Topics Covered:

- Modeling Bipedal Robots
- Hybrid Systems
- Trajectory Generation

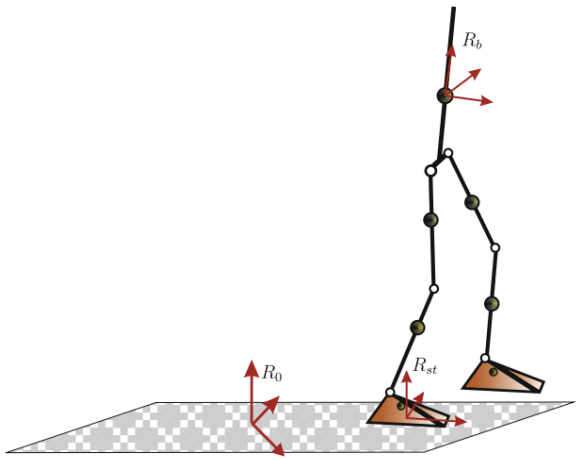
Additional Reading:

- Feedback Control of Dynamic Bipedal Robot Locomotion
- Models, feedback control, and open problems of 3D bipedal robotic walking
- Dynamic Walking: Toward Agile and Efficient Bipedal Robots

Note: This information will *not* be on the final exam.

Dynamic models for bipedal robots

Most bipedal systems are modeled as the following (image taken from Models, feedback control, and open problems of 3D bipedal robotic walking):



with R_0 being a fixed inertial frame, R_{st} being a frame attached to the stance foot, and R_b a frame attached to the torso. There are two general methods for describing the states of this system: a *pinned* model, and an *unpinned* model.

The *pinned* model assumes that the stance foot is "pinned" to the ground (i.e., the ground contact is assumed to be satisfied). As long as this condition is held, then the full system states can be

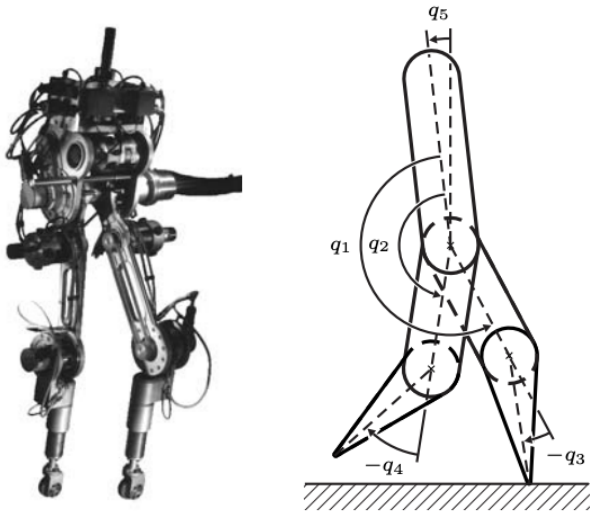
described by the joint angles and the position/orientation of the stance foot.

The *unpinned* model instead does not place any assumptions on the stance foot, but instead augments the system state to include information about the "floating-base frame". This leaves us with the augmented set of coordinates:

$$q = (p_b^\top, \phi_b^\top, \theta^\top)^\top \in \mathcal{Q} = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^m$$

where $p_b \in \mathbb{R}^3$ is the Cartesian position of frame R_b and orientation $\phi_b \in SO(3)$ of frame R_b with respect to the fixed frame R_0. Our joint angles (joint-space coordinates) are denoted as usual as $\theta \in \mathbb{R}^m$.

Throughout lecture, we will be focusing on a specific robot model called RABBIT:



In this case, we have four joint angles:

$$\begin{aligned} \theta &= (\theta_{sh}, \theta_{sk}, \theta_{nsh}, \theta_{nsk})^\top \\ &= (q_1, q_3, q_2, q_4)^\top \qquad \text{(as shown in the figure above)} \end{aligned}$$

with the subscripts denoting stance hip, stance knee, nonstance hip, and nonstance knee. In the planar case, the floating-base frame is represented as the x and y position of the torso, and the orientation of the torso (denoted in the diagram as q_5).

Continuous-Time Equations of Motion

As discussed in the last lecture, the continuous-time dynamics of the system can be modeled using the standard robot equations of motion (obtained using the Lagrangian mechanics):

$$\tau = M(q)\ddot{q} + H(q, \dot{q})$$

Except now, since all of our coordinates aren't actuated, we will introduce an *actuation matrix* B to only assign torque to the actuated coordinates:

$$Bu = M(q) + H(q, \dot{q})$$

In the scenario where the system is fully-actuated, B is simply the identity matrix. If instead we used the pinned model for RABBIT, we would have: $q = (q_1, q_2, q_3, q_4, q_5)^\top$, with:

$$B = \begin{bmatrix} I_{4\times 4} \\ 0_{1\times 4} \end{bmatrix}$$

Next, we will use something called D'Alembert's principle to model ground contact forces. Specifically, we will augment our system with:

$$M(q)\ddot{q} + H(q, \dot{q}) = B(q)u + \delta F_{ext}$$
$$M(q)\ddot{q} + H(q, \dot{q}) = B(q)u + J_{st}(q)^\top F_{st}$$

Here, δF_{ext} are any external forces acting on the robot. The contact force with the ground can then be explicitly modeled using the Jacobian:

$$\begin{bmatrix} v_{st} \\ \omega_{st} \end{bmatrix} = J_{st}(q)\dot{q} \in \mathbb{R}^{N_c \times 1}$$

Here, N_c is the number of contact constraints. For a planar foot, we have $N_c = 3$ (two forces and one torque). For a planar point-foot we have $N_c = 2$ (two forces). For a full 3D planar foot, we have $N_c = 6$ (three forces and three torques). Lastly, F_{st} represents the wrench applied at the stance foot.

When considering the unpinned model, we can enforce the constraint that the stance foot is stationary by enforcing the kinematic constraint:

$$\eta_{st}(q) = \begin{bmatrix} p_{st} \\ \phi_{st} \end{bmatrix} = \text{constant}$$

We typically call this constraint the *holonomic constraint*. When this constraint is enforced (the position and orientation of the stance foot is constant), their velocity and acceleration should also be zero, which can be enforced by the following:

$$J_{st}(q)\dot{q} = 0$$
$$J_{st}(q)\ddot{q} + \dot{J}_{st}(q)\dot{q} = 0_{N_c \times 1}$$

Friction Cone Constraints

There is a known relationship between the allowable forces on the stance foot and whether or not the holonomic constraints will be satisfied. Thus, we can impose further constraints on the forces applied to the stance foot. Specifically, we can enforce the *friction cone constraints*:

$$\sqrt{(F_{st}^{fx})^2 + (F_{st}^{fy})^2} \leq \mu F_{st}^{fz}$$

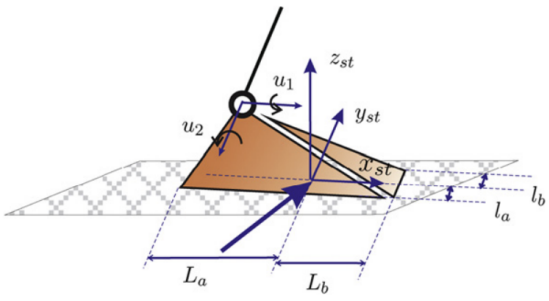
with μ being the coefficient of friction. This friction cone can be relaxed to a friction pyramid to allow for linear constraints:

$$|F_{st}^{fx}| \leq \frac{\mu}{\sqrt{2}} F_{st}^{fz}$$
$$|F_{st}^{fy}| \leq \frac{\mu}{\sqrt{2}} F_{st}^{fz}$$

Lastly, even if the foot isn't slipping, it could still tip. So to avoid tipping, we can enforce the *zero moment point* constraint that forces the COP to lie within the support polygon of the foot:

$$-l_b F_{st}^{fz} \leq F_{st}^{mx} \leq l_a F_{st}^{fz}$$
$$-L_a F_{st}^{fz} \leq F_{st}^{my} \leq L_b F_{st}^{fz}$$

assuming the following foot geometry:



Discrete-Time Equations of Motion

We model the instantaneous change in the system velocity at impact the *impact model*. While there are various ways to model this interaction, one way is to assume conservation of momentum as proposed by Hurmuzlu and Marghitu

$$M(q)(\dot{q}^+ - \dot{q}^-) = J_{st}(q)^\top F_{imp}$$

with F_{imp} being the integral of the impulsive contact wrench over the impact duration (typically assumed to be very small), $\dot{q}^-$ being the velocity just before impact, and $\dot{q}^+$ being the velocity just after impact. This equation, combined with a kinematic constraint of the post-impact state:

$$J(q)\dot{q}^+ = 0$$

gives us our impact model:

$$\begin{bmatrix} M(q) & J_{st}(q)^\top \\ J_{st}(q) & 0 \end{bmatrix} \begin{bmatrix} \dot{q}^+ \\ F_{imp} \end{bmatrix} = \begin{bmatrix} M(q)\dot{q}^- \\ 0 \end{bmatrix}$$

Block matrix inversion yields the direct mapping:

$$\dot{q}^+ = \underbrace{(I - M^{-1}J_{st}^\top (J_{st}M^{-1}J_{st}^\top)^{-1}J_{st}M^{-1})}_{\Delta(q)}\dot{q}^- \qquad \text{(dropped input } (q) \text{ for notation)}$$

$$\dot{q}^+ = \Delta(q)\dot{q}^-$$

Note here that q is assumed to be the same pre and post-impact, but we could assign a relabeling matrix such that we have $q^+ = Rq^-$. This would change our impact map to be:

$$\begin{bmatrix} q^+ \\ \dot{q}^+ \end{bmatrix} = \begin{bmatrix} Rq^- \\ R\Delta(q^-)\dot{q}^- \end{bmatrix}$$

Hybrid Systems

We can represent the combination of continuous-time dynamics and discrete-time dynamics as a *hybrid system*. For notation, we will combine q and $\dot{q}$ into a single state $x = (q^\top, \dot{q}^\top)^\top$. We can then represent our continuous-time dynamics as:

$$\dot{x} = f(x) + g(x)u$$

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ -M(q)^{-1}H(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} 0 \\ M(q)^{-1}B \end{bmatrix} u$$

Each node of the hybrid system contains continuous-time dynamics with a pre-defined set of ground contact constraints. The transition between a node and the next is then governed by the discrete-time dynamics of the impact model. We can enforce when to trigger the discrete transition as a switching (or impact) condition $\phi(x)$ When this condition is equal to zero, the system will transition to the next node. Using this condition, we can represet the set of coordinates that are satisfied when the condition is true as the sets belonging to a *switching surface*:

$$\mathcal{S} = \{x \in \mathcal{D} \mid \phi(x) = 0,\ \dot{\phi} < 0\}$$

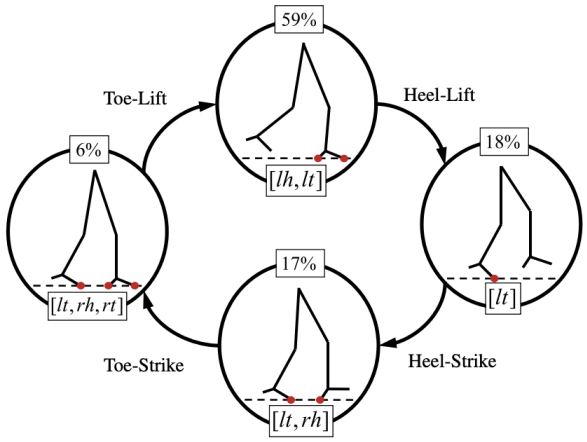
Typically for walking, the switching condition is selected to be the height of the non-stance foot: $\phi(x) = p^z_{nsf}(x)$. This transforms our switching surface to be:

$$\mathcal{S} = \{x \in \mathcal{D} \mid p^z_{nsf}(x) = 0,\ \dot{p}^z_{nsf} < 0\}$$

Finally, we can represent our hybrid system as:

$$\mathcal{HC} = \begin{cases} \dot{x} = f(x) + g(x)u & x \notin \mathcal{S} \\ x^+ = \Delta(x^-) & x^- \in \mathcal{S} \end{cases}$$

More complex hybrid systems can be constructed in situations where there are multiple contact domains. For example, full "foot-rolling" walking can be captured by the following graph:



Trajectory Generation

The goal of trajectory generation for bipedal robots is to synthesize a set of polynomials for the controllable states of our system such that the full-order model remains periodically stable. To do this, we pick a set of *outputs* (also called *virtual constraints*) y that we want to construct. While these outputs can be any function of the state, they are commonly chosen to be the joint angles/velocities. For RABBIT, we will choose the desired outputs to be:

$$y_d(q) = \begin{bmatrix} q_1^d \\ q_2^d \\ q_3^d \\ q_4^d \end{bmatrix}$$

Our goal for control is then drive the outputs to zero, which is equivalent to driving the actual outputs to the desired outputs:

$$y = y_d - y_a$$

Ensuring that this output is zero through control is why we often call it a *virtual constraint*.

Once equipped with our hybrid system model and our choice of virtual constraints, we can construct a trajectory generation optimization problem to solve for the polynomials of the desired outputs such that the following inequality and equality constraints are held:

Inequality Constraints

- The phasing variable ϕ is strictly increasing, $\dot{\phi} > 0$ along the solution of each domain
- the solution respects the domain of admissibility (joint limits)
- positive vertical reaction force on the stance foot (no take off constraint)

- friction constraints
- bounds on allowed actuator torques
- the swing foot is positioned above the ground (unless a double-support phase)

Equality Constraints

- conditions at the domain transitions impose periodicity
- desired walking speed
- other desired walking characteristics (step length, step duration, step height, step width)

Cost function

Typically, the cost function for this optimization problem is taken to be the norm of the control input. However, one of the most well-behaved cost functions is the *cost of transport* which also accounts for energy relative to the associated step length:

$$J = \frac{1}{SL}\int_0^T \|u(t)\|_2^2 dt$$

Optimization Problem Formulation

Mathematically, we can write down this optimization problem as the following:

$$\{\alpha^*, X^*\} = \underset{\alpha, X}{\text{argmin}}\ J(X)$$

subject to:

$$\begin{aligned}
&\dot{x} = f(x) + g(x)u^*(x) && \text{(Satisfies Closed-Loop Dynamics)}\\
&\Delta(y(x^-)) = y(x^+) && \text{(Periodic Condition)}\\
&X_{\text{min}} \preceq X \preceq X_{\text{max}} && \text{(Decision Variables)}\\
&c_{\text{min}} \leq c(X) \leq c_{\text{max}} && \text{(Physical Constraints)}\\
&a_{\text{min}} \leq a(X) \leq a_{\text{max}} && \text{(Feature Constraints)}
\end{aligned}$$

where α^* is our collection of polynomial coefficients for the desired outputs, and $X = (x_0, \ldots, x_N, T)$ is the collection of all decision variables with x_i being the state at the i^th discretization of time and T being the total duration of the trajectory.

Methods for Trajectory Optimization

Since this optimization problem is nonlinear, it can be very challenging to solve. For an interesting read about a few appraoches to solving these optimization problems, check out this blog post here.

For our appraoch to trajectory optimization, a past graduate student project was the development of the FROST toolbox which constructs these hybrid system trajectory optimization problems in MATLAB using IPOPT: Frost Website. I've also tried to document an example gait generation setup for the RABBIT model (plus a version that has flat feet with actuated ankles) in the following repository: rabbit_opt_example.