Lecture 14: Trajectory Optimization via Collocation

- Last time: Shooting $\quad u(t) \leftarrow U(t, x, p)$

$$\min_{u(\cdot)} \int_0^{t_f} \underline{\ell}(t, x, u)\,dt + \ell_f(x(t_f)) \Rightarrow \min_{p} \overbrace{\int \underbrace{\underline{\ell}(t, x(t), U(t, x, p)}_{:= \ell(t, x, p)}\,dt + \ell_f(x(t_f))}^{L(p)}$$

$$\dot{x} = \underline{f}(t, x, u) \qquad\qquad \dot{x} = \underline{f}(t, x, U(t, x, p)) = f(t, x, p)$$

- ∞-Dimensional opt. prob
- Impossible to solve (exactly) on digital computer.

- Finite Dim. Nonlinear prog
- Objective by simulating

Today:
- Finish sensitivity analysis
- Numerical integration fundamentals
- Direct Collocation

Optimal Control Family Tree

$$\min_{u(\cdot)} \int_0^{t_f} \ell(t, x(t), u(t))\,dt + \ell_f(x(t_f))$$

Optimal Control

Dynamic Programming

Value iteration

HJB / Bellman Eq

Tabulation in State Space

- Small problems

Indirect methods

PMP

Solve a boundary value problem

- Difficult to make a good guess for $x(t)$
- "Optimize, then discretize"

Direct Methods

Transform OCP into a nonlinear programing problem

"discretize then optimize"

Direct Collocation

- Discretize $u(t)$
- Discretize $x(t)$
- Add constraints that approximate sim.

Multi Shooting

- Discretize $u(t)$
- Simulate multiple Segments
- Add constraints to make them line up

Single shooting

- Discretize $u(t)$
- Simulate to find $x(t)$

REVIEW

Getting Better Gradients: Forward sensitivity analysis $\quad x \in \mathbb{R}^n \quad p \in \mathbb{R}^s$

$$x(0) = x_0(p) \qquad x(t) = \int_0^t f(t, x(t), p)\, dt$$

How does $x(t)$ change with p:

$$\frac{\partial x(t)}{\partial p} = \int_0^t \frac{\partial}{\partial p}\left[f(t, x(t), p) \right] dt = \int_0^t \left[\frac{\partial f}{\partial p} + \frac{\partial f}{\partial x}\frac{\partial x}{\partial p} \right] dt$$

System of $n \times S$ ODES for $\partial x(t) / \partial p$

$$\boxed{\begin{aligned} \frac{d}{dt}\left[\frac{\partial x(t)}{\partial p} \right] &= \frac{\partial f}{\partial p} + \frac{\partial f}{\partial x}\left[\frac{\partial x(t)}{\partial p} \right] \\ \frac{\partial x(0)}{\partial p} &= \frac{\partial x_0}{\partial p} \end{aligned}}$$

REVIEW

Evaluating Gradients: Forward sensitivity analysis

Cost Function

$$L_f(p) = \int_0^{t_F} \ell(t, x, p)\, dt + \ell_f(x(t_F))$$

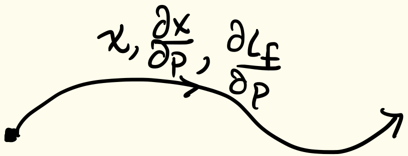
Gradient of cost:

Known from solution of ODEs on the previous slide

$$\frac{\partial L_f}{\partial p} = \int_0^{t_f} \left[\frac{\partial \ell}{\partial p} + \frac{\partial \ell}{\partial x}\frac{\partial x}{\partial p} \right] dt + \frac{\partial \ell_f}{\partial x} \cdot \frac{\partial x(t_f)}{\partial p} \Leftarrow S \text{ integrals to compute}$$

Can be computed analytically from $x(t)$, p

Summary:



Integration forward in time

Overall:

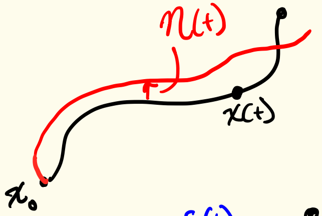
- One pass fwd in time
- $n \times s + s = (n+1)s$ integrals to finally compute

$$\frac{\partial L_f}{\partial p} \in \mathbb{R}^s$$

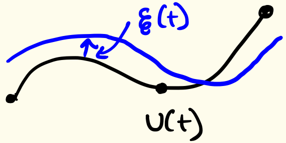
... costly when s large...

REVIEW

Toward a necessary condition for $u(\cdot)$ to be extremal:



$$V(x_0, u(\cdot)) = \int_{t_0}^{t_f} \left(\ell(t, x, u) + \lambda^T [f - \dot{x}] \right) dt + \ell_f(x(t_f))$$



Integration By Parts:

$$\int_{t_0}^{t_f} \lambda^T \dot{x}\, dt = \lambda^T x \Big|_{t_0}^{t_f} - \int_{t_0}^{t_f} \dot{\lambda}^T x\, dt$$

Functional:

$$V(x_0, u(\cdot)) = \int_{t_0}^{t_f} \Big(\overbrace{\ell + \lambda^T f}^{:= \mathcal{H}(t, x, u, \lambda)} + \dot{\lambda}^T x \Big) dt + \ell_f(x(t_f)) - \lambda^T x \Big|_{t_0}^{t_f}$$

Careful choice of $\lambda(t)$

$$\lambda(t_f) = \frac{\partial \ell_f}{\partial x}^T \Bigg|_{x = x(t_f)} \qquad \dot{\lambda} = -\left[\frac{\partial \mathcal{H}}{\partial x} \right]^T$$

Back to variation

$$V(x_0, u(\cdot) + \varepsilon(\cdot)) - V(x_0, u(\cdot)) = \int_{t_0}^{t_f} \frac{\partial H}{\partial u} \varepsilon\, dt + h.o.t.$$

Reverse Mode Sensitivity Analysis:

$$F(x): \mathbb{R}^n \to \mathbb{R}$$

$$\frac{\partial F}{\partial x} = (\nabla_x F)^T$$

$$L_f(p) = \int_{t_0}^{t_f} \big(\overbrace{l + \lambda^T f}^{:= \mathcal{H}(t, x, p, \lambda)} + \dot{\lambda}^T x \big) dt + l_f(x(t_f)) - \lambda^T x \Big|_{t_0}^{t_f}$$

$$\frac{dL_f}{dp} = \int_{t_0}^{t_f} \left[\frac{\partial \mathcal{H}}{\partial p} + \cancel{\frac{\partial \mathcal{H}}{\partial x} \frac{\partial x}{\partial p}} + \cancel{\dot{\lambda}^T \frac{\partial x}{\partial p}} \right] dt + \cancel{\frac{\partial l_f}{\partial x} \frac{\partial x(t_f)}{\partial p}} - \cancel{\lambda(t_f)^T \frac{\partial x(t_f)}{\partial p}} + \lambda(0)^T \frac{\partial x(0)}{\partial p}$$

Usual Antics

$$\lambda(t_f) = \nabla_x l_f \Big|_{x(t_f)} \qquad -\dot{\lambda} = \nabla_x \mathcal{H}$$

Then:

$$\frac{dL_f}{dp} = \int_0^{t_f} \frac{\partial \mathcal{H}}{\partial p} dt + \lambda(0)^T \frac{\partial x(0)}{\partial p} = \int_0^{t_f} \left[\frac{\partial l}{\partial p} + \lambda^T \frac{\partial f}{\partial p} \right] dt + \lambda(0)^T \frac{\partial x_0}{\partial p}$$

Reverse Mode Sensitivity Summary:

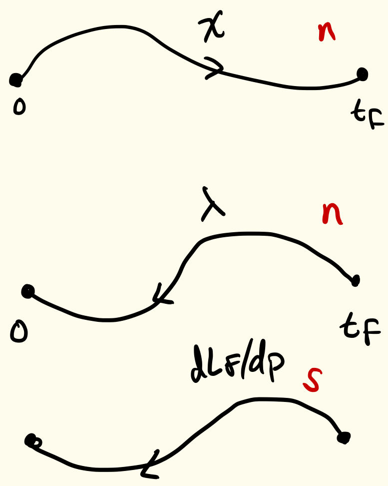
① Simulate $\dot{x} = f(t, x, p)$

② Set $\lambda(t_f) = \left[\frac{\partial l_f}{\partial x}(x(t_f))\right]^T$

$$-\dot{\lambda} = \nabla_x \mathcal{H}$$

Integrate

$$\frac{dL_f}{dp} = \int_0^{t_f} \frac{\partial \mathcal{H}}{\partial p}\, dt + \lambda(0)^T \frac{\partial x_0}{\partial p}$$



"Back prop through time"

Overall:

- 2 passes
- $n+s$ integrals to compute $\frac{dL_f}{dp} \in \mathbb{R}^s$ (more efficient $s \gg 1$)

Lecture 14: Trajectory Optimization via Collocation

- Last time: Shooting $\quad u(t) \Leftarrow U(t, x, p)$

$$\min_{U(\cdot)} \int_0^{t_f} \underline{l}(t, x(t), u(t))\, dt + l_f(x(t_f)) \Rightarrow \min_{p} \int \underbrace{l(t, x(t), U(t, x, p))}_{:= l(t, x, p)} dt + l_f(x(t_f))$$

$$\dot{x} = \underline{f}(t, x, u) \qquad\qquad \dot{x} = \underline{f}(t, x, u(t, x, p)) = f(t, x, p)$$

- ∞-dimensional optimization variable
- Impossible to carry out on a computer

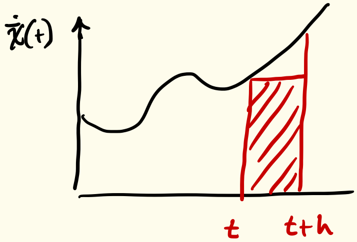
- Finite dimensional optimization variable
- Objective function computed via simulating the ODE for x

- Today:
  - Finish up sensitivity analysis
  - Numerical Integration Fundamentals
  - Direct Collocation

Numerical Integration Schemes: given $\bar{x}(t)$, approximate $\bar{x}(t+h)$

$$\dot{\bar{x}} = f(t, \bar{x})$$

First-order: Euler Integration (Forward Euler)



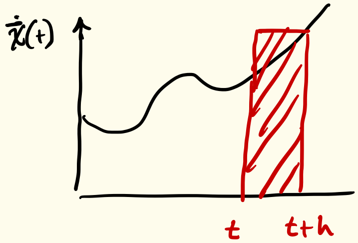
$$x(t+h) = x(t) + h\,f(t, x(t))$$

- Denote Time stepping

$$x_{k+1} = x_k + h\,f(t_k, x_k)$$

- x_{k+1} "explicitly" computed from x_k

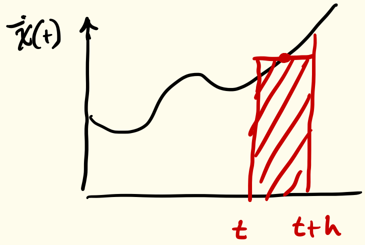
Backward Euler x_{k+1} is such that



$$x_{k+1} = x_k + h\,f(t_{k+1}, x_{k+1})$$

- Have to solve for x_{k+1}
- We call such methods "implicit" schemes

Second order Runge Kutta: Explicit

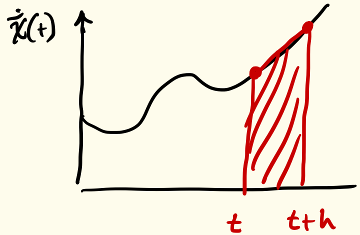


$$x_{k+1} = x_k + h\, f(t_{k+1/2}, x_{k+1/2})$$

where

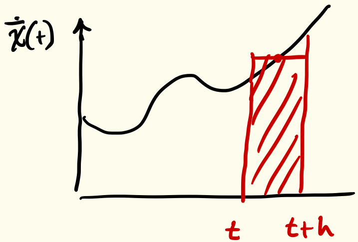
$$x_{k+1/2} = x_k + \frac{h}{2} f(t_k, x_k)$$

Trapezoidal Rule: Implicit



$$x_{k+1} = x_k + \frac{h}{2}\left(f(t_k, x_k) + f(t_{k+1}, x_{k+1})\right)$$

Modified Midpoint: Semi-implicit



$$x_{k+1} = x_k + h\, f(t_{k+1/2}, x_{k+1/2})$$

$$x_{k+1/2} = x_k + \frac{h}{2} f(t_{k+1/2}, x_{k+1/2})$$

Local Truncation Error: Suppose $x_k = \underline{X}(t)$ and let $x_{k+1} \approx \underline{X}(t+h)$ via an integration scheme. We define the local truncation error

$$LTE(h) = x_{k+1} - \underline{X}(t+h)$$

We will denote

$$LTE = O(h^n) \text{ if its on the order of } h^n \text{ for "small" } h$$

FwD Euler $x_{k+1} = x_k + h f(t_k, x_k)$

WLOG $t_k = 0$: $\underline{X}(t)$ true solution

$$X(h) = x(0) + h\dot{x}(0) + \frac{h^2}{2}\ddot{x}(\varepsilon) \qquad \varepsilon \in [0, h]$$

$$= \underbrace{x_k + h f(t_k, x_k)} + \frac{h^2}{2}\ddot{x}(\varepsilon)$$

$$= x_{k+1} + \frac{h^2}{2}\ddot{x}(\varepsilon)$$

$$LTE(h) = O(h^2)$$

Example: LTE of 2nd order explicit RK $\qquad \dot{x} = f(x) \Rightarrow \ddot{x} = \frac{\partial f}{\partial x}\dot{x} = \frac{\partial f}{\partial x} f(x)$

$$x_{k+1} = x_k + h\, f\left(x_k + \tfrac{h}{2} f(x_k)\right)$$

$$= x_k + h\left[f(x_k) + \frac{\partial f}{\partial x}\left(\tfrac{h}{2} f(x_k)\right) + O(h^2)\right]$$

$$= x_k + h f(x_k) + \frac{h^2}{2}\frac{\partial f}{\partial x} f(x_k) + O(h^3)$$

$$= \underline{x}(0) + h\,\underline{\dot{x}}(0) + \frac{h^2}{2}\underline{\ddot{x}}(0) + O(h^3)$$

$$= \left[\underline{x}(h) + O(h^3)\right] + O(h^3)$$

$$= \underline{x}(h) + O(h^3)$$

$$\Rightarrow \quad LTE = O(h^3)$$

	1st order RK	2nd order RK	...	4th order explicit RK	$n>4$ explicit nth-ord RK
LTE	$O(h^2)$	$O(h^3)$		$O(h^5)$	$O(h^{n+1})$
	1 eval of f	2 eval of f		4 evals of f	# evals of $f > n$
	1 stage	2 stage			

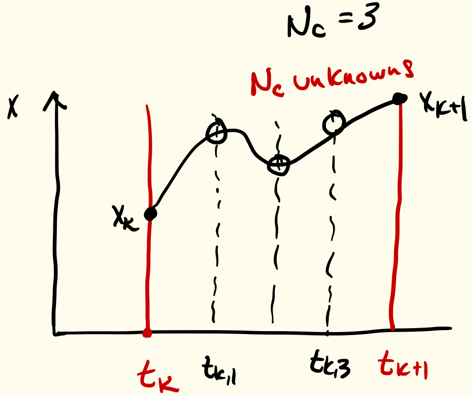
This motivates implicit methods...

Collocation: Approximating the solution as a polynomial to derive integration schemes

$$x(t) = \sum_{j=0}^{N} a_j t^j$$ TO BE DETERMINED

N+1 unknowns

$$\dot{x}(t) = \sum_{j=1}^{N} j\, a_j t^{j-1}$$



Consider N_c collocation points

$$\{t_{k,1}, \; \dots, \; t_{k,N_c}\}$$

$$\{x_{k,1}, \dots, x_{k,N_c}\}$$ State @ collocation pts

Collocation constraints

(1) $x_k = x(0)$ (N_c) $x(t_{k,i}) = x_{k,i}$ (N_c) $\dot{x}(t_{k,i}) = f(t_{k,i}, x_{k,i})$

choose $N_c = N \Rightarrow 2N_c + 1$ equations for $2N_c + 1$ unknowns

$$x_{k+1} = \sum_{j=0}^{N} a_j t_{k+1}^j \Rightarrow O(h^{N+1})$$ truncation error

Trapezoid Rule: Collocation points $\{0, 1\}$ LTE $O(h^3)$

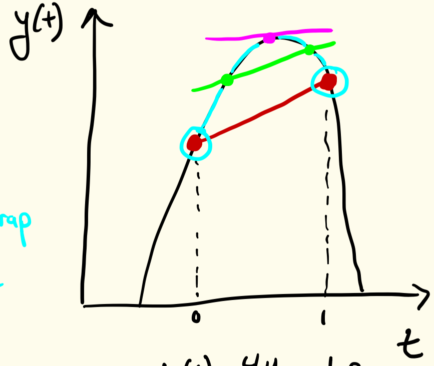
$$\int_0^1 y(t)\,dt = \tfrac{1}{2}y(0) + \tfrac{1}{2}y(1)$$

Midpoint Rule: Collocation Point $\{1/2\}$ LTE $O(h^3)$

$$\int_0^1 y(t)\,dt = y(\tfrac{1}{2})$$

This collocation point is in an optimal location ⇒ Same LTE as trap rule despite 1 less colloc pt!

$$\int_0^1 y(t)\,dt = ?$$



$y(t)$ - 4th order polynomial

Simpsons Rule: Collocation points $\{0, \tfrac{1}{2}, 1\}$ LTE $O(h^5)$

$$\int_0^1 y(t)\,dt = \tfrac{1}{6}\left(y(0) + 4y(\tfrac{1}{2}) + y(1)\right)$$

+ One more pt. than trap
+ Point in optimal spot
⇒ Two orders better LTE than Trap.

Gauss Quadrature: 2 collocation points $\{.211, .788\}$

$$\int_0^1 y(t)\,dt = \tfrac{1}{2}y(.211) + \tfrac{1}{2}y(.788)$$

LTE $O(h^5)$

- Same # pts as Trap
+ Both in the optimal location
⇒ Two orders better LTE than trap

Next Time:

- How to characterize optimal placement of collocation points
- Using collocation for direct optimization to solve an OCP
- Putting all the methods together:
  - Differential Dynamic Programming