Norwegian University of Science and Technology

TTK4135 – Lecture 8
Open-Loop Dynamic optimization

Lecturer: Lars Imsland

Outline

- Static vs dynamic optimization (and “quasi-dynamic”)
- Dynamic optimization = optimization of dynamic systems
- How to construct objective function for dynamic optimization
- Batch approach vs recursive approach for solving dynamic optimization problems

Reference: F&H Ch. 3,4

Course overview

Optimization problems we study	Use of optimization in control
• Class: Linear programming • Method: Simplex $$\begin{aligned} \min \quad & c^\top x \\ \text{subject to} \quad & Ax \le b \\ & Cx = d \end{aligned}$$	• (Some use in MPC, but not typical)

• Class: Quadratic programming
• Method: Active set
$$\begin{aligned} \min \quad & \frac{1}{2}x^\top Gx + c^\top x \\ \text{subject to} \quad & Ax \le b \\ & Cx = d \end{aligned}$$
	• Dynamic optimization using linear models
	  • Open loop dynamic optimization
	  • Closed loop dynamic optimization/ Model Predictive Control
	  • (Linear Quadratic Control)

• Class: Nonlinear programming
• Methods
  • Without constraints: Linesearch methods
  • With constraints: SQP
$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g(x) = 0 \\ & h(x) \ge 0 \end{aligned}$$
	• Dynamic optimization using nonlinear models
	  • Nonlinear Model Predictive Control

Static vs dynamic optimization

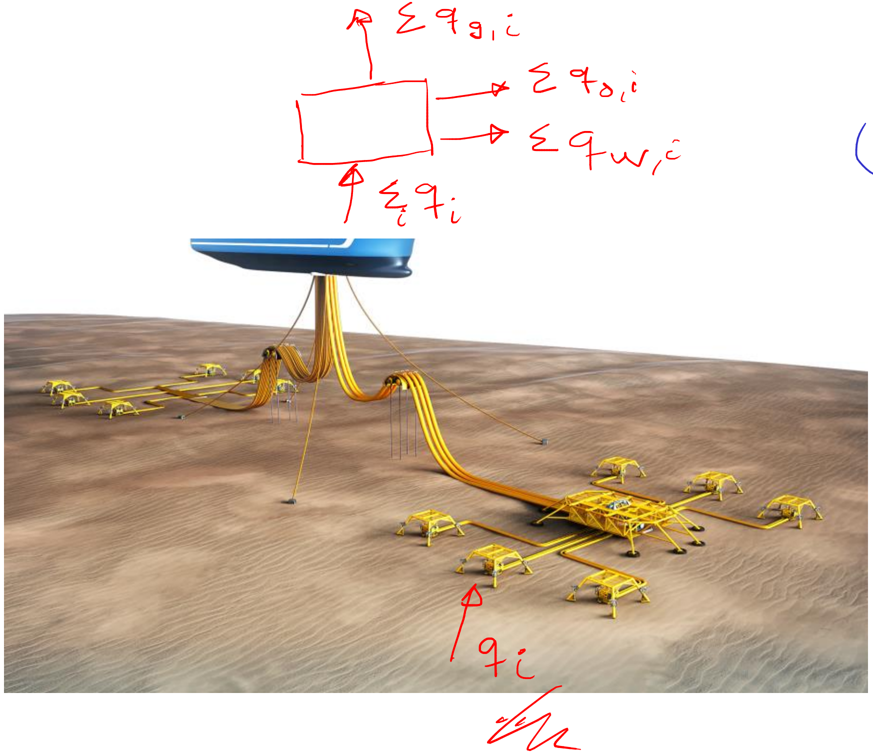
When using optimization for solving practical problems (that is, we optimize some *process*) we have two cases:

- The model of the process is time independent, resulting in static optimization
  - Common in finance, economic optimization, …
  - Recall farming example
- The model of the process is time dependent, resulting in dynamic optimization
  - The typical case in control engineering
  - The process is a mechanical system (boat, drone, robot, …), chemical process (e.g. chemical reactor, process plant, …), electrical grid, …

- F&H argues for a third option called quazi-dynamic optimization
  - The process is slowly time-varying, and can be assumed to be static for the purposes of optimization
  - We take care of the time-varying effects by re-solving regularly (or when the problem has changed sufficiently)

Oil production

(example of quasi-dynamic optimization, Ex. 2 in F&H)

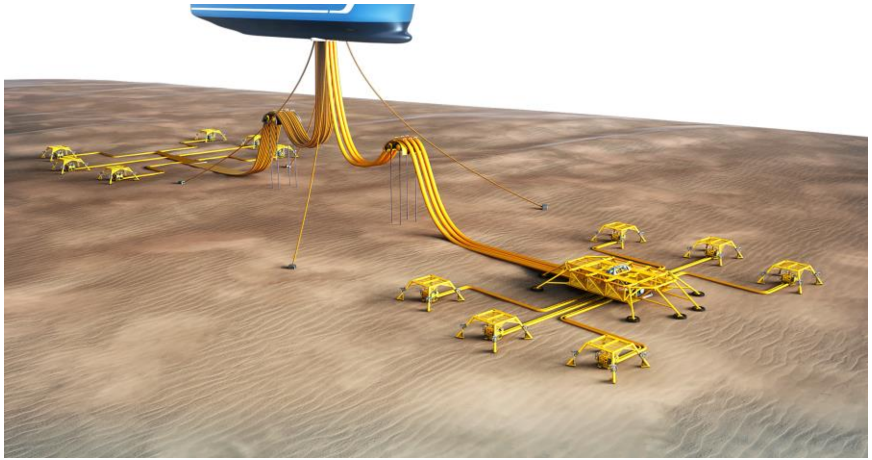


Model:

$$
(*)\quad \left\{ \begin{aligned}
q_{g,i} &= \alpha_{g,i} \cdot q_i, \quad i=1,\dots,n \\
q_{w,i} &= \alpha_{w,i} \cdot q_i, \quad i=1,\dots,n \\
q_{o,i} &= (1-\alpha_{g,i}-\alpha_{w,i})\, q_i, \quad i=1,\dots,n \\
\sum_i q_{g,i} &\le q_{g,max} \\
\sum_i q_{w,i} &\le q_{w,max}
\end{aligned} \right.
$$

Want to optimize oil production

Oil production
(example of quasi-dynamic optimization, Ex. 2 in F&H)



$$\min_{z} \; -\sum_{i=1}^{n} q_{o,i} \quad \text{s.t } (*)$$

$$z = (q_1, q_{g,1}, q_{w,1}, q_{o,1}, q_2, \ldots)^T$$

Static optimization problem.

Model has parameters:

$\alpha_{g,i}, \alpha_{w,i}, q_{g,max}, q_{w,max}$

Parameters change → re-optimize!

"quasi-dynamic"

Dynamic models, linearization
(in this course)

$$x_{t+1} = g_t(x_t, u_t) \qquad \text{(nonlinear)}$$
$$x_{t+1} = A_t x_t + B_t u_t \qquad \text{(LTV)}$$
$$x_{t+1} = A x_t + B u_t \qquad \text{(LTI)}$$

Linearization:

- About "nominal trajectory" $\bar{x}_{t+1} = g(\bar{x}_t, \bar{u}_t)$
- About "stationary point" $\bar{x} = g(\bar{x}, \bar{u})$, $\bar{x}, \bar{u}$ constant

Define perturbation: $x_t = \bar{x}_t + \delta x_t$, $u_t = \bar{u}_t + \delta u_t$

$$\cancel{\bar{x}_{t+1}} + \delta x_{t+1} = x_{t+1} = g(x_t, u_t) = g(\bar{x}_t + \delta x_t, \bar{u}_t + \delta u_t)$$
$$\approx \cancel{g(\bar{x}_t, \bar{u}_t)} + \underbrace{\left.\frac{\partial g}{\partial x}\right|_{(\bar{x}_t, \bar{u}_t)}}_{A_t} \cdot \delta x_t + \underbrace{\left.\frac{\partial g}{\partial u}\right|_{(\bar{x}_t, \bar{u}_t)}}_{B_t} \cdot \delta u_t$$

Dynamic models, linearization
(in this course)

$$x_{t+1} = g(x_t, u_t) \qquad \text{(nonlinear)}$$
$$x_{t+1} = A_t x_t + B_t u_t \qquad \text{(LTV)}$$
$$x_{t+1} = A x_t + B u_t \qquad \text{(LTI)}$$

$$\boxed{\delta x_{t+1} = A_t \, \delta x_t + B_t \, \delta u_t} \quad \text{LTV}$$

$\bar{x}_t = \bar{x}, \ \bar{u}_t = \bar{u}$: $$\boxed{\delta x_{t+1} = A \delta x_t + B \, \delta u_t} \quad \text{LTI}$$

General dynamic optimization problem

$$\min_{z} \sum_{t=0}^{N-1} f_t(x_{t+1}, u_t)$$

s.t. $x_{t+1} = g(x_t, u_t), \quad t = 0, \ldots, N-1$

$h(x_t, u_t) \leq 0, \quad t = 0, \ldots, N$

x_0: given

N: prediction horizon

$z = (u_0, x_1, u_1, x_2, u_2, \ldots, u_{N-1}, x_N)^T$

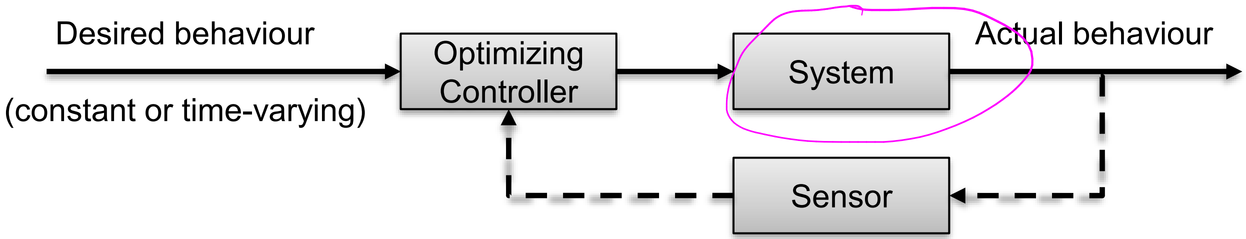
f_t: stage cost

$x^{low} \leq x_t \leq x^{high}$

$u^{low} \leq u_t \leq u^{high}$

$h(x_t, u_t)$: state- and input constr.

Possible objectives (stage costs) in dynamic optimization



Typical objectives in control:

- Penalize deviations from a constant reference/setpoint (*regulation*), or deviations from a reference trajectory (*tracking*).

Other types of objectives:

- Economic objectives. Optimize economic profit: maximize production (e.g. oil), and/or minimize costs (e.g energy or raw material)
- Limit tear and wear of equipment (e.g. valves)
- Reach a specific endpoint as fast as possible
- Reach a specific endpoint, possibly avoiding obstacles

"Standard" stage costs in dynamic optimization

General quadratic stage cost:

$$\to \quad f_t(x_{t+1}, u_t) = \frac{1}{2} x_{t+1}^T \underset{\succeq 0}{Q_t} x_{t+1} + d_{x,t}^T x_{t+1} + \frac{1}{2} u_t^T \underset{\succ 0}{R_t} u_t + d_{u,t}^T u_t$$

Special cases:

- Regulation: $y_t = H x_t \to y_{ref} = 0$. Replace x_{t+1} with y_{t+1}

$$y_{t+1}^T \tilde{Q}_t y_{t+1} = x_t^T \underbrace{H^T \tilde{Q}_t H}_{Q_t} x_t$$

$$(y_{t+1} - y_{ref})$$

"Standard" stage costs in dynamic optimization

tracking: Want $x_t \to x_{t,ref}$ (and $u_t \to u_{t,ref}$)

$$f_t(x_{t+1}, u_t) = \frac{1}{2}(x_t - x_{t,ref})^T Q (x_t - x_{t,ref}) + \frac{1}{2}(u_t - u_{t,ref})^T R (u_t - u_{t,ref})$$

$$= \frac{1}{2} x_t^T Q x_t + \underbrace{x_t^T Q x_{t,ref}}_{d_{x,t}} + \cancel{\frac{1}{2} x_{t,ref}^T Q x_{t,ref}} + \dots$$

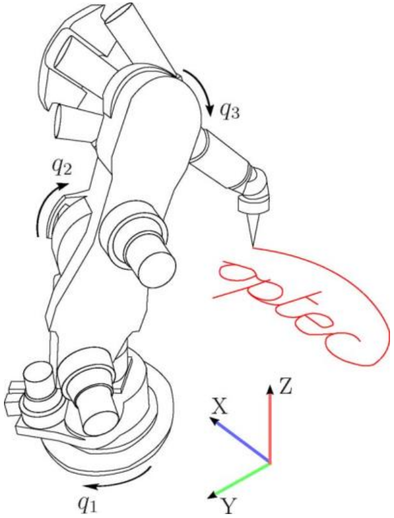
= "general case"

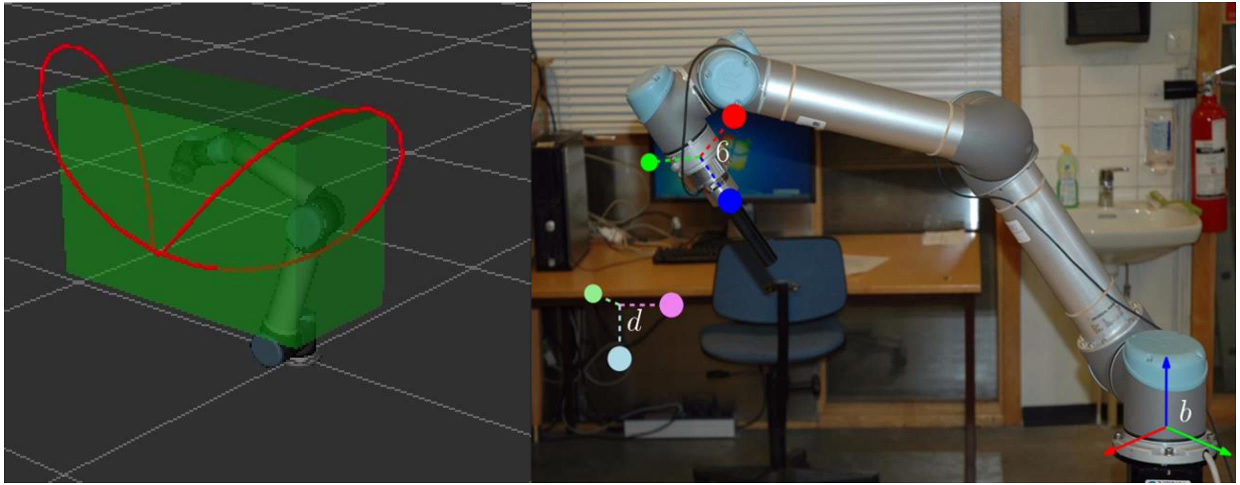
- Penalize input moves $\Delta u_t = u_t - u_{t-1} \to u_t = u_{t-1} + \Delta u_t$

$$f_t(x_{t+1}, u_t) = \dots + \frac{1}{2} \Delta u_t^T S \Delta u_t$$

$S > 0$ (instead of $R > 0$)

Examples tracking





Race car trajectory tracking (Formula student 2020, ETH)

https://www.youtube.com/watch?v=vkVBi9LWJo0
https://arxiv.org/abs/2003.04882

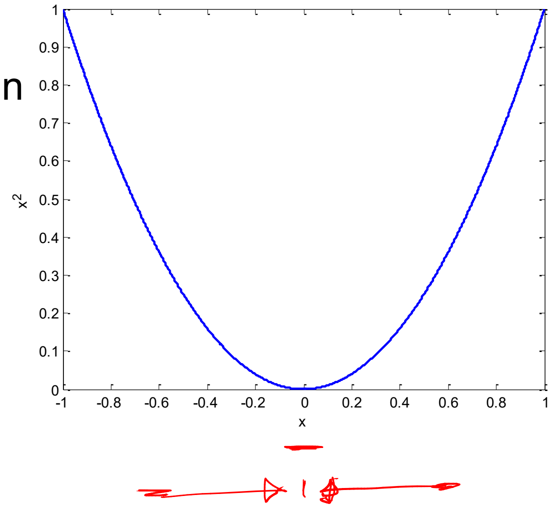
Why quadratic objective?

Two main reasons:

- Because it is convenient, mathematically
  - Smooth is good, both for analysis and numerical optimization
  - Give linear gradients

- Because it is natural; the effect is often desirable
  - Tends to ignore small deviations
  - Tends to punish large deviations

- However, other types of objective functions are also used



Standard linear dynamic optimization problem

$$\min_{z} \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q x_{t+1} + \frac{1}{2} u_t^T R u_t$$

$$\text{s.t.} \quad x_{t+1} = A x_t + B u_t, \quad t = 0, \ldots, N-1$$

$$x^{low} \leq x_t \leq x^{high}, \quad t = 1, \ldots, N$$

$$u^{low} \leq u_t \leq u^{high}, \quad t = 0, \ldots, N-1$$

$$x_0 : \text{given}$$

$$z = (u_0, x_1, u_1, x_2, \ldots, u_{N-1}, x_N)^T$$

Standard linear dynamic optimization problem

Batch approach v1, "Full space"

$x_1 = Ax_0 + Bu_0$

$x_2 = Ax_1 + Bu_1$

$$\min_{z} \ \frac{1}{2} z^T \underbrace{\begin{bmatrix} R & 0 & 0 & & \\ 0 & Q & 0 & \cdots & \\ 0 & 0 & R & & \\ & & & Q & \\ & \vdots & & & \ddots \end{bmatrix}}_{G} z \quad \text{s.t.} \quad \begin{bmatrix} -B & I & 0 & \cdots & & & 0 \\ 0 & -A & -B & I & 0 & \cdots & \\ 0 & 0 & 0 & -A & -B & I & 0 \\ & \vdots & & & & & \end{bmatrix} z = \begin{bmatrix} Ax_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$u_0 \geq u^{low}$

$u_0 \leq u^{high}$

$$\begin{bmatrix} I & 0 & 0 & 0 & \cdots & 0 \\ -I & 0 & 0 & 0 & \cdots & 0 \\ 0 & I & 0 & 0 & \cdots & 0 \\ 0 & -I & 0 & 0 & \cdots & 0 \\ & & & & & \end{bmatrix} z \geq \begin{bmatrix} u^{low} \\ -u^{high} \\ x^{low} \\ -x^{high} \\ \vdots \end{bmatrix}$$

$\boxed{\text{QP!}}$

Standard linear dynamic optimization problem

Batch approach v2, “Reduced space”

$$x_1 = A x_0 + B u_0$$

$$x_2 = A x_1 + B u_1 = A(A x_0 + B u_0) + B u_1 = A^2 x_0 + AB u_0 + B u_1$$

$$x_3 = A x_2 + B u_2 = \ldots$$

Toeplitz-matrix

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix}}_{\mathcal{X}} = \underbrace{\begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \end{bmatrix}}_{S^x} x_0 + \underbrace{\begin{bmatrix} B & 0 & 0 & 0 & \cdots \\ AB & B & 0 & 0 & \cdots \\ A^2B & AB & B & 0 & \cdots \\ \vdots & \vdots & & & \ddots \end{bmatrix}}_{S^u} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \end{bmatrix}}_{\mathcal{U}}$$

$$\boxed{\mathcal{X} = S^x x_0 + S^u \mathcal{U}}$$

Standard linear dynamic optimization problem

Batch approach v2, “Reduced space”

$$\min_{u} \ \frac{1}{2}\left(S^x x_0 + S^u u\right)^T \begin{bmatrix} Q & & & 0 \\ & Q & & \\ & & Q & \\ 0 & & & \ddots \end{bmatrix} \left(S^x x_0 + S^u u\right) + \frac{1}{2} u^T \begin{bmatrix} R & & 0 \\ & R & \\ 0 & & \ddots \end{bmatrix} u$$

$$\text{s.t.} \quad \begin{bmatrix} x^{low} \\ \vdots \\ \vdots \end{bmatrix} \leq S^x x_0 + S^u u \leq \begin{bmatrix} x^{high} \\ \vdots \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} u^{low} \\ \vdots \\ \vdots \end{bmatrix} \leq u \leq \begin{bmatrix} u^{high} \\ \vdots \\ \vdots \end{bmatrix}$$

QP!

in u

Linear quadratic control:
Dynamic optimization without (inequality) constraints

$$
\begin{aligned}
\min_{z} \sum_{t=0}^{N-1} & \, x_{t+1}^\top Q x_{t+1} + u_t^\top R u_t \\
\text{s.t.} \quad & x_{t+1} = A x_t + B u_t, \quad t = 0, 1, \dots, N-1 \\
& z = (u_0, x_1, u_1, \dots, u_{N-1}, x_N)^\top
\end{aligned}
$$

Three approaches for implementation:

- Batch approach v1, "full space" – solve as QP
- Batch approach v2, "reduced space" – solve as QP
- Recursive approach – solve as linear state feedback

Linear Quadratic Control
Batch approach v1, “Full space” QP

- Formulate with model as equality constraints, all inputs and states as optimization variables

$$
\begin{aligned}
\min_z \; & \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^\top Q x_{t+1} + \frac{1}{2} u_t^\top R u_t \\
\text{s.t.} \quad & x_{t+1} = A x_t + B u_t, \quad t = 0, 1, \dots, N-1 \\
& z = (u_0, x_1, u_1, \dots, u_{N-1}, x_N)^\top
\end{aligned}
\quad \Leftrightarrow
$$

$$
\begin{aligned}
\min_z \quad & \frac{1}{2} z^\top \begin{pmatrix} R & & & & \\ & Q & & & \\ & & R & & \\ & & & \ddots & \\ & & & & Q \end{pmatrix} z \\
\text{s.t.} \quad & \begin{pmatrix} -B & I & & & & & \\ & -A & -B & I & & & \\ & & & -A & -B & I & \\ & & & & \ddots & \ddots & \\ & & & & -A & -B & I \end{pmatrix} z = \begin{pmatrix} Ax_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\
& z = (u_0, x_1, u_1, \dots, u_{N-1}, x_N)^\top
\end{aligned}
$$

Linear Quadratic Control
Batch approach v2, "Reduced space" QP

$$\min_z \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^\top Q x_{t+1} + \frac{1}{2} u_t^\top R u_t$$
$$\text{s.t.} \quad x_{t+1} = A x_t + B u_t, \quad t = 0, 1, \ldots, N-1$$
$$z = (u_0, x_1, u_1, \ldots, u_{N-1}, x_N)^\top$$

- Use model to eliminate states as variables
  - Future states as function of inputs and initial state

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{pmatrix} x_0 + \begin{pmatrix} B & & & & \\ AB & B & & & \\ A^2 & AB & B & & \\ \vdots & \vdots & \vdots & \ddots & \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \ldots & B \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} = S^x x_0 + S^u U$$

  - Insert into objective (no constraints!)

$$\min_U \frac{1}{2} (S^x x_0 + S^u U)^\top \mathbf{Q} (S^x x_0 + S^u U) + \frac{1}{2} U^\top \mathbf{R} U$$

$$\mathbf{Q} = \begin{pmatrix} Q & & \\ & Q & \\ & & \ddots \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} R & & \\ & R & \\ & & \ddots \end{pmatrix}$$

  - Solution found by setting gradient equal to zero:

$$U = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} = -\underbrace{\left((S^u)^\top \mathbf{Q} S^u + \mathbf{R}\right)^{-1} (S^u)^\top \mathbf{Q} S^x}_{F} x_0 = -F x_0$$

Linear Quadratic Control
Recursive approach

$$\min_z \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^\top Q x_{t+1} + \frac{1}{2} u_t^\top R u_t$$
$$\text{s.t.} \quad x_{t+1} = A x_t + B u_t, \quad t = 0, 1, \ldots, N-1$$
$$z = (u_0, x_1, u_1, \ldots, u_{N-1}, x_N)^\top$$

- By writing up the KKT-conditions, we can show (we will do this later) that the solution can be formulated as:

$$u_t = -K_t x_t$$

where the feedback gain matrix is derived by

$$\left.\begin{aligned} K_t &= R^{-1} B^\top P_{t+1} (I + B R^{-1} B^\top P_{t+1})^{-1} A, && t = 0, \ldots, N-1 \\ P_t &= Q + A^\top P_{t+1} (I + B R^{-1} B^\top P_{t+1})^{-1} A, && t = 0, \ldots, N-1 \\ P_N &= Q \end{aligned}\right\} \text{Riccati}$$

Comments to the three solution approaches

- All give same numerical solution
  - If problem is strictly convex (Q psd, R pd), solution is unique
- Batch approaches give open-loop solution, recursive approach give feedback solution
  - Means the recursive solution is more robust in implementation -> always use this for LQC

$$\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} = -Fx_0 \qquad \text{vs} \qquad u_t = -K_t x_t$$

- But what if we have constraints on inputs and states:
  - **Straightforward to add constraints as inequalities to batch approaches (both becomes convex QPs)**
  - Much more difficult to add constraints to the recursive approach

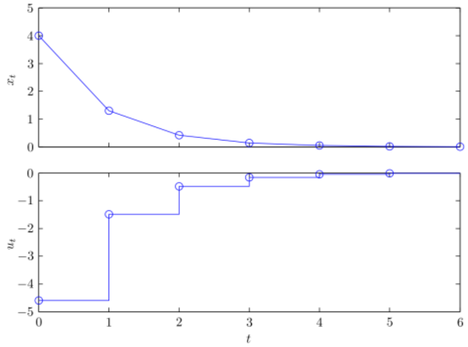
- How to to add feedback (and thereby robustness) to batch approaches?
  - **Model predictive control!**

The significance of weights

$$\min \sum_{t=0}^{5} q\, x_{t+1}^2 + r\, u_t^2$$

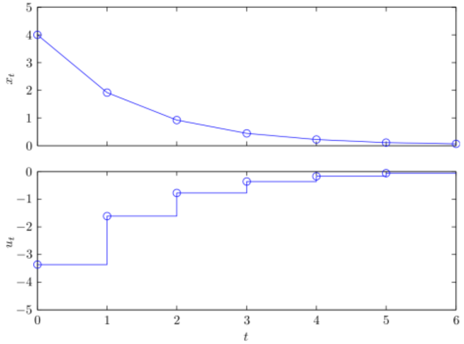
$$\text{s.t.} \quad x_{t+1} = 0.9x_t + 0.5u_t, \quad t = 0, \ldots, 5$$

$q = 5,\ r = 1$



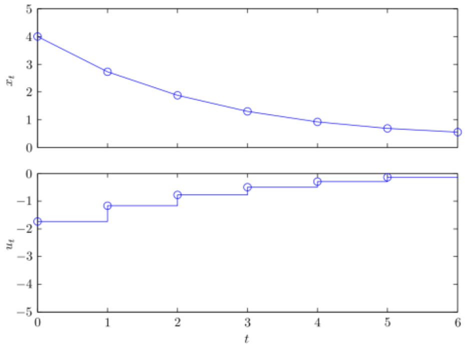
$$\sum_{t=0}^{N-1} x_{t+1}^2 = 1.9, \qquad \sum_{t=0}^{N-1} u_t^2 = 23.6$$

$q = 2,\ r = 1$



$$\sum_{t=0}^{N-1} x_{t+1}^2 = 4.8, \qquad \sum_{t=0}^{N-1} u_t^2 = 14.7$$

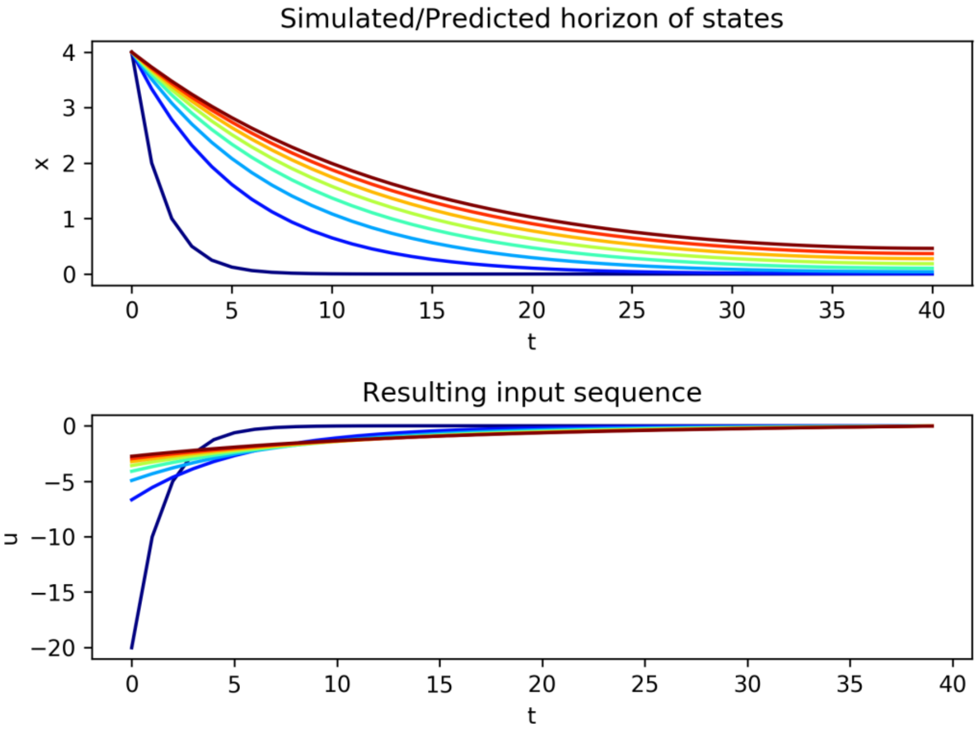
$q = 1,\ r = 2$



$$\sum_{t=0}^{N-1} x_{t+1}^2 = 14.3, \qquad \sum_{t=0}^{N-1} u_t^2 = 5.3$$

Significance of weights – Ratios

$$x_{t+1} = 1.001x_t + 0.1u_t, q = 5, r \in [0.1, \dots, 10]$$



Open loop vs closed loop

- Next week: How to use open-loop optimization for closed-loop (feedback!)
  - This is called Model Predictive Control