

LPE Reading Group

30 Oct 01

9

Today: 1) Gradient_{method} for Optim. Control Problems

2) Shooting (Forw., Back, Multiple)

3) RIOTS

4) Next 2 Lectures

Green
Red Erase
Blue
Black

$$\boxed{\min_x L(x)}$$

$$\boxed{\frac{\partial L}{\partial f} = -\lambda} \text{ s.t. } f(x) = 0 \quad (\underline{= 0})$$

$$\frac{\partial}{\partial x}, \frac{\partial}{\partial \lambda} \min_{x, \lambda} L(x) + \lambda^T f(x)$$

$$\underline{\lambda = -[F_x^{-1} L_x]^T}$$

$$J^* = \min_{u(\cdot)} \left\{ \int_0^T L(x, u) dt + V(x(T)) \right\}$$

$$s.t. \dot{x} = f(x, u)$$

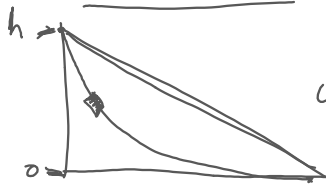
$$b \leq \gamma(x)(t) \leq ub \quad t \in [0, T]$$

$$\lambda^T(t) [F - \dot{x}]$$

$$\underline{\dot{\lambda}^T = -F_x^{-1} \lambda - L_x}$$

P. 9) Gradient

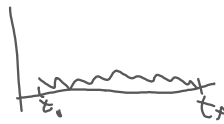
Brachistochrone



$$H_u = 0$$

comp
r

- guess $u^0(t)$, $t \in [0, t_f]$
- compute $x^0(t)$ via $\dot{x}^0 = f(x^0(t), u^0(t))$ $x(t_f^0) = x_0$
given
- compute $\lambda^0(t) \rightarrow \dot{\lambda} = -f_x^T \lambda - L_x \rightarrow$ ~~$\lambda(t_f)$~~
- compute $H_u = (L_u + \lambda^T f_u)(t)$ $\lambda(t_f) = \frac{\partial \phi^T}{\partial x(t_f)}$
- ~~set~~ set $\delta u = -\epsilon H_u^T(t)$
- repeat until
 update $\rightarrow u^1 = u^0 + \delta u$, $u^0 \leftarrow u^1$
 $\delta J = -\epsilon \int_{t_0}^{t_f} (H_u^T H_u) dt$ small



$$\underline{\dot{u}}(x, \lambda) \leftarrow \frac{\partial H}{\partial u} = 0 \Rightarrow$$

Forw. shooting a) guess $\lambda(t_0)$

b) integ. forward

Back shoot. a) $x(t_f)$

b) integ. back.

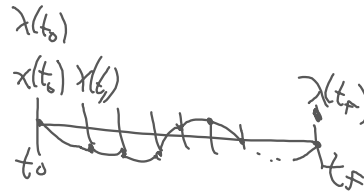
Multiple Shoot. \rightarrow

$$\begin{aligned} & \phi F(x, u^T(x, \lambda), t) \\ \dot{x} &= g_1(x, \lambda, t), \quad x(t_0) \text{ given} \\ \dot{\lambda} &= g_2(x, \lambda, t), \quad \lambda(t_f) = \left(\frac{\partial \phi}{\partial x(t_f)} \right)^T \\ & \downarrow \text{Find: } x^*(t), \lambda^*(t) \end{aligned}$$

$$\underline{\underline{-f_x | \lambda - L_x}} \quad \underline{\underline{u^*}}$$

$$\dot{x} = f(x, u)$$

$$x(t; x_0) = x_0 + \int_{t_0}^t f(x, u) dt$$



RIOTS Recursive Integration Optimal Trajectory Solver

problem

$$\min \left\{ \phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x, u, \tau) d\tau \right\}$$

s.t. $\dot{x} = f(x, u, t), x(t_0) \text{ given } \lambda(t)$

$$C(x, u, t) \leq 0, t \in [t_0, t_f]$$

$$\left[\begin{array}{l} \psi_1(x(t_f)) \leq 0 \\ \psi_2(x(t_f)) = 0 \end{array} \right]$$

Issue with " \leq "
is \rightarrow introduces combinatorial problem

Sketch:

- gradient algorithm
- computes " $u(\cdot)$ "
- gradient by the Adjoint
- ($\delta J = \int_{t_0}^{t_f} H_u \delta u dt$)
- " u " paramet. as Spline
- RK ode solver

Splines - piecewise polynomials

$$u = \sum_{i=1}^N c_i \psi_i \quad \text{Basis Functions}$$

$$\sum_{i=1}^N c_i \psi_i(t)$$

NLP solver - NPSOL

Comm. pckge (Gill, Murray, Saunders)

uses SQP

Major iterations

guess c_{coeff}

$$J(c_{coeff}) \approx J(c_{coeff}^0)$$

$$\bar{c}(c_{coeff}) \approx \bar{c}(c_{coeff}^0) + \nabla_{c_{coeff}} J \delta c_{coeff}$$

$$\bar{c}(c_{coeff}^0) + \nabla \bar{c} \delta c_{coeff} + \delta c_{coeff}^T H J \delta c_{coeff}$$

$t \in [t_0, t_f]$

st

$$[t_0, t_1]$$

$$c_1^1, \dots, c_N^1$$

$$[t_1, t_2]$$

$$c_1^2, \dots, c_N^2, \dots$$

$$\equiv c_{coeff}$$

inside R101s

• everything (x, λ) determined by u

→ new prob.

$$(*) \begin{cases} \min J(u) \\ \text{s.t. } \bar{c}(u) \leq 0 \end{cases}$$

$$x+z \leq 0$$

$$-(x+z) \leq 0$$

NLP

• u represented as Spline

converts $(*) \Rightarrow$ $\min J(c_0, \dots, c_N)$

$$\text{s.t. } \bar{c}(c_0, \dots, c_N) \leq 0$$

Nonlinear Iter's

Quad. Approx of J
 Linear \rightarrow \mathcal{C} } \rightarrow find δc_{coeff}
 Convex Problem

Minor Iterations

Solve \rightarrow

$$\min \{ \bar{J}(c_{coeff}^o) + \nabla \bar{J} \delta c_{coeff} + \delta c_{coeff}^T H \bar{J} \delta c_{coeff} \}$$

$$\rightarrow \text{s.t. } \bar{c}(c_{coeff}^o) + \nabla \bar{c} \delta c_{coeff} \leq 0$$


$$\rightarrow \delta c_{coeff}$$

update \rightarrow

$$c_{coeff}^o \leftarrow c_{coeff}^o + \delta c$$

iterate until \bar{J} small

Nonlinear Trajectory Generation \mathbb{Z}
 uses collocation: Represent (x, u) ^{states & inputs} [Those necessary to recover all (x, u) algebraically]
 as B-splines: piece-wise polynomials defined over intervals (time) by knotpoints.
 idea \rightarrow look for sol'n in subspace of \mathbb{Z} (coefficients)



$$\begin{cases} x = \cos \theta u_1 \\ y = \sin \theta u_1 \\ \dot{\theta} = \tan \theta u_1 \\ \dot{\theta} = u_2 \end{cases} \quad \begin{matrix} x \\ y \\ \theta \\ u_1 \\ u_2 \end{matrix} = \sum c_i \psi_i(t) \Big]_{\mathbb{Z}} \quad \frac{x}{y} = \frac{\cos \theta}{\sin \theta} \rightarrow \theta, u_1$$