EECS/ME/BioE 106B Homework 3: Path Planning

Spring 2023

Problem 1: Convex Model Predictive Control

In this problem, we'll develop some theoretical background for model predictive control (MPC), which allows us to solve path planning and control problems with a single optimization problem. In particular, we'll work on analyzing a special case of model predictive control: convex model predictive control.

In the field of optimization, there are a special class of optimization problems known as *convex optimization problems*. These problems have the special property that any local minimum of the cost function subject to the optimization constraints is also a *global* minimum!

In this question, we'll show that a special case of MPC may be formulated as the following convex optimization problem:

$$z^* = \arg\min_z \ z^T C z + c^T z + k \tag{1}$$

$$\text{s.t. } Dz + b = 0 \tag{2}$$

Where $C \succeq 0$ is a positive semidefinite matrix (symmetric and all eigenvalues ≥ 0), c is a vector, and k a scalar. With this convex optimization problem in mind, let's discuss the MPC optimization problem. Suppose we have a controllable discrete time system:

$$x(k+1) = Ax(k) + Bu(k), \ x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{3}$$

Can we devise an optimization problem such that we can drive the system to a desired state x_d optimally? We can formulate the following MPC optimization problem:

$$x^*, u^* = \arg\min_{x,u} \ (x_N - x_d)^T P (x_N - x_d) + \sum_{k=0}^{N-1} [(x_k - x_d)^T Q (x_k - x_d) + u_k^T R u_k]$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \ k = 0, 1, ..., N-1$$

$$x(0) = x_0$$

Where P, Q, R are positive semidefinite matrices (symmetric and all eigenvalues ≥ 0), N is a positive integer, and $x(0) = x_0$ is the initial condition of the system.

By solving this problem, we can find the optimal sequences of states and inputs $x_0, ..., x_{N-1}$, $u_0, ..., u_{N-1}$ that take us from our initial condition to our desired state. If we can prove that this problem is convex, we'll have shown that any state and input sequence that provides a local minimum of the cost function will also provide a global minimum of the cost function.

Questions

In this question, we'll formulate a model predictive control optimization problem for a linear system, $\dot{x} = Ax + Bu$ as the convex optimization problem:[1]

$$z^* = \arg\min z^T C z + c^T z + k, \text{ s.t. } Dz + b = 0 \tag{4}$$

1. The MPC optimization problem refers to a discrete time system, yet our system is continuous time! Recall that the general solution to the differential equation $\dot{x} = Ax + Bu$ with an initial condition $x(0) = x_0$ is given by the expression:

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \tag{5}$$

   If we use a sampling time Δt, and $u(t)$ is constant over the sampling interval $[t, t + \Delta t]$, show that we can discretize the system perfectly with no error as:

$$x(k+1) = A_d x(k) + B_d u(k) \tag{6}$$

   Where $A_d = e^{A\Delta t}$ and $B_d = \int_0^{\Delta t} e^{A(\Delta t - \tau)}Bd\tau$. This form of discretization, where we hold an input constant over a sampling interval, is known as a zero order hold discretization. *Hint: Examine the general solution for $x(t)$ from t to $t + \Delta t$.*

2. Let's now try rewriting MPC as a convex optimization problem. If z is defined:

$$z^T = \begin{bmatrix} x_0^T & x_1^T & \dots & x_N^T & u_0^T & u_1^T & \dots & u_{N-1}^T \end{bmatrix} \tag{7}$$

   Find an expression for a matrix C, a vector c, and a scalar k such that:

$$z^T C z + c^T z + k = (x_N - x_d)^T P (x_N - x_d) + \sum_{k=0}^{N-1} [(x_k - x_d)^T Q (x_k - x_d) + u_k^T R u_k] \tag{8}$$

   Then, show that your choice of C is positive semidefinite. *Hint: Try using block matrices!*

3. When reformulating the optimization constraints, it'll be useful to have a general solution for x_k. Prove by induction on k that for an initial condition x_0, x_k is given by the formula:

$$x_k = A^k x_0 + \sum_{i=0}^{k-1} A^{k-i-1} B u_i \tag{9}$$

   Hint: To prove a statement by induction, first show it is true for the base case ($k = 1$), then assume it is true for $k = p > 1$ and show this implies it is true for $k = p + 1$.

4. With the discrete time solution in mind, let's rewrite our optimization constraint in convex form. If z is defined as above, find expressions for a matrix D and a vector b such that the constraint:

$$Dz + b = 0 \tag{10}$$

   Is equivalent to the two original optimization constraints:

$$x_{k+1} = Ax_k + Bu_k, \; k = 0, 1, ..., N-1 \tag{11}$$

$$x(0) = x_0 \tag{12}$$

   After completing this step, you'll have shown that the MPC optimization problem above is actually a convex optimization problem! *Hint: Use your answer to the previous question to form a relationship $X = Bx(0) + LU$, where $x(0)$ is the initial condition, $X^T = [x_1^T, ..., x_N^T]$, $U^T = [u_1^T, ..., u_{N-1}^T]$ are vectors containing the state and input sequences. How can you get z from X, U?*

[1] Note that this problem is a QP! The extra $+k$ is simply to keep our cost function the same as the original MPC cost, but we could eliminate it without affecting the optimization solution.

Problem 2: Planning With Nonlinear Model Predictive Control

In this question, we'll focus on a more general class of model predictive control: nonlinear model predictive control (NMPC). As we'll soon see, our simple convex MPC formulation can be easily modified to incorporate constraints such as nonlinear dynamics, obstacles, input limits, and more! Note that in this more complex formulation of MPC, we can no longer in general make our optimization problem convex.
Let's think about how we can use nonlinear model predictive control to solve the path planning and control problem for a turtlebot that navigates around a set of obstacles.
The dynamics of the turtlebot are described by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{13}$$

Where x, y are the coordinates of the center of the turtlebot, ϕ is the orientation angle of the turtlebot, and v, ω are the input velocity and angular velocity to the turtlebot.
Suppose we want to drive our turtlebot from a starting position, $[x_0, y_0]$ to a final position $[x_d, y_d]$, while avoiding a set of O circular obstacles, each of radius r_i and center x_i, y_i. The following constraint:

$$(x_k - x_i)^2 + (y_k - y_i)^2 \geq r_i^2,\ k = 0, ..., N-1,\ i = 1, ..., O \tag{14}$$

Ensures that the the turtlebot always remains outside of each obstacle at each time step k.
We know that in practice, the motors of a turtlebot have limits to their strength! In MPC, we may easily account for these limits with the input constraints:

$$u_{min} \leq u_k \leq u_{max},\ k = 0, 1, ..., N-1 \tag{15}$$

Note that the inequality here is applied to each element of the input vector. Take a moment to notice how flexible the MPC optimiziation problem is in incorporating different constraints on the system! Using the constraints, we may formulate the following nonlinear MPC optimization problem for the planning and control of the turtlebot around the obstacles:

$$x^*, u^* = \arg\min_{x,u} (x_N - x_d)^T P (x_N - x_d) + \sum_{k=0}^{N-1} [(x_k - x_d)^T Q (x_k - x_d) + u_k^T R u_k] \tag{16}$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k),\ k = 0, 1, ..., N-1 \tag{17}$$

$$x(0) = x_0 \tag{18}$$

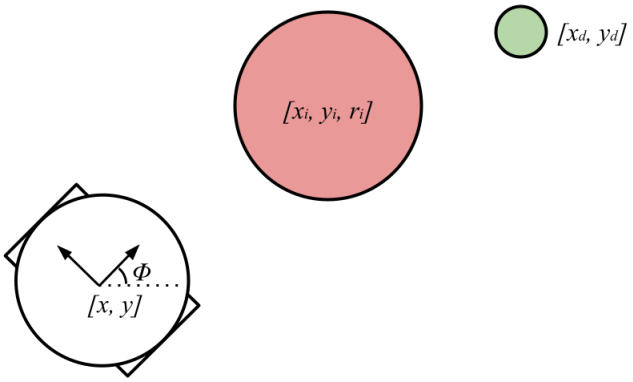
$$(x_k - x_i)^2 + (y_k - y_i)^2 \geq r_i^2,\ k = 0, ..., N-1,\ i = 0, ..., O \tag{19}$$

$$u_{min} \leq u_k \leq u_{max},\ k = 0, 1, ..., N-1 \tag{20}$$

It's important to note that although our turtlebot system is a continuous time dynamical system, the model predictive control problem is formulated for *discrete* time systems! We'll therefore have to discretize our dynamics before applying MPC.
As with before, a model predictive controller will solve this optimization problem at each time step, execute the first input in the solution sequence, re-solve at the next time step, and continue the process. Note that now, however, our problem will no longer be convex, but will rather be a general nonlinear optimization problem.

Questions

In this question, we'll implement the model predictive controller discussed above using a flexible nonlinear optimization library called CasADi. If you've never used CasADi before, we recommend reviewing the tutorial notebook here before completing this question. Casadi also has extensive online documentation. If you'd like to complete this portion in an online Colab notebook environment, please find the notebook here.



Above: Can we use MPC to plan around a set of obstacles to a goal state?

1. Before applying model predictive control, we must discretize the turtlebot dynamics from $\dot{x} = f(x, u)$ to $x(k) = f(x(k), u(k))$. Recall that if we use a sampling interval of Δt, we may use Euler discretization to discretize our system as:

$$x(k+1) = x(k) + f(x, u)\Delta t \tag{21}$$

   Go to **controller.py** and implement the *discrete_dyn()* function.

2. We may now add the constraints to our optimization problem. In **controller.py**, implement the functions *add_input_constraint()*, *add_obs_constraint()*, and *add_dyn_constraint()*.

3. Now that we've declared our optimization problem and variables, we can define our cost function. Go to **controller.py** and implement the *add_cost()* function.

4. To speed up the optimization solution time, we can "warm-start" the optimization with a guess of the solution. One potential guess for the solution is a straight line between the initial and final positions as a function of time step k. At $k = 0$, x_k should be at x_0, and at $k = N$, x_k should be at x_d. In between these states, we should plot out a smooth straight line as a function of k. Go to **controller.py** and implement the *add_warm_start()* function using either this or another warm-start guess.

5. Let's focus on some of the tuneable parameters. Recall that in this problem, we want to drive the turtlebot to a desired x, y location, and don't care about the orientation. With this in mind, explain in words why the following matrix might be a good choice for Q:

$$Q = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ \lambda_i \geq 0 \tag{22}$$

   Hint: Try multiplying out $(x - x_d)^T Q (x - x_d)$ *for arbitrary* ϕ_d.

6. To test your code, run the file **run_simulation.py**. On line 30 of this file, you can select the value of the horizon, N. Tune the values of Q, R, P and N until the turtlebot smoothly converges to the goal state. Attach the plots provided by the simulation to your solution. *Hint: Start with diagonal* Q, R, P *with weights of* 1 *- remember that these matrices must be positive semidefinite!*

Problem 3: Kinematic Constraints

Thus far in our study of the dynamics of robotic systems, we have avoided directly treating the effect of *kinematic constraints* - constraints that impact the motion - on the dynamics of the system. In this question, we'll analyze some nuances of these constraints.
Let's imagine that we have a physical system, whose dynamics may be described using a vector of generalized coordinates, $q \in \mathbb{R}^n$.

$$q = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} \in \mathbb{R}^n \tag{23}$$

Recall that in the study of dynamics, generalized coordinates help us describe the positions of different components of the system.
If we were to take a Lagrangian approach to describing the motion of our system, we would have to write out n differential equations of the form:

$$\frac{d}{dt}\Big(\frac{\partial L}{\partial \dot{q}_i}\Big) - \frac{\partial L}{\partial q} = f_{nc,i},\ 1 \leq i \leq n \tag{24}$$

Where $L = T - V$ is the Lagrangian of the system and f_{nc_i} is the set of external, or non-conservative, forces along the q_i direction. How would our dynamics change if we were to impose constraints on the motion of our system? We may constrain the velocities of our generalized coordinates, $\dot{q}$, using a *Pfaffian constraint*.

Definition 1 ***Pfaffian Constraint***
A Pfaffian constraint imposes a constraint on the velocities of the generalized coordiantes of a system. A set of k Pfaffian constraints is written:

$$\omega_i(q)\dot{q} = 0,\ i = 1, 2, ..., k \tag{25}$$

Where $\omega_i(q) = [\omega_{i1}(q), ..., \omega_{in}(q)]$ is a row vector that depends on q.

Pfaffian constraints, which directly constrain the velocities of a system, *do not* necessarily directly constrain the positions a system can go to! We can have a constraint on the velocities of our generalized coordinates *but* still be able to travel to any position, q. A Pfaffian constraint which *does* directly constrain the positions of a system is called an integrable constraint.

Definition 2 ***Integrable Pfaffian Constraint***
A Pfaffian constraint $\omega_i(q)\dot{q} = 0$ on the velocities $\dot{q}$ of a system is said to be integrable if there exists a constraint $h_i(q)$, purely on the positions q, such that:

$$\omega_i(q)\dot{q} = 0 \rightarrow h_i(q) = 0 \tag{26}$$
$$h_i(q) = 0 \rightarrow \omega_i(q)\dot{q} = 0 \tag{27}$$

Thus, a Pfaffian constraint is integrable if it is identical to a constraint *purely* on the positions of the generalized coordinates. If every constraint in a set of k Pfaffian constraints $\omega_i(q)\dot{q} = 0$ is integrable, the set of constraints is said to be *holonomic*. If none of the constrains are integrable, the set of constraints is said to be *completely nonholonomic*. As it happens, determining if a Pfaffian constraint is holonomic or nonholonomic is an extremely nontrivial process!

Questions

1. Let's begin by analyzing the effect of integrable constraints on physical systems. Suppose we have a system with a vector of generalized coordinates:

$$q = [q_1, q_2, ..., q_n]^T \in \mathbb{R}^n \tag{28}$$

We know from Lagrangian dynamics that the second order differential equations of motion of the system may be found by computing:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q} = f_{nc,i},\ 1 \leq i \leq n \tag{29}$$

Imagine that we solve for the second order ODEs of $n-1$ generalized coordinates using Lagrange's equations, where we choose $q_1, ...q_{n-1}$ from the set of generalized coordinates without loss of generality.

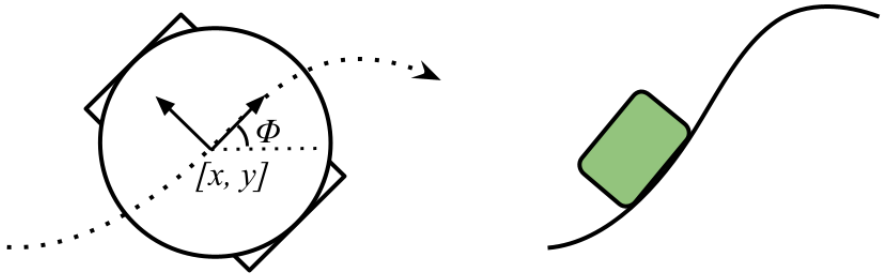
$$\ddot{q}_1, \ddot{q}_2, ..., \ddot{q}_{n-1} \tag{30}$$

Suppose we add a single holonomic Pfaffian constraint, $\omega(q)\dot{q} = 0$, to the system. Assuming the constraint is a function of q_n, show that we can use the constraint to find $\ddot{q}_n$ as a function of $q_1, q_2, ..., q_{n-1}$ and their first and second time derivatives.
This proves that an integrable constraint *reduces* the number of G.C.s we need to describe a system, as we don't need to directly apply Lagrange's equations to find $\ddot{q}_n$. *Hint: If the constraint is holonomic, is it equivalent to an algebraic constraint $h(q) = 0$?*

2. Previously, we showed that integrable constraints reduce the number of generalized coordinates we need to describe the system's motion! Using this observation, we develop the following criteria for intuitively identifying holonomic and nonholonomic constraints:

   (a) **Holonomic:** The constraint restricts the values of G.C. positions we can reach.
   (b) **Nonholonomic:** The constraint doesn't effect the G.C. positions we can reach.

   Using this criteria, explain why the constraint on the turtlebot's motion is nonholonomic while the constraint on the rollercoaster's motion is holonomic.



System 1: Turtlebot *System 2: Rollercoaster*

   You may assume the rollercoaster travels along a rigid path.

3. A turtlebot moves with the constraint that it cannot slide side to side. This means that there is *no component* of velocity perpendicular to the heading of the robot. Show that this constraint may be expressed in Pfaffian form as:

$$\omega(q)\dot{q} = \begin{bmatrix} -\sin\phi & \cos\phi & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = 0 \tag{31}$$

   This is an example of a nonholonomic constraint!

Problem 4: Nonholonomic Controllability

We know that if we solve for the Lagrangian dynamics of a system in terms of its generalized coordiantes, q, we get a system of second order differential equations of the form:

$$M\ddot{q} + C\dot{q} + N = \Gamma \tag{32}$$

Once we have these equations, we can design feedback controllers in terms of q to drive the system to some desired state. Can we exploit the structure of constraints to find another way of controlling the system? Suppose we have a set of k independent nonholonomic constraints:

$$\omega_i(q)\dot{q} = 0, 1 \leq i \leq k \tag{33}$$

If we apply k independent constraints to an n dimensional system, there are $n-k$ independent directions that $\dot{q}$ can point in! This means that the null space of $\omega(q)_i$ is $n-k$ dimensional. We may therefore define a basis for the null space:

$$g_1(q), g_2(q), ..., g_m(q),\ m = n-k \tag{34}$$

Such that $\omega_i(q)g_j(q) = 0$ for $1 \leq i \leq k, 1 \leq j \leq m$. Note that each g_j is a vector field that expresses a direction $\dot{q}$ is allowed to point in! This means that every linear combination:

$$u_1 g_1(q) + u_2 g_2(q) + ... + u_m g_m(q),\ u_i \in \mathbb{R} \tag{35}$$

Must also satisfy the constraint, where $u_i \in \mathbb{R}$ is arbitrary. Thus, if:

$$\dot{q} = u_1 g_1(q) + u_2 g_2(q) + ... + u_m g_m(q) \tag{36}$$

$\dot{q}$ *must* satisfy the Pfaffian constraint! Since u_i are arbitrary, this system is a *control system* in terms of inputs $u_1, ..., u_m$ that respects the kinematic constraints of the system. We may use this control system just like any other to move our generalized coordinates to a desired position. How do we know if we can use the inputs to drive the system to *any* position q? The answer to this question lies in *nonlinear controllability*, which we will develop with the help of a few tools.

Definition 3 ***Lie Bracket***
The Lie bracket of two vector fields $f(q), g(q)$ is defined:

$$[f, g](q) = \frac{\partial g}{\partial q} f(q) - \frac{\partial f}{\partial q} g(q) \tag{37}$$

The Lie bracket measures whether flows of equal time along f and g commute.

Definition 4 ***Lie Algebra***
The Lie algebra of a set of vector fields $\{g_1, g_2\}$, denoted $\mathcal{L}(g_1, g_2)$, is the span of all linear combinations of g_1, g_2, their Lie brackets, and higher order Lie brackets:

$$g_1, g_2, [g_1, g_2], [g_1, [g_1, g_2]], [g_2, [g_1, g_2]], ... \tag{38}$$

For a set of m vector fields, $g_1, ..., g_m$, the Lie algebra $\mathcal{L}(g_1, ..., g_m)$ is similarly defined by taking the span of the vector fields and their Lie brackets with each other.

Let's learn how to use Lie brackets and algebras to study the controllability of constrained nonlinear systems!

Questions

1. Let's return to the turtlebot system. Recall that previously, we identified that the turtlebot has the following nonholonomic constraint on its dynamics:

$$\begin{bmatrix} -\sin\phi & \cos\phi & 0 \end{bmatrix} \dot{q} = 0 \tag{39}$$

Since the turtlebot has a single constraint and three generalized coordinates (x, y, ϕ), there are two vector fields g_1, g_2 that form a basis for the null space of this constraint. Show:

$$g_1(q) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \end{bmatrix}^T, \; g_2(q) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{40}$$

Form a basis for the null space of this constraint. Then, prove that all $\dot{q}$ of the following form satisfy the constraint $\omega(q)\dot{q} = 0$, where $u_1, u_2 \in \mathbb{R}$:

$$\dot{q} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2 \tag{41}$$

2. We now have the turtlebot control system $\dot{q} = g_1(q)u_1 + g_2(q)u_2$, which uses two linearly independent vector fields, $g_1(q), g_2(q)$. Calculate the Lie bracket of these two vector fields:

$$g_3 = [g_1, g_2] \tag{42}$$

3. Now, we have three vector fields, g_1, g_2, g_3. Prove g_1, g_2, g_3 form a basis for $\mathbb{R}^3$ by showing:

$$\det \begin{bmatrix} | & | & | \\ g_1 & g_2 & g_3 \\ | & | & | \end{bmatrix} \neq 0, \; \forall q \in \mathbb{R}^3 \tag{43}$$

4. We want to devise a method to determine if we can steer the turtlebot to any state q_d from any starting state q_0 while respecting the constraint. Consider the following theorem:[2]

Theorem 1 ***Small Time Local Controllability***
A system is small time locally controllable at a point q_0 if the set of states the system can reach in finite time starting from q_0 forms a ball around q_0.
If the dimension of $\mathcal{L}(g_1, ..., g_m)$ is equal to the dimension of q, and the input vector $[u_1, ..., u_m]$ can be chosen to be any vector in $\mathbb{R}^m$, then the system:

$$\dot{q} = g_1(q)u_1 + ... + g_m(q)u_m \tag{44}$$

Is small time locally controllable.

Using this theorem, prove that the turtlebot is a small time locally controllable system. You may assume the inputs u_1, u_2 to the turtlebot can be arbitrary. *Hint: Recall that the Lie Algebra of g_1, g_2, denoted $\mathcal{L}(g_1, g_2)$, is the span of g_1, g_2, and their Lie brackets. Try using your answer to the previous part to identify its dimension!*

[2] The statement that the input may be chosen to be any vector in $\mathbb{R}^m$ may be reduced to the less restrictive condition that the positive span of the inputs is $\mathbb{R}^m$.