A Practical Introduction to Control, Numerics and Machine Learning

Day 1

Summer School IFAC CPDE 2022

Workshop on Control of Systems Governed by Partial Differential Equations

Daniël Veldman

Chair in Dynamics, Control, and Numerics, Friedrich-Alexander-University Erlangen-Nürnberg

Contents

Chair DYNAMICS, CONTROL AND NUMERICS FAU

DDS Department of DATA SCIENCE

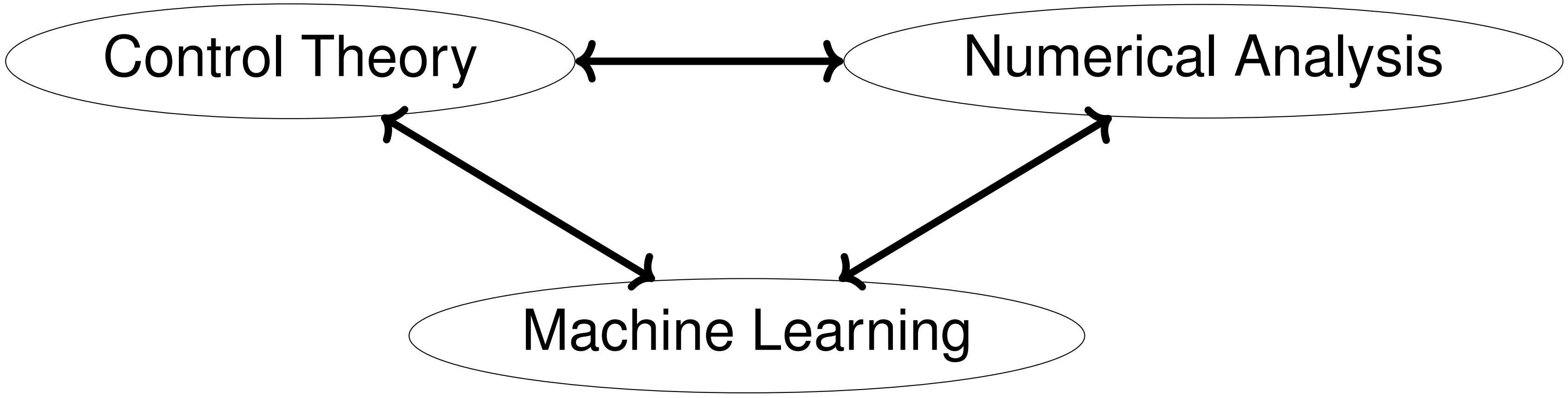
FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG NATURWISSENSCHAFTLICHE FAKULTÄT

1.A Introduction

A Practical Introduction to Control, Numerics and Machine Learning

Prof. Enrique Zuazua, Dr. Daniël Veldman.

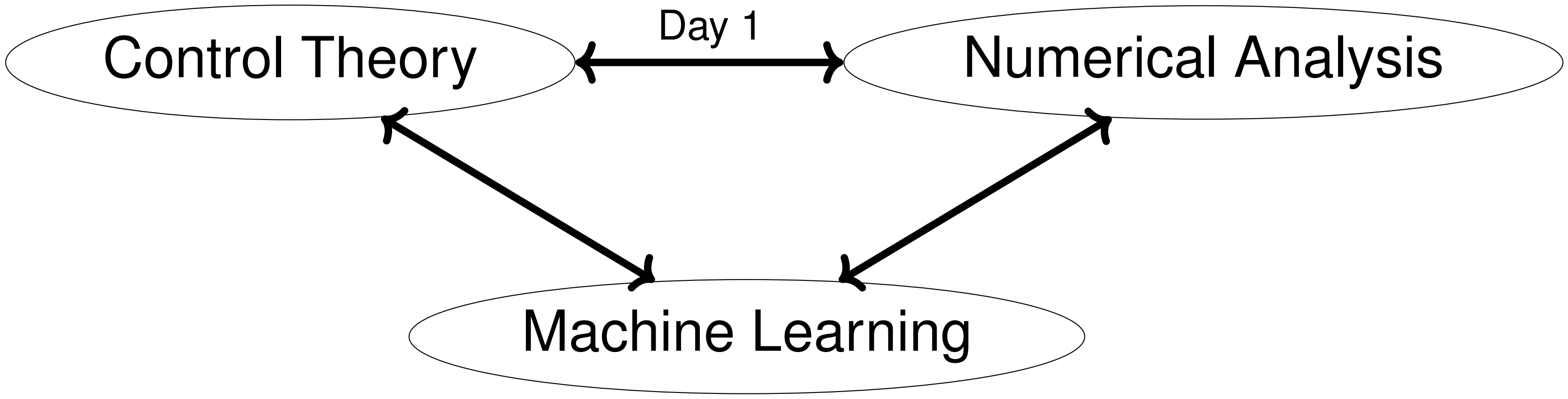
This course covers the interface between Control, Numerics, and Machine Learning (Supervised Learning and Universal Approximation)



A Practical Introduction to Control, Numerics and Machine Learning

Prof. Enrique Zuazua, Dr. Daniël Veldman.

This course covers the interface between Control, Numerics, and Machine Learning (Supervised Learning and Universal Approximation)

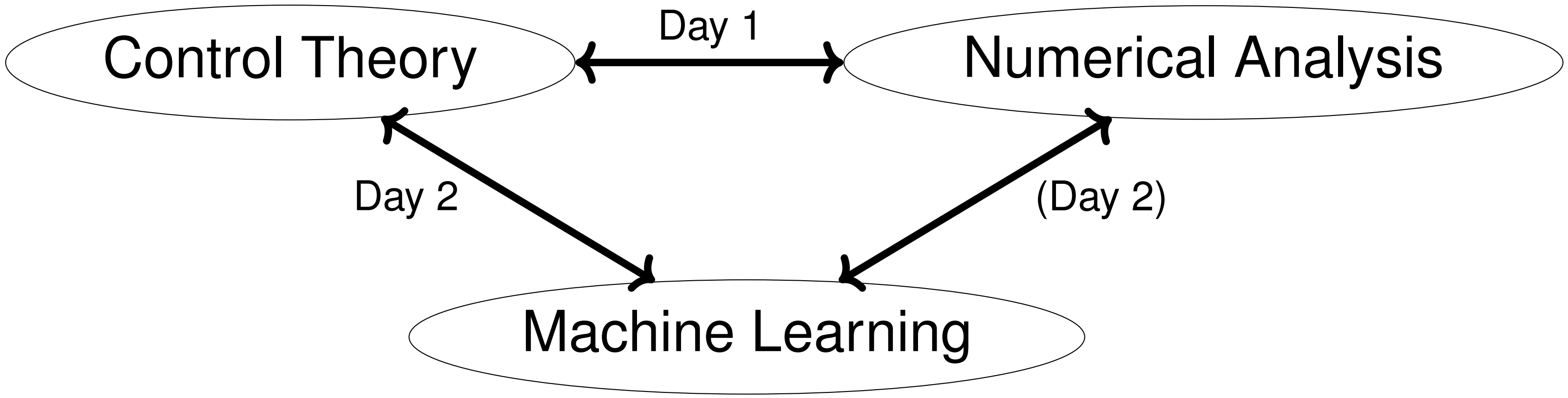


- Day 1: Discretization of optimal control problems

A Practical Introduction to Control, Numerics and Machine Learning

Prof. Enrique Zuazua, Dr. Daniël Veldman.

This course covers the interface between Control, Numerics, and Machine Learning (Supervised Learning and Universal Approximation)

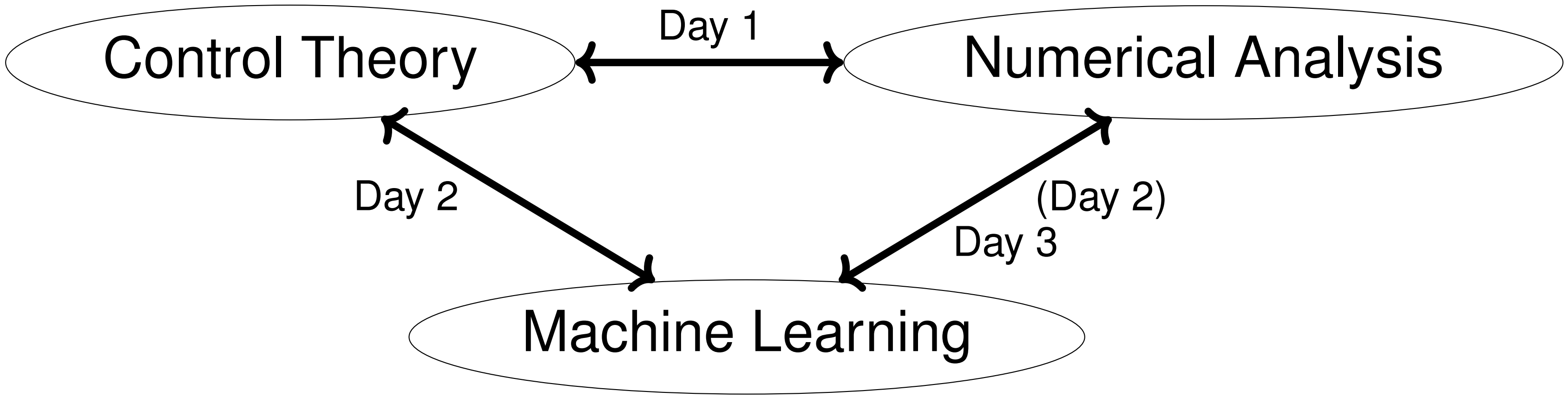


- Day 1: Discretization of optimal control problems
- Day 2: Backpropagation in (residual) neural networks

A Practical Introduction to Control, Numerics and Machine Learning

Prof. Enrique Zuazua, Dr. Daniël Veldman.

This course covers the interface between Control, Numerics, and Machine Learning (Supervised Learning and Universal Approximation)



- Day 1: Discretization of optimal control problems
- Day 2: Backpropagation in (residual) neural networks
- Day 3: Stochastic algorithms for the training of (residual) neural networks

There is a Matlab (Octave) exercise for every day.

Finite-dimensional optimal control problem

Prototypical problem:

$$\min_{\mathbf{u}\in L^2([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t)-\mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t)-\mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

subject to the dynamics

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

Finite-dimensional optimal control problem

Prototypical problem:

$$\min_{\mathbf{u}\in L^2([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t)-\mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t)-\mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

subject to the dynamics

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\mathrm{init}}.$$

Here,

- $\mathbf{u}(t) : [0,T] \to \mathbb{R}^q$ is the control,
- $\mathbf{x}(t) : [0,T] \to \mathbb{R}^N$ is the state,
- $\mathbf{x}_d(t) : [0,T] \to \mathbb{R}^N$ is the desired state (target),
- $\mathbf{w}(t) : [0,T] \to \mathbb{R}^N$ is an offset (disturbance),
- $\mathbf{x}_{\mathrm{init}} \in \mathbb{R}^N$ is the initial state,
- $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the system matrix,
- $\mathbf{E} \in \mathbb{R}^{N\times N}$ is the (invertible) mass matrix,
- $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is a positive semi-definite weighting matrix,
- $\mathbf{R} \in \mathbb{R}^{q\times q}$ is a positive definite weighting matrix.

Finite-dimensional optimal control problem

Tomorrow, we will also consider the nonlinear version

$$\min_{\mathbf{u}\in L^\infty([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \int_0^T f_0(t,\mathbf{x}(t),\mathbf{u}(t))\,\mathrm{d}t + g_0(\mathbf{x}(T)),$$

subject to the dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t,\mathbf{x}(t),\mathbf{u}(t)), \qquad\qquad \mathbf{x}(0) = \mathbf{x}_{\mathrm{init}}.$$

Finite-dimensional optimal control problem

Tomorrow, we will also consider the nonlinear version

$$\min_{\mathbf{u}\in L^\infty([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \int_0^T f_0(t,\mathbf{x}(t),\mathbf{u}(t))\,\mathrm{d}t + g_0(\mathbf{x}(T)),$$

subject to the dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t,\mathbf{x}(t),\mathbf{u}(t)), \qquad\qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

Here,

- $\mathbf{u}(t) : [0,T] \to \mathbb{R}^q$ is the control,
- $\mathbf{x}(t) : [0,T] \to \mathbb{R}^N$ is the state,
- $\mathbf{x}_{\text{init}} \in \mathbb{R}^N$ is the initial state,
- $\mathbf{F} : [0,T] \times \mathbb{R}^N \times \mathbb{R}^q \to \mathbb{R}^N$ describes the dynamics,
- $f_0 : [0,T] \times \mathbb{R}^N \times \mathbb{R}^q \to \mathbb{R}_+$ is the running cost,
- $g_0 : \mathbb{R}^N \to \mathbb{R}_+$ is the terminal cost.

Remarks

- Problems of this form appear in numerous applications.
  Mechatronics, Aeronautics, Chemical Engineering, etc.
  An example will be provided in the Matlab exercise at the end of the lecture.

- We focus on the case where the dimension N of the state space is large.
  This case for example happens when considering
  - the (spatial) discretization of Partial Differential Equations (PDEs),
  - large scale systems of interacting particles or agents,
  - training of neural networks on large data sets.

  Riccati theory cannot be applied because of large computational cost.

- For the Linear-Quadratic (LQ) problem, the minimizer $\mathbf{u}^*(t)$ exists and is unique when $\mathbf{Q} \succcurlyeq 0$ and $\mathbf{R} \succ 0$.
  (see slides in Appendix TODO)
  For the nonlinear problem, the minimizer $\mathbf{u}^*(t)$ exists (under suitable assumptions), but it does not need to be unique.

1.B A basic gradient descent algorithm

Gradient descent

Question: How to we compute the minimizer u^* of a (convex) functional $J(u)$.

Basic idea: Start from an initial guess u_0.
Compute iterates by updating u_k in the direction of the steepest descent (i.e. $-\nabla J$),

$$u_{k+1} = u_k - \beta_k \nabla J(u_k), \qquad\qquad \beta_k > 0,$$

where β denotes the step size.

Gradient descent

Question: How to we compute the minimizer u^* of a (convex) functional $J(u)$.

Basic idea: Start from an initial guess u_0.
Compute iterates by updating u_k in the direction of the steepest descent (i.e. $-\nabla J$),

$$u_{k+1} = u_k - \beta_k \nabla J(u_k), \qquad \beta_k > 0,$$

where β denotes the step size.

Three problems:

- How to compute ∇J?
- How to choose the stepsize β_k?
- When do we stop the iterations?

Computation of the gradient/ sensitivity analysis

By definition of the gradient, we have that

$$\langle \nabla J, \tilde{u} \rangle := \lim_{h \to 0} \frac{J(u + h\tilde{u}) - J(u)}{h} = \frac{\partial J}{\partial u}(u)\tilde{u},$$

for all perturbations $\tilde{u}$.

Note:

- $\nabla J(u)$ and $\frac{\partial J}{\partial u}$ are not the same:
  $\nabla J(u)$ is a column vector and $\frac{\partial J}{\partial u}$ is a row vector.
- We can use any innerproduct $\langle \cdot, \cdot \rangle$ at the LHS.
  This will not affect $\frac{\partial J}{\partial u}$ but it will change ∇J!

Computation of the gradient/ sensitivity analysis

By definition of the gradient, we have that

$$\langle \nabla J, \tilde{u} \rangle := \lim_{h\to 0} \frac{J(u+h\tilde{u}) - J(u)}{h} = \frac{\partial J}{\partial u}(u)\tilde{u},$$

for all perturbations $\tilde{u}$.

Note:

- $\nabla J(u)$ and $\frac{\partial J}{\partial u}$ are not the same:
  $\nabla J(u)$ is a column vector and $\frac{\partial J}{\partial u}$ is a row vector.
- We can use any innerproduct $\langle \cdot, \cdot \rangle$ at the LHS.
  This will not affect $\frac{\partial J}{\partial u}$ but it will change ∇J!

Two examples:

- When $\langle x, y \rangle = x^\top y$, i.e. when we use the standard Euclidean inner product

$$\nabla J = \left(\frac{\partial J}{\partial u}\right)^\top.$$

- When we use a weighted inner product $\langle x, y \rangle = x^\top \mathbf{W} y$, for a symmetric and positive definite matrix $\mathbf{W}$, we get that

$$\nabla J = \mathbf{W}^{-1}\left(\frac{\partial J}{\partial u}\right)^\top.$$

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\text{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\text{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(\mathbf{x}(\mathbf{u}), \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 1: By finite differences.
Choose a step size h (typically 10^{-5}) and approximate for every $m \in \{1, 2, \dots, M\}$

$$\left(\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}}(\mathbf{u})\right)_m = \frac{\mathrm{d}\tilde{J}}{\mathrm{d}u_m}(\mathbf{u}) \approx \frac{\tilde{J}(\mathbf{u} + h\mathbf{e}_m) - J(\mathbf{u})}{h} = \frac{J(\mathbf{x} + \delta\mathbf{x}_m, \mathbf{u} + h\mathbf{e}_m) - J(\mathbf{x}, \mathbf{u})}{h},$$

where $\delta\mathbf{x}_m$ satisfies

$$\mathbf{A}\delta\mathbf{x}_m + h\mathbf{B}\mathbf{e}_m = \mathbf{0}.$$

Note: we need to solve M linear systems in N unknowns.
This is very time-consuming when M and N are large.

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\mathrm{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(-\mathbf{A}^{-1}\mathbf{B}\mathbf{u}, \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 2: Analytically.
Similarly, as in the exercise we can use the chain rule to find

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = \frac{\partial J}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}} + \frac{\partial J}{\partial \mathbf{u}} = -\frac{\partial J}{\partial \mathbf{x}}\mathbf{A}^{-1}\mathbf{B} + \frac{\partial J}{\partial \mathbf{u}}.$$

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\mathrm{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(-\mathbf{A}^{-1}\mathbf{B}\mathbf{u}, \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 2: Analytically.
Similarly, as in the exercise we can use the chain rule to find

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = \frac{\partial J}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}} + \frac{\partial J}{\partial \mathbf{u}} = -\frac{\partial J}{\partial \mathbf{x}}\mathbf{A}^{-1}\mathbf{B} + \frac{\partial J}{\partial \mathbf{u}}.$$

The computational cost depends on where you put the brackets:

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = -\frac{\partial J}{\partial \mathbf{x}}(\mathbf{A}^{-1}\mathbf{B}) + \frac{\partial J}{\partial \mathbf{u}} = -\left(\frac{\partial J}{\partial \mathbf{x}}\mathbf{A}^{-1}\right)\mathbf{B} + \frac{\partial J}{\partial \mathbf{u}}.$$

Note: the first expression requires the solution of M linear system in N unknowns, whereas the second requires requires the solution of 1 linear system in N unknowns.

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\text{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\text{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(\mathbf{x}(\mathbf{u}), \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 3: Using the Lagrangian.
Introduce the vector of Lagrange multipliers $\boldsymbol{\lambda}$ and form the Lagrangian

$$\mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = J(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}).$$

Take the partial derivative w.r.t. $\mathbf{u}$ to find the Jacobian

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = \frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \frac{\partial J}{\partial \mathbf{u}} + \boldsymbol{\lambda}^\top \mathbf{B}\mathbf{u}.$$

Set the partial derivative w.r.t. $\mathbf{x}$ to zero to determine $\boldsymbol{\lambda}$:

$$\mathbf{0} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \boldsymbol{\lambda}^\top \mathbf{A}, \qquad -\boldsymbol{\lambda}^\top = \frac{\partial J}{\partial \mathbf{x}} \mathbf{A}^{-1}, \qquad \boldsymbol{\lambda} = -\left(\mathbf{A}^\top\right)^{-1} \left(\frac{\partial J}{\partial \mathbf{x}}\right)^\top.$$

The result is the same as for answer 2 (with well-placed brackets).

The choice of the step size

We have that

$$\begin{aligned} J(u_{k+1}) = J(u_k - \beta_k \nabla J(u_k)) &= J(u_k) - \beta_k \frac{\partial J}{\partial u_k} \nabla J(u_k) + O(\beta_k^2) \\ &= J(u_k) - \beta_k \langle \nabla J(u_k), \nabla J(u_k) \rangle + O(\beta_k^2). \end{aligned}$$

As long as we are not at a critical point ($\nabla J(u_k) = 0$) $\langle \nabla J(u_k), \nabla J(u_k) \rangle > 0$, so

$$J(u_{k+1}) < J(u_k)$$

for $\beta_k > 0$ small enough.

The choice of the step size

We have that

$$\begin{aligned} J(u_{k+1}) = J(u_k - \beta_k \nabla J(u_k)) &= J(u_k) - \beta_k \frac{\partial J}{\partial u_k} \nabla J(u_k) + O(\beta_k^2) \\ &= J(u_k) - \beta_k \langle \nabla J(u_k), \nabla J(u_k) \rangle + O(\beta_k^2). \end{aligned}$$

As long as we are not at a critical point ($\nabla J(u_k) = 0$) $\langle \nabla J(u_k), \nabla J(u_k) \rangle > 0$, so

$$J(u_{k+1}) < J(u_k)$$

for $\beta_k > 0$ small enough.

We can thus take the following simple but effective approach (used at every iteration).

- Choose a step size $\beta > 0$.
- Compute $J(u_k - \beta \nabla J(u_k))$.
- If $J(u_k - \beta \nabla J(u_k)) < J(u_k)$, we accept this step size.
  If not, we reduce the step size (e.g. by a factor 2) and recompute $J(u_k - \beta \nabla J(u_k))$.

This should always lead to a $\beta_k > 0$ such that $J(u_k - \beta_k \nabla J(u_k)) < J(u_k)$.
(Provided that $\nabla J(u_k)$ is computed sufficiently accurate)

A proof for the convergence of the gradient descent algorithm is in Appendix TODO.

Termination/convergence conditions

Typical convergence conditions:

- Relative decrease in the cost functional is sufficiently small:

$$J(u_k) - J(u_{k+1}) < \mathtt{tol} J(u_k).$$

- Relative change in iterates is sufficiently small:

$$|u_{k-1} - u_k| < \mathtt{tol}|u_k|.$$

- The gradient is sufficiently small:

$$|\nabla J(u_k)| < \mathtt{tol}.$$

In the first two conditions, we typically use `tol` $\in [10^{-6}, 10^{-3}]$.

Often not all three conditions are checked simultaneously, but only one or two are used.

Note: `tol` in the last condition is an absolute tolerance, while `tol` in the first two conditions is a relative tolerance.
A reasonable magnitude for the absolute tolerance might be difficult to estimate.

Pseudo code of the resulting gradient descent algorithm

```
Choose an initial guess $u_0$
Choose an initial step size $\beta$
Compute $J_0 = J(u_0)$.
for $i$ = 1: max_iters
    Compute $g_0 = \nabla J(u_0)$.
    Set $J_1 = \infty$ and $\beta = 4\beta$.
    while $J_1 > J_0$
        Set $\beta = \beta/2$.
        Set $u_1 = u_0 - \beta g_0$.
        Compute $J_1 = J(u_1)$.
    if convergence conditions are satisfied
        Return $u_1$, $J_1$.
    Set $u_0 = u_1$
    Set $J_0 = J_1$
```

Improved step size selection

For a convex C^2-functional $J(\mathbf{u})$,
we can estimate the stepsize based on a quadratic approximation:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k), \qquad \beta_k > 0,$$

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3),$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla J(\mathbf{u}_k) \rangle,$$
$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u}_k + \theta \nabla J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Note: G is positive because we update in a descent direction.
H is positive because J is convex.

Improved step size selection

For a convex C^2-functional $J(\mathbf{u})$,
we can estimate the stepsize based on a quadratic approximation:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k), \qquad \beta_k > 0,$$

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3),$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla J(\mathbf{u}_k) \rangle,$$
$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u}_k + \theta \nabla J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Note: G is positive because we update in a descent direction.
H is positive because J is convex.
Set derivative of the quadratic approximation to zero:

$$-G + H\beta_{k,\mathrm{opt}} = 0, \qquad \beta_{k,\mathrm{opt}} = \frac{G}{H}.$$

Improved step size selection

For a convex C^2-functional $J(\mathbf{u})$,
we can estimate the stepsize based on a quadratic approximation:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k), \qquad\qquad \beta_k > 0,$$

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3),$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla J(\mathbf{u}_k) \rangle,$$
$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u}_k + \theta \nabla J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Note: G is positive because we update in a descent direction.
H is positive because J is convex.
Set derivative of the quadratic approximation to zero:

$$-G + H\beta_{k,\mathrm{opt}} = 0, \qquad\qquad \beta_{k,\mathrm{opt}} = \frac{G}{H}.$$

When J is quadratic, $J(\mathbf{u}_k + \beta_{k,\mathrm{opt}} \nabla J(\mathbf{u}_k)) = J(\mathbf{u}_k) - \beta_{k,\mathrm{opt}} G + \frac{H}{2}\beta_{k,\mathrm{opt}}^2 = J(\mathbf{u}_k) - \frac{G^2}{2H}$
When J is not quadratic, there are higher order terms and we cannot guarantee that $J(\mathbf{u}_k + \beta_{k,\mathrm{opt}} \nabla J(\mathbf{u}_k)) \leq J(\mathbf{u}_k)$. We still need to do a line search (starting from $\beta_{k,\mathrm{opt}}$).

Computation of H (example)

Consider the optimization problem

$$\min_{u\in U_{\mathrm{ad}}} \tfrac{1}{2}\mathbf{x}^\top\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^\top\mathbf{R}\mathbf{u}$$

with $\mathbf{Q}=\mathbf{Q}^\top$, $\mathbf{R}=\mathbf{R}^\top$, $\mathbf{u}\in U_{\mathrm{ad}}\subset\mathbb{R}^M$, and $\mathbf{x}\in\mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u}=\mathbf{0}.$$

As explained before, we can compute the gradient $\nabla J(\mathbf{u}_k)$ at the current iterate $\mathbf{u}_k$.
We want to compute

$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u_k}+\theta\nabla J(\mathbf{u}_k))\right]_{\theta=0}.$$

Observe that

$$\begin{aligned} J(\mathbf{u}_k+\theta\nabla J) &= \tfrac{1}{2}(\mathbf{x}_k+\theta\mathbf{x}_k^\nabla)^\top\mathbf{Q}(\mathbf{x}_k+\theta\mathbf{x}_k^\nabla) + \tfrac{1}{2}(\mathbf{u}_k+\theta\nabla J(\mathbf{u}_k))^\top\mathbf{R}(\mathbf{u}_k+\theta\nabla J(\mathbf{u}_k)) \\ &= \tfrac{1}{2}\mathbf{x}_k^\top\mathbf{Q}\mathbf{x}_k + \tfrac{1}{2}\mathbf{u}_k^\top\mathbf{R}\mathbf{u}_k + \theta\left(\mathbf{x}_k^\top\mathbf{Q}\mathbf{x}_k^\nabla + \mathbf{u}_k^\top\mathbf{R}\nabla J(\mathbf{u}_k)\right) \\ &\qquad \theta^2\left(\tfrac{1}{2}\left(\mathbf{x}_k^\nabla\right)^\top\mathbf{Q}\mathbf{x}_k^\nabla + \tfrac{1}{2}(\nabla J(\mathbf{u}_k))^\top\mathbf{R}\nabla J(\mathbf{u}_k)\right), \end{aligned}$$

where $\mathbf{x}_k=\mathbf{A}^{-1}\mathbf{B}\mathbf{u}_k$ and $\mathbf{x}_k^\nabla=\mathbf{A}^{-1}\mathbf{B}\nabla J(\mathbf{u}_k)$. Differentiating twice to θ, we obtain

$$H = \left(\mathbf{x}_k^\nabla\right)^\top\mathbf{Q}\mathbf{x}_k^\nabla + (\nabla J(\mathbf{u}_k))^\top\mathbf{R}\nabla J(\mathbf{u}_k).$$

Pseudo code of the gradient descent algorithm with improved step size

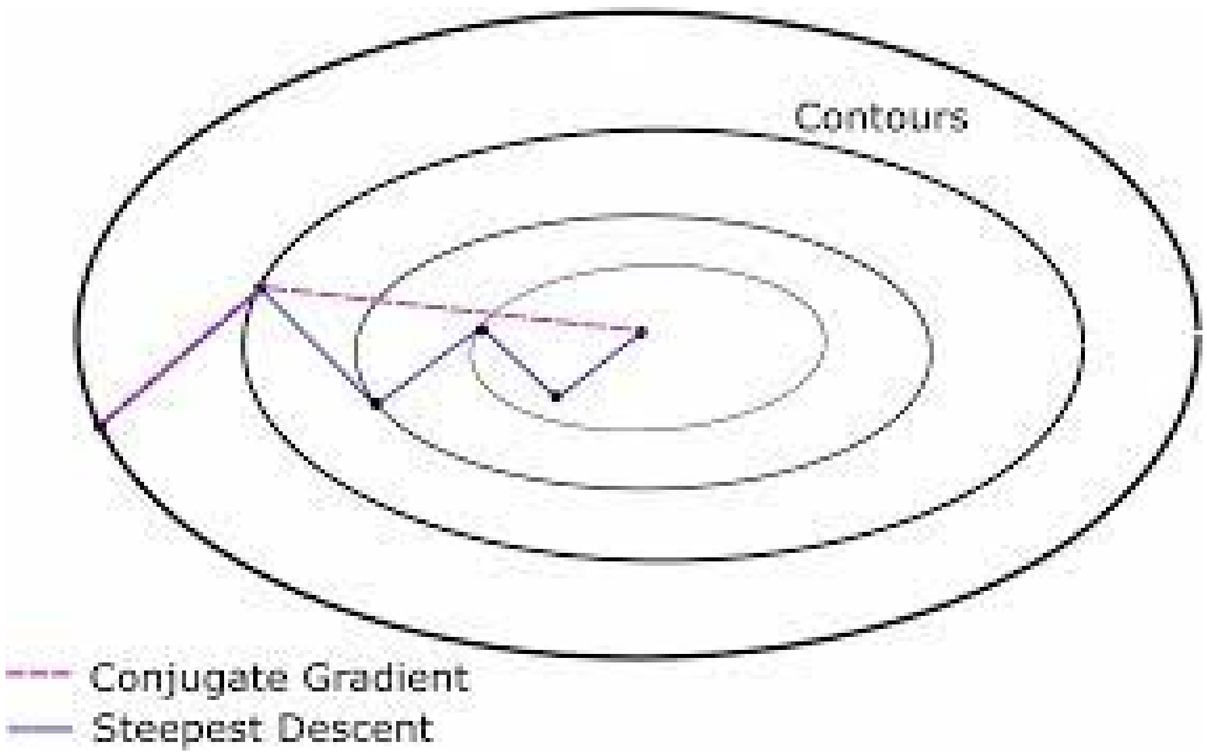
For the Linear-Quadratic (LQ) problem, the following algorithm can be used.

```
Choose an initial guess $u_0$
Choose an initial step size $\beta$
Compute $J_0 = J(u_0)$.
for $i$ = 1: max_iters
    Compute $g_0 = \nabla J(u_0)$.
    Compute $H = \frac{\mathrm{d}^2}{\mathrm{d}\theta^2} \nabla J(u_0 + \theta g_0)$.
    Set $\beta_{\mathrm{opt}} = G/H$.
    Set $u_1 = u_0 - \beta_{\mathrm{opt}} g_0$.
    Compute $J_1 = J(u_1)$.
    if convergence conditions are satisfied
        Return $u_1$, $J_1$.
    Set $u_0 = u_1$
    Set $J_0 = J_1$
```

Other algorithms

There are many more gradient-based algorithms.
Gradient-descent/steepest descent is the simplest one.
For quadratic problems, the Conjugate Gradient (CG) method is the best method.
When optimizing $u \in \mathbb{R}^M$, it converges in at most M iterations to the minimizer.
For nonquadratic problems, other algorithms can be more effective.

see e.g. Ascher, The chaotic nature of faster gradient descent methods



Chair
DYNAMICS, CONTROL
AND NUMERICS
FAU

Department of
DATA SCIENCE

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
NATURWISSENSCHAFTLICHE
FAKULTÄT

1.C Gradient computation in optimal control problems

Optimal control

We now consider optimization over time dependent functions $\mathbf{u}(t) \in L^2([0,T],\mathbb{R}^M)$.

The prototypical problem:

$$\min_{\mathbf{u}\in L^2([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

subject to the dynamics

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

Optimal control

We now consider optimization over time dependent functions $\mathbf{u}(t) \in L^2([0,T],\mathbb{R}^M)$.

The prototypical problem:

$$\min_{\mathbf{u}\in L^2([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t)-\mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t)-\mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

subject to the dynamics

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

Assumption: $\mathbf{Q}$ and $\mathbf{R}$ are symmetric.
Assumption: the matrix $\mathbf{Q}$ is positive semi-definite and the matrix $\mathbf{R}$ is positive definite, i.e.

$$\mathbf{x}^\top\mathbf{Q}\mathbf{x} \geq 0, \qquad \mathbf{u}^\top\mathbf{R}\mathbf{u} > 0,$$

for all $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{0} \neq \mathbf{u} \in \mathbb{R}^M$.

Theorem

Under this assumption, the minimizer $\mathbf{u}^*(t)$ of $J(\mathbf{u})$ exists and is unique.

This follows because J is α-convex, with $\alpha = \lambda_{\text{min}}(\mathbf{R}) > 0$.

What is the gradient now?

We are now optimizing over the infinite-dimensional space $L^2([0,T],\mathbb{R}^M)$.
We can therefore no longer use that

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle = \frac{\mathrm{d}J}{\mathrm{d}\mathbf{u}}(\mathbf{u})\tilde{\mathbf{u}},$$

and simply use the chain rule to find $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{u}}(\mathbf{u})$.
(It is not so clear what $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{u}}(\mathbf{u})$ now is supposed to mean!)

We therefore start from the basic definition:

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle = \lim_{h\to 0} \frac{J(\mathbf{u}+h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h}.$$

Note: this is the Gateaux derivative of J at point $\mathbf{u}$ in the direction $\tilde{\mathbf{u}}$.

The directional derivative (1/2)

$$J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

$$J(\mathbf{u}+h\tilde{\mathbf{u}}) = \frac{1}{2}\int_0^T (\mathbf{x}^h(t)-\mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}^h(t)-\mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t)+h\tilde{\mathbf{u}}(t))^\top \mathbf{R}(\mathbf{u}(t)+h\tilde{\mathbf{u}}(t))\,\mathrm{d}t$$

$$\mathbf{E}\dot{\mathbf{x}}^h(t) = \mathbf{A}\mathbf{x}^h(t) + \mathbf{B}(\mathbf{u}(t) + h\tilde{\mathbf{u}}(t)), \qquad \mathbf{x}^h(0) = \mathbf{x}_{\text{init}}.$$

The directional derivative (1/2)

$$J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

$$J(\mathbf{u}+h\tilde{\mathbf{u}}) = \frac{1}{2}\int_0^T (\mathbf{x}^h(t) - \mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}^h(t) - \mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t)+h\tilde{\mathbf{u}}(t))^\top \mathbf{R}(\mathbf{u}(t)+h\tilde{\mathbf{u}}(t))\,\mathrm{d}t$$

$$\mathbf{E}\dot{\mathbf{x}}^h(t) = \mathbf{A}\mathbf{x}^h(t) + \mathbf{B}(\mathbf{u}(t) + h\tilde{\mathbf{u}}(t)), \qquad \mathbf{x}^h(0) = \mathbf{x}_{\text{init}}.$$

Write:

$$\mathbf{x}^h(t) = \mathbf{x}(t) + h\tilde{\mathbf{x}}(t), \qquad \tilde{\mathbf{x}}(t) = \frac{\mathbf{x}^h(t) - \mathbf{x}(t)}{h},$$

Then:

$$\begin{aligned}\mathbf{E}\dot{\tilde{\mathbf{x}}}(t) = \mathbf{E}\frac{\dot{\mathbf{x}}^h(t) - \dot{\mathbf{x}}(t)}{h} &= \frac{1}{h}\Big(\mathbf{A}\mathbf{x}^h(t) + \mathbf{B}(\mathbf{u}(t) + h\tilde{\mathbf{u}}(t)) - \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\Big) \\ &= \mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t), \qquad \tilde{\mathbf{x}}(0) = \mathbf{0}.\end{aligned}$$

The directional derivative (2/2)

$$J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$
$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

$$\begin{aligned} J(\mathbf{u}+h\tilde{\mathbf{u}}) = \frac{1}{2}\int_0^T &(\mathbf{x}(t) + h\tilde{\mathbf{x}}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t) + h\tilde{\mathbf{x}}(t) - \mathbf{x}_d(t))\,\mathrm{d}t \\ &+ \frac{1}{2}\int_0^T (\mathbf{u}(t) + h\tilde{\mathbf{u}}(t))^\top \mathbf{R}(\mathbf{u}(t) + h\tilde{\mathbf{u}}(t))\,\mathrm{d}t, \end{aligned}$$
$$\mathbf{E}\dot{\tilde{\mathbf{x}}}(t) = \mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t), \qquad \tilde{\mathbf{x}}(0) = \mathbf{0}.$$

It is now easy to verify that

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}}\rangle = \lim_{h\to 0}\frac{J(\mathbf{u}+h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h} = \int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}\tilde{\mathbf{x}}(t)\,\mathrm{d}t + \int_0^T (\mathbf{u}(t))^\top \mathbf{R}\tilde{\mathbf{u}}(t)\,\mathrm{d}t.$$

But what is the gradient now?

We have found a way to compute the directional derivative:

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle = \lim_{h \to 0} \frac{J(\mathbf{u} + h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h} = \int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}\tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\mathbf{u}(t))^\top \mathbf{R}\tilde{\mathbf{u}}(t)\, \mathrm{d}t.$$

$$\mathbf{E}\dot{\tilde{\mathbf{x}}}(t) = \mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t), \qquad \tilde{\mathbf{x}}(0) = \mathbf{0}.$$

But what is the gradient now?

We have found a way to compute the directional derivative:

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle = \lim_{h \to 0} \frac{J(\mathbf{u} + h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h} = \int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}\tilde{\mathbf{x}}(t)\,\mathrm{d}t + \int_0^T (\mathbf{u}(t))^\top \mathbf{R}\tilde{\mathbf{u}}(t)\,\mathrm{d}t.$$

$$\mathbf{E}\dot{\tilde{\mathbf{x}}}(t) = \mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t), \qquad \tilde{\mathbf{x}}(0) = \mathbf{0}.$$

But how do we find the gradient now?

Not so obvious:
In a finite dimensional space,
we could choose a basis for the space of perturbations $\tilde{\mathbf{u}}(t)$
and evaluate the directional gradient for all basis vectors.

However, in an infinite-dimensional space, this is not possible.
(we never finish evaluating the directional gradient for all basis functions)

The way out: the adjoint state

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle = \lim_{h\to 0} \frac{J(\mathbf{u}+h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h} = \int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}\tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\mathbf{u}(t))^\top \mathbf{R}\tilde{\mathbf{u}}(t)\, \mathrm{d}t.$$

$$\mathbf{E}\dot{\tilde{\mathbf{x}}}(t) = \mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t), \qquad \tilde{\mathbf{x}}(0) = \mathbf{0}.$$

We define the adjoint state $\varphi(t)$ as the solution of

$$-\mathbf{E}^\top \dot{\varphi}(t) = \mathbf{A}^\top \varphi(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t)), \qquad \varphi(T) = \mathbf{0}.$$

Question 1

We have that:

$$\mathbf{E}\dot{\tilde{\mathbf{x}}}(t) = \mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t), \qquad \tilde{\mathbf{x}}(0) = \mathbf{0}.$$

$$-\mathbf{E}^\top \dot{\boldsymbol{\varphi}}(t) = \mathbf{A}^\top \boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t)), \qquad \boldsymbol{\varphi}(T) = \mathbf{0}.$$

What is

$$\int_0^T \frac{d}{dt}\left((\boldsymbol{\varphi}(t))^\top \mathbf{E}\tilde{\mathbf{x}}(t)\right)\, \mathrm{d}t?$$

A) $(\boldsymbol{\varphi}(0))^\top \mathbf{E}\tilde{\mathbf{x}}(0) - (\boldsymbol{\varphi}(T))^\top \mathbf{E}\tilde{\mathbf{x}}(T)$

B) $(\boldsymbol{\varphi}(T))^\top \mathbf{E}\tilde{\mathbf{x}}(T) - (\boldsymbol{\varphi}(0))^\top \mathbf{E}\tilde{\mathbf{x}}(0)$

C) $(\boldsymbol{\varphi}(T))^\top \mathbf{E}\tilde{\mathbf{x}}(T)$

D) 0

E) None of the above.

Question 2

We have that:

$$\mathbf{E}\dot{\tilde{\mathbf{x}}}(t) = \mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t), \qquad \tilde{\mathbf{x}}(0) = \mathbf{0}.$$
$$-\mathbf{E}^\top\dot{\boldsymbol{\varphi}}(t) = \mathbf{A}^\top\boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t)), \qquad \boldsymbol{\varphi}(T) = \mathbf{0}.$$

What is

$$\int_0^T \frac{d}{dt}\Big((\boldsymbol{\varphi}(t))^\top\mathbf{E}\tilde{\mathbf{x}}(t)\Big)\ \mathrm{d}t = \int_0^T \Big(\mathbf{E}^\top\dot{\boldsymbol{\varphi}}(t)\Big)^\top\tilde{\mathbf{x}}(t)\ \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top\mathbf{E}\dot{\tilde{\mathbf{x}}}(t)\ \mathrm{d}t?$$

A) $\int_0^T \left(\mathbf{A}^\top\boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\right)^\top \tilde{\mathbf{x}}(t)\ \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \left(\mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t)\right)\ \mathrm{d}t$

B) $-\int_0^T \left(\mathbf{A}^\top\boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\right)^\top \tilde{\mathbf{x}}(t)\ \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \left(\mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t)\right)\ \mathrm{d}t$

C) $\int_0^T \left(\mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\right)^\top \tilde{\mathbf{x}}(t)\ \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \left(\mathbf{B}\tilde{\mathbf{u}}(t)\right)\ \mathrm{d}t$

D) $-\int_0^T \left(\mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\right)^\top \tilde{\mathbf{x}}(t)\ \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \left(\mathbf{B}\tilde{\mathbf{u}}(t)\right)\ \mathrm{d}t$

Answer question 2

$$\int_0^T \frac{d}{dt}\left((\boldsymbol{\varphi}(t))^\top \mathbf{E}\tilde{\mathbf{x}}(t)\right)\, \mathrm{d}t = \int_0^T \left(\mathbf{E}^\top \dot{\boldsymbol{\varphi}}(t)\right)^\top \tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \mathbf{E}\dot{\tilde{\mathbf{x}}}(t)\, \mathrm{d}t$$
$$= -\int_0^T \left(\mathbf{A}^\top \boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t))\right)^\top \tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \left(\mathbf{A}\tilde{\mathbf{x}}(t) + \mathbf{B}\tilde{\mathbf{u}}(t)\right)\, \mathrm{d}t$$
$$= -\int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}\tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \mathbf{B}\tilde{\mathbf{u}}(t)\, \mathrm{d}t$$

The gradient

Expression for the directional derivative:

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle = \lim_{h \to 0} \frac{J(\mathbf{u} + h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h} = \int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q} \tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\mathbf{u}(t))^\top \mathbf{R} \tilde{\mathbf{u}}(t)\, \mathrm{d}t.$$

Combining the answers from question 1 and 2:

$$-\int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q} \tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \mathbf{B} \tilde{\mathbf{u}}(t)\, \mathrm{d}t = 0.$$

The gradient

Expression for the directional derivative:

$$\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle = \lim_{h \to 0} \frac{J(\mathbf{u} + h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h} = \int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}\tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\mathbf{u}(t))^\top \mathbf{R}\tilde{\mathbf{u}}(t)\, \mathrm{d}t.$$

Combining the answers from question 1 and 2:

$$-\int_0^T (\mathbf{x}(t) - \mathbf{x}_d(t))^\top \mathbf{Q}\tilde{\mathbf{x}}(t)\, \mathrm{d}t + \int_0^T (\boldsymbol{\varphi}(t))^\top \mathbf{B}\tilde{\mathbf{u}}(t)\, \mathrm{d}t = 0.$$

Therefore also:

$$\begin{aligned}\langle \nabla J(\mathbf{u}), \tilde{\mathbf{u}} \rangle &= \lim_{h \to 0} \frac{J(\mathbf{u} + h\tilde{\mathbf{u}}) - J(\mathbf{u})}{h} = \int_0^T (\boldsymbol{\varphi}(t))^\top \mathbf{B}\tilde{\mathbf{u}}(t)\, \mathrm{d}t + \int_0^T (\mathbf{u}(t))^\top \mathbf{R}\tilde{\mathbf{u}}(t)\, \mathrm{d}t \\ &= \int_0^T \left(\mathbf{B}^\top \boldsymbol{\varphi}(t) + \mathbf{R}\mathbf{u}(t)\right)^\top \tilde{\mathbf{u}}(t)\, \mathrm{d}t = \langle \mathbf{B}^\top \boldsymbol{\varphi} + \mathbf{R}\mathbf{u}, \tilde{\mathbf{u}} \rangle_{L^2}\end{aligned}$$

Resulting gradient (w.r.t. the standard L^2-innerproduct):

$$\left(\nabla J(\mathbf{u})\right)(t) = \mathbf{B}^\top \boldsymbol{\varphi}(t) + \mathbf{R}\mathbf{u}(t).$$

An algorithm for the computation of the gradient

Computation of $\nabla J(\mathbf{u})$ (gradient in the point $\mathbf{u}(t)$)

- Compute the solution $\mathbf{x}(t)$ (the state) of

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

- Compute the solution of $\boldsymbol{\psi}(t) = \boldsymbol{\varphi}(T-t)$ of

$$\mathbf{E}^\top\dot{\boldsymbol{\psi}}(t) = \mathbf{A}^\top\boldsymbol{\psi}(t) + \mathbf{Q}(\mathbf{x}(T-t) - \mathbf{x}_d(T-t)), \qquad \boldsymbol{\psi}(0) = \mathbf{0}.$$

- The gradient is now given by

$$\left(\nabla J(\mathbf{u})\right)(t) = \mathbf{B}^\top\boldsymbol{\varphi}(t) + \mathbf{R}\mathbf{u}(t) = \mathbf{B}^\top\boldsymbol{\psi}(T-t) + \mathbf{R}\mathbf{u}(t).$$

Step size selection can be done in the same way as explained the previous lecture.

Remaining problem: we still need to discretize time!

Chair
DYNAMICS, CONTROL
AND NUMERICS
FAU

DDS Department of DATA SCIENCE

FAU FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
NATURWISSENSCHAFTLICHE
FAKULTÄT

1.D Time discretization of optimal control problems

Two main approaches

- Discretize-then-optimize
  Approach: First discretize the cost functional and the forward dynamics,
  then compute the gradient for the discretized problem,
  and use a gradient-based optimization algorithm
  (e.g., conjugate gradients or the one from the previous lecture).

- Optimize-then-discretize
  Approach: first find the equation for the adjoint state,
  then discretize the cost functional, the forward dynamics, and the adjoint equation,
  use the discretized adjoint equation to compute the gradient,
  which can again be used in a gradient-based optimization algorithm.

Two main approaches

- Discretize-then-optimize
  Approach: First discretize the cost functional and the forward dynamics,
  then compute the gradient for the discretized problem,
  and use a gradient-based optimization algorithm
  (e.g., conjugate gradients or the one from the previous lecture).

- Optimize-then-discretize
  Approach: first find the equation for the adjoint state,
  then discretize the cost functional, the forward dynamics, and the adjoint equation,
  use the discretized adjoint equation to compute the gradient,
  which can again be used in a gradient-based optimization algorithm.

Discretize-then-optimize leads to the most accurate solutions of the discretized problem.

In certain cases, solutions of the discretized optimal control contain spurious artefacts,
that can be avoided by certain the optimize-then-discretize approaches.
see e.g. Dogin, Morin, Nochetto, Verani, discrete gradient flows for shape optimization and applications, 2007

Ervedoza, Zuazua, Numerical Approximation of Exact Controls for Waves, 2013

In some cases, the result of both approaches coincide.
In this case, we say that the approximation of the gradient is *discretely consistent*.

These ideas will now be demonstrated for a particular example.

Discretization of the forward dynamics

We want to discretize the problem:

$$\min_{\mathbf{u}\in L^2([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t)-\mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t)-\mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

subject to the dynamics

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

We consider a uniform grid $t_k = (k-1)\Delta t$ ($k = 1, 2, \dots, N_T$), so $\Delta t = T/(N_T - 1)$.
We denote $\mathbf{x}_k \approx \mathbf{x}(t_k)$ and $\mathbf{u}_k = \mathbf{u}(t_k)$.

We discretize the dynamics with the Crank-Nicolson scheme:

$$\mathbf{E}\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta t} = \tfrac{1}{2}\left(\mathbf{A}\mathbf{x}_k + \mathbf{B}u_k\right) + \tfrac{1}{2}\left(\mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1}\right) = \mathbf{A}\frac{\mathbf{x}_k + \mathbf{x}_{k-1}}{2} + \mathbf{B}\frac{\mathbf{u}_k + \mathbf{u}_{k-1}}{2}.$$

Starting from the given initial condition $\mathbf{x}_1 = \mathbf{x}_{\text{init}}$, we compute $\mathbf{x}_k$ from $\mathbf{x}_{k-1}$ by solving

$$\left(\mathbf{E} - \tfrac{\Delta t}{2}\mathbf{A}\right)\mathbf{x}_k = \left(\mathbf{E} + \tfrac{\Delta t}{2}\mathbf{A}\right)\mathbf{x}_{k-1} + \Delta t\mathbf{B}\frac{\mathbf{u}_k + \mathbf{u}_{k-1}}{2}, \qquad k = 2, 3, \dots, N_T.$$

Discretization of the forward dynamics

We want to discretize the problem:

$$\min_{\mathbf{u}\in L^2([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t)-\mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t)-\mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

subject to the dynamics

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

Observe: We only use

- N_T state variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_T}$,
- $N_T - 1$ control variables

$$\mathbf{u}_{k-1/2} = \frac{\mathbf{u}_k + \mathbf{u}_{k-1}}{2}, \qquad k = 2,3,\ldots N_T$$

In these new variables, the discretization of the forward dynamics becomes:

$$\left(\mathbf{E} - \tfrac{\Delta t}{2}\mathbf{A}\right)\mathbf{x}_k = \left(\mathbf{E} + \tfrac{\Delta t}{2}\mathbf{A}\right)\mathbf{x}_{k-1} + \Delta t\mathbf{B}\mathbf{u}_{k-1/2}, \qquad \mathbf{x}_1 = \mathbf{x}_{\text{init}}$$

Discretization of the cost functional

We want to discretize the problem:

$$\min_{\mathbf{u}\in L^2([0,T],\mathbb{R}^q)} J(\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}(t)-\mathbf{x}_d(t))^\top \mathbf{Q}(\mathbf{x}(t)-\mathbf{x}_d(t))\,\mathrm{d}t + \frac{1}{2}\int_0^T (\mathbf{u}(t))^\top \mathbf{R}\mathbf{u}(t)\,\mathrm{d}t,$$

subject to the dynamics

$$\mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}.$$

We consider a uniform grid $t_k = (k-1)\Delta t$ ($k = 1, 2, \ldots, N_T$), so $\Delta t = T/(N_T - 1)$.
We denote $\mathbf{x}_k \approx \mathbf{x}(t_k)$ and $\mathbf{u}_k = \mathbf{u}(t_k)$.

We discretize the first part with the trapezoid rule and the second part with the midpoint rule.

$$J = \frac{\Delta t}{4}\sum_{k=2}^{N_T}\Big[(\mathbf{x}_{k-1}-\mathbf{x}_d(t_{k-1}))^\top \mathbf{Q}\,(\mathbf{x}_{k-1}-\mathbf{x}_d(t_{k-1})) + (\mathbf{x}_k-\mathbf{x}_d(t_k))^\top \mathbf{Q}\,(\mathbf{x}_k-\mathbf{x}_d(t_k))\Big]$$

$$+\frac{\Delta t}{2}\sum_{k=2}^{N_T}\mathbf{u}_{k-1/2}^\top \mathbf{R}\mathbf{u}_{k-1/2}$$

Discretization of the adjoint state (optimize-then-discretize)

In the continuous time setting, we could compute the gradient from the adjoint state:

$$-\mathbf{E}^\top \dot{\boldsymbol{\varphi}}(t) = \mathbf{A}^\top \boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t)), \qquad \boldsymbol{\varphi}(T) = \mathbf{0}.$$

$$(\nabla J(\mathbf{u}))\,(t) = \mathbf{B}^\top \boldsymbol{\varphi}(t) + \mathbf{R}\mathbf{u}(t).$$

Discretization of the adjoint state (optimize-then-discretize)

In the continuous time setting, we could compute the gradient from the adjoint state:

$$-\mathbf{E}^\top \dot{\boldsymbol{\varphi}}(t) = \mathbf{A}^\top \boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t)), \qquad \boldsymbol{\varphi}(T) = \mathbf{0}.$$

$$\left(\nabla J(\mathbf{u})\right)(t) = \mathbf{B}^\top \boldsymbol{\varphi}(t) + \mathbf{R}\mathbf{u}(t).$$

We can now also use the Crank-Nicolson scheme to discretize the adjoint equation. We therefore introduce the adjoint variables in the grid points

$$\boldsymbol{\varphi}_k, \qquad k = 1, 2, \ldots, N_T.$$

We then integrate *backward in time* starting from the final condition $\boldsymbol{\varphi}_{N_T} = \mathbf{0}$

$$-\mathbf{E}^\top \frac{\boldsymbol{\varphi}_k - \boldsymbol{\varphi}_{k-1}}{\Delta t} = \mathbf{A}^\top \frac{\boldsymbol{\varphi}_k + \boldsymbol{\varphi}_{k-1}}{2} + \mathbf{Q}\frac{\mathbf{x}_k + \mathbf{x}_{k-1} - \mathbf{x}_d(t_k) - \mathbf{x}_d(t_{k-1})}{2},$$

$$\left(\mathbf{E}^\top - \tfrac{\Delta t}{2}\mathbf{A}^\top\right)\boldsymbol{\varphi}_{k-1} = \left(\mathbf{E}^\top + \tfrac{\Delta t}{2}\mathbf{A}^\top\right)\boldsymbol{\varphi}_k + \tfrac{\Delta t}{2}\mathbf{Q}\left(\mathbf{x}_k + \mathbf{x}_{k-1} - \mathbf{x}_d(t_k) - \mathbf{x}_d(t_{k-1})\right),$$

which is an equation from which $\boldsymbol{\varphi}_{k-1}$ can be solved from $\boldsymbol{\varphi}_k$.

Discretization of the adjoint state (optimize-then-discretize)

In the continuous time setting, we could compute the gradient from the adjoint state:

$$-\mathbf{E}^\top \dot{\boldsymbol{\varphi}}(t) = \mathbf{A}^\top \boldsymbol{\varphi}(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t)), \qquad \boldsymbol{\varphi}(T) = \mathbf{0}.$$

$$(\nabla J(\mathbf{u}))(t) = \mathbf{B}^\top \boldsymbol{\varphi}(t) + \mathbf{R}\mathbf{u}(t).$$

We can now also use the Crank-Nicolson scheme to discretize the adjoint equation. We therefore introduce the adjoint variables in the grid points

$$\boldsymbol{\varphi}_k, \qquad k = 1, 2, \dots, N_T.$$

We then integrate *backward in time* starting from the final condition $\boldsymbol{\varphi}_{N_T} = \mathbf{0}$

$$-\mathbf{E}^\top \frac{\boldsymbol{\varphi}_k - \boldsymbol{\varphi}_{k-1}}{\Delta t} = \mathbf{A}^\top \frac{\boldsymbol{\varphi}_k + \boldsymbol{\varphi}_{k-1}}{2} + \mathbf{Q}\frac{\mathbf{x}_k + \mathbf{x}_{k-1} - \mathbf{x}_d(t_k) - \mathbf{x}_d(t_{k-1})}{2},$$

$$\left(\mathbf{E}^\top - \tfrac{\Delta t}{2}\mathbf{A}^\top\right)\boldsymbol{\varphi}_{k-1} = \left(\mathbf{E}^\top + \tfrac{\Delta t}{2}\mathbf{A}^\top\right)\boldsymbol{\varphi}_k + \tfrac{\Delta t}{2}\mathbf{Q}\left(\mathbf{x}_k + \mathbf{x}_{k-1} - \mathbf{x}_d(t_k) - \mathbf{x}_d(t_{k-1})\right),$$

which is an equation from which $\boldsymbol{\varphi}_{k-1}$ can be solved from $\boldsymbol{\varphi}_k$.
Note gradient is (just as the control $\mathbf{u}_{k-1/2}$) defined in the intermediate grid points

$$(\nabla J)_{k-1/2} = \mathbf{B}^\top \frac{\boldsymbol{\varphi}_k + \boldsymbol{\varphi}_{k-1}}{2} + \mathbf{R}\mathbf{u}_{k-1/2}.$$

Discretely consistent gradient (discretize-then-optimize)

In the continuous time setting, we could compute the gradient from the adjoint state:

$$-\mathbf{E}^\top \dot{\varphi}(t) = \mathbf{A}^\top \varphi(t) + \mathbf{Q}(\mathbf{x}(t) - \mathbf{x}_d(t)), \qquad \varphi(T) = \mathbf{0}.$$

$$(\nabla J(\mathbf{u}))(t) = \mathbf{B}^\top \varphi(t) + \mathbf{R}\mathbf{u}(t).$$

We use the same discretization for the forward dynamics and cost functional as before. We now use adjoint variables defined in the intermediate points (just as the controls):

$$\varphi_{k-1/2}, \qquad k = 2, 3, \ldots N_T.$$

We then form the (discretized) Lagrangian

$$\mathcal{L} = \frac{\Delta t}{4} \sum_{k=2}^{N_T} \left[(\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1}))^\top \mathbf{Q} (\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1})) + (\mathbf{x}_k - \mathbf{x}_d(t_k))^\top \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_d(t_k)) \right]$$

$$+ \frac{\Delta t}{2} \sum_{k=2}^{N_T} \mathbf{u}_{k-1/2}^\top \mathbf{R} \mathbf{u}_{k-1/2} + \Delta t \sum_{k=2}^{N_T} \varphi_{k-1/2}^\top \left(\mathbf{A} \frac{\mathbf{x}_k + \mathbf{x}_{k-1}}{2} + \mathbf{B}\mathbf{u}_{k-1/2} - \mathbf{E} \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta t} \right)$$

$$+ \varphi_0^\top (\mathbf{x}_1 - \mathbf{x}_{\text{init}}).$$

Note: the adjoint states are introduced as Lagrange multipliers.
Note: we have also introduced φ_0 as Lagrange multiplier for the initial condition, but we will see that we do not need φ_0 to compute the gradient.

Equations for the adjoint state (discretize-then-optimize)

We have defined the (discretized) Lagrangian

$$\mathcal{L} = \frac{\Delta t}{4} \sum_{k=2}^{N_T} \left[(\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1}))^\top \mathbf{Q} \left(\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1}) \right) + (\mathbf{x}_k - \mathbf{x}_d(t_k))^\top \mathbf{Q} \left(\mathbf{x}_k - \mathbf{x}_d(t_k) \right) \right]$$

$$+ \frac{\Delta t}{2} \sum_{k=2}^{N_T} \mathbf{u}_{k-1/2}^\top \mathbf{R} \mathbf{u}_{k-1/2} + \Delta t \sum_{k=2}^{N_T} \boldsymbol{\varphi}_{k-1/2}^\top \left(\mathbf{A} \frac{\mathbf{x}_k + \mathbf{x}_{k-1}}{2} + \mathbf{B} \mathbf{u}_{k-1/2} - \mathbf{E} \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta t} \right)$$

$$+ \boldsymbol{\varphi}_0^\top (\mathbf{x}_1 - \mathbf{x}_{\text{init}}).$$

Equations for the adjoint state (discretize-then-optimize)

We have defined the (discretized) Lagrangian

$$\mathcal{L} = \frac{\Delta t}{4} \sum_{k=2}^{N_T} \left[\left(\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1})\right)^\top \mathbf{Q} \left(\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1})\right) + \left(\mathbf{x}_k - \mathbf{x}_d(t_k)\right)^\top \mathbf{Q} \left(\mathbf{x}_k - \mathbf{x}_d(t_k)\right) \right]$$
$$+ \frac{\Delta t}{2} \sum_{k=2}^{N_T} \mathbf{u}_{k-1/2}^\top \mathbf{R} \mathbf{u}_{k-1/2} + \Delta t \sum_{k=2}^{N_T} \boldsymbol{\varphi}_{k-1/2}^\top \left(\mathbf{A} \frac{\mathbf{x}_k + \mathbf{x}_{k-1}}{2} + \mathbf{B}\mathbf{u}_{k-1/2} - \mathbf{E} \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta t} \right)$$
$$+ \boldsymbol{\varphi}_0^\top (\mathbf{x}_1 - \mathbf{x}_{\text{init}}).$$

Setting the derivatives of $\mathcal{L}$ w.r.t. the adjoint variables $\boldsymbol{\varphi}_{k-1/2}$ gives the equations for the forward dynamics.

Equations for the adjoint state (discretize-then-optimize)

We have defined the (discretized) Lagrangian

$$\mathcal{L} = \frac{\Delta t}{4}\sum_{k=2}^{N_T}\left[\left(\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1})\right)^\top \mathbf{Q}\left(\mathbf{x}_{k-1} - \mathbf{x}_d(t_{k-1})\right) + \left(\mathbf{x}_k - \mathbf{x}_d(t_k)\right)^\top \mathbf{Q}\left(\mathbf{x}_k - \mathbf{x}_d(t_k)\right)\right]$$
$$+\frac{\Delta t}{2}\sum_{k=2}^{N_T}\mathbf{u}_{k-1/2}^\top \mathbf{R}\mathbf{u}_{k-1/2} + \Delta t\sum_{k=2}^{N_T}\boldsymbol{\varphi}_{k-1/2}^\top\left(\mathbf{A}\frac{\mathbf{x}_k+\mathbf{x}_{k-1}}{2} + \mathbf{B}\mathbf{u}_{k-1/2} - \mathbf{E}\frac{\mathbf{x}_k-\mathbf{x}_{k-1}}{\Delta t}\right)$$
$$+\boldsymbol{\varphi}_0^\top\left(\mathbf{x}_1 - \mathbf{x}_{\text{init}}\right).$$

Setting the derivatives of $\mathcal{L}$ w.r.t. the adjoint variables $\boldsymbol{\varphi}_{k-1/2}$ gives the equations for the forward dynamics.
Requiring that the derivatives of $\mathcal{L}$ w.r.t. the state variables $\mathbf{x}_k$ are zero gives the equations for the adjoint state:

$$0 = \frac{\partial\mathcal{L}}{\partial\mathbf{x}_{N_T}} = \frac{\Delta t}{2}\left(\mathbf{x}_{N_T} - \mathbf{x}_d(t_{N_T})\right)^\top\mathbf{Q} + \boldsymbol{\varphi}_{N_T-1/2}^\top\left(\tfrac{\Delta t}{2}\mathbf{A} - \mathbf{E}\right)$$

and, for $k = N_T-1, N_T-2, \dots, 2$ and for $k=1$

$$0 = \frac{\partial\mathcal{L}}{\partial\mathbf{x}_k} = \Delta t\left(\mathbf{x}_k - \mathbf{x}_d(t_k)\right)^\top\mathbf{Q} + \boldsymbol{\varphi}_{k-1/2}^\top\left(\tfrac{\Delta t}{2}\mathbf{A} - \mathbf{E}\right) + \boldsymbol{\varphi}_{k+1/2}^\top\left(\tfrac{\Delta t}{2}\mathbf{A} + \mathbf{E}\right),$$
$$0 = \frac{\partial\mathcal{L}}{\partial\mathbf{x}_1} = \frac{\Delta t}{2}\left(\mathbf{x}_1 - \mathbf{x}_d(t_1)\right)^\top\mathbf{Q} + \boldsymbol{\varphi}_{1+1/2}^\top\left(\tfrac{\Delta t}{2}\mathbf{A} + \mathbf{E}\right) + \boldsymbol{\varphi}_0^\top.$$

Equations for the adjoint state (discretize-then-optimize)

We have defined the (discretized) Lagrangian

$$\mathcal{L} = \frac{\Delta t}{4}\sum_{k=2}^{N_T}\Big[(\mathbf{x}_{k-1}-\mathbf{x}_d(t_{k-1}))^\top \mathbf{Q}\,(\mathbf{x}_{k-1}-\mathbf{x}_d(t_{k-1})) + (\mathbf{x}_k-\mathbf{x}_d(t_k))^\top \mathbf{Q}\,(\mathbf{x}_k-\mathbf{x}_d(t_k))\Big]$$
$$+\frac{\Delta t}{2}\sum_{k=2}^{N_T}\mathbf{u}_{k-1/2}^\top\mathbf{R}\mathbf{u}_{k-1/2} + \Delta t\sum_{k=2}^{N_T}\boldsymbol{\varphi}_{k-1/2}^\top\left(\mathbf{A}\frac{\mathbf{x}_k+\mathbf{x}_{k-1}}{2}+\mathbf{B}\mathbf{u}_{k-1/2}-\mathbf{E}\frac{\mathbf{x}_k-\mathbf{x}_{k-1}}{\Delta t}\right)$$
$$+\boldsymbol{\varphi}_0^\top(\mathbf{x}_1-\mathbf{x}_{\text{init}}).$$

Requiring that the derivatives of $\mathcal{L}$ w.r.t. the state variables $\mathbf{x}_k$ are zero gives the equations for the adjoint state.
We can now compute the adjoint states as follows: Start by solving $\boldsymbol{\varphi}_{N_T-1/2}$ from

$$\left(\mathbf{E}^\top-\frac{\Delta t}{2}\mathbf{A}^\top\right)\boldsymbol{\varphi}_{N_T-1/2} = \frac{\Delta t}{2}\mathbf{Q}(\mathbf{x}_{N_T}-\mathbf{x}_d(t_{N_T})),$$

and then iteratively compute $\boldsymbol{\varphi}_{k-1/2}$ from

$$\left(\mathbf{E}^\top-\frac{\Delta t}{2}\mathbf{A}^\top\right)\boldsymbol{\varphi}_{k-1/2} = \left(\mathbf{E}^\top+\frac{\Delta t}{2}\mathbf{A}^\top\right)\boldsymbol{\varphi}_{k+1/2} + \Delta t\mathbf{Q}(\mathbf{x}_k-\mathbf{x}_d(t_k)),$$

for $k = N_T-1, N_T-2, \ldots, 2$ using the previously obtained $\boldsymbol{\varphi}_{k+1/2}$.

Gradient computation (discretize-then-optimize)

We have defined the (discretized) Lagrangian

$$\mathcal{L} = \frac{\Delta t}{4}\sum_{k=2}^{N_T}\left[(\mathbf{x}_{k-1}-\mathbf{x}_d(t_{k-1}))^\top \mathbf{Q}\,(\mathbf{x}_{k-1}-\mathbf{x}_d(t_{k-1})) + (\mathbf{x}_k-\mathbf{x}_d(t_k))^\top \mathbf{Q}\,(\mathbf{x}_k-\mathbf{x}_d(t_k))\right]$$
$$+\frac{\Delta t}{2}\sum_{k=2}^{N_T}\mathbf{u}_{k-1/2}^\top \mathbf{R}\mathbf{u}_{k-1/2} + \Delta t\sum_{k=2}^{N_T}\boldsymbol{\varphi}_{k-1/2}^\top\left(\mathbf{A}\frac{\mathbf{x}_k+\mathbf{x}_{k-1}}{2} + \mathbf{B}\mathbf{u}_{k-1/2} - \mathbf{E}\frac{\mathbf{x}_k-\mathbf{x}_{k-1}}{\Delta t}\right)$$
$$+\boldsymbol{\varphi}_0^\top(\mathbf{x}_1-\mathbf{x}_{\text{init}}).$$

Taking the partial derivative of $\mathcal{L}$ w.r.t. $\mathbf{u}_{k-1/2}$ gives the (total) derivative

$$\left(\frac{\mathrm{d}J}{\mathrm{d}\mathbf{u}_{k-1/2}}\right)_{k-1/2} = \Delta t\boldsymbol{\varphi}_{k-1/2}^\top\mathbf{B} + \Delta t\mathbf{u}_{k-1/2}^\top\mathbf{R}.$$

The gradient then follows after defining an inner product

$$\langle\mathbf{u},\mathbf{v}\rangle = \Delta t\sum_{k=2}^{N_T}\mathbf{u}_{k-1/2}^\top\mathbf{v}_{k-1/2}, \qquad (\nabla J)_{k-1/2} = \mathbf{B}^\top\boldsymbol{\varphi}_{k-1/2} + \mathbf{R}\mathbf{u}_{k-1/2}.$$

Remark: The resulting equations have a similar structure as the equations for the optimize-then-discretize approach, but the schemes are different.

(this example was inspired by Apel and Flaig, Crank-Nicolson Schemes for Optimal Control Problems with Evolution Equations)

Step size selection

The step size can again be selected based on a quadratic expansion:

$$J(\mathbf{u} - \beta \nabla J) = J(\mathbf{u}) - \beta G + \frac{\beta^2}{2} H, \qquad \beta_{\text{opt}} = G/H.$$

Linear term:

$$G = \Delta t \sum_{k=2}^{N_T} (\nabla J)_{k-1/2}^{\top} (\nabla J)_{k-1/2}.$$

Step size selection

The step size can again be selected based on a quadratic expansion:

$$J(\mathbf{u} - \beta \nabla J) = J(\mathbf{u}) - \beta G + \tfrac{\beta^2}{2} H, \qquad \beta_{\text{opt}} = G/H.$$

Linear term:

$$G = \Delta t \sum_{k=2}^{N_T} (\nabla J)_{k-1/2}^\top (\nabla J)_{k-1/2}.$$

For the quadratic term, we need the solution of

$$\mathbf{E}\dot{\mathbf{x}}^\nabla(t) = \mathbf{A}\mathbf{x}^\nabla(t) + \mathbf{B}(\nabla J)(t), \qquad \mathbf{x}^\nabla(0) = \mathbf{0}.$$

In discretized form:

$$\left(\mathbf{E} - \tfrac{\Delta t}{2}\mathbf{A}\right) \mathbf{x}_k^\nabla = \left(\mathbf{E} + \tfrac{\Delta t}{2}\mathbf{A}\right) \mathbf{x}_{k-1}^\nabla + \Delta t \mathbf{B}(\nabla J)_{k-1/2}, \qquad \mathbf{x}_1^\nabla = \mathbf{0}.$$

The quadratic term H is then given by

$$H = \frac{\Delta t}{2} \sum_{k=2}^{N_T} \left[\left(\mathbf{x}_k^\nabla\right)^\top \mathbf{Q}\mathbf{x}_k^\nabla + \left(\mathbf{x}_{k-1}^\nabla\right)^\top \mathbf{Q}\mathbf{x}_{k-1}^\nabla \right] + \Delta t \sum_{k=2}^{N_T} (\nabla J)_{k-1/2}^\top \mathbf{R} (\nabla J)_{k-1/2}.$$

Note: we do not need a line search because the considered cost functional is quadratic.

1.E Appendix: Existence and uniqueness of minimizers

Existence of the infimum

We consider the minimization of a functional $J : U \to \mathbb{R}$ over a normed space U.
Note: U can be infinite dimensional.

We assume that $J(u) \geq 0$ for all $u \in U$.

We are also given a subset $U_{\mathrm{ad}} \subseteq U$ of admissible values for u.

Existence of the infimum

We consider the minimization of a functional $J : U \to \mathbb{R}$ over a normed space U.
Note: U can be infinite dimensional.

We assume that $J(u) \geq 0$ for all $u \in U$.

We are also given a subset $U_{\mathrm{ad}} \subseteq U$ of admissible values for u.

Then $\{J(u) \mid u \in U_{\mathrm{ad}}\}$ is a subset of $\mathbb{R}$ that is bounded from below (by 0). Therefore,

$$\inf_{u \in U_{\mathrm{ad}}} J(u) = \inf\{J(u) \mid u \in U_{\mathrm{ad}}\},$$

exists.
By definition of the infimum, there thus exists a sequence $u_1, u_2, u_3, \ldots$ in U_{ad} such that

$$J(u_k) \to \inf_{u \in U_{\mathrm{ad}}} J(u).$$

This sequence is called a *minimizing sequence*.

Existence of the minimizer (finite dimensional case)

Question: does

$$\min_{u\in U_{\mathrm{ad}}} J(u)$$

exist? In other words, is there a minimizer $u^* \in U_{\mathrm{ad}}$ such that

$$J(u^*) = \inf_{u\in U_{\mathrm{ad}}} J(u)?$$

Existence of the minimizer (finite dimensional case)

Question: does

$$\min_{u \in U_{\mathrm{ad}}} J(u)$$

exist? In other words, is there a minimizer $u^* \in U_{\mathrm{ad}}$ such that

$$J(u^*) = \inf_{u \in U_{\mathrm{ad}}} J(u)?$$

First consider the case where U is finite dimensional.

Observe, if U_{ad} is closed and the minimizing sequence $u_1, u_2, u_3, \ldots$ is bounded, then it also has a limit in U_{ad}. This limit is a minimizer u^*.

Two important cases:

- U_{ad} is bounded and closed.
  It is immediate that the minimizing sequence is bounded.

Existence of the minimizer (finite dimensional case)

Question: does

$$\min_{u \in U_{\text{ad}}} J(u)$$

exist? In other words, is there a minimizer $u^* \in U_{\text{ad}}$ such that

$$J(u^*) = \inf_{u \in U_{\text{ad}}} J(u)?$$

First consider the case where U is finite dimensional.

Observe, if U_{ad} is closed and the minimizing sequence $u_1, u_2, u_3, \ldots$ is bounded, then it also has a limit in U_{ad}. This limit is a minimizer u^*.

Two important cases:

- U_{ad} is bounded and closed.
  It is immediate that the minimizing sequence is bounded.
- $J(u)$ is coercive, i.e. $J(u_k) \to \infty$ if $|u_k| \to \infty$. Note: it is sufficient that $J(u) \geq |u|^2$.
  Then we can reason as follows.
  Suppose that the minimizing sequence $u_1, u_2, u_3, \ldots$ is unbounded.
  Then there exists a subsequence $u_{k_1}, u_{k_2}, u_{k_3}, \ldots$ such that $|u_{k_j}| \to \infty$.
  But $J(u_{k_j}) > |u_{k_j}|^2$, so also $J(u_{k_j}) \to \infty$.
  But then $J(u_{k_j})$ is not a minimizing sequence. Contradiction.
  Conclusion: the minimizing sequence must be bounded.

Existence of the minimizer (infinite dimensional case)

Question: does

$$\min_{u\in U_{\mathrm{ad}}} J(u)$$

exist? In other words, is there a minimizer $u^* \in U_{\mathrm{ad}}$ such that

$$J(u^*) = \inf_{u\in U_{\mathrm{ad}}} J(u)?$$

The infinite dimensional case is much more subtle.

Problem: We can no longer be sure that a bounded sequence has a (strong) limit.
In other words, we do no longer have compactness.

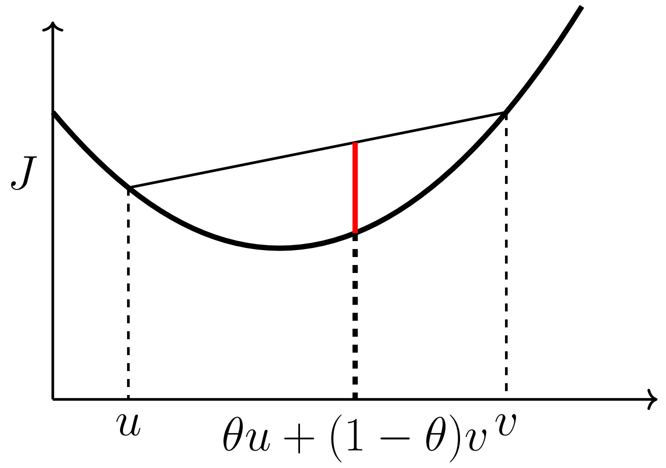
Typical example: consider $U_{\mathrm{ad}} = L^2(0,\pi)$ and consider the sequence $u_k = \sin(kx)$.
This sequence converges weakly to zero, but does not have a strong limit.

We will come back to this problem in a few slides.

Uniqueness of the minimizer (convex analysis)

The functional $J(u)$ is called α-convex iff

$$J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v) - \frac{\alpha\theta(1-\theta)}{2}|u-v|^2, \qquad \theta \in [0,1].$$



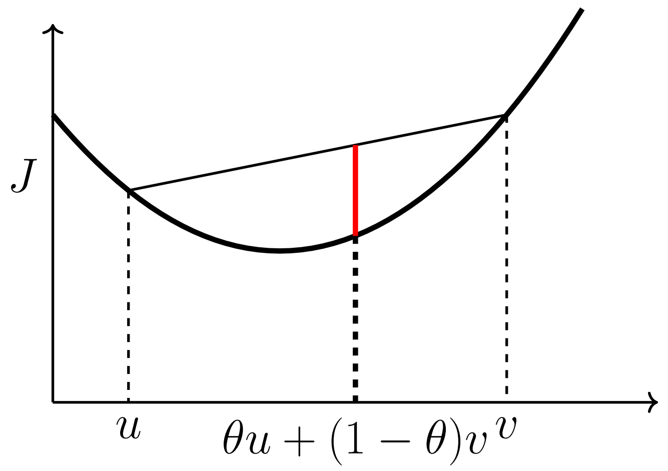
The admissible set U_{ad} is convex when $u, v \in U_{\text{ad}}$

$$\theta u + (1-\theta)v \in U_{\text{ad}}, \qquad \theta \in [0,1].$$

Uniqueness of the minimizer (convex analysis)

The functional $J(u)$ is called α-convex iff

$$J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v) - \frac{\alpha\theta(1-\theta)}{2}|u-v|^2, \qquad \theta \in [0,1].$$



The admissible set U_{ad} is convex when $u, v \in U_{\mathrm{ad}}$

$$\theta u + (1-\theta)v \in U_{\mathrm{ad}}, \qquad \theta \in [0,1].$$

Uniqueness of the minimizer:
Suppose that there are two points $u, v \in U_{\mathrm{ad}}$ such that $J(u) = J(v) = \min_{u\in U_{\mathrm{ad}}} J(u)$.

$$J(\theta u + (1-\theta)v) \leq \min_{u\in U_{\mathrm{ad}}} J(u) - \frac{\alpha\theta(1-\theta)}{2}|u-v|^2 < \min_{u\in U_{\mathrm{ad}}} J(u),$$

and $\theta u + (1-\theta)v \in U_{\mathrm{ad}}$. Contradiction.

Existence of the minimizer (infinite dimensional case, revisited)

Question: does

$$\min_{u \in U_{\mathrm{ad}}} J(u)$$

exist? In other words, is there a minimizer $u^* \in U_{\mathrm{ad}}$ such that

$$J(u^*) = \inf_{u \in U_{\mathrm{ad}}} J(u)?$$

Consider a minimizing sequence $u_1, u_2, u_3, \ldots$.
The minimizing sequence is bounded when U_{ad} is bounded or when J is coercive.
The bounded minimizing sequence $u_1, u_2, u_3, \ldots$ has a weak limit v.

Existence of the minimizer (infinite dimensional case, revisited)

Question: does

$$\min_{u \in U_{\text{ad}}} J(u)$$

exist? In other words, is there a minimizer $u^* \in U_{\text{ad}}$ such that

$$J(u^*) = \inf_{u \in U_{\text{ad}}} J(u)?$$

Consider a minimizing sequence $u_1, u_2, u_3, \ldots$.
The minimizing sequence is bounded when U_{ad} is bounded or when J is coercive.
The bounded minimizing sequence $u_1, u_2, u_3, \ldots$ has a weak limit v.

Now three problems remain:

- Is the weak limit $v \in U_{\text{ad}}$?
  If U_{ad} is strongly closed and convex, it is also weakly closed (Hahn-Banach).
- Do we have that $J(v) = \lim_{k\to\infty} J(u_k) = \inf_{u \in U_{\text{ad}}} J(u)$?
  This is achieved by assuming that J is weakly lower semi-continuous (by definition).
- Does the minimizing sequence $u_1, u_2, u_3, \ldots$ also converge strongly to v?
  This follows from the previous point and the strong convexity of J (with $\theta = \frac{1}{2}$):

$$J(v) \leq J(\tfrac{u_k+v}{2}) \leq \frac{J(u_k)+J(v)}{2} - \frac{\alpha}{8}|u_k - v|^2, \quad \Rightarrow \quad \frac{\alpha}{8}|u_k - v|^2 \leq \frac{J(u_k)-J(v)}{2} \to 0.$$

Chair DYNAMICS, CONTROL AND NUMERICS FAU

DDS Department of DATA SCIENCE

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG NATURWISSENSCHAFTLICHE FAKULTÄT

1.F Appendix: Inequality constraints

Inequality constraints

Consider the optimization problem

$$\min_{u \in U_{\text{ad}}} J(\mathbf{u}) = J(\mathbf{x}(\mathbf{u}), \mathbf{u})$$

with $\mathbf{u} \in U_{\text{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{Ax} + \mathbf{Bu} = \mathbf{0}.$$

We distinguish between two types of constraints:

- Constraints on $\mathbf{u}$ ('input constraints'), $g(\mathbf{u}) \geq \mathbf{0}$
- Constraints on $\mathbf{x}(\mathbf{u})$ ('state constraints') $h(\mathbf{x}(\mathbf{u})) \geq \mathbf{0}$.

Input constraints can be easily incorporated with the projected gradient method.

Projected gradient method

Suppose we want to solve an optimization problem with the constraints:

$$a \leq u_m \leq b, \qquad\qquad m \in \{1, 2, \ldots, M\}.$$

(This thus defines the admissible set U_{ad})

Projected gradient method

Suppose we want to solve an optimization problem with the constraints:

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \ldots, M\}.$$

(This thus defines the admissible set U_{ad})

Problem: We do not know whether $\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ is in U_{ad}.
(Even when $\mathbf{u}_k \in U_{\mathrm{ad}}$)

Solution: Project $\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ onto the U_{ad}, i.e. do the update as

$$\mathbf{u}_{k+1} = \Pi_{U_{\mathrm{ad}}} (\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)) \in U_{\mathrm{ad}}.$$

Projected gradient method

Suppose we want to solve an optimization problem with the constraints:

$$a \leq u_m \leq b, \qquad\qquad m \in \{1, 2, \ldots, M\}.$$

(This thus defines the admissible set U_{ad})

Problem: We do not know whether $\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ is in U_{ad}.
(Even when $\mathbf{u}_k \in U_{\text{ad}}$)

Solution: Project $\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ onto the U_{ad}, i.e. do the update as

$$\mathbf{u}_{k+1} = \Pi_{U_{\text{ad}}} \left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)\right) \in U_{\text{ad}}.$$

In general, the projection onto the admissible set is difficult to compute
(it requires the solution of another optimization problem).

However, for the considered admissible set, the computation is straightforward:

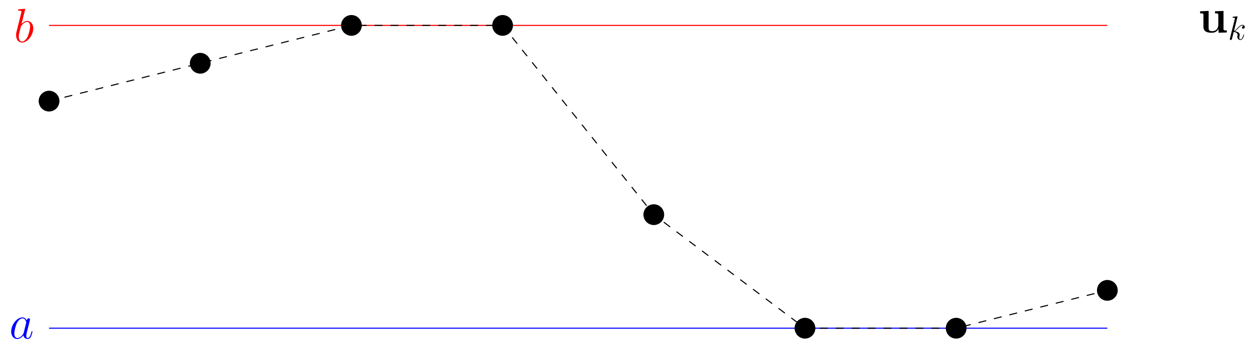
$$\left(\Pi_{U_{\text{ad}}}(\mathbf{u})\right)_m = \begin{cases} a & u_m \leq a, \\ u_m & a < u_m < b, \\ b & u_m \geq b. \end{cases}$$

Projected gradient method (graphical illustration)

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \ldots, M\}.$$

$$\mathbf{u}_{k+1} = \Pi_{U_{\text{ad}}}\left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)\right) \in U_{\text{ad}}$$

$$\left(\Pi_{U_{\text{ad}}}(\mathbf{u})\right)_m = \begin{cases} a & u_m \leq a \\ u_m & a < u_m < b \\ b & u_m \geq b \end{cases}$$

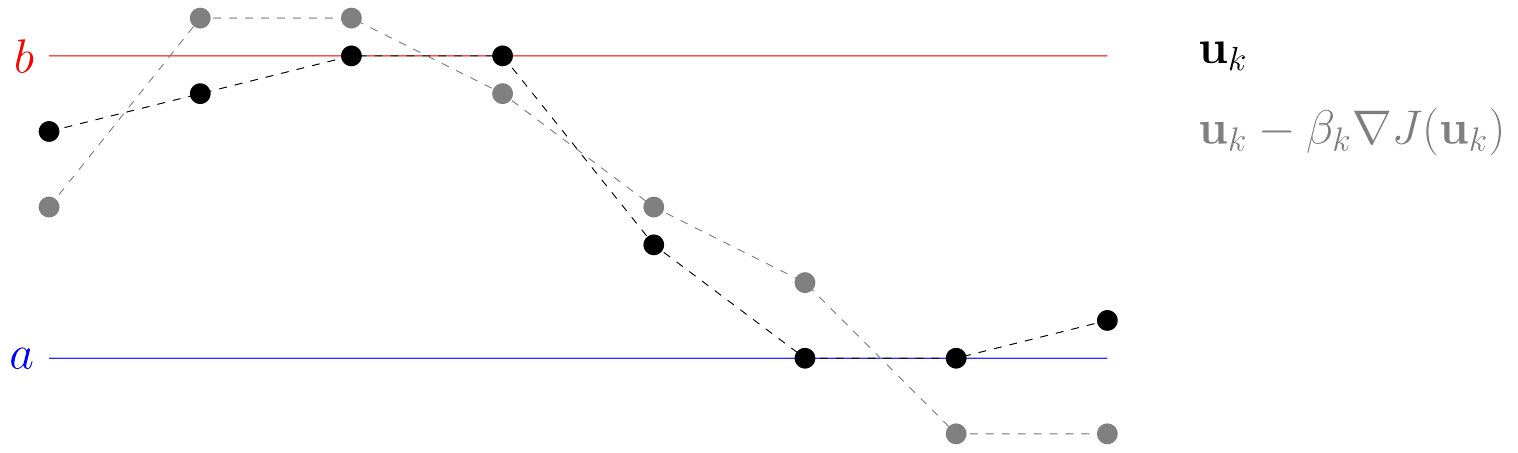


Projected gradient method (graphical illustration)

$$a \le u_m \le b, \qquad m \in \{1, 2, \dots, M\}.$$

$$\mathbf{u}_{k+1} = \Pi_{U_{\mathrm{ad}}}\left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)\right) \in U_{\mathrm{ad}}$$

$$\left(\Pi_{U_{\mathrm{ad}}}(\mathbf{u})\right)_m = \begin{cases} a & u_m \le a \\ u_m & a < u_m < b \\ b & u_m \ge b \end{cases}$$

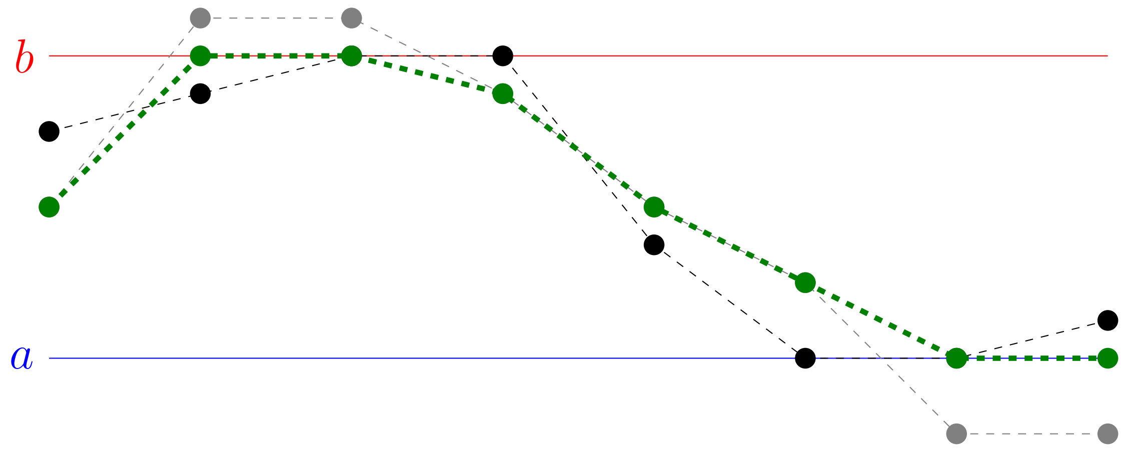


Projected gradient method (graphical illustration)

$$a \leq u_m \leq b, \qquad\qquad m \in \{1, 2, \dots, M\}.$$

$$\mathbf{u}_{k+1} = \Pi_{U_{\mathrm{ad}}}\left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)\right) \in U_{\mathrm{ad}}$$

$$\left(\Pi_{U_{\mathrm{ad}}}\left(\mathbf{u}\right)\right)_m = \begin{cases} a & u_m \leq a \\ u_m & a < u_m < b \\ b & u_m \geq b \end{cases}$$



$\mathbf{u}_k$

$\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$

$\Pi_{U_{\mathrm{ad}}}(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k))$

Quadratic approximation for the projected gradient

We replace $\nabla J(\mathbf{u}_k)$ by

$$\nabla \Pi J(\mathbf{u}_k) = -\lim_{h \downarrow 0} \frac{\Pi(\mathbf{u}_k - h\nabla J(\mathbf{u}_k)) - \mathbf{u}_k}{h}$$

$\nabla \Pi J(\mathbf{u}_k)$ is equal to $\nabla J(\mathbf{u}_k)$ except for entries where the $-\nabla J(\mathbf{u}_k)$ is pointing out of the admissible set.

Explicitly,

$$(\nabla \Pi J(\mathbf{u}_k))_m = \begin{cases} 0 & (\mathbf{u}_k)_m = a \text{ and } (\nabla J(\mathbf{u}_k))_m \geq 0 \\ & \quad \text{or } (\mathbf{u}_k)_m = b \text{ and } (\nabla J(\mathbf{u}_k))_m \leq 0 \\ (\nabla J(\mathbf{u}_k))_m & \text{otherwise.} \end{cases}$$

Quadratic approximation for the projected gradient

We replace $\nabla J(\mathbf{u}_k)$ by

$$\nabla \Pi J(\mathbf{u}_k) = -\lim_{h \downarrow 0} \frac{\Pi(\mathbf{u}_k - h\nabla J(\mathbf{u}_k)) - \mathbf{u}_k}{h}$$

$\nabla \Pi J(\mathbf{u}_k)$ is equal to $\nabla J(\mathbf{u}_k)$ except for entries where the $-\nabla J(\mathbf{u}_k)$ is pointing out of the admissible set.

Explicitly,

$$(\nabla \Pi J(\mathbf{u}_k))_m = \begin{cases} 0 & (\mathbf{u}_k)_m = a \text{ and } (\nabla J(\mathbf{u}_k))_m \geq 0 \\ & \quad\text{or } (\mathbf{u}_k)_m = b \text{ and } (\nabla J(\mathbf{u}_k))_m \leq 0 \\ (\nabla J(\mathbf{u}_k))_m & \text{otherwise.} \end{cases}$$

We then can use the quadratic approximation:

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3)$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla \Pi J(\mathbf{u}_k) \rangle$$

$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u_k} + \theta \nabla \Pi J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Computation of H with projected gradient (example)

Consider the optimization problem

$$\min_{u\in U_{\mathrm{ad}}} \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^\top \mathbf{R}\mathbf{u}$$

with $\mathbf{Q} = \mathbf{Q}^\top$, $\mathbf{R} = \mathbf{R}^\top$, $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$, and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

We have the ‘projected gradient’ (which is a bad name) $\nabla\Pi J(\mathbf{u}_k)$.

Compute the state resulting from the projected gradient

$$\mathbf{x}_k^{\nabla\Pi} = -\mathbf{A}^{-1}\left(\mathbf{B}\nabla\Pi J(\mathbf{u}_k)\right).$$

We can then compute

$$H = \left(\mathbf{x}_k^{\nabla\Pi}\right)^\top \mathbf{Q}\mathbf{x}_k^{\nabla\Pi} + \left(\nabla\Pi J(\mathbf{u}_k)\right)^\top \mathbf{R}\nabla\Pi J(\mathbf{u}_k).$$

State constraints

For state constraints (i.e. constraints on $\mathbf{x}(\mathbf{u})$),
it is not so straightforward to determine the projection on the admissible set.

State constraints can for example be included using a penalty function method, but we will not discuss this further in this course.