Today: Applications

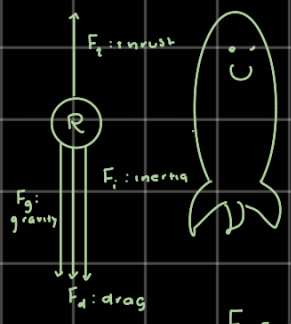
- Control (LQR)

Linear Quadratic Regulator

$x[t+1] = f(x[t], u(t))$

$u(t)$: control; $x[t]$: dynamical system

eg: Rocket



goal: maximize height by time T

$x_1(t)$: height

$x_2(t)$: vertical speed

$x_3(t)$: weight of rocket

$F_i = x_3 \ddot{x}_1 = x_3 \dot{x}_2 \longrightarrow \dot{x}_1(t) = x_2(t)$

$F_d = \underbrace{\rho(x_1)}_{\text{Friction wrt atmospheric density}} \cdot x_2^2 = C_d$

$x_3 \dot{x}_2 = -C_d \rho(x_1) x_2^2 - g x_3 + C_T u_t$

$\Rightarrow \dot{x}_2 = -\frac{C_d}{x_3} \rho(x_1) x_2^2 - g + \frac{C_T u(t)}{x_3(t)}$ } $x = x(t)$

$F_g = g x_3$

$F_T = \underbrace{C_T}_{\text{const}} \cdot \dot{x}_3$ (proportional to fuel ejection)

$\Rightarrow \dot{x}_3(t) = -u(t)$

$(x_1(0), x_2(0), x_3(0)) = (0, 0, M)$

$$\vec{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \qquad \dot{\vec{x}}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = F(\vec{x}(t), \vec{u}(t))$$

$$\max \; x_1(T)$$
$$\text{s.t. } \dot{x}(t) = F(\vec{x}(t), \vec{u}(t)) \quad \forall 0 \le t \le T$$

Back to LQR:

$\vec{x}(t+1) = A\vec{x}(t) + Bu(t)$

↳ LTI (linear-time invariant bc A & B don't depend on t)

cost (that we're trying to minimize subject to dynamics (above))

$$\sum_{t=0}^{N-1} \frac{1}{2} (\vec{x}_t^T Q \vec{x}_t + \vec{u}_t^T R \vec{u}_t) + \frac{1}{2} \vec{x}_N^T Q_f \vec{x}_N$$

Q, R: PD matrices

↳ can change Q, R fcn to account for whatever we want to optimize

↳ eg if you only care that it reaches a certain point, can choose $Q, R = 0 \Rightarrow Q_f = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ just to pull out that distance

$$\begin{aligned} &\text{minimize}_{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_N, u_1, u_2, \cdots, u_N} \quad \sum_{\tau=0}^{N-1} \frac{1}{2}\left(\vec{x}_t^{\,T} Q\, \vec{x}_t^{\,T} + \vec{u}_t^{\,T} R \vec{u}_t\right) + \frac{1}{2}\vec{x}_N Q_f \vec{x}_N \\ &\text{s.t. } \vec{x}_{t+1} = A\vec{x}_t + B\vec{u}_t \quad t = 0, 1, \cdots, N-1 \end{aligned}$$

↳ this is an equality-constrained QP

Riccati Equation

- standard derivation uses dynamic programming (Bellman)

But we'll be using Riccati (contingent on KKT conds)

- Lagrangian: $\mathcal{L}(\underbrace{\vec{x}_0, \cdots, \vec{x}_N, \vec{u}_0, \cdots, \vec{u}_N}_{\text{primal variables}}, \vec{\lambda}_1, \cdots, \vec{\lambda}_N)$

↳ N equality constraints ($\lambda_1 \to \lambda_N$)

assume symmetric matrices (for ease of calc)

$$\mathcal{L} = \sum_{\tau=0}^{N-1} \frac{1}{2}\left(\vec{x}_t^{\,T} Q \vec{x}_t + \vec{u}_t R \vec{u}_t\right) + \frac{1}{2}\vec{x}_N^{\,T} Q_f \vec{x}_N + \sum_{t=0}^{N-1} \vec{\lambda}_{t+1}^{\,T}\left(A\vec{x}_t + B\vec{u}_t - \vec{x}_{t+1}\right)$$

↳ No inequality constraints ∴ affine equality constraints

↳ Strong duality holds (Slater's cond holds)

KKT Conditions

↳ Primal feasible → not interesting (just rewriting constraints)

↳ Dual Feasibility ∴ comp. slackness → don't apply (no inequality constraints)

↳ $\nabla\mathcal{L}$ (now things get interesting!)

↳ wrt all primal variables

$$\begin{aligned} &① \quad \nabla_{\vec{u}_t}\mathcal{L} = R\vec{u}_t + B^T\vec{\lambda}_{t+1} = 0 \quad \forall t = 0, 1, \cdots, N-1 \\ &② \quad \nabla_{\vec{x}_t}\mathcal{L} = Q\vec{x}_t + A^T\vec{\lambda}_{t+1} - \vec{\lambda}_t = 0 \\ &③ \quad \nabla_{\vec{x}_N}\mathcal{L} = Q_f\vec{x}_N - \lambda_N = 0 \end{aligned} \Bigg| \; t = 1, \cdots, N-1$$

$$\begin{aligned} &② \to ④ \quad \vec{\lambda}_t = A^T\vec{\lambda}_{t+1} + Q\vec{x}_t \\ &③ \to ⑤ \quad \vec{\lambda}_N = Q_f\vec{x}_N \\ &① \to ⑥ \quad \vec{u}_t = -R^{-1}B^T\vec{\lambda}_{t+1} \end{aligned}$$

"dynamics of the adjoint system"

$\vec{\lambda}$ = co-state

Approach: Backwards induction

Reminder: goal: Find optimal $\vec{u}_t$

Induction Hypothesis : $\vec{\lambda}_{t+1} = P_{t+1} \cdot \vec{x}_{t+1}$

Base case $(t = N)$

$$\vec{\lambda}_N = Q_f \vec{x}_N \Rightarrow P_N = Q_f$$

To show $\vec{\lambda}_t = P_t \cdot \vec{x}_t$:

$$\vec{x}_{t+1} = P_{t+1} \vec{x}_{t+1} = P_{t+1} (A\vec{x}_t + B\vec{u}_t)$$

$$\vec{x}_{t+1} = P_{t+1} (A\vec{x}_t + B(-R^{-1} B^T \vec{\lambda}_{t+1}))$$

$$\vec{x}_{t+1} = P_{t+1} A\vec{x}_t - P_{t+1} B R^{-1} B^T \vec{\lambda}_{t+1}$$

$$\vec{x}_{t+1} - P_{t+1} A\vec{x}_t = P_{t+1} B R^{-1} B^T \vec{\lambda}_{t+1}$$

$$\vec{\lambda}_{t+1} = (I + P_{t+1} B R^{-1} B^T)^{-1} P_{t+1} A\vec{x}_t$$

$$\vec{\lambda}_t = A^T \vec{\lambda}_{t+1} + Q\vec{x}_t$$

$$= [A^T (I + P_{t+1} B R^{-1} B^T)^{-1} P_{t+1} A + Q] \vec{x}_t$$

this is invertible but you should prove this

↳ define this to be P_t

$$= P_t \vec{x}_t$$

Note: Now we have a recursive relationship btwn P_t & P_{t+1}

$$P_t = \text{Func}(P_{t+1})$$

& btwn λ_t & λ_{t+1}

↳ & from there we can get the $\vec{u}_t$