Numerical Optimal Control
– Part 3: Function space methods –

SADCO Summer School and Workshop on Optimal and Model Predictive Control
– OMPC 2013, Bayreuth –

Matthias Gerdts

Institute of Mathematics and Applied Computing
Department of Aerospace Engineering
Universität der Bundeswehr München (UniBw M)
matthias.gerdts@unibw.de
http://www.unibw.de/lrt1/gerdts

Fotos: http://de.wikipedia.org/wiki/München
Magnus Manske (Panorama), Luidger (Theatinerkirche), Kurmis (Chin. Turm), Arad Mojtahedi (Olympiapark), Max-k (Deutsches Museum), Oliver Raupach (Friedensengel), Andreas Praefcke (Nationaltheater)
Schedule and Contents

<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00 - 10:30</td>
<td>Introduction, overview, examples, indirect method</td>
</tr>
<tr>
<td>10:30 - 11:00</td>
<td>Coffee break</td>
</tr>
<tr>
<td>11:00 - 12:30</td>
<td>Discretization techniques, structure exploitation, calculation of gradients, extensions: sensitivity analysis, mixed-integer optimal control</td>
</tr>
<tr>
<td>12:30 - 14:00</td>
<td>Lunch break</td>
</tr>
<tr>
<td>14:00 - 15:30</td>
<td>Function space methods: Gradient and Newton type methods</td>
</tr>
<tr>
<td>15:30 - 16:00</td>
<td>Coffee break</td>
</tr>
<tr>
<td>16:00 - 17:30</td>
<td>Numerical experiments</td>
</tr>
</tbody>
</table>
Contents

Introduction

Necessary Conditions

Adjoint Formalism

Gradient Method
  Gradient Method in Finite Dimensions
  Gradient Method for Optimal Control Problems
  Extensions
  Gradient Method for Discrete Problem
  Examples

Lagrange-Newton Method
  Lagrange-Newton Method in Finite Dimensions
  Lagrange-Newton Method in Infinite Dimensions
  Application to Optimal Control
  Search Direction
  Examples
  Extensions
Contents

Introduction

Necessary Conditions

Adjoint Formalism

Gradient Method
  Gradient Method in Finite Dimensions
  Gradient Method for Optimal Control Problems
  Extensions
  Gradient Method for Discrete Problem
  Examples

Lagrange-Newton Method
  Lagrange-Newton Method in Finite Dimensions
  Lagrange-Newton Method in Infinite Dimensions
  Application to Optimal Control
  Search Direction
  Examples
  Extensions
Overview on Solution Methods

- **DAE Optimal Control Problem**
  - Discretization: Approximation by finite dimensional problem
  - Function space approach
    - Direct Approach: Methods for discretized optimization problem
      - Reduced approach (direct shooting)
        - SQP methods
          - Interior-Point Methods
          - Gradient methods
          - Penalty methods
          - Multiplier methods
          - Dynamic programming
      - Full discretization (collocation)
    - Indirect approach based on finite dimensional optimality system
      - Methods for finite dimensional complementarity problems and variational inequalities (semismooth Newton, Josephy-Newton, fixed-point iteration, projection methods)
    - Indirect approach based on infinite dimensional optimality system
      - Semi-analytical methods (indirect method, boundary value problems)
        - Reduced approach
        - Full approach
      - Methods for infinite dimensional complementarity problems and variational inequalities (semismooth Newton, Josephy-Newton, fixed-point iteration, projection methods)
      - SQP methods
        - Interior-Point Methods
        - Gradient methods
        - Penalty methods
        - Multiplier methods
        - Dynamic programming
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \quad \text{subject to } G(z) \in K, \ H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

What are the difficulties?
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \quad \text{subject to } G(z) \in K, \ H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

▶ no immediate approximation error: algorithms work in the same spaces as the problem

What are the difficulties?

▶ detailed functional analytic background necessary (cannot be expected in an industrial context)
▶ discretizations become necessary at lower level anyway; so, why not discretize right away?
▶ theoretical difficulties with, e.g., state constraints (multipliers are measures; how to handle them numerically?)
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \quad \text{subject to } \quad G(z) \in K, \quad H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

- no immediate approximation error: algorithms work in the same spaces as the problem
- massive exploitation of structure possible

What are the difficulties?

- detailed functional analytic background necessary (cannot be expected in an industrial context)
- discretizations become necessary at lower level anyway; so, why not discretize right away?
- theoretical difficulties with, e.g., state constraints (multipliers are measures; how to handle them numerically?)
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \quad \text{subject to } \quad G(z) \in K, \quad H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

- no immediate approximation error: algorithms work in the same spaces as the problem
- massive exploitation of structure possible
- subtle requirements can get lost in discretizations (e.g. smoothing operator for semismooth methods)

What are the difficulties?

- detailed functional analytic background necessary (cannot be expected in an industrial context)
- discretizations become necessary at lower level anyway; so, why not discretize right away?
- theoretical difficulties with, e.g., state constraints (multipliers are measures; how to handle them numerically?)
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \quad \text{subject to } G(z) \in K, \ H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

▶ no immediate approximation error: algorithms work in the same spaces as the problem
▶ massive exploitation of structure possible
▶ subtle requirements can get lost in discretizations (e.g. smoothing operator for semismooth methods)
▶ methods can be very fast

What are the difficulties?

▶ detailed functional analytic background necessary (cannot be expected in an industrial context)
▶ discretizations become necessary at lower level anyway; so, why not discretize right away?
▶ theoretical difficulties with, e.g., state constraints (multipliers are measures; how to handle them numerically?)
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \quad \text{subject to } \quad G(z) \in K, \ H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

- no immediate approximation error: algorithms work in the same spaces as the problem
- massive exploitation of structure possible
- subtle requirements can get lost in discretizations (e.g. smoothing operator for semismooth methods)
- methods can be very fast

What are the difficulties?

- detailed functional analytic background necessary (cannot be expected in an industrial context)
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \text{ subject to } G(z) \in K, \ H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

- no immediate approximation error: algorithms work in the same spaces as the problem
- massive exploitation of structure possible
- subtle requirements can get lost in discretizations (e.g. smoothing operator for semismooth methods)
- methods can be very fast

What are the difficulties?

- detailed functional analytic background necessary (cannot be expected in an industrial context)
- discretizations become necessary at lower level anyway; so, why not discretize right away?
Function Space Methods

Paradigm

Analyze and develop methods for some infinite dimensional optimization problem (e.g. an optimal control problem) of type

\[
\text{Minimize } J(z) \quad \text{subject to } G(z) \in K, \ H(z) = 0
\]

in the same Banach or Hilbert spaces where \( z, G \) and \( H \) live.

Why is this useful?

- no immediate approximation error: algorithms work in the same spaces as the problem
- massive exploitation of structure possible
- subtle requirements can get lost in discretizations (e.g. smoothing operator for semismooth methods)
- methods can be very fast

What are the difficulties?

- detailed functional analytic background necessary (cannot be expected in an industrial context)
- discretizations become necessary at lower level anyway; so, why not discretize right away?
- theoretical difficulties with, e.g., state constraints (multipliers are measures; how to handle them numerically?)
Functional Analysis

- **Banach space**: complete normed vector space \((X, \| \cdot \|_X)\)
Functional Analysis

- **Banach space**: complete normed vector space \((X, \| \cdot \|_X)\)
- **Hilbert space**: complete vector space \(X\) with inner product \(\langle \cdot, \cdot \rangle_X\times_X\)
Functional Analysis

- **Banach space**: complete normed vector space $(X, \| \cdot \|_X)$
- **Hilbert space**: complete vector space $X$ with inner product $\langle \cdot, \cdot \rangle_X \times X$
- **Dual space** $X^*$ of a vector space $X$ is defined by
  
  $X^* := \{ f : X \to \mathbb{R} \mid f \text{ linear and continuous} \}$, \quad \|f\|_{X^*} = \sup_{\|x\|=1} |f(x)|$

  $(X^*, \| \cdot \|_{X^*})$ is a Banach space, if $(X, \| \cdot \|_X)$ is a Banach space.
Functional Analysis

- **Banach space**: complete normed vector space \((X, \| \cdot \|_X)\)
- **Hilbert space**: complete vector space \(X\) with inner product \(\langle \cdot, \cdot \rangle_{X \times X}\)
- **Dual space** \(X^*\) of a vector space \(X\) is defined by
  \[
  X^* := \{ f : X \to \mathbb{R} \mid f \text{ linear and continuous} \}, \quad \|f\|_{X^*} = \sup_{\|x\|=1} |f(x)|
  \]
  \((X^*, \| \cdot \|_{X^*})\) is a Banach space, if \((X, \| \cdot \|_X)\) is a Banach space.
- For Banach spaces \(X\) and \(Y\), the **Fréchet derivative** of \(f : X \to Y\) at \(\hat{x}\) is a linear and continuous operator \(f'(\hat{x}) : X \to Y\) with the property
  \[
  \|f(\hat{x} + h) - f(\hat{x}) - f'(\hat{x})h\|_Y = o(\|h\|_X).
  \]
  The Fréchet derivative of \(f\) at \(\hat{x}\) in direction \(d\) is denoted by \(f'(\hat{x})d\).
Functional Analysis

- **Banach space**: complete normed vector space \((X, \| \cdot \|_X)\)
- **Hilbert space**: complete vector space \(X\) with inner product \(\langle \cdot, \cdot \rangle_X\)
- **Dual space** \(X^*\) of a vector space \(X\) is defined by
  \[ X^* := \{ f : X \to \mathbb{R} \mid f \text{ linear and continuous} \}, \quad \|f\|_{X^*} = \sup_{\|x\|=1} |f(x)| \]

\((X^*, \| \cdot \|_{X^*})\) is a Banach space, if \((X, \| \cdot \|_X)\) is a Banach space.

- For Banach spaces \(X\) and \(Y\), the Fréchet derivative of \(f : X \to Y\) at \(\hat{x}\) is a linear and continuous operator \(f'(\hat{x}) : X \to Y\) with the property
  \[ \|f(\hat{x} + h) - f(\hat{x}) - f'(\hat{x})h\|_Y = o(\|h\|_X). \]

The Fréchet derivative of \(f\) at \(\hat{x}\) in direction \(d\) is denoted by \(f'(\hat{x})d\).

- The Fréchet derivative of a function \(f : X \times U \to Y\) at \((\hat{x}, \hat{u})\) is given by
  \[ f'(\hat{x}, \hat{u})(x, u) = f'_x(\hat{x})x + f'_u(\hat{u})u, \]

  where \(f'_x\) and \(f'_u\) are the partial derivatives of \(f\) w.r.t. \(x\) and \(u\), respectively.
Functional Analysis

Lebesgue spaces

For $1 \leq p \leq \infty$ the Lebesgue spaces are defined by

$$L^p(\mathcal{I}, \mathbb{R}^n) := \{ f : \mathcal{I} \rightarrow \mathbb{R}^n \mid \|f\|_p < \infty \}$$

with

$$\|f\|_p := \left( \int_{\mathcal{I}} \|f(t)\|^p \right)^{1/p} \quad (1 \leq p < \infty),$$

$$\|f\|_\infty := \text{ess sup}_{t \in \mathcal{I}} \|f(t)\|.$$  

Properties:

- $L^2(\mathcal{I}, \mathbb{R}^n)$ is a Hilbert space with inner product $\langle f, g \rangle = \int_{\mathcal{I}} f(t)^\top g(t) dt$
- $L^\infty(\mathcal{I}, \mathbb{R}^n)$ is the dual space of $L^1(\mathcal{I}, \mathbb{R}^n)$, but not vice versa
- $L^p(\mathcal{I}, \mathbb{R}^n)$ and $L^q(\mathcal{I}, \mathbb{R}^n)$ are dual to each other, if $1/p + 1/q = 1$
Sobolev spaces

For $1 \leq p \leq \infty$ and $q \in \mathbb{N}$ the Sobolev spaces are defined by

$$W^{q,p}(\mathcal{I}, \mathbb{R}^n) := \{ f : \mathcal{I} \rightarrow \mathbb{R}^n \mid \|f\|_{q,p} < \infty \}$$

with

$$\|f\|_{q,p} := \left( \sum_{j=0}^{q} \|f^{(j)}\|_p \right)^{1/p} \quad (1 \leq p \leq \infty)$$

Properties:

- $W^{1,2}([t_0, t_f], \mathbb{R}^n)$ is a Hilbert space with inner product

  $$\langle f, g \rangle = f(t_0)^\top g(t_0) + \int_{t_0}^{t_f} f(t)^\top g(t) \, dt$$

- $W^{1,1}(\mathcal{I}, \mathbb{R}^n)$ is the space of absolutely continuous functions, i.e. functions satisfy

  $$f(t) = f(t_0) + \int_{t_0}^{t} f'(\tau) \, d\tau \quad \text{in } \mathcal{I} = [t_0, t_f].$$
Contents

Introduction

Necessary Conditions

Adjoint Formalism

Gradient Method
  Gradient Method in Finite Dimensions
  Gradient Method for Optimal Control Problems
  Extensions
  Gradient Method for Discrete Problem
  Examples

Lagrange-Newton Method
  Lagrange-Newton Method in Finite Dimensions
  Lagrange-Newton Method in Infinite Dimensions
  Application to Optimal Control
  Search Direction
  Examples
  Extensions
Infinite Optimization Problem

Throughout we restrict the discussion to the following class of optimization problems.

Problem (Infinite Optimization Problem (NLP))

Given:

- Banach spaces \((Z, \| \cdot \|_Z), (Y, \| \cdot \|_Y)\)
- mappings \(J : Z \rightarrow \mathbb{R}, H : Z \rightarrow Y\)
- convex set \(S \subseteq Z\)

Minimize \(J(z)\) subject to \(z \in S, \ H(z) = 0\)
### Infinite Optimization Problem

#### Theorem (KKT Conditions)

**Assumptions:**
- $\hat{z}$ is a local minimum of NLP
- $J$ is Fréchet-differentiable at $\hat{z}$, $H$ is continuously Fréchet-differentiable at $\hat{z}$
- $S$ is convex with non-empty interior
- **Mangasarian-Fromowitz Constraint Qualification (MFCQ):** $H'(\hat{z})$ is surjective and there exists $d \in \text{int}(S - \{\hat{z}\})$ with $H'(\hat{z})(d) = 0$

Then there exists a multiplier $\lambda^* \in Y^*$ such that

$$J'(\hat{z})(z - \hat{z}) + \lambda^*(H'(\hat{z})(z - \hat{z})) \geq 0 \quad \forall z \in S$$

For a proof see [1, Theorems 3.1,4.1].

On the Existence and Nonexistence of Lagrange Multipliers in Banach Spaces.
Infinite Optimization Problem

Lagrange function:

\[ L(z, \lambda^*) := J(z) + \lambda^*(H(z)) \]

Special cases:
Infinite Optimization Problem

Lagrange function:

\[ L(z, \lambda^*) := J(z) + \lambda^*(H(z)) \]

Special cases:

- If \( S = Z \), then

\[ J'(\hat{z})(z) + \lambda^*(H'(\hat{z})(z)) = 0 \quad \forall z \in Z \]

or equivalently

\[ L'_Z(\hat{z}, \lambda^*) = 0 \]
Infinite Optimization Problem

Lagrange function:

\[ L(z, \lambda^*) := J(z) + \lambda^*(H(z)) \]

Special cases:

- If \( S = Z \), then

\[ J'(\hat{z})(z) + \lambda^*(H'(\hat{z})(z)) = 0 \quad \forall z \in Z \]

or equivalently

\[ L'_Z(\hat{z}, \lambda^*) = 0 \]

- If \( H \) is not present, then

\[ J'(\hat{z})(z - \hat{z}) \geq 0 \quad \forall z \in S \]

If \( Z \) is a Hilbert space, then this condition is equivalent with the nonsmooth equation

\[ \hat{z} = \Pi_S (\hat{z} - \alpha J'(\hat{z})) \quad (\alpha > 0, \Pi_S \text{ projection onto } S) \]
Infinite Optimization Problem

Lagrange function:

\[ L(z, \lambda^*) := J(z) + \lambda^* (H(z)) \]

Special cases:

- If \( S = Z \), then
  \[ J'(\hat{z})(z) + \lambda^* (H'(\hat{z})(z)) = 0 \quad \forall z \in Z \]
  or equivalently
  \[ L'_z(\hat{z}, \lambda^*) = 0 \]

- If \( H \) is not present, then
  \[ J'(\hat{z})(z - \hat{z}) \geq 0 \quad \forall z \in S \]
  If \( Z \) is a Hilbert space, then this condition is equivalent with the nonsmooth equation
  \[ \hat{z} = \Pi_S (\hat{z} - \alpha J'(\hat{z})) \quad (\alpha > 0, \Pi_S \text{ projection onto } S) \]

- If \( S = Z \) and \( H \) is not present, then
  \[ J'(\hat{z})(z - \hat{z}) = 0 \quad \forall z \in S \]
Contents

Introduction

Necessary Conditions

Adjoint Formalism

Gradient Method
  Gradient Method in Finite Dimensions
  Gradient Method for Optimal Control Problems
  Extensions
  Gradient Method for Discrete Problem
  Examples

Lagrange-Newton Method
  Lagrange-Newton Method in Finite Dimensions
  Lagrange-Newton Method in Infinite Dimensions
  Application to Optimal Control
  Search Direction
  Examples
  Extensions
Adjoint Formalism

Equality constrained NLP (EQ-NLP)

Minimize \( J(x, u) \) subject to \( Ax = Bu \).

\( A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, J : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R} \) differentiable
Adjoint Formalism

### Equality constrained NLP (EQ-NLP)

Minimize $J(x, u)$ subject to $Ax = Bu$. 

$A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, J: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ differentiable

**Solution operator**: Let $A$ be nonsingular. Then:

$$Ax = Bu \iff x = A^{-1}Bu =: Su$$

$S: \mathbb{R}^m \rightarrow \mathbb{R}^n, u \mapsto x = Su$ is called solution operator.
### Adjoint Formalism

**Equality constrained NLP (EQ-NLP)**

<table>
<thead>
<tr>
<th>Minimize</th>
<th>( J(x, u) )</th>
<th>subject to</th>
<th>( Ax = Bu ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \in \mathbb{R}^{n \times n} ), ( B \in \mathbb{R}^{n \times m} ), ( J : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R} ) differentiable</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Solution operator:** Let \( A \) be nonsingular. Then:

\[
Ax = Bu \iff x = A^{-1}Bu =: Su
\]

\( S : \mathbb{R}^m \rightarrow \mathbb{R}^n, u \mapsto x = Su \) is called **solution operator**.

**Reduced objective functional:**

\[
J(x, u) = J(Su, u) =: j(u), \quad j : \mathbb{R}^m \rightarrow \mathbb{R}.
\]
Equality constrained NLP (EQ-NLP)

Minimize \( J(x, u) \) subject to \( Ax = Bu \).

\( A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, J : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R} \) differentiable

Solution operator: Let \( A \) be nonsingular. Then:

\[
Ax = Bu \quad \iff \quad x = A^{-1}Bu =: Su
\]

\( S : \mathbb{R}^m \rightarrow \mathbb{R}^n, u \mapsto x = Su \) is called solution operator.

Reduced objective functional:

\[
J(x, u) = J(Su, u) =: j(u), \quad j : \mathbb{R}^m \rightarrow \mathbb{R}.
\]

Reduced NLP (R-NLP)

Minimize \( j(u) = J(Su, u) \) w.r.t. \( u \in \mathbb{R}^m \).
## Adjoint Formalism

### Task

**Compute gradient of** $j$ **at a given** $\hat{u}$, for instance in a gradient based optimization method or for the evaluation of necessary optimality conditions.**
Task
Compute gradient of $j$ at a given $\hat{u}$, for instance in a gradient based optimization method or for the evaluation of necessary optimality conditions.

Differentiation yields

$$j'(\hat{u}) = J_x'(S\hat{u}, \hat{u})S + J_u'(S\hat{u}, \hat{u})$$

$$= J_x'(S\hat{u}, \hat{u})A^{-1}B + J_u'(S\hat{u}, \hat{u})$$
Adjoint Formalism

**Task**

Compute gradient of $j$ at a given $\hat{u}$, for instance in a gradient based optimization method or for the evaluation of necessary optimality conditions.

Differentiation yields

$$j'(\hat{u}) = J'_x(S\hat{u}, \hat{u})S + J'_u(S\hat{u}, \hat{u})$$

$$= J'_x(S\hat{u}, \hat{u})A^{-1}B + J'_u(S\hat{u}, \hat{u})$$

Computation of $A^{-1}$ is expensive. Try to avoid it!
Adjoint Formalism

**Task**

Compute gradient of $j$ at a given $\hat{u}$, for instance in a gradient based optimization method or for the evaluation of necessary optimality conditions.

Differentiation yields

$$j'(\hat{u}) = J'_x(S\hat{u}, \hat{u})S + J'_u(S\hat{u}, \hat{u})$$

$$= J'_x(S\hat{u}, \hat{u})A^{-1}B + J'_u(S\hat{u}, \hat{u})$$

Computation of $A^{-1}$ is expensive. Try to avoid it!

To this end define **adjoint vector** $\lambda$ by

$$\lambda^\top := J'_x(S\hat{u}, \hat{u})A^{-1}$$
Adjoint Formalism

Adjoint equation

\[ A^\top \lambda = \nabla_x J(S\hat{u}, \hat{u}) \]

The gradient of \( j \) at \( \hat{u} \) reads

\[ j''(\hat{u}) = \lambda^\top B + J'_\hat{u}(S\hat{u}, \hat{u}) \]
Adjoint Formalism

Adjoint equation

\[ A^\top \lambda = \nabla_x J(S\hat{u}, \hat{u}) \]

The gradient of \( j \) at \( \hat{u} \) reads

\[ j'(\hat{u}) = \lambda^\top B + J'_u(S\hat{u}, \hat{u}) \]

Connection to KKT conditions: Lagrange function of EQ-NLP

\[ L(x, u, \lambda) := J(x, u) + \lambda^\top (Bu - Ax) \]

If \( \lambda \) solves adjoint equation, then

\[ L'_x(\hat{x}, \hat{u}, \lambda) = J'_x(\hat{x}, \hat{u}) - \lambda^\top A = 0 \]
Adjoint Formalism

Adjoint equation

\[ A^\top \lambda = \nabla_x J(S\hat{u}, \hat{u}) \]

The gradient of \( j \) at \( \hat{u} \) reads

\[ j'(\hat{u}) = \lambda^\top B + J'_u(S\hat{u}, \hat{u}) \]

Connection to KKT conditions: Lagrange function of EQ-NLP

\[ L(x, u, \lambda) := J(x, u) + \lambda^\top (Bu - Ax) \]

If \( \lambda \) solves adjoint equation, then

\[ L'_x(\hat{x}, \hat{u}, \lambda) = J'_x(\hat{x}, \hat{u}) - \lambda^\top A = 0 \]

This choice of \( \lambda \) automatically satisfies one part of the KKT conditions.
Adjoint Formalism

Adjoint equation

\[ A^T \lambda = \nabla_x J(S\hat{u}, \hat{u}) \]

The gradient of \( j \) at \( \hat{u} \) reads

\[ j'(\hat{u}) = \lambda^T B + J'_u(S\hat{u}, \hat{u}) \]

Connection to KKT conditions: Lagrange function of EQ-NLP

\[ L(x, u, \lambda) := J(x, u) + \lambda^T (Bu - Ax) \]

If \( \lambda \) solves adjoint equation, then

\[ L'_x(\hat{x}, \hat{u}, \lambda) = J'_x(\hat{x}, \hat{u}) - \lambda^T A = 0 \]

This choice of \( \lambda \) automatically satisfies one part of the KKT conditions. The second part

\[ 0 = L'_u(\hat{x}, \hat{u}, \lambda) = J'_u(\hat{x}, \hat{u}) + \lambda^T B = j'(\hat{u}) \]

is only satisfied at a stationary point \( \hat{u} \) of the reduced problem R-NLP!
Adjoint Formalism

Let $X$ and $Y$ be Banach spaces and $X^*$ and $Y^*$ their topological dual spaces.

**Adjoint operator**

Let $\mathcal{A} : X \rightarrow Y$ be a bounded linear operator. The operator $\mathcal{A}^* : Y^* \rightarrow X^*$ with the property

$$(\mathcal{A}^* y^*)(x) = y^*(\mathcal{A} x) \quad \forall x \in X, y^* \in Y^*$$

is called adjoint operator (it is a bounded linear operator). Notion: $(\mathcal{A}^* y^*, x) = (y^*, \mathcal{A} x)$ “dual pairing”

**Equality constrained NLP (EQ-NLP)**

Minimize $J(x, u)$ subject to $\mathcal{A} x = \mathcal{B} u.$

$\mathcal{A} : X \rightarrow Y, \mathcal{B} : U \rightarrow Y$ bounded linear operators, $J : X \times U \rightarrow \mathbb{R}$ Fréchet differentiable, $X, U, Y$ Banach spaces
Solution operator: Let $\mathcal{A}$ be nonsingular. Then:

$$\mathcal{A}x = Bu \Leftrightarrow x = \mathcal{A}^{-1}Bu =: Su$$

with solution operator $S : U \longrightarrow X$, $u \mapsto x = Su$. 
Solution operator: Let $A$ be nonsingular. Then:

$$Ax = Bu \quad \iff \quad x = A^{-1}Bu =: Su$$

with solution operator $S : U \rightarrow X, u \mapsto x = Su$.

Reduced objective functional:

$$J(x, u) = J(Su, u) =: j(u), \quad j : U \rightarrow \mathbb{R}.$$
Adjoint Formalism

Solution operator: Let $\mathcal{A}$ be nonsingular. Then:

$$\mathcal{A} x = \mathcal{B} u \iff x = \mathcal{A}^{-1} \mathcal{B} u =: S u$$

with solution operator $S : U \rightarrow X, u \mapsto x = S u$.

Reduced objective functional:

$$J(x, u) = J(S u, u) =: j(u), \quad j : U \rightarrow \mathbb{R}.$$

Reduced NLP (R-NLP)

Minimize $j(u) = J(S u, u)$ w.r.t. $u \in U$. 
Adjoint Formalism

Differentiation at $\hat{u}$ in direction $u$ yields

\[
j'(\hat{u})u = J'_x(S\hat{u}, \hat{u})Su + J'_u(S\hat{u}, \hat{u})u
= J'_x(S\hat{u}, \hat{u})A^{-1}Bu + J'_u(S\hat{u}, \hat{u})u
\]
Adjoint Formalism

Differentiation at \( \hat{u} \) in direction \( u \) yields

\[
\begin{align*}
    j'(\hat{u})u &= J'_x(S\hat{u}, \hat{u})Su + J'_u(S\hat{u}, \hat{u})u \\
    &= J'_x(S\hat{u}, \hat{u})A^{-1}Bu + J'_u(S\hat{u}, \hat{u})u
\end{align*}
\]

Computation of \( A^{-1} \) is expensive. Try to avoid it!
Adjoint Formalism

Differentiation at \( \hat{u} \) in direction \( u \) yields

\[
j'(\hat{u})u = J'_x(S\hat{u}, \hat{u})Su + J'_u(S\hat{u}, \hat{u})u
\]

\[
= J'_x(S\hat{u}, \hat{u})A^{-1}Bu + J'_u(S\hat{u}, \hat{u})u
\]

Computation of \( A^{-1} \) is expensive. Try to avoid it!

Define adjoint \( \lambda \in Y^* \) by

\[
y^* := J'_x(S\hat{u}, \hat{u})A^{-1} \quad \Leftrightarrow \quad \begin{aligned}
y^*(Ax) &= J'_x(S\hat{u}, \hat{u})x \\
&= (A^*y^*)x
\end{aligned}
\]

Adjoint equation

\[
A^*y^* = J'_x(S\hat{u}, \hat{u}) \quad \text{(operator equation in } X^*\text{)}
\]
Adjoint Formalism

Differentiation at \( \hat{u} \) in direction \( u \) yields

\[
j'(\hat{u})u = J'_x(S\hat{u}, \hat{u})Su + J'_u(S\hat{u}, \hat{u})u
\]

\[
= J'_x(S\hat{u}, \hat{u})A^{-1}Bu + J'_u(S\hat{u}, \hat{u})u
\]

Computation of \( A^{-1} \) is expensive. Try to avoid it!

Define adjoint \( \lambda \in Y^* \) by

\[
y^* := J'_x(S\hat{u}, \hat{u})A^{-1} \quad \Leftrightarrow \quad y^*(Ax) = J'_x(S\hat{u}, \hat{u})x
\]

\[
= (A^*y^*)x
\]

Adjoint equation

\[
A^*y^* = J'_x(S\hat{u}, \hat{u}) \quad \text{(operator equation in } X^* )
\]

The gradient of \( j \) at \( \hat{u} \) in direction \( u \) reads

\[
j'(\hat{u})u = y^*Bu + J'_u(S\hat{u}, \hat{u})u
\]
Adjoint Formalism

Connection to KKT conditions:

- Lagrange function of EQ-NLP

\[ L(x, u, y^*) := J(x, u) + y^*(B u - Ax) \]
Connection to KKT conditions:

- Lagrange function of EQ-NLP
  \[ L(x, u, y^*) := J(x, u) + y^*(Bu - Ax) \]

- \( y^* \) solves adjoint equation \( \Rightarrow \)
  \[ L_x'(\hat{x}, \hat{u}, y^*)x = J_x'(\hat{x}, \hat{u})x - y^*(Ax) = J_x'(\hat{x}, \hat{u})x - (A^*y^*)x = 0 \]

This choice of \( y^* \) automatically satisfies one part of the KKT conditions.
Adjoint Formalism

Connection to KKT conditions:

▶ Lagrange function of EQ-NLP

\[ L(x, u, y^*) := J(x, u) + y^*(Bu - Ax) \]

▶ \( y^* \) solves adjoint equation \( \implies \)

\[ L_x'(\hat{x}, \hat{u}, y^*)x = J_x'(\hat{x}, \hat{u})x - y^*(Ax) = J_x'(\hat{x}, \hat{u})x - (A^*y^*)x = 0 \]

This choice of \( y^* \) automatically satisfies one part of the KKT conditions.

▶ The second part

\[ 0 = L_u'(\hat{x}, \hat{u}, y^*) = J_u'(\hat{x}, \hat{u}) + y^*B = j'(\hat{u}) \]

is only satisfied at a stationary point \( \hat{u} \) of the reduced problem R-NLP!
How to compute the adjoint operator?

**Theorem (Riesz)**

*Given:* Hilbert space $X$, inner product $\langle \cdot, \cdot \rangle$, dual space $X^*$.  
For every $f^* \in X^*$ there exists a unique $f \in X$ such that $\|f^*\|_{X^*} = \|f\|_X$ and  
\[ f^*(x) = \langle f, x \rangle \quad \forall x \in X. \]

**Example (linear ODE)**

*Given:* $\mathcal{I} := [0, T]$, linear mapping $\mathcal{A} : W^{1,2}(\mathcal{I}, \mathbb{R}^n) =: X \longrightarrow Y := L^2(\mathcal{I}, \mathbb{R}^n) \times \mathbb{R}^n$, \[
(\mathcal{A}x)(\cdot) = \begin{pmatrix}
x'(\cdot) - A(\cdot)x(\cdot) \\
x(0)
\end{pmatrix}.
\]

$X$ and $Y$ are Hilbert spaces. According to Riesz’ theorem for $y^* \in Y^*$ and $x^* \in X^*$ there exist $(\lambda, \sigma) \in Y$ and $\mu \in X$ with  
\[
y^*(h, k) = \int_{\mathcal{I}} \lambda(t)^\top h(t) \, dt + \sigma^\top k \quad \forall (h, k) \in Y
\]
\[
x^*(x) = \mu(0)^\top x(0) + \int_{\mathcal{I}} \mu'(t)^\top x'(t) \, dt \quad \forall x \in X
\]
How to compute the adjoint operator?

Example (linear ODE, continued)

We intend to identify \( \lambda, \sigma, \) and \( \mu \) such that \( y^*(Ax) = (A^*y^*)(x) \) holds for all \( x \in X \) and \( y^* \in Y^* \). Note, that \( A^*y^* \in X^* \).

By partial integration we obtain

\[
y^*(Ax) = \int_I \lambda(t) \top (x'(t) - A(t)x(t)) \, dt + \sigma \top x(0)
\]
How to compute the adjoint operator?

Example (linear ODE, continued)

We intend to identify $\lambda$, $\sigma$, and $\mu$ such that $y^*(Ax) = (A^*y^*)(x)$ holds for all $x \in X$ and $y^* \in Y^*$. Note, that $A^*y^* \in X^*$.

By partial integration we obtain

$$y^*(Ax) = \int_X \lambda(t)^T (x'(t) - A(t)x(t)) \, dt + \sigma^T x(0)$$

$$= - \left[ \left( - \int_{t}^{T} \lambda(s)^T A(s) \, ds \right) x(t) \right]_0^T + \sigma^T x(0) + \int_{t}^{T} \left( \lambda(t)^T - \int_{t}^{T} \lambda(s)^T A(s) \, ds \right) x'(t) \, dt$$
How to compute the adjoint operator?

Example (linear ODE, continued)

We intend to identify \( \lambda, \sigma, \) and \( \mu \) such that \( y^*(Ax) = (A^*y^*)(x) \) holds for all \( x \in X \) and \( y^* \in Y^* \). Note, that \( A^*y^* \in X^* \).

By partial integration we obtain

\[
y^*(Ax) = \int_I \lambda(t)^T (x'(t) - A(t)x(t)) \ dt + \sigma^T x(0)
\]

\[
= -\left[ \left( -\int_t^T \lambda(s)^T A(s)ds \right) x(t) \right]_0^T + \sigma^T x(0)
\]

\[
+ \int_I \left( \lambda(t)^T - \int_t^T \lambda(s)^T A(s)ds \right) x'(t) dt
\]

\[
= \left( \sigma^T - \int_0^T \lambda(s)^T A(s)ds \right) x(0)
\]

\[
+ \int_I \left( \lambda(t)^T - \int_t^T \lambda(s)^T A(s)ds \right) x'(t) dt
\]

\[
= \mu^T(0)
\]

\[
= \mu'(t)^T
\]
How to compute the adjoint operator?

Example (linear ODE, continued)

We intend to identify \( \lambda, \sigma, \) and \( \mu \) such that \( y^*(Ax) = (A^*y^*)(x) \) holds for all \( x \in X \) and \( y^* \in Y^* \). Note, that \( A^*y^* \in X^* \).

By partial integration we obtain

\[
y^*(Ax) = \int_{I} \lambda(t)^T (x'(t) - A(t)x(t)) \, dt + \sigma^T x(0)
\]

\[
= - \left[ \left( - \int_{t}^{T} \lambda(s)^T A(s) ds \right) x(t) \right]_{0}^{T} + \sigma^T x(0)
\]

\[
+ \int_{I} \left( \lambda(t)^T - \int_{t}^{T} \lambda(s)^T A(s) ds \right) x'(t) \, dt
\]

\[
= \left( \sigma^T - \int_{0}^{T} \lambda(s)^T A(s) ds \right) x(0) + \int_{I} \left( \lambda(t)^T - \int_{t}^{T} \lambda(s)^T A(s) ds \right) x'(t) \, dt
\]

\[
= \left( \sigma^T - \int_{0}^{T} \lambda(s)^T A(s) ds \right) x(0) + \left[ \left( \lambda(t)^T - \int_{t}^{T} \lambda(s)^T A(s) ds \right) x'(t) \right]_{0}^{T}
\]

\[
= \mu(0)^T + \mu'(T)^T
\]

\[
= \mu'(t)^T
\]

\[
= (A^*y^*)(x).
\]
How to compute the adjoint operator?

**Example (linear ODE, continued)**

We intend to identify \( \lambda, \sigma, \) and \( \mu \) such that \( y^*((Ax)) = (A^*y^*)(x) \) holds for all \( x \in X \) and \( y^* \in Y^* \). Note, that \( A^*y^* \in X^* \).

By partial integration we obtain

\[
y^*(Ax) = \int_I \lambda(t)^T (x'(t) - A(t)x(t)) \, dt + \sigma^T x(0)
\]

\[
= -\left[ \left( -\int_t^T \lambda(s)^T A(s)ds \right) x(t) \right]_0^T + \sigma^T x(0)
\]

\[
+ \int_I \left( \lambda(t)^T - \int_t^T \lambda(s)^T A(s)ds \right) x'(t) \, dt
\]

\[
= \left( \sigma^T - \int_0^T \lambda(s)^T A(s)ds \right) x(0) + \int_I \left( \lambda(t)^T - \int_t^T \lambda(s)^T A(s)ds \right) x'(t) \, dt
\]

\[
= \mu^*(0)^T + \mu'(t)^T
\]

\[
= (A^*y^*)(x).
\]

The latter equation defines the adjoint operator since \( y^*(Ax) = (A^*y^*)(x) \) holds for all \( x \in X \), \( y^* \in Y^* \).
How to compute the adjoint equation?

Example (OCP and adjoint equation)

Optimal control problem: (assumption: $\varphi$ Fréchet differentiable)

Minimize $J(x, u) := \varphi(x(T))$ s.t. $\begin{pmatrix} x'(t) - A(t)x(t) \\ x(0) \end{pmatrix} = \begin{pmatrix} B(t)u(t) \\ 0 \end{pmatrix} = \mathcal{A}(x)(t)$

Adjoint equation: For every $x \in W^{1,2}(\mathcal{I}, \mathbb{R}^n)$ we have

$$0 = (\mathcal{A}^* y^*)(x) - J'(\hat{x}, \hat{u})(x)$$
How to compute the adjoint equation?

Example (OCP and adjoint equation)

Optimal control problem: (assumption: \( \varphi \) Fréchet differentiable)

Minimize \( J(x, u) := \varphi(x(T)) \) s.t. 
\[
\begin{pmatrix}
  x'(t) - A(t)x(t) \\
  x(0)
\end{pmatrix} = \begin{pmatrix}
  B(t)u(t) \\
  0
\end{pmatrix}
\]

Adjoint equation: For every \( x \in W^{1,2}(I, \mathbb{R}^n) \) we have

\[
0 = (A^*y^*)(x) - J'(\hat{x}, \hat{u})(x)
\]

\[
= \left( \sigma - \lambda(s)^T A(s) ds \right) x(0) + \int_I \left( \lambda(t)^T - \int_t^T \lambda(s)^T A(s) ds \right) x'(t) dt
\]

\[- \varphi'(\hat{x}(T))x(T)\]
How to compute the adjoint equation?

Example (OCP and adjoint equation)

Optimal control problem: (assumption: \( \varphi \) Fréchet differentiable)

Minimize

\[
J(x, u) := \varphi(x(T))
\]

s.t.

\[
\begin{pmatrix}
  x'(t) - A(t)x(t) \\
  x(0)
\end{pmatrix} =
\begin{pmatrix}
  B(t)u(t) \\
  0
\end{pmatrix}
\]

Adjoint equation: For every \( x \in W^{1,2}(\mathcal{I}, \mathbb{R}^n) \) we have

\[
0 = (A^* \dot{y}^*)(x) - J'(\dot{x}, \dot{u})(x)
\]

\[
= \left( \sigma^T - \int_0^T \lambda(s)^T A(s) ds \right) x(0) + \int_{\mathcal{I}} \left( \lambda(t)^T - \int_t^T \lambda(s)^T A(s) ds \right) x'(t) dt
\]

\[- \varphi'(\hat{x}(T))x(T) \]

Application of variation lemma (DuBois-Reymond) yields

\[
\lambda(t)^T = \lambda(T)^T + \int_t^T \lambda(s)^T A(s) ds
\]

and

\[
\lambda(0) = \sigma, \quad \lambda(T)^T = \varphi'(\hat{x}(T)).
\]
How to compute the adjoint equation?

Example (OCP and adjoint equation, continued)

- Adjoint equation:
  \[ \dot{\lambda}(t) = -A(t)^\top \lambda(t), \quad \lambda(T) = \nabla \varphi(\hat{x}(T)). \]

- Gradient of reduced objective functional \( j(u) = J(x(u), u) \):
  \[
  j'(\hat{u})(u) = y^*(Bu) + J'_u(\hat{x}, \hat{u})u \\
  = \int_I \lambda(t)^\top B(t)u(t) \, dt + \sigma^\top 0 \\
  = \int_I \lambda(t)^\top B(t)u(t) \, dt \quad (u \in L^2(I, \mathbb{R}^m))
  \]

- Stationary point:
  \[
  0 = j'(\hat{u})(u) = \int_I \lambda(t)^\top B(t)u(t) \, dt \quad \forall u \in L^2(I, \mathbb{R}^m)
  \]

implies
\[
0 = B(t)^\top \lambda(t) \quad \text{a.e. in } I
\]
Contents

Introduction

Necessary Conditions

Adjoint Formalism

Gradient Method
  Gradient Method in Finite Dimensions
  Gradient Method for Optimal Control Problems
  Extensions
  Gradient Method for Discrete Problem
  Examples

Lagrange-Newton Method
  Lagrange-Newton Method in Finite Dimensions
  Lagrange-Newton Method in Infinite Dimensions
  Application to Optimal Control
  Search Direction
  Examples
  Extensions
Gradient Method in Finite Dimensions

Unconstrained minimization problem

Minimize $J(u)$ w.r.t. $u \in \mathbb{R}^n$

$J : \mathbb{R}^n \rightarrow \mathbb{R}$ continuously differentiable
Gradient Method in Finite Dimensions

Unconstrained minimization problem

Minimize $J(u)$ w.r.t. $u \in \mathbb{R}^n$

$J : \mathbb{R}^n \rightarrow \mathbb{R}$ continuously differentiable

Gradient Method for Finite Dimensional Problems

(0) Let $u^{(0)} \in \mathbb{R}^n$, $\beta \in (0, 1)$, $\sigma \in (0, 1)$, and $k := 0$. 
Gradient Method in Finite Dimensions

Unconstrained minimization problem

Minimize $J(u)$ w.r.t. $u \in \mathbb{R}^n$

$J : \mathbb{R}^n \rightarrow \mathbb{R}$ continuously differentiable

Gradient Method for Finite Dimensional Problems

(0) Let $u^{(0)} \in \mathbb{R}^n$, $\beta \in (0, 1)$, $\sigma \in (0, 1)$, and $k := 0$.

(1) Compute $d^{(k)} := -\nabla J(u^{(k)})$. 
Gradient Method in Finite Dimensions

Unconstrained minimization problem

Minimize \( J(u) \) \ w.r.t. \( u \in \mathbb{R}^n \)

\( J : \mathbb{R}^n \rightarrow \mathbb{R} \) continuously differentiable

Gradient Method for Finite Dimensional Problems

(0) Let \( u^{(0)} \in \mathbb{R}^n \), \( \beta \in (0, 1) \), \( \sigma \in (0, 1) \), and \( k := 0 \).

(1) Compute \( d^{(k)} := -\nabla J(u^{(k)}) \).

(2) If \( ||d^{(k)}|| \approx 0 \), STOP.
Gradient Method in Finite Dimensions

Unconstrained minimization problem

Minimize $J(u)$ w.r.t. $u \in \mathbb{R}^n$

$J : \mathbb{R}^n \rightarrow \mathbb{R}$ continuously differentiable

Gradient Method for Finite Dimensional Problems

(0) Let $u^{(0)} \in \mathbb{R}^n$, $\beta \in (0, 1)$, $\sigma \in (0, 1)$, and $k := 0$.
(1) Compute $d^{(k)} := -\nabla J(u^{(k)})$.
(2) If $\|d^{(k)}\| \approx 0$, STOP.
(3) Perform line-search: Find smallest $j \in \{0, 1, 2, \ldots\}$ with

$$J(u^{(k)} + \beta^j d^{(k)}) \leq J(u^{(k)}) - \sigma \beta^j \|\nabla J(u^{(k)})\|_2^2$$

and set $\alpha_k := \beta^j$. 
Unconstrained minimization problem

Minimize \( J(u) \) w.r.t. \( u \in \mathbb{R}^n \)

\( J : \mathbb{R}^n \rightarrow \mathbb{R} \) continuously differentiable

Gradient Method for Finite Dimensional Problems

(0) Let \( u^{(0)} \in \mathbb{R}^n \), \( \beta \in (0, 1) \), \( \sigma \in (0, 1) \), and \( k := 0 \).

(1) Compute \( d^{(k)} := -\nabla J(u^{(k)}) \).

(2) If \( \|d^{(k)}\| \approx 0 \), STOP.

(3) Perform line-search: Find smallest \( j \in \{0, 1, 2, \ldots\} \) with

\[
J(u^{(k)} + \beta^j d^{(k)}) \leq J(u^{(k)}) - \sigma \beta^j \|\nabla J(u^{(k)})\|_2^2
\]

and set \( \alpha_k := \beta^j \).

(4) Set \( u^{(k+1)} := u^{(k)} + \alpha_k d^{(k)} \), \( k := k + 1 \), and go to (1).
Gradient Method

Pro’s:
- requires only first derivatives
- easy to implement
- global convergence achieved by Armijo linesearch
- extension: projected gradient method for simple constraints

Con’s:
- only linear convergence rate
- only unconstrained minimization (except simple bounds)
Optimal control problem (OCP)

Given: \( \mathcal{I} := [t_0, t_f], \bar{x} \in \mathbb{R}^{n_x} \), continuously differentiable functions

\[
\varphi : \mathbb{R}^{n_x} \longrightarrow \mathbb{R}, \\
f_0 : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \longrightarrow \mathbb{R}, \\
f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \longrightarrow \mathbb{R}^{n_x}.
\]

Minimize

\[
\Gamma(x, u) := \varphi(x(t_f)) + \int_{\mathcal{I}} f_0(x(t), u(t)) dt
\]

w.r.t. \( x \in W^{1,\infty}(\mathcal{I}, \mathbb{R}^{n_x}), u \in L^{\infty}(\mathcal{I}, \mathbb{R}^{n_u}) \) subject to the constraints

\[
x'(t) = f(x(t), u(t)) \quad \text{a.e. in } \mathcal{I}, \\
x(t_0) = \bar{x}.
\]

Define \( X := W^{1,\infty}(\mathcal{I}, \mathbb{R}^{n_x}) \) and \( U := L^{\infty}(\mathcal{I}, \mathbb{R}^{n_u}) \).
Assumptions: (existence of solution operator)

- For every \( u \in U \) the initial value problem

\[
\begin{align*}
    x'(t) &= f(x(t), u(t)) \quad \text{a.e. in } I, \\
    x(t_0) &= \bar{x}
\end{align*}
\]

has a unique solution \( x = S(u) \in X \).

- The solution mapping \( S : U \rightarrow X \) is continuously Fréchet-differentiable.

Reduced optimal control problem (R-OCP)

Minimize \( J(u) := \Gamma(S(u), u) \) subject to \( u \in U \).

Gradient method requires gradient of reduced objective function \( J \) at some \( \hat{u} \in U \).
Gradient Method for Optimal Control Problems

**Assumptions:** (existence of solution operator)
- For every $u \in U$ the initial value problem

$$
\begin{align*}
x'(t) &= f(x(t), u(t)) \quad \text{a.e. in } I, \\
x(t_0) &= \bar{x}
\end{align*}
$$

has a unique solution $x = S(u) \in X$.
- The solution mapping $S : U \to X$ is continuously Fréchet-differentiable.

**Reduced optimal control problem (R-OCP)**

Minimize $J(u) := \Gamma(S(u), u)$ subject to $u \in U$.

Gradient method requires gradient of reduced objective function $J$ at some $\hat{u} \in U$. How does the gradient look like?
Computation of Gradient

- In $\mathbb{R}^n$:

$$
\nabla J(\hat{u}) := J'(u)^\top = \begin{pmatrix}
\frac{\partial J}{\partial u_1}(\hat{u}) \\
\vdots \\
\frac{\partial J}{\partial u_n}(\hat{u})
\end{pmatrix}, \quad J'(\hat{u})(u) = \nabla J(\hat{u})^\top u = \langle J(\hat{u}), u \rangle_{\mathbb{R}^n \times \mathbb{R}^n},
$$
Computation of Gradient

**In \( \mathbb{R}^n \):**

\[
\nabla J(\hat{u}) := J'(u)^\top = \begin{pmatrix}
\frac{\partial J}{\partial u_1}(\hat{u}) \\
\vdots \\
\frac{\partial J}{\partial u_n}(\hat{u})
\end{pmatrix},
\]

\[
J'(\hat{u})(u) = \nabla J(\hat{u})^\top u = \langle J(\hat{u}), u \rangle_{\mathbb{R}^n \times \mathbb{R}^n},
\]

**In a Hilbert space setting, i.e. \( J : U \rightarrow \mathbb{R}, \ U \) Hilbert space:**

\[
J'(\hat{u}) \in U^* \quad \text{Riesz} \quad \exists \eta(\hat{u}) \in U : J'(\hat{u})(u) = \langle \eta(\hat{u}), u \rangle_{U \times U}
\]

Hence, \( \nabla J(\hat{u}) := \eta(\hat{u}) \in U \) is the gradient of \( J \) at \( \hat{u} \).
Computation of Gradient

- In $\mathbb{R}^n$:

$$\nabla J(\hat{u}) := J'(u)^\top = \begin{pmatrix}
\frac{\partial J}{\partial u_1}(\hat{u}) \\
\vdots \\
\frac{\partial J}{\partial u_n}(\hat{u})
\end{pmatrix}, \quad J'(\hat{u})(u) = \nabla J(\hat{u})^\top u = \langle J(\hat{u}), u \rangle_{\mathbb{R}^n \times \mathbb{R}^n},$$

- In a Hilbert space setting, i.e $J : U \rightarrow \mathbb{R}$, $U$ Hilbert space:

$$J'(\hat{u}) \in U^* \quad \text{Riesz} \quad \exists \eta(\hat{u}) \in U : J'(\hat{u})(u) = \langle \eta(\hat{u}), u \rangle_{U \times U}$$

Hence, $\nabla J(\hat{u}) := \eta(\hat{u}) \in U$ is the gradient of $J$ at $\hat{u}$.

- But, in our case $U = L^\infty(\mathcal{I}, \mathbb{R}^n_u)$ is not a Hilbert space and hence the functional $J'(\hat{u}) \in U^*$ does not a priori have such a nice representation as in the above cases.
Computation of Gradient

- In $\mathbb{R}^n$:

$$\nabla J(\hat{u}) := J'(u)^\top = \begin{pmatrix}
\frac{\partial J}{\partial u_1}(\hat{u}) \\
\vdots \\
\frac{\partial J}{\partial u_n}(\hat{u})
\end{pmatrix}, \quad J'(\hat{u})(u) = \nabla J(\hat{u})^\top u = \langle J(\hat{u}), u \rangle_{\mathbb{R}^n \times \mathbb{R}^n},$$

- In a Hilbert space setting, i.e $J : U \rightarrow \mathbb{R}$, $U$ Hilbert space:

$$J'(\hat{u}) \in U^* \xrightarrow{\text{Riesz}} \exists \eta(\hat{u}) \in U : J'(\hat{u})(u) = \langle \eta(\hat{u}), u \rangle_{U \times U}$$

Hence, $\nabla J(\hat{u}) := \eta(\hat{u}) \in U$ is the gradient of $J$ at $\hat{u}$.

- But, in our case $U = L^\infty(\mathcal{I}, \mathbb{R}^{nu})$ is not a Hilbert space and hence the functional $J'(\hat{u}) \in U^*$ does not a priori have such a nice representation as in the above cases.

How to define the gradient in this case? $\rightarrow$ use formal Lagrange technique (see also metric gradient, see [1])

Computation of Gradient by Formal Lagrange Technique

Hamilton function:
\[ \mathcal{H}(x, u, \lambda) := f_0(x, u) + \lambda^T f(x, u) \]

Auxiliary functional: \((\hat{u}, \hat{x} = S(\hat{u})\) given; \(\lambda \in X\) to be specified later)
\[ \tilde{J}(\hat{u}) := J(\hat{u}) + \langle \lambda, f(\hat{x}, \hat{u}) - \hat{x}' \rangle_{L^2(\mathcal{I}, \mathbb{R}^n_x) \times L^2(\mathcal{I}, \mathbb{R}^n_x)} \]
Hamilton function:

\[ \mathcal{H}(x, u, \lambda) := f_0(x, u) + \lambda^\top f(x, u) \]

Auxiliary functional: \((\hat{u}, \hat{x} = S(\hat{u})\) given; \(\lambda \in X\) to be specified later)

\[ \tilde{J}(\hat{u}) := J(\hat{u}) + \langle \lambda, f(\hat{x}, \hat{u}) - \hat{x}' \rangle_{L^2(I, \mathbb{R}^{nx}) \times L^2(I, \mathbb{R}^{nx})} \]

\[ = \varphi(\hat{x}(t_f)) + \int_I \mathcal{H}(\hat{x}(t), \hat{u}(t), \lambda(t)) - \lambda(t)^\top \hat{x}'(t) \, dt \]
Computation of Gradient by Formal Lagrange Technique

Hamilton function:
\[ \mathcal{H}(x, u, \lambda) := f_0(x, u) + \lambda^\top f(x, u) \]

Auxiliary functional: \((\hat{u}, \hat{x} = S(\hat{u}) \text{ given}; \lambda \in X \text{ to be specified later})\)
\[ \tilde{J}(\hat{u}) := J(\hat{u}) + \langle \lambda, f(\hat{x}, \hat{u}) - \hat{x}' \rangle_{L^2(\mathcal{I}, \mathbb{R}^{nx}) \times L^2(\mathcal{I}, \mathbb{R}^{nx})} \]
\[ = \varphi(\hat{x}(t_f)) + \int_\mathcal{I} \mathcal{H}(\hat{x}(t), \hat{u}(t), \lambda(t)) - \lambda(t)^\top \hat{x}'(t) \, dt \]
\[ \overset{p.l.}{=} \varphi(\hat{x}(t_f)) - \left[ \lambda(t)^\top \hat{x}(t) \right]_{t_0}^{t_f} + \int_\mathcal{I} \mathcal{H}(\hat{x}(t), \hat{u}(t), \lambda(t)) + \lambda'(t)^\top \hat{x}(t) \, dt. \]
Computation of Gradient by Formal Lagrange Technique

Hamilton function:
\[
\mathcal{H}(x, u, \lambda) := f_0(x, u) + \lambda^T f(x, u)
\]

Auxiliary functional: \((\hat{u}, \hat{x} = S(\hat{u})\) given; \(\lambda \in X\) to be specified later)
\[
\tilde{J}(\hat{u}) := J(\hat{u}) + \langle \lambda, f(\hat{x}, \hat{u}) - \hat{x}' \rangle_{L^2(\mathcal{I}, \mathbb{R}^{nx}) \times L^2(\mathcal{I}, \mathbb{R}^{nx})}
\]
\[
= \varphi(\hat{x}(t_f)) + \int_{\mathcal{I}} \mathcal{H}(\hat{x}(t), \hat{u}(t), \lambda(t)) - \lambda(t)^T \hat{x}'(t) \, dt
\]
\[
p.l. \quad \varphi(\hat{x}(t_f)) - \left[ \lambda(t)^T \hat{x}(t) \right]_{t_0}^{t_f} + \int_{\mathcal{I}} \mathcal{H}(\hat{x}(t), \hat{u}(t), \lambda(t)) + \lambda'(t)^T \hat{x}(t) \, dt.
\]

Fréchet derivative: (exploit \(S'(\hat{u})(t_0) = 0\))
\[
\tilde{J}'(\hat{u})(u) = \left( \varphi'(\hat{x}(t_f)) - \lambda(t_f)^T \right) S'(\hat{u})(t_f)
\]
\[
+ \int_{\mathcal{I}} \left( \mathcal{H}'_x[t] + \lambda'(t)^T \right) S'(\hat{u})(t) + \mathcal{H}'_u[t] u(t) \, dt.
\]

Computation of \(S'(\hat{u})\) is expensive! Eliminate red terms:
Computation of Gradient by Formal Lagrange Technique

Hamilton function:

\[ \mathcal{H}(x, u, \lambda) := f_0(x, u) + \lambda^\top f(x, u) \]

Auxiliary functional: \((\hat{u}, \hat{x} = S(\hat{u}) \text{ given; } \lambda \in X \text{ to be specified later})\)

\[ \tilde{J}(\hat{u}) := J(\hat{u}) + \langle \lambda, f(\hat{x}, \hat{u}) - \hat{x}' \rangle_{L^2(\mathcal{I}, \mathbb{R}^{nx}) \times L^2(\mathcal{I}, \mathbb{R}^{nx})} \]

\[ = \varphi(\hat{x}(t_f)) + \int_\mathcal{I} \mathcal{H}(\hat{x}(t), \hat{u}(t), \lambda(t)) - \lambda(t)^\top \hat{x}'(t) \, dt \]

\[ \overset{p.l.}{=} \varphi(\hat{x}(t_f)) - \left[ \lambda(t)^\top \hat{x}(t) \right]_{t_0}^{t_f} + \int_\mathcal{I} \mathcal{H}(\hat{x}(t), \hat{u}(t), \lambda(t)) + \lambda'(t)^\top \hat{x}(t) \, dt. \]

Fréchet derivative: (exploit \(S'(\hat{u})(t_0) = 0\))

\[ \tilde{J}'(\hat{u})(u) = \left( \varphi'(\hat{x}(t_f)) - \lambda(t_f)^\top \right) S'(\hat{u})(t_f) \]

\[ + \int_\mathcal{I} \left( \mathcal{H}_x'[t] + \lambda'(t)^\top \right) S'(\hat{u})(t) + \mathcal{H}_u'[t] u(t) \, dt. \]

Computation of \(S'(\hat{u})\) is expensive! Eliminate red terms:

\[ \lambda'(t) = -\mathcal{H}_x'(\hat{x}(t), \hat{u}(t), \lambda(t))^\top \]

\[ \lambda(t_f) = \varphi'(\hat{x}(t_f))^\top \]

\[ \{ \text{adjoint ODE} \} \]
**Computation of Gradient**

"Gradient form":

\[
\tilde{J}'(\hat{u})(u) = \int_{\mathcal{I}} \mathcal{H}'_u[t]u(t)dt = \langle \nabla_u \mathcal{H}, u \rangle_{L^2(\mathcal{I}, \mathbb{R}^{nu}) \times L^2(\mathcal{I}, \mathbb{R}^{nu})}.
\]

**Theorem (steepest descent)**

The direction

\[
d(t) := -\frac{1}{\|\mathcal{H}'_u\|_2} \mathcal{H}'_u[t]^{\top}
\]
solves

Minimize \( \tilde{J}'(\hat{u})(u) \) w.r.t. \( u \) subject to \( \|u\|_2 = 1 \).

**Proof:** For every \( u \) with \( \|u\|_2 = 1 \) we have by the Schwarz inequality,

\[
|\tilde{J}'(\hat{u})(u)| \leq \|\mathcal{H}'_u\|_2 \cdot \|u\|_2 = \|\mathcal{H}'_u\|_2
\]

and for \( d \) we have

\[
\tilde{J}'(\hat{u})(d) = -\|\mathcal{H}'_u\|_2.
\]

\( \square \)
Computation of Gradient

Relation between \( \tilde{J}'(\hat{u}) \) and \( J'(\hat{u}) \), for a proof see [5, Chapter 8]:

**Theorem**

Let \( \hat{u} \in U \) and \( \hat{x} = S(\hat{u}) \) be given. Let \( \lambda \) satisfy the adjoint ODE

\[
\lambda' = -H_x'(\hat{x}, \hat{u}, \lambda)^\top, \quad \lambda(t_f) = \varphi'(\hat{x}(t_f))^\top.
\]

Then:

\[
\tilde{J}'(\hat{u})(u) = J'(\hat{u})(u) \quad \forall u \in U.
\]

Owing to the theorem we may define the gradient of \( J \) at \( \hat{u} \) as follows.

**Definition (Gradient of reduced objective functional)**

Let \( \hat{u} \in U \) and \( \hat{x} = S(\hat{u}) \) be given. Let \( \lambda \) satisfy the adjoint ODE. The gradient \( \nabla J(\hat{u}) \in U \) of \( J \) at \( \hat{u} \) is defined by

\[
\nabla J(\hat{u})(\cdot) := H_u'(\hat{x}(\cdot), \hat{u}(\cdot), \lambda(\cdot))^\top.
\]
Gradient Method

Gradient method for R-OCP

(0) Choose \( u^{(0)} \in U, \beta \in (0, 1), \sigma \in (0, 1), \) and set \( k := 0. \)

(1) Solve the ODE
\[
x' = f(x, u^{(k)}), \quad x(t_0) = \bar{x},
\]
and the adjoint ODE
\[
\lambda' = -H' \mathcal{H}_x(x, u^{(k)}, \lambda) \top, \quad \lambda(t_f) = \varphi'(x(t_f)) \top.
\]
Denote the solution by \( x^{(k)} = S(u^{(k)}) \) and \( \lambda^{(k)} \).

(2) If \( \|H'_u\|_2 \approx 0 \), STOP.

(3) Set
\[
d^{(k)}(t) := -H'_u(x^{(k)}(t), u^{(k)}(t), \lambda^{(k)}(t)) \top.
\]

(4) Perform an Armijo line-search: Find smallest \( j \in \{0, 1, 2, \ldots \} \) with
\[
J(u^{(k)} + \beta^j d^{(k)}) \leq J(u^{(k)}) - \sigma \beta^j \|H'_u(x^{(k)}, u^{(k)}, \lambda^{(k)})\|_2^2
\]
and set \( \alpha_k := \beta^j \).

(5) Set \( u^{(k+1)} := u^{(k)} + \alpha_k d^{(k)}, \ k := k + 1, \) and go to (1).
### Theorem (Convergence)

Suppose that the gradient method does not terminate. Let \( u_* \) be an accumulation point of the sequence \( \{u^{(k)}\}_{k \in \mathbb{N}} \) generated by the gradient method and \( x_* := S(u_*) \). Then:

\[
\| \nabla J(u_*) \|_2 = 0.
\]
Gradient Method – Examples

Example

Minimize $x_2(1)$ subject to the constraints

$$x_1'(t) = -x_1(t) + \sqrt{3}u(t), \quad x_1(0) = 2,$$

$$x_2'(t) = \frac{1}{2} \left( x_1(t)^2 + u(t)^2 \right), \quad x_2(0) = 0.$$  

Output of gradient method: $(u^{(0)}) \equiv 0, \beta = 0.9, \sigma = 0.1$, symplectic Euler, $N = 100$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\alpha_k$</th>
<th>$J(u^{(k)})$</th>
<th>$|H'<em>u|</em>\infty$</th>
<th>$|H'_u|_2^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00000000E+00</td>
<td>0.87037219E+00</td>
<td>0.14877655E+01</td>
<td>0.65717322E+00</td>
</tr>
<tr>
<td>1</td>
<td>0.10000000E+01</td>
<td>0.72406641E+00</td>
<td>0.61765831E+00</td>
<td>0.21168343E+00</td>
</tr>
<tr>
<td>2</td>
<td>0.10000000E+01</td>
<td>0.68017548E+00</td>
<td>0.35175249E+00</td>
<td>0.72633493E-01</td>
</tr>
<tr>
<td>3</td>
<td>0.10000000E+01</td>
<td>0.66515486E+00</td>
<td>0.20519966E+00</td>
<td>0.24977643E-01</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>23</td>
<td>0.10000000E+01</td>
<td>0.65728265E+00</td>
<td>0.47781223E-05</td>
<td>0.13406009E-10</td>
</tr>
<tr>
<td>24</td>
<td>0.10000000E+01</td>
<td>0.65728265E+00</td>
<td>0.28158031E-05</td>
<td>0.46203574E-11</td>
</tr>
<tr>
<td>25</td>
<td>0.10000000E+01</td>
<td>0.65728265E+00</td>
<td>0.16459056E-05</td>
<td>0.15877048E-11</td>
</tr>
<tr>
<td>26</td>
<td>0.10000000E+01</td>
<td>0.65728265E+00</td>
<td>0.97422070E-06</td>
<td>0.54326016E-12</td>
</tr>
</tbody>
</table>
Example (continued)

Some iterates (red) and converged solution (blue):
Gradient Method – Examples

Example

Minimize $x_2(1) + 2.5(x_1(1) - 1)^2$ subject to the constraints

$$x_1'(t) = u(t) - 15 \exp(-2t), \quad x_1(0) = 4,$$
$$x_2'(t) = \frac{1}{2} \left( u(t)^2 + x_1(t)^3 \right), \quad x_2(0) = 0.$$

Output of gradient method: \( (u^{(0)} \equiv 0, \beta = 0.9, \sigma = 0.1, \text{symplectic Euler, } N = 100) \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \alpha_k )</th>
<th>( J(u^{(k)}) )</th>
<th>( |H'<em>u|</em>\infty )</th>
<th>( |H'_u|^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00000000E+00</td>
<td>0.32263509E+02</td>
<td>0.17750257E+02</td>
<td>0.23786744E+03</td>
</tr>
<tr>
<td>1</td>
<td>0.31381060E+00</td>
<td>0.21450571E+02</td>
<td>0.16994586E+02</td>
<td>0.18987670E+03</td>
</tr>
<tr>
<td>2</td>
<td>0.25418658E+00</td>
<td>0.15604667E+02</td>
<td>0.91485607E+01</td>
<td>0.78563930E+02</td>
</tr>
<tr>
<td>3</td>
<td>0.28242954E+00</td>
<td>0.11414005E+02</td>
<td>0.75417222E+01</td>
<td>0.39835695E+02</td>
</tr>
<tr>
<td>4</td>
<td>0.25418658E+00</td>
<td>0.84261386E+01</td>
<td>0.22532128E-05</td>
<td>0.28170368E-11</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>61</td>
<td>0.28242954E+00</td>
<td>0.84261386E+01</td>
<td>0.22532128E-05</td>
<td>0.28170368E-11</td>
</tr>
<tr>
<td>62</td>
<td>0.28242954E+00</td>
<td>0.84261386E+01</td>
<td>0.16626581E-05</td>
<td>0.17267323E-11</td>
</tr>
<tr>
<td>63</td>
<td>0.28242954E+00</td>
<td>0.84261386E+01</td>
<td>0.16639237E-05</td>
<td>0.13398784E-11</td>
</tr>
<tr>
<td>64</td>
<td>0.22876792E+00</td>
<td>0.84261386E+01</td>
<td>0.83196204E-06</td>
<td>0.21444710E-12</td>
</tr>
</tbody>
</table>
Example (continued)

Some iterates (red) and converged solution (blue):

- State $x_1$ over time $t$ [s]
- Control $u(t)$ over time $t$ [s]
- Adjoint lambda1 over time $t$ [s]
Projected Gradient Method

We add “simple” constraints \( u \in \mathcal{U} \) with a convex set \( \mathcal{U} \subset U \) to the reduced problem.

Reduced optimal control problem (R-OCP)

Minimize \( J(u) := \Gamma(S(u), u) \) subject to \( u \in \mathcal{U} \subset U \).

Assumptions:
- the projection \( \Pi_{\mathcal{U}} : U \rightarrow \mathcal{U} \) is easy to compute
Projected Gradient Method

We add “simple” constraints \( u \in \mathcal{U} \) with a convex set \( \mathcal{U} \subset U \) to the reduced problem.

**Reduced optimal control problem (R-OCP)**

Minimize \( J(u) := \Gamma(S(u), u) \) subject to \( u \in \mathcal{U} \subset U \).

**Assumptions:**
- the projection \( \Pi_{\mathcal{U}} : U \rightarrow \mathcal{U} \) is easy to compute

**Example:** For box constraints

\[ \mathcal{U} = \{ u \in \mathbb{R} \mid a \leq u \leq b \} \]

the projection computes to

\[
\Pi_{\mathcal{U}}(u) = \max\{a, \min\{b, u\}\} = \begin{cases} 
  a, & \text{if } u < a, \\
  u, & \text{if } a \leq u \leq b, \\
  b, & \text{if } u > b 
\end{cases}
\]
Projected Gradient Method

We add “simple” constraints $u \in \mathcal{U}$ with a convex set $\mathcal{U} \subset U$ to the reduced problem.

**Reduced optimal control problem (R-OCP)**

Minimize $J(u) := \Gamma(S(u), u)$ subject to $u \in \mathcal{U} \subset U$.

**Assumptions:**
- the projection $\Pi_{\mathcal{U}} : U \longrightarrow \mathcal{U}$ is easy to compute

**Example:** For box constraints

$$\mathcal{U} = \{ u \in \mathbb{R} | a \leq u \leq b \}$$

the projection computes to

$$\Pi_{\mathcal{U}}(u) = \max\{a, \min\{b, u\}\} = \begin{cases} a, & \text{if } u < a, \\ u, & \text{if } a \leq u \leq b, \\ b, & \text{if } u > b \end{cases}$$

**Optimality:**

$$J'(\hat{u})(u - \hat{u}) \geq 0 \quad \forall u \in \mathcal{U} \iff \hat{u} = \Pi_{\mathcal{U}}(\hat{u} - \alpha J'(\hat{u})) \quad (\alpha > 0)$$
Projected Gradient Method

The projected gradient method requires a feasible initial \(u^{(0)}\) and differs from the gradient method in one of the following two components:

**Version 1:** In iteration \(k\) compute

\[
\tilde{u}^{(k)} := \Pi_u \left( u^{(k)} + d^{(k)} \right)
\]

and use the direction \(\tilde{d}^{(k)} := \tilde{u}^{(k)} - u^{(k)}\) instead of \(d^{(k)}\) in steps (4) and (5), i.e.

\[
J \left( u^{(k)} + \beta \tilde{d}^{(k)} \right) \leq J(u^{(k)}) + \sigma \beta \left| J'(u^{(k)}) \right| \tilde{d}^{(k)} , \quad u^{(k+1)} = u^{(k)} + \alpha_k \tilde{d}^{(k)}
\]
Projected Gradient Method

The projected gradient method requires a feasible initial $u^{(0)}$ and differs from the gradient method in one of the following two components:

**Version 1:** In iteration $k$ compute

$$\tilde{u}^{(k)} := \Pi_u \left( u^{(k)} + d^{(k)} \right)$$

and use the direction $\tilde{d}^{(k)} := \tilde{u}^{(k)} - u^{(k)}$ instead of $d^{(k)}$ in steps (4) and (5), i.e.

$$J \left( u^{(k)} + \beta^i \tilde{d}^{(k)} \right) \leq J(u^{(k)}) + \sigma \beta^i J'(u^{(k)}) \tilde{d}^{(k)}, \quad u^{(k+1)} = u^{(k)} + \alpha_k \tilde{d}^{(k)}$$

**Version 2:** In iteration $k$ use the projection within the Armijo linesearch in step (4), i.e.

$$J \left( \Pi_u \left( u^{(k)} + \beta^j d^{(k)} \right) \right) \leq J(u^{(k)}) + \sigma \beta^j J'(u^{(k)}) d^{(k)}$$

and set $u^{(k+1)} := \Pi_u \left( u^{(k)} + \alpha_k d^{(k)} \right)$ in step (5)
Example

Minimize \( x_2(1) + 2.5(x_1(1) - 1)^2 \) subject to the constraints

\[
\begin{align*}
x'_1(t) &= u(t) - 15 \exp(-2t), \\
x'_2(t) &= \frac{1}{2} \left( u(t)^2 + x_1(t)^3 \right), \\
u(t) &\in U := \{ u \in \mathbb{R} \mid 1 \leq u \leq 3 \}.
\end{align*}
\]

Output of projected gradient method (version 1): \( (u^{(0)} \equiv 1, \beta = 0.9, \sigma = 0.1, \) symplectic Euler, \( N = 100) \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \alpha_k )</th>
<th>( J(u^{(k)}) )</th>
<th>( | u^{(k)} - \Pi(U(u^{(k)} - J'(u^{(k)})) |_\infty )</th>
<th>( J'(u^{(k)})d(k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00000000E+00</td>
<td>0.19145273E+02</td>
<td>0.20000000E+01</td>
<td>-0.21446656E+02</td>
</tr>
<tr>
<td>1</td>
<td>0.10000000E+01</td>
<td>0.87483967E+01</td>
<td>0.20000000E+01</td>
<td>-0.10674827E+01</td>
</tr>
<tr>
<td>2</td>
<td>0.47829690E+00</td>
<td>0.86918573E+01</td>
<td>0.95659380E+00</td>
<td>-0.39783867E+00</td>
</tr>
<tr>
<td>3</td>
<td>0.10000000E+01</td>
<td>0.85622908E+01</td>
<td>0.16165415E+01</td>
<td>-0.34588659E+00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>56</td>
<td>0.81000000E+00</td>
<td>0.84783285E+01</td>
<td>0.13795810E-05</td>
<td>-0.18974642E-12</td>
</tr>
<tr>
<td>57</td>
<td>0.72900000E+00</td>
<td>0.84783285E+01</td>
<td>0.13795797E-05</td>
<td>-0.98603024E-13</td>
</tr>
<tr>
<td>58</td>
<td>0.81000000E+00</td>
<td>0.84783285E+01</td>
<td>0.90526344E-06</td>
<td>-0.25693860E-13</td>
</tr>
</tbody>
</table>
Projected Gradient Method – Examples

Example (continued)

Some iterates (red) and converged solution (blue):
Instead of applying the “function space” gradient method to OCP, we could first discretize OCP and apply the standard gradient method to the discretized problem.

Problem (Discretized optimal control problem (D-OCP))

Given: \( \bar{x} \in \mathbb{R}^{nx} \), grid \( G_N \), and continuously differentiable functions defined by

\[ \varphi : \mathbb{R}^{nx} \rightarrow \mathbb{R}, \quad f_0 : \mathbb{R}^{nx} \times \mathbb{R}^{nu} \rightarrow \mathbb{R}, \quad f : \mathbb{R}^{nx} \times \mathbb{R}^{nu} \rightarrow \mathbb{R}^{nx}, \]

\[ G_N := \{ t_i \mid t_i = t_0 + ih, \ i = 0, 1, \ldots, N \}, \quad h = (t_f - t_0)/N, \ N \in \mathbb{N}. \]

Minimize

\[ \Gamma_N(x, y, u) := \varphi(x_N) + h \sum_{i=0}^{N-1} f_0(x_i, u_i) \]

w.r.t. \( x = (x_0, x_1, \ldots, x_N)^T \in \mathbb{R}^{(N+1)nx}, \ u = (u_0, \ldots, u_{N-1})^T \in \mathbb{R}^{nu} \) subject to the constraints

\[ \frac{x_{i+1} - x_i}{h} = f(x_i, u_i) \quad i = 0, \ldots, N - 1, \]

\[ x_0 = \bar{x}. \]
Gradient Method for the Discretized Problem

Denote by

\[ S : \mathbb{R}^{Nn_u} \rightarrow \mathbb{R}^{(N+1)n_x}, \quad u \mapsto x = S(u), \]

the solution operator, that maps the control input \( u \) to the solution \( x \) of the discrete dynamics.

Problem (Reduced Discretized Problem (RD-OCP))

\[ \text{Minimize} \quad J_N(u) := \Gamma_N(S(u), u) \quad \text{w.r.t.} \quad u \in \mathbb{R}^{Nn_u}. \]
Gradient Method for the Discretized Problem

Auxiliary functional: \((\hat{u}, \hat{x} = S(u))\) given; \(\lambda \in \mathbb{R}^{(N+1)n_x}\) to be defined later

\[
\tilde{J}_N(\hat{u}) := \varphi(\hat{x}_N) + h \sum_{i=0}^{N-1} \left( \mathcal{H}(\hat{x}_i, \hat{u}_i, \lambda_{i+1}) - \lambda_{i+1}^T \left( \frac{\hat{x}_{i+1} - \hat{x}_i}{h} \right) \right)
\]

Derivative: \((S'_i(\hat{u})\) denotes the sensitivity matrix \(\frac{\partial \hat{x}_i}{\partial u}\); exploit \(S'_0(\hat{u}) = 0\)

\[
\tilde{J}'_N(\hat{u}) = \varphi'(\hat{x}_N)S'_N(\hat{u})
\]

\[
= \varphi'(\hat{x}_N)S'_N(\hat{u}) + h \sum_{i=0}^{N-1} \left( \mathcal{H}'_x[t_i]S'_i(\hat{u}) + \mathcal{H}'_u[t_i] \frac{\partial \hat{u}_i}{\partial \hat{u}} - \lambda_{i+1}^T \left( \frac{S'_{i+1}(\hat{u}) - S'_i(\hat{u})}{h} \right) \right)
\]

\[
= \left( \varphi'(\hat{x}_N) - \lambda_N^T \right) S'_N(\hat{u}) + \left( h\mathcal{H}'_x[t_0] + \lambda_1^T \right) S'_0(\hat{u}) + h \sum_{i=1}^{N-1} \left( \mathcal{H}'_x[t_i] + \frac{1}{h} \left( \lambda_{i+1}^T - \lambda_i^T \right) \right) S'_i(\hat{u}) + h \sum_{i=0}^{N-1} \mathcal{H}'_u[t_i] \frac{\partial \hat{u}_i}{\partial \hat{u}}
\]

Avoid calculation of sensitivities \(S'_i\)! Choose \(\lambda\) such that red terms vanish.
Gradient Method for the Discretized Problem

Discrete adjoint ODE:

\[
\frac{\lambda_{i+1} - \lambda_i}{h} = -\mathcal{H}'_x(\hat{x}_i, \hat{u}_i, \lambda_{i+1}), \quad i = 0, \ldots, N - 1,
\]
\[
\lambda_N = \varphi'(\hat{x}_N)^\top
\]

Gradient of auxiliary functional:

\[
\tilde{J}'_N(\hat{u}) = h \sum_{i=0}^{N-1} \mathcal{H}_u'[t_i] \frac{\partial \hat{u}_i}{\partial \hat{u}} = h \left( \begin{array}{ccc} \mathcal{H}_u'[t_0] & \mathcal{H}_u'[t_1] & \cdots & \mathcal{H}_u'[t_{N-1}] \end{array} \right).
\]
Gradient Method for the Discretized Problem

Link to gradient of reduced objective functional $J_N$:

**Theorem**

Let $\hat{u} \in \mathbb{R}^{Nn_u}$ be given and let $\lambda \in \mathbb{R}^{(N+1)n_x}$ satisfy the discrete adjoint ODE. Then,

$$\nabla J_N(\hat{u}) = \tilde{J}_N'(\hat{u})^\top.$$  

**Consequence:**

- The gradient method for RD-OCP uses in iteration $k$ the search direction

$$d^{(k)} = -\nabla J_N(u^{(k)}) = -h \begin{pmatrix} \mathcal{H}_u'[t_0]^\top \\ \vdots \\ \mathcal{H}_u'[t_{N-1}]^\top \end{pmatrix}.$$  

- This is the same search direction as in the “function space” gradient method, except that it is scaled by $h$. Slower convergence expected!
Gradient Method for the Discretized Problem

Gradient method for RD-OCP

(0) Choose \( u^{(0)} \in \mathbb{R}^{N_{nu}} \), \( \beta \in (0, 1) \), \( \sigma \in (0, 1) \), and set \( k := 0 \).

(1) Solve
\[
\begin{align*}
\frac{x_{i+1} - x_i}{h} &= f(x_i, u^{(k)}_i) \quad (i = 0, \ldots, N - 1), \quad x_0 = \bar{x}, \\
\frac{\lambda_{i+1} - \lambda_i}{h} &= -\mathcal{H}_x'(\hat{x}_i, \hat{u}^{(k)}_i, \lambda_{i+1}) \quad (i = 0, \ldots, N - 1), \quad \lambda_N = \varphi'(\hat{x}_N)^\top
\end{align*}
\]

(2) Set
\[
\alpha^{(k)} := -\nabla J_N(u^{(k)}) = -h \begin{pmatrix}
\mathcal{H}_u'[t_0]^\top \\
\vdots \\
\mathcal{H}_u'[t_{N-1}]^\top
\end{pmatrix}
\]

(3) If \( \|\alpha^{(k)}\|_2 \approx 0 \), STOP.

(4) Perform an Armijo line-search: Find smallest \( j \in \{0, 1, 2, \ldots\} \) with
\[
J_N(u^{(k)} + \beta^j \alpha^{(k)}) \leq J_N(u^{(k)}) - \sigma \beta^j \|\alpha^{(k)}\|_2^2
\]
and set \( \alpha_k := \beta^j \).

(5) Set \( u^{(k+1)} := u^{(k)} + \alpha_k \alpha^{(k)} \), \( k := k + 1 \), and go to (1).
Gradient Method for the Discretized Problem – Example

Example (compare “function space” equivalent example)

Minimize $x_2(1)$ subject to the constraints

$$x'_1(t) = -x_1(t) + \sqrt{3}u(t), \quad x_1(0) = 2,$$
$$x'_2(t) = \frac{1}{2} \left(x_1(t)^2 + u(t)^2\right), \quad x_2(0) = 0.$$

Output of gradient method for RD-OCP: ($u^{(0)} = 0, \ N = 100, \ \beta = 0.9, \ \sigma = 0.1$)

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\alpha_k$</th>
<th>$J(u^{(k)})$</th>
<th>$|H'<em>u|</em>\infty$</th>
<th>$|H'_u|_2^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00000000E+00</td>
<td>0.87037219E+00</td>
<td>0.14877677E-01</td>
<td>0.65717185E-02</td>
</tr>
<tr>
<td>1</td>
<td>0.10000000E+01</td>
<td>0.86385155E+00</td>
<td>0.14671320E-01</td>
<td>0.63689360E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>813</td>
<td>0.10000000E+01</td>
<td>0.65728265E+00</td>
<td>0.10356781E-05</td>
<td>0.76292003E-11</td>
</tr>
<tr>
<td>814</td>
<td>0.10000000E+01</td>
<td>0.65728265E+00</td>
<td>0.10655524E-05</td>
<td>0.77345706E-11</td>
</tr>
<tr>
<td>815</td>
<td>0.10000000E+01</td>
<td>0.65728265E+00</td>
<td>0.98588398E-06</td>
<td>0.74390524E-11</td>
</tr>
</tbody>
</table>

Observation: Since the search direction in RD-OCP is scaled by $h$, we need much more iterations compared to the “function space” gradient method, which required only 26 iterations at the same accuracy.
Contents

Introduction

Necessary Conditions

Adjoint Formalism

Gradient Method
  Gradient Method in Finite Dimensions
  Gradient Method for Optimal Control Problems
  Extensions
  Gradient Method for Discrete Problem
  Examples

Lagrange-Newton Method
  Lagrange-Newton Method in Finite Dimensions
  Lagrange-Newton Method in Infinite Dimensions
  Application to Optimal Control
  Search Direction
  Examples
  Extensions
Lagrange-Newton Method

Pro’s:
▶ locally quadratic convergence rate (fast convergence)
▶ can handle nonlinear equality constraints
▶ global convergence achieved by Armijo linesearch

Con’s:
▶ requires second derivatives
▶ higher implementation effort compared to gradient method
Lagrange-Newton Method in Finite Dimensions

Consider the equality constrained nonlinear optimization problem:

**Equality constrained optimization problem (E-NLP)**

Minimize \( J(x, u) \) subject to \( H(x, u) = 0 \)

\( J : \mathbb{R}^{nx} \times \mathbb{R}^{nu} \rightarrow \mathbb{R}, \ H : \mathbb{R}^{nx} \times \mathbb{R}^{nu} \rightarrow \mathbb{R}^{nH} \) twice continuously differentiable

Lagrange function;

\[ L(x, u, \lambda) := J(x, u) + \lambda^\top H(x, u) \]

**Theorem (KKT conditions)**

Let \((\hat{x}, \hat{u})\) be a local minimum of E-NLP and let \( H'(\hat{x}, \hat{u}) \) have full rank. Then there exists a multiplier \( \lambda \in \mathbb{R}^{nH} \) such that

\[ \nabla_{(x,u)} L(\hat{x}, \hat{u}, \lambda) = 0. \]
Lagrange-Newton Method in Finite Dimensions

Idea of Lagrange-Newton method: Apply Newton’s method to the optimality system

\[ T(\hat{x}, \hat{u}, \lambda) = 0 \quad \text{with} \quad T(x, u, \lambda) := \begin{pmatrix} \nabla_{(x,u)}L(x, u, \lambda) \\ H(x, u) \end{pmatrix} = \begin{pmatrix} \nabla_x L(x, u, \lambda) \\ \nabla_u L(x, u, \lambda) \\ H(x, u) \end{pmatrix} \]

Newton system:

\[ T'(x^{(k)}, u^{(k)}, \lambda^{(k)}) d^{(k)} = -T(x^{(k)}, u^{(k)}, \lambda^{(k)}) \]

with

\[ T'(x, u, \lambda) = \begin{pmatrix} L''_{xx}(x, u, \lambda) & L''_{xu}(x, u, \lambda) & H'_x(x, u)^T \\ L''_{ux}(x, u, \lambda) & L''_{uu}(x, u, \lambda) & H'_u(x, u)^T \\ H'_x(x, u) & H'_u(x, u) & 0 \end{pmatrix} \]
Theorem (Nonsingularity)

$T'(x, u, \lambda)$ is nonsingular, if the following conditions hold:

- $L''_{(x,u),(x,u)}(x, u, \lambda)$ is positive definite on the nullspace of $H'(x, u)$, that is
  \[ v^\top L''_{(x,u),(x,u)}(x, u, \lambda)v > 0 \quad \forall v : H'(x, u)v = 0. \]

- $H'(x, u)$ has full rank

Note: These conditions are actually sufficient, if $(x, u, \lambda)$ is a stationary point of $L$. 
Lagrange-Newton Method in Infinite Dimensions

Consider the equality constrained nonlinear optimization problem:

Equality constrained optimization problem (E-NLP)

Minimize \( J(x, u) \) subject to \( H(x, u) = 0 \)

\( J : X \times U \to \mathbb{R}, \ H : X \times U \to \Lambda \) twice continuously Fréchet differentiable, \( X, U, \Lambda \) Banach spaces

Lagrange function; \( L : X \times U \times \Lambda^* \to \mathbb{R} \) with

\[
L(x, u, \lambda^*) := J(x, u) + \lambda^*(H(x, u))
\]

Theorem (KKT conditions)

Let \((\hat{x}, \hat{u})\) be a local minimum of E-NLP and let \(H'(\hat{x}, \hat{u})\) be surjective. Then there exists a multiplier \(\lambda^* \in \Lambda^*\) such that

\[
L'_x(x, u)(\hat{x}, \hat{u}, \lambda^*) = 0.
\]
Lagrange-Newton Method in Infinite Dimensions

Idea of Lagrange-Newton method:
Use Newton’s method to find a zero \( \hat{z} = (\hat{x}, \hat{u}, \hat{\lambda}^*) \in Z := X \times U \times \Lambda^* \) of the operator \( T : Z \longrightarrow Y \) defined by

\[
T(x, u, \lambda^*) := \begin{pmatrix}
L'_x(x, u, \lambda^*) \\
L'_u(x, u, \lambda^*) \\
H(x, u)
\end{pmatrix}
\]

Observe:

\[
\begin{align*}
L'_x(x, u, \lambda^*)x &= J'_x(x, u)x + \lambda^*(H'_x(x, u)x) = J'_x(x, u)x + (H'_x(x, u)^* \lambda^*)x \\
L'_u(x, u, \lambda^*)u &= J'_u(x, u)u + \lambda^*(H'_u(x, u)u) = J'_u(x, u)u + (H'_u(x, u)^* \lambda^*)u
\end{align*}
\]

where

\[
H'_x(x, u)^* : Y^* \longrightarrow X^* \quad \text{and} \quad H'_u(x, u)^* : Y^* \longrightarrow U^*
\]
denote the respective adjoint operators.
Local Lagrange-Newton Method

(0) Choose \( z^{(0)} \in Z \) and set \( k := 0 \).

(1) If \( \| T(z^{(k)}) \|_Y \approx 0 \), STOP.

(2) Compute the search direction \( d^{(k)} \) from
\[
T'(z^{(k)})(d^{(k)}) = -T(z^{(k)}).
\]

(3) Set \( z^{(k+1)} := z^{(k)} + d^{(k)} \), \( k := k + 1 \), and go to (1).
Theorem (Nonsingularity)

\( T'(x, u, \lambda^*) \) is nonsingular, if the following conditions hold:

1. \( L''_{(x,u),(x,u)}(x, u, \lambda^*) \) is uniformly positive definite on the nullspace of \( H'(x, u) \), i.e. there exists \( C > 0 \) such that

\[
\mathbf{v}^\top L''_{(x,u),(x,u)}(x, u, \lambda^*) \mathbf{v} \geq C \| \mathbf{v} \|^2_{X \times U} \quad \forall \mathbf{v} \in X \times U : H'(x, u)\mathbf{v} = 0.
\]

2. \( H'(x, u) \) is surjective

**Note:** These conditions are actually sufficient, if \( (x, u, \lambda) \) is a stationary point of \( L \), see [1, Theorem 5.6], [2, Theorem 2.3].


Local Lagrange-Newton Method

Theorem (local convergence)

Let \( z_* \) be a zero of \( T \). Suppose there exist constants \( \Delta > 0 \) and \( C > 0 \) such that for every \( z \in B_\Delta(z_*) \) the derivative \( T'(z) \) is non-singular and

\[
\| T'(z)^{-1} \|_{\mathcal{L}(Y,Z)} \leq C.
\]

(a) If \( \varphi, f_0, f, \psi \) are twice continuously differentiable, then there exists \( \delta > 0 \) such that the local Lagrange-Newton method is well-defined for every \( z^{(0)} \in B_\delta(z_*) \) and the sequence \( \{ z^{(k)} \}_{k \in \mathbb{N}} \) converges superlinearly to \( z_* \) for every \( z^{(0)} \in B_\delta(z_*) \).

(b) If the second derivatives of \( \varphi, f_0, f, \psi \) are locally Lipschitz continuous, then the convergence in (a) is quadratic.

(c) If in addition to the assumption in (a), \( T(z^{(k)}) \neq 0 \) for all \( k \), then the residual values converge superlinearly:

\[
\lim_{k \to \infty} \frac{\| T(z^{(k+1)}) \|_Y}{\| T(z^{(k)}) \|_Y} = 0.
\]
Global Lagrange-Newton Method

Merit function for globalization:

\[ \gamma(z) := \frac{1}{2} \| T(z) \|_2^2 \]

Globalized Lagrange-Newton Method

(0) Choose \( z^{(0)} \in Z, \beta \in (0, 1), \sigma \in (0, 1/2) \).

(1) If \( \gamma(z^{(k)}) \approx 0 \), STOP.

(2) Compute the search direction \( d^{(k)} \) from \( T'(z^{(k)})(d^{(k)}) = -T(z^{(k)}) \).

(3) Find smallest \( j \in \{0, 1, 2, \ldots \} \) with

\[ \gamma(z^{(k)} + \beta^j d^{(k)}) \leq \gamma(z^{(k)}) + \sigma \beta^j \gamma'(z^{(k)})(d^{(k)}) \]

and set \( \alpha_k := \beta^j \).

(4) Set \( z^{(k+1)} := z^{(k)} + \alpha_k d^{(k)} \), \( k := k + 1 \), and go to (1).

Note: \( \gamma : Z \rightarrow \mathbb{R} \) is Fréchet-differentiable with

\[ \gamma'(z^{(k)})(d^{(k)}) = -2\gamma(z^{(k)}) = -\| T(z^{(k)}) \|_2^2 \]
Problem (Optimal control problem (OCP))

Given: $\mathcal{I} := [t_0, t_f]$, twice continuously differentiable functions $\varphi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \rightarrow \mathbb{R}$, $f_0 : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$, $\psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x}$.

Minimize

$$J(x, u) := \varphi(x(t_0), x(t_f)) + \int_{\mathcal{I}} f_0(x(t), u(t)) \, dt$$

with respect to $x \in X := W^{1, \infty}(\mathcal{I}, \mathbb{R}^{n_x})$ and $u \in U := L^{\infty}(\mathcal{I}, \mathbb{R}^{n_u})$ subject to the constraints

$$x'(t) = f(x(t), u(t)) \quad a.e. \ in \ \mathcal{I},$$
$$0 = \psi(x(t_0), x(t_f)).$$

Remark:
- A partially reduced approach is possible, where $x = x(\cdot; u, x_0)$ is expressed as a function of initial value $x_0$ and $u$. The constraint $\psi(x_0, x(t_f; u, x_0)) = 0$ remains. This is the function space equivalent of the direct shooting method.
Lagrange-Newton Method – Application to Optimal Control

Hamilton function:
\[ \mathcal{H}(x, u, \lambda) := f_0(x, u) + \lambda^T f(x, u) \]

Define
\[ \kappa := \varphi + \sigma^T \psi \]

Theorem (Minimum principle, KKT conditions)

Let \((x_*, u_*)\) be a local minimizer of OCP. Let \(H'(x_*, u_*)\) be surjective. Then there exist multipliers \(\lambda_* \in W^{1, \infty}(I, \mathbb{R}^{nx})\) and \(\sigma_* \in \mathbb{R}^{n\psi}\) with

\[
\begin{align*}
    x'_*(t) - f(x_*(t), u_*(t)) &= 0 \\
    \lambda'_*(t) + \mathcal{H}'_x(x_*(t), u_*(t), \lambda_*(t))^T &= 0 \\
    \psi(x_*(t_0), x_*(t_f)) &= 0 \\
    \lambda_*(t_0) + \kappa'_{x_0}(x_*(t_0), x_*(t_f), \sigma_*)^T &= 0 \\
    \lambda_*(t_f) - \kappa'_{x_f}(x_*(t_0), x_*(t_f), \sigma_*)^T &= 0 \\
    \mathcal{H}'_u(x_*(t), u_*(t), \lambda_*(t))^T &= 0
\end{align*}
\]

Root finding problem:
\[ T(z_*) = 0, \quad T : Z \rightarrow Y \]
Lagrange-Newton Method – Application to Optimal Control

Definition of operator $T$:

$$T(z)(\cdot) := \begin{pmatrix} x'(\cdot) - f(x(\cdot), u(\cdot)) \\ x'(\cdot) + H'_x(x(\cdot), u(\cdot), \lambda(\cdot))^T \\ \psi(x(t_0), x(t_f)) \\ \lambda(t_0) + \kappa'_{x_0}(x(t_0), x(t_f), \sigma)^T \\ \lambda(t_f) - \kappa'_{x_f}(x(t_0), x(t_f), \sigma)^T \\ H'_u(x(\cdot), u(\cdot), \lambda(\cdot))^T \end{pmatrix}$$

with

$$z := (x, u, \lambda, \sigma)$$

$$Z := X \times U \times X \times \mathbb{R}^{n_{\psi}}$$

$$Y := L^\infty(\mathcal{I}, \mathbb{R}^{nx}) \times L^\infty(\mathcal{I}, \mathbb{R}^{nx}) \times \mathbb{R}^{n_{\psi}} \times \mathbb{R}^{nx} \times \mathbb{R}^{nx} \times L^\infty(\mathcal{I}, \mathbb{R}^{nu})$$
Lagrange-Newton Method – Computation of Search Direction

Newton direction:

\[ T'(z^{(k)})d = -T(z^{(k)}), \quad d = (x, u, \lambda, \sigma) \in Z \]

Fréchet derivative: (evaluated at \( z^{(k)} \))

\[
T'(z^{(k)})(z) = \begin{pmatrix}
    x' - f'_x x - f'_u u \\
    \lambda' + H''_{xx} x + H''_{xu} u + H''_{x\lambda} \lambda \\
    \psi'_0 x(t_0) + \psi'_f x(t_f) \\
    \lambda(t_0) + \kappa''_{x_0x_0} x(t_0) + \kappa''_{x_0x_f} x(t_f) + \kappa''_{x_0\sigma} \sigma \\
    \lambda(t_f) - \kappa''_{x_fx_0} x(t_0) - \kappa''_{x_fx_f} x(t_f) - \kappa''_{x_f\sigma} \sigma \\
    H''_{ux} x + H''_{uu} u + H''_{u\lambda} \lambda
\end{pmatrix}
\]
Newton direction is equivalent to the linear DAE boundary value problem

\[
\begin{pmatrix}
    x' \\
    \lambda' \\
    0
\end{pmatrix} - \begin{pmatrix}
    f'_x & 0 & f'_u \\
    -\mathcal{H}''_{xx} & -\mathcal{H}''_{x\lambda} & -\mathcal{H}''_{xu} \\
    -\mathcal{H}''_{ux} & -\mathcal{H}''_{u\lambda} & -\mathcal{H}''_{uu}
\end{pmatrix} \begin{pmatrix}
    x \\
    \lambda \\
    u
\end{pmatrix} = - \begin{pmatrix}
    (x^{(k)})' - f \\
    (\lambda^{(k)})' + (\mathcal{H}'_x)^T \\
    (\mathcal{H}'_u)^T
\end{pmatrix}
\]

with boundary conditions

\[
\begin{pmatrix}
    \psi'_{x_0} & 0 & 0 \\
    \kappa''_{x_0 x_0} & Id & \kappa''_{x_0 \sigma} \\
    -\kappa''_{x_f x_0} & 0 & -\kappa''_{x_f \sigma}
\end{pmatrix} \begin{pmatrix}
    x(0) \\
    \lambda(t_0) \\
    \sigma
\end{pmatrix} + \begin{pmatrix}
    \psi'_{x_f} & 0 & 0 \\
    \kappa''_{x_0 x_f} & 0 & 0 \\
    -\kappa''_{x_f x_f} & Id & 0
\end{pmatrix} \begin{pmatrix}
    x(1) \\
    \lambda(t_f) \\
    \sigma
\end{pmatrix} = - \begin{pmatrix}
    \psi \\
    \lambda^{(k)}(t_0) + (\kappa'_{x_0})^T \\
    \lambda^{(k)}(t_f) - (\kappa'_{x_f})^T
\end{pmatrix}.
\]

Hence: In each Newton iteration, we need to solve the above linear BVP.
Theorem

The differential-algebraic equation (DAE) in BVP has index one (i.e. the last equality can be solved for $u$), if the matrix function

$$M(t) := \mathcal{H}''_{uu}[t]$$

is non-singular for almost every $t \in [0, 1]$ and $\|M(t)^{-1}\| \leq C$ for some constant $C$ and almost every $t \in [0, 1]$.

If $M(\cdot)$ is singular:

- BVP contains a differential-algebraic equation of higher index, which is numerically unstable.
- boundary conditions may become infeasible
Example (Trolley)

\[(x_1, z)\]

\[\ell\]

\[-m_2g\]
Example (Trolley, continued)

Dynamics:

\[
\begin{align*}
x_1' &= x_3 \\
x_2' &= x_4 \\
x_3' &= \frac{m_2 \ell^3 \sin(x_2)x_4^2 - m_2 \ell^2 u + m_2 l_y \ell x_4^2 \sin(x_2) - l_y u}{-m_1 m_2 \ell^2 - m_1 l_y - m_2^2 \ell^2 - m_2 l_y + m_2^2 \ell^2 \cos(x_2)^2} \\
&\quad + \frac{m_2 \ell^2 g \cos(x_2) \sin(x_2)}{-m_1 m_2 \ell^2 - m_1 l_y - m_2^2 \ell^2 - m_2 l_y + m_2^2 \ell^2 \cos(x_2)^2} \\
x_4' &= \frac{m_2 \ell \left(m_2 \ell \cos(x_2)x_4^2 \sin(x_2) - \cos(x_2)u + g \sin(x_2)(m_1 + m_2)\right)}{-m_1 m_2 \ell^2 - m_1 l_y - m_2^2 \ell^2 - m_2 l_y + m_2^2 \ell^2 \cos(x_2)^2}
\end{align*}
\]

Parameters:

\[g = 9.81, \ m_1 = 0.3, \ m_2 = 0.5, \ \ell = 0.75, \ r = 0.1, \ l_y = 0.002.\]
Example (Trolley, continued)

Minimize

\[ \frac{1}{2} \int_0^{t_f} u(t)^2 + 5x_4(t)^2 \, dt \]

subject to the ODE, the initial conditions

\[ x_1(0) = x_2(0) = x_3(0) = x_4(0) = 0, \]

and the terminal conditions

\[ x_1(t_f) = 1, \quad x_2(t_f) = x_3(t_f) = x_4(t_f) = 0 \]

within the fixed time \( t_f = 2.7 \).
Example (Trolley, continued)

Output of Lagrange-Newton method: \( (N = 1000, \text{Euler}) \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \alpha_k )</th>
<th>( | T(z^{(k)}) |_2^2 )</th>
<th>( | \sigma^{(k)} |_\infty )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000000E+00</td>
<td>0.100000E+01</td>
<td>0.451981E+01</td>
</tr>
<tr>
<td>1</td>
<td>0.100000E+01</td>
<td>0.688773E-03</td>
<td>0.473501E-02</td>
</tr>
<tr>
<td>2</td>
<td>0.100000E+01</td>
<td>0.809983E-12</td>
<td>0.118366E-06</td>
</tr>
<tr>
<td>3</td>
<td>0.100000E+01</td>
<td>0.160897E-24</td>
<td>0.141058E-11</td>
</tr>
</tbody>
</table>

Iterations for different stepsizes \( N \): mesh independence; linear CPU

<table>
<thead>
<tr>
<th>( N )</th>
<th>CPU time [s]</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.022</td>
<td>3</td>
</tr>
<tr>
<td>200</td>
<td>0.050</td>
<td>3</td>
</tr>
<tr>
<td>400</td>
<td>0.093</td>
<td>3</td>
</tr>
<tr>
<td>800</td>
<td>0.174</td>
<td>3</td>
</tr>
<tr>
<td>1600</td>
<td>0.622</td>
<td>3</td>
</tr>
<tr>
<td>3200</td>
<td>0.822</td>
<td>3</td>
</tr>
<tr>
<td>6400</td>
<td>1.900</td>
<td>4</td>
</tr>
<tr>
<td>12800</td>
<td>3.771</td>
<td>4</td>
</tr>
<tr>
<td>25600</td>
<td>7.939</td>
<td>4</td>
</tr>
</tbody>
</table>
Lagrange-Newton Method – Examples

Example (Trolley, continued)
Lagrange-Newton Method – Example from Chemical Engineering

**DAE index-1 optimal control problem:** (chemical reaction of substances $A$, $B$, $C$, and $D$)

Minimize

$$-M_C(t_f) + 10^{-2} \int_0^{t_f} F_B(t)^2 + Q(t)^2 \, dt$$

w.r.t. controls $F_B$ (feed rate of substance $B$) and cooling power $Q$ subject to the index-1 DAE

\[
\begin{align*}
M_A' &= -V \cdot A_1 \cdot e^{-E_1/\TR} \cdot C_A \cdot C_B \\
M_B' &= F_B - V \left( A_1 e^{-E_1/\TR} \cdot C_A \cdot C_B + A_2 \cdot e^{-E_2/\TR} \cdot C_B \cdot C_C \right) \\
M_C' &= V \left( A_1 e^{-E_1/\TR} \cdot C_A \cdot C_B - A_2 \cdot e^{-E_2/\TR} \cdot C_B \cdot C_C \right) \\
M_D' &= V \cdot A_2 \cdot e^{-E_2/\TR} \cdot C_B \cdot C_C \\
H' &= 20F_B - Q - V \left( -A_1 e^{-E_1/\TR} \cdot C_A \cdot C_B - 75A_2 \cdot e^{-E_2/\TR} \cdot C_B \cdot C_C \right) \\
0 &= H - \sum_{i=A,B,C,D} M_i \left( \alpha_i (T_R - T_{ref}) + \frac{\beta_i}{2} \left( T_R^2 - T_{ref}^2 \right) \right)
\end{align*}
\]

where

\[
V = \sum_{i=A,B,C,D} \frac{M_i}{\rho_i}, \quad C_i = M_i / V, \quad i = A, B, C, D.
\]

Lagrange-Newton Method – Example from Chemical Engineering

Lagrange-Newton method:
\((N = 20000\text{ intervals})\)

<table>
<thead>
<tr>
<th>(k)</th>
<th>(\alpha_k)</th>
<th>(|T(z^{(k)})|_2^2)</th>
<th>(|a^{(k)}|_\infty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000000E+00</td>
<td>0.465186E+12</td>
<td>0.599335E+05</td>
</tr>
<tr>
<td>1</td>
<td>0.100000E+01</td>
<td>0.759821E+10</td>
<td>0.523150E+07</td>
</tr>
<tr>
<td>2</td>
<td>0.147809E-01</td>
<td>0.755228E+10</td>
<td>0.127262E+07</td>
</tr>
<tr>
<td>3</td>
<td>0.423912E-01</td>
<td>0.745716E+10</td>
<td>0.835212E+05</td>
</tr>
<tr>
<td>4</td>
<td>0.100000E+01</td>
<td>0.351002E+09</td>
<td>0.344908E+06</td>
</tr>
<tr>
<td>5</td>
<td>0.121577E+00</td>
<td>0.340325E+09</td>
<td>0.667305E+05</td>
</tr>
<tr>
<td>6</td>
<td>0.100000E+01</td>
<td>0.131555E+08</td>
<td>0.370395E+05</td>
</tr>
<tr>
<td>7</td>
<td>0.100000E+01</td>
<td>0.295389E+07</td>
<td>0.169245E+04</td>
</tr>
<tr>
<td>8</td>
<td>0.100000E+01</td>
<td>0.114958E+01</td>
<td>0.336606E+00</td>
</tr>
<tr>
<td>9</td>
<td>0.100000E+01</td>
<td>0.852576E-11</td>
<td>0.104227E-02</td>
</tr>
<tr>
<td>10</td>
<td>0.100000E+01</td>
<td>0.125527E-11</td>
<td>0.658317E-03</td>
</tr>
<tr>
<td>11</td>
<td>0.100000E+01</td>
<td>0.283055E-14</td>
<td>0.605453E-05</td>
</tr>
</tbody>
</table>
Lagrange-Newton Method – Example from Chemical Engineering
Numerical Optimal Control – Part 3: Function space methods – Matthias Gerdts

Lagrange-Newton Method – Example from Chemical Engineering

Numerical optimization methods are often used to solve optimal control problems. One such method is the Lagrange-Newton method, which combines the ideas of the augmented Lagrange method and Newton's method to solve constrained optimization problems. The method iteratively updates the control variables and adjoint variables to minimize the cost functional while satisfying the constraints.

In this example from chemical engineering, we consider a system described by the following equations:

\[ \dot{x}(t) = f(x(t), u(t)) \quad \text{and} \quad J(u) = \int_{t_0}^{t_f} L(x(t), u(t)) \, dt \]

where \( x(t) \) is the state vector, \( u(t) \) is the control vector, \( f \) is the system dynamics, \( L \) is the Lagrangian, and \( J \) is the cost functional to be minimized.

The Lagrange multiplier equations are given by:

\[ \dot{\lambda}_i(t) = \frac{\partial L}{\partial x_i(t)} \quad \text{and} \quad \dot{u}(t) = -\frac{\partial L}{\partial u(t)} + \sum_{i=1}^m \lambda_i(t) \frac{\partial f_i(x, u)}{\partial u} \]

where \( \lambda_i(t) \) are the adjoint variables.

The augmented Lagrangian is defined as:

\[ L_a(x, u, \lambda, \rho) = L(x, u) + \frac{\rho}{2} \sum_{i=1}^m \lambda_i^2(t) \]

where \( \rho \) is the penalty parameter.

The Lagrange-Newton method iteratively updates the control and adjoint variables by solving the following system:

\[ \dot{x}(t) = f(x(t), u(t)) \quad \text{and} \quad \dot{u}(t) = -\frac{\partial L}{\partial u(t)} + \sum_{i=1}^m \lambda_i(t) \frac{\partial f_i(x, u)}{\partial u} \]

The adjoint equations are solved simultaneously with the state equations, and the control variables are updated until convergence is achieved.

In the example shown, we consider a simplified chemical engineering problem with five state variables and five adjoint variables. The state variables \( x_1(t), \ldots, x_5(t) \) represent the system states, and the adjoint variables \( \lambda_1(t), \ldots, \lambda_5(t) \) represent the Lagrange multipliers.

The plots illustrate the time evolution of the state variables and adjoint variables over a time interval of 20 seconds. The graphs show the convergence of the solution as the iterations proceed, indicating successful optimization.
Lagrange-Newton Method – Navier-Stokes Example

Optimal control problem:

Minimize

\[ \frac{1}{2} \int_Q \|y(t, x) - y_d(t, x)\|^2 \, dx \, dt + \frac{\delta}{2} \int_Q \|u(t, x)\|^2 \, dx \, dt \]

w.r.t. velocity \( y \), pressure \( p \) and control \( u \) subject to the 2D Navier-Stokes equations

\[
y_t = \frac{1}{\text{Re}} \Delta y - (y \cdot \nabla)y - \nabla p + u \quad \text{in} \ Q := (0, t_f) \times \Omega, \]

\[ 0 = \text{div}(y) \quad \text{in} \ Q, \]

\[ 0 = y(0, x) \quad \text{for} \ x \in \Omega := (0, 1) \times (0, 1), \]

\[ 0 = y(t, x) \quad \text{for} \ (t, x) \in (0, t_f) \times \partial \Omega. \]

Given: desired velocity field

\[
y_d(t, x_1, x_2) = (-q(t, x_1)q'_{x_2}(t, x_2), q(t, x_2)q'_{x_1}(t, x_1))^\top, \]

\[ q(t, z) = (1 - z)^2(1 - \cos(2\pi z t)) \]

Lagrange-Newton Method – Navier-Stokes Example

Discretization by method of lines: (details omitted)

Minimize
\[
\frac{1}{2} \int_{0}^{t_f} \| y_h(t) - y_{d,h}(t) \|^2 dt + \frac{\delta}{2} \int_{0}^{t_f} \| u_h(t) \|^2 dt
\]
subject to the index-2 DAE

\[
\dot{y}_h(t) = \frac{1}{\text{Re}} A_h y_h(t) - \frac{1}{2} \begin{pmatrix}
\quad y_h(t)^\top H_{h,1} y_h(t) \\
\vdots \\
\quad y_h(t)^\top H_{h,2(N-1)^2} y_h(t)
\end{pmatrix} - B_h p_h(t) + u_h(t),
\]

\[
0 = B_h^\top y_h(t),
\]

\[
y_h(0) = 0.
\]
Lagrange-Newton Method – Navier-Stokes Example

Pressure $p$ at $t = 0.6$, $t = 1.0$, $t = 1.4$ and $t = 1.967$:

(Parameters: $t_f = 2$, $\delta = 10^{-5}$, $Re = 1$, $N = 31$, $N_t = 60$, $n_x = 2(N - 1)^2 = 1800$, $n_y = (N - 1)^2 = 900$, $n_u = 1800$ controls)
Desired flow (left), controlled flow (middle), control (right) at $t = 0.6$, $t = 1.0$, $t = 1.4$ and $t = 1.967$: 
Lagrange-Newton Method – Navier-Stokes Example

Output of Lagrange-Newton method:

Solve Stokes problem as initial guess:

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\int_0^1 f_0[t] , dt$</th>
<th>$\alpha_{k-1}$</th>
<th>$| T(z^{(k)}) |_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$1.763432802e+03$</td>
<td></td>
<td>$5.938741958e+01$</td>
</tr>
<tr>
<td>1</td>
<td>$3.986109778e+02$</td>
<td>$1.0000000000000$</td>
<td>$7.255927209e-10$</td>
</tr>
</tbody>
</table>

Solve Navier-Stokes problem:

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\int_0^1 f_0[t] , dt$</th>
<th>$\alpha_{k-1}$</th>
<th>$| T(z^{(k)}) |_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$3.986109778e+02$</td>
<td></td>
<td>$1.879632471e+04$</td>
</tr>
<tr>
<td>1</td>
<td>$3.988553051e+02$</td>
<td>$1.0000000000000$</td>
<td>$1.183777963e+01$</td>
</tr>
<tr>
<td>2</td>
<td>$3.988549264e+02$</td>
<td>$1.0000000000000$</td>
<td>$4.521653291e-04$</td>
</tr>
<tr>
<td>3</td>
<td>$3.988549264e+02$</td>
<td>$1.0000000000000$</td>
<td>$9.576226032e-10$</td>
</tr>
</tbody>
</table>
Lagrange-Newton Method – Extensions

Treatment of inequality constraints:

▶ sequential-quadratic programming (SQP)
  Idea: solve linear-quadratic optimization problems to obtain a search direction

▶ interior-point methods (IP)
  Idea: solve a sequence of barrier problems (equivalent: perturbation of complementarity conditions in KKT conditions)

▶ semismooth Newton methods
  Idea: transform complementarity conditions into equivalent (nonsmooth!) equation
References

The Lagrange-Newton method for infinite dimensional optimization problems.

Sequential Quadratic Programming in Banach Spaces.

The Lagrange-Newton method for nonlinear optimal control problems.

The Lagrange-Newton method for state constrained optimal control problems.

"Optimal Control of ODEs and DAEs".

"Optimization with PDE constraints".

"Lagrange multiplier approach to variational problems and applications".

Numerical Solution of Optimal Control Problems with State Constraints by Sequential Quadratic Programming in Function Space.

"Optimale Steuerung partieller Differentialgleichungen".
Resources

**Software**: (available for registered users; free for academic use)

- **OCPID-DAE1** (optimal control and parameter identification with differential-algebraic equations of index 1):
  
  http://www.optimal-control.de

- **sqpfiltertoolbox**: SQP method for dense NLPs
  
  http://www.optimal-control.de

- **WORHP** (Büskens/Gerdts): SQP method for sparse large-scale NLPs
  
  http://www.worhp.de

- **QPSOL**: interior-point and nonsmooth Newton methods for sparse large-scale convex quadratic programs
  
  available upon request

**Robotics lab at UniBw M**: research stays with use of lab equipment upon request

- **KUKA youBot robot** (2 arm robot on a platform); 3 scale rc cars; LEGO Mindstorms robots; quarter car testbench; quadcopter
More Resources

**Further optimal control software:**
- CasADi, ACADO: M. Diehl et al.; http://casadi.org; http://sourceforge.net/p/acado/
- NUOCCCS: C. Büskens, University of Bremen
- MUSCOD--II: H.G. Bock et al., IWR Heidelberg; http://www.iwr.uni-heidelberg.de/~agbock/RESEARCH/muscod.php
- MISER: K.L. Teo et al., Curtin University, Perth; http://school.maths.uwa.edu.au/les/miser/
- PSOPT: http://www.psopt.org/
- ...

**Further optimization software:**
- KNITRO (sparse large-scale problems): Ziena Optimization; http://www.ziena.com/knitro.htm
- IPOPT (sparse large-scale problems): A. Wächter; https://projects.coin-or.org/Ipopt
- filterSQP: R. Fletcher, S. Leyffer; http://www.mcs.anl.gov/leyffer/solvers.html
- ooQP: M. Gertz, S. Wright; http://pages.cs.wisc.edu/swright/ooqp/
- ...

**Software for boundary value problems:**
- BOUNDSCO: H. J. Oberle, University of Hamburg; http://www.math.uni-hamburg.de/home/oberle/software.html
- COLDAE: U. Ascher; www.cs.ubc.ca/ ascher/coldae.f
- ...

**Links:**
- Decision Tree for Optimization Software; http://plato.la.asu.edu/guide.html
- CUTER: large collection of optimization test problems; http://www.cuter.rl.ac.uk/
- COPS: large-scale optimization test problems; http://www.mcs.anl.gov/~more/cops/
- MINTOC: testcases for mixed-integer optimal control; http://mintoc.de/
- ...

Numerical Optimal Control – Part 3: Function space methods –
Matthias Gerdts
Announcement: youBot Robotics Hackathon

Description:

- student programming contest
- addresses students and PhD’s who like to realize projects with the KUKA youBot robot
- 12 participants from 5 Universities (UniBw M, Bayreuth, TUM, TU Berlin, Maastricht)
Thanks for your Attention!

Questions?

Further information:

matthias.gerdts@unibw.de
www.unibw.de/irt1/gerdts
www.optimal-control.de

Fotos: http://de.wikipedia.org/wiki/München
Magnus Manske (Panorama), Luidger (Theatinerkirche), Kurmis (Chin. Turm), Arad Mojtahedi (Olympiapark), Max-k (Deutsches Museum), Oliver Raupach (Friedensengel), Andreas Praefcke (Nationaltheater)