ME 6402 – Lecture 26

HYBRID SYSTEMS AND ZERO DYNAMICS

April 15 2025

Overview:

- Model walking as a hybrid system
- Introduce hybrid zero dynamics

Additional Reading:

- J.W Grizzle, C. Chevallereau, R.W. Sinnet, A.D. Ames. "Models, feedback control, and open problems of 3D bipedal robotic walking." Automatica. 2014.

Hybrid Systems

Hybrid systems combine continuous and discrete dynamics, making it a particularly useful tool for systems with discrete impact events such as bipedal robots.

Definition: *Hybrid System.* A simple or single-domain hybrid system is a tuple:

$$\mathcal{H} = (\mathcal{D}, S, \Delta, f),$$

where

- $\mathcal{D}$ is the domain with $\mathcal{D} \subseteq \mathbb{R}^n$ a connected subset of the state space $\mathbb{R}^n$.
- $S \subset \mathcal{D}$ is a proper subset of the domain, called the guard or switching surface.
- $\Delta : S \to \mathcal{D}$ is a smooth map called the reset map often representing the discrete impact dynamics.
- f is the vector field on $\mathcal{D}$ representing the continuous dynamics.

For a system with impulsive discrete dynamics, the hybrid system can be written as:

$$\mathcal{H} = \begin{cases} \dot{x} = f(x) & \text{if } x \in \mathcal{D} \setminus S \\ x^+ = \Delta(x) & \text{if } x \in S \end{cases}$$

where x^- is the *pre-impact* state and x^+ is the *post-impact* state.

Definition: *Hybrid Control System.* A hybrid control system is a tuple:

$$\mathcal{HC} = (\mathcal{D}, U, S, \Delta, f, g)$$

where the additions are:

- $U \subseteq \mathbb{R}^m$ is the set of admissible controls
- (f, g) is a control system on $\mathcal{D}$

Again, a system with impulsive effects can be written as:

$$\mathcal{H} = \begin{cases} \dot{x} = f(x) + g(x)u & \text{if } x \in \mathcal{D} \setminus S \\ x^+ = \Delta(x) & \text{if } x \in S \end{cases}$$

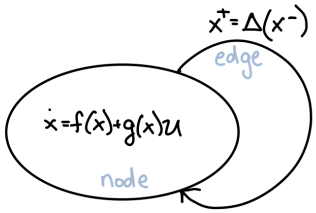
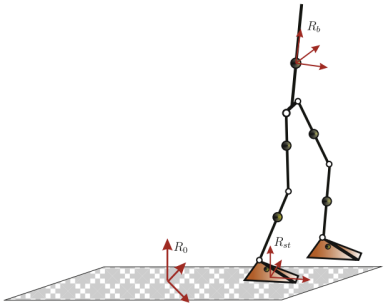


Figure 1: A simple hybrid control system with one domain and one switching surface.

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 $\theta \in \mathbb{R}^m$.

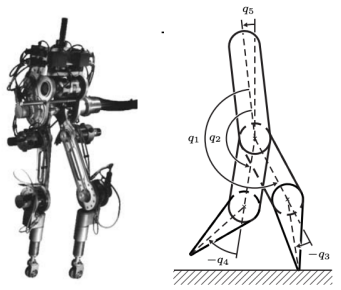


Figure 2: The RABBIT robot (left) and its schematic (right).

A planar robot that is often used in the literature is the RABBIT robot:

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.

For simplicity, we will assume a pinned model, in which the full system state can be represented through the addition of q_5 which will represent the angle of the torso with respect to the vertical axis.

Continuous-Time Equations of Motion

In the case of a walking robot, 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:

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

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 J_{st} being the Jacobian of the stance foot with respect to the center of mass, 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:

$$\dot{q}^+ = \underbrace{(I - M^{-1}J_{st}^\top(J_{st}M^{-1}J_{st}^\top)^{-1}J_{st}M^{-1})}_{\Delta(q)}\dot{q}^-$$

(dropped input (q) 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:

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

Return to Hybrid Systems

In summary, 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*:

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

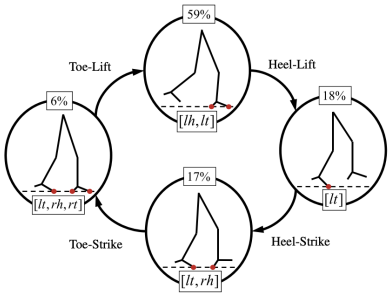
Typically for walking, the switching condition is selected to be the height of the non-stance foot: $\phi(x) = p^z_{nsf}(x)$. This transforms our switching surface to be:

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

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:



Periodic Orbits

Suppose that $\varphi(t, x_0)$ is a periodic solution with initial condition $x_0 \in \mathcal{D}$ and a period $T > 0$. A periodic orbit is a solution of the form:

$$\mathcal{O} = \{\varphi(t, x_0) \mid t \in [0, T]\}$$

Definition: *Periodic.* We say that φ is periodic with period T if $\varphi(T, \Delta(x^*)) = x^*$.

Definition: *Hybrid Periodic Orbit.* A set $\mathcal{O}$ is a hybrid periodic orbit with fixed point x^* if $\mathcal{O} = \{\varphi(t, x^*) \mid 0 \leq t \leq T\}$ for a periodic solution φ.

Associated with a periodic orbit is a Poincaré map which defines a discrete time dynamical system on a subspace of the domain of

the dynamical system. This system determines the stability of the periodic orbit. Taking S to be the Poincaré section, we can obtain the Poincaré map $P : S \to S$ which is a partial function:

$$P(x) = \varphi(T_I(x), \Delta(x)),$$

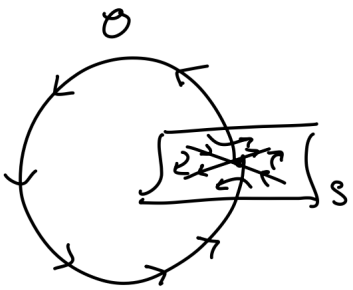
where $T_I : S \to \mathbb{R}_{\geq 0}$ is the time-to-impact function:

$$T_I(x) = \min\{t \geq 0 \mid \varphi(t, \Delta(x)) \in S\}, \quad x \in S$$

The stability of the Poincaré map determines the stability of the periodic orbit $\mathcal{O}$.

Theorem: Periodic Stability. *Given a hybrid system $\mathcal{H}$ with a periodic orbit $\mathcal{O}$, the associated Poincaré map P is (locally) exponentially stable at the fixed point x^* as a discrete time system $x_{k+1} = P(x_k)$ if and only if the periodic orbit $\mathcal{O}$ is (locally) exponentially stable.*

A graphical illustration of the Poincaré map and stability analysis via projection onto a Poincaré section is shown below:



Note: It is often not possible to analytically compute the Poincaré map, so instead it is numerically approximated via its Jacobian. If the eigenvalues of the Jacobian have magnitude less than 1, the stability of the periodic orbit $\mathcal{O}$ has been numerically verified.

Feedback Control for Biped Robots

Stable locomotion can be achieved by constructing a feedback controller that renders the periodic orbit $\mathcal{O}$ locally exponentially stable. In the case of an underactuated robot (such as RABBIT), we can use a *hybrid zero dynamics* approach to construct a feedback controller that stabilizes the periodic orbit even when the system has zero dynamics.

The first step, is to select our choice of outputs:

$$y(q) := y_a(q) - y_d(\tau(q), \alpha) \in \mathbb{R}^m, \quad \tau(q) = \frac{\theta(q) - \theta^+}{\theta^- - \theta^+}$$

where $y_d(\tau, \alpha)$ is the desired output trajectory parameterized by some set of coefficients α, and a *phasing variable* $\theta : \mathcal{D} \to [0, 1]$ with θ^+ being its value post-impact and θ^- being its value pre-impact.

By modeling our system using the Euler-Lagrange Equations of Motion, and by selecting an output that is only a function of q, we know that the outputs will be relative degree. In other words, we know that we can differentiate the output twice to obtain:

$$\ddot{y}(x) = L_f^2 y(x) + L_g L_f y(x) u$$

Therefore, selecting the feedback linearizing control law

$$u(x) = (L_g L_f y(x))^{-1} \left(-L_f^2 y(x) + v \right)$$

yields the linear system

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} v$$

If we select the auxiliary control law

$$v = \frac{1}{\varepsilon^2} K_P y - \frac{1}{\varepsilon} K_D \dot{y}$$

then our closed loop system becomes the linear system:

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \frac{1}{\varepsilon} \begin{bmatrix} 0 & \varepsilon I \\ -\frac{1}{\varepsilon} K_P - K_D & \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

with K_P and K_D selected so that the system is stable for all $0 < \varepsilon < 1$. The choice of ε forces the system to converge at a user-specified rate. Specifically, the feedback controller will drive y and $\dot{y}$ to zero exponentially fast at a rate of $1/\varepsilon$.

Zero Dynamics

As the feedback control law is driving $y \to 0$ and $\dot{y} \to 0$, it is driving the continuous dynamics to the zero dynamics surface:

$$\mathcal{Z}_\alpha := \{ x \in \mathcal{D} \mid y(x) = 0, \dot{y}(x) = 0 \}$$

For more information on zero dynamics applied to bipedal robots, I highly recommend "Hybrid Zero Dynamics of Planar Biped Walkers" by E. Westervelt et al. If you'd like even more details, Jessy Grizzle has written an entire textbook around this concept titled "Feedback Control of Dynamic Bipedal Robot Locomotion".

The main benefit of projecting our system onto the Zero Dynamics Surface is that it reduces the dimensionality of our system and allows us to analyze the periodic stability of only the zero dynamics. This restricted hybrid system is written as:

$$\mathcal{HZ}_\alpha := \begin{cases} \dot{z} = w(0,z) & \text{if } z \in \mathcal{Z}_\alpha \setminus S \bigcap \mathcal{Z}_\alpha \\ z^+ = \Delta(z^-) & \text{if } z^- \in S \bigcap \mathcal{Z}_\alpha \end{cases}$$

Then, our theorem from before can be restated as:

$$\mathcal{O}_{Z_\alpha} \text{ is exponentially stable} \quad \Longrightarrow \quad \mathcal{O} \text{ is exponentially stable}$$

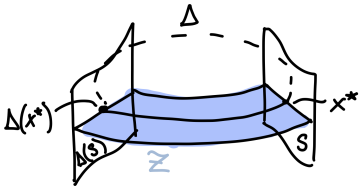


Figure 3: Illustration of hybrid zero dynamics

APPENDIX: 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:

$$\dot{x} = f(x) + g(x)u^*(x) \qquad \text{(Satisfies Closed-Loop Dynamics)}$$

$$\Delta(y(x^-)) = y(x^+) \qquad \text{(Periodic Condition)}$$

$$X_{\text{min}} \preceq X \preceq X_{\text{max}} \qquad \text{(Decision Variables)}$$

$$c_{\text{min}} \leq c(X) \leq c_{\text{max}} \qquad \text{(Physical Constraints)}$$

$$a_{\text{min}} \leq a(X) \leq a_{\text{max}} \qquad \text{(Feature Constraints)}$$

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.