Lecture 8: Open loop dynamic optimization

- 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: B&H Ch. 3,4

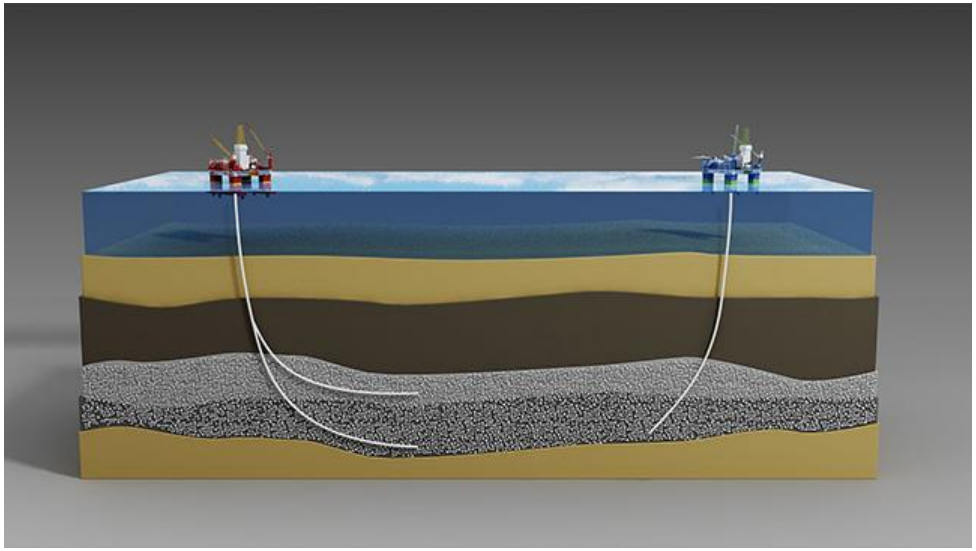
Static vs dynamic optimization

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

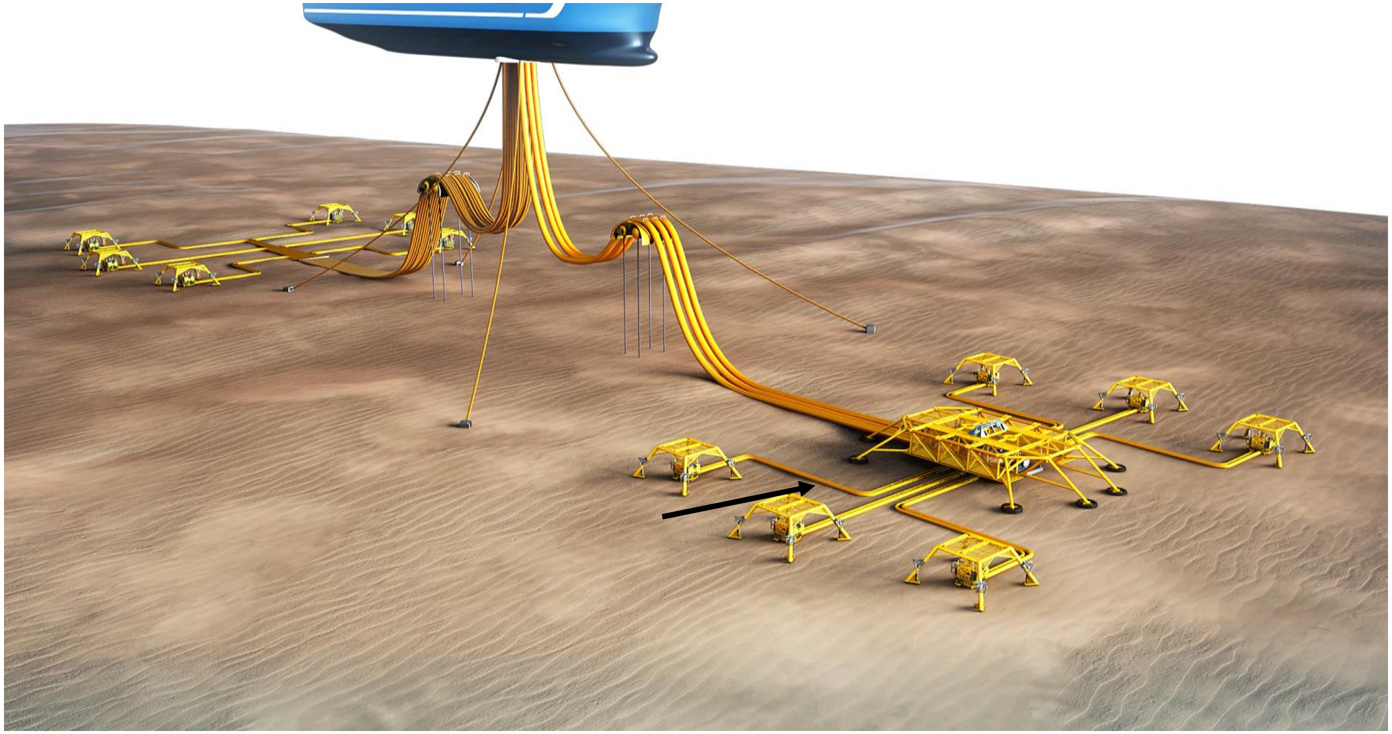
- The model of the process is time independent, resulting in static optimization
  - Common in finance, economic optimization, …
  - Recall farming example
- The model of the process is time dependent, resulting in dynamic optimization
  - The typical case in control
  - The process is a mechanical system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

- 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 resolving regularly (or when the model has changed sufficiently)

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



www.corec.no



Possible objectives in dynamic optimization

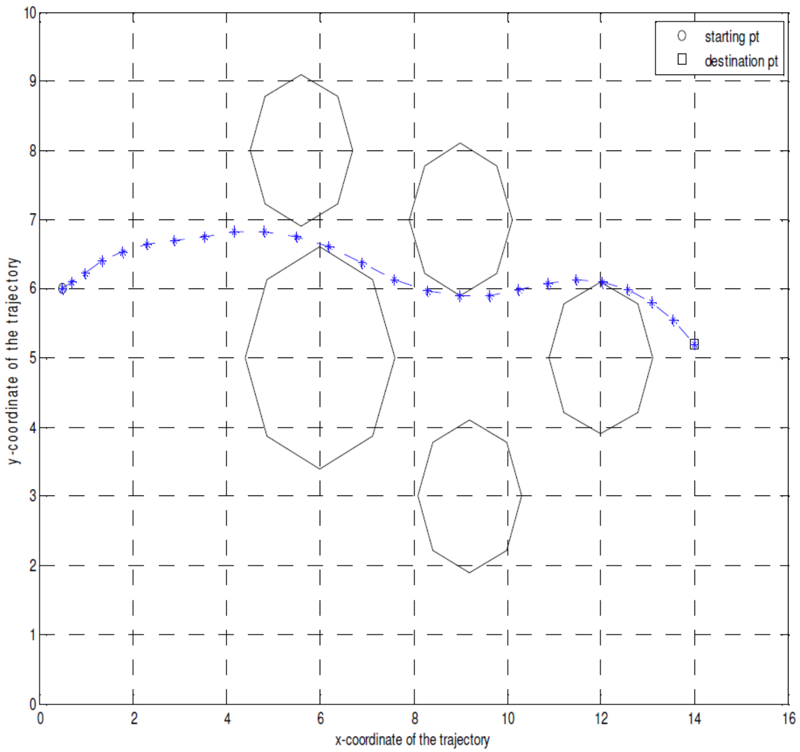
- Penalize deviations from a constant reference/setpoint (*regulation*) or deviations from a reference trajectory (*tracking*). Very often used in optimization for control.

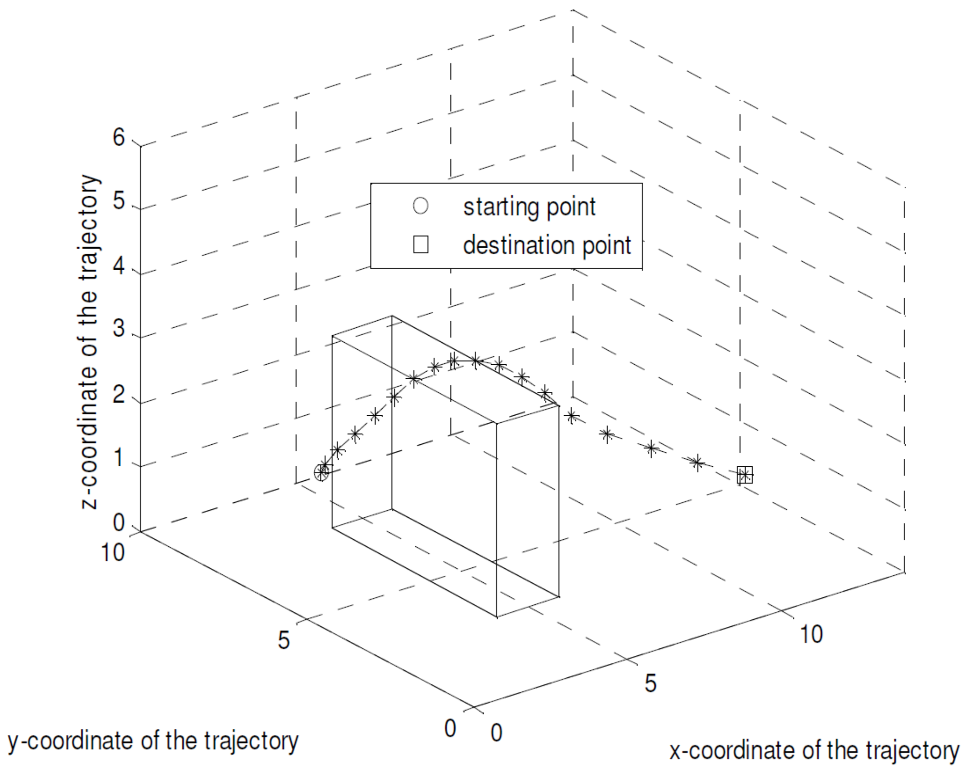
- Economic objectives. Optimize economic profit: maximize production (e.g. oil), and/or minimize costs (e.g g energy or raw material)
- Limit tear and wear of equipment (e.g. valves)

- Reach a specific endpoint, possibly avoiding obstacles
- Reach a specific endpoint as fast as possible

Example: path planning

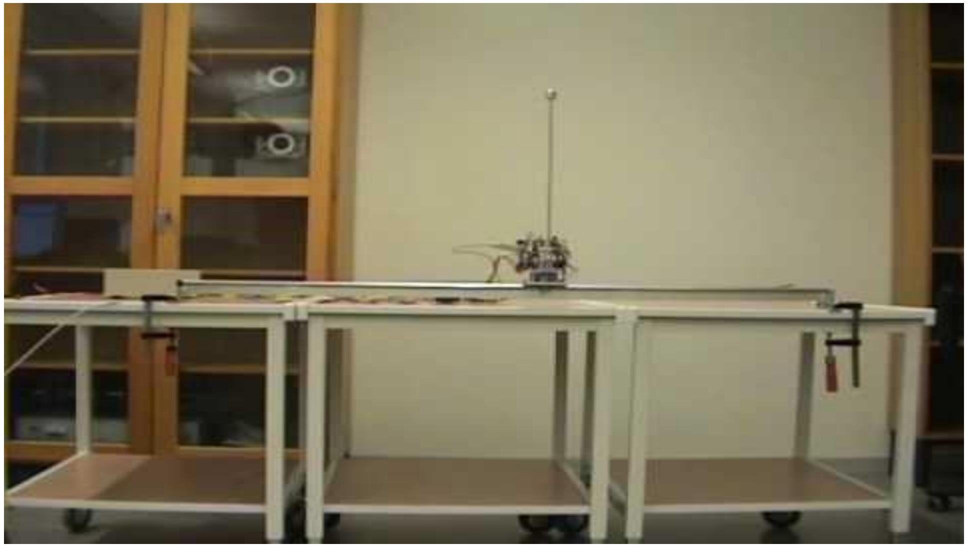
$$
\begin{aligned}
\min \quad & \text{"time from } a \text{ to } b\text{"} \\
\text{s.t.} \quad & \text{"kinematic equations"} \\
& \text{"obstacle avoidance"}
\end{aligned}
$$





Ademoye et al., Path planning via CPLEX optimization, 40th SE Symp. on System Theory, 2008.

Time-optimal pendulum trajectories

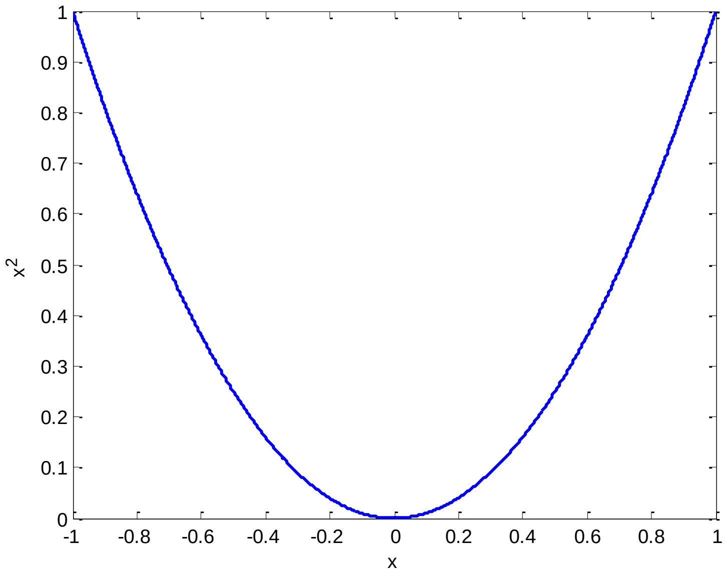


Swing-up

Obstacle avoidance

Developed in MSc-thesis by
P. Giselsson, LTH, Sweden

Why quadratic objective?



Two reasons:

- Because it is convenient, mathematically
  - for analysis and numerical optimization, and “smoothness”
  - Lead to linear gradients
- Because it is natural; the effect is often desirable
  - Tends to ignore small deviations
  - Tends to punish large deviations
- Other objectives are possible

Linear quadratic control: Dynamic optimization without 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 solution

- 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

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

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

$$\min_z \quad \frac{1}{2} z^\top \begin{pmatrix} R & & & & \\ & Q & & & \\ & & R & & \\ & & & \ddots & \\ & & & & Q \end{pmatrix} z$$

$$\text{s.t.} \begin{pmatrix} -B & I & & & & & & \\ & -A & -B & I & & & & \\ & & & -A & -B & I & & \\ & & & & \ddots & \ddots & & \\ & & & & & -A & -B & I \end{pmatrix} z = \begin{pmatrix} Ax_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

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

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

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

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

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

  - Insert into objective (no constraints!)

$$\min_U \frac{1}{2} \left(S^x x_0 + S^u U\right)^\top \mathbf{Q} \left(S^x x_0 + S^u U\right) + \frac{1}{2} U^\top \mathbf{R} U \qquad \mathbf{Q} = \begin{pmatrix} Q & & \\ & Q & \\ & & \ddots \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} R & & \\ & R & \\ & & \ddots \end{pmatrix}$$

  - Solution found by setting gradient equal to zero:

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

Linear Quadratic Control
Recursive approach

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

- 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

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

Comments to the three solution approaches

- All give same numerical solution
  - If problem is strictly convex (Q psd, R pd), solution is unique
- The batch approaches give an open-loop solution, the recursive approach give a closed-loop solution
  - Implies the recursive solution is more robust in implementation

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

- Constraints:
  - Straightforward to add constraints to batch approaches (both becomes convex QPs)
  - Much more difficult to add constraints to the recursive approach

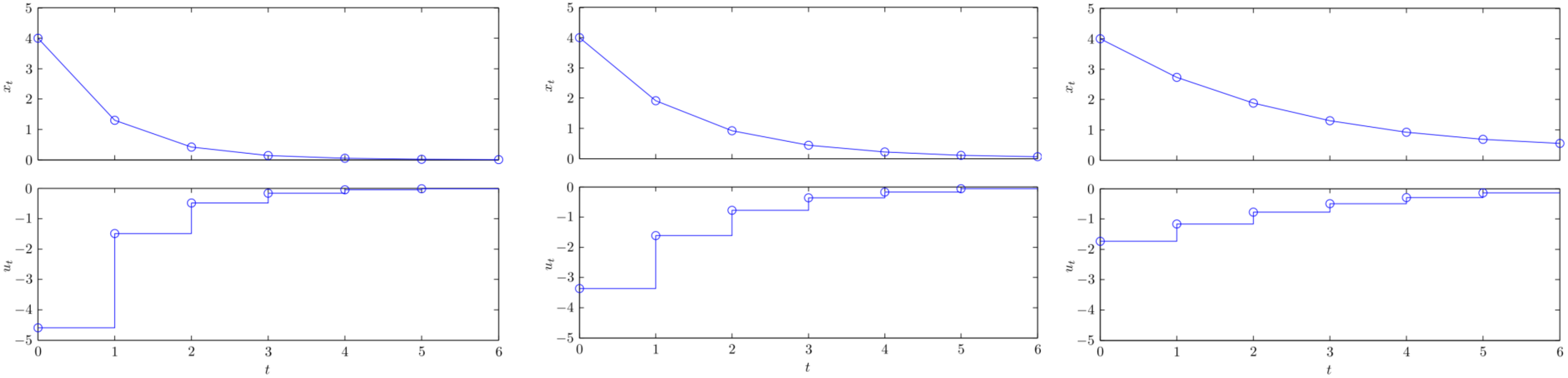
- How to to add feedback (and thereby robustness) to batch approaches?
  - Model predictive control!

The significance of weigths

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

$$\text{s.t.} \quad x_{t+1} = 0.9x_t + 0.5u_t, \quad t = 0, \dots, N-1$$

$q = 5,\ r = 1$ $\qquad$ $q = 2,\ r = 1$ $\qquad$ $q = 1,\ r = 2$



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

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

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

Open loop vs closed loop

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