C106B Discussion 8: Midsemester Review and Wrenches

1 Introduction

We've covered a lot of ground this semester! Let's look back on the range of topics we've done and see how it all fits together.

1. **Systems and state:** We started talking about systems, which determine how the state evolves over time. This course focuses specifically on robotic systems, so the state usually comprises of some set of variables that represent physical qualities of the robot.
2. **Control:** Robotic systems usually have some way to provide input, allowing us to control how they behave. We know that linear systems are particularly nice to work with, so feedback linearization allows us to figure out how to set our inputs to drive our state in the way we want. We also discussed Lyapunov stability, a technique to check whether our state will converge to a desired location.
3. **Constraints:** Before we plan our path, we want to ensure that certain constraints are followed because our robot might not be able to reach certain places or velocities. These come in the form of Pfaffian Constraints on our velocity $\dot{q}$, which may also limit our positions if they are integrable (holonomic).
4. **SLAM:** The map of the space we are in might not be very accurate or even nonexistent. Also, the sensors on our robot's wheel's aren't great. We want to figure out the obstacles around us while keeping an accurate estimate of our machine's location, thereby simultaneously mapping and localizing - SLAM!
5. **Path Planning:** We now have 3 important pieces of our puzzle: the limitations of our robot, how to control it, and any obstacles around us. Now, we need to actually decide how to get to our goal! This is the problem of path planning. Optimization planners, RRTs, and sinusoids all return a series of inputs that take our system from our starting position to the goal. MPC builds on this by replanning every few time steps, adjusting the input as needed and accounting for error (hence the title of controller).

2 Systems

A system tells us how the state evolves over time. It is described using a combination of the state itself (the smallest set of variables to completely describe the current configuration), the input variables, and the output vector (what we control or measure with sensors).

Problem 1: The Van der Pol oscillator is used in many circuits is used in many electric circuits because of its stable oscillations. It has applications to biology as well, modeling flow in the aorta, which carries blood out of our heart. It follows the below model:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0 \qquad (1)$$

where μ is a scalar damping coefficient. By choosing a good set of state variables, write the above model in state space form.

State: $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \rightarrow \begin{array}{l} y_1 = x \\ y_2 = \dot{x} \end{array}$

Describe as DE

$\dot{x} = Ax$

$\downarrow$

$\ddot{x} = \mu(1-x^2)\dot{x} + x$

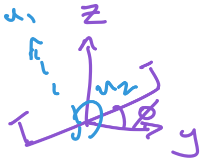
$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & \mu(1-x^2) \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

3 Control

3.1 Feedback Linearization and Dynamic Extension

Nonlinear systems can be tricky to control. We might generate a trajectory where we know exactly how we want our state to evolve. That is, we know what we want our output values or their derivatives to be. However, figuring out the inputs that lead to these can be quite difficult because the system is not perfectly invertible. As a result, we use a technique called *dynamic extension*, which involves taking the derivatives of the output until our input values appear, in order to achieve feedback linearization. Let's take an example of this below.

Problem 2: The planar quadrotor has the following system:



u_1 : upward force

u_2 : Torque

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ 0 \\ \dot{z} \\ -g \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\frac{1}{m}\sin\phi & 0 \\ 0 & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2}$$

$$\boxed{\ddot{\phi} = \frac{u_2}{I_{xx}}} \qquad \tau = I \cdot \ddot{\phi}$$

Our output f is $[y, z]^T$, the position of the planar quadrotor (We can use f instead of y to avoid confusion with the state variable). We know what we want our accelerations $\ddot{y}$ and $\ddot{z}$ to be (perhaps from path planning). Calculate a control law to determine inputs that achieve this acceleration.

$$y = A(x) \cdot u + b$$

$$u = A^{-1}(y^d - b)$$

comes from path planner

$$y = A(x)\left[A^{-1}(y^d - b)\right] + b$$

$$= y^d - b + b$$

$$y = y^d$$

PROBLEM: A might not be invertible

$$f = \begin{bmatrix} y \\ z \end{bmatrix}$$ → output → we want to control this

$$\begin{bmatrix} \ddot{y} \\ \ddot{z} \end{bmatrix}$$ → Generated from trajectory

Find inputs that let us follow a particular $\ddot{y}^{(d)}$ $\ddot{z}^{(d)}$

We know: (from the given system)

$$\underbrace{\begin{bmatrix}\ddot{y}\\ \ddot{z}\end{bmatrix}}_{\ddot{f}} = \underbrace{\begin{bmatrix}0\\ -g\end{bmatrix}}_{b} + \underbrace{\begin{bmatrix}-1/m \sin\phi & 0\\ 1/m \cos\phi & 0\end{bmatrix}}_{A}\begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

Set u_1, u_2 to

$$\underline{\vec{u} = A^{-1}(\ddot{f} - b)} \longrightarrow \text{BUT } A \text{ is not invertible}$$

Idea: 1) Take another derivative:
2) Change inputs: make a variable $v_1 = \dot{u}_1$, put u_1 in our state
→ Inputs are v_1 & u_2

$$\begin{bmatrix}y^{(3)}\\ z^{(3)}\end{bmatrix} = f(q, \dot{q}) \longrightarrow \text{Depends on } \dot{\phi}$$

$$= \begin{bmatrix}-\frac{\dot{u}_1}{m}\sin\phi - \frac{u_1}{m}\dot{\phi}\cos\phi\\ \frac{\dot{u}_1}{m}\cos\phi - \frac{u_1}{m}\dot{\phi}\sin\phi\end{bmatrix}$$

Take another derivative

→ $\ddot{\phi}$ shows up !!

$$u_2 = \frac{\ddot{\phi}}{I_{xx}}$$

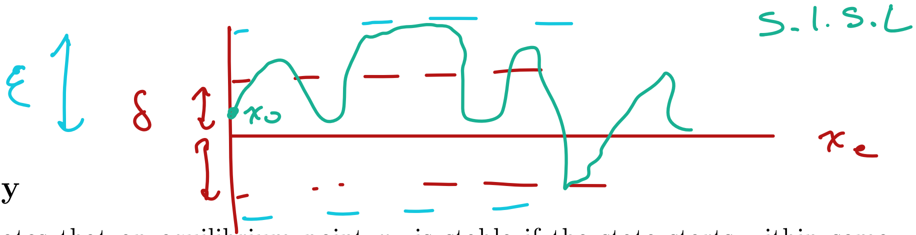
→ Inputs are $\ddot{u}_1$ and u_2

because $\begin{bmatrix}y^{(4)}\\ z^{(4)}\end{bmatrix} = \begin{bmatrix} b \end{bmatrix} + \begin{bmatrix} A \end{bmatrix}\begin{bmatrix}\ddot{u}_1\\ u_2\end{bmatrix}$

$$\boxed{\begin{bmatrix} \ddot{u}_1 \\ u_2 \end{bmatrix} = A^{-1}\left(f^{(4)} - b\right)}$$

A is now invertible because it has entries in both columns

To get u_1, integrate twice



3.2 Lyapunov Stability

Lyapunov stability formally states that an equilibrium point x_e is stable if the state starts within some distance δ of that point and remains within ϵ of it for all time. (This is a weaker definition of stability because it is local, having a particular starting condition - a stronger version exists for global exponential stability).

The Basic Theorem of Lyapunov uses the idea of a positive definite *energy function* for the system $V(x,t)$ such that if its time derivative $\dot{V}(x,t) \leq 0$ for all t, then the origin of the system is locally stable.

In your homework, you constructed a *control Lyapunov function* to drive a quadrotor along a desired trajectory. It ensures error converges to 0 rather than the position of the quadrotor. An optimization problem can be used to find inputs over time while ensuring the derivative of the Lyapunov function remains negative (or remains below a negative value, a stricter condition that speeds up convergence).

Problem 3
Now, we will directly try to drive the position to the origin. Consider the following model for a unicycle model robot. The state is (x, y, θ) which represents the position of the center of the robot relative to some fixed origin along with its current heading. The control inputs are the linear velocity v and the angular velocity ω.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{3}$$

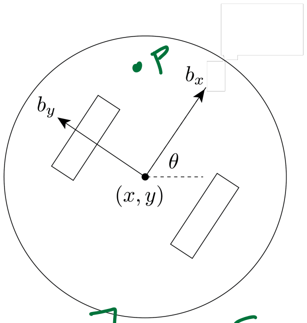
In this problem, we will explore a technique called *point-offset* control for controlling unicycle model robots like the Turtlebot. Consider a point p attached rigidly to the robot at a distance δ from the center, in front of the robot. So, the position of p is given by:

$$\begin{bmatrix}p_x\\ p_y\end{bmatrix} = \begin{bmatrix} x + \delta\cos\theta \\ y + \delta\sin\theta \end{bmatrix} \tag{4}$$

Now consider the problem of driving the turtlebot to some neighbourhood of the origin. Instead of driving the turtlebot directly, we will instead attempt to control the robot so that the point p goes to the origin. Then, the turtlebot will be in a neighbourhood of radius δ around the origin. In the next few problems, we will develop a control law to drive p to the origin, and prove its stability.

1. Let the body frame axes of the turtlebot be $b_x = (\cos\theta, \sin\theta)^T$ and $b_y = (-\sin\theta, \cos\theta)^T$, as shown in the figure blow. Show that

$$\dot{p} = v b_x + \delta\omega b_y \tag{5}$$



$$\begin{bmatrix}\dot{p}_x\\ \dot{p}_y\end{bmatrix} = \begin{bmatrix}\dot{x} - \delta\dot{\theta}\sin\theta \\ \dot{y} + \delta\dot{\theta}\cos\theta\end{bmatrix} = \begin{bmatrix} v\cos\theta - \delta w\sin\theta \\ v\sin\theta + \delta w\cos\theta \end{bmatrix}$$

$$= v b_x + \delta w b_y$$

2. Say we apply the following feedback control law on the robot:

$$v = -b_x^T p, \qquad \omega = -\frac{1}{\delta} b_y^T p \tag{6}$$

Using the Lyapunov function

$$V = \frac{1}{2} p^T p \tag{7}$$

show that the point p converges asymptotically to the origin. Is the stability global?

$$\dot p = v b_x + \delta \omega b_y = -\left((b_x^T p) b_x + (b_y^T p) b_y \right)$$

$$\text{proj} = \frac{b^T p}{\|b\|} b$$

projection of point p onto the axes of the turtlebot

$$\dot p = -p$$

$$\frac{1}{2} p^2 \rightarrow 2 p \dot p$$

$$V = \frac{1}{2} p^T p \longrightarrow \dot V = p^T \dot p = -p^T p \longrightarrow \text{negative definite}$$

$$\left.\begin{array}{l} V > 0 \\ \dot V < 0 \end{array}\right\} \text{Equilibrium } p = 0 \text{ S.S.L (Lyapunov stable)}$$

4 Constraints

We can't always hit the velocities or positions that we want it to for a particular system. As a result, the path we plan and the control law we create must follow some constraints.

A Pfaffian constraint on the velocity is expressed as $A(q)\dot q = 0$. The velocities that we can reach are therefore in the nullspace of A - these vectors meet the given condition. These constraints are *holonomic* if they are integrable and thereby also form position constraints, where $h(q) = 0$.

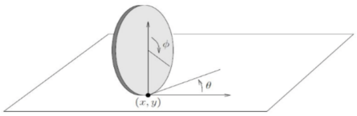
The Lie bracket of 2 vector fields $f(q)$ and $g(q)$ is given by

$$L_g f(x) = \frac{df}{dx} \cdot g(x)$$

$$\underline{\frac{\partial g}{\partial q} f(q) - \frac{\partial f}{\partial q} g(q)}$$

If the nullspace of A is given by $g_{1\ldots n}$, then the span of the g vectors, their Lie brackets, and higher order Lie brackets is the Lie algebra. If the dimension of the Lie algebra equals the dimension of the state, our system is controllable. → can reach any desired position (locally)

Problem 4: Find the Pfaffian constraints of a rolling disk. Are these holonomic? Why or why not?



5 Path Planning

We've covered multiple different types of path planners. An optimization planner meets some constraints on the motion of our robot to avoid obstacles while calculating the best possible inputs to form our path. RRTs generate small motion primitives that are chained together to move our robot towards our goal. The sinusoidal planner uses the oscillation of the sine function to generate inputs that move each part of our state, one at a time.

Problem 5: In your project, you've seen that optimization planners tend to form the best path. Would they work just as effectively on a 7-DOF robot arm in an obstacle-rich workspace? Why or why not?

6 SLAM

If our environment is unknown, we want to map out our obstacles and keep an estimate of where our robot is. However, our sensor readings on both fronts will have noise, which is assumed to be Gaussian in nature. SLAM performs the tasks of generating a plot of our space and accounting for the errors in our measurements. The front end finds features of interest, and the back end eliminates noise with a Kalman Filter.

7 Wrenches

Last semester, we started our discussion of robotic arm movement talking about *kinematics*. This deals with the different positions and angles our body frame can potentially reach. The orientation of the **B** frame with respect to the **A** frame is given by the forward kinematic map:

$$g_{AB}(\theta_1) = e^{\hat{\xi}_1 \theta_1} g_{AB}(0)$$

We then discussed *kinetics*, which deal with velocities and accelerations. The relative velocity of a point given in the body frame for some angular velocity $\dot{\theta}$ is

$$v_{q_B} = \hat{V}^b_{AB} q_B$$

$$V^b_{AB} = \xi_1 \dot{\theta} = \begin{bmatrix} v^b_{AB} \\ \omega^b_{AB} \end{bmatrix}$$

Now, we move to dynamics for an arm - analyzing the relationship between forces applied on the body and its motion! A *wrench* follows the same kind of linear/angular form as twists:

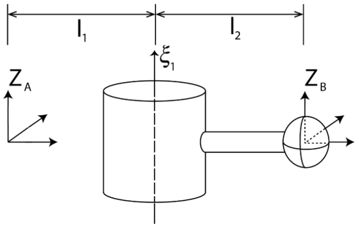
$$\Gamma = \begin{bmatrix} f \\ \tau \end{bmatrix}$$

where f is a linear force component, and τ is a torque.

Unlike angular velocities, however, to compute the total torque on some joint, we use the *transpose* of the twist.

$$\tau_1 = \xi^T \Gamma$$

Problem 6: How does a wrench applied on the **B** frame affect the torque at joint ξ_1?


l₁
l₂
ξ₁
Z_A
Z_B