Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems

Russ Tedrake and Scott Kuindersma
Robot Locomotion Group
MIT CSAIL
All of this is done using MATLAB...

and we’re giving away the code.
What is Drake?

- Library for simulating, analyzing, and designing controllers for nonlinear dynamical systems
- Class hierarchy built on top of Simulink
- Exploits mathematical structure of nonlinear systems for design and analysis
- Used for research and teaching at MIT
You could use Drake for...

- Analyze the stability of your systems
- Design nonlinear feedback controllers for complicated (nonlinear, underactuated) dynamical systems
- Perform fast IK, trajectory and feedback-motion planning
- Compute invariant "funnels" along trajectories (derived from your own motion planning software) for robust motion planning
- Explore your own algorithms using Drake's implementations of model robot systems
Why MATLAB?

• Rapid prototyping
• Easy interface to solvers
  • SNOPT, Gurobi, Mosek, SeDuMi, CPLEX, CVX(GEN),…
• Access to Simulink ODE solvers
• Easy visualization, GUIs, I/O, debugging
• Interaction with Java and C++ (via mex)
Drake Features

• Many algorithms available for planning, verification, control, state estimation

• Support for URDFs

• Integration with Bullet for simulation and collision detection

• OpenGL visualizer

• Lots of examples: Atlas, PR2, fixed-wing aircraft, …
Structured Nonlinearity

- Simulink is a powerful tool combining and simulating dynamical systems
- Drake adds class structure for exploiting mathematical structure of nonlinear systems
- Important for modern nonlinear control synthesis
Polynomial Dynamical Systems

- State space description:
  \[
  \dot{x} = f(x, u, t) \\
  y = g(x, u, t)
  \]
  where \( f \) and \( g \) are polynomial

- Verifying stability with Lyapunov functions can be done via convex optimization

- Region of attraction analysis via bilinear optimization
Algebraic Rigid Body Dynamics

- Rigid body equations of motion can often be expressed as polynomials, e.g.,

\[ ml^2 \ddot{\theta} + b \dot{\theta} + mgl \sin \theta = \tau \]

- Change coordinates and add a constraint:

\[
\sin \theta \rightarrow s, \quad \cos \theta \rightarrow c \quad s^2 + c^2 = 1
\]
Quadrotor ROA

Verification with Inertia Variation of 10 Percent

Nominal Verified Basin
Uncertain Verified Basin
Quadrotor Example

classdef PlanarQuadPlant < SecondOrderSystem
  properties % based on (Bouadi, Bouchoucha, Tadjine 2007)
    L = 0.25; % length of rotor arm
    m = 0.486; % mass of quadrotor
    I = 0.00383; % moment of inertia
    g = 9.81; % gravity
  end

  methods
    function obj = PlanarQuadPlant()
      obj = obj@SecondOrderSystem(3,2,true);
      obj = obj.setOutputFrame(obj.getStateFrame); % allow full-state feedback
    end

    function qdd = sodynamics(obj,t,q,qd,u)
      % Implement the second-order dynamics
      qdd = [ -sin(q(3))/obj.m*(u(1)+u(2));
             -obj.g + cos(q(3))/obj.m*(u(1)+u(2));
             obj.L/obj.I*(-u(1)+u(2))];
    end

    function [c,V] = hoverLQR(obj)
      x0 = Point(obj.getStateFrame,zeros(6,1));
      u0 = Point(obj.getInputFrame,obj.m*obj.g/2 * [1;1]);
      Q = diag([10 10 10 1 1 (obj.L/2/pi)]); % Q = diag([10*ones(1,3) ones(1,3)]);
      R = [0.1 0.05; 0.05 0.1]; % R = diag([0.1 0.1]);
      [c,V0] = tilqr(obj,x0,u0,Q,R);
      sys = feedback(obj,c);
      pp = sys.taylorApprox(0,x0,[],3); % make polynomial approximation
      options=struct();
      options.degLl=2;
      V=regionOfAttraction(pp,V0,options);
    end
end
Simple Walking Models
Atlas Planning & Control

- Whole-body collision-free motion planning
- Footstep planning
- Dynamic whole-body control for locomotion and manipulation
- LCM interface to other software components
Motion Planning

- Nonlinear optimization (via SNOPT)
- Large library of constraint classes for specifying problems
- Integration with Bullet for collision detection
- Fast: whole-body planning at interactive rates
Walking Control

- Quadratic programming-based controller
- Significant speed requirement: exploits mex
- Custom efficient QP solver in Drake
- Exploits fast computation of dynamics, kinematics, and contact constraints (and derivatives)
Atlas balancing video (demonstrates controller and motion planner)
Online Education

• Underactuated Robotics course at MIT
• Upcoming edX course in Fall 2014
Where to get it

- [http://drake.mit.edu](http://drake.mit.edu)
- Support for Linux, Mac, and Windows
- Active and growing user community