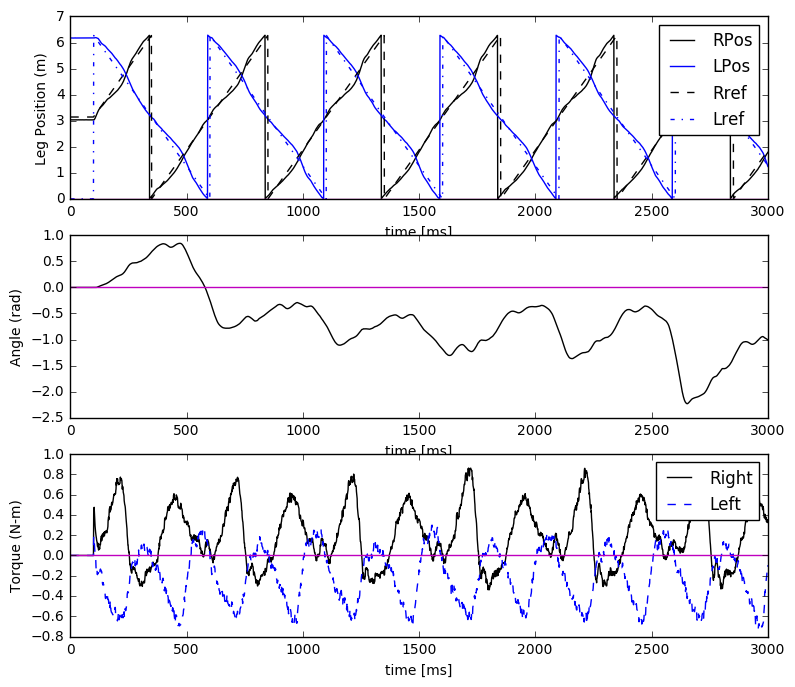
|  |  |  |
| --- | --- | --- |
| Parameter | Value | Notes |
| ODE time step [s] | 1e-4 | Model doesn’t work at lower settings, and higher settings are much slower |
| dt | 10 ms | Smaller dt improves control/telemetry granularity at the cost of simulation speed |
| Use ‘quickStep’ | True | Fast, approximate, iterative solver |
| QuickStep iterations | 100 | Smaller settings are faster but less accurate |
| Internal scaling | 1 | Or 10, no significant impact |
| Global ERP | 0.6 | No significant impact |
| Global CFM | 1e-5 | Must be nonzero for numerical stability, no significant impact otherwise |
| Simulation passes per frame (ppf) | 1 | Larger values are slightly faster but significantly reduces framerate |
| Floor Friction | 1 | Amount of static friction, no relation to the coefficient of static friction |
| Floor linear, angular damping | 0 | Set between 0 and 1, higher values simulate inelastic surfaces e.g. carpet |
| Maximum contacts | 12 | Higher values have no significant impact |

**Table 2**: List of global simulation parameters for ODE. Other physics engines offer different parameters, such as more fine-grained material properties and friction models.

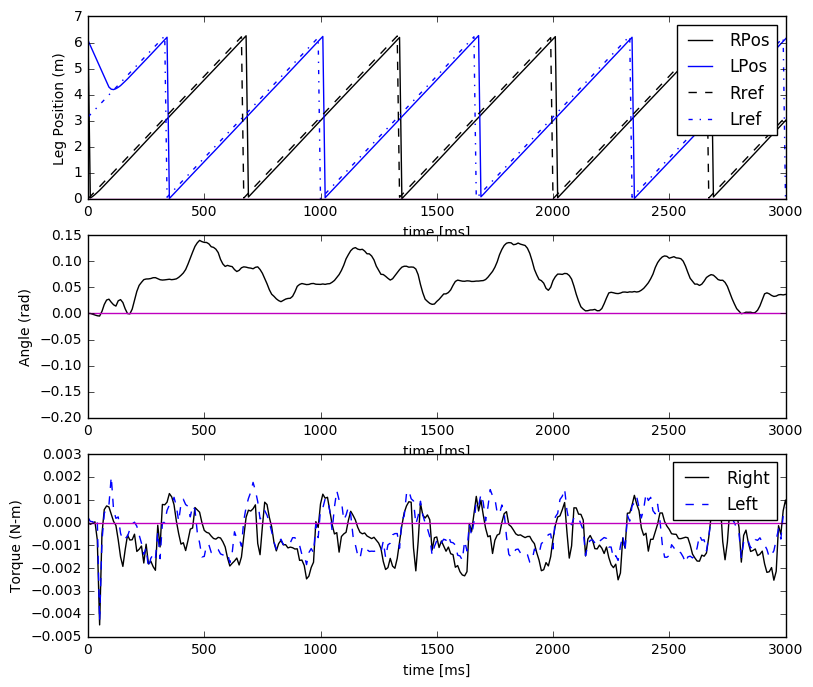
To efficiently conduct experiments in simulation, I rewrote the Lua script that runs the robot and created a test automation framework. The framework makes it possible to run the simulation interactively in the visual interface or to test many combinations of parameters overnight using the command line. Simulation parameters can be set from either the V-REP script simulation parameters dialog or via command line arguments. The Lua script reads telemetry from sensors and can also obtain any desired ground truth data from the V-REP simulator, allowing direct readings of positions, velocities, forces, etc when needed. It outputs the data in three ways: printed in a console while the simulation runs, plotted onto an interactive graph in V-REP, and written to a telemetry file. I extended the existing telemetry format to record additional simulation data while maintaining backwards compatibility. Finally, I refactored and extended existing code that processes telemetry files into a Python module using the pandas time series library, making it easy to graph relevant data [4].

## Results

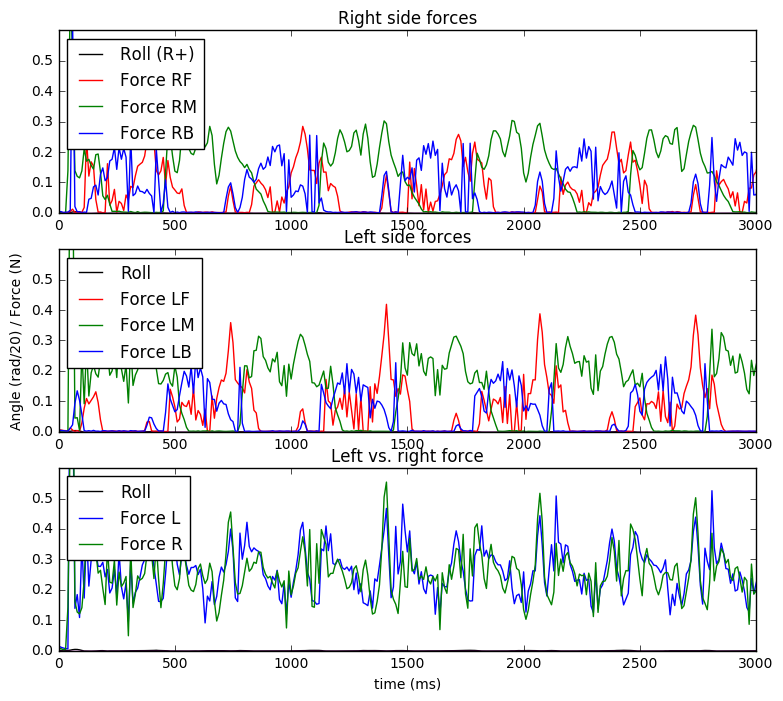
Leg positions match commanded leg positions, and torques and ground reaction forces are reasonable. Depending on parameters, there may be some drift in yaw angle (Figure 7, Figure 8).



**Figure 6**: Position, angle, and torque telemetry for the real robot at a 180 degree offset, for comparison.

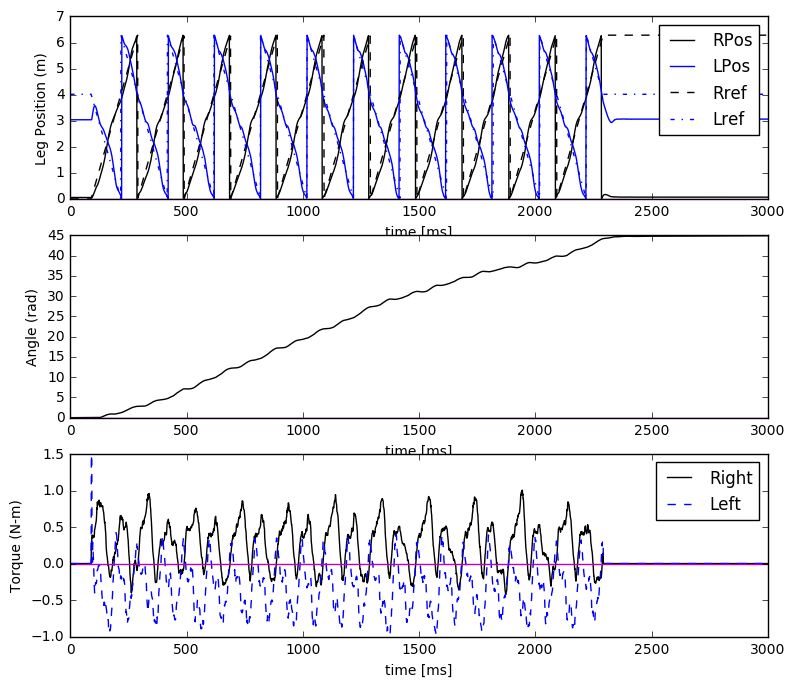


**Figure 7**: Position, angle, and torque for the simulated robot.

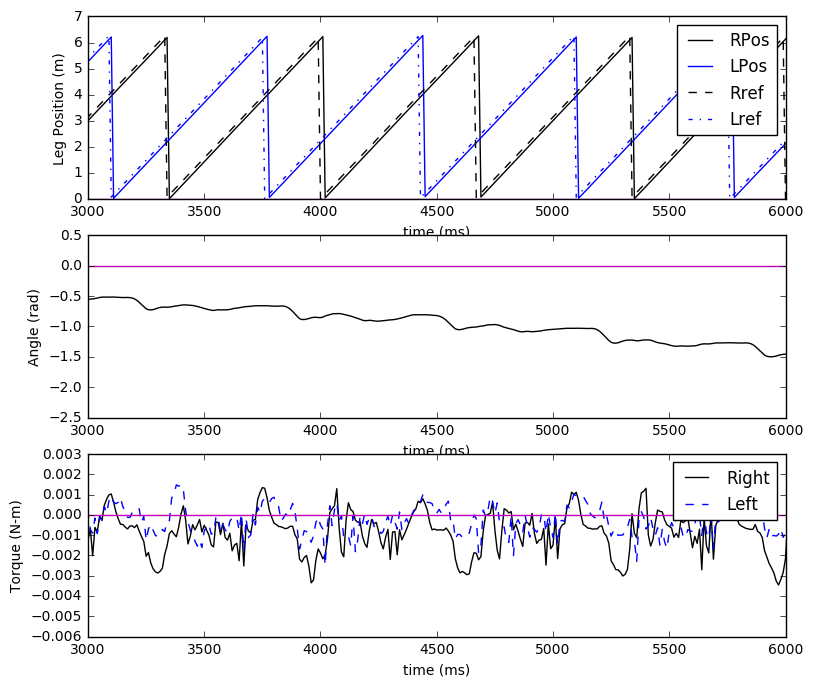


**Figure 8**: Left and right ground reaction forces for the simulated VelociROACH. Forces are relatively symmetric.

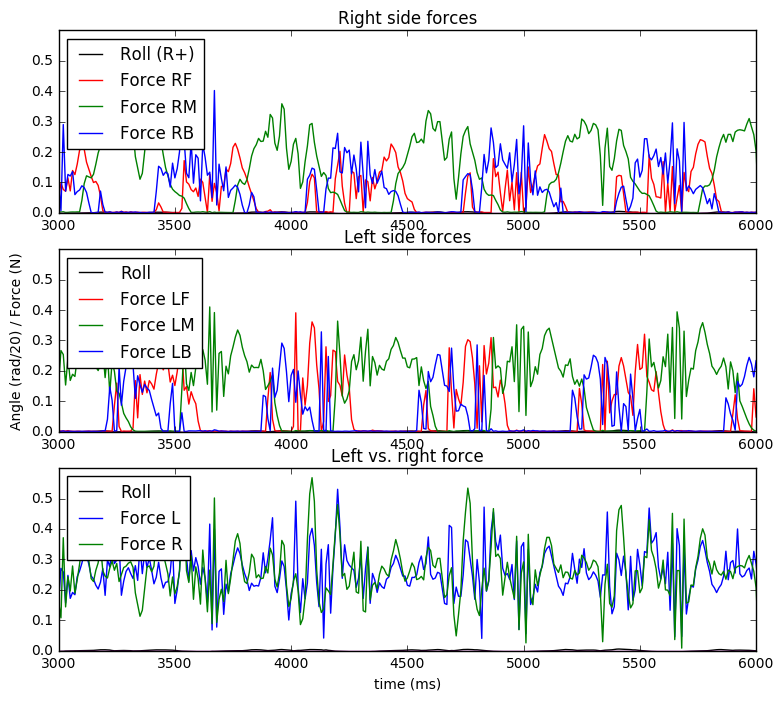
I tested whether the simulation could reproduce roll oscillation modulated turning from Haldane 2014. I produced left-leaning and right-leaning gaits by setting the frequency offset, but the data is not as clean as in the paper. Turning only worked at 1.5 Hz compared to 5 Hz for the original paper.



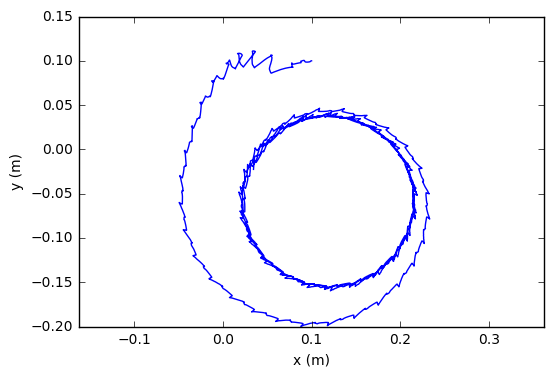
**Figure 9**: Position, angle, and torque telemetry for the real robot at a 130 degree offset, for comparison.



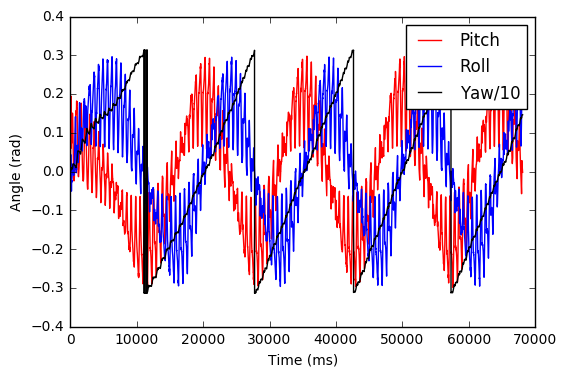
**Figure 10**: Position, angle, and torque for the simulation at an offset of 130 degrees.



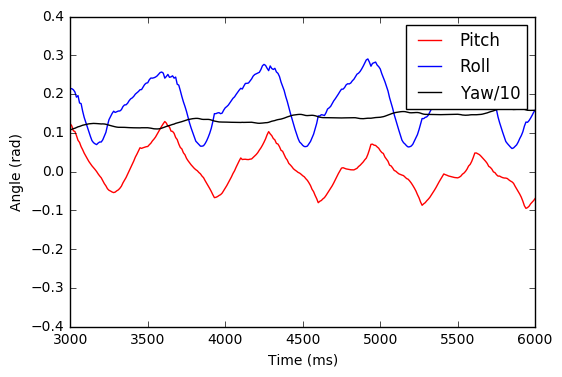
**Figure 11**: Ground reaction forces for the simulation at an offset of 130 degrees.



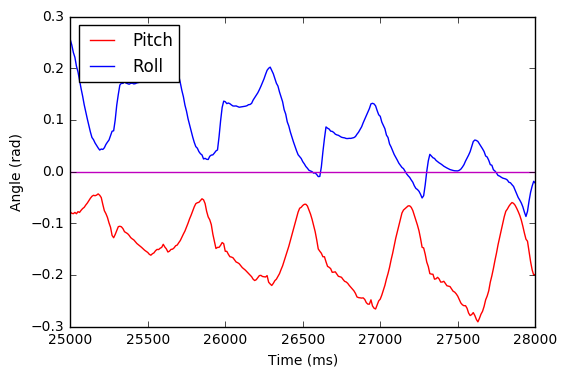
**Figure 12**: Turning behavior of the 10x scaled simulation at 1.5 Hz. Robot starts at the top, and stabilizes into a constant-rate turning gait after approximately one full turn.



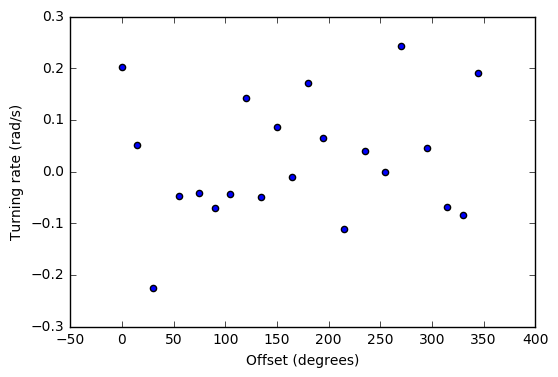
**Figure 13**: Pitch and roll angles for the 10x scaled simulation at 1.5 Hz over four full turns. The gait dynamics are complicated, having low-frequency changes over a full turning cycle (T=15 s), even though yaw rate remains constant over small periods. Pitch and roll does not have zero mean for extended periods of time.



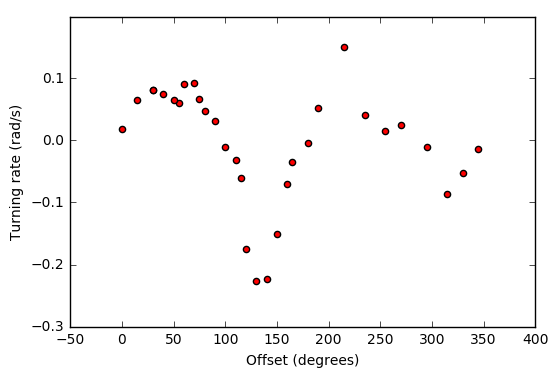
**Figure 14**: Pitch and roll angles for the 10x scaled simulation at 1.5 Hz from 3-6 s. This is a zoomed-in version of Figure 13. It is mostly periodic over the leg cycle (T=0.7 s).



**Figure 15**: Pitch and roll angles for the 10x scaled simulation at 1.5 Hz from 25-28 s. It looks like there is a phase transition.

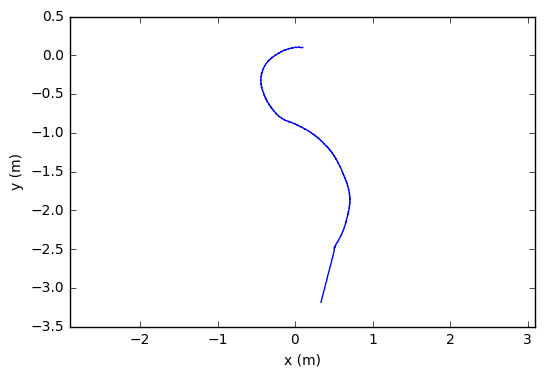


**Figure 16**: Turning rate vs. offset angle for the 10x scaled model at 3 Hz. There is no discernable relationship.



**Figure 17**: Turning rate vs. offset angle for the 10x scaled model at 1.5 Hz. Turning rate has a good response between 130 and 210 degrees.

Finally, I started investigating control methods for the simulated VelociROACH. I commanded the robot to turn first counterclockwise, then clockwise. If the turning rate determines the gait, the robot should follow an S-shaped path. Instead, I found large transient effects when switching between the two phases, suggesting that control depends on the current state of the robot (Figure 17). More research is needed, but these results seem to rule out a naïve control which just adjusts the offset angle using Figure 16. Instead, a controller may need to consider the current roll and pitch of the robot and/or develop more sophisticated angle sequences when transitioning between gaits.



**Figure 18**: Resultant path after switching phase offsets. Robot started at the top.

References

|  |  |
| --- | --- |
| [1] | P. Holmes, R. J. Full and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges," *Siam Review,* vol. 48, no. 2, pp. 207-304, 2006. |
| [2] | D. W. Haldane, K. C. Peterson, F. L. G. Bermudez and R. S. Fearing, "Animal-inspired design and aerodynamic stabilization of a hexapedal millirobot," *International Conference on Robotics and Automation,* pp. 3279-3286, 2013. |
| [3] | R. W. Cottle, "Numerical methods for complementarity problems in engineering and applied science," Berlin, 1977. |
| [4] | W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, Austin, 2010. |