

CIS 540 – Fall 2009: Homework 4, Due November 23.

For this homework, if you wish, you can work in groups of 2.

Problem 1 (Inverted Pendulum)

Consider a simple pendulum in Fig. 1. The pendulum is a rod with a rotational joint at one end and a mass at the other end. For simplicity, we assume that friction and the weight of the rod are negligible. A motor is placed at the pivot to provide an external torque to control the pendulum.

From Newton's law for rotating objects, the dynamics of the pendulum system is described by the following nonlinear differential equation:

$$-ml^2\ddot{\varphi}(t) + mgl \sin \varphi(t) = u(t) \quad (1)$$

where m is the weight of the mass, g the gravitational acceleration, l the length of the rod, φ the angle of the pendulum measured clockwise from the upward vertical, and u the external torque in counterclockwise direction. The external torque u will be used to control the pendulum.

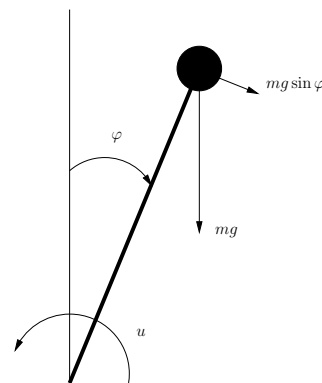


Figure 1: Inverted Pendulum

- Define the state of the system and derive its state-space model. The model should be in the form $\dot{x}(t) = f(x(t), u(t))$.
- An equilibrium of this system is a point x^* such that $f(x^*, 0) = 0$, i.e. the system will remain at state x^* given that the control is 0. Find all the equilibria of the system and point out the corresponding positions (φ^*) of the pendulum. For each equilibrium, if there is a small deviation from it (i.e. $\varphi = \varphi^* + \epsilon$ for a small ϵ) while the control is still 0, will the pendulum swing back to this equilibrium by itself?
- We would like to control the pendulum, via the motor at the pivot, so that it swings to and stays at the upright position ($\varphi = 0$). A classical approach to designing controllers for nonlinear systems is to linearize the model about an operating point then design a controller for that linearized model and use it for the original system. For this particular system, the operating point that we concern with is $\varphi = 0$. Using the fact that $\sin \varphi \approx \varphi$ for small φ , linearize the differential equation (1) and the state-space model in question (a).
- We will use a *Proportional-Derivative* (PD) controller for the linearized model. The feedback control law is $u(t) = \alpha\varphi(t) + \beta\dot{\varphi}(t)$, where α and β are parameters that we need to design. Write the equations of the closed-loop system which consists of both the linearized model and the PD controller. For what values of the parameters α and β that the closed-loop system will be stable? (Hint: all eigenvalues of the system matrix should have negative real parts.)
- Suppose m and l are such that $ml^2 = 1$ ($kg.m^2$) and $mgl = 1$ ($N.m$). Simulate the closed-loop system using either MATLAB or Simulink with different values of α and β . You can choose any initial position $\varphi(0)$ and initial angular velocity $\dot{\varphi}(0)$. Experiment with the parameters. Plot and discuss your results.

Problem 2

In this problem, you will implement a simple symbolic reachability analysis tool for one-dimensional discrete-time dynamical systems. Though you are encouraged to use MATLAB for this exercise, you can use any programming language and tool that you like (except verification tools, of course). The programming tool that you use should have plotting capability (e.g. MATLAB, Mathematica, Maple, Python + matplotlib...).

A closed interval in \mathbb{R} , denoted $[a, b]$ where $a, b \in \mathbb{R}$ and $a \leq b$, is the set of all real numbers between a and b . More precisely, $[a, b] = \{x \in \mathbb{R} | a \leq x \leq b\}$. A popular representation of subsets of \mathbb{R} is as unions of closed intervals, for example $[0, 1] \cup [4, 5]$ is the set $\{x \in \mathbb{R} | 0 \leq x \leq 1 \text{ or } 4 \leq x \leq 5\}$.

a. Implement a programming library for representation and computations of unions of closed intervals in \mathbb{R} . Explain succinctly and rigorously your implementation of the followings (you do not need to submit your code), assuming $A = \bigcup_i [a_i, b_i]$ and $B = \bigcup_j [c_j, d_j]$:

- How did you represent closed intervals and unions of them? What data structure did you use? How did you represent the empty set?
- Union (disjunction) of sets: $A \cup B$.
- Difference of sets: $A \setminus B$.
- Check if A is a subset of B : $A \subseteq B$.
- Check for emptiness: is $A = \emptyset$.
- Sum of a set and a scalar or another set: $A + B = \{z | \exists x \in A, \exists y \in B : z = x + y\}$.
- Product of a set and a scalar: $\alpha * A = \{z | \exists x \in A : z = \alpha x\}$.
- Square of a set: $A^2 = \{z | \exists x \in A : z = x^2\}$.

You also need to implement the ability to plot a union of closed intervals as vertical line segments (see Fig. 2 for an example).

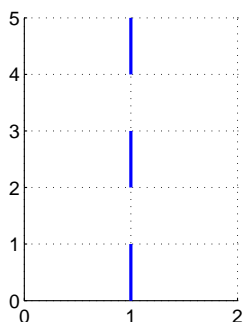


Figure 2: Plotting of the interval $[0, 1] \cup [2, 3] \cup [4, 5]$.

b. Consider a discrete-time dynamical system of the form $x_{k+1} = f(x_k, u_k)$, $k \geq 0$, where $x_k \in \mathbb{R}$ is the state, u_k is the control and is constrained in a set $U \subseteq \mathbb{R}$, $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a function using only the operations in part (a). The set of initial states is $Init$. Let $Reach_k$ be the set of reachable states at step k , i.e. $Reach_k$ is the set of all x such that there exists an execution of the system starting from some $x_0 \in Init$ under some controls u_0, u_1, \dots, u_{k-1} in U and ending at $x_k = x$. The set $Reach$ of all reachable states is $Reach = \bigcup_{k=0,1,\dots} Reach_k$. Describe a breadth-first search algorithm to compute $Reach_k$ upto a maximum number of steps N and also compute $Reach$. The algorithm should terminate as soon as there is no new state added to $Reach$, or when it reaches N steps. In the latter case, $Reach$ will be the set of reachable states for the first N steps. Implement the algorithm using the library you developed in part (a), provided that $Init$ and U are unions of closed intervals. Apply your code to the following examples. In each case, report how the algorithm stopped and at which step, print out $Reach$ (in its minimal form as a union of disjoint closed intervals), and plot $Reach_k$ for k from 0 to N .

- $x_{k+1} = -0.95x_k + u_k$, $Init = [1, 2]$, $U = [-0.1, 0.1]$, $N = 100$;
- $x_{k+1} = -0.96x_k + u_k$, $Init = [1, 2]$, $U = [-0.1, 0.1]$, $N = 100$;
- $x_{k+1} = -0.95x_k + u_k$, $Init = [1, 2]$, $U = [-0.2, 0.2]$, $N = 100$;
- $x_{k+1} = 0.5x_k^2 + u_k$, $Init = [1.8, 1.89]$, $U = [0, 0.1]$, $N = 40$;
- $x_{k+1} = 0.5x_k^2 + u_k$, $Init = [1.8, 1.9]$, $U = [0, 0.1]$, $N = 40$ (do not plot $Reach_k$).