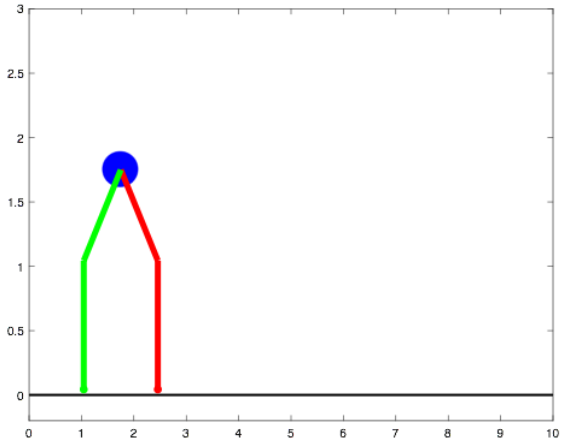
Dynamic Optimization Homework 3 Writeup

Kazu Otani (kotani)

The goal of this assignment was to reimplement Contact Invariant Optimization. I attempted to implement the algorithm on a simplified model of a bipedal walker.

Problem formulation



To make the optimization easier and faster, I made some simplifications to the dynamics model of the bipedal walker. The original Contact Invariant Optimization paper [1] made similar (but less drastic) simplification to their dynamics model. A comparison is shown below.

My simplifications/assumptions	Igor Mordatch's simplifications/assumptions

• 2D
• Point feet
• Any joint angles/torques achievable
• Point mass at torso
  ○ Massless legs
• Quasistatic (ignore velocities)
• Pre-defined potential contact points
	• Parametrize the end-effectors instead of joints
	  ○ 6 DOF forces at contacts
	• Kinematic constraints are enforced as costs
	• Infinitely strong muscles (and no penalty for using them)
	• Point mass at torso
	  ○ Massless limbs
	• Pre-defined potential contact points

In my formulation, I assumed (at the suggestion of Prof. Atkeson) that any joint angles and torques were achievable. This (and the simple environment) allowed me to ignore joint torques and optimize directly over foot forces. This led to further simplification in the optimization (compared to original CIO implementation): I removed the auxiliary contact variables from the state vector, which in turn allowed me to take out the concept of "contact phases" used in the original implementation.

The vector of "states" that I optimized over is the vector of foot forces and positions:

$$[Fy_1, Fz_1, y_1, z_1, Fy_2, Fz_2, y_2, z_2]$$

The rest of the state variables (joint angles, center of mass position) can be calculated from the foot force sequence, given an initial configuration.
The dynamics of the center of mass are simplified so that the foot forces directly affect the center of mass velocity:

$$\dot{y} = \Sigma_{feet} F_y$$
$$\dot{z} = \Sigma_{feet} F_z - mg$$

The desired behavior of the character, which in this case is walking forward in the positive x-direction, was encoded in the cost function. The trajectory was encouraged to look somewhat natural by imposing constraints on the movements.

Costs
The original CIO cost function was:

$$L(\mathbf{s}) = L_{\text{CI}}(\mathbf{s}) + L_{\text{Physics}}(\mathbf{s}) + L_{\text{Task}}(\mathbf{s}) + L_{\text{Hint}}(\mathbf{s})$$

I formulated my cost function to resemble this structure.

- L_{CI} : "Realness" of force
  - realness_factor * footforce * 1/(z_foot+epsilon)
  - The "contact violation vector" in the CIO paper can be simplified to a scalar function of foot height, since the environment only consists of a flat ground at z=0
  - Gradually increase w_{CI}
- $L_{physics}$
  - Feasibility of foot position
    - Make sure foot positions are within reach of the torso
      - norm(com - foot_pos) < leg_length
  - Contacts must be realistic
    - Contact force vectors must point out of the ground: Fz > 0
    - Friction cone: abs(Fy) < Fz*mu_static
  - Feet cannot slide at contacts
- L_{task} : Distance between desired center of mass position and current position

Constraints

- Foot forces must keep robot standing
  - Foot forces keep robot up against gravity: sum(footforces) = [0, m*g]
  - Zero moment about center of mass: sum(r x footforces) = 0
- Lower and upper bounds on the foot forces

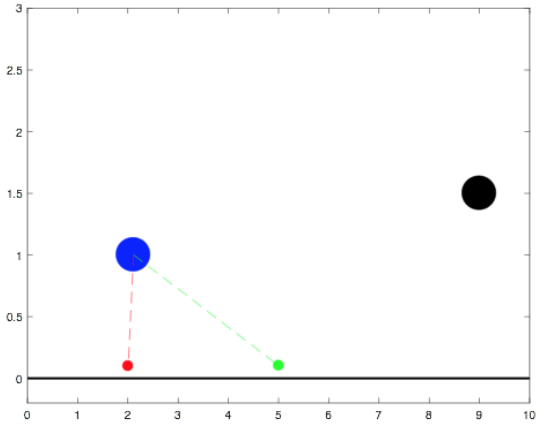
<u>Optimization</u>
I followed a similar optimization procedure as the one outlined in [1]. CIO is analogous to simulated annealing for trajectory optimization. In the beginning, randomness and jumps to suboptimal regions are encouraged for the sake of exploration. Then, physical reality is slowly imposed until the trajectory converges to a feasible and nearly-optimal solution.

The trajectory is optimized in three phases:

1. Only L_{Task} is enabled.
    a. Finds a trajectory that achieves task goals, without any regard for physical feasibility.
2. Enable all four terms, with $L_{physics}$ down-weighted by 0.1.
    a. Starts to consider physical feasibility.
3. Enable all terms.
    a. Finally finds a trajectory that both achieves the task goal and is physically feasible.

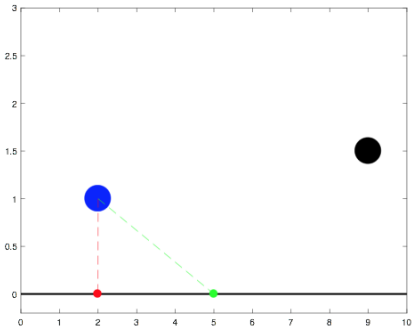
The initialization for Phase 1 is a static initial pose.
The solutions obtained at the end of phases 1 and 2 are perturbed with small zero-mean Gaussian noise (to break any symmetries) and used to initialize the next phase.

Results

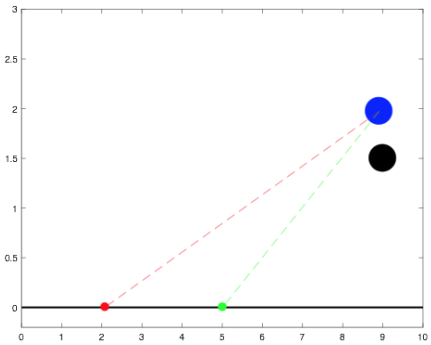


Above is the initial condition from which I started all optimizations. As noted above, the legs are assumed to be able to achieve any joint angles. Because of this, I can represent the legs as "virtual legs", which are valid as long as the feet are within 2*link_length of the center of mass. The desired center of mass position is represented by the black dot in the top right.

First, I tested that the contact-invariant cost was encouraging the optimizer to find physically realistic foot forces and positions. To do this, I set w_{CI} to 1, and all other costs to zero. The expected result was that the optimizer would move to satisfy the constraints of keeping the robot's mass up against gravity, while moving the feet towards the ground to minimize the contact-invariant cost. This worked as expected, as shown below.



Next, I tried imposing L_{task} to push the foot forces to move the center of mass towards the target. As expected, the foot forces moved the center of mass closer to the target in order to minimize the task cost. Note that here, I am not yet imposing $L_{physics}$, which accounts for reachability of the foot position and (soft) constraints on moment around the center of mass.



Last, I imposed $L_{physics}$. Here, things did not work as expected. I found that the interactions between multiple terms in the physics cost did not work well out-of-the-box. Because of the way I calculated the various components of $L_{physics}$, they were often on different orders of magnitude. I had to add arbitrary scaling factors to account for this.

In the end, the best result I was able to get was a trajectory in which the robot slides its feet in a walking motion towards the goal position. Obviously this goes against the whole point of Contact Invariant Optimization (which is supposed to find a sequence of contact sets).

Some possible reasons for my failure are:

- I didn't account for contact set phases (which were used in the original implementation), and instead opted to have the algorithm have a continuous definition of "contact set".
- Because of the way my moment constraint was formulated, it seemed like the optimization often fell into local minima with balanced foot forces and zero moment, and was unable to get out of it.

Due to my incomplete implementation, I am unable to accurately evaluate the performance claims of the original Contact Invariant Optimization paper. However, it seems that there are more parameters to be tuned than the paper initially admits. In addition to hand-designed task costs for specific tasks, Mordatch

also uses L_{hint} in the first 2 phases of optimization, which help the character find feasible movements satisfying criteria such as ZMP. Nevertheless, this is still a significant improvement from previous methods, where contact sets had to be explicitly designated beforehand. In the future (after I get this simple implementation running…), I would like to investigate methods to further simplify CIO or similar algorithms so that it can be run in real-time.

References

[1] Mordatch, Igor, Emanuel Todorov, and Zoran Popović. "Discovery of complex behaviors through contact-invariant optimization." *ACM Transactions on Graphics (TOG)* 31.4 (2012): 43.