ME 193B / 292B: Feedback Control of Legged Robots
Cassie Mini-Project

1 Motivation

In this mini-project, you will be implementing balance control strategies for CASSIE (Figure 1a), a land bird inspired, high-dimensional 3D bipedal robot developed by Agility Robotics. Like walking and running, maintaining balance is an important task for legged robots to perform, for example when manipulating objects or performing operations such as turning valves. It is important that the balance control strategy be robust to external perturbations like large pushes or uncertainties in the robot model (for example, when lifting heavy objects, the true mass of the object may not be known). Over the years, several balance strategies have emerged ranging from static balance control to methods based on the centroidal dynamics of the robot. Section 5 talks about a few of these methods.

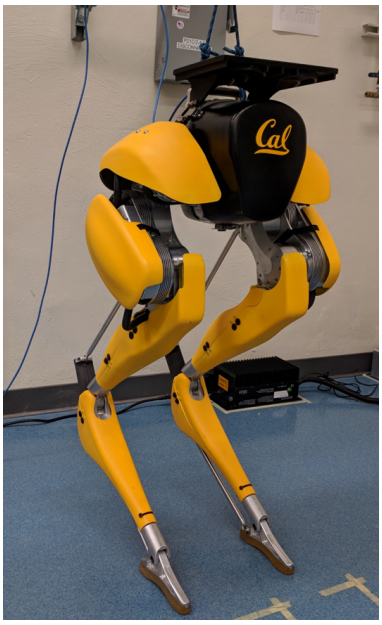
2 Robot Model

CASSIE's floating base model has 20 degrees of freedom, with 7 joints in each leg (Figure 1b). Out of the 7 joints, only 5 are actuated by motors, the remaining 2 joints are connected by rigid springs. In this mini-project, we will ignore the springs for simplicity and model them as fixed joints (or equivalently, making the spring constant infinity). Therefore, there are a total of 5 motors to be controlled per leg corresponding to the joints: (a) hip yaw (rotation about the vertical axis), (b) hip abduction (outwards/inwards motions), (c) hip pitch, (d) knee pitch and (e) toe pitch.

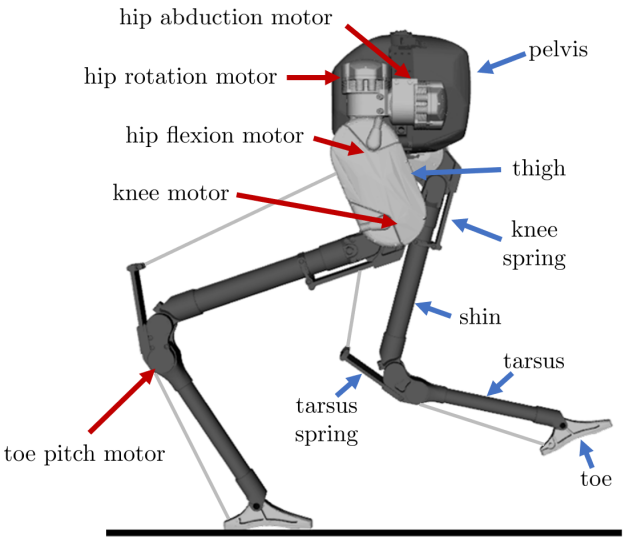
Similar to HW 1 for the three-link robot, CASSIE's dynamical equations can be written in the form

$$D\ddot{q} + C\dot{q} + G = Bu + \Sigma_i^4 J_i^T F_i, \tag{1}$$

where $q \in \mathbb{R}^{20}$ are the joint angles,



(a) CASSIE series robot at UC Berkeley.



(b) Various joints and links of CASSIE.

$$q = \begin{bmatrix} \text{x} \\ \text{y} \\ \text{z} \\ \text{yaw}(\phi) \\ \text{pitch}(\theta) \\ \text{roll}(\psi) \\ \text{left hip abduction} \\ \text{right hip abduction} \\ \text{left hip rotation (yaw)} \\ \text{right hip rotation (yaw)} \\ \text{left hip flexion (pitch)} \\ \text{right hip flexion (pitch)} \\ \text{left knee joint} \\ \text{right knee joint} \\ \text{left knee spring (constrained to 0)} \\ \text{right knee spring (constrained to 0)} \\ \text{left ankle joint (constrained to q12 = deg2rad(13) - q10)} \\ \text{right ankle joint (constrained to q19 = deg2rad(13) - q17)} \\ \text{left toe joint} \\ \text{right toe joint} \end{bmatrix}. \tag{2}$$

Joints indicated in red are acuated by motors with motor torques $u \in \mathbb{R}^{10}$. We will assume that the robot makes 4 point contacts, two at each foot - at the front and back as indicated in Figure 2. The vectors $F_i \in \mathbb{R}^3$ contain the x, y and z contact forces for each of the point contacts and the matrices $J_i \in \mathbb{R}^{20\times 3}$ are the jacobian matrices corresponding to these contact points. The coefficient of friction between the contacting points and the ground is 0.8.

While the D, C and G matrices can be obtained similarly as in HW 1 using Lagrange's equations of motion, in this mini-project, we use the Featherstone package which employs *recursive Newton-Euler* methods

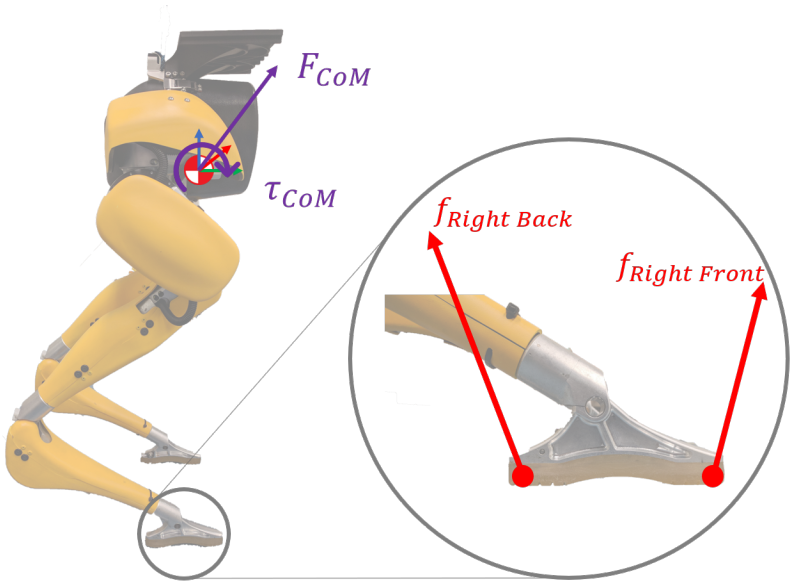


Figure 2: Figure illustrating contact points on the right foot. Contact points are similarly defined for the left foot. Purple arrows indicate the external force applied for the forces defined in `externalForce.m`.

to obtain the dynamical equations. (This is yet another method for obtaining the dynamics of systems and you certainly do not have to dive into the details of this for the purpose of this mini project.)

3 Problem Statement

There are several potential definitions for humanoid robot balance. In this mini-project, we assume a loss of balance when the robot falls (i.e. when the z position of the robot falls below a certain threshold). Your goal is to design a feedback controller for the 10 joint torques u to maintain balance while standing in place. Your controller will be tested with several external perturbations such as pushes in the forwards, sidewards and vertical directions as well as external torques at the pelvis. Your controller should be able to recover from these perturbations and should stabilize the system within 5 seconds. By stability, we mean that the velocity of the joints q must converge to zero. You will be assigned a score based on the following metric:

$$J = \begin{cases} 0 \ \text{ if } z <= 0.1m \text{ or } |P_i| > M_i \\ \frac{\Sigma_i M_i - |P_i|}{\Sigma_i M_i}, \ \text{otherwise} \end{cases} \tag{3}$$

with

$$P = \begin{bmatrix} \text{Translational Speed at the end of 5 seconds, } \|v\|_2 \\ \text{Angular Speed at the end of 5 seconds, } \|\omega\|_2 \\ \text{Settling time of } \|v\|_2 \end{bmatrix}, \tag{4}$$

$$M = \begin{bmatrix} 5m/s \\ 1rad/s \\ 5s \end{bmatrix}, \tag{5}$$

with v and ω being the translational and angular velocities of the pelvis, defined as

$$v = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \quad \omega = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{6}$$

Note that the score does not take into consideration the x and y positions of the robot. This opens up the potential to use push recovery strategies so as to take a step to reject large perturbations.

We will test your controller for 5 different perturbations, i.e., five different external forces will be applied to the robot and the performance of your controller is evaluated. For each perturbation test, you will receive a score based on the metric in (3). Your final score will be a sum of the individual scores you receive.

4 Simulation

4.1 Student functions

We will provide the code for simulation the dynamics. You will need to modify only these two files:

1. `student_setup.m` - You may define any parameters (in the form of a `struct` variable) that your controller would require. *You will also need to enter your student id and your team name* before you make any submissions.

2. `studentController.m` - You will define your controller here, i.e. you will calculate the 10 motor inputs u as a function of the current state and time.

4.2 Other functions

In the `other_functions` subdirectory, there are the following additional functions (you do not have to modify any of these functions, except possibly `ExternalForce.m` for testing your controller with a different perturbation):

1. `getInitialState.m` provides the initial conditions for the simulation. These initial conditions correspond to Cassie standing with its initial center-of-mass located at $\begin{bmatrix} -0.0103 & 0.0000 & 0.8894 \end{bmatrix}^T$ and the pelvis orientation aligned with the world frame. All joint velocities are initialized to 0 rad/s.

2. `ExternalForce.m` contains the external perturbation that is applied to your simulation. Currently a zero perturbation is applied. You can modify this function, if you want, to test the robustness of your controller. The output of this function is a 6×1 wrench (force-torque pair) applied at the torso.

3. `cassie_eom.m` contains the dynamics of Cassie. This function is called by the numerical integration (`ode45` / `ode15s`) to simulate the system.

4. `falldetet.m` contains the event function that detects when Cassie has fallen and stops simulation.

5. `calcScore.m` contains the code to compute the score for your simulation. The scoring is as detailed in the previous section.

4.3 Helper functions

We also provide several helper functions to help you design your controller. You do not have to use these functions unless you want to.

1. `computeComPosVel.m` - Computes the center-of-mass position and velocity.

2. `computeComJacobian.m` - computes the Jacobian of the position of the center-of-mass.

3. `computeFootPositions.m` - computes the positions of the four foot contact points.
4. `computeFootJacobians.m` - computes the Jacobians corresponding to the four contact points.
5. `HandC` - computes the mass matrix D and the sum $C\dot{q} + G - \Sigma_i J_i^T F_i$.

4.4 Libraries

There are a bunch of functions from the featherstone spatial vector toolbox in the `Libraries` folder. You don't have to even look at these functions.

5 Balance Strategies

Below are some potential balance strategies to get you started. They are ordered in roughly increasing order of complexity:

1. *Position control based balancing:* This is a simple strategy based on controlling the joint positions or certain output functions such as the position of the center-of-mass (through a linear feedback controller). A recent paper [1] implements such a strategy on a CASSIE series robot.
2. *Contact-Force based balancing:* Any interaction of the robot with world are thorough contact forces. For balancing and locomotion tasks, these contacts are primarily at the feet. Regulating the contact forces at the feet directly influences the center of mass positions and velocities as well as the global orientation of the robot, which is the key idea behind these methods. See [4, 2] for more details.
3. *Momentum-based balancing:* The main assumption in [4] is that the robot is in static equilibrium at all times. While this assumption is valid for slow and quasi-static motions, it breaks when there are large accelerations/forces acting on the system. The quantities that are directly regulated by the contact forces and torques are the rate of change of linear and angular momentum about the center-of-mass. Methods that regulate the center-of-mass linear and angular momentum are able maintain balance even on non-stationary ground. The momentum-based strategy is similar to how a human would balance - one would swing their arms or move their torso when subjected to a large push, which effectively changes their momentum about the center-of-mass. [3] is an excellent reference for such methods.
4. *Push recovery and stepping based balancing:* When it is no longer possible to keep the center-of mass of the robot within the support polygon (a key requirement for standing in place), it is important for the robot to take a step (or possibly multiple steps). This greatly improves the robot's capability to withstand large unknown forces. See [5] for more details.

6 Next Steps

Download the `cassie_simulator.zip` file from bCourses. To run a simulation, execute the script `run_cassie.m` file in the `cassie` folder. This should simulate the system for 5 seconds, or until the robot has fallen down.

7 Submission Procedure

- Create a team (instructions in bcourses / Piazza).
- When you are happy with your controller performance, execute `pack_o_matic`. This will pack up your controller for testing with 5 unknown perturbations on the leaderboard server. This function needs an active connection to the internet to upload the files. This function could take a while to run. If everything goes fine, you will see an appropriate message in Matlab and the leaderboard updated to

reflect your new score for your submission. Just making a submission with the given code will get you a score of around 80/500. You can make any number of submissions. Only your top score of all your submissions will be used. Leaderboard is displayed live at this link provided on bcourses.

- In order to keep things fair, please do not try to hack into the leaderboard server, or try to phish out the 5 unknown perturbations that your controller will be tested for. If we find out that one of the teams is employing such a strategy, their submissions will be nullified and a score of zero will be assigned.
- Your score on this project is whatever you get on the leaderboard. Each team member can make their own submission if they want to. However, please use the same team name so that it is easier to see the rankings of teams. The leaderboard could display multiple scores for the same team (if multiple team members of the same team are submitting solutions), but only the top score will be used towards the grade of the entire team.

References

[1] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," *arXiv preprint arXiv:1809.07279*, 2018.

[2] B. Henze, M. A. Roa, and C. Ott, "Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1522–1543, 2016.

[3] S.-H. Lee and A. Goswami, "Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on.* IEEE, 2010, pp. 3157–3162.

[4] C. Ott, M. A. Roa, and G. Hirzinger, "Posture and balance control for biped robots based on contact force optimization," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on.* IEEE, 2011, pp. 26–33.

[5] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on.* IEEE, 2006, pp. 200–207.