Lecture #9

Goals
1) Review MPC
2) Review Project
3) Trajectory Optimization
4) Solving Nonlinear Programs
5) Collocation
6) Free-Time Optimization
7) Combining Everything

III. Trajectory Optimization

Def: A *trajectory optimization problem* is

$$\min_{u:[0,1]\to\mathbb{R}^m} \int_0^1 L(t,x(t),u(t))dt + \phi(x(1))$$

$$\text{subject to} \quad \dot{x}(t) = f(t,x(t),u(t)), \quad \forall t\in[0,1]$$

$$x(0) = x_0$$

$$g_i(x(t)) \leq 0 \qquad \forall t\in[0,1], \forall i\in\{1,\dots,p_{ie}\}$$

where $L:[0,1]\times\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}$ is the *running cost* and is assumed to be twice continuously differentiable, $\phi:\mathbb{R}^n\to\mathbb{R}$ is the *final cost* and is assumed to be twice continuously differentiable, $g_i:\mathbb{R}^n\to\mathbb{R}$ are inequality constraints and are also assumed to be twice continuously differentiable and the dynamics $f:[0,1]\times\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^n$ are twice continuously differentiable.

Note: Several strategies to solve this problem:

1. indirect methods

   construct a necessary condition for optimality, then discretize this condition to generate a nonlinear program.

   Ex: $\min\limits_{x \in \mathbb{R}} J(x)$

   Necessary condition for optimality is $\nabla J(x) = 0$
   try to find zeros of $\nabla J(x)$.

   Unfortunately finding such necessary conditions for optimal control problems w/ constraints is hard.

2. direct methods

   discretize optimal control problems then solve using nonlinear programming.

$$(*) \quad \min_{\{u(\Delta T k)\}_{k=0}^{\frac{1}{\Delta T}-1}} \Delta T \sum_{k=0}^{\frac{1}{\Delta T}-1} L(\Delta T k, x(\Delta T k), u(\Delta T k)) + \varphi(x(1))$$

$$x(\Delta T(k+1)) = x(\Delta T k) + \Delta T f(\Delta T k, x(\Delta T k), u(\Delta T k)) \quad \forall k = \{0, \dots, \tfrac{1}{\Delta T}-1\}$$

$$x(0) = x_0$$

$$g_i(x(\Delta T k)) \le 0 \quad \forall k \in \{0, \dots, \tfrac{1}{\Delta T}\} \quad \forall i \in \{0, \dots, p_{ie}\}$$

How do we solve this problem?

- Use nonlinear optimization
- Need gradients of the cost and constraints to solve this problem.

IV. Solving Nonlinear Programs

Recall

$$\min_{z \in Z} \quad J(z)$$

subject to $\quad g_i(z) \leq 0 \quad \forall i \in \{1, \dots, p_{ie}\}$

$\quad h_i(z) = 0 \quad \forall i \in \{1, \dots, p_e\}$

Note: 1) the problem in the earlier section $(\star)$ can be rewritten as:

$$z = \begin{bmatrix} u(0) \\ u(\Delta T) \\ \vdots \\ u(1-\Delta T) \\ x(0) \end{bmatrix} \quad \text{so } Z = \mathbb{R}^{\frac{m}{\Delta T}} \times \mathbb{R}^n$$

Notice that

$$x(\Delta T) = x(0) + \Delta T\, f(\Delta T, x(0), u(0)) = F_1(z)$$

Can similarly define

$$x(k\Delta T) = F_k(z)$$

So the cost function and constraints in $(\star)$ can be written as a function of z.

2) All nonlinear optimization solvers have similar format. We focus on fmincon:

z= fmincon(cost, z0, A, b, Aeq, beq, lb, ub, nonlcon, options)

i.c. for optimization

$Az \leq b$

$A_{eq} z = b_{eq}$

$lb \leq z \leq ub$

3) $[J(z), \nabla J(z)] = \text{cost}(z)$

4) $[g, h, \nabla g, \nabla h] = \text{nonlcon}(z)$

$$(g(i))(z) = g_i(z) \leq 0 \quad \forall i \in \{1, \dots, p_{ie}\}$$
$$(h(i))(z) = h_i(z) = 0 \quad \forall i \in \{1, \dots, p_e\}$$

$$[\nabla g]_{ij} = \frac{\partial g_i(k)}{\partial z_j}(z)$$
$$[\nabla h]_{ij} = \frac{\partial h_i(k)}{\partial z_j}(z)$$

5) The last argument to fmincon is a set of options make sure to set them as follows:

```
options = optimoptions('fmincon', ...,
          'SpecifyConstraintGradient', true, ...,
          'SpecifyObjectiveGradient', true)
```

6) For our optimal control problem, we have to compute gradients of a constraint g: assume $p_{ie} = 1$

$$g(1) := g(F_1(z)) = g(x(0) + \Delta T f(0, x(0), u(0)))$$
$$g(2) := g(F_2(z)) = g(x(\Delta T) + \Delta T f(\Delta T, x(\Delta T), u(\Delta T)))$$

but $x(\Delta T) = x(0) + \Delta T f(0, x(0), u(0))$

↓

So $g(1)$'s gradient is nonzero only for $x(0)$ and $u(0)$ term, but $g(2)$'s gradient is nonzero for $x(0)$, $u(0)$, and $u(1)$ and derivative is complex due to application of chain rule.
Only gets worse as we keep going further along.

V Collocation

To avoid this problem of having to write a complex gradient, we will append the states of the system to the decision variables:

$$\min_{\substack{\{u(\Delta T k)\}_{k=0}^{\frac{1}{\Delta T}-1} \\ \{x(\Delta T k)\}_{k=0}^{\frac{1}{\Delta T}}}} \Delta T \sum_{k=0}^{\frac{1}{\Delta T}-1} L(\Delta T, x(\Delta T k), u(\Delta T k)) + \varphi(x(1))$$

subject to $-x(\Delta T(k+1)) + x(\Delta T k) + \Delta T f(\Delta T, x(\Delta T k), u(\Delta T k)) = 0$

$$g(x(\Delta T k)) \leq 0 \quad \forall k \in \{0, \ldots, \tfrac{1}{\Delta T}\}$$

assume for just this presentation that it is one constraint

In this case, we treat the states and inputs as independent variables that are only linked by the equality constraint:

$g(2) = g(x(2\Delta T))$

The gradient of $g(2)$ is treated as zero except for the term related to $x(2\Delta T)$.

<u>Note:</u> 1) Larger # of decision variables
2) sparser much easier to evaluate constraints
3) usually much better performance.

<u>VI.</u> Free-Time Optimization

What if our goal is to get the car from point 1 to point 2 as quickly as possible?
The final time should not be 1, but a decision variable for us to optimize over.

$t = vs$

<u>Theorem:</u> Suppose $\dot{x}(t) = f(t, x(t), u(t))$ with $t \in [0, s]$ and $x(0) = x_0$.
Suppose we define:

$$(**) \qquad \begin{bmatrix} \dot{\tau} \\ \dot{\tilde{x}} \end{bmatrix} = \begin{bmatrix} 0 \\ \tau f(\tau, \tilde{x}(t), u(st)) \end{bmatrix}$$

$\forall t \in [0, 1]$ and $\tau(0) = s$, and $\tilde{x}(0) = x_0$.

a) If $x : [0, s] \to \mathbb{R}^n$ is a solution to f, then $\tilde{x}(t) = x(t/s)$ $\forall t \in [0, 1]$ is a solution to $(**)$.

b) Similarly if $\tilde{x} : [0, 1] \to \mathbb{R}^n$ is a solution to $(**)$ then $x(st) = \tilde{x}(t)$ $\forall t \in [0, 1]$ is a solution to f.

Note: 1) By optimizing over this larger state space, one can optimize over the i.c. s to try to find the fastest route between two points:

$$\min_{\substack{\{u(\Delta T k)\}_{k=0}^{\frac{1}{\Delta T}-1} \\ \left\{\begin{bmatrix} \tau(\Delta T k) \\ \tilde{x}(\Delta T k) \end{bmatrix}\right\}_{k=0}^{\frac{1}{\Delta T}-1}}} \text{cost}$$

Subject to FWD Euler
state constraints
$\tau(0) \geq 0$

2) This transformation is called the free time transformation.

3) Be careful if you have no final cost since $\tau(0) = 0$ is a good solution.

VII Putting Everything Together.

1) Generate offline solutions using trajectory optimization

2) Use QP-MPC in real-time to follow trajectory and avoid obstacles, and deal w/ tracking errors.

Scanned by CamScanner

Issues

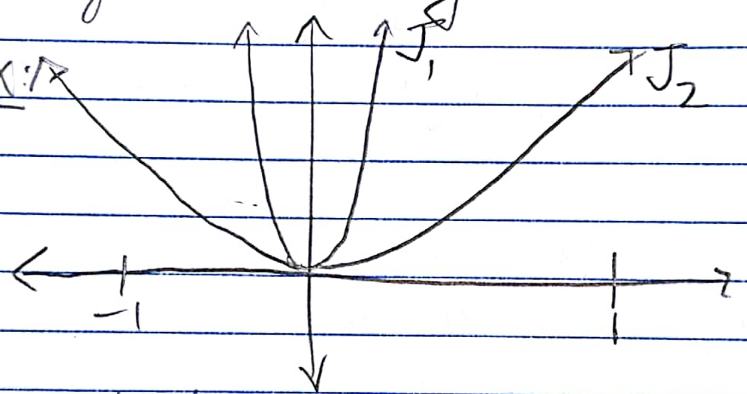
a. nonlinear programming is prone to local minima

i) initialize cleverly

ii) try different representations of costs and constraints to avoid bad minima and slow computation.

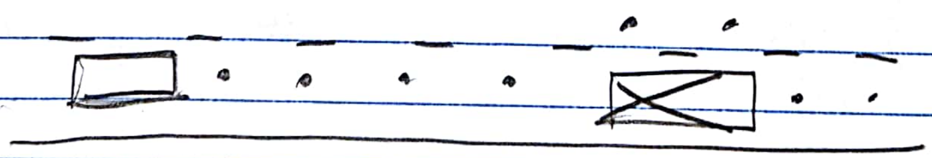
b. success of nonlinear programming relies on gradient scaling

Ex:



small changes in x lead to large changes in J_1 when compared to J_2, but both have same minima. J_1's gradients make it harder to use.

c. Be careful when you discretize:



Solver generates the dot, but the solution may clip the obstacle since it is continuous not discrete, be careful about using time free scaling and making ΔT large.