An Introduction to Optimal Control and Recent Solution Strategies

Lars Grüne

Parts of the slides were written together with Timm Faulwasser, TU Hamburg

Mathematical Institute, University of Bayreuth, Germany

Università degli Studi di Padova, Italy, 9–12 September 2025

Download of Slides

The latest version of these slides is available from https://t1p.de/optimal_control_padova

The moodle page for this course is https://stem.elearning.unipd.it/enrol/index.php?id=13066

This version of the slides is from 10 September 2025

Outline of the course

Part 1: Optimal Control Problems — An Introduction

Part 2: Solution Concepts

Part 3: Numerical Solution Methods

Part 4: Model Predictive Control

Part 5: Deep Neural Networks for High-Dimensional Problems

Outline of the course

Part 1: Optimal Control Problems — An Introduction

Part 2: Solution Concepts

Part 3: Numerical Solution Methods

Part 4: Model Predictive Control

Part 5: Deep Neural Networks for High-Dimensional Problems

Purpose of this part

Introduce control systems and optimal control problems

Purpose of this part

Introduce control systems and optimal control problems

Explain the ingredients of an optimal control problem

Purpose of this part

Introduce control systems and optimal control problems

Explain the ingredients of an optimal control problem

Explain the difference between open-loop and closed-loop optimal control

Purpose of this part

Introduce control systems and optimal control problems

Explain the ingredients of an optimal control problem

Explain the difference between open-loop and closed-loop optimal control

Contents of this part

Part 1: Optimal Control Problems — An Introduction

- Optimal Control Problems
- Continuous-time Optimal Control Problems
- Discrete-time Optimal Control Problems
- Open-loop and Closed-loop Optimal Control

Optimal Control Problems

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(t, x(t), \mathbf{u}(t))$$

or in discrete time

$$x^+(t) := x(t+1) = g(t, x(t), \mathbf{u}(t))$$

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0, \ t \geq t_0$$

or in discrete time

$$x^+(t) := x(t+1) = g(t, x(t), \mathbf{u}(t))$$

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0, \ t \geq t_0$$

or in discrete time

$$x^+(t) := x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0, \ t = t_0, t_0+1, t_0+2, \ldots$$

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0,\ t \geq t_0$$

or in discrete time

$$x^+(t) := x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0,\ t = t_0, t_0+1, t_0+2, \ldots$$

where $f\colon \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, $g\colon \mathbb{Z} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are maps with appropriate regularity and $\mathbf{u}(\cdot)$ is either a suitable control function or a control sequence, in both cases with values in $\mathbb{R}^m$

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0,\ t \geq t_0$$

or in discrete time

$$x^+(t) := x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0,\ t = t_0, t_0+1, t_0+2, \ldots$$

where $f\colon \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, $g\colon \mathbb{Z} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are maps with appropriate regularity and $\mathbf{u}(\cdot)$ is either a suitable control function or a control sequence, in both cases with values in $\mathbb{R}^m$

We call $x \in \mathbb{R}^n$ the state and $u \in \mathbb{R}^m$ the control input

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0,\ t \geq t_0$$

or in discrete time

$$x^+(t) := x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0,\ t = t_0, t_0+1, t_0+2, \ldots$$

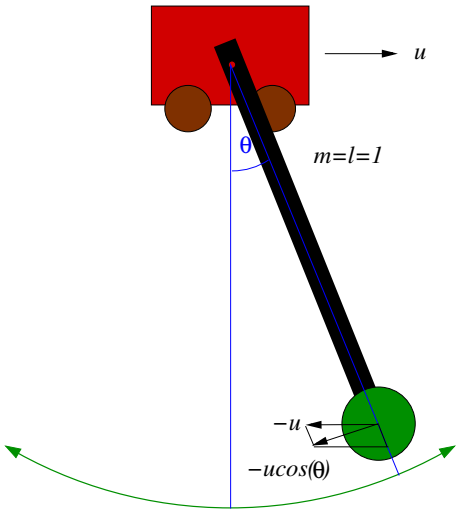
where $f\colon \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, $g\colon \mathbb{Z} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are maps with appropriate regularity and $\mathbf{u}(\cdot)$ is either a suitable control function or a control sequence, in both cases with values in $\mathbb{R}^m$

We call $x \in \mathbb{R}^n$ the state and $u \in \mathbb{R}^m$ the control input

The bold $\mathbf{u}(\cdot)$ indicates control functions or sequences with $\mathbf{u}(t) = u \in U$

Example: Pendulum on a cart

Example: Pendulum on a cart



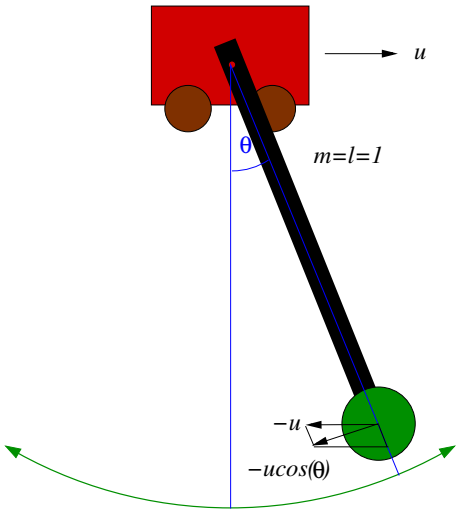
$x_1 = \theta =$ angle
$x_2 =$ angular velocity
$x_3 =$ cart position
$x_4 =$ cart velocity
$u \;\; =$ cart acceleration

$\rightsquigarrow$ control system

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -kx_2 - g\sin(x_1) - u\cos(x_1) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= u
\end{aligned}$$

Example: Pendulum on a cart

Example: Pendulum on a cart



$x_1 = \theta =$ angle
$x_2 =$ angular velocity
$x_3 =$ cart position
$x_4 =$ cart velocity
$u \;\; =$ cart acceleration

$\rightsquigarrow$ control system

$$
\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -k_1 x_2 - k_2(g\sin(x_1) - u\cos(x_1)) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= x_5 \\
\dot{x}_5 &= x_6 \\
\dot{x}_6 &= k_3 u(t) - k_4 x_6(t) - k_5 x_5
\end{aligned}
$$

What is an Optimal Control Problem?

Informal definition of optimal control:

Determine a control function that causes a control system to minimize a performance criterion and—at the same time— satisfy physical constraints.

What is an Optimal Control Problem?

Informal definition of optimal control:

Determine a control function that causes a control system to minimize a performance criterion and—at the same time— satisfy physical constraints.

Generic Optimal Control Problem (OCP) – Continuous time:

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to (OCP$_c$)

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0 \\ \forall\, t &\in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m \\ \forall\, t &\in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n \end{aligned}$$

Optimal control in continuous time and discrete time

Continuous-time OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Optimal control in continuous time and discrete time

Continuous-time OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Discrete-time OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in \mathbb{N}_{[t_0, t_0+N-1]} : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in \mathbb{N}_{[t_0, t_0+N]} : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Optimal control in continuous time and discrete time

Continuous-time OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Discrete-time OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in \mathbb{N}_{[t_0, t_0+N-1]} : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in \mathbb{N}_{[t_0, t_0+N]} : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Ingredients

- Dynamics?
- Class of control functions? Definition of state and input constraints?
- Performance criterion?

Continuous-Time Optimal Control Problems

Continuous-time optimal control problems

Generic Optimal Control Problem (OCP):

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to (OCP$_c$)

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0 \\ \forall t &\in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m \\ \forall t &\in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n \end{aligned}$$

Ingredients

- Dynamics?
- Class of control functions? Definition of state and input constraints?
- Performance criterion?

Control functions and dynamics

We want to ensure the existence of a unique solution of the initial value problem

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$

Control functions and dynamics

We want to ensure the existence of a unique solution of the initial value problem

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$

If

- f is Lipschitz continuous in x uniformly for u and t from bounded sets
- $\mathbf{u}$ is measurable and locally essentially bounded, i.e., $\mathbf{u} \in L_\infty^{loc}$

Control functions and dynamics

We want to ensure the existence of a unique solution of the initial value problem

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$

If

- f is Lipschitz continuous in x uniformly for u and t from bounded sets
- $\mathbf{u}$ is measurable and locally essentially bounded, i.e., $\mathbf{u} \in L_\infty^{loc}$

then the Theorem of Carathéodory guarantees the existence of a unique solution

Control functions and dynamics

We want to ensure the existence of a unique solution of the initial value problem

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$

If

- f is Lipschitz continuous in x uniformly for u and t from bounded sets
- $\mathbf{u}$ is measurable and locally essentially bounded, i.e., $\mathbf{u} \in L_\infty^{loc}$

then the Theorem of Carathéodory guarantees the existence of a unique solution, at least on some interval $[t_0, t_{\max})$

Control functions and dynamics

We want to ensure the existence of a unique solution of the initial value problem

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$

If

- f is Lipschitz continuous in x uniformly for u and t from bounded sets
- $\mathbf{u}$ is measurable and locally essentially bounded, i.e., $\mathbf{u} \in L_\infty^{loc}$

then the Theorem of Carathéodory guarantees the existence of a unique solution, at least on some interval $[t_0, t_{\max})$

We denote this solution by

$$x_{\mathbf{u}}(t, t_0, x_0)$$

Control functions and dynamics

We want to ensure the existence of a unique solution of the initial value problem

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$

If

- f is Lipschitz continuous in x uniformly for u and t from bounded sets
- $\mathbf{u}$ is measurable and locally essentially bounded, i.e., $\mathbf{u} \in L_\infty^{loc}$

then the Theorem of Carathéodory guarantees the existence of a unique solution, at least on some interval $[t_0, t_{\max})$

We denote this solution by

$$x_{\mathbf{u}}(t, t_0, x_0)$$

(we may omit t_0, x_0, and/or $\mathbf{u}$ when clear from the context)

Performance criteria

Cost functional $J : \mathbb{R}^n \times L_\infty^{loc} \to \mathbb{R}$ in Lagrange form

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) dt$$

- Stage cost or running cost[1] $\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, continuous and continuously differentiable w.r.t. x
- t_0, t_f can be fixed or free, i.e., we may minimize not only over $\mathbf{u}$ but also over t_0 and/or t_f
- $t_f = \infty$ is possible; we will come back to this later

[1] In older texts also called Lagrange function

Performance criteria

- Cost functional $J : \mathbb{R}^n \times L^{loc}_\infty \to \mathbb{R}$ in Lagrange form

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) dt$$

  Stage cost $\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$

Performance criteria

- Cost functional $J : \mathbb{R}^n \times L^{loc}_\infty \to \mathbb{R}$ in Lagrange form

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) dt$$

  Stage cost $\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$

- Cost functional $J : \mathbb{R}^n \times L^{loc}_\infty \to \mathbb{R}$ in Mayer form

$$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

  Initial and terminal cost $L : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$

Performance criteria

- Cost functional $J : \mathbb{R}^n \times L^{loc}_\infty \to \mathbb{R}$ in Lagrange form

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) dt$$

  Stage cost $\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$

- Cost functional $J : \mathbb{R}^n \times L^{loc}_\infty \to \mathbb{R}$ in Mayer form

$$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

  Initial and terminal cost $L : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$

- Cost functional $J : \mathbb{R}^n \times L^{loc}_\infty \to \mathbb{R}$ in Bolza form

$$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f)) + \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) dt$$

Conversion of criteria

Lagrange form: $$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Mayer form: $$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

Conversion of criteria

Lagrange form: $$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Mayer form: $$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

The criteria can be converted into each other

Conversion of criteria

Lagrange form: $$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Mayer form: $$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

The criteria can be converted into each other, e.g. from Lagrange to Mayer:

Conversion of criteria

Lagrange form: $$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Mayer form: $$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

The criteria can be converted into each other, e.g. from Lagrange to Mayer:

Add an additional state c with differential equation $\dot{c}(t) = \ell(t, x(t), \mathbf{u}(t))$

Conversion of criteria

Lagrange form: $$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Mayer form: $$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

The criteria can be converted into each other, e.g. from Lagrange to Mayer:

Add an additional state c with differential equation $\dot{c}(t) = \ell(t, x(t), \mathbf{u}(t))$

Then we get

$$c(t_f) = c_0 + \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Conversion of criteria

Lagrange form: $$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Mayer form: $$J(x_0, \mathbf{u}(\cdot)) = L(t_0, x(t_0), t_f, x(t_f))$$

The criteria can be converted into each other, e.g. from Lagrange to Mayer:

Add an additional state c with differential equation $\dot{c}(t) = \ell(t, x(t), \mathbf{u}(t))$

Then we get

$$c(t_f) = c_0 + \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Thus, for $\tilde{x} = \begin{pmatrix} c \\ x \end{pmatrix}$, $\tilde{x}_0 = \begin{pmatrix} 0 \\ x_0 \end{pmatrix}$ and $L(t_0, \tilde{x}(t_0), t_f, \tilde{x}(t_f)) = c(t_f)$ we obtain

$$L(t_0, \tilde{x}(t_0), t_f, \tilde{x}(t_f)) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

Constraints

In this course, we limit ourselves to pointwise state and control constraints

$$x(t) \in \mathbb{X}, \quad \mathbf{u}(t) \in \mathbb{U} \quad \text{for (almost) all } t \in [t_0, t_f]$$

for given sets $\mathbb{X} \subseteq \mathbb{R}^n$, $\mathbb{U} \subseteq \mathbb{R}^m$

Constraints

In this course, we limit ourselves to pointwise state and control constraints

$$x(t) \in \mathbb{X}, \quad \mathbf{u}(t) \in \mathbb{U} \quad \text{for (almost) all } t \in [t_0, t_f]$$

for given sets $\mathbb{X} \subseteq \mathbb{R}^n$, $\mathbb{U} \subseteq \mathbb{R}^m$

In an implementation, one would describe these sets by inequalities, e.g.,

$$\mathbb{X} = \{x \in \mathbb{R}^n \,|\, g_i(x) \leq 0 \text{ for all } i = 1, \ldots, n_g\}$$

for given functions $g_1, \ldots, g_{n_g} : \mathbb{R}^n \to \mathbb{R}$

Constraints

In this course, we limit ourselves to pointwise state and control constraints

$$x(t) \in \mathbb{X}, \quad \mathbf{u}(t) \in \mathbb{U} \quad \text{for (almost) all } t \in [t_0, t_f]$$

for given sets $\mathbb{X} \subseteq \mathbb{R}^n$, $\mathbb{U} \subseteq \mathbb{R}^m$

In an implementation, one would describe these sets by inequalities, e.g.,

$$\mathbb{X} = \{x \in \mathbb{R}^n \,|\, g_i(x) \leq 0 \text{ for all } i = 1, \ldots, n_g\}$$

for given functions $g_1, \ldots, g_{n_g} : \mathbb{R}^n \to \mathbb{R}$

More general constraints would be, e.g., mixed constraints such as $(x, u) \in \mathbb{Y}$ or integral constraints such as $\int_{t_0}^{t_f} h(x(t), \mathbf{u}(t))dt \leq 0$

Constraints

In this course, we limit ourselves to pointwise state and control constraints

$$x(t) \in \mathbb{X}, \quad \mathbf{u}(t) \in \mathbb{U} \quad \text{for (almost) all } t \in [t_0, t_f]$$

for given sets $\mathbb{X} \subseteq \mathbb{R}^n$, $\mathbb{U} \subseteq \mathbb{R}^m$

In an implementation, one would describe these sets by inequalities, e.g.,

$$\mathbb{X} = \{x \in \mathbb{R}^n \,|\, g_i(x) \leq 0 \text{ for all } i = 1, \ldots, n_g\}$$

for given functions $g_1, \ldots, g_{n_g} : \mathbb{R}^n \to \mathbb{R}$

More general constraints would be, e.g., mixed constraints such as $(x, u) \in \mathbb{Y}$ or integral constraints such as $\int_{t_0}^{t_f} h(x(t), \mathbf{u}(t))dt \leq 0$

Constraints can also be incorporated in a "soft" way, by choosing ℓ to satisfy

$$\ell(x, u) = \infty \quad \text{if } (x, u) \notin \mathbb{X} \times \mathbb{U}$$

In practice one often chooses ℓ to be very large instead of actually ∞

Example – Formulating an optimal control problem

The simplified dynamics of a car on a 1d road are given by

$$\dot{x} = Ax + Bu = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u, \qquad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

where x_1 is the position, x_2 is the velocity, and the control input u is the acceleration

Example – Formulating an optimal control problem

The simplified dynamics of a car on a 1d road are given by

$$\dot{x} = Ax + Bu = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u, \qquad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

where x_1 is the position, x_2 is the velocity, and the control input u is the acceleration

Question: How would you specify performance criterion and constraints of a physically meaningful optimal control problem that steers the car from $x_1 = 1$ to $x_1 = 0$ on the interval $[t_0, t_f]$?

Admissible and feasible controls

Definition: A control $\mathbf{u}(\cdot) \in L_\infty^{loc}$ is said to be feasible for initial value x_0 on $[t_0, t_f]$ if

- the solution $x_{\mathbf{u}}(\cdot; t_0, x_0)$ is defined on $[t_0, t_f]$
- $\mathbf{u}(\cdot)$ and $x_{\mathbf{u}}(\cdot; t_0, x_0)$ satisfy the constraints for almost all $t \in [t_0, t_f]$

Admissible and feasible controls

Definition: A control $\mathbf{u}(\cdot) \in L_\infty^{loc}$ is said to be feasible for initial value x_0 on $[t_0, t_f]$ if

- the solution $x_{\mathbf{u}}(\cdot; t_0, x_0)$ is defined on $[t_0, t_f]$
- $\mathbf{u}(\cdot)$ and $x_{\mathbf{u}}(\cdot; t_0, x_0)$ satisfy the constraints for almost all $t \in [t_0, t_f]$

Then, $(\mathbf{u}(\cdot), x_{\mathbf{u}}(\cdot; t_0, x_0))$ is called a feasible pair

Admissible and feasible controls

Definition: A control $\mathbf{u}(\cdot) \in L_\infty^{loc}$ is said to be feasible for initial value x_0 on $[t_0, t_f]$ if

- the solution $x_{\mathbf{u}}(\cdot; t_0, x_0)$ is defined on $[t_0, t_f]$
- $\mathbf{u}(\cdot)$ and $x_{\mathbf{u}}(\cdot; t_0, x_0)$ satisfy the constraints for almost all $t \in [t_0, t_f]$

Then, $(\mathbf{u}(\cdot), x_{\mathbf{u}}(\cdot; t_0, x_0))$ is called a feasible pair

Definition: The set

$$\mathcal{U}^{[t_0,t_f]}(x_0) \doteq \left\{\mathbf{u}(\cdot) \in L_\infty^{loc} \mid \mathbf{u}(\cdot) \text{ is feasible for initial value } x_0 \text{ on } [t_0, t_f]\right\}$$

is called the set of feasible controls. Short hand notation: $\mathcal{U}$

Infinite-horizon objectives

Often one is interested in $t_f = \infty$. Then the Lagrange term in the objective is written as

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \ell(t, x(t), \mathbf{u}(t)) dt$$

Infinite-horizon objectives

Often one is interested in $t_f = \infty$. Then the Lagrange term in the objective is written as

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \ell(t, x(t), \mathbf{u}(t))dt$$

Infinite-horizon objectives are used for tasks that last indefinitely long

Infinite-horizon objectives

Often one is interested in $t_f = \infty$. Then the Lagrange term in the objective is written as

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \ell(t, x(t), \mathbf{u}(t))dt$$

Infinite-horizon objectives are used for tasks that last indefinitely long

Sometimes, exponential discounting is considered

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \beta^t \ell(t, x(t), \mathbf{u}(t))dt$$

with $\beta^t = e^{-\delta t}$, $\delta > 0$

Infinite-horizon objectives

Often one is interested in $t_f = \infty$. Then the Lagrange term in the objective is written as

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \ell(t, x(t), \mathbf{u}(t))dt$$

Infinite-horizon objectives are used for tasks that last indefinitely long

Sometimes, exponential discounting is considered

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \beta^t \ell(t, x(t), \mathbf{u}(t))dt$$

with $\beta^t = e^{-\delta t}$, $\delta > 0$. For bounded ℓ, discounting ensures that also on infinite horizons

$$J(x_0, \mathbf{u}(\cdot)) < \infty$$

Infinite-horizon objectives

Often one is interested in $t_f = \infty$. Then the Lagrange term in the objective is written as

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \ell(t, x(t), \mathbf{u}(t))dt$$

Infinite-horizon objectives are used for tasks that last indefinitely long

Sometimes, exponential discounting is considered

$$J(x_0, \mathbf{u}(\cdot)) = \int_{t_0}^{\infty} \beta^t \ell(t, x(t), \mathbf{u}(t))dt$$

with $\beta^t = e^{-\delta t}$, $\delta > 0$. For bounded ℓ, discounting ensures that also on infinite horizons

$$J(x_0, \mathbf{u}(\cdot)) < \infty$$

Without discounting, additional conditions on the control system are needed to ensure finiteness. Typically, these are controllability conditions

Why consider infinite-horizon problems?

Of course, no real process runs infinitely long

Why consider infinite-horizon problems?

Of course, no real process runs infinitely long

Yet, many processes have no "natural" end time

Why consider infinite-horizon problems?

Of course, no real process runs infinitely long

Yet, many processes have no "natural" end time. They run until they are switched off

Why consider infinite-horizon problems?

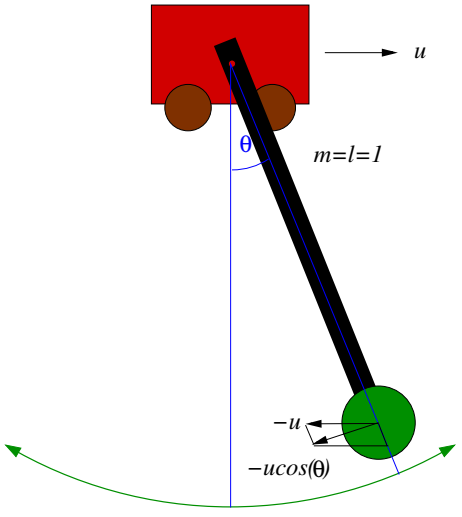
Of course, no real process runs infinitely long

Yet, many processes have no "natural" end time. They run until they are switched off

For instance, the frequent task of controlling a system to a desired state or reference solution x_{ref} and keeping it there is naturally posed as an infinite-horizon optimal control problem

Example: Pendulum on a cart

Example: Pendulum on a cart



$x_1 = \theta =$ angle
$x_2 =$ angular velocity
$x_3 =$ cart position
$x_4 =$ cart velocity
$u \;\; =$ cart acceleration

$$\ell(x,u) = \|x - x^*\|^2 + \lambda u^2$$

Swing-up to and balancing at the upright position can be achieved by an infinite-horizon optimal control with cost $\ell(x,u) = \|x - x^*\|^2 + \mu u^2$

Discrete-Time Optimal Control Problems

Performance criterion in discrete time

Recall the discrete-time dynamics $x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$

Performance criterion in discrete time

Recall the discrete-time dynamics $x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$

General (Bolza) performance criterion:

$$J(x_0, \mathbf{u}(\cdot)) = \sum_{t=t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) + L(t_0, x(t_0), t_f, x(t_f))$$

- Stage cost or running cost $\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$
- Mayer term (initial and/or terminal cost): $L : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$
- Again, t_0 and t_f can be fixed or free

Infinite-horizon OCP in discrete time

$$\min_{\mathbf{u}(\cdot)} \sum_{t=t_0}^{\infty} \beta^t \ell(t, x(t), \mathbf{u}(t))$$

subject to (OCP$_d$)

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(0) = x_0$$
$$\forall t \in \mathbb{N} : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in \mathbb{N} : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Here, again, $\beta \in (0, 1]$ is the discount factor

Infinite-horizon OCP in discrete time

$$\min_{\mathbf{u}(\cdot)} \sum_{t=t_0}^{\infty} \beta^t \ell(t, x(t), \mathbf{u}(t))$$

subject to (OCP$_d$)

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(0) = x_0$$
$$\forall t \in \mathbb{N} : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in \mathbb{N} : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Here, again, $\beta \in (0, 1]$ is the discount factor

($\beta = 1$ yields a problem without discounting)

Discretization in time

Continuous-time systems can be converted into discrete-time systems

Discretization in time

Continuous-time systems can be converted into discrete-time systems:

For discretizing the continuous-time control system $\dot{x}(t) = f(t, x(t), \mathbf{u}(t))$ in time, select a sampling time step $h > 0$ and choose a finite dimensional space of control functions that is compatible with h

Discretization in time

Continuous-time systems can be converted into discrete-time systems:

For discretizing the continuous-time control system $\dot{x}(t) = f(t, x(t), \mathbf{u}(t))$ in time, select a sampling time step $h > 0$ and choose a finite dimensional space of control functions that is compatible with h

Simplest choice: piecewise constant controls, i.e., $\mathbf{u}$ constant on each interval $[kh, (k+1)h)$, $k \in \mathbb{N}$, with value u_k

Discretization in time

Continuous-time systems can be converted into discrete-time systems:

For discretizing the continuous-time control system $\dot{x}(t) = f(t, x(t), \mathbf{u}(t))$ in time, select a sampling time step $h > 0$ and choose a finite dimensional space of control functions that is compatible with h

Simplest choice: piecewise constant controls, i.e., $\mathbf{u}$ constant on each interval $[kh, (k+1)h)$, $k \in \mathbb{N}$, with value u_k

Then, set $g(k, x, u)$ to be the solution $x_{\mathbf{u}}((k+1)h, kh, x)$ with control $\mathbf{u} \equiv u_k$

Discretization in time

Continuous-time systems can be converted into discrete-time systems:

For discretizing the continuous-time control system $\dot{x}(t) = f(t, x(t), \mathbf{u}(t))$ in time, select a sampling time step $h > 0$ and choose a finite dimensional space of control functions that is compatible with h

Simplest choice: piecewise constant controls, i.e., $\mathbf{u}$ constant on each interval $[kh, (k+1)h)$, $k \in \mathbb{N}$, with value u_k

Then, set $g(k, x, u)$ to be the solution $x_{\mathbf{u}}((k+1)h, kh, x)$ with control $\mathbf{u} \equiv u_k$. This way,

$$x(k+1) = g(k, x(k), \mathbf{u}(k))$$

exactly reproduces the continuous-time solutions at times $t = kh$, $k \in \mathbb{N}$

Discretization in time

Continuous-time systems can be converted into discrete-time systems:

For discretizing the continuous-time control system $\dot{x}(t) = f(t, x(t), \mathbf{u}(t))$ in time, select a sampling time step $h > 0$ and choose a finite dimensional space of control functions that is compatible with h

Simplest choice: piecewise constant controls, i.e., $\mathbf{u}$ constant on each interval $[kh, (k+1)h)$, $k \in \mathbb{N}$, with value u_k

Then, set $g(k, x, u)$ to be the solution $x_{\mathbf{u}}((k+1)h, kh, x)$ with control $\mathbf{u} \equiv u_k$. This way,

$$x(k+1) = g(k, x(k), \mathbf{u}(k))$$

exactly reproduces the continuous-time solutions at times $t = kh$, $k \in \mathbb{N}$

Defining the discrete-time cost ℓ as $\int_{kh}^{(k+1)h} \ell(t, x(t), u)dt$ exactly reproduces the cost

Discretization in time

Continuous-time systems can be converted into discrete-time systems:

For discretizing the continuous-time control system $\dot{x}(t) = f(t, x(t), \mathbf{u}(t))$ in time, select a sampling time step $h > 0$ and choose a finite dimensional space of control functions that is compatible with h

Simplest choice: piecewise constant controls, i.e., $\mathbf{u}$ constant on each interval $[kh, (k+1)h)$, $k \in \mathbb{N}$, with value u_k

Then, set $g(k, x, u)$ to be the solution $x_{\mathbf{u}}((k+1)h, kh, x)$ with control $\mathbf{u} \equiv u_k$. This way,

$$x(k+1) = g(k, x(k), \mathbf{u}(k))$$

exactly reproduces the continuous-time solutions at times $t = kh$, $k \in \mathbb{N}$

Defining the discrete-time cost ℓ as $\int_{kh}^{(k+1)h} \ell(t, x(t), u)dt$ exactly reproduces the cost

Note: This does not yet include any numerical computation of $x(h)$ and $\int_{kh}^{(k+1)h} \ell(t, x(t), u)dt$, which may be needed in addition

Open-loop and closed-loop (or feedback) optimal control

Optimal value function and controls

The function

$$V(x_0) := \inf_{\mathbf{u}(\cdot) \in \mathcal{U}^{[t_0, t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

is called the optimal value function of the problem

Optimal value function and controls

The function

$$V(x_0) := \inf_{\mathbf{u}(\cdot) \in \mathcal{U}^{[t_0, t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

is called the optimal value function of the problem

(Here and in the following we consider t_0 and t_f fixed. If these times can vary, one can define $V(t_0, x_0)$, $V(t_f, x_0)$, or $V(t_0, t_f, x_0)$)

Optimal value function and controls

The function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

is called the optimal value function of the problem

(Here and in the following we consider t_0 and t_f fixed. If these times can vary, one can define $V(t_0, x_0)$, $V(t_f, x_0)$, or $V(t_0, t_f, x_0)$)

A control $\mathbf{u}^\star(\cdot) \in \mathcal{U}$ is called optimal for x_0 if

$$J(x_0, \mathbf{u}^\star(\cdot)) = V(x_0)$$

Optimal value function and controls

The function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

is called the optimal value function of the problem

(Here and in the following we consider t_0 and t_f fixed. If these times can vary, one can define $V(t_0, x_0)$, $V(t_f, x_0)$, or $V(t_0, t_f, x_0)$)

A control $\mathbf{u}^\star(\cdot) \in \mathcal{U}$ is called optimal for x_0 if

$$J(x_0, \mathbf{u}^\star(\cdot)) = V(x_0)$$

The corresponding optimal trajectory is denoted by

$$x^\star(t) = x_{\mathbf{u}^\star}(t, t_0, x_0)$$

Optimal value function and controls

The function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

is called the optimal value function of the problem

(Here and in the following we consider t_0 and t_f fixed. If these times can vary, one can define $V(t_0, x_0)$, $V(t_f, x_0)$, or $V(t_0, t_f, x_0)$)

A control $\mathbf{u}^\star(\cdot) \in \mathcal{U}$ is called optimal for x_0 if

$$J(x_0, \mathbf{u}^\star(\cdot)) = V(x_0)$$

The corresponding optimal trajectory is denoted by

$$x^\star(t) = x_{\mathbf{u}^\star}(t, t_0, x_0)$$

Note: If $\mathbf{u}^\star(\cdot)$ exists for x_0, then the "inf" in the definition of $V(x_0)$ is a "min"

Open-loop optimal control

Given an OCP, let $\mathbf{u}^\star(\cdot)$ be the optimal control for initial value x_0

Open-loop optimal control

Given an OCP, let $\mathbf{u}^\star(\cdot)$ be the optimal control for initial value x_0

This function is determined at time t_0 for all future times $t \in [t_0, t_f]$, depending on x_0

Open-loop optimal control

Given an OCP, let $\mathbf{u}^\star(\cdot)$ be the optimal control for initial value x_0

This function is determined at time t_0 for all future times $t \in [t_0, t_f]$, depending on x_0

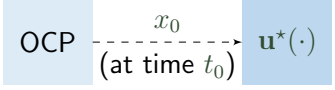
Such a function is called an open-loop optimal control

Open-loop optimal control

Given an OCP, let $\mathbf{u}^\star(\cdot)$ be the optimal control for initial value x_0

This function is determined at time t_0 for all future times $t \in [t_0, t_f]$, depending on x_0

Such a function is called an open-loop optimal control

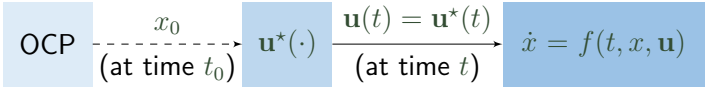


Open-loop optimal control

Given an OCP, let $\mathbf{u}^\star(\cdot)$ be the optimal control for initial value x_0

This function is determined at time t_0 for all future times $t \in [t_0, t_f]$, depending on x_0

Such a function is called an open-loop optimal control

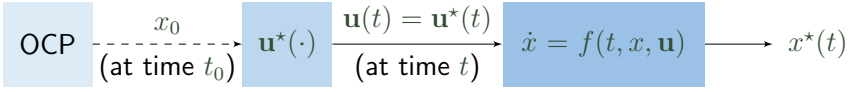


Open-loop optimal control

Given an OCP, let $\mathbf{u}^\star(\cdot)$ be the optimal control for initial value x_0

This function is determined at time t_0 for all future times $t \in [t_0, t_f]$, depending on x_0

Such a function is called an open-loop optimal control



Closed-loop optimal control

If there exists a map $F : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ such that all optimal controls satisfy

$$\mathbf{u}^\star(t) = F(t, x^\star(t))$$

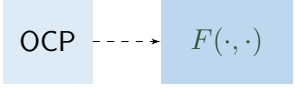
for all $t \in [t_0, t_f]$, then $\mathbf{u}^\star(\cdot)$ is called closed-loop optimal control and F is an optimal feedback law

Closed-loop optimal control

If there exists a map $F : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ such that all optimal controls satisfy

$$\mathbf{u}^\star(t) = F(t, x^\star(t))$$

for all $t \in [t_0, t_f]$, then $\mathbf{u}^\star(\cdot)$ is called closed-loop optimal control and F is an optimal feedback law

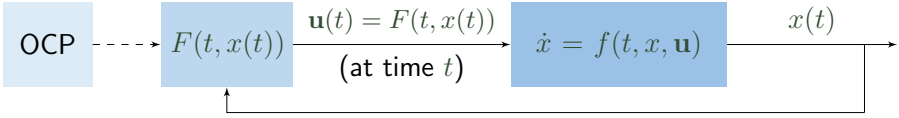


Closed-loop optimal control

If there exists a map $F : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ such that all optimal controls satisfy

$$\mathbf{u}^\star(t) = F(t, x^\star(t))$$

for all $t \in [t_0, t_f]$, then $\mathbf{u}^\star(\cdot)$ is called closed-loop optimal control and F is an optimal feedback law

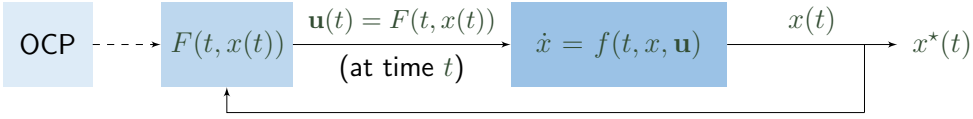


Closed-loop optimal control

If there exists a map $F : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ such that all optimal controls satisfy

$$\mathbf{u}^\star(t) = F(t, x^\star(t))$$

for all $t \in [t_0, t_f]$, then $\mathbf{u}^\star(\cdot)$ is called closed-loop optimal control and F is an optimal feedback law



Quiz

Why are (optimal) controls in feedback form preferred?

- They can react to perturbations
- They are easier to implement in practice
- They are easier to compute

Quiz

Why are (optimal) controls in feedback form preferred?

- They can react to perturbations
- They are easier to implement in practice
- They are easier to compute

Solution: see the pendulum example

Summary of Part 1: Optimal Control Problems — An Introduction

Outline of the course

Part 1: Optimal Control Problems — An Introduction

Part 2: Solution Concepts

Part 3: Numerical Solution Methods

Part 4: Model Predictive Control

Part 5: Deep Neural Networks for High-Dimensional Problems

Purpose of this part

Introduce classical solution concepts for optimal control problems

Purpose of this part

Introduce classical solution concepts for optimal control problems

Explain the differences between these concepts

Purpose of this part

Introduce classical solution concepts for optimal control problems

Explain the differences between these concepts

Provide the basis for numerical solution methods

Contents of this part

Part 2: Solution Concepts

- Dynamic Programming
- Euler-Lagrange Equations
- Pontryagin's Minimum Principle

Dynamic Programming

Optimal value function

Recall the definition of the optimal value function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

Optimal value function

Recall the definition of the optimal value function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

Dynamic Programming is a concept that relates optimal value functions and optimal feedback controls

Optimal value function

Recall the definition of the optimal value function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

Dynamic Programming is a concept that relates optimal value functions and optimal feedback controls. For simplicity, we introduce it for time-invariant problems (ℓ, g and constraints do not depend on t)

Optimal value function

Recall the definition of the optimal value function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

Dynamic Programming is a concept that relates optimal value functions and optimal feedback controls. For simplicity, we introduce it for time-invariant problems (ℓ, g and constraints do not depend on t)

We begin with the discrete-time setting

Optimal value function

Recall the definition of the optimal value function

$$V(x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}^{[t_0,t_f]}(x_0)} J(x_0, \mathbf{u}(\cdot))$$

Dynamic Programming is a concept that relates optimal value functions and optimal feedback controls. For simplicity, we introduce it for time-invariant problems (ℓ, g and constraints do not depend on t)

We begin with the discrete-time setting

A technical assumption that we make throughout this part:

The cost function ℓ satisfies $\ell \geq 0$ or ℓ is bounded and $\beta < 1$

Dynamic programming principle in discrete time

Theorem: (Dynamic programming principle *or* Bellman equation) Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data. The optimal value function satisfies for all $x_0 \in \mathbb{R}^n$

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Dynamic programming principle in discrete time

Theorem: (Dynamic programming principle *or* Bellman equation) Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data. The optimal value function satisfies for all $x_0 \in \mathbb{R}^n$

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." [Bellman '57]

Dynamic programming principle in discrete time

Theorem: (Dynamic programming principle *or* Bellman equation) Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data. The optimal value function satisfies for all $x_0 \in \mathbb{R}^n$

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Sketch of proof: For J we have

$$J(x_0, \mathbf{u}(\cdot)) = \sum_{t=0}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) = \ell(x(0), \mathbf{u}(0)) + \sum_{t=1}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t))$$

Dynamic programming principle in discrete time

Theorem: (Dynamic programming principle *or* Bellman equation) Consider the discrete-time infinite-horizon OCP (OCP$_d$) with time-invariant problem data. The optimal value function satisfies for all $x_0 \in \mathbb{R}^n$

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Sketch of proof: For J we have

$$\begin{aligned} J(x_0, \mathbf{u}(\cdot)) = \sum_{t=0}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) &= \ell(x(0), \mathbf{u}(0)) + \sum_{t=1}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) \\ &= \ell(x(0), \mathbf{u}(0)) + \beta \sum_{t=0}^{\infty} \beta^t \ell(x(t+1), \mathbf{u}(t+1)) \end{aligned}$$

Dynamic programming principle in discrete time

Theorem: (Dynamic programming principle *or* Bellman equation) Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data. The optimal value function satisfies for all $x_0 \in \mathbb{R}^n$

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Sketch of proof: For J we have

$$\begin{aligned} J(x_0, \mathbf{u}(\cdot)) = \sum_{t=0}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) &= \ell(x(0), \mathbf{u}(0)) + \sum_{t=1}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) \\ &= \ell(x(0), \mathbf{u}(0)) + \beta \sum_{t=0}^{\infty} \beta^t \ell(x(t+1), \mathbf{u}(t+1)) \\ &= \ell(x(0), \mathbf{u}(0)) + \beta J(x(1), \mathbf{u}(\cdot + 1)) \end{aligned}$$

Dynamic programming principle in discrete time

Theorem: (Dynamic programming principle *or* Bellman equation) Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data. The optimal value function satisfies for all $x_0 \in \mathbb{R}^n$

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Sketch of proof: For J we have, writing $u = \mathbf{u}(0)$

$$\begin{aligned} J(x_0, \mathbf{u}(\cdot)) = \sum_{t=0}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) &= \ell(x(0), \mathbf{u}(0)) + \sum_{t=1}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) \\ &= \ell(x(0), \mathbf{u}(0)) + \beta \sum_{t=0}^{\infty} \beta^t \ell(x(t+1), \mathbf{u}(t+1)) \\ &= \ell(x(0), \mathbf{u}(0)) + \beta J(x(1), \mathbf{u}(\cdot + 1)) \\ &= \ell(x_0, u) + \beta J(g(x_0, u), \mathbf{u}(\cdot + 1)) \end{aligned}$$

Dynamic programming principle in discrete time

Theorem: (Dynamic programming principle *or* Bellman equation) Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data. The optimal value function satisfies for all $x_0 \in \mathbb{R}^n$

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Sketch of proof: For J we have, writing $u = \mathbf{u}(0)$

$$\begin{aligned} J(x_0, \mathbf{u}(\cdot)) = \sum_{t=0}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) &= \ell(x(0), \mathbf{u}(0)) + \sum_{t=1}^{\infty} \beta^t \ell(x(t), \mathbf{u}(t)) \\ &= \ell(x(0), \mathbf{u}(0)) + \beta \sum_{t=0}^{\infty} \beta^t \ell(x(t+1), \mathbf{u}(t+1)) \\ &= \ell(x(0), \mathbf{u}(0)) + \beta J(x(1), \mathbf{u}(\cdot + 1)) \\ &= \ell(x_0, u) + \beta J(g(x_0, u), \mathbf{u}(\cdot + 1)) \end{aligned}$$

This equality carries over to the infimum over these expressions □

Characterisation of optimal open-loop controls

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Theorem: Consider the discrete-time infinite-horizon OCP (OCP$_d$) with time-invariant problem data and a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$\ell(x(t), \mathbf{u}(t)) + \beta V(g(x(t), \mathbf{u}(t))) = \inf_{u \in U} \{\ell(x(t), u) + \beta V(g(x(t), u))\}$$

holds for all $t = 0, 1, 2, \ldots$

Characterisation of optimal open-loop controls

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Theorem: Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data and a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$\ell(x(t), \mathbf{u}(t)) + \beta V(g(x(t), \mathbf{u}(t))) = \inf_{u \in U} \{\ell(x(t), u) + \beta V(g(x(t), u))\}$$

holds for all $t = 0, 1, 2, \ldots$ Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is optimal for $x_0 = x(0)$

Characterisation of optimal open-loop controls

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Theorem: Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data and a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$\ell(x(t), \mathbf{u}(t)) + \beta V(g(x(t), \mathbf{u}(t))) = \inf_{u \in U} \{\ell(x(t), u) + \beta V(g(x(t), u))\}$$

holds for all $t = 0, 1, 2, \ldots$ Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is optimal for $x_0 = x(0)$

Sketch of proof: (DPP) and the assumption implies

$$\ell(x(t), \mathbf{u}(t)) = V(x(t)) - \beta V(g(x(t), \mathbf{u}(t)))$$

Characterisation of optimal open-loop controls

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Theorem: Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data and a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$\ell(x(t), \mathbf{u}(t)) + \beta V(g(x(t), \mathbf{u}(t))) = \inf_{u \in U} \{\ell(x(t), u) + \beta V(g(x(t), u))\}$$

holds for all $t = 0, 1, 2, \ldots$ Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is optimal for $x_0 = x(0)$

Sketch of proof: (DPP) and the assumption implies

$$\ell(x(t), \mathbf{u}(t)) = V(x(t)) - \beta V(g(x(t), \mathbf{u}(t))) = V(x(t)) - \beta V(x(t+1))$$

Characterisation of optimal open-loop controls

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Theorem: Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data and a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$\ell(x(t), \mathbf{u}(t)) + \beta V(g(x(t), \mathbf{u}(t))) = \inf_{u \in U} \{\ell(x(t), u) + \beta V(g(x(t), u))\}$$

holds for all $t = 0, 1, 2, \ldots$ Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is optimal for $x_0 = x(0)$

Sketch of proof: (DPP) and the assumption implies

$$\ell(x(t), \mathbf{u}(t)) = V(x(t)) - \beta V(g(x(t), \mathbf{u}(t))) = V(x(t)) - \beta V(x(t+1))$$

$$\Rightarrow \qquad \sum_{t=0}^{T-1} \beta^t \ell(x(t), \mathbf{u}(t)) = V(x_0) - \beta^T V(x(T))$$

Characterisation of optimal open-loop controls

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Theorem: Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data and a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$\ell(x(t), \mathbf{u}(t)) + \beta V(g(x(t), \mathbf{u}(t))) = \inf_{u \in U} \{\ell(x(t), u) + \beta V(g(x(t), u))\}$$

holds for all $t = 0, 1, 2, \ldots$ Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is optimal for $x_0 = x(0)$

Sketch of proof: (DPP) and the assumption implies

$$\ell(x(t), \mathbf{u}(t)) = V(x(t)) - \beta V(g(x(t), \mathbf{u}(t))) = V(x(t)) - \beta V(x(t+1))$$

$$\Rightarrow \qquad \sum_{t=0}^{T-1} \beta^t \ell(x(t), \mathbf{u}(t)) = V(x_0) - \beta^T V(x(T)) \;\to\; V(x_0) \quad \text{as } T \to \infty$$

Characterisation of optimal open-loop controls

$$V(x_0) = \inf_{u \in \mathbb{U}} \{\ell(x_0, u) + \beta V(g(x_0, u))\} \qquad \text{(DPP)}$$

Theorem: Consider the discrete-time infinite-horizon OCP (OCP_d) with time-invariant problem data and a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$\ell(x(t), \mathbf{u}(t)) + \beta V(g(x(t), \mathbf{u}(t))) = \inf_{u \in U} \{\ell(x(t), u) + \beta V(g(x(t), u))\}$$

holds for all $t = 0, 1, 2, \ldots$ Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is optimal for $x_0 = x(0)$

Sketch of proof: (DPP) and the assumption implies

$$\ell(x(t), \mathbf{u}(t)) = V(x(t)) - \beta V(g(x(t), \mathbf{u}(t))) = V(x(t)) - \beta V(x(t+1))$$

$$\Rightarrow \qquad \sum_{t=0}^{T-1} \beta^t \ell(x(t), \mathbf{u}(t)) = V(x_0) - \beta^T V(x(T)) \;\to\; V(x_0) \quad \text{as } T \to \infty$$

This shows the claim since $\quad J(x, \mathbf{u}(\cdot)) = \lim_{T \to \infty} \sum_{t=0}^{T-1} \beta^t \ell(x(t), \mathbf{u}(t))$ □

Characterisation of optimal feedback controls

Consider now a feedback control $F : \mathbb{R}^n \to \mathbb{U}$. This control is applied via

$$x^+(t) := x(t+1) = g(x(t), F(x(t))), \quad x(0) = x_0, \ t = 0, 1, 2, \ldots$$

It thus generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

Characterisation of optimal feedback controls

Consider now a feedback control $F : \mathbb{R}^n \to \mathbb{U}$. This control is applied via

$$x^+(t) := x(t+1) = g(x(t), F(x(t))), \quad x(0) = x_0, \ t = 0, 1, 2, \ldots$$

It thus generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

We call a feedback $F^\star$ optimal, if for each $x_0 \in \mathbb{R}^n$ the control $\mathbf{u}^\star$ generated by $F^\star$ satisfies

$$J(x_0, \mathbf{u}^\star) = V(x_0)$$

Characterisation of optimal feedback controls

Consider now a feedback control $F : \mathbb{R}^n \to \mathbb{U}$. This control is applied via

$$x^+(t) := x(t+1) = g(x(t), F(x(t))), \quad x(0) = x_0, \ t = 0, 1, 2, \ldots$$

It thus generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

We call a feedback $F^\star$ optimal, if for each $x_0 \in \mathbb{R}^n$ the control $\mathbf{u}^\star$ generated by $F^\star$ satisfies

$$J(x_0, \mathbf{u}^\star) = V(x_0)$$

Corollary: A feedback F satisfying

$$\ell(x, F(x)) + \beta V(g(x, F(x))) = \inf_{u \in \mathbb{U}} \{\ell(x, u) + \beta V(g(x, u))\}$$

for all $x \in \mathbb{X}$ is an optimal feedback law

Characterisation of optimal feedback controls

Consider now a feedback control $F : \mathbb{R}^n \to \mathbb{U}$. This control is applied via

$$x^+(t) := x(t+1) = g(x(t), F(x(t))), \quad x(0) = x_0, \ t = 0, 1, 2, \ldots$$

It thus generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

We call a feedback $F^\star$ optimal, if for each $x_0 \in \mathbb{R}^n$ the control $\mathbf{u}^\star$ generated by $F^\star$ satisfies

$$J(x_0, \mathbf{u}^\star) = V(x_0)$$

Corollary: A feedback F satisfying

$$\ell(x, F(x)) + \beta V(g(x, F(x))) = \inf_{u \in \mathbb{U}} \{\ell(x, u) + \beta V(g(x, u))\}$$

for all $x \in \mathbb{X}$ is an optimal feedback law

Sketch of proof: One checks that the controls generated by F satisfy the conditions of the previous theorem □

Dynamic programming principle in continuous time

Consider (OCP_c) with time-invariant problem data on the infinite horizon $[0, \infty)$.

$$V(x_0) = \inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \left\{ \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt + \beta^\tau V(x(\tau)) \right\}$$

Problem: The "inf" is still over a function $\mathbf{u}(\cdot)$, not over a value $u \in \mathbb{U}$

Dynamic programming principle in continuous time

Consider (OCP_c) with time-invariant problem data on the infinite horizon $[0, \infty)$.

$$V(x_0) = \inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \left\{ \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt + \beta^\tau V(x(\tau)) \right\}$$

Problem: The "inf" is still over a function $\mathbf{u}(\cdot)$, not over a value $u \in \mathbb{U}$

Remedy: Send $\tau \to 0$, after rearranging terms and dividing by τ:

$$\inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \left\{ \frac{\beta^\tau V(x(\tau)) - V(x(0))}{\tau} + \frac{1}{\tau} \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt \right\} = 0$$

Dynamic programming principle in continuous time

Consider (OCP_c) with time-invariant problem data on the infinite horizon $[0, \infty)$.

$$V(x_0) = \inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \left\{ \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt + \beta^\tau V(x(\tau)) \right\}$$

Problem: The "inf" is still over a function $\mathbf{u}(\cdot)$, not over a value $u \in \mathbb{U}$

Remedy: Send $\tau \to 0$, after rearranging terms and dividing by τ:

$$\inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \left\{ \frac{\beta^\tau V(x(\tau)) - V(x(0))}{\tau} + \frac{1}{\tau} \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt \right\} = 0$$

$$(\tau \to 0) \Rightarrow \qquad \inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \left\{ \frac{d}{dt}\bigg|_{t=0} \beta^t V(x(t)) + \ell(x(0), \mathbf{u}(0)) \right\} = 0$$

Dynamic programming principle in continuous time

Consider (OCP_c) with time-invariant problem data on the infinite horizon $[0, \infty)$.

$$V(x_0) = \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt + \beta^\tau V(x(\tau)) \right\}$$

Problem: The "inf" is still over a function $\mathbf{u}(\cdot)$, not over a value $u \in \mathbb{U}$

Remedy: Send $\tau \to 0$, after rearranging terms and dividing by τ:

$$\inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \frac{\beta^\tau V(x(\tau)) - V(x(0))}{\tau} + \frac{1}{\tau} \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt \right\} = 0$$

$$(\tau \to 0) \Rightarrow \quad \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \left. \frac{d}{dt} \right|_{t=0} \beta^t V(x(t)) + \ell(x(0), \mathbf{u}(0)) \right\} = 0$$

$$\Leftrightarrow \quad \inf_{u\in\mathbb{U}} \{ -\delta V(x_0) + DV(x_0) f(x_0, u) + \ell(x_0, u) \} = 0$$

with $\delta = -\ln \beta$

Dynamic programming principle in continuous time

Consider (OCP_c) with time-invariant problem data on the infinite horizon $[0, \infty)$.

$$V(x_0) = \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt + \beta^\tau V(x(\tau)) \right\}$$

Problem: The "inf" is still over a function $\mathbf{u}(\cdot)$, not over a value $u \in \mathbb{U}$

Remedy: Send $\tau \to 0$, after rearranging terms and dividing by τ:

$$\inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \frac{\beta^\tau V(x(\tau)) - V(x(0))}{\tau} + \frac{1}{\tau} \int_0^\tau \beta^t \ell(x(t), \mathbf{u}(t)) dt \right\} = 0$$

$$(\tau \to 0) \Rightarrow \quad \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \left. \frac{d}{dt} \right|_{t=0} \beta^t V(x(t)) + \ell(x(0), \mathbf{u}(0)) \right\} = 0$$

$$\Leftrightarrow \quad \inf_{u\in\mathbb{U}} \{ -\delta V(x_0) + DV(x_0) f(x_0, u) + \ell(x_0, u) \} = 0$$

with $\delta = -\ln \beta$. This is the Hamilton-Jacobi-Bellman equation

Dynamic programming principle in continuous time

Consider (OCP_c) with time-invariant problem data on the infinite horizon $[0,\infty)$.

$$V(x_0) = \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \int_0^\tau \beta^t \ell(x(t),\mathbf{u}(t))dt + \beta^\tau V(x(\tau)) \right\}$$

Problem: The "inf" is still over a function $\mathbf{u}(\cdot)$, not over a value $u \in \mathbb{U}$

Remedy: Send $\tau \to 0$, after rearranging terms and dividing by τ:

$$\inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \frac{\beta^\tau V(x(\tau)) - V(x(0))}{\tau} + \frac{1}{\tau}\int_0^\tau \beta^t \ell(x(t),\mathbf{u}(t))dt \right\} = 0$$

$$(\tau \to 0) \Rightarrow \quad \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \left.\frac{d}{dt}\right|_{t=0} \beta^t V(x(t)) + \ell(x(0),\mathbf{u}(0)) \right\} = 0$$

$$\Leftrightarrow \quad \inf_{u\in\mathbb{U}} \{-\delta V(x_0) + DV(x_0)f(x_0,u) + \ell(x_0,u)\} = 0$$

with $\delta = -\ln\beta$. This is the Hamilton-Jacobi-Bellman equation

(here $\lim_{t\searrow 0}\mathbf{u}(t) = \mathbf{u}(0)$ is assumed; the proof can be modified if this does not hold)

Characterisation of optimal open-loop controls

$$\delta V(x_0) = \inf_{u \in \mathbb{U}} \{DV(x_0) f(x_0, u) + \ell(x_0, u)\} \qquad \text{(HJB)}$$

Characterisation of optimal open-loop controls

$$\delta V(x_0) = \inf_{u \in \mathbb{U}} \{DV(x_0) f(x_0, u) + \ell(x_0, u)\} \qquad \text{(HJB)}$$

Theorem: Consider a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$DV(x(t)) f(x(t), \mathbf{u}(t)) + \ell(x(t), \mathbf{u}(t)) = \inf_{u \in \mathbb{U}} \{DV(x(t)) f(x(t), u) + \ell(x(t), u)\}$$

holds for all $t \geq 0$

Characterisation of optimal open-loop controls

$$\delta V(x_0) = \inf_{u \in \mathbb{U}} \{DV(x_0) f(x_0, u) + \ell(x_0, u)\} \qquad \text{(HJB)}$$

Theorem: Consider a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$DV(x(t)) f(x(t), \mathbf{u}(t)) + \ell(x(t), \mathbf{u}(t)) = \inf_{u \in \mathbb{U}} \{DV(x(t)) f(x(t), u) + \ell(x(t), u)\}$$

holds for all $t \geq 0$

Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is an optimal control for initial value $x_0 = x(0)$

Characterisation of optimal open-loop controls

$$\delta V(x_0) = \inf_{u \in \mathbb{U}} \{DV(x_0) f(x_0, u) + \ell(x_0, u)\} \qquad \text{(HJB)}$$

Theorem: Consider a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$DV(x(t)) f(x(t), \mathbf{u}(t)) + \ell(x(t), \mathbf{u}(t)) = \inf_{u \in \mathbb{U}} \{DV(x(t)) f(x(t), u) + \ell(x(t), u)\}$$

holds for all $t \geq 0$

Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is an optimal control for initial value $x_0 = x(0)$

Sketch of proof: Using (HJB) and integrating the equation from 0 to T yields

$$\int_0^T \beta^t \ell(x(t), \mathbf{u}(t)) dt = V(x_0) - \beta^T V(x(T))$$

Characterisation of optimal open-loop controls

$$\delta V(x_0) = \inf_{u \in \mathbb{U}} \{DV(x_0) f(x_0, u) + \ell(x_0, u)\} \qquad \text{(HJB)}$$

Theorem: Consider a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$DV(x(t)) f(x(t), \mathbf{u}(t)) + \ell(x(t), \mathbf{u}(t)) = \inf_{u \in \mathbb{U}} \{DV(x(t)) f(x(t), u) + \ell(x(t), u)\}$$

holds for all $t \geq 0$

Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is an optimal control for initial value $x_0 = x(0)$

Sketch of proof: Using (HJB) and integrating the equation from 0 to T yields

$$\int_0^T \beta^t \ell(x(t), \mathbf{u}(t)) dt = V(x_0) - \beta^T V(x(T)) \;\to\; V(x_0)$$

as $T \to \infty$

Characterisation of optimal open-loop controls

$$\delta V(x_0) = \inf_{u \in \mathbb{U}} \{DV(x_0) f(x_0, u) + \ell(x_0, u)\} \qquad \text{(HJB)}$$

Theorem: Consider a trajectory $x(\cdot)$ with control $\mathbf{u}(\cdot)$ and assume that

$$DV(x(t)) f(x(t), \mathbf{u}(t)) + \ell(x(t), \mathbf{u}(t)) = \inf_{u \in \mathbb{U}} \{DV(x(t)) f(x(t), u) + \ell(x(t), u)\}$$

holds for all $t \geq 0$

Then $\mathbf{u}^\star(\cdot) = \mathbf{u}(\cdot)$ is an optimal control for initial value $x_0 = x(0)$

Sketch of proof: Using (HJB) and integrating the equation from 0 to T yields

$$\int_0^T \beta^t \ell(x(t), \mathbf{u}(t)) dt = V(x_0) - \beta^T V(x(T)) \;\to\; V(x_0)$$

as $T \to \infty$. This shows the claim since $J(x, \mathbf{u}(\cdot)) = \lim_{T \to \infty} \int_0^T \beta^t \ell(x(t), \mathbf{u}(t)) dt$ □

Characterisation of optimal feedback controls

Consider again a feedback control $F : \mathbb{R}^n \to \mathbb{U}$, now in continuous time:

$$\dot{x}(t) = f(x(t), F(x(t))), \quad x(0) = x_0, \ t \geq 0$$

Characterisation of optimal feedback controls

Consider again a feedback control $F : \mathbb{R}^n \to \mathbb{U}$, now in continuous time:

$$\dot{x}(t) = f(x(t), F(x(t))), \quad x(0) = x_0, \ t \geq 0$$

Assuming that this equation has a solution, F generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

Characterisation of optimal feedback controls

Consider again a feedback control $F : \mathbb{R}^n \to \mathbb{U}$, now in continuous time:

$$\dot{x}(t) = f(x(t), F(x(t))), \quad x(0) = x_0, \ t \geq 0$$

Assuming that this equation has a solution, F generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

Recall: $F^\star$ is optimal if for each $x_0 \in \mathbb{R}^n$ the control $\mathbf{u}^\star$ generated by $F^\star$ satisfies

$$J(x_0, \mathbf{u}^\star) = V(x_0)$$

Characterisation of optimal feedback controls

Consider again a feedback control $F : \mathbb{R}^n \to \mathbb{U}$, now in continuous time:

$$\dot{x}(t) = f(x(t), F(x(t))), \quad x(0) = x_0, \ t \geq 0$$

Assuming that this equation has a solution, F generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

Recall: $F^\star$ is optimal if for each $x_0 \in \mathbb{R}^n$ the control $\mathbf{u}^\star$ generated by $F^\star$ satisfies

$$J(x_0, \mathbf{u}^\star) = V(x_0)$$

Corollary: A feedback F satisfying

$$DV(x)f(x, F(x)) + \ell(x, F(x)) = \inf_{u \in \mathbb{U}} \{DV(x)f(x, u) + \ell(x, u)\}$$

for all $x \in \mathbb{R}^n$ is an optimal feedback law

Characterisation of optimal feedback controls

Consider again a feedback control $F : \mathbb{R}^n \to \mathbb{U}$, now in continuous time:

$$\dot{x}(t) = f(x(t), F(x(t))), \quad x(0) = x_0, \ t \geq 0$$

Assuming that this equation has a solution, F generates the control $\mathbf{u}(t) = F(x(t))$, depending on x_0

Recall: $F^\star$ is optimal if for each $x_0 \in \mathbb{R}^n$ the control $\mathbf{u}^\star$ generated by $F^\star$ satisfies

$$J(x_0, \mathbf{u}^\star) = V(x_0)$$

Corollary: A feedback F satisfying

$$DV(x)f(x, F(x)) + \ell(x, F(x)) = \inf_{u \in \mathbb{U}} \{DV(x)f(x, u) + \ell(x, u)\}$$

for all $x \in \mathbb{R}^n$ is an optimal feedback law

Sketch of proof: One checks that the controls generated by F satisfy the conditions of the previous theorem $\square$

Dynamic programming for finite-horizon problems

Dynamic programming also works for finite-horizon problems

Dynamic programming for finite-horizon problems

Dynamic programming also works for finite-horizon problems

Task: minimize

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \int_{t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t))\, dt + \beta^{t_f} L(x(t_f))$$

or

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \sum_{t=t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t)) + \beta^{t_f} L(x(t_f))$$

Dynamic programming for finite-horizon problems

Dynamic programming also works for finite-horizon problems

Task: minimize

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \int_{t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t))\, dt + \beta^{t_f} L(x(t_f))$$

or

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \sum_{t=t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t)) + \beta^{t_f} L(x(t_f))$$

The optimal value function is then time dependent $V(t_0, t_f, x_0) := \inf_{\mathbf{u}(\cdot) \in \mathcal{U}} J(t_0, t_f, x_0, \mathbf{u}(\cdot))$

Dynamic programming for finite-horizon problems

Dynamic programming also works for finite-horizon problems

Task: minimize

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \int_{t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t))\, dt + \beta^{t_f} L(x(t_f))$$

or

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \sum_{t=t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t)) + \beta^{t_f} L(x(t_f))$$

The optimal value function is then time dependent $V(t_0, t_f, x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} J(t_0, t_f, x_0, \mathbf{u}(\cdot))$

The Bellman equation becomes

$$V(t_0, t_f, x_0) = \inf_{u\in\mathbb{U}} \{\ell(x_0, u) + \beta V(t_0+1, t_f, g(x_0, u))\} \qquad \text{if } t_0 < t_f, \quad V(t_f, t_f, x) = L(x)$$

Dynamic programming for finite-horizon problems

Dynamic programming also works for finite-horizon problems

Task: minimize

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \int_{t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t))\, dt + \beta^{t_f} L(x(t_f))$$

or

$$J(t_0, t_f, x_0, \mathbf{u}(\cdot)) := \sum_{t=t_0}^{t_f} \beta^t \ell(x(t), \mathbf{u}(t)) + \beta^{t_f} L(x(t_f))$$

The optimal value function is then time dependent $V(t_0, t_f, x_0) := \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} J(t_0, t_f, x_0, \mathbf{u}(\cdot))$

The Bellman equation becomes

$$V(t_0, t_f, x_0) = \inf_{u\in\mathbb{U}} \{\ell(x_0, u) + \beta V(t_0+1, t_f, g(x_0, u))\} \qquad \text{if } t_0 < t_f, \quad V(t_f, t_f, x) = L(x)$$

and the Hamilton-Jacobi-Bellman equation reads

$$\delta \frac{\partial}{\partial t_0} V(t_0, t_f, x_0) = \inf_{u\in\mathbb{U}} \left\{ \frac{\partial}{\partial x} V(t_0, t_f, x_0) f(x_0, u) + \ell(x, u) \right\} \text{ if } t_0 < t_f, \quad V(t_f, t_f, x) = L(x)$$

A note on optimal feedback laws

Note: It follows from dynamic programming theory, that optimal feedback laws (if they exist)

- do not depend on time if the problem data is time-invariant and $t_f = \infty$
- do in general depend on time in all other cases

Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation is a partial differential equation (PDE)

If the optimal value function V is C^1, then it satisfies this equation uniquely (under appropriate boundary conditions)

Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation is a partial differential equation (PDE)

If the optimal value function V is C^1, then it satisfies this equation uniquely (under appropriate boundary conditions)

Problem: In many practical examples, V is not C^1

Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation is a partial differential equation (PDE)

If the optimal value function V is C^1, then it satisfies this equation uniquely (under appropriate boundary conditions)

Problem: In many practical examples, V is not C^1

Remedy: Use viscosity solution theory [Lions '82; Crandall/Lions '83]

Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation is a partial differential equation (PDE)

If the optimal value function V is C^1, then it satisfies this equation uniquely (under appropriate boundary conditions)

Problem: In many practical examples, V is not C^1

Remedy: Use viscosity solution theory [Lions '82; Crandall/Lions '83]

This is a weak solution concept that allows for a general existence and uniqueness result

Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation is a partial differential equation (PDE)

If the optimal value function V is C^1, then it satisfies this equation uniquely (under appropriate boundary conditions)

Problem: In many practical examples, V is not C^1

Remedy: Use viscosity solution theory [Lions '82; Crandall/Lions '83]

This is a weak solution concept that allows for a general existence and uniqueness result. However, it does not simplify computations

Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation is a partial differential equation (PDE)

If the optimal value function V is C^1, then it satisfies this equation uniquely (under appropriate boundary conditions)

Problem: In many practical examples, V is not C^1

Remedy: Use viscosity solution theory [Lions '82; Crandall/Lions '83]

This is a weak solution concept that allows for a general existence and uniqueness result. However, it does not simplify computations

Particularly, the computation of the optimal feedback law from

$$DV(x)f(x,F(x)) + \ell(x,F(x)) = \inf_{u\in\mathbb{U}} \{DV(x)f(x,u) + \ell(x,u)\}$$

is highly nontrivial in the viscosity solution framework

Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation is a partial differential equation (PDE)

If the optimal value function V is C^1, then it satisfies this equation uniquely (under appropriate boundary conditions)

Problem: In many practical examples, V is not C^1

Remedy: Use viscosity solution theory [Lions '82; Crandall/Lions '83]

This is a weak solution concept that allows for a general existence and uniqueness result. However, it does not simplify computations

Particularly, the computation of the optimal feedback law from

$$DV(x)f(x, F(x)) + \ell(x, F(x)) = \inf_{u \in \mathbb{U}} \{DV(x)f(x, u) + \ell(x, u)\}$$

is highly nontrivial in the viscosity solution framework

$\rightsquigarrow$ It is often much easier to discretize the problem in time and use the discrete-time theory

Intermediate Summary – Dynamic programming and the HJBE

The solution to an OCP can be characterized by dynamic programming

Intermediate Summary – Dynamic programming and the HJBE

The solution to an OCP can be characterized by dynamic programming

- Bellman equation in discrete time (time-invariant infinite-horizon problem)

$$V(x) = \inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\} \qquad \text{(DDP)}$$

- Hamilton-Jacobi-Bellman equation in continuous time (time-invariant finite-horizon)

$$\delta \frac{\partial}{\partial t_0} V(t_0, t_f, x) = \inf_{u \in \mathbb{U}} \left\{ \frac{\partial}{\partial x} V(t_0, t_f, x) f(x,u) + \ell(x,u) \right\} \qquad \text{(HJB)}$$

Intermediate Summary – Dynamic programming and the HJBE

The solution to an OCP can be characterized by dynamic programming

- Bellman equation in discrete time (time-invariant infinite-horizon problem)

$$V(x) = \inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\} \qquad \text{(DDP)}$$

- Hamilton-Jacobi-Bellman equation in continuous time (time-invariant finite-horizon)

$$\delta \frac{\partial}{\partial t_0} V(t_0, t_f, x) = \inf_{u \in \mathbb{U}} \left\{ \frac{\partial}{\partial x} V(t_0, t_f, x) f(x,u) + \ell(x,u) \right\} \qquad \text{(HJB)}$$

- From these equations, optimal feedback laws may be obtained, which are often required in practice

Intermediate Summary – Dynamic programming and the HJBE

The solution to an OCP can be characterized by dynamic programming

- Bellman equation in discrete time (time-invariant infinite-horizon problem)

$$V(x) = \inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\} \qquad \text{(DDP)}$$

- Hamilton-Jacobi-Bellman equation in continuous time (time-invariant finite-horizon)

$$\delta \frac{\partial}{\partial t_0} V(t_0, t_f, x) = \inf_{u \in \mathbb{U}} \left\{ \frac{\partial}{\partial x} V(t_0, t_f, x) f(x,u) + \ell(x,u) \right\} \qquad \text{(HJB)}$$

- From these equations, optimal feedback laws may be obtained, which are often required in practice
- But, solving the DPP or the HJB equation is in general difficult!
  (we will come back to this later)

Euler-Lagrange Equations

Problem Setup

The Euler-Lagrange equations allow to compute open-loop control functions for fixed initial values in a simplified setting:

$$\min_{\mathbf{u}(\cdot)} \quad \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to: (P)

$$\dot{x} = f(t, x, \mathbf{u}), \quad x(t_0) = x_0$$
$$\mathbf{u}(\cdot) \in \mathcal{C}[t_0, t_f]^m$$

$$f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \quad f \in \mathcal{C}^0 \text{ w.r.t. } (t, x, u), \quad f \in \mathcal{C}^1 \text{ w.r.t. } (x, u)$$
$$\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \ , \quad \ell \in \mathcal{C}^0 \text{ w.r.t. } (t, x, u), \quad \ell \in \mathcal{C}^1 \text{ w.r.t. } (x, u)$$

Short hand notation:

$$D_w Z = \frac{\partial}{\partial w} Z, \quad (D_w Z)^\top = Z_w, \quad Z \in \{f, \ell, L, \dots\} \text{ and } w \in \{x, u, t\}$$

Problem Setup

The Euler-Lagrange equations allow to compute open-loop control functions for fixed initial values in a simplified setting (continuous control functions, no constraints)

$$\min_{\mathbf{u}(\cdot)} \quad \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to: (P)

$$\dot{x} = f(t, x, \mathbf{u}), \quad x(t_0) = x_0$$
$$\mathbf{u}(\cdot) \in \mathcal{C}[t_0, t_f]^m$$

$$f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \quad f \in \mathcal{C}^0 \text{ w.r.t. } (t, x, u), \quad f \in \mathcal{C}^1 \text{ w.r.t. } (x, u)$$
$$\ell : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \ , \quad \ell \in \mathcal{C}^0 \text{ w.r.t. } (t, x, u), \quad \ell \in \mathcal{C}^1 \text{ w.r.t. } (x, u)$$

Short hand notation:

$$D_w Z = \frac{\partial}{\partial w} Z, \quad (D_w Z)^\top = Z_w, \quad Z \in \{f, \ell, L, \dots\} \text{ and } w \in \{x, u, t\}$$

The approach by Euler and Lagrange

$$\min_{\mathbf{u}(\cdot)} \quad \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to: (P)

$$0 = \dot{x} - f(t, x, \mathbf{u}), \quad x(t_0) = x_0$$

$$\mathbf{u}(\cdot) \in \mathcal{C}[t_0, t_f]^m$$

The approach by Euler and Lagrange

$$\min_{\mathbf{u}(\cdot)} \quad \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to: (P)

$$0 = \dot{x} - f(t, x, \mathbf{u}), \quad x(t_0) = x_0$$
$$\mathbf{u}(\cdot) \in \mathcal{C}[t_0, t_f]^m$$

Rewrite equality constraint imposed by the dynamics

$$J(x_0, \mathbf{u}) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) + \lambda(t)^\top \big(\dot{x}(t) - f(t, x(t), \mathbf{u}(t))\big)\, dt$$

with Lagrange multiplier $\lambda : [t_0, t_f] \to \mathbb{R}^n$

The approach by Euler and Lagrange

$\rightsquigarrow$ Equality constraints included in the objective

$$J(x_0, \mathbf{u}) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) + \lambda(t)^\top \big(\dot{x}(t) - f(t, x(t), \mathbf{u}(t))\big)\, dt$$

Apply integration by parts to $\lambda^\top \dot{x}$

$$\lambda^\top \dot{x} = -x^\top \dot{\lambda} + \frac{d}{dt}\left(x^\top \lambda\right) \qquad \Leftrightarrow \qquad \int \lambda^\top \dot{x} dt = \int -x^\top \dot{\lambda} dt + \left(x^\top \lambda\right)$$

to obtain

$$J(x_0, \mathbf{u}) = \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t)) - \Big(x(t)^\top \dot{\lambda}(t) + \lambda(t)^\top f(t, x(t), \mathbf{u}(t))\Big) dt + x^\top \lambda \Big|_{t_0}^{t_f}$$

The approach by Euler and Lagrange

Consider a small perturbation of the optimal control $\mathbf{u}^\star(\cdot)$

$$\mathbf{u}^\star(\cdot) + \eta \cdot \mathbf{v}(\cdot), \qquad \eta \in \mathbb{R},\ \eta \approx 0$$

which generates a perturbed state trajectory

$$x_{\mathbf{u}^\star + \eta \cdot \mathbf{v}}(t, x_0)$$

and the limit

$$\lim_{\eta \to 0} \frac{J(x_0, \mathbf{u}^\star + \eta \cdot \mathbf{v}) - J(x_0, \mathbf{u}^\star)}{\eta} = \left.\frac{\partial}{\partial \eta}\right|_{\eta=0} J(x_0, \mathbf{u}^\star + \eta \cdot \mathbf{v})$$

Optimality implies[2]

$$0 = \left.\frac{\partial}{\partial \eta}\right|_{\eta=0} J(x_0, \mathbf{u}^\star + \eta \cdot \mathbf{v}) = \mathfrak{d}J(x_0, \mathbf{u}^\star, \mathbf{v}) = 0$$

[2] $\mathfrak{d}J(x_0, \mathbf{u}^\star, \mathbf{v})$ denotes the Gateaux derivative w.r.t. $\mathbf{u}$.

The approach by Euler and Lagrange

Let

$$\delta x := \frac{\partial}{\partial \eta}\bigg|_{\eta=0} x_{\mathbf{u}^\star + \eta \cdot \mathbf{v}}(t, x_0),$$

called sensitivity or variation of $x(\cdot)$ with respect to the input perturbation $\mathbf{v}$

The approach by Euler and Lagrange

Let

$$\delta x := \frac{\partial}{\partial \eta}\bigg|_{\eta=0} x_{\mathbf{u}^\star + \eta \cdot \mathbf{v}}(t, x_0),$$

called sensitivity or variation of $x(\cdot)$ with respect to the input perturbation $\mathbf{v}$

$$\begin{aligned}\mathfrak{d}J(x_0, \mathbf{u}^\star, \mathbf{v}) = \int_{t_0}^{t_f} &\Big(\ell_x(t, x^\star(t), \mathbf{u}^\star(t)) - f_x^\top(t, x^\star(t), \mathbf{u}^\star(t))\lambda(t) - \dot\lambda(t)\Big)^\top \delta x(t) \\ &+ \big(\ell_u(t, x^\star(t), \mathbf{u}^\star(t)) - f_u^\top(t, x^\star(t), \mathbf{u}^\star(t))\lambda(t)\big)^\top \mathbf{v}\, dt \\ &- \underbrace{\delta x(t_0)^\top \lambda(t_0)}_{=0} + \delta x(t_f)^\top \lambda(t_f) \stackrel{!}{=} 0\end{aligned}$$

The approach by Euler and Lagrange

Let

$$\delta x := \frac{\partial}{\partial \eta}\bigg|_{\eta=0} x_{\mathbf{u}^\star + \eta \cdot \mathbf{v}}(t, x_0),$$

called sensitivity or variation of $x(\cdot)$ with respect to the input perturbation $\mathbf{v}$

$$\begin{aligned}\mathfrak{d}J(x_0, \mathbf{u}^\star, \mathbf{v}) = \int_{t_0}^{t_f} &\Big(\ell_x(t, x^\star(t), \mathbf{u}^\star(t)) - f_x^\top(t, x^\star(t), \mathbf{u}^\star(t))\lambda(t) - \dot\lambda(t)\Big)^\top \delta x(t) \\ &+ \big(\ell_u(t, x^\star(t), \mathbf{u}^\star(t)) - f_u^\top(t, x^\star(t), \mathbf{u}^\star(t))\lambda(t)\big)^\top \mathbf{v}\, dt \\ &- \underbrace{\delta x(t_0)^\top \lambda(t_0)}_{=0} + \delta x(t_f)^\top \lambda(t_f) \stackrel{!}{=} 0\end{aligned}$$

We obtain

$$\begin{aligned}0 &= \ell_x(t, x^\star, \mathbf{u}^\star) - f_x^\top(t, x^\star, \mathbf{u}^\star)\lambda - \dot\lambda \\ 0 &= \ell_u(t, x^\star, \mathbf{u}^\star) - f_u^\top(t, x^\star, \mathbf{u}^\star)\lambda \\ 0 &= \lambda(t_f)\end{aligned}$$

First-order necessary conditions of optimality

Theorem (First-order necessary conditions):
Suppose that $u^\star(\cdot) \in \mathcal{C}[t_0, t_f]^m$ is a local minimizer of Problem (P) and $x^\star(\cdot) \in \mathcal{C}^1[t_0, t_f]^n$, $x^\star(t) = x_{\mathbf{u}^\star}(t, t_0, x_0)$ is the corresponding solution.

Then there exists a function $\lambda^\star(\cdot) \in \mathcal{C}^1[t_0, t_f]^n$ such that, for all $t \in [t_0, t_f]$, the triple $(u^\star(\cdot), x^\star(\cdot), \lambda^\star(\cdot))$ satisfies:

$$\begin{aligned} \dot{x^\star} = &\quad f(t, x^\star, \mathbf{u}^\star), && x^\star(t_0) = x_0 \\ \dot{\lambda}^\star = &-\ell_x(t, x^\star, \mathbf{u}^\star) - f_x^\top(t, x^\star, \mathbf{u}^\star)\lambda^\star, && \lambda^\star(t_f) = 0 \qquad \text{(E-L)} \\ 0 = &\quad \ell_u(t, x^\star, \mathbf{u}^\star) + f_{\mathbf{u}}^\top(t, x^\star, \mathbf{u}^\star)\lambda^\star. \end{aligned}$$

First-order necessary conditions – Remarks

- (E-L) are known as Euler-Lagrange equations

First-order necessary conditions – Remarks

- (E-L) are known as Euler-Lagrange equations
- Unknowns: $(u^\star(\cdot), x^\star(\cdot), \lambda^\star(\cdot)) \in \mathcal{C}[t_0, t_f]^m \times \mathcal{C}^1[t_0, t_f]^n \times \mathcal{C}^1[t_0, t_f]^n$

First-order necessary conditions – Remarks

- (E-L) are known as Euler-Lagrange equations
- Unknowns: $(u^\star(\cdot), x^\star(\cdot), \lambda^\star(\cdot)) \in \mathcal{C}[t_0, t_f]^m \times \mathcal{C}^1[t_0, t_f]^n \times \mathcal{C}^1[t_0, t_f]^n$
- (E-L) are first-order necessary conditions of (P). Hence any triple $(\mathbf{u}(\cdot), x(\cdot), \lambda(\cdot))$ solving (E-L) is also referred to as an extremal

First-order necessary conditions – Remarks

- (E-L) are known as Euler-Lagrange equations
- Unknowns: $(u^\star(\cdot), x^\star(\cdot), \lambda^\star(\cdot)) \in \mathcal{C}[t_0, t_f]^m \times \mathcal{C}^1[t_0, t_f]^n \times \mathcal{C}^1[t_0, t_f]^n$
- (E-L) are first-order necessary conditions of (P). Hence any triple $(\mathbf{u}(\cdot), x(\cdot), \lambda(\cdot))$ solving (E-L) is also referred to as an extremal
- The variable λ is called adjoint or costate. It is the OCP counterpart to a Lagrange multiplier in static nonlinear optimization

First-order necessary conditions – Remarks

- (E-L) are known as Euler-Lagrange equations
- Unknowns: $(u^\star(\cdot), x^\star(\cdot), \lambda^\star(\cdot)) \in \mathcal{C}[t_0, t_f]^m \times \mathcal{C}^1[t_0, t_f]^n \times \mathcal{C}^1[t_0, t_f]^n$
- (E-L) are first-order necessary conditions of (P). Hence any triple $(\mathbf{u}(\cdot), x(\cdot), \lambda(\cdot))$ solving (E-L) is also referred to as an extremal
- The variable λ is called adjoint or costate. It is the OCP counterpart to a Lagrange multiplier in static nonlinear optimization
- If there is a terminal cost L in (P) (i.e., a Mayer term depending only on t_f and $x(t_f)$), then
$$\lambda(t_f) = L_x(t_f, x(t_f))$$

First-order necessary conditions – Remarks

- (E-L) are known as Euler-Lagrange equations
- Unknowns: $(u^\star(\cdot), x^\star(\cdot), \lambda^\star(\cdot)) \in \mathcal{C}[t_0, t_f]^m \times \mathcal{C}^1[t_0, t_f]^n \times \mathcal{C}^1[t_0, t_f]^n$
- (E-L) are first-order necessary conditions of (P). Hence any triple $(\mathbf{u}(\cdot), x(\cdot), \lambda(\cdot))$ solving (E-L) is also referred to as an extremal
- The variable λ is called adjoint or costate. It is the OCP counterpart to a Lagrange multiplier in static nonlinear optimization
- If there is a terminal cost L in (P) (i.e., a Mayer term depending only on t_f and $x(t_f)$), then
$$\lambda(t_f) = L_x(t_f, x(t_f))$$
- For terminal constraints $x(t_f) = x_f$, the condition on $\lambda(t_f)$ is replaced by $x(t_f) = x_f$

First-order necessary conditions – Remarks

- (E-L) are known as Euler-Lagrange equations
- Unknowns: $(u^\star(\cdot), x^\star(\cdot), \lambda^\star(\cdot)) \in \mathcal{C}[t_0, t_f]^m \times \mathcal{C}^1[t_0, t_f]^n \times \mathcal{C}^1[t_0, t_f]^n$
- (E-L) are first-order necessary conditions of (P). Hence any triple $(\mathbf{u}(\cdot), x(\cdot), \lambda(\cdot))$ solving (E-L) is also referred to as an extremal
- The variable λ is called adjoint or costate. It is the OCP counterpart to a Lagrange multiplier in static nonlinear optimization
- If there is a terminal cost L in (P) (i.e., a Mayer term depending only on t_f and $x(t_f)$), then
$$\lambda(t_f) = L_x(t_f, x(t_f))$$
- For terminal constraints $x(t_f) = x_f$, the condition on $\lambda(t_f)$ is replaced by $x(t_f) = x_f$
- For general terminal constraints $\psi(t_f, x(t_f)) = 0$, we obtain
$$\lambda(t_f) = L_x(t_f, x(t_f)) + \nu^\top \Psi_x(t_f, x(t_f))$$

First-order necessary conditions – Remarks

The Euler-Lagrange equations can be rewritten more concisely using the Hamiltonian function: $H : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$

$$H(t, x, u, \lambda) := \ell(t, x, u) + \lambda^\top f(t, x, u) = \ell(t, x, u) + \langle \lambda,\ f(t, x, u) \rangle$$

Notation for scalar product of $w, z, \in \mathbb{R}^n$: $\langle w,\ z \rangle = w^\top z$

[3] H.J. Sussmann and J.C. Willems, "300 years of optimal control: from the Brachystochrone to the Maximum Principle". In: IEEE Control Systems 17.3 (1997), pp. 32–44

First-order necessary conditions – Remarks

The Euler-Lagrange equations can be rewritten more concisely using the Hamiltonian function: $H : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$

$$H(t, x, u, \lambda) := \ell(t, x, u) + \lambda^\top f(t, x, u) = \ell(t, x, u) + \langle \lambda,\ f(t, x, u) \rangle$$

Notation for scalar product of $w, z, \in \mathbb{R}^n$: $\langle w,\ z \rangle = w^\top z$

Euler-Lagrange equations using Hamiltonian:

$$\begin{aligned} \dot{x^\star} = &\quad H_\lambda(t, x^\star, \mathbf{u}^\star, \lambda^\star), \quad x^\star(t_0) = x_0 \\ \dot{\lambda}^\star = &-H_x(t, x^\star, \mathbf{u}^\star, \lambda^\star), \quad \lambda^\star(t_f) = 0 \\ 0 = &\quad H_u(t, x^\star, \mathbf{u}^\star, \lambda^\star) \end{aligned}$$

[3] H.J. Sussmann and J.C. Willems, "300 years of optimal control: from the Brachystochrone to the Maximum Principle". In: IEEE Control Systems 17.3 (1997), pp. 32–44

First-order necessary conditions – Remarks

The Euler-Lagrange equations can be rewritten more concisely using the Hamiltonian function: $H : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$

$$H(t, x, u, \lambda) := \ell(t, x, u) + \lambda^\top f(t, x, u) = \ell(t, x, u) + \langle \lambda,\ f(t, x, u) \rangle$$

Notation for scalar product of $w, z, \in \mathbb{R}^n$: $\langle w,\ z \rangle = w^\top z$

Euler-Lagrange equations using Hamiltonian:

$$\begin{aligned} \dot{x^\star} = &\quad H_\lambda(t, x^\star, \mathbf{u}^\star, \lambda^\star), \quad x^\star(t_0) = x_0 \\ \dot{\lambda}^\star = &-H_x(t, x^\star, \mathbf{u}^\star, \lambda^\star), \quad \lambda^\star(t_f) = 0 \\ 0 = &\quad H_u(t, x^\star, \mathbf{u}^\star, \lambda^\star) \end{aligned}$$

Good overview on the history of optimal control: [Sussmann/Willems '97][3]

[3] H.J. Sussmann and J.C. Willems, "300 years of optimal control: from the Brachystochrone to the Maximum Principle". In: IEEE Control Systems 17.3 (1997), pp. 32–44

First-order necessary conditions – Remarks

- If (P) is time-invariant (ℓ, f do not depend on t), then the Hamiltonian is constant along optimal solutions: $\frac{d}{dt}H(t, x^\star(t), \mathbf{u}^\star(t), \lambda^\star(t)) = 0$

[4] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

First-order necessary conditions – Remarks

- If (P) is time-invariant (ℓ, f do not depend on t), then the Hamiltonian is constant along optimal solutions: $\frac{d}{dt}H(t, x^\star(t), \mathbf{u}^\star(t), \lambda^\star(t)) = 0$
- Change of Hamiltonian along optimal trajectory

$$\frac{d}{dt}H(t, x, \mathbf{u}, \lambda) = H_t + \langle H_x, f(t, x, \mathbf{u})\rangle + \langle H_{\mathbf{u}}, \dot{\mathbf{u}}\rangle + \langle f(t, x, \mathbf{u}), \dot{\lambda}\rangle$$

[4] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

First-order necessary conditions – Remarks

- If (P) is time-invariant (ℓ, f do not depend on t), then the Hamiltonian is constant along optimal solutions: $\frac{d}{dt}H(t, x^\star(t), \mathbf{u}^\star(t), \lambda^\star(t)) = 0$
- Change of Hamiltonian along optimal trajectory

$$\frac{d}{dt}H(t, x, \mathbf{u}, \lambda) = H_t + \langle H_x, f(t, x, \mathbf{u})\rangle + \langle H_{\mathbf{u}}, \dot{\mathbf{u}}\rangle + \langle f(t, x, \mathbf{u}), \dot{\lambda}\rangle$$

- Euler-Lagrange equations hold for local minima and maxima

[4] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

First-order necessary conditions – Remarks

- If (P) is time-invariant (ℓ, f do not depend on t), then the Hamiltonian is constant along optimal solutions: $\frac{d}{dt}H(t, x^\star(t), \mathbf{u}^\star(t), \lambda^\star(t)) = 0$
- Change of Hamiltonian along optimal trajectory

$$\frac{d}{dt}H(t, x, \mathbf{u}, \lambda) = H_t + \langle H_x, f(t, x, \mathbf{u})\rangle + \langle H_{\mathbf{u}}, \dot{\mathbf{u}}\rangle + \langle f(t, x, \mathbf{u}), \dot{\lambda}\rangle$$

- Euler-Lagrange equations hold for local minima and maxima. How can we tell one from the other?

[4] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

First-order necessary conditions – Remarks

- If (P) is time-invariant (ℓ, f do not depend on t), then the Hamiltonian is constant along optimal solutions: $\frac{d}{dt}H(t, x^\star(t), \mathbf{u}^\star(t), \lambda^\star(t)) = 0$
- Change of Hamiltonian along optimal trajectory

$$\frac{d}{dt}H(t, x, \mathbf{u}, \lambda) = H_t + \langle H_x, f(t, x, \mathbf{u})\rangle + \langle H_{\mathbf{u}}, \dot{\mathbf{u}}\rangle + \langle f(t, x, \mathbf{u}), \dot{\lambda}\rangle$$

- Euler-Lagrange equations hold for local minima and maxima. How can we tell one from the other?

  $\rightsquigarrow$ Second order necessary conditions (Legendre-Clebsch condition):

$$D^2H = H_{uu} \succeq 0 \text{ for minima,} \qquad D^2H = H_{uu} \preceq 0 \text{ for maxima}$$

[4] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

First-order necessary conditions – Remarks

- If (P) is time-invariant (ℓ, f do not depend on t), then the Hamiltonian is constant along optimal solutions: $\frac{d}{dt}H(t, x^\star(t), \mathbf{u}^\star(t), \lambda^\star(t)) = 0$
- Change of Hamiltonian along optimal trajectory

$$\frac{d}{dt}H(t, x, \mathbf{u}, \lambda) = H_t + \langle H_x, f(t, x, \mathbf{u})\rangle + \langle H_{\mathbf{u}}, \dot{\mathbf{u}}\rangle + \langle f(t, x, \mathbf{u}), \dot{\lambda}\rangle$$

- Euler-Lagrange equations hold for local minima and maxima. How can we tell one from the other?

  $\rightsquigarrow$ Second order necessary conditions (Legendre-Clebsch condition):

$$D^2H = H_{uu} \succeq 0 \text{ for minima,} \qquad D^2H = H_{uu} \preceq 0 \text{ for maxima}$$

- A readable introduction: [Chachuat '09][4]

[4] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

A toy example

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^1 \frac{1}{2}\mathbf{u}^2(t)dt$$

subject to

$$\dot{x}(t) = \mathbf{u}(t) - x(t), \quad x(0) = 1,\ x(1) = 0$$

Task: Write the Euler-Lagrange equations for this problem

A toy example

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^1 \frac{1}{2}\mathbf{u}^2(t)dt$$

subject to

$$\dot{x}(t) = \mathbf{u}(t) - x(t), \quad x(0) = 1,\ x(1) = 0$$

Task: Write the Euler-Lagrange equations for this problem

$$\begin{aligned} H(x,u,\lambda) &= \frac{1}{2}u^2 + \lambda^T(u-x) \\ \dot{x}^\star &= \quad H_\lambda(x^\star,\mathbf{u}^\star,\lambda^\star) = u^\star - x^\star \\ \dot{\lambda}^\star &= -H_x(x^\star,\mathbf{u}^\star,\lambda^\star) = \lambda^\star \\ 0 &= \quad H_u(x^\star,\mathbf{u}^\star,\lambda^\star) = u^\star + \lambda^\star \\ x^\star(0) &= 1, \quad x^\star(1) = 0 \end{aligned}$$

Discrete-time counterpart to (E-L)

Continuous-time OCP

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to

$$\dot{x} = f(t, x, \mathbf{u}), \quad x(0) = x_0$$

Discrete-time OCP (= NLP)

$$\min_{\mathbf{u}(\cdot)} \quad \sum_{t=0}^{N-1} \ell(t, x(t), \mathbf{u}(t))$$

subject to

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(0) = x_0$$

With $H = \ell + \lambda^\top f$:

$$\begin{aligned} \dot{x^\star} &= H_\lambda(t, x^\star, \mathbf{u}^\star, \lambda^\star), & x^\star(0) &= x_0 \\ \dot{\lambda}^\star &= -H_x(t, x^\star, \mathbf{u}^\star, \lambda^\star), & \lambda^\star(t_f) &= 0 \\ 0 &= H_u(t, x^\star, \mathbf{u}^\star, \lambda^\star) \end{aligned}$$

?

Discrete-time Euler-Lagrange equations

Define the Lagrangian pointwise in time

$$\widehat{\mathcal{L}}(t) \doteq \ell(x(t), \mathbf{u}(t)) + \lambda(t+1)^\top (g(t, x(t), \mathbf{u}(t)) - x(t+1))$$

and

$$\mathcal{L}(t, x, \mathbf{u}, \lambda) \doteq \lambda(0)^\top (x_0 - x(0)) + \sum_{t=0}^{N-1} \widehat{\mathcal{L}}(t)$$

Stationarity of the Lagrangian

$$\begin{aligned} \mathcal{L}_\lambda = 0 \quad &\rightsquigarrow \quad x^\star(t+1) = g(t, x^\star(t), \mathbf{u}^\star(t)) \\ \mathcal{L}_x = 0 \quad &\rightsquigarrow \quad \lambda^\star(t) = g_x^\top \lambda^\star(t+1) + \ell_x, \quad \lambda^\star(N) = 0 \qquad \text{(E-L}_\text{d}\text{)} \\ \mathcal{L}_u = 0 \quad &\rightsquigarrow \quad 0 = g_u^\top \lambda^\star(t+1) + \ell_u \end{aligned}$$

These are the discrete-time Euler-Lagrange equations

Comparison – Discrete-time and continuous-time Euler-Lagrange

Discrete-time OCP

$$\min_{\mathbf{u}(\cdot)} \sum_{t=0}^{N-1} \ell(t, x(t), \mathbf{u}(t))$$

subject to

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(0) = x_0$$

$$\begin{aligned} x^\star(t+1) &= g(t, x^\star(t), \mathbf{u}^\star(t)), & x(0) &= x_0 \\ \lambda^\star(t) &= g_x^\top \lambda^\star(t+1) + \ell_x, & \lambda^\star(N) &= 0 \\ 0 &= g_u^\top \lambda^\star(t+1) + \ell_u \end{aligned}$$

Comparison – Discrete-time and continuous-time Euler-Lagrange

Continuous-time OCP

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to

$$\dot{x} = f(t, x, \mathbf{u}), \quad x(0) = x_0$$

Discrete-time OCP

$$\min_{\mathbf{u}(\cdot)} \quad \sum_{t=0}^{N-1} \ell(t, x(t), \mathbf{u}(t))$$

subject to

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(0) = x_0$$

With $H = \ell + \lambda^\top f$:

$$\begin{aligned} \dot{x^\star} &= H_\lambda(t, x^\star, \mathbf{u}^\star, \lambda^\star), & x^\star(0) &= x_0 \\ \dot{\lambda}^\star &= -H_x(t, x^\star, \mathbf{u}^\star, \lambda^\star), & \lambda^\star(t_f) &= 0 \\ 0 &= H_u(t, x^\star, \mathbf{u}^\star, \lambda^\star) \end{aligned}$$

$$\begin{aligned} x^\star(t+1) &= g(t, x^\star(t), \mathbf{u}^\star(t)), & x(0) &= x_0 \\ \lambda^\star(t) &= g_x^\top \lambda^\star(t+1) + \ell_x, & \lambda^\star(N) &= 0 \\ 0 &= g_u^\top \lambda^\star(t+1) + \ell_u \end{aligned}$$

Comparison – Discrete-time and continuous-time Euler-Lagrange

Continuous-time OCP

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to

$$\dot{x} = f(t, x, \mathbf{u}), \quad x(0) = x_0$$

Discrete-time OCP

$$\min_{\mathbf{u}(\cdot)} \quad \sum_{t=0}^{N-1} \ell(t, x(t), \mathbf{u}(t))$$

subject to

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(0) = x_0$$

$$\begin{aligned} \dot{x^\star} &= f(t, x^\star, \mathbf{u}^\star), \quad x^\star(0) = x_0 \\ \dot{\lambda}^\star &= -f_x^\top \lambda^\star - \ell_x, \quad \lambda^\star(t_f) = 0 \\ 0 &= f_u^\top \lambda^\star + \ell_u \end{aligned}$$

$$\begin{aligned} x^\star(t+1) &= g(t, x^\star(t), \mathbf{u}^\star(t)), \quad x(0) = x_0 \\ \lambda^\star(t) &= g_x^\top \lambda^\star(t+1) + \ell_x, \quad \lambda^\star(N) = 0 \\ 0 &= g_u^\top \lambda^\star(t+1) + \ell_u \end{aligned}$$

Comparison – Discrete-time and continuous-time Euler-Lagrange

Continuous-time OCP

$$\min_{\mathbf{u}(\cdot)} \int_0^{t_f} \ell(t, x(t), \mathbf{u}(t))dt$$

subject to

$$\dot{x} = f(t, x, \mathbf{u}), \quad x(0) = x_0$$

Discrete-time OCP

$$\min_{\mathbf{u}(\cdot)} \sum_{t=0}^{N-1} \ell(t, x(t), \mathbf{u}(t))$$

subject to

$$x(t+1) = g(t, x(t), \mathbf{u}(t)), \quad x(0) = x_0$$

$$\begin{aligned} \dot{x^\star} &= f(t, x^\star, \mathbf{u}^\star), \quad x^\star(0) = x_0 \\ \dot{\lambda}^\star &= -f_x^\top \lambda^\star - \ell_x, \quad \lambda^\star(t_f) = 0 \\ 0 &= f_u^\top \lambda^\star + \ell_u \end{aligned}$$

$$\begin{aligned} x^\star(t+1) &= g(t, x^\star(t), \mathbf{u}^\star(t)), \quad x(0) = x_0 \\ \lambda^\star(t) &= g_x^\top \lambda^\star(t+1) + \ell_x, \quad \lambda^\star(N) = 0 \\ 0 &= g_u^\top \lambda^\star(t+1) + \ell_u \end{aligned}$$

But: There is no fully equivalent discrete-time counterpart of the Hamiltonian

Pontryagin's Minimum Principle

Preliminaries

We now allow for measurable controls and add constraints to the OCP (P). We start with input constraints

Preliminaries

We now allow for measurable controls and add constraints to the OCP (P). We start with input constraints

Problem setup (time invariant, free end time, terminal constraint):

$$\min_{\mathbf{u}(\cdot),t_f} \quad \int_{t_0}^{t_f} \ell(x(t),\mathbf{u}(t))dt$$

subject to

$$\begin{aligned} &\dot{x} = f(x,\mathbf{u}), \quad x(t_0) = x_0 \\ &\mathbf{u}(\cdot) \in L_\infty^{loc}\left([t_0,t_f],\mathbb{U}\right), \quad \mathbb{U} \subseteq \mathbb{R}^m \\ &t_f \in [t_0,T], \quad T < \infty \\ &x(t_f) = x_1 \end{aligned} \tag{$\mathsf{P}_{\mathsf{PMP}}$}$$

$$f:\ \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \quad f \in \mathcal{C}^0 \text{ w.r.t. } (x,u), \quad f \in \mathcal{C}^1 \text{ w.r.t. } (x)$$
$$\ell:\ \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}\ , \quad \ell \in \mathcal{C}^0 \text{ w.r.t. } (x,u), \quad \ell \in \mathcal{C}^1 \text{ w.r.t. } (x)$$

Towards Pontryagin's minimum principle

Reformulation in Mayer form

$$c(t) = \int_{t_0}^{t_f} \ell(x(t), \mathbf{u}(t))dt$$

$$\tilde{x}(t) = \begin{pmatrix} c(t) \\ x(t) \end{pmatrix}$$

$$\dot{\tilde{x}}(t) = \tilde{f}(x(t), \mathbf{u}(t)) = \begin{pmatrix} \ell(x(t), u(t)) \\ f(x(t), \mathbf{u}(t)) \end{pmatrix}$$

$$\tilde{x}(t_0) = \begin{pmatrix} 0 \\ x_0 \end{pmatrix}$$

Towards Pontryagin's minimum principle

Reformulation in Mayer form

$$c(t) = \int_{t_0}^{t_f} \ell(x(t), \mathbf{u}(t))dt$$

$$\tilde{x}(t) = \begin{pmatrix} c(t) \\ x(t) \end{pmatrix}$$

$$\dot{\tilde{x}}(t) = \tilde{f}(x(t), \mathbf{u}(t)) = \begin{pmatrix} \ell(x(t), u(t)) \\ f(x(t), \mathbf{u}(t)) \end{pmatrix}$$

$$\tilde{x}(t_0) = \begin{pmatrix} 0 \\ x_0 \end{pmatrix}$$

$$\min_{\mathbf{u}(\cdot), t_f} \quad c(t_f)$$

subject to

$$\dot{\tilde{x}} = \tilde{f}(x, \mathbf{u}), \quad \tilde{x}(t_0) = [0, x_0]^\top$$

$$\mathbf{u}(\cdot) \in \hat{\mathcal{C}}\left([t_0, t_f], \mathbb{U}\right), \quad \mathbb{U} \subseteq \mathbb{R}^m$$

$$t_f \in [t_0, T], \quad T < \infty$$

$$\begin{pmatrix} 0 & I \end{pmatrix} \tilde{x}(t_f) = x_1$$

Towards Pontryagin's minimum principle

Reformulation in Mayer form

$$c(t) = \int_{t_0}^{t_f} \ell(x(t), \mathbf{u}(t))dt$$

$$\tilde{x}(t) = \begin{pmatrix} c(t) \\ x(t) \end{pmatrix}$$

$$\dot{\tilde{x}}(t) = \tilde{f}(x(t), \mathbf{u}(t)) = \begin{pmatrix} \ell(x(t), u(t)) \\ f(x(t), \mathbf{u}(t)) \end{pmatrix}$$

$$\tilde{x}(t_0) = \begin{pmatrix} 0 \\ x_0 \end{pmatrix}$$

$$\min_{\mathbf{u}(\cdot), t_f} \quad c(t_f)$$

subject to

$$\dot{\tilde{x}} = \tilde{f}(x, \mathbf{u}), \quad \tilde{x}(t_0) = [0, x_0]^\top$$

$$\mathbf{u}(\cdot) \in \hat{\mathcal{C}}\left([t_0, t_f], \mathbb{U}\right), \quad \mathbb{U} \subseteq \mathbb{R}^m$$

$$t_f \in [t_0, T], \quad T < \infty$$

$$\begin{pmatrix} 0 & I \end{pmatrix} \tilde{x}(t_f) = x_1$$

Hamiltonian for reformulated problem

$$H(x, \mathbf{u}, \tilde{\lambda}) = \left\langle \tilde{\lambda},\ \tilde{f}(x, \mathbf{u}) \right\rangle = \lambda_0(t)\ell(x(t), \mathbf{u}(t)) + \lambda(t)^\top f(x(t), \mathbf{u}(t)) \qquad \tilde{\lambda} = \begin{pmatrix} \lambda_0 \\ \lambda \end{pmatrix}$$

Pontryagin's minimum principle[5]

Theorem: Suppose that $(u^\star(\cdot), t_f^\star) \in L_\infty^{loc}([t_0, t_f], \mathbb{U}) \times [t_0, T]$ is a global minimizer of Problem ($\mathsf{P}_{\mathsf{PMP}}$) and let $\tilde{x}^\star(\cdot)$ be the corresponding extended solution $\tilde{x}^\star(\cdot) = \tilde{x}_{\mathbf{u}^\star}(\cdot, t_0, \tilde{x}_0)$.

[5] V. G. Boltyanskii et al. "On the theory of optimal processes". In: Doklady Akademii Nauk SSSR 110 (1956), pp. 7–10, L. S. Pontryagin et al. The Mathematical Theory of Optimal Processes. John Wiley & Sons Inc., New York, 1962

Pontryagin's minimum principle[5]

Theorem: Suppose that $(u^\star(\cdot), t_f^\star) \in L_\infty^{loc}([t_0, t_f], \mathbb{U}) \times [t_0, T]$ is a global minimizer of Problem ($\mathsf{P}_{\mathsf{PMP}}$) and let $\tilde{x}^\star(\cdot)$ be the corresponding extended solution $\tilde{x}^\star(\cdot) = \tilde{x}_{\mathbf{u}^\star}(\cdot, t_0, \tilde{x}_0)$.

Then there exists an absolutely continuous function

$$\tilde{\lambda}^\star(\cdot) = (\lambda_0^\star(\cdot), \lambda^\star(\cdot))^\top,$$

$\tilde{\lambda}^\star(t) \neq [0, \ldots, 0]^\top$ for all $t \in [t_0, t_f]$, such that $(\mathbf{u}^\star(\cdot),\ \tilde{x}^\star(\cdot),\ \tilde{\lambda}^\star(\cdot))$ satisfy

$$\begin{aligned} \dot{\tilde{x}}^\star(t) &= \quad H_{\tilde{\lambda}}\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right), \qquad \tilde{x}(t_0) = (0, x_0)^\top \\ \dot{\tilde{\lambda}}^\star(t) &= -H_{\tilde{x}}\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right) \end{aligned}$$

with $H(x, \mathbf{u}, \tilde{\lambda}) = \left\langle \tilde{\lambda},\ \tilde{f}(x, \mathbf{u}) \right\rangle$

[5] V. G. Boltyanskii et al. "On the theory of optimal processes". In: Doklady Akademii Nauk SSSR 110 (1956), pp. 7–10, L. S. Pontryagin et al. The Mathematical Theory of Optimal Processes. John Wiley & Sons Inc., New York, 1962

Pontryagin's minimum principle[5]

Theorem: Suppose that $(u^\star(\cdot), t_f^\star) \in L_\infty^{loc}([t_0, t_f], \mathbb{U}) \times [t_0, T]$ is a global minimizer of Problem ($\mathsf{P}_{\mathsf{PMP}}$) and let $\tilde{x}^\star(\cdot)$ be the corresponding extended solution $\tilde{x}^\star(\cdot) = \tilde{x}_{\mathbf{u}^\star}(\cdot, t_0, \tilde{x}_0)$.

Then there exists an absolutely continuous function

$$\tilde{\lambda}^\star(\cdot) = (\lambda_0^\star(\cdot), \lambda^\star(\cdot))^\top,$$

$\tilde{\lambda}^\star(t) \neq [0, \ldots, 0]^\top$ for all $t \in [t_0, t_f]$, such that $(\mathbf{u}^\star(\cdot),\ \tilde{x}^\star(\cdot),\ \tilde{\lambda}^\star(\cdot))$ satisfy

$$\begin{aligned} \dot{\tilde{x}}^\star(t) &= \quad H_{\tilde{\lambda}}\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right), \qquad \tilde{x}(t_0) = (0, x_0)^\top \\ \dot{\tilde{\lambda}}^\star(t) &= -H_{\tilde{x}}\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right) \end{aligned}$$

with $H(x, \mathbf{u}, \tilde{\lambda}) = \left\langle \tilde{\lambda},\ \tilde{f}(x, \mathbf{u}) \right\rangle$ and ...

[5] V. G. Boltyanskii et al. "On the theory of optimal processes". In: Doklady Akademii Nauk SSSR 110 (1956), pp. 7–10, L. S. Pontryagin et al. The Mathematical Theory of Optimal Processes. John Wiley & Sons Inc., New York, 1962

Pontryagin's minimum principle

. . . and:

i) The function $H\left(x^\star(t), v, \tilde{\lambda}^\star(t)\right)$ attains its minimum on $\mathbb{U}$ at $v = \mathbf{u}^\star(t)$ for almost all $t \in [t_0, t_f^\star]$:

$$H\left(x^\star(t), v, \tilde{\lambda}^\star(t)\right) \geq H\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right).$$

Pontryagin's minimum principle

... and:

i) The function $H\left(x^\star(t), v, \tilde{\lambda}^\star(t)\right)$ attains its minimum on $\mathbb{U}$ at $v = \mathbf{u}^\star(t)$ for almost all $t \in [t_0, t_f^\star]$:

$$H\left(x^\star(t), v, \tilde{\lambda}^\star(t)\right) \geq H\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right).$$

ii) For all $t \in [t_0, t_f^\star]$ it holds

$$\lambda_0^\star(t) \equiv const. \geq 0, \qquad H\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right) = const.$$

Pontryagin's minimum principle

... and:

i) The function $H\left(x^\star(t), v, \tilde{\lambda}^\star(t)\right)$ attains its minimum on $\mathbb{U}$ at $v = \mathbf{u}^\star(t)$ for almost all $t \in [t_0, t_f^\star]$:

$$H\left(x^\star(t), v, \tilde{\lambda}^\star(t)\right) \geq H\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right).$$

ii) For all $t \in [t_0, t_f^\star]$ it holds

$$\lambda_0^\star(t) \equiv const. \geq 0, \qquad H\left(x^\star(t), \mathbf{u}^\star(t), \tilde{\lambda}^\star(t)\right) = const.$$

iii) If the final time t_f is free, the following transversality condition holds

$$H\left(x^\star(t_f^\star), \mathbf{u}^\star(t_f^\star), \tilde{\lambda}^\star(t_f^\star)\right) = 0.$$

Proofs

D. Liberzon. Calculus of Variations and Optimal Control Theory: A Concise Introduction. Princeton University Press, 2012.
http://liberzon.csl.illinois.edu/teaching/cvoc.pdf

E.R. Pinch. Optimal Control and the Calculus of Variations. Oxford University Press, 1995

Comments and Variants

- The (scalar) extra adjoint λ_0 is constant and non-negative

Comments and Variants

- The (scalar) extra adjoint λ_0 is constant and non-negative
- If $t_f < \infty$ is fixed and the terminal state is unconstrained

$$\lambda(t_f) = 0$$

Comments and Variants

- The (scalar) extra adjoint λ_0 is constant and non-negative
- If $t_f < \infty$ is fixed and the terminal state is unconstrained

$$\lambda(t_f) = 0$$

- If $t_f < \infty$ and a Mayer term L is considered

$$\lambda(t_f) = L_x(x(t_f))$$

Comments and Variants

- The (scalar) extra adjoint λ_0 is constant and non-negative
- If $t_f < \infty$ is fixed and the terminal state is unconstrained

$$\lambda(t_f) = 0$$

- If $t_f < \infty$ and a Mayer term L is considered

$$\lambda(t_f) = L_x(x(t_f))$$

- For general terminal constraints $\psi(t_f, x(t_f)) = 0$, we obtain

$$\lambda(t_f) = L_x(t_f, x(t_f)) + \nu^\top \Psi_x(t_f, x(t_f))$$

The extra adjoint λ_0

The Hamiltonian in the Euler-Lagrange equations (E-L) is

$$H(x, \mathbf{u}, \lambda) = \quad \ell(x(t), \mathbf{u}(t)) + \lambda^\top f(x(t), \mathbf{u}(t))$$

and in the PMP we have

$$H(x, \mathbf{u}, \underbrace{\lambda_0, \lambda}_{\tilde{\lambda}}) = \lambda_0 \ell(x(t), \mathbf{u}(t)) + \lambda^\top f(x(t), \mathbf{u}(t))$$

The extra adjoint λ_0

The Hamiltonian in the Euler-Lagrange equations (E-L) is

$$H(x, \mathbf{u}, \lambda) = \quad \ell(x(t), \mathbf{u}(t)) + \lambda^\top f(x(t), \mathbf{u}(t))$$

and in the PMP we have

$$H(x, \mathbf{u}, \underbrace{\lambda_0, \lambda}_{\tilde{\lambda}}) = \lambda_0 \ell(x(t), \mathbf{u}(t)) + \lambda^\top f(x(t), \mathbf{u}(t))$$

In the absence of state constraints[6], we can normalize $\lambda_0 = 1$, such that both Hamiltonians coincide

[6] This condition is only sufficient; if state constraints are present, then this may still be possible. OCPs with $\lambda_0 = 0$ are particularly complicated and called abnormal

The PMP vs. the Euler-Lagrange equations

The Euler-Lagrange equations (E-L) require for $\mathbf{u}^\star$

$$\forall t \in [t_0, t_f^\star]: \quad 0 = \quad \ell_u(x^\star, \mathbf{u}^\star) + f_u^\top(x^\star, \mathbf{u}^\star)\lambda^\star = H_u\left(x^\star, \mathbf{u}^\star, \lambda\right)$$

while the PMP reads

$$\forall t \in [t_0, t_f^\star]: \quad H\left(x^\star, v, \tilde{\lambda}^\star\right) \geq H\left(x^\star, \mathbf{u}^\star, \tilde{\lambda}^\star\right)$$

The PMP vs. the Euler-Lagrange equations

The Euler-Lagrange equations (E-L) require for $\mathbf{u}^\star$

$$\forall t \in [t_0, t_f^\star]: \quad 0 = \quad \ell_u(x^\star, \mathbf{u}^\star) + f_u^\top(x^\star, \mathbf{u}^\star)\lambda^\star = H_u\left(x^\star, \mathbf{u}^\star, \lambda\right)$$

while the PMP reads

$$\forall t \in [t_0, t_f^\star]: \quad H\left(x^\star, v, \tilde{\lambda}^\star\right) \geq H\left(x^\star, \mathbf{u}^\star, \tilde{\lambda}^\star\right)$$

Minimizing H with respect to u gives the necessary condition

$$H_u\left(x^\star, \mathbf{u}^\star, \tilde{\lambda}\right) = 0 = \lambda_0 \ell_u(t, x^\star, \mathbf{u}^\star) + f_u^\top(t, x^\star, \mathbf{u}^\star)\lambda^\star$$

which by normalizing $\lambda_0 = 1$ matches the condition for the controls in (E-L)

The PMP vs. the Euler-Lagrange equations

The Euler-Lagrange equations (E-L) require for $\mathbf{u}^\star$

$$\forall t \in [t_0, t_f^\star]: \quad 0 = \quad \ell_u(x^\star, \mathbf{u}^\star) + f_u^\top(x^\star, \mathbf{u}^\star)\lambda^\star = H_u\left(x^\star, \mathbf{u}^\star, \lambda\right)$$

while the PMP reads

$$\forall t \in [t_0, t_f^\star]: \quad H\left(x^\star, v, \tilde{\lambda}^\star\right) \geq H\left(x^\star, \mathbf{u}^\star, \tilde{\lambda}^\star\right)$$

Minimizing H with respect to u gives the necessary condition

$$H_u\left(x^\star, \mathbf{u}^\star, \tilde{\lambda}\right) = 0 = \lambda_0 \ell_u(t, x^\star, \mathbf{u}^\star) + f_u^\top(t, x^\star, \mathbf{u}^\star)\lambda^\star$$

which by normalizing $\lambda_0 = 1$ matches the condition for the controls in (E-L)

$\rightsquigarrow$ The PMP generalises the Euler-Lagrange equations

The PMP vs. the Euler-Lagrange equations

The Euler-Lagrange equations (E-L) require for $\mathbf{u}^\star$

$$\forall t \in [t_0, t_f^\star]: \quad 0 = \quad \ell_u(x^\star, \mathbf{u}^\star) + f_u^\top(x^\star, \mathbf{u}^\star)\lambda^\star = H_u\left(x^\star, \mathbf{u}^\star, \lambda\right)$$

while the PMP reads

$$\forall t \in [t_0, t_f^\star]: \quad H\left(x^\star, v, \tilde{\lambda}^\star\right) \geq H\left(x^\star, \mathbf{u}^\star, \tilde{\lambda}^\star\right)$$

Minimizing H with respect to u gives the necessary condition

$$H_u\left(x^\star, \mathbf{u}^\star, \tilde{\lambda}\right) = 0 = \lambda_0 \ell_u(t, x^\star, \mathbf{u}^\star) + f_u^\top(t, x^\star, \mathbf{u}^\star)\lambda^\star$$

which by normalizing $\lambda_0 = 1$ matches the condition for the controls in (E-L)

$\rightsquigarrow$ The PMP generalises the Euler-Lagrange equations

Note: The principle was originally developed for maximisation problems, hence it was called Pontryagin's Maximum Principle

A toy example

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^1 \frac{1}{2}\mathbf{u}^2(t)dt$$

subject to

$$\dot{x}(t) = \mathbf{u}(t) - x(t), \quad x(0) = 1,\ x(1) = 0$$
$$\mathcal{U} = [-0.6, 0]$$

Task: State the Hamiltonian and write the PMP for this problem

A toy example

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^1 \frac{1}{2}\mathbf{u}^2(t)dt$$

subject to

$$\dot{x}(t) = \mathbf{u}(t) - x(t), \quad x(0) = 1,\ x(1) = 0$$
$$\mathcal{U} = [-0.6, 0]$$

Task: State the Hamiltonian and write the PMP for this problem

$$H(x, u, \lambda) = \lambda_0 \frac{1}{2}u^2 + \lambda^T(u - x)$$
$$\dot{x}^\star = \quad H_\lambda(x^\star, \mathbf{u}^\star, \lambda^\star) = u^\star - x^\star$$
$$\dot{\lambda}^\star = -H_x(x^\star, \mathbf{u}^\star, \lambda^\star) = \lambda^\star$$
$$\lambda_0 \frac{1}{2}\mathbf{u}^{\star 2} + \lambda^{\star T}(\mathbf{u}^\star - x^\star) \geq \lambda_0 \frac{1}{2}u^2 + \lambda^T(u - x^\star)$$
$$x^\star(0) = 1, \quad x^\star(1) = 0$$

A pitfall example – The PMP for infinite horizon problems

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^{t_f} -(1-x(t))\mathbf{u}(t)dt$$

subject to

$$\dot{x}(t) = (1-x(t))\mathbf{u}(t), \quad x(0) = 0$$
$$u(\cdot) \in L_\infty^{loc}([0,t_f],[0,1])$$

- Horizons: $t_f < \infty$ and $t_f = \infty$

A pitfall example – The PMP for infinite horizon problems

$$\min_{\mathbf{u}(\cdot)} \int_0^{t_f} -(1-x(t))\mathbf{u}(t)dt$$

subject to

$$\dot{x}(t) = (1-x(t))\mathbf{u}(t), \quad x(0) = 0$$
$$u(\cdot) \in L_\infty^{loc}([0,t_f],[0,1])$$

- Horizons: $t_f < \infty$ and $t_f = \infty$

Observe that

$$J(x_0,\mathbf{u}) = -x(t_f) = e^{-\int_0^{t_f} \mathbf{u}(\tau)d\tau} - 1$$

which gives

$$\mathbf{u}^\star(t) \equiv 1$$
$$x^\star(t) = 1 - e^{-t}$$

for any horizon $t_f > 0$.

A pitfall example – The PMP for infinite horizon problems

$$\min_{\mathbf{u}(\cdot)} \int_0^{t_f} -(1-x(t))\mathbf{u}(t)dt$$

subject to

$$\dot{x}(t) = (1-x(t))\mathbf{u}(t), \quad x(0) = 0$$
$$u(\cdot) \in L_\infty^{loc}([0,t_f],[0,1])$$

- Horizons: $t_f < \infty$ and $t_f = \infty$

Observe that

$$J(x_0,\mathbf{u}) = -x(t_f) = e^{-\int_0^{t_f} \mathbf{u}(\tau)d\tau} - 1$$

which gives

$$\mathbf{u}^\star(t) \equiv 1$$
$$x^\star(t) = 1 - e^{-t}$$

for any horizon $t_f > 0$.

For $t_f < \infty$, the terminal condition for the adjoint λ is $\lambda(t_f) = 0$

A pitfall example – The PMP for infinite horizon problems

$$\min_{\mathbf{u}(\cdot)} \int_0^{t_f} -(1-x(t))\mathbf{u}(t)dt$$

subject to

$$\dot{x}(t) = (1-x(t))\mathbf{u}(t), \quad x(0) = 0$$
$$u(\cdot) \in L_\infty^{loc}([0,t_f],[0,1])$$

- Horizons: $t_f < \infty$ and $t_f = \infty$

Observe that

$$J(x_0,\mathbf{u}) = -x(t_f) = e^{-\int_0^{t_f} \mathbf{u}(\tau)d\tau} - 1$$

which gives

$$\mathbf{u}^\star(t) \equiv 1$$
$$x^\star(t) = 1 - e^{-t}$$

for any horizon $t_f > 0$.

For $t_f < \infty$, the terminal condition for the adjoint λ is $\lambda(t_f) = 0$

This suggests that for $t_f = \infty$, the condition might be $\lim_{t\to\infty} \lambda(t) = 0$

A pitfall example – The PMP for infinite horizon problems

The Hamiltonian reads

$$\begin{aligned} H(x, \mathbf{u}, \lambda_0, \lambda) &= -\lambda_0(1-x)\mathbf{u} + \lambda(1-x)\mathbf{u} \\ &= \quad (\lambda - \lambda_0)(1-x)\mathbf{u} \end{aligned}$$

Thus, the PMP entails

$$\dot{\lambda} = -H_x = (\lambda - \lambda_0)\mathbf{u}$$

with $\lambda_0 > 0$ (since there are no state constraints)

A pitfall example – The PMP for infinite horizon problems

The Hamiltonian reads

$$\begin{aligned} H(x, \mathbf{u}, \lambda_0, \lambda) &= -\lambda_0(1-x)\mathbf{u} + \lambda(1-x)\mathbf{u} \\ &= \quad (\lambda - \lambda_0)(1-x)\mathbf{u} \end{aligned}$$

Thus, the PMP entails

$$\dot{\lambda} = -H_x = (\lambda - \lambda_0)\mathbf{u}$$

with $\lambda_0 > 0$ (since there are no state constraints)

With the (already known) optimal control $\mathbf{u}^\star(t) \equiv 1$ we obtain

$$\lambda^\star(t) = (\lambda(0) - \lambda_0)e^t + \lambda_0$$

A pitfall example – The PMP for infinite horizon problems

The Hamiltonian reads

$$\begin{aligned} H(x, \mathbf{u}, \lambda_0, \lambda) &= -\lambda_0(1-x)\mathbf{u} + \lambda(1-x)\mathbf{u} \\ &= \quad (\lambda - \lambda_0)(1-x)\mathbf{u} \end{aligned}$$

Thus, the PMP entails

$$\dot{\lambda} = -H_x = (\lambda - \lambda_0)\mathbf{u}$$

with $\lambda_0 > 0$ (since there are no state constraints)

With the (already known) optimal control $\mathbf{u}^\star(t) \equiv 1$ we obtain

$$\lambda^\star(t) = (\lambda(0) - \lambda_0)e^t + \lambda_0$$

$\rightsquigarrow$ regardless of how we choose $\lambda(0)$, we never obtain $\lim_{t\to\infty} \lambda(t) = 0$

The PMP for infinite-horizon problems?

For infinite horizons,

- the general structure of the PMP remains unchanged
- But, if there is no terminal constraint on the state, the PMP does not provide a boundary/transversality condition[7] for the adjoint $\lim_{t\to\infty} \lambda(t)$

[7] This major issue was first observed in H. Halkin, "Necessary conditions for optimal control problems with infinite horizons". In: Econometrica: Journal of the Econometric Society 42.2 (1974), pp. 267–272.

[8] T. Faulwasser and C.M. Kellett, "On continuous-time infinite horizon optimal control—Dissipativity, stability, and transversality". In: Automatica 134 (2021), 109907

The PMP for infinite-horizon problems?

For infinite horizons,

- the general structure of the PMP remains unchanged
- But, if there is no terminal constraint on the state, the PMP does not provide a boundary/transversality condition[7] for the adjoint $\lim_{t\to\infty} \lambda(t)$

Remedies are known for problems with particular properties[8] (beyond the scope of this course)

[7] This major issue was first observed in H. Halkin, "Necessary conditions for optimal control problems with infinite horizons". In: Econometrica: Journal of the Econometric Society 42.2 (1974), pp. 267–272.

[8] T. Faulwasser and C.M. Kellett, "On continuous-time infinite horizon optimal control—Dissipativity, stability, and transversality". In: Automatica 134 (2021), 109907

The link between the HJBE and the PMP

Recall the HJBE for undiscounted problems:

$$\frac{\partial}{\partial t}V(t,t_f,x) = \inf_{u\in\mathcal{U}}\left\{\frac{\partial}{\partial x}V(t,t_f,x)f(x,u) + \ell(x,u)\right\} \quad \text{if } t_0 < t_f$$

The point-wise in time minimisation of the Hamiltonian in the PMP gives

$$H\left(x^\star(t),\mathbf{u}^\star,\tilde{\lambda}^\star(t)\right) = \min_{u\in\mathcal{U}} H\left(x^\star(t),u,\tilde{\lambda}^\star(t)\right) = \min_{u\in\mathcal{U}}(\lambda(t)^\star)^\top f(x^\star,u)+\lambda_0\ell(x^\star,u)$$

The link between the HJBE and the PMP

Recall the HJBE for undiscounted problems:

$$\frac{\partial}{\partial t}V(t,t_f,x) = \inf_{u\in\mathcal{U}}\left\{\frac{\partial}{\partial x}V(t,t_f,x)f(x,u)+\ell(x,u)\right\} \quad \text{if } t_0 < t_f$$

The point-wise in time minimisation of the Hamiltonian in the PMP gives

$$H\left(x^\star(t),\mathbf{u}^\star,\tilde{\lambda}^\star(t)\right) = \min_{u\in\mathcal{U}} H\left(x^\star(t),u,\tilde{\lambda}^\star(t)\right) = \min_{u\in\mathcal{U}}(\lambda(t)^\star)^\top f(x^\star,u)+\lambda_0\ell(x^\star,u)$$

and for $\lambda_0 = 1$ we observe the identity

$$\frac{\partial}{\partial x}V(t,t_f,x) = (\lambda^\star)^\top(t)$$

$\rightsquigarrow$ the adjoint $\lambda^\star$ is the derivative of the optimal value function w.r.t. x

Extensions of the presented PMP

- The PMP can be extended to problems with mixed input-state constraints[9]

$$g(x(t), \mathbf{u}(t)) \leq 0$$

[9] R.F. Hartl, S.P. Sethi, R.G. Vickson. "A survey of the maximum principles for optimal control problems with state constraints". In: SIAM Review 37.2 (1995), pp. 181–218

Extensions of the presented PMP

- The PMP can be extended to problems with mixed input-state constraints[9]

$$g(x(t), \mathbf{u}(t)) \leq 0$$

- These constraints require additional adjoints $\nu : [t_0, t_f] \to \mathbb{R}^{n_g}$ in the PMP and further extensions of the Hamiltonian

[9] R.F. Hartl, S.P. Sethi, R.G. Vickson. "A survey of the maximum principles for optimal control problems with state constraints". In: SIAM Review 37.2 (1995), pp. 181–218

Extensions of the presented PMP

- The PMP can be extended to problems with mixed input-state constraints[9]

$$g(x(t), \mathbf{u}(t)) \leq 0$$

- These constraints require additional adjoints $\nu : [t_0, t_f] \to \mathbb{R}^{n_g}$ in the PMP and further extensions of the Hamiltonian
- There exist discrete-time counterparts of the PMP, but they turn out to be more restrictive than KKT conditions

[9] R.F. Hartl, S.P. Sethi, R.G. Vickson. "A survey of the maximum principles for optimal control problems with state constraints". In: SIAM Review 37.2 (1995), pp. 181–218

Quiz

Why is it important that the PMP works for L_∞^{loc}-functions?

- Because there are examples where such functions are needed
- Because this simplifies its proof
- It ain't really important, the authors who wrote it simply did it this way

Why do we need controls from L_∞^{loc}?

This example[10] shows why is it important that the PMP allows for $u \in L_\infty^{loc}$:

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^{10} x_1(t)^2 \, dt$$

subject to

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathbf{u}, \quad x(0) = x_0$$

$$\forall t \in [0, 10]: \quad \mathbf{u}(t) \in [-1, 1]$$

[10] A.T. Fuller. "Relay control systems optimized for various performance criteria". In: Automation and remote control, Proc. first world congress IFAC Moscow. Vol. 1. 1960, pp. 510–519

Why do we need controls from L_∞^{loc}?

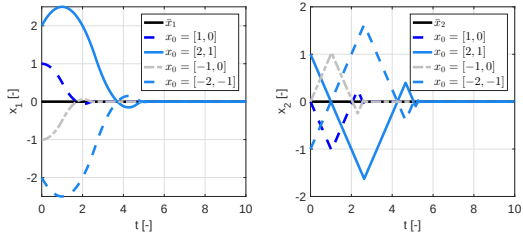
This example[10] shows why is it important that the PMP allows for $u \in L_\infty^{loc}$:

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^{10} x_1(t)^2 \, dt$$

subject to

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathbf{u}, \quad x(0) = x_0$$

$$\forall t \in [0, 10]: \quad \mathbf{u}(t) \in [-1, 1]$$



[10] A.T. Fuller. "Relay control systems optimized for various performance criteria". In: Automation and remote control, Proc. first world congress IFAC Moscow. Vol. 1. 1960, pp. 510–519

Why do we need controls from L_∞^{loc}?

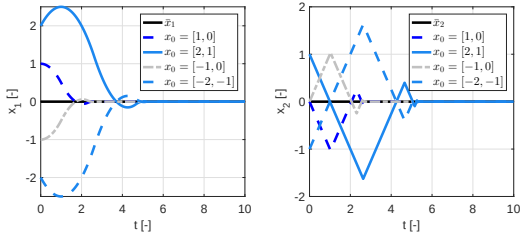
This example[10] shows why is it important that the PMP allows for $u \in L_\infty^{loc}$:

$$\min_{\mathbf{u}(\cdot)} \int_0^{10} x_1(t)^2\, dt$$

subject to

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathbf{u}, \quad x(0) = x_0$$

$$\forall t \in [0, 10]: \quad \mathbf{u}(t) \in [-1, 1]$$



The optimal controls take only values

$$\mathbf{u}^\star(t) \in \{-1, 0, 1\}$$

but has infinitely many switches

[10] A.T. Fuller. "Relay control systems optimized for various performance criteria". In: Automation and remote control, Proc. first world congress IFAC Moscow. Vol. 1. 1960, pp. 510–519

Why do we need controls from L_∞^{loc}?

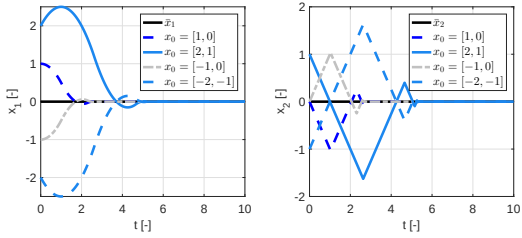
This example[10] shows why is it important that the PMP allows for $u \in L_\infty^{loc}$:

$$\min_{\mathbf{u}(\cdot)} \quad \int_0^{10} x_1(t)^2 \, dt$$

subject to

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathbf{u}, \quad x(0) = x_0$$

$$\forall t \in [0, 10]: \quad \mathbf{u}(t) \in [-1, 1]$$



The optimal controls take only values

$$\mathbf{u}^\star(t) \in \{-1, 0, 1\}$$

but has infinitely many switches

Easy problems can admit complicated analytic solutions!

[10] A.T. Fuller. "Relay control systems optimized for various performance criteria". In: Automation and remote control, Proc. first world congress IFAC Moscow. Vol. 1. 1960, pp. 510–519

Summary of Part 2: Solution Concepts

- Both Dynamic Programming and the PMP provide optimality conditions for the values of $\mathbf{u}^\star$ that are pointwise in time, i.e.,

  independent conditions on $\mathbf{u}^\star(t)$ for each $t \in [t_0, t_f]$

Summary of Part 2: Solution Concepts

- Both Dynamic Programming and the PMP provide optimality conditions for the values of $\mathbf{u}^\star$ that are pointwise in time, i.e.,

  independent conditions on $\mathbf{u}^\star(t)$ for each $t \in [t_0, t_f]$

- The conditions rely on auxiliary quantities:

Summary of Part 2: Solution Concepts

- Both Dynamic Programming and the PMP provide optimality conditions for the values of $\mathbf{u}^\star$ that are pointwise in time, i.e.,

  independent conditions on $\mathbf{u}^\star(t)$ for each $t \in [t_0, t_f]$

- The conditions rely on auxiliary quantities:
  - For dynamic programming: The optimal value function V, characterized by a partial differential equation or a discrete time functional equation

Summary of Part 2: Solution Concepts

- Both Dynamic Programming and the PMP provide optimality conditions for the values of $\mathbf{u}^\star$ that are pointwise in time, i.e.,

  independent conditions on $\mathbf{u}^\star(t)$ for each $t \in [t_0, t_f]$

- The conditions rely on auxiliary quantities:
  - For dynamic programming: The optimal value function V, characterized by a partial differential equation or a discrete time functional equation
  - For the PMP: The optimal adjoint $\lambda^\star$, characterized by an ordinary differential or difference equation

Summary of Part 2: Solution Concepts

- Both Dynamic Programming and the PMP provide optimality conditions for the values of $\mathbf{u}^\star$ that are pointwise in time, i.e.,

  independent conditions on $\mathbf{u}^\star(t)$ for each $t \in [t_0, t_f]$

- The conditions rely on auxiliary quantities:
  - For dynamic programming: The optimal value function V, characterized by a partial differential equation or a discrete time functional equation
  - For the PMP: The optimal adjoint $\lambda^\star$, characterized by an ordinary differential or difference equation
- Dynamic programming gives necessary and sufficient conditions for optimal open-loop and closed-loop controls

Summary of Part 2: Solution Concepts

- Both Dynamic Programming and the PMP provide optimality conditions for the values of $\mathbf{u}^\star$ that are pointwise in time, i.e.,

  independent conditions on $\mathbf{u}^\star(t)$ for each $t \in [t_0, t_f]$

- The conditions rely on auxiliary quantities:
  - For dynamic programming: The optimal value function V, characterized by a partial differential equation or a discrete time functional equation
  - For the PMP: The optimal adjoint $\lambda^\star$, characterized by an ordinary differential or difference equation
- Dynamic programming gives necessary and sufficient conditions for optimal open-loop and closed-loop controls
- The PMP gives necessary conditions for an optimal open-loop control

Summary of Part 2: Solution Concepts

- Both Dynamic Programming and the PMP provide optimality conditions for the values of $\mathbf{u}^\star$ that are pointwise in time, i.e.,

  independent conditions on $\mathbf{u}^\star(t)$ for each $t \in [t_0, t_f]$

- The conditions rely on auxiliary quantities:
  - For dynamic programming: The optimal value function V, characterized by a partial differential equation or a discrete time functional equation
  - For the PMP: The optimal adjoint $\lambda^\star$, characterized by an ordinary differential or difference equation
- Dynamic programming gives necessary and sufficient conditions for optimal open-loop and closed-loop controls
- The PMP gives necessary conditions for an optimal open-loop control
- In high dimensions, finding $\lambda^\star$ is usually much easier than finding V

Outline of the course

Part 1: Optimal Control Problems — An Introduction

Part 2: Solution Concepts

Part 3: Numerical Solution Methods

Part 4: Model Predictive Control

Part 5: Deep Neural Networks for High-Dimensional Problems

Purpose of this part

Introduce numerical methods for solving optimal control problems

Purpose of this part

Introduce numerical methods for solving optimal control problems

Compare methods that compute optimal feedback laws with methods computing open-loop optimal controls and trajectories

Purpose of this part

Introduce numerical methods for solving optimal control problems

Compare methods that compute optimal feedback laws with methods computing open-loop optimal controls and trajectories

Discuss the numerical effort

Contents of this part

Part 3: Numerical Solution Methods

- Methods based on Dynamic Programming, computing feedback laws
  - Classical methods for Hamilton-Jacobi-Bellman equations
  - Deep Reinforcement Learning
- Methods computing open-loop optimal controls
  - Methods based on Pontryagin's Maximum Principle
  - Direct Solution Methods

Methods based on Dynamic Programming

Conceptual approach

For brevity, we only look at infinite-horizon problems in this section

Conceptual approach

For brevity, we only look at infinite-horizon problems in this section

Recall that optimal feedback laws are characterized by the relations

$$\ell(x, F^\star(x)) + \beta V(g(x, F^\star(x))) = \inf_{u \in \mathbb{U}} \{\ell(x, u) + \beta V(g(x, u))\}$$

in discrete time and

$$DV(x)f(x, F^\star(x)) + \ell(x, F^\star(x)) = \inf_{u \in \mathbb{U}} \{DV(x)f(x, u) + \ell(x, u)\}$$

in continuous time

Conceptual approach

For brevity, we only look at infinite-horizon problems in this section

Recall that optimal feedback laws are characterized by the relations

$$\ell(x, F^\star(x)) + \beta V(g(x, F^\star(x))) = \inf_{u \in \mathbb{U}} \{\ell(x, u) + \beta V(g(x, u))\}$$

in discrete time and

$$DV(x)f(x, F^\star(x)) + \ell(x, F^\star(x)) = \inf_{u \in \mathbb{U}} \{DV(x)f(x, u) + \ell(x, u)\}$$

in continuous time

Hence, we now discuss numerical methods for calculating V, as then F can be computed from V as a minimiser of the above expressions

Conceptual approach

For brevity, we only look at infinite-horizon problems in this section

Recall that optimal feedback laws are characterized by the relations

$$\ell(x, F^\star(x)) + \beta V(g(x, F^\star(x))) = \inf_{u \in \mathbb{U}} \{\ell(x, u) + \beta V(g(x, u))\}$$

in discrete time and

$$DV(x)f(x, F^\star(x)) + \ell(x, F^\star(x)) = \inf_{u \in \mathbb{U}} \{DV(x)f(x, u) + \ell(x, u)\}$$

in continuous time

Hence, we now discuss numerical methods for calculating V, as then F can be computed from V as a minimiser of the above expressions

We start with the continuous-time case using the Hamilton-Jacobi-Bellman equation

Numerical solution of the HJB equation

Since the Hamilton-Jacobi-Bellman equation

$$\inf_{u \in \mathbb{U}} \{-\delta V(x_0) + DV(x_0) f(x_0, u) + \ell(x_0, u)\} = 0$$

is a partial differential equation, many standard numerical methods for PDEs can be applied. In the literature, one can find finite differences, finite elements, finite volume methods etc.

[11] M. Falcone and R. Ferretti, Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations, SIAM, 2013

Numerical solution of the HJB equation

Since the Hamilton-Jacobi-Bellman equation

$$\inf_{u\in\mathbb{U}} \{-\delta V(x_0) + DV(x_0)f(x_0,u) + \ell(x_0,u)\} = 0$$

is a partial differential equation, many standard numerical methods for PDEs can be applied. In the literature, one can find finite differences, finite elements, finite volume methods etc.

Here we explain the so-called semi-Lagrangian discretisation[11], which is tailored to HJB equations

[11] M. Falcone and R. Ferretti, Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations, SIAM, 2013

Numerical solution of the HJB equation

Since the Hamilton-Jacobi-Bellman equation

$$\inf_{u\in\mathbb{U}} \{-\delta V(x_0) + DV(x_0)f(x_0,u) + \ell(x_0,u)\} = 0$$

is a partial differential equation, many standard numerical methods for PDEs can be applied. In the literature, one can find finite differences, finite elements, finite volume methods etc.

Here we explain the so-called semi-Lagrangian discretisation[11], which is tailored to HJB equations

Its advantages are unconditional stability and the possibility to obtain error bounds for the feedback law computed from the approximation to V

[11] M. Falcone and R. Ferretti, Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations, SIAM, 2013

Semi-Lagrangian discretisation

$$\inf_{u \in \mathbb{U}} \{-\delta V(x) + DV(x) f(x,u) + \ell(x,u)\} = 0$$

In semi-Lagrangian discretisation, the equation is first discretised in time and then in space

Semi-Lagrangian discretisation

$$\inf_{u\in\mathbb{U}} \{-\delta V(x) + DV(x)f(x,u) + \ell(x,u)\} = 0$$

In semi-Lagrangian discretisation, the equation is first discretised in time and then in space:

Step 1: For a time step $h > 0$, approximate

$$-\delta V(x) + DV(x)f(x,u) \approx \frac{e^{-\delta h}V(\tilde{x}_u(h,x)) - V(x)}{h},$$

where $\tilde{x}_u(h,x)$ is a numerical approximation of the solution $x_{\mathbf{u}}(h,x)$ with $\mathbf{u} \equiv u$, e.g., the Euler approximation $\tilde{x}_u(h,x) = x + hf(x,u)$

Semi-Lagrangian discretisation

$$\inf_{u\in\mathbb{U}} \{-\delta V(x) + DV(x)f(x,u) + \ell(x,u)\} = 0$$

In semi-Lagrangian discretisation, the equation is first discretised in time and then in space:

Step 1: For a time step $h > 0$, approximate

$$-\delta V(x) + DV(x)f(x,u) \approx \frac{e^{-\delta h}V(\tilde{x}_u(h,x)) - V(x)}{h},$$

where $\tilde{x}_u(h,x)$ is a numerical approximation of the solution $x_{\mathbf{u}}(h,x)$ with $\mathbf{u} \equiv u$, e.g., the Euler approximation $\tilde{x}_u(h,x) = x + hf(x,u)$

Rearrange the terms to obtain the semi-discretised equation

$$V_h(x) = \inf_{u\in\mathbb{U}} \left\{h\ell(x_0,u) + e^{-\delta h}V_h(\tilde{x}_u(h,x))\right\} = 0$$

Semi-Lagrangian discretisation

$$\inf_{u\in\mathbb{U}} \{-\delta V(x) + DV(x)f(x,u) + \ell(x,u)\} = 0$$

In semi-Lagrangian discretisation, the equation is first discretised in time and then in space:

Step 1: For a time step $h > 0$, approximate

$$-\delta V(x) + DV(x)f(x,u) \approx \frac{e^{-\delta h}V(\tilde{x}_u(h,x)) - V(x)}{h},$$

where $\tilde{x}_u(h,x)$ is a numerical approximation of the solution $x_{\mathbf{u}}(h,x)$ with $\mathbf{u} \equiv u$, e.g., the Euler approximation $\tilde{x}_u(h,x) = x + hf(x,u)$

Rearrange the terms to obtain the semi-discretised equation

$$V_h(x) = \inf_{u\in\mathbb{U}} \left\{h\ell(x_0,u) + e^{-\delta h}V_h(\tilde{x}_u(h,x))\right\} = 0$$

(effectively, this step reverses part of the derivation of the HJB equation)

Semi-Lagrangian discretisation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x_0, u) + e^{-\delta h} V_h(\tilde{x}_u(h, x)) \right\} = 0$$

Step 2: We now solve this equation approximately on a compact set $C \subset \mathbb{R}^n$ using spatial discretization

Semi-Lagrangian discretisation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x_0, u) + e^{-\delta h} V_h(\tilde{x}_u(h, x)) \right\} = 0$$

Step 2: We now solve this equation approximately on a compact set $C \subset \mathbb{R}^n$ using spatial discretization:

Choose a finite dimensional function space $\mathcal{F}_k$ on C (e.g., continuous and piecewise linear functions defined on a simplicid grid with space step size $k > 0$; adaptive discretisation schemes may also be used)

Semi-Lagrangian discretisation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x_0, u) + e^{-\delta h} V_h(\tilde{x}_u(h, x)) \right\} = 0$$

Step 2: We now solve this equation approximately on a compact set $C \subset \mathbb{R}^n$ using spatial discretization:

Choose a finite dimensional function space $\mathcal{F}_k$ on C (e.g., continuous and piecewise linear functions defined on a simplicid grid with space step size $k > 0$; adaptive discretisation schemes may also be used)

Let Π_k be a projection from the space of bounded functions $W : C \to \mathbb{R}$ to $\mathcal{F}_k$

Semi-Lagrangian discretisation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x_0, u) + e^{-\delta h} V_h(\tilde{x}_u(h, x)) \right\} = 0$$

Step 2: We now solve this equation approximately on a compact set $C \subset \mathbb{R}^n$ using spatial discretization:

Choose a finite dimensional function space $\mathcal{F}_k$ on C (e.g., continuous and piecewise linear functions defined on a simplicid grid with space step size $k > 0$; adaptive discretisation schemes may also be used)

Let Π_k be a projection from the space of bounded functions $W : C \to \mathbb{R}$ to $\mathcal{F}_k$

Find $V_{hk} \in \mathcal{F}_k$ satisfying

$$V_{hk}(x) = \Pi_k \left(\inf_{u \in \mathbb{U}} \left\{ e^{-\delta h} V_{hk}(\tilde{x}_u(h, x)) + h\ell(x_0, u) \right\} \right) \qquad \text{for all } x \in C$$

Semi-Lagrangian discretisation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x_0, u) + e^{-\delta h} V_h(\tilde{x}_u(h, x)) \right\} = 0$$

Step 2: We now solve this equation approximately on a compact set $C \subset \mathbb{R}^n$ using spatial discretization:

Choose a finite dimensional function space $\mathcal{F}_k$ on C (e.g., continuous and piecewise linear functions defined on a simplicid grid with space step size $k > 0$; adaptive discretisation schemes may also be used)

Let Π_k be a projection from the space of bounded functions $W : C \to \mathbb{R}$ to $\mathcal{F}_k$

Find $V_{hk} \in \mathcal{F}_k$ satisfying

$$V_{hk}(x) = \Pi_k \left(\inf_{u \in \mathbb{U}} \left\{ e^{-\delta h} V_{hk}(\tilde{x}_u(h, x)) + h\ell(x_0, u) \right\} \right) \qquad \text{for all } x \in C$$

Note: Control input constraints are readily implementable, but state constraints lead to boundary conditions that may complicate the computation

2d Example: Optimal Investment in a Firm

[Feichtinger et al. '00, Gr./Semmler '04]

$x_1 =$ invested capital, $x_2 =$ investment, $u =$ change of investment

$$\dot{x}_1(t) = x_2(t) - \sigma x_1(t), \qquad \dot{x}_2(t) = u(t)$$

$$-\ell(x,u) = k_1\sqrt{x_1} - \frac{x_1}{1+k_2x_1^4} - c_1x_2 - \frac{c_2x_2^2}{2} - \frac{\alpha u^2}{2}, \quad \beta = e^{-0.04}$$

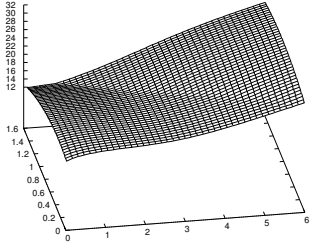
2d Example: Optimal Investment in a Firm

[Feichtinger et al. '00, Gr./Semmler '04]

x_1 = invested capital, x_2 = investment, u = change of investment

$$\dot{x}_1(t) = x_2(t) - \sigma x_1(t), \qquad \dot{x}_2(t) = u(t)$$

$$-\ell(x,u) = k_1\sqrt{x_1} - \frac{x_1}{1+k_2x_1^4} - c_1x_2 - \frac{c_2x_2^2}{2} - \frac{\alpha u^2}{2}, \quad \beta = e^{-0.04}$$



optimal value function

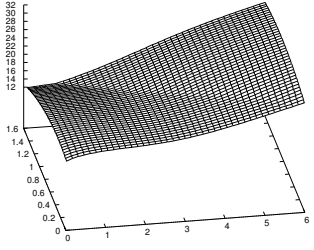
2d Example: Optimal Investment in a Firm

[Feichtinger et al. '00,
Gr./Semmler '04]

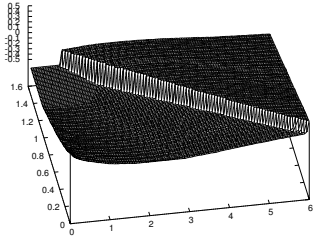
x_1 = invested capital, x_2 = investment, u = change of investment

$$\dot{x}_1(t) = x_2(t) - \sigma x_1(t), \qquad \dot{x}_2(t) = u(t)$$

$$-\ell(x,u) = k_1\sqrt{x_1} - \frac{x_1}{1+k_2x_1^4} - c_1x_2 - \frac{c_2x_2^2}{2} - \frac{\alpha u^2}{2}, \quad \beta = e^{-0.04}$$



optimal value function



optimal feedback law

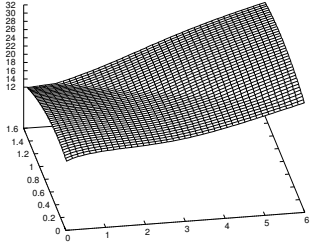
2d Example: Optimal Investment in a Firm

[Feichtinger et al. '00,
Gr./Semmler '04]

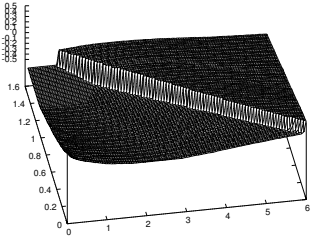
x_1 = invested capital, x_2 = investment, u = change of investment

$$\dot{x}_1(t) = x_2(t) - \sigma x_1(t), \qquad \dot{x}_2(t) = u(t)$$

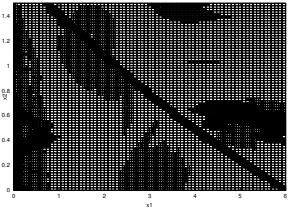
$$-\ell(x,u) = k_1\sqrt{x_1} - \frac{x_1}{1+k_2x_1^4} - c_1x_2 - \frac{c_2x_2^2}{2} - \frac{\alpha u^2}{2}, \quad \beta = e^{-0.04}$$



optimal value function



optimal feedback law



discretization grid

Semi-Lagrangian discretisation and the Bellman equation

The semi-discrete HJB equation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x,u) + e^{-\delta h} V_h(\tilde{x}_u(h,x)) \right\} = 0$$

is nothing but the Bellman equation

$$V(x) = \inf_{u \in \mathbb{U}} \left\{ \ell(x,u) + \beta V(g(x,u)) \right\}$$

if we write $g(x,u) = \tilde{x}_u(h,x)$, $\beta = e^{-\delta h}$, ℓ in place of $h\ell$, and V in place of V_h

Semi-Lagrangian discretisation and the Bellman equation

The semi-discrete HJB equation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x,u) + e^{-\delta h} V_h(\tilde{x}_u(h,x)) \right\} = 0$$

is nothing but the Bellman equation

$$V(x) = \inf_{u \in \mathbb{U}} \left\{ \ell(x,u) + \beta V(g(x,u)) \right\}$$

if we write $g(x,u) = \tilde{x}_u(h,x)$, $\beta = e^{-\delta h}$, ℓ in place of $h\ell$, and V in place of V_h

This has several implications:

- The feedback obtained from V_{hk} can be implemented as an approximately optimal feedback for the exact time-discretised (sampled) problem

Semi-Lagrangian discretisation and the Bellman equation

The semi-discrete HJB equation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x,u) + e^{-\delta h} V_h(\tilde{x}_u(h,x)) \right\} = 0$$

is nothing but the Bellman equation

$$V(x) = \inf_{u \in \mathbb{U}} \left\{ \ell(x,u) + \beta V(g(x,u)) \right\}$$

if we write $g(x,u) = \tilde{x}_u(h,x)$, $\beta = e^{-\delta h}$, ℓ in place of $h\ell$, and V in place of V_h

This has several implications:

- The feedback obtained from V_{hk} can be implemented as an approximately optimal feedback for the exact time-discretised (sampled) problem
  $\rightsquigarrow$ the technical difficulties in continuous-time are avoided

Semi-Lagrangian discretisation and the Bellman equation

The semi-discrete HJB equation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x,u) + e^{-\delta h} V_h(\tilde{x}_u(h,x)) \right\} = 0$$

is nothing but the Bellman equation

$$V(x) = \inf_{u \in \mathbb{U}} \left\{ \ell(x,u) + \beta V(g(x,u)) \right\}$$

if we write $g(x,u) = \tilde{x}_u(h,x)$, $\beta = e^{-\delta h}$, ℓ in place of $h\ell$, and V in place of V_h

This has several implications:

- The feedback obtained from V_{hk} can be implemented as an approximately optimal feedback for the exact time-discretised (sampled) problem
  $\rightsquigarrow$ the technical difficulties in continuous-time are avoided
- This also leads to rigorous error estimates

Semi-Lagrangian discretisation and the Bellman equation

The semi-discrete HJB equation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x,u) + e^{-\delta h} V_h(\tilde{x}_u(h,x)) \right\} = 0$$

is nothing but the Bellman equation

$$V(x) = \inf_{u \in \mathbb{U}} \left\{ \ell(x,u) + \beta V(g(x,u)) \right\}$$

if we write $g(x,u) = \tilde{x}_u(h,x)$, $\beta = e^{-\delta h}$, ℓ in place of $h\ell$, and V in place of V_h

This has several implications:

- The feedback obtained from V_{hk} can be implemented as an approximately optimal feedback for the exact time-discretised (sampled) problem
  $\rightsquigarrow$ the technical difficulties in continuous-time are avoided
- This also leads to rigorous error estimates
- The method in Step 2 can be applied to discrete-time problems

Semi-Lagrangian discretisation and the Bellman equation

The semi-discrete HJB equation

$$V_h(x) = \inf_{u \in \mathbb{U}} \left\{ h\ell(x,u) + e^{-\delta h} V_h(\tilde{x}_u(h,x)) \right\} = 0$$

is nothing but the Bellman equation

$$V(x) = \inf_{u \in \mathbb{U}} \left\{ \ell(x,u) + \beta V(g(x,u)) \right\}$$

if we write $g(x,u) = \tilde{x}_u(h,x)$, $\beta = e^{-\delta h}$, ℓ in place of $h\ell$, and V in place of V_h

This has several implications:

- The feedback obtained from V_{hk} can be implemented as an approximately optimal feedback for the exact time-discretised (sampled) problem
  $\rightsquigarrow$ the technical difficulties in continuous-time are avoided
- This also leads to rigorous error estimates
- The method in Step 2 can be applied to discrete-time problems
- Any method for discrete-time problems can be used in place of Step 2

Solving the Bellman equation

How to find $V \in \mathcal{F}_k$ satisfying

$$V(x) = \Pi_k \left(\inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\} \right) \qquad \text{for all } x \in C \ ?$$

First, since $\mathcal{F}_k$ is finite dimensional, it suffices to check the equation for x from a finite set $C_k \subset C$

In the classical approaches, V is then obtained iteratively:

- value iteration: $V_{i+1}(x) := \Pi_k \left(\inf_{u \in U} \{\ell(x,u) + \beta V_i(g(x,u))\} \right) \quad \forall\, x \in C_k$

Solving the Bellman equation

How to find $V \in \mathcal{F}_k$ satisfying

$$V(x) = \Pi_k \left(\inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\} \right) \qquad \text{for all } x \in C \ ?$$

First, since $\mathcal{F}_k$ is finite dimensional, it suffices to check the equation for x from a finite set $C_k \subset C$

In the classical approaches, V is then obtained iteratively:

- value iteration: $V_{i+1}(x) := \Pi_k \left(\inf_{u \in U} \{\ell(x,u) + \beta V_i(g(x,u))\} \right) \quad \forall\, x \in C_k$
- policy iteration:
  - choose F_i such that $u = F_i(x)$ minimises $\{\ell(x,u) + \beta V_i(g(x,u))\} \qquad \forall\, x \in C_k$
  - compute V_{i+1} satisfying $V_{i+1}(x) = \Pi_k \left(\ell(x,u) + \beta V_{i+1}(g(x,F_i(x))) \right) \quad \forall\, x \in C_k$

Solving the Bellman equation

How to find $V \in \mathcal{F}_k$ satisfying

$$V(x) = \Pi_k \left(\inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\} \right) \qquad \text{for all } x \in C \ ?$$

First, since $\mathcal{F}_k$ is finite dimensional, it suffices to check the equation for x from a finite set $C_k \subset C$

In the classical approaches, V is then obtained iteratively:

- value iteration: $V_{i+1}(x) := \Pi_k \left(\inf_{u \in U} \{\ell(x,u) + \beta V_i(g(x,u))\} \right) \quad \forall\, x \in C_k$
- policy iteration:
  - choose F_i such that $u = F_i(x)$ minimises $\{\ell(x,u) + \beta V_i(g(x,u))\} \qquad \forall\, x \in C_k$
  - compute V_{i+1} satisfying $V_{i+1}(x) = \Pi_k \left(\ell(x,u) + \beta V_{i+1}(g(x,F_i(x))) \right) \quad \forall\, x \in C_k$

Value iteration converges linearly, policy iteration quadratically

Q-Learning

In practice, a closed expression for $g(x, u)$ may not always be available. For instance, we may only be able make experiments in order to determine $x^+ = g(x, u)$ from x and u, but do not have a mathematical model

Q-Learning

In practice, a closed expression for $g(x,u)$ may not always be available. For instance, we may only be able make experiments in order to determine $x^+ = g(x,u)$ from x and u, but do not have a mathematical model

Remedy: Compute $Q(x,u) = \ell(x,u) + \beta V(g(x,u))$ instead of $V(x)$

Q-Learning

In practice, a closed expression for $g(x,u)$ may not always be available. For instance, we may only be able make experiments in order to determine $x^+ = g(x,u)$ from x and u, but do not have a mathematical model

Remedy: Compute $Q(x,u) = \ell(x,u) + \beta V(g(x,u))$ instead of $V(x)$

characterization of the optimal feedback law

$$\ell(x, F^\star(x)) + \beta V(g(x, F^\star(x))) = \inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\}$$

Q-Learning

In practice, a closed expression for $g(x,u)$ may not always be available. For instance, we may only be able make experiments in order to determine $x^+ = g(x,u)$ from x and u, but do not have a mathematical model

Remedy: Compute $Q(x,u) = \ell(x,u) + \beta V(g(x,u))$ instead of $V(x)$

If Q is known, the characterization of the optimal feedback law

$$\ell(x, F^\star(x)) + \beta V(g(x, F^\star(x))) = \inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\}$$

changes to

$$Q(x, F^\star(x)) = \inf_{u \in \mathbb{U}} Q(x,u)$$

Q-Learning

In practice, a closed expression for $g(x,u)$ may not always be available. For instance, we may only be able make experiments in order to determine $x^+ = g(x,u)$ from x and u, but do not have a mathematical model

Remedy: Compute $Q(x,u) = \ell(x,u) + \beta V(g(x,u))$ instead of $V(x)$

If Q is known, the characterization of the optimal feedback law

$$\ell(x, F^\star(x)) + \beta V(g(x, F^\star(x))) = \inf_{u \in \mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\}$$

changes to

$$Q(x, F^\star(x)) = \inf_{u \in \mathbb{U}} Q(x,u)$$

$\rightsquigarrow$ $F^\star$ can be determined without knowing g

Q-Learning

In practice, a closed expression for $g(x,u)$ may not always be available. For instance, we may only be able make experiments in order to determine $x^+ = g(x,u)$ from x and u, but do not have a mathematical model

Remedy: Compute $Q(x,u) = \ell(x,u) + \beta V(g(x,u))$ instead of $V(x)$

If Q is known, the characterization of the optimal feedback law

$$\ell(x, F^\star(x)) + \beta V(g(x, F^\star(x))) = \inf_{u\in\mathbb{U}} \{\ell(x,u) + \beta V(g(x,u))\}$$

changes to

$$Q(x, F^\star(x)) = \inf_{u\in\mathbb{U}} Q(x,u)$$

$\rightsquigarrow$ $F^\star$ can be determined without knowing g

An iterative algorithm for obtaining Q is classical Q-learning:

$Q_{i+1}(x_i,u_i) := \ell(x_i,u_i) + \beta \inf_{u\in U} Q_i(x_i^+,u)$ for a sequence of experimental or simulated data x_i, u_i, x_i^+ with $x_i^+ = g(x_i,u_i)$. Often, $x_{i+1} = x_i^+$ is chosen

Deep Reinforcement Learning

Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality

Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

100 discretisation points if $n = 2$

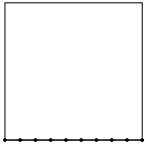
Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

100 discretisation points if $n = 2$



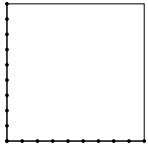
Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

100 discretisation points if $n = 2$



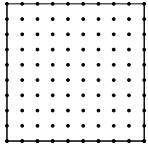
Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

100 discretisation points if $n = 2$



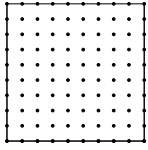
Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

100	discretisation points if $n = 2$
100 000	discretisation points if $n = 5$



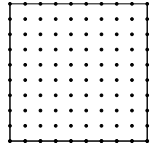
Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

100	discretisation points if $n = 2$
100 000	discretisation points if $n = 5$
1 000 000 000	discretisation points if $n = 9$



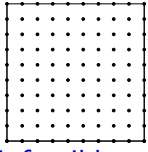
Curse of dimensionality

Besides various technical issues, the main conceptual challenge of numerically approximating V is the notorious curse of dimensionality:

If we solve the Bellman equation with grid- or mesh based spatial approximation techniques on $C \subset \mathbb{R}^n$, then the numerical effort grows exponentially with the space dimension n

Example: Assume 10 discretisation steps are needed in each coordinate direction. Then we obtain

100	discretisation points if $n = 2$
100 000	discretisation points if $n = 5$
1 000 000 000	discretisation points if $n = 9$



The problem quickly becomes computationally infeasible

Deep Learning approaches for computing V or Q

In deep learning approaches, there is no explicit space discretization via a grid or mesh. Rather the functions V or Q are approximations by deep Neural Networks

For those not familiar with deep neural networks, we provide a brief introduction

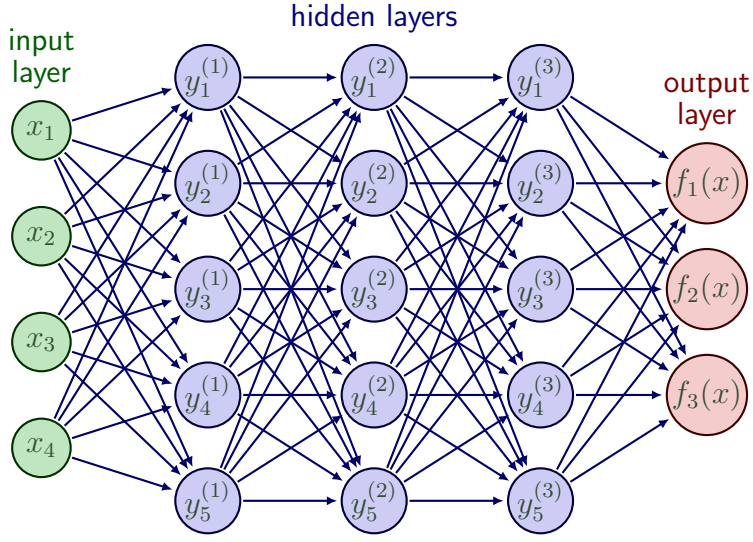
Excursion: Neural Networks

Quiz

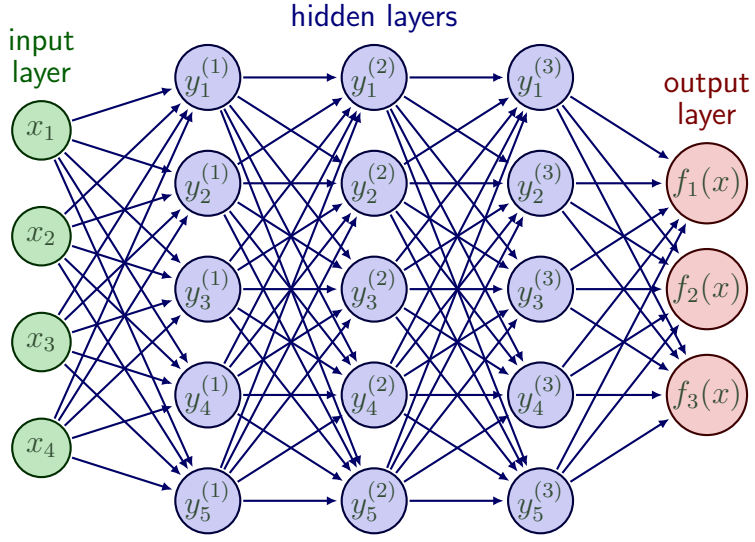
Which of the following objects does best describe the nature of a neural network?

- A matrix
- A function
- A sequence
- An equation

Neural Networks

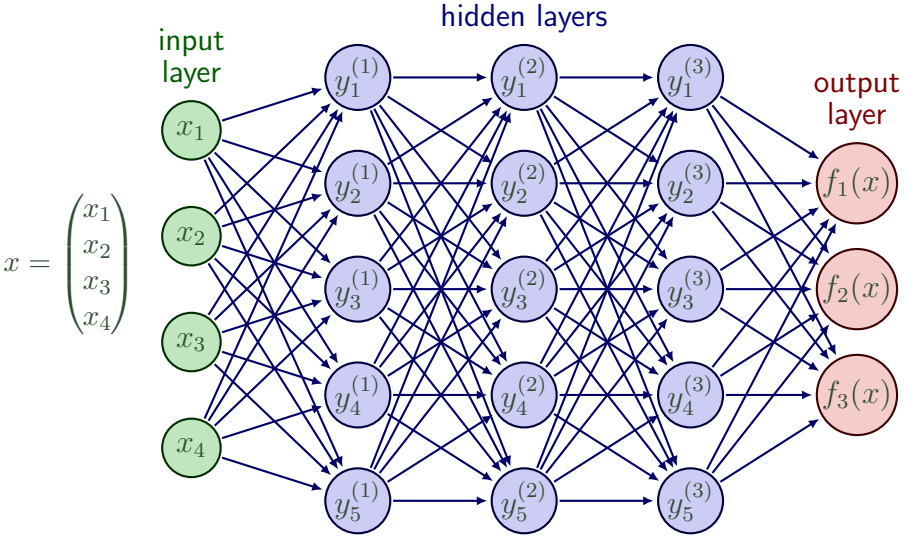


Neural Networks



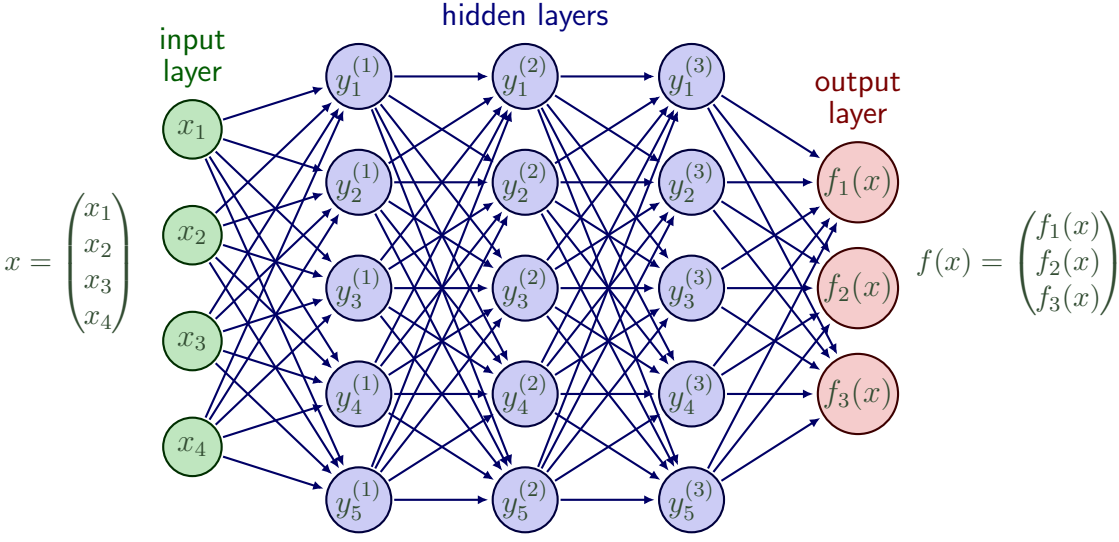
An NN represents a function $f(x)$

Neural Networks



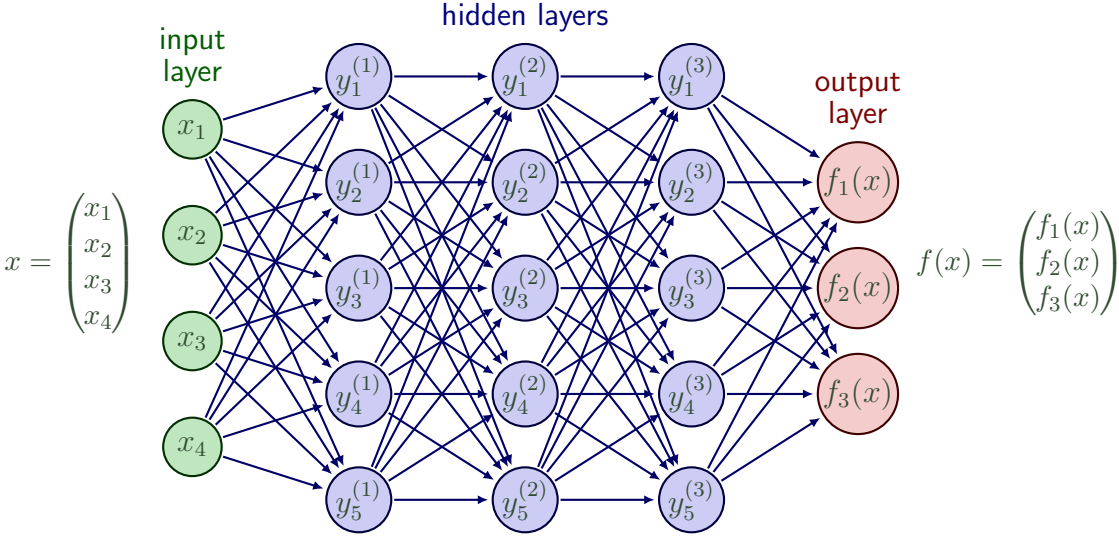
An NN represents a function $f(x)$

Neural Networks



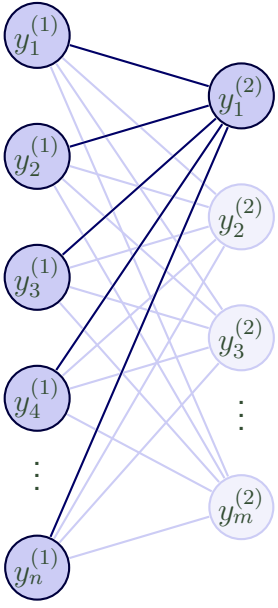
An NN represents a function $f(x)$

Neural Networks

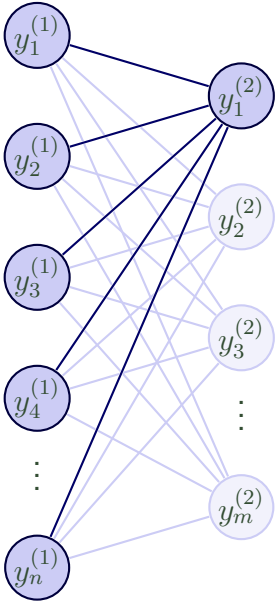


An NN represents a function $f(x)$, here $f : \mathbb{R}^4 \to \mathbb{R}^3$

How are the values $y_i^{(l)}$ calculated?

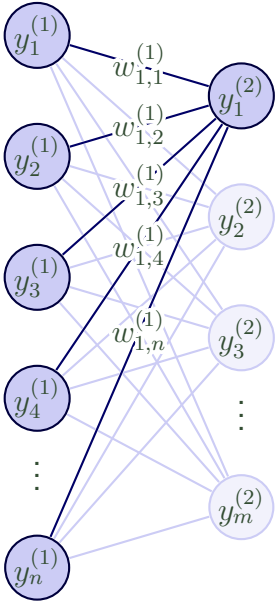


How are the values $y_i^{(l)}$ calculated?



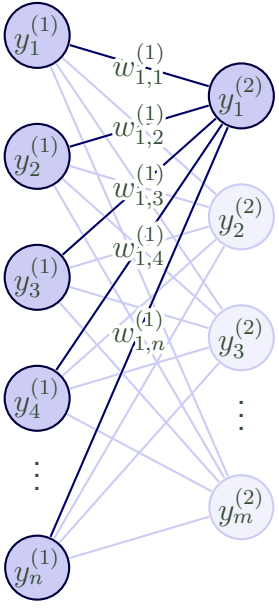
$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

How are the values $y_i^{(l)}$ calculated?



$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

How are the values $y_i^{(l)}$ calculated?

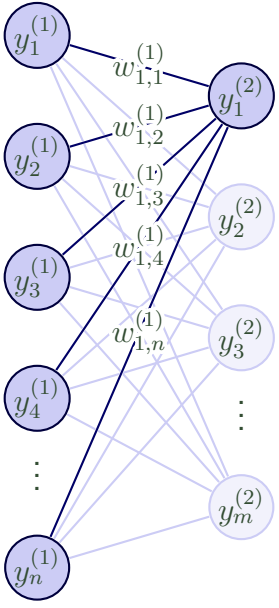


$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

$w_{i,j}^{(l)}$ = weights

$b_i^{(l)}$ = bias terms

How are the values $y_i^{(l)}$ calculated?



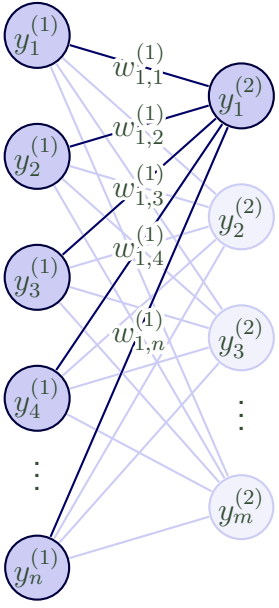
$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

$$y_i^{(l+1)} = \sigma\left(\sum_{j=1}^{n} w_{i,j}^{(l)} y_j^{(l)} + b_i^{(l)}\right)$$

$w_{i,j}^{(l)}$ = weights

$b_i^{(l)}$ = bias terms

How are the values $y_i^{(l)}$ calculated?



$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

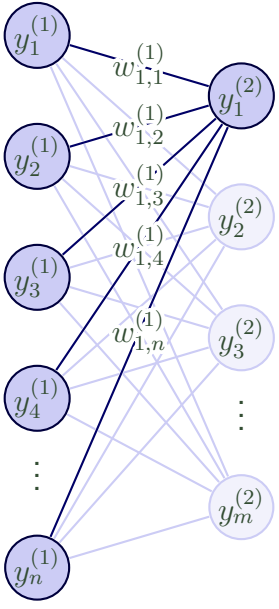
$$y_i^{(l+1)} = \sigma\left(\sum_{j=1}^{n} w_{i,j}^{(l)} y_j^{(l)} + b_i^{(l)}\right)$$

$w_{i,j}^{(l)}$ = weights

$b_i^{(l)}$ = bias terms

$\sigma : \mathbb{R} \to \mathbb{R}$ is called activation function

How are the values $y_i^{(l)}$ calculated?



$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

$$y_i^{(l+1)} = \sigma\left(\sum_{j=1}^{n} w_{i,j}^{(l)} y_j^{(l)} + b_i^{(l)}\right)$$

$w_{i,j}^{(l)}$ = weights

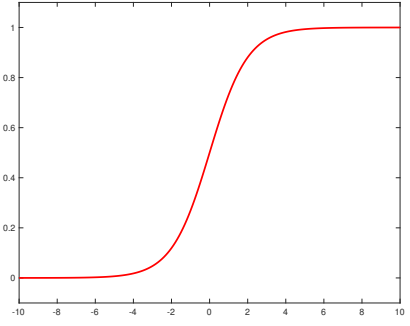
$b_i^{(l)}$ = bias terms

$\sigma : \mathbb{R} \to \mathbb{R}$ is called activation function

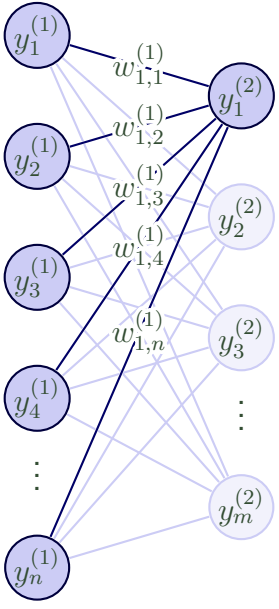
Examples:

sigmoid:

$\sigma(y) = \frac{1}{1+e^{-y}}$



How are the values $y_i^{(l)}$ calculated?



$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

$$y_i^{(l+1)} = \sigma\left(\sum_{j=1}^{n} w_{i,j}^{(l)} y_j^{(l)} + b_i^{(l)}\right)$$

$w_{i,j}^{(l)}$ = weights

$b_i^{(l)}$ = bias terms

$\sigma : \mathbb{R} \to \mathbb{R}$ is called activation function

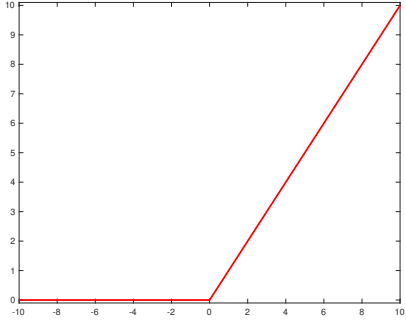
Examples:

sigmoid:

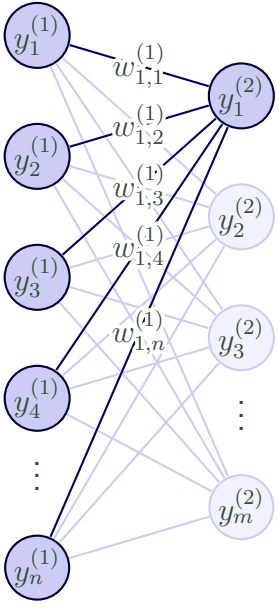
$\sigma(y) = \frac{1}{1+e^{-y}}$

ReLU:

$\sigma(y) = \max\{0, y\}$



How are the values $y_i^{(l)}$ calculated?



$$y_1^{(2)} = \sigma\left(w_{1,1}^{(1)} y_1^{(1)} + w_{1,2}^{(1)} y_2^{(1)} + \ldots + w_{1,n}^{(1)} y_n^{(1)} + b_1^{(1)}\right)$$

$$y_i^{(l+1)} = \sigma\left(\sum_{j=1}^{n} w_{i,j}^{(l)} y_j^{(l)} + b_i^{(l)}\right)$$

$w_{i,j}^{(l)}$ = weights

$b_i^{(l)}$ = bias terms

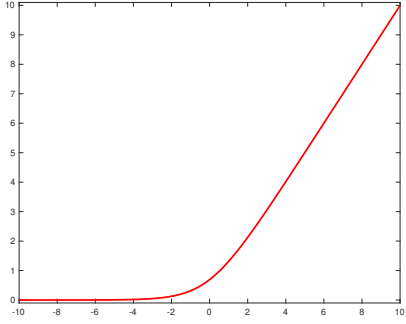
$\sigma : \mathbb{R} \to \mathbb{R}$ is called activation function

Examples:

sigmoid:
$\sigma(y) = \frac{1}{1+e^{-y}}$

ReLU:
$\sigma(y) = \max\{0, y\}$

softplus:
$\sigma(y) = \log(1 + e^y)$



Training

Training describes the process of finding the right weights $w_{i,j}^{(l)}$ and bias terms $b_i^{(l)}$ such that the network approximates the desired function φ_d

This is done using a training algorithm based on optimisation methods

Training

Training describes the process of finding the right weights $w_{i,j}^{(l)}$ and bias terms $b_i^{(l)}$ such that the network approximates the desired function φ_d

This is done using a training algorithm based on optimisation methods

A loss function determines the objective of the optimisation

Training

Training describes the process of finding the right weights $w_{i,j}^{(l)}$ and bias terms $b_i^{(l)}$ such that the network approximates the desired function φ_d

This is done using a training algorithm based on optimisation methods

A loss function determines the objective of the optimisation

Typical loss function when data x_i and $y_i = \varphi_d(x_i)$ of the desired function is known:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\varphi(x_i, \theta) - y_i)^2$$

$x_1, \ldots, x_N$ = function arguments $\quad \theta = (w_{i,j}^{(l)}, b_i^{(l)})_{i,j,l}$ = parameters of the NN

$y_1, \ldots, y_N$ = function values of φ_d $\quad \varphi(x, \theta)$ = function represented by the NN

Training

Training describes the process of finding the right weights $w_{i,j}^{(l)}$ and bias terms $b_i^{(l)}$ such that the network approximates the desired function φ_d

This is done using a training algorithm based on optimisation methods

A loss function determines the objective of the optimisation

Typical loss function when data x_i and $y_i = \varphi_d(x_i)$ of the desired function is known:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\varphi(x_i, \theta) - y_i)^2$$

$x_1, \ldots, x_N$ = function arguments $\quad \theta = (w_{i,j}^{(l)}, b_i^{(l)})_{i,j,l}$ = parameters of the NN

$y_1, \ldots, y_N$ = function values of φ_d $\quad \varphi(x, \theta)$ = function represented by the NN

This is the standard loss function for regression ("supervised learning"), i.e., for learning a function with prescribed values at the data points

Stochastic Gradient Descent

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\varphi(x_i, \theta) - y_i)^2$$

Stochastic Gradient Descent

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\varphi(x_i, \theta) - y_i)^2$$

Minimising $L(\theta)$ is usually done via so-called Gradient Descent

These methods compute a minimizing θ iteratively via

$$\theta_{i+1} := \theta_i - \alpha_i \nabla L(\theta_i)$$

for some suitably chosen step size $\alpha_i > 0$ ("learning rate")

Stochastic Gradient Descent

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\varphi(x_i, \theta) - y_i)^2$$

Minimising $L(\theta)$ is usually done via so-called Gradient Descent

These methods compute a minimizing θ iteratively via

$$\theta_{i+1} := \theta_i - \alpha_i \nabla L(\theta_i)$$

for some suitably chosen step size $\alpha_i > 0$ ("learning rate")

These are very efficient to implement, since $\frac{d}{d\theta} f(x, \theta)$ can be computed iteratively for NNs via Backpropagation (essentially: the chain rule)

Stochastic Gradient Descent

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\varphi(x_i,\theta) - y_i)^2$$

Minimising $L(\theta)$ is usually done via so-called Gradient Descent

These methods compute a minimizing θ iteratively via

$$\theta_{i+1} := \theta_i - \alpha_i \nabla L(\theta_i)$$

for some suitably chosen step size $\alpha_i > 0$ ("learning rate")

These are very efficient to implement, since $\frac{d}{d\theta} f(x,\theta)$ can be computed iteratively for NNs via Backpropagation (essentially: the chain rule)

However, for huge data sets, the computation of ∇L involves a huge sum

Stochastic Gradient Descent

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\varphi(x_i,\theta) - y_i)^2$$

Minimising $L(\theta)$ is usually done via so-called Gradient Descent

These methods compute a minimizing θ iteratively via

$$\theta_{i+1} := \theta_i - \alpha_i \nabla L(\theta_i)$$

for some suitably chosen step size $\alpha_i > 0$ ("learning rate")

These are very efficient to implement, since $\frac{d}{d\theta} f(x,\theta)$ can be computed iteratively for NNs via Backpropagation (essentially: the chain rule)

However, for huge data sets, the computation of ∇L involves a huge sum

Remedy: In each gradient step, compute only the derivative for a randomly selected subset of (x_i, y_i)

Stochastic Gradient Descent

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\varphi(x_i,\theta) - y_i)^2$$

Minimising $L(\theta)$ is usually done via so-called Gradient Descent

These methods compute a minimizing θ iteratively via

$$\theta_{i+1} := \theta_i - \alpha_i \nabla L(\theta_i)$$

for some suitably chosen step size $\alpha_i > 0$ ("learning rate")

These are very efficient to implement, since $\frac{d}{d\theta} f(x,\theta)$ can be computed iteratively for NNs via Backpropagation (essentially: the chain rule)

However, for huge data sets, the computation of ∇L involves a huge sum

Remedy: In each gradient step, compute only the derivative for a randomly selected subset of (x_i, y_i) $\rightarrow$ "Stochastic Gradient Descent"

Stochastic Gradient Descent

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\varphi(x_i,\theta) - y_i)^2$$

Minimising $L(\theta)$ is usually done via so-called Gradient Descent

These methods compute a minimizing θ iteratively via

$$\theta_{i+1} := \theta_i - \alpha_i \nabla L(\theta_i)$$

for some suitably chosen step size $\alpha_i > 0$ ("learning rate")

These are very efficient to implement, since $\frac{d}{d\theta} f(x,\theta)$ can be computed iteratively for NNs via Backpropagation (essentially: the chain rule)

However, for huge data sets, the computation of ∇L involves a huge sum

Remedy: In each gradient step, compute only the derivative for a randomly selected subset of (x_i, y_i) $\rightarrow$ "Stochastic Gradient Descent"

$\rightarrow$ Example: Learning a sine function

Deep Learning approaches for computing V or Q

In deep learning approaches, V_{NN} or Q_{NN} are approximations to V or Q by deep NNs (DNNs)

Deep Learning approaches for computing V or Q

In deep learning approaches, V_{NN} or Q_{NN} are approximations to V or Q by deep NNs (DNNs) and a possibility for choosing the loss function is

$$\frac{1}{N}\sum_{j=1}^{N} |V_{NN}(x_j) - \inf_{u\in U}\{\ell(x_j,u) + \beta V_{NN}(g(x_j,u))\}|^2$$

for data x_j or, respectively,

$$\frac{1}{N}\sum_{j=1}^{N} |Q_{NN}(x_j,u_j) - \ell(x_j,u_j) - \beta \inf_{u\in U} Q_{NN}(x_j^+,u))|^2$$

for data x_j, u_j, x_j^+, $j = 1,\ldots,N$

Deep Learning approaches for computing V or Q

In deep learning approaches, V_{NN} or Q_{NN} are approximations to V or Q by deep NNs (DNNs) and a possibility for choosing the loss function is

$$\frac{1}{N}\sum_{j=1}^{N} |V_{NN}(x_j) - \inf_{u\in U}\{\ell(x_j,u) + \beta V_{NN}(g(x_j,u))\}|^2$$

for data x_j or, respectively,

$$\frac{1}{N}\sum_{j=1}^{N} |Q_{NN}(x_j,u_j) - \ell(x_j,u_j) - \beta \inf_{u\in U} Q_{NN}(x_j^+,u))|^2$$

for data x_j, u_j, x_j^+, $j = 1,\ldots,N$

"Loss functions penalize violations of Bellman equation in data points"

Deep Learning approaches for computing V or Q

In deep learning approaches, V_{NN} or Q_{NN} are approximations to V or Q by deep NNs (DNNs) and a possibility for choosing the loss function is

$$\frac{1}{N}\sum_{j=1}^{N} |V_{NN}(x_j) - \inf_{u\in U}\{\ell(x_j,u) + \beta V_{NN}(g(x_j,u))\}|^2$$

for data x_j or, respectively,

$$\frac{1}{N}\sum_{j=1}^{N} |Q_{NN}(x_j,u_j) - \ell(x_j,u_j) - \beta \inf_{u\in U} Q_{NN}(x_j^+,u))|^2$$

for data x_j, u_j, x_j^+, $j = 1,\ldots,N$

"Loss functions penalize violations of Bellman equation in data points"

In continuous time, the Hamilton-Jacobi-Bellman-based loss function for V reads

$$\frac{1}{N}\sum_{j=1}^{N} |\delta V_{NN}(x_j) + \inf_{u\in U}\{\ell(x_j,u) + DV_{NN}(x_j)f(x_j,u))\}|^2$$

Deep Learning approaches, implementation details

Minimising these loss functions is a hard problem

Deep Learning approaches, implementation details

Minimising these loss functions is a hard problem

Remedies:

- V_{NN} or Q_{NN} are computed iteratively, e.g., using the loss function

$$L(V_{NN,i+1}, x) = \big(V_{NN,i+1}(x) - \inf_{u \in U} \{\ell(x,u) + \beta V_{NN,i}(g(x,u))\} \big)^2$$

Deep Learning approaches, implementation details

Minimising these loss functions is a hard problem

Remedies:

- V_{NN} or Q_{NN} are computed iteratively, e.g., using the loss function

$$L(V_{NN,i+1}, x) = \Big(\underbrace{V_{NN,i+1}(x)}_{=\varphi(x,\theta)} - \underbrace{\inf_{u \in U} \{\ell(x,u) + \beta V_{NN,i}(g(x,u))\}}_{=\varphi_d(x)} \Big)^2$$

  $\rightsquigarrow$ can be implemented as a standard regression problem

Deep Learning approaches, implementation details

Minimising these loss functions is a hard problem

Remedies:

- V_{NN} or Q_{NN} are computed iteratively, e.g., using the loss function

$$L(V_{NN,i+1}, x) = \Big(\underbrace{V_{NN,i+1}(x)}_{=\varphi(x,\theta)} - \underbrace{\inf_{u \in U} \{\ell(x,u) + \beta V_{NN,i}(g(x,u))\}}_{=\varphi_d(x)} \Big)^2$$

  $\rightsquigarrow$ can be implemented as a standard regression problem

- Instead of computing the "inf" in L, an approximate minimizer $F_{NN,i}$ is stored in a second Neural Network $\rightsquigarrow$ "actor-critic method"

Deep Learning approaches, implementation details

Minimising these loss functions is a hard problem

Remedies:

- V_{NN} or Q_{NN} are computed iteratively, e.g., using the loss function

$$L(V_{NN,i+1}, x) = \Big(\underbrace{V_{NN,i+1}(x)}_{=\varphi(x,\theta)} - \underbrace{\inf_{u \in U} \{\ell(x,u) + \beta V_{NN,i}(g(x,u))\}}_{=\varphi_d(x)} \Big)^2$$

  $\rightsquigarrow$ can be implemented as a standard regression problem

- Instead of computing the "inf" in L, an approximate minimizer $F_{NN,i}$ is stored in a second Neural Network $\rightsquigarrow$ "actor-critic method"
- Once an approximation to F is stored, the fact that F optimizes the cost can be used in the design of the loss function $\rightsquigarrow$ "policy gradient method"

Deep Learning approaches, implementation details

- Data points are often generated along trajectories, as long as constraints are violated or a pre-defined goal is reached $\rightsquigarrow$ "episodes"

Deep Learning approaches, implementation details

- Data points are often generated along trajectories, as long as constraints are violated or a pre-defined goal is reached $\rightsquigarrow$ "episodes"
- Generated data is used several times

Deep Learning approaches, implementation details

- Data points are often generated along trajectories, as long as constraints are violated or a pre-defined goal is reached $\rightsquigarrow$ "episodes"
- Generated data is used several times
- For the iterative loss function

$$\frac{1}{N}\sum_{j=1}^{N} |V_{NN,i+1}(x_j) - \inf_{u\in U} \{\ell(x_j,u) + \beta V_{NN,i}(g(x_j,u))\}|^2$$

  a new iteration is performed after every couple of episodes

Deep Learning approaches, implementation details

- Data points are often generated along trajectories, as long as constraints are violated or a pre-defined goal is reached $\rightsquigarrow$ "episodes"
- Generated data is used several times
- For the iterative loss function

$$\frac{1}{N}\sum_{j=1}^{N} |V_{NN,i+1}(x_j) - \inf_{u\in U} \{\ell(x_j,u) + \beta V_{NN,i}(g(x_j,u))\}|^2$$

  a new iteration is performed after every couple of episodes
- The controls for generating the x_j are chosen as minimisers of $\ell + \beta V_{NN,i}$ ("exploitation"), with randomly generated exceptions ("exploration")

Deep Learning approaches, implementation details

- Data points are often generated along trajectories, as long as constraints are violated or a pre-defined goal is reached $\rightsquigarrow$ "episodes"
- Generated data is used several times
- For the iterative loss function

$$\frac{1}{N}\sum_{j=1}^{N} |V_{NN,i+1}(x_j) - \inf_{u\in U}\{\ell(x_j,u) + \beta V_{NN,i}(g(x_j,u))\}|^2$$

  a new iteration is performed after every couple of episodes
- The controls for generating the x_j are chosen as minimisers of $\ell + \beta V_{NN,i}$ ("exploitation"), with randomly generated exceptions ("exploration")
- On `https://pylessons.com/` or on `https://spinningup.openai.com` templates for Deep Reinforcement Learning can be obtained

Deep Learning approaches, implementation details

- Data points are often generated along trajectories, as long as constraints are violated or a pre-defined goal is reached $\rightsquigarrow$ "episodes"
- Generated data is used several times
- For the iterative loss function

$$\frac{1}{N}\sum_{j=1}^{N} |V_{NN,i+1}(x_j) - \inf_{u\in U}\{\ell(x_j,u) + \beta V_{NN,i}(g(x_j,u))\}|^2$$

  a new iteration is performed after every couple of episodes
- The controls for generating the x_j are chosen as minimisers of $\ell + \beta V_{NN,i}$ ("exploitation"), with randomly generated exceptions ("exploration")
- On `https://pylessons.com/` or on `https://spinningup.openai.com` templates for Deep Reinforcement Learning can be obtained

We will see in the Part 5 of this course whether Deep Learning can really do better than grid- or meshed-based approaches

Methods computing open-loop optimal controls

How to solve OCPs for open-loop controls?

Core challenge: How to compute the function $\mathbf{u}^\star(\cdot)$?

How to solve OCPs for open-loop controls?

Core challenge: How to compute the function $\mathbf{u}^\star(\cdot)$?

Two main options:

How to solve OCPs for open-loop controls?

Core challenge: How to compute the function $\mathbf{u}^\star(\cdot)$?

Two main options:

1. Apply PMP to obtain necessary optimality conditions (NCOs)

2. Solve NCO differential equations by discretization in time

$\rightsquigarrow$ Indirect solution methods

How to solve OCPs for open-loop controls?

Core challenge: How to compute the function $\mathbf{u}^\star(\cdot)$?

Two main options:

1. Apply PMP to obtain necessary optimality conditions (NCOs)
2. Solve NCO differential equations by discretization in time

$\rightsquigarrow$ Indirect solution methods

1. Discretize OCP in time to obtain a finite-dimensional nonlinear optimization problem (NLP)
2. Solve NLP by suitable optimisation algorithm

$\rightsquigarrow$ Direct solution methods

How to solve OCPs for open-loop controls?

Core challenge: How to compute the function $\mathbf{u}^\star(\cdot)$?

Two main options:

1. Apply PMP to obtain necessary optimality conditions (NCOs)
2. Solve NCO differential equations by discretization in time

$\rightsquigarrow$ Indirect solution methods

1. Discretize OCP in time to obtain a finite-dimensional nonlinear optimization problem (NLP)
2. Solve NLP by suitable optimisation algorithm

$\rightsquigarrow$ Direct solution methods

"First optimise, then discretise"

How to solve OCPs for open-loop controls?

Core challenge: How to compute the function $\mathbf{u}^\star(\cdot)$?

Two main options:

1. Apply PMP to obtain necessary optimality conditions (NCOs)
2. Solve NCO differential equations by discretization in time

$\rightsquigarrow$ Indirect solution methods

"First optimise, then discretise"

1. Discretize OCP in time to obtain a finite-dimensional nonlinear optimization problem (NLP)
2. Solve NLP by suitable optimisation algorithm

$\rightsquigarrow$ Direct solution methods

"First discretise, then optimise"

Methods based on Pontryagin's Maximum Principle
"Indirect Solution Methods"

Basic idea of indirect shooting

"First optimise, then discretise"

$$\min_{\mathbf{u}(\cdot)} \quad \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt + L(t_f, x(t_f))$$

subject to: (P)

$$\begin{aligned} \dot{x} &= f(t, x, \mathbf{u}), \quad x(t_0) = x_0 \\ \mathbf{u}(\cdot) &\in L_\infty([t_0, t_f], \mathbb{U}) \\ 0 &= \Psi(t_f, x(t_f)) \end{aligned}$$

Basic idea of indirect shooting

"First optimise, then discretise"

$$\min_{\mathbf{u}(\cdot)} \quad \int_{t_0}^{t_f} \ell(t, x(t), \mathbf{u}(t))dt + L(t_f, x(t_f))$$

subject to: (P)

$$\begin{aligned} \dot{x} &= f(t, x, \mathbf{u}), \quad x(t_0) = x_0 \\ \mathbf{u}(\cdot) &\in L_\infty([t_0, t_f], \mathbb{U}) \\ 0 &= \Psi(t_f, x(t_f)) \end{aligned}$$

NCOs:

$$\begin{aligned} \dot{x}^\star = &\quad H_\lambda(t, x^\star, \mathbf{u}^\star, \lambda^\star), \quad x^\star(t_0) = x_0 \\ \dot{\lambda}^\star = &-H_x(t, x^\star, \mathbf{u}^\star, \lambda^\star), \\ \lambda^\star(t_f) = &\quad L_x(t_f, x^\star(t_f)) + (\nu^\star)^\top \Psi_x(t_f, x^\star(t_f)), \\ 0 = &\quad H_u(t, x^\star, \mathbf{u}^\star, \lambda^\star), \\ 0 = &\quad \Psi(t_f, x^\star(t_f)) \end{aligned}$$

Basic idea of indirect shooting

Main difficulty: The NCOs contain a boundary value problem

Basic idea of indirect shooting

Main difficulty: The NCOs contain a boundary value problem

Idea: Split NCOs into two parts:

- NCOs enforced (approximately) at each iteration
- NCOs enforced (approximately) upon convergence

Basic idea of indirect shooting

Main difficulty: The NCOs contain a boundary value problem

Idea: Split NCOs into two parts:

- NCOs enforced (approximately) at each iteration
- NCOs enforced (approximately) upon convergence

NCOs:

$$\begin{aligned}
\dot{x}^\star &= H_\lambda(t, x^\star, \mathbf{u}^\star, \lambda^\star), \quad x^\star(t_0) = x_0 \\
\dot{\lambda}^\star &= -H_x(t, x^\star, \mathbf{u}^\star, \lambda^\star), \\
\lambda^\star(t_f) &= L_x(t_f, x^\star(t_f)) + (\nu^\star)^\top \Psi_x(t_f, x^\star(t_f)), \\
0 &= H_u(t, x^\star, \mathbf{u}^\star, \lambda^\star), \\
0 &= \Psi(t_f, x^\star(t_f))
\end{aligned}$$

A basic indirect shooting algorithm

Choose $\varepsilon > 0$. Guess λ_0^0, ν^0. Set $k = 0$

1. Solve numerically from t_0 to t_f

$$\begin{aligned} \dot{x}^k &= \quad H_\lambda(t, x^k, \mathbf{u}^k, \lambda^k), \quad x(t_0)^k = x_0 \\ \dot{\lambda}^k &= -H_x(t, x^k, \mathbf{u}^k, \lambda^k), \quad \lambda(t_0)^k = \lambda_0^k \\ 0 &= \quad H_u(t, x^k, \mathbf{u}^k, \lambda^k) \end{aligned}$$

A basic indirect shooting algorithm

Choose $\varepsilon > 0$. Guess λ_0^0, ν^0. Set $k = 0$

1. Solve numerically from t_0 to t_f

$$\begin{aligned} \dot{x}^k &= \quad H_\lambda(t, x^k, \mathbf{u}^k, \lambda^k), \quad x(t_0)^k = x_0 \\ \dot{\lambda}^k &= -H_x(t, x^k, \mathbf{u}^k, \lambda^k), \quad \lambda(t_0)^k = \lambda_0^k \\ 0 &= \quad H_u(t, x^k, \mathbf{u}^k, \lambda^k) \end{aligned}$$

2. Compute defect of transversality and terminal conditions:

$$\mathcal{F}(\lambda_0^k, \nu^k) \doteq \begin{pmatrix} \lambda(t_f)^k - \quad L_x(t_f, x(t_f)^k) - (\nu^k)^\top \Psi_x(t_f, x(t_f)^k) \\ \Psi(t_f, x(t_f)^k) \end{pmatrix}$$

3. If $\|\mathcal{F}(\lambda_0^k, \nu^k)\| \leq \varepsilon \quad \rightarrow \quad$ STOP

A basic indirect shooting algorithm

Choose $\varepsilon > 0$. Guess λ_0^0, ν^0. Set $k = 0$

1. Solve numerically from t_0 to t_f

$$\begin{aligned} \dot{x}^k &= \quad H_\lambda(t, x^k, \mathbf{u}^k, \lambda^k), \quad x(t_0)^k = x_0 \\ \dot{\lambda}^k &= -H_x(t, x^k, \mathbf{u}^k, \lambda^k), \quad \lambda(t_0)^k = \lambda_0^k \\ 0 &= \quad H_u(t, x^k, \mathbf{u}^k, \lambda^k) \end{aligned}$$

2. Compute defect of transversality and terminal conditions:

$$\mathcal{F}(\lambda_0^k, \nu^k) \doteq \begin{pmatrix} \lambda(t_f)^k - \quad L_x(t_f, x(t_f)^k) - (\nu^k)^\top \Psi_x(t_f, x(t_f)^k) \\ \Psi(t_f, x(t_f)^k) \end{pmatrix}$$

3. If $\|\mathcal{F}(\lambda_0^k, \nu^k)\| \leq \varepsilon \quad \rightarrow \quad$ STOP
4. Update λ_0^k, ν^k to enforce $\mathcal{F}(\lambda_0^k, \nu^k) \to 0$. $k \leftarrow k + 1$. GOTO Step 1

How to do Step 4?

- We want

$$\lim_{k\to\infty} \mathcal{F}(\lambda_0^k, \nu^k) = 0.$$

This is a root finding problem, e.g., we can apply Newton's method[12]

[12] Or any quasi Newton method, also stepsize parameters could be introduced.

How to do Step 4?

- We want

$$\lim_{k\to\infty} \mathcal{F}(\lambda_0^k, \nu^k) = 0.$$

  This is a root finding problem, e.g., we can apply Newton's method[12]

- Compute defect gradients $D_{\lambda_0^k}\mathcal{F}$ and $D_{\nu^k}\mathcal{F}$ and solve the Newton step

$$\begin{pmatrix} D_{\lambda_0^k}\mathcal{F} & D_{\nu^k}\mathcal{F} \end{pmatrix} \begin{pmatrix} \Delta\lambda^k \\ \Delta\nu^k \end{pmatrix} = -\mathcal{F}(\lambda_0^k, \nu^k).$$

[12] Or any quasi Newton method, also stepsize parameters could be introduced.

How to do Step 4?

- We want

$$\lim_{k\to\infty} \mathcal{F}(\lambda_0^k, \nu^k) = 0.$$

  This is a root finding problem, e.g., we can apply Newton's method[12]

- Compute defect gradients $D_{\lambda_0^k}\mathcal{F}$ and $D_{\nu^k}\mathcal{F}$ and solve the Newton step

$$\begin{pmatrix} D_{\lambda_0^k}\mathcal{F} & D_{\nu^k}\mathcal{F} \end{pmatrix} \begin{pmatrix} \Delta\lambda^k \\ \Delta\nu^k \end{pmatrix} = -\mathcal{F}(\lambda_0^k, \nu^k).$$

- Update

$$\begin{pmatrix} \lambda_0^{k+1} \\ \nu^{k+1} \end{pmatrix} = \begin{pmatrix} \lambda_0^k \\ \nu^k \end{pmatrix} + \begin{pmatrix} \Delta\lambda^k \\ \Delta\nu^k \end{pmatrix}.$$

[12] Or any quasi Newton method, also stepsize parameters could be introduced.

Example

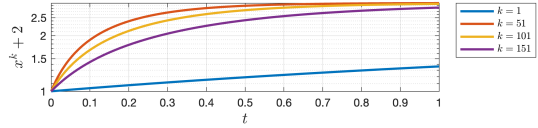
$$\min_{\mathbf{u}(\cdot)} \quad \int_0^1 \frac{1}{2}\mathbf{u}^2(t)dt$$

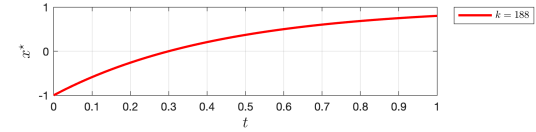
subject to

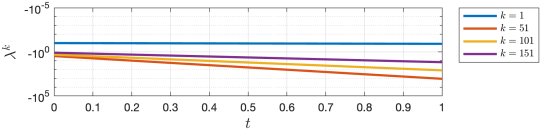
$$\dot{x}(t) = \mathbf{u}(t)(1 - x(t))$$
$$x(0) = x_0,\ x(1) = x_1$$

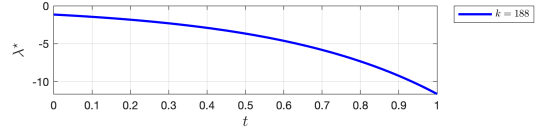
Example

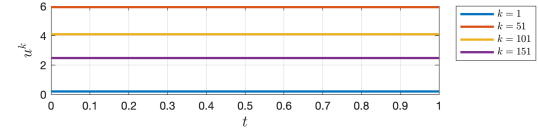
Indirect shooting as sketched above: $\nu^0 = 0, \lambda_0^0 = -0.1, x_0 = -1, x_1 = 0.75$

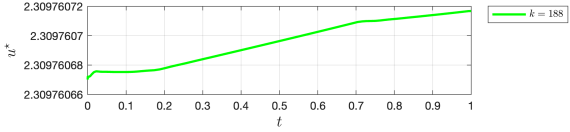






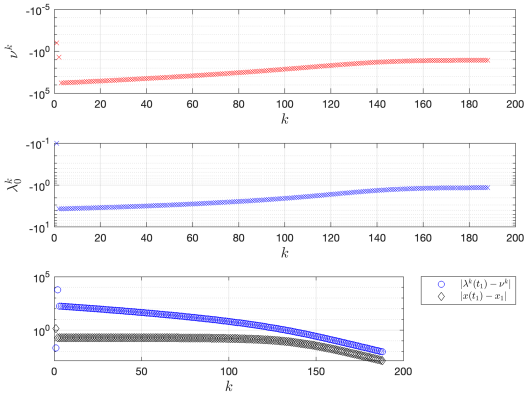


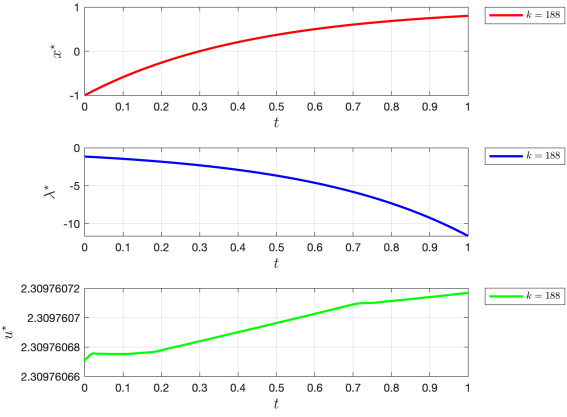




Example

Indirect shooting as sketched above: $\nu^0 = 0, \lambda_0^0 = -0.1$, $x_0 = -1$, $x_1 = 0.75$





$\nu^\star = -11.51$; $\lambda_0^\star = -1.1543$, $|\lambda^\star(t_f) - \nu^\star| = 9.4 \cdot 10^{-3}$, $|x^\star(t_f) - x_1| = 1.43 \cdot 10^{-3}$

Remarks on indirect methods

- Requires to set up the NCOs from the PMP

[13] T. Englert, A. Völz, F. Mesmer, S. Rhein, K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: Optimization and Engineering 20.3 (2019), pp. 769–809

Remarks on indirect methods

- Requires to set up the NCOs from the PMP
- Input constraints can be considered (not in the example)

[13] T. Englert, A. Völz, F. Mesmer, S. Rhein, K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: Optimization and Engineering 20.3 (2019), pp. 769–809

Remarks on indirect methods

- Requires to set up the NCOs from the PMP
- Input constraints can be considered (not in the example)
- Indirect methods can be very precise and memory efficient (no need to store large matrices)[13]

[13] T. Englert, A. Völz, F. Mesmer, S. Rhein, K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: Optimization and Engineering 20.3 (2019), pp. 769–809

Remarks on indirect methods

- Requires to set up the NCOs from the PMP
- Input constraints can be considered (not in the example)
- Indirect methods can be very precise and memory efficient (no need to store large matrices)[13]
- In case of active state constraints $\rightsquigarrow$ much more complicated NCOs $\rightsquigarrow$ indirect methods become tedious to apply

[13] T. Englert, A. Völz, F. Mesmer, S. Rhein, K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: Optimization and Engineering 20.3 (2019), pp. 769–809

Remarks on indirect methods

- Requires to set up the NCOs from the PMP
- Input constraints can be considered (not in the example)
- Indirect methods can be very precise and memory efficient (no need to store large matrices)[13]
- In case of active state constraints $\rightsquigarrow$ much more complicated NCOs $\rightsquigarrow$ indirect methods become tedious to apply
- Unstable dynamics $\dot{x} = f(x, \mathbf{u})$ lead to numerical issues $\rightsquigarrow$ good initial guesses needed in Newton's method $\rightsquigarrow$ often requires manual attention

[13] T. Englert, A. Völz, F. Mesmer, S. Rhein, K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: Optimization and Engineering 20.3 (2019), pp. 769–809

Remarks on indirect methods

- Requires to set up the NCOs from the PMP
- Input constraints can be considered (not in the example)
- Indirect methods can be very precise and memory efficient (no need to store large matrices)[13]
- In case of active state constraints $\rightsquigarrow$ much more complicated NCOs $\rightsquigarrow$ indirect methods become tedious to apply
- Unstable dynamics $\dot{x} = f(x, \mathbf{u})$ lead to numerical issues $\rightsquigarrow$ good initial guesses needed in Newton's method $\rightsquigarrow$ often requires manual attention
- Either adjoint dynamics or state dynamics are unstable

[13] T. Englert, A. Völz, F. Mesmer, S. Rhein, K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: Optimization and Engineering 20.3 (2019), pp. 769–809

Direct Solution Methods

Direct solution approaches – Main idea

"First discretize, then optimize"

Convert OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Direct solution approaches – Main idea

"First discretize, then optimize"

Convert OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

into nonlinear opt. problem (NLP)

$$\min_{\xi \in \mathbb{R}^{n_\xi}} \mathrm{f}(\xi)$$

subject to

$$\mathrm{h}(\xi) = 0$$
$$\mathrm{g}(\xi) \leq 0$$

Direct solution approaches – Main idea

"First discretize, then optimize"

Convert OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Optimization in a function space

into nonlinear opt. problem (NLP)

$$\min_{\xi \in \mathbb{R}^{n_\xi}} \mathrm{f}(\xi)$$

subject to

$$\mathrm{h}(\xi) = 0$$
$$\mathrm{g}(\xi) \leq 0$$

Direct solution approaches – Main idea

"First discretize, then optimize"

Convert OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Optimization in a function space

into nonlinear opt. problem (NLP)

$$\min_{\xi \in \mathbb{R}^{n_\xi}} \mathrm{f}(\xi)$$

subject to

$$\mathrm{h}(\xi) = 0$$
$$\mathrm{g}(\xi) \leq 0$$

Optimization in a finite-dimensional vector space

Direct solution approaches – Main idea

"First discretize, then optimize"

Convert OCP

$$\min_{\mathbf{u}(\cdot)} J(x_0, \mathbf{u}(\cdot))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subseteq \mathbb{R}^n$$

Optimization in a function space

into nonlinear opt. problem (NLP)

$$\min_{\xi \in \mathbb{R}^{n_\xi}} \mathrm{f}(\xi)$$

subject to

$$\mathrm{h}(\xi) = 0$$
$$\mathrm{g}(\xi) \leq 0$$

Optimization in a finite-dimensional vector space

- How to discretize $\mathbf{u}(\cdot)$? How to obtain a solution to $\dot{x} = f(x, u)$?

Input discretization?

Parametrize $\mathbf{u}(\cdot)$ by finitely many parameters u_k, $k = 1, \ldots, N_{opt}$

$$\mathbf{u}(\cdot) \in L_\infty([t_0, t_f], \mathbb{U}) \quad \rightsquigarrow \quad \mathbf{u}(t) = \sum_{k=1}^{N_{opt}} u_k \phi_k(t)$$

with basis functions $\phi_k(\cdot)$

Input discretization?

Parametrize $\mathbf{u}(\cdot)$ by finitely many parameters u_k, $k = 1, \ldots, N_{opt}$

$$\mathbf{u}(\cdot) \in L_\infty([t_0, t_f], \mathbb{U}) \quad \rightsquigarrow \quad \mathbf{u}(t) = \sum_{k=1}^{N_{opt}} u_k \phi_k(t)$$

with basis functions $\phi_k(\cdot)$

The parameters u_k then become the optimization variables in the finite-dimensional problem

Input discretization?

Parametrize $\mathbf{u}(\cdot)$ by finitely many parameters u_k, $k = 1, \ldots, N_{opt}$

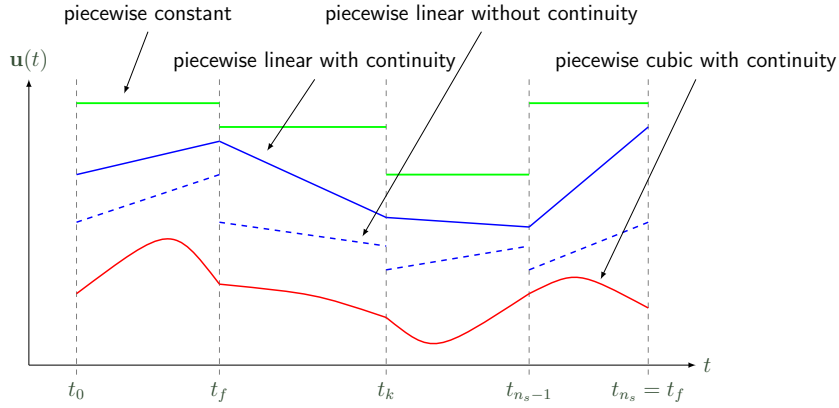
$$\mathbf{u}(\cdot) \in L_\infty([t_0, t_f], \mathbb{U}) \quad \rightsquigarrow \quad \mathbf{u}(t) = \sum_{k=1}^{N_{opt}} u_k \phi_k(t)$$

with basis functions $\phi_k(\cdot)$

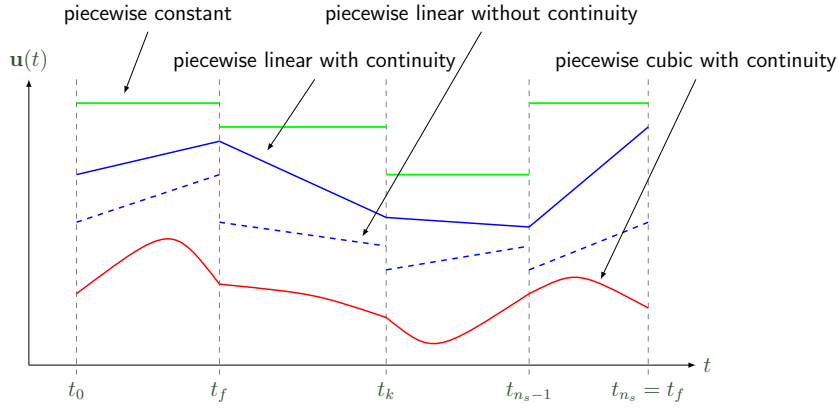
The parameters u_k then become the optimization variables in the finite-dimensional problem

Which basis functions to use?

Input discretization?



Input discretization?



Very often, one simply uses piecewise constant input parametrizations

Discretisation of dynamics?

How to solve $\dot{x} = f(t, x, \mathbf{u})$?

Discretisation of dynamics?

How to solve $\dot{x} = f(t, x, \mathbf{u})$?

Here: only simple fixed step-size integration methods

These can be written as discrete-time control systems

$$h \doteq t_{k+1} - t_k = const.$$
$$x(t_{k+1}) = g(t_k, x(t_k), u_k)$$

Discretisation of dynamics?

How to solve $\dot{x} = f(t, x, \mathbf{u})$?

Here: only simple fixed step-size integration methods

These can be written as discrete-time control systems

$$h \doteq t_{k+1} - t_k = const.$$
$$x(t_{k+1}) = g(t_k, x(t_k), u_k)$$

Simplest choice: Euler scheme $g(t, x, u) = x + hf(t, x, u)$

(but more sophisticated schemes like Heun, classical Runge-Kutta, or implicit methods may be advantageous)

Considered OCP

$$\min_{\mathbf{u}(\cdot),p} L(t_f, x(t_f))$$

subject to (OCP)

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subset \mathbb{R}^n$$

Considered OCP

$$\min_{\mathbf{u}(\cdot),p} L(t_f, x(t_f))$$

subject to (OCP)

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subset \mathbb{R}^n$$

For the illustration of the procedure we consider a problem without terminal conditions. This can be adapted to most common settings, such as

- free end time problems
- periodic boundary conditions
- free initial conditions
- ...

Direct single shooting (simultaneous)

$$\min_{\mathbf{u}(\cdot),p} L(t_f, x(t_f))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subset \mathbb{R}^n$$

Direct single shooting (simultaneous)

$$\min_{\mathbf{u}(\cdot),p} L(t_f, x(t_f))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subset \mathbb{R}^n$$

- Discretized problem: (with t_k replaced by k)

Direct single shooting (simultaneous)

$$\min_{\mathbf{u}(\cdot),p} L(t_f, x(t_f))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subset \mathbb{R}^n$$

- Discretized problem: (with t_k replaced by k)

$$\min_{u(0),\ldots,u(N-1),x(N),p} L(N, x(N)) \qquad \text{subject to}$$
$$x(0) = x_0, \quad x(k+1) = g(k, x(k), u(k))$$
$$\forall k \in \{0, \ldots, N-1\} : \quad u(k) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall k \in \{0, \ldots, N\} : \quad x(k) \in \mathbb{X}$$

Remarks on direct single shooting

- The state $x(N)$ (on which the cost $L(N, x(N))$ depends) is computed "in one shot" from x_0, hence the name

Remarks on direct single shooting

- The state $x(N)$ (on which the cost $L(N, x(N))$ depends) is computed "in one shot" from x_0, hence the name
- ODE is (approximately) satisfied for all iterates

Remarks on direct single shooting

- The state $x(N)$ (on which the cost $L(N, x(N))$ depends) is computed "in one shot" from x_0, hence the name
- ODE is (approximately) satisfied for all iterates
- Easy to code

Remarks on direct single shooting

- The state $x(N)$ (on which the cost $L(N, x(N))$ depends) is computed "in one shot" from x_0, hence the name
- ODE is (approximately) satisfied for all iterates
- Easy to code
- Constraints are enforced at discretization points only

Remarks on direct single shooting

- The state $x(N)$ (on which the cost $L(N, x(N))$ depends) is computed "in one shot" from x_0, hence the name
- ODE is (approximately) satisfied for all iterates
- Easy to code
- Constraints are enforced at discretization points only
- The NLP has $n_u \cdot N + n_p$ decision variables

Remarks on direct single shooting

- The state $x(N)$ (on which the cost $L(N, x(N))$ depends) is computed "in one shot" from x_0, hence the name
- ODE is (approximately) satisfied for all iterates
- Easy to code
- Constraints are enforced at discretization points only
- The NLP has $n_u \cdot N + n_p$ decision variables
- Unstable systems lead to highly sensitive dependence of the solution from the control input $\rightsquigarrow$ NLP difficult to solve

Direct multiple shooting

$$\min_{\mathbf{u}(\cdot),p} L(t_f, x(t_f))$$

subject to

$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subset \mathbb{R}^n$$

Direct multiple shooting

$$\min_{\mathbf{u}(\cdot),p} L(t_f, x(t_f))$$

subject to

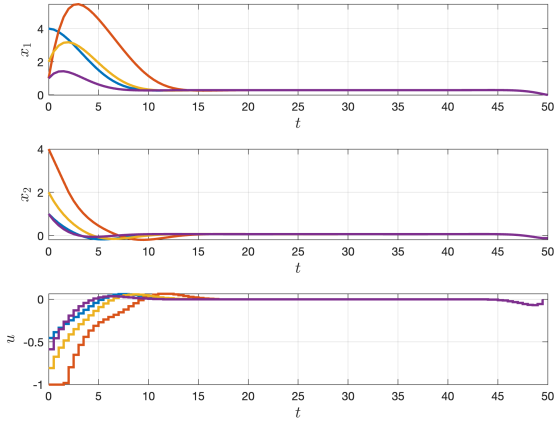
$$\dot{x}(t) = f(t, x(t), \mathbf{u}(t)), \quad x(t_0) = x_0$$
$$\forall t \in [t_0, t_f] : \mathbf{u}(t) \in \mathbb{U} \subset \mathbb{R}^m$$
$$\forall t \in [t_0, t_f] : x(t) \in \mathbb{X} \subset \mathbb{R}^n$$

- Introduce $N+1$ (vector valued) additional decision variables $\xi(0), \ldots, \xi(N-1)$
- Solve ODE separately on N small intervals, enforce continuity through additional constraints upon convergence

$$\min_{u(0),\ldots,u(N-1),\xi(0),\ldots,\xi(N-1),p} L(N, x(N)) \quad \text{subject to}$$

$$\forall k \in \{0, \ldots, N-1\} : x(k+1) = g(k, \xi(k), u(k)),$$
$$\forall k \in \{1, \ldots, N-1\} : \xi(k) = x(k) \text{ and } \xi_0 = x_0$$
$$\forall k \in \{0, \ldots, N-1\} : u(k) \in \mathbb{U} \subset \mathbb{R}^m, \quad \forall k \in \{0, \ldots, N\} : x(k) \in \mathbb{X} \subset \mathbb{R}^n$$

Example – Direct multiple shooting



Revisit the optimal investment problem: $x_1 =$ invested capital, $x_2 =$ investment, $\mathbf{u}$ $=$ change of investment

Solved with CasADi[13] and ipopt

[14] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, Moritz Diehl. "CasADi: a software framework for nonlinear optimization and optimal control". In: Mathematical Programming Computation 11 (2019), pp. 1–36

Remarks on direct multiple shooting

- The state $x(N)$ is now computed in N (=multiple) shots from x_0, which are coupled via constraints[15]

[15] H.G. Bock and K.-J. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: Proc. 9th IFAC World Congress. 1984

Remarks on direct multiple shooting

- The state $x(N)$ is now computed in N (=multiple) shots from x_0, which are coupled via constraints[15]
- ODE is satisfied upon convergence of NLP solver

[15] H.G. Bock and K.-J. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: Proc. 9th IFAC World Congress. 1984

Remarks on direct multiple shooting

- The state $x(N)$ is now computed in N (=multiple) shots from x_0, which are coupled via constraints[15]
- ODE is satisfied upon convergence of NLP solver
- Constraints are enforced at discretization points only

[15] H.G. Bock and K.-J. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: Proc. 9th IFAC World Congress. 1984

Remarks on direct multiple shooting

- The state $x(N)$ is now computed in N (=multiple) shots from x_0, which are coupled via constraints[15]
- ODE is satisfied upon convergence of NLP solver
- Constraints are enforced at discretization points only
- $n_u \cdot N + n_x \cdot (N+1) + n_x \cdot N + n_p$ decision variables

[15] H.G. Bock and K.-J. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: Proc. 9th IFAC World Congress. 1984

Remarks on direct multiple shooting

- The state $x(N)$ is now computed in N (=multiple) shots from x_0, which are coupled via constraints[15]
- ODE is satisfied upon convergence of NLP solver
- Constraints are enforced at discretization points only
- $n_u \cdot N + n_x \cdot (N+1) + n_x \cdot N + n_p$ decision variables
- Handles unstable systems much better than single shooting

[15] H.G. Bock and K.-J. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: Proc. 9th IFAC World Congress. 1984

Remarks on direct multiple shooting

- The state $x(N)$ is now computed in N (=multiple) shots from x_0, which are coupled via constraints[15]
- ODE is satisfied upon convergence of NLP solver
- Constraints are enforced at discretization points only
- $n_u \cdot N + n_x \cdot (N+1) + n_x \cdot N + n_p$ decision variables
- Handles unstable systems much better than single shooting
- Workhorse method for many MPC implementations, see Part 4 of the course

[15] H.G. Bock and K.-J. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: Proc. 9th IFAC World Congress. 1984

Other direct methods?

- Collocation: parametrization of control and state trajectories via piecewise polynomials

[16] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

Other direct methods?

- Collocation: parametrization of control and state trajectories via piecewise polynomials
- Pseudo-spectral methods: high order orthogonal polynomial approximation of state and input trajectories

[16] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

Other direct methods?

- Collocation: parametrization of control and state trajectories via piecewise polynomials
- Pseudo-spectral methods: high order orthogonal polynomial approximation of state and input trajectories
- Combinations of the above, e.g., one can use multiple shooting with collocation for the integration, ...

[16] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

Other direct methods?

- Collocation: parametrization of control and state trajectories via piecewise polynomials
- Pseudo-spectral methods: high order orthogonal polynomial approximation of state and input trajectories
- Combinations of the above, e.g., one can use multiple shooting with collocation for the integration, ...

  $\rightsquigarrow$ all of these improve constraint satisfaction between discretisation points

[16] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

Other direct methods?

- Collocation: parametrization of control and state trajectories via piecewise polynomials
- Pseudo-spectral methods: high order orthogonal polynomial approximation of state and input trajectories
- Combinations of the above, e.g., one can use multiple shooting with collocation for the integration, ...

  $\rightsquigarrow$ all of these improve constraint satisfaction between discretisation points
- Decision tree for optimization software: `https://plato.asu.edu/guide.html`

[16] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

Other direct methods?

- Collocation: parametrization of control and state trajectories via piecewise polynomials
- Pseudo-spectral methods: high order orthogonal polynomial approximation of state and input trajectories
- Combinations of the above, e.g., one can use multiple shooting with collocation for the integration, ...

  $\rightsquigarrow$ all of these improve constraint satisfaction between discretisation points
- Decision tree for optimization software: `https://plato.asu.edu/guide.html`
- For further introductory reading, see Chapter 5 of [Chachuat '09][16]

[16] B. Chachuat. Nonlinear and Dynamic Optimization—From Theory to Practice. EPFL, 2009. URL: https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

Summary of Part 3: Numerical Solution Methods

- Dynamic programming yields an optimal feedback law, indirect and direct methods yield open-loop optimal trajectories and controls

Summary of Part 3: Numerical Solution Methods

- Dynamic programming yields an optimal feedback law, indirect and direct methods yield open-loop optimal trajectories and controls
- Numerical effort of dynamic programming grows much faster with the state dimension than that of indirect and direct methods

Summary of Part 3: Numerical Solution Methods

- Dynamic programming yields an optimal feedback law, indirect and direct methods yield open-loop optimal trajectories and controls
- Numerical effort of dynamic programming grows much faster with the state dimension than that of indirect and direct methods
- Deep reinforcement learning may provide a remedy, but for which problems?

Summary of Part 3: Numerical Solution Methods

- Dynamic programming yields an optimal feedback law, indirect and direct methods yield open-loop optimal trajectories and controls
- Numerical effort of dynamic programming grows much faster with the state dimension than that of indirect and direct methods
- Deep reinforcement learning may provide a remedy, but for which problems?
- Indirect methods are more accurate, but also considerably slower and more complicated to handle than direct methods

Summary of Part 3: Numerical Solution Methods

- Dynamic programming yields an optimal feedback law, indirect and direct methods yield open-loop optimal trajectories and controls
- Numerical effort of dynamic programming grows much faster with the state dimension than that of indirect and direct methods
- Deep reinforcement learning may provide a remedy, but for which problems?
- Indirect methods are more accurate, but also considerably slower and more complicated to handle than direct methods
- Multiple shooting much better for unstable problems than single shooting

Summary of Part 3: Numerical Solution Methods

- Dynamic programming yields an optimal feedback law, indirect and direct methods yield open-loop optimal trajectories and controls
- Numerical effort of dynamic programming grows much faster with the state dimension than that of indirect and direct methods
- Deep reinforcement learning may provide a remedy, but for which problems?
- Indirect methods are more accurate, but also considerably slower and more complicated to handle than direct methods
- Multiple shooting much better for unstable problems than single shooting

Conclusion so far: None of the classical techniques are suitable for high-dimensional optimal feedback control problems

Summary of Part 3: Numerical Solution Methods

- Dynamic programming yields an optimal feedback law, indirect and direct methods yield open-loop optimal trajectories and controls
- Numerical effort of dynamic programming grows much faster with the state dimension than that of indirect and direct methods
- Deep reinforcement learning may provide a remedy, but for which problems?
- Indirect methods are more accurate, but also considerably slower and more complicated to handle than direct methods
- Multiple shooting much better for unstable problems than single shooting

Conclusion so far: None of the classical techniques are suitable for high-dimensional optimal feedback control problems

The last two parts of this course present techniques that overcome this limitation for optimal control problems with particular properties

Outline of the course

Part 1: Optimal Control Problems — An Introduction

Part 2: Solution Concepts

Part 3: Numerical Solution Methods

Part 4: Model Predictive Control

Part 5: Deep Neural Networks for High-Dimensional Problems

Purpose of this part

Introduce MPC as an approximate solution method for infinite-horizon optimal feedback control, which does not rely on dynamic programming

Purpose of this part

Introduce MPC as an approximate solution method for infinite-horizon optimal feedback control, which does not rely on dynamic programming

Explain the turnpike property as the crucial structural property that enables MPC to yield near-optimal solutions

Purpose of this part

Introduce MPC as an approximate solution method for infinite-horizon optimal feedback control, which does not rely on dynamic programming

Explain the turnpike property as the crucial structural property that enables MPC to yield near-optimal solutions

Present a challenging industrial use case

Contents of this part

Part 4: Model Predictive Control

- Model Predictive Control
- The Turnpike Property
- Main Performance Result
- Use Case: Optimal Startup of a Combined Cycle Power Plant

Model predictive control

Model predictive control

Turnpike properties are pivotal for analysing Model Predictive Control (MPC), one of the most successful advanced control methods

Model predictive control

Turnpike properties are pivotal for analysing Model Predictive Control (MPC), one of the most successful advanced control methods

MPC approximates an optimal control problem on an infinite horizon

$$\underset{\mathbf{u}}{\text{minimise}} \quad J_\infty(x_0, \mathbf{u}) = \sum_{t=0}^{\infty} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$$

by the iterative solution of finite horizon problems

$$\underset{\mathbf{u}}{\text{minimise}} \quad J_N(x_0, \mathbf{u}) = \sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$$

with fixed $N \in \mathbb{N}$

Model predictive control

Turnpike properties are pivotal for analysing Model Predictive Control (MPC), one of the most successful advanced control methods

MPC approximates an optimal control problem on an infinite horizon

$$\underset{\mathbf{u}}{\text{minimise}} \quad J_\infty(x_0, \mathbf{u}) = \sum_{t=0}^{\infty} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$$

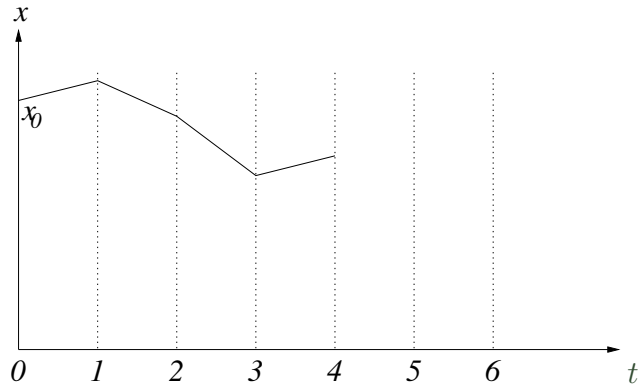
by the iterative solution of finite horizon problems

$$\underset{\mathbf{u}}{\text{minimise}} \quad J_N(x_0, \mathbf{u}) = \sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$$

with fixed $N \in \mathbb{N}$. How do we get a feedback law F?

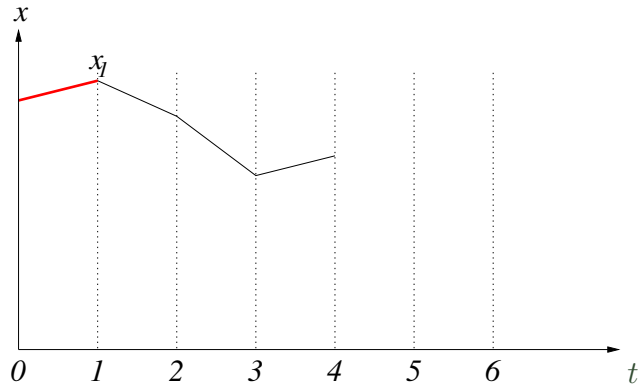
MPC from the trajectory point of view

MPC from the trajectory point of view



black = predictions (open loop optimisation)

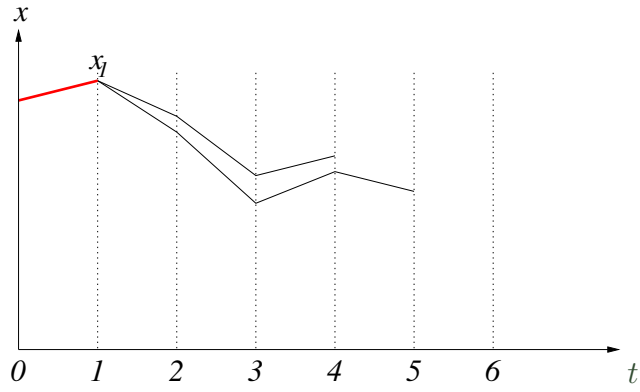
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

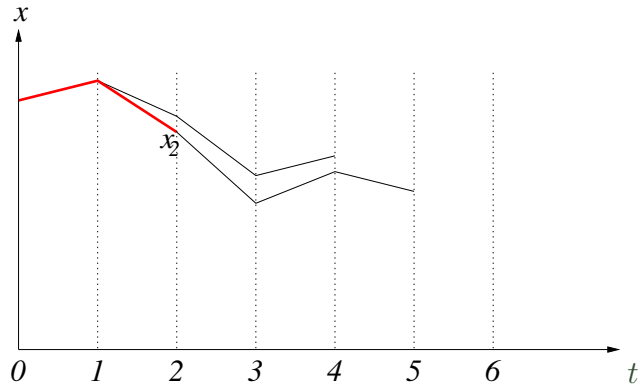
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

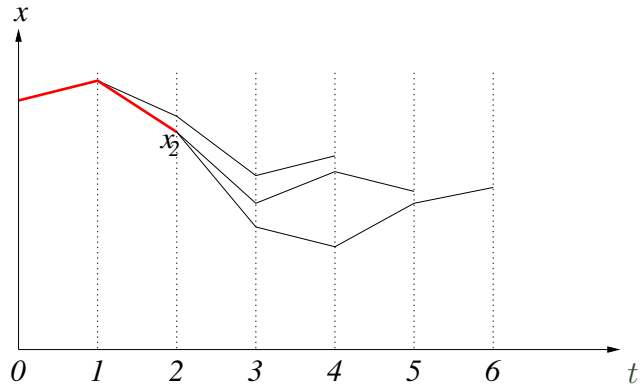
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

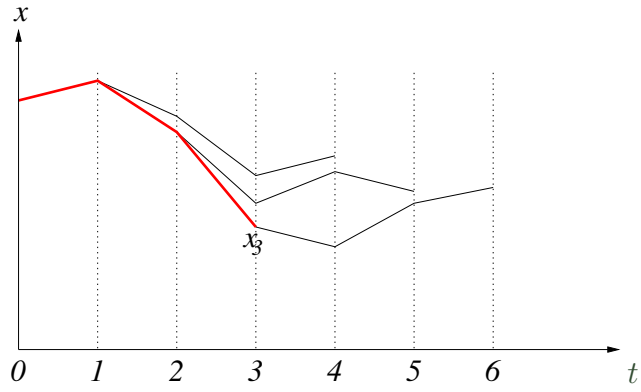
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

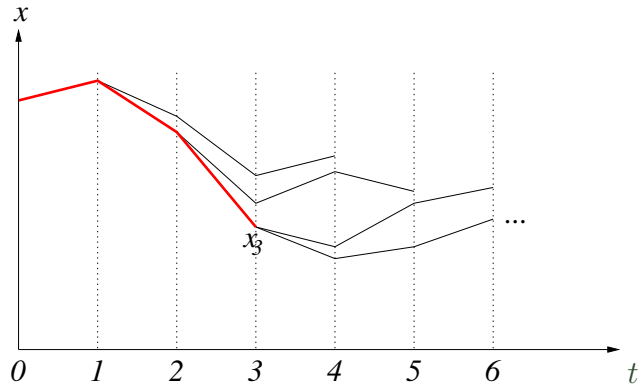
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

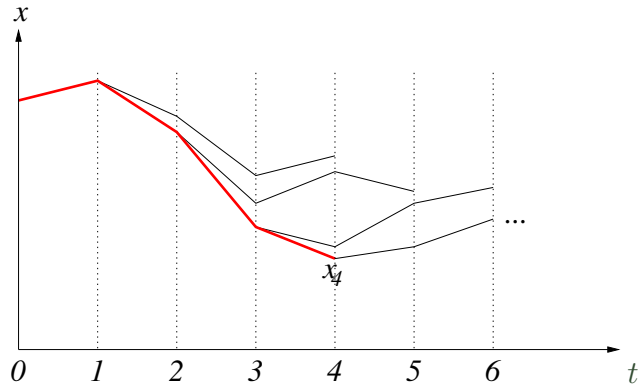
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

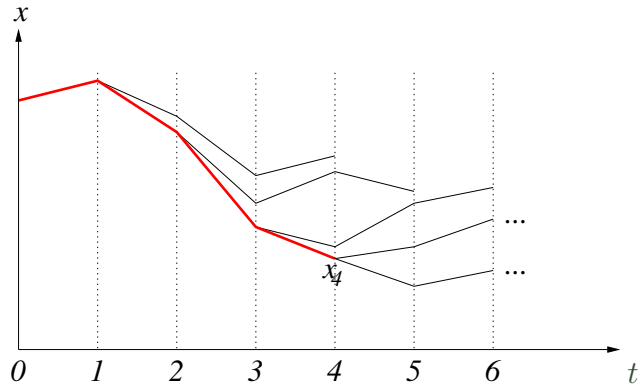
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

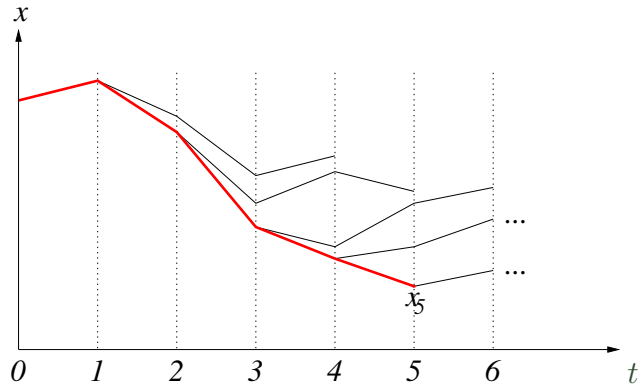
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

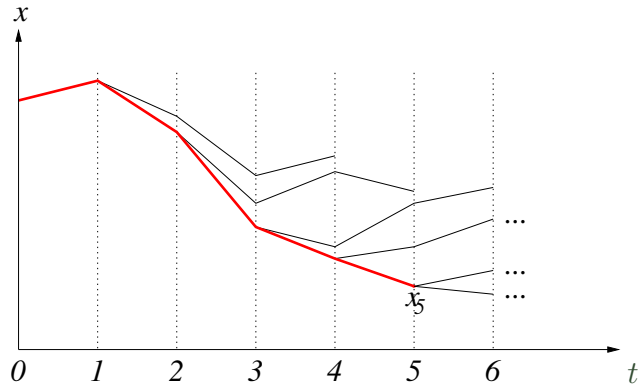
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

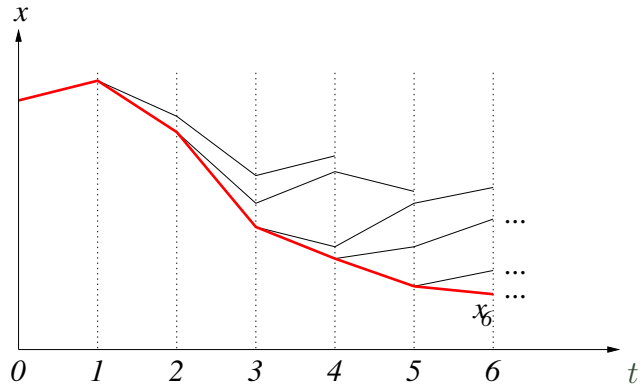
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

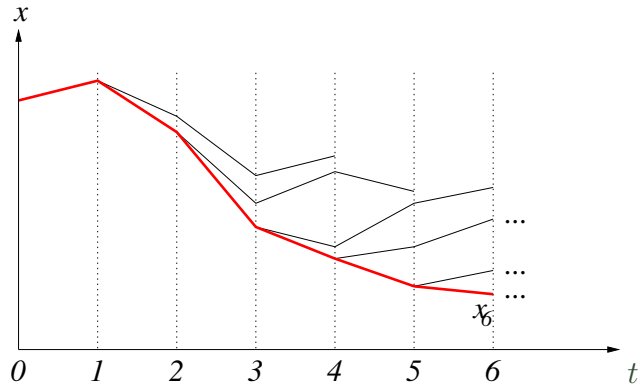
MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

MPC from the trajectory point of view



black = predictions (open loop optimisation)

red = MPC closed loop $x_{MPC}(t)$

The feedback control value $F(x_{MPC}(t))$ is evaluated online as the first element of the finite horizon optimal control sequence

The turnpike property

The turnpike property

—

a temporal redundancy property

The turnpike property

The turnpike property describes a behaviour of (approximately) optimal trajectories for a finite horizon optimal control problem

$$\underset{\mathbf{u}}{\text{minimise}} \quad J_N(x, \mathbf{u}) = \sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$$

with a cost function $\ell : X \times U \to \mathbb{R}$
and state and input constraints $x_{\mathbf{u}}(t) \in X$, $\mathbf{u}(t) \in U$

The turnpike property

The turnpike property describes a behaviour of (approximately) optimal trajectories for a finite horizon optimal control problem

$$\underset{\mathbf{u}}{\text{minimise}} \quad J_N(x, \mathbf{u}) = \sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$$

with a cost function $\ell : X \times U \to \mathbb{R}$
and state and input constraints $x_{\mathbf{u}}(t) \in X$, $\mathbf{u}(t) \in U$

Informal description of the turnpike property:

Any optimal trajectory stays near an equilibrium x^e most of the time

The turnpike property

The turnpike property describes a behaviour of (approximately) optimal trajectories for a finite horizon optimal control problem

$$\underset{\mathbf{u}}{\text{minimise}} \quad J_N(x, \mathbf{u}) = \sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$$

with a cost function $\ell : X \times U \to \mathbb{R}$
and state and input constraints $x_{\mathbf{u}}(t) \in X$, $\mathbf{u}(t) \in U$

Informal description of the turnpike property:

Any optimal trajectory stays near an equilibrium x^e most of the time

We illustrate the property by two simple examples

Example: minimum energy control

Example: Keep the state of the system inside a given interval X minimising the quadratic control effort

$$\ell(x,u) = u^2$$

with dynamics

$$x^+ = 2x + u$$

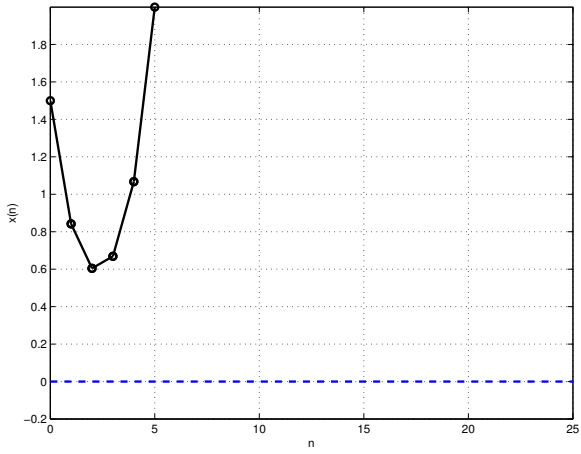
and constraints $\mathbb{X} = [-2,2]$, $\mathbb{U} = [-3,3]$

Quiz

Which is the "cheapest" state for keeping the system inside $\mathbb{X} = [-2, 2]$?

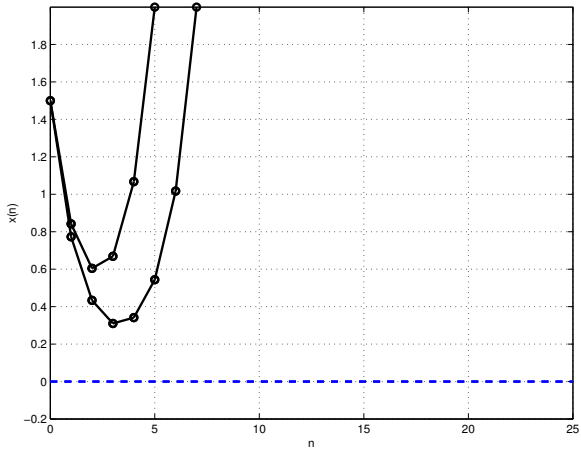
- $x = 2$
- $x = 0$
- Any $|x| < \varepsilon$ for $\varepsilon > 0$ sufficiently small

Example: optimal trajectories



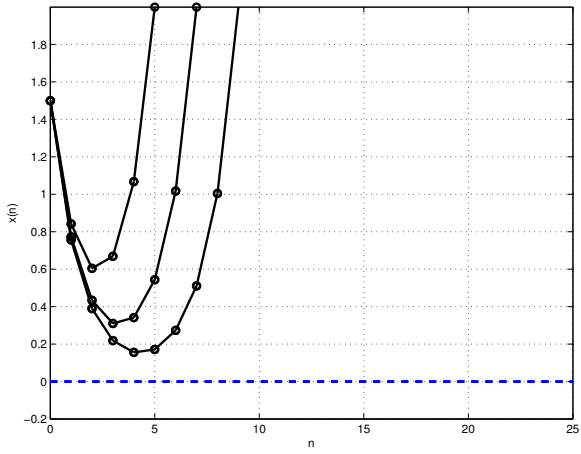
Optimal trajectory for $N = 5$

Example: optimal trajectories



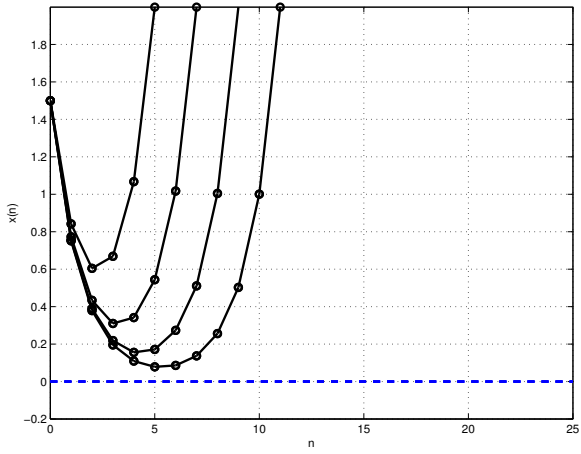
Optimal trajectories for $N = 5, \ldots, 7$

Example: optimal trajectories



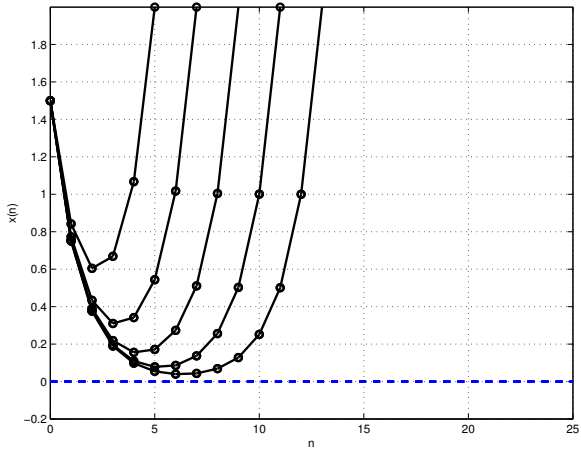
Optimal trajectories for $N = 5, \ldots, 9$

Example: optimal trajectories



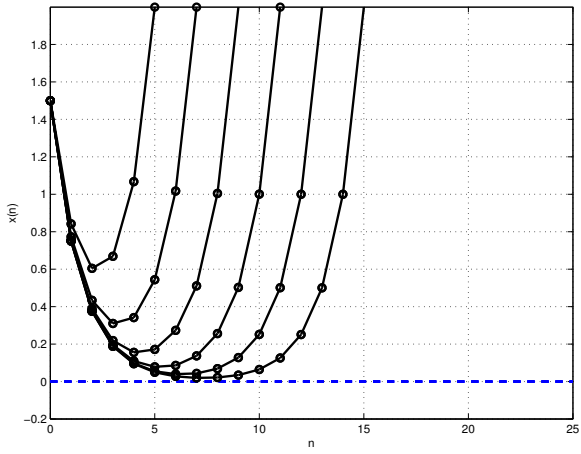
Optimal trajectories for $N = 5, \ldots, 11$

Example: optimal trajectories



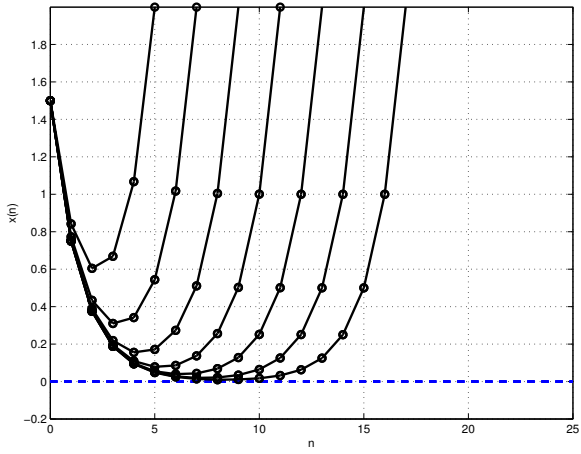
Optimal trajectories for $N = 5, \ldots, 13$

Example: optimal trajectories



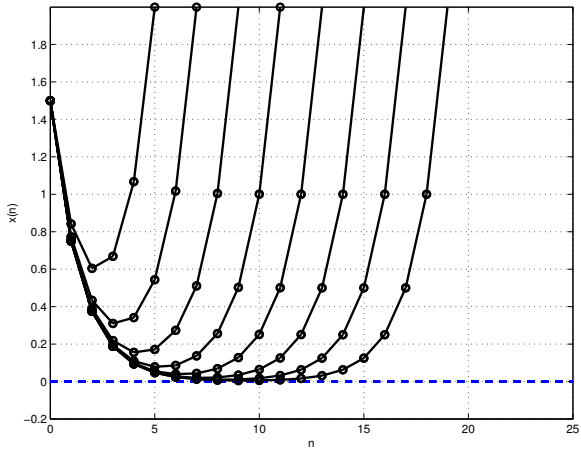
Optimal trajectories for $N = 5, \ldots, 15$

Example: optimal trajectories



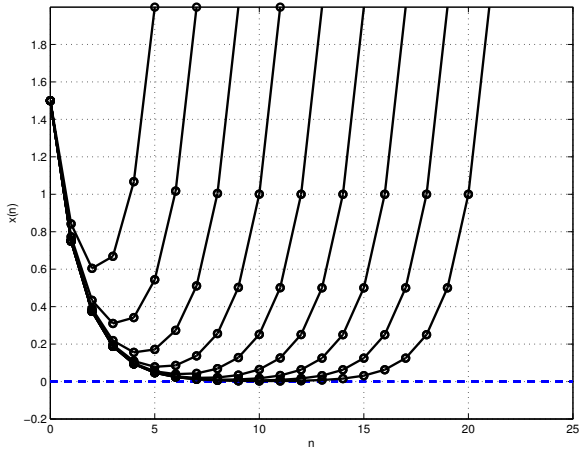
Optimal trajectories for $N = 5, \ldots, 17$

Example: optimal trajectories



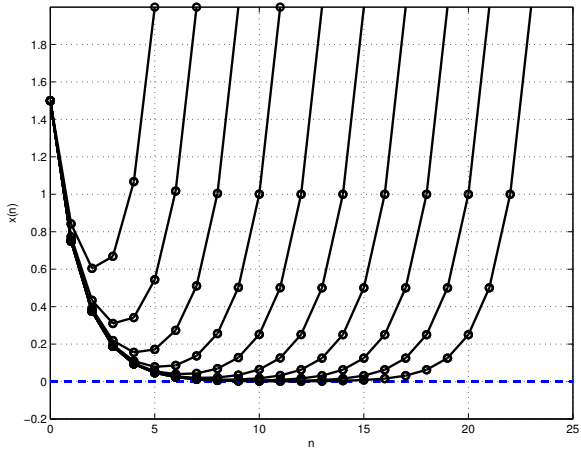
Optimal trajectories for $N = 5, \ldots, 19$

Example: optimal trajectories



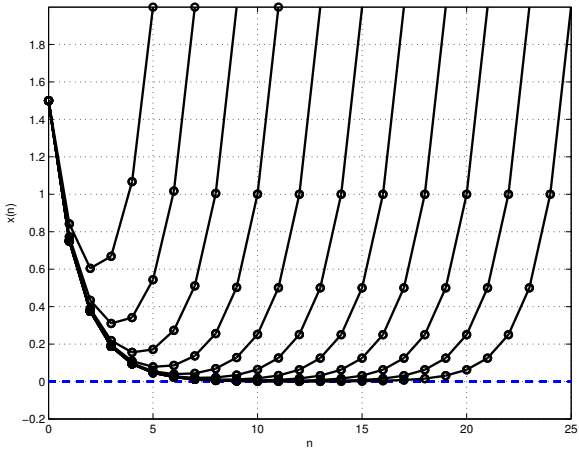
Optimal trajectories for $N = 5, \ldots, 21$

Example: optimal trajectories



Optimal trajectories for $N = 5, \ldots, 23$

Example: optimal trajectories



Optimal trajectories for $N = 5, \ldots, 25$

Example: a macroeconomic model

Consider a classical 1d macroeconomic model

[Brock/Mirman '72]

Minimise the finite horizon objective $\sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$ with

$$\ell(x,u) = -\ln(Ax^\alpha - u), \quad A = 5,\ \alpha = 0.34$$

and dynamics $x(k+1) = u(k)$ on $\mathbb{X} = \mathbb{U} = [0, 10]$

Example: a macroeconomic model

Consider a classical 1d macroeconomic model

[Brock/Mirman '72]

Minimise the finite horizon objective $\sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$ with

$$\ell(x,u) = -\ln(Ax^\alpha - u), \quad A = 5,\ \alpha = 0.34$$

and dynamics $x(k+1) = u(k)$ on $\mathbb{X} = \mathbb{U} = [0, 10]$

x = invested capital; u = investment in next time step

Example: a macroeconomic model

Consider a classical 1d macroeconomic model

[Brock/Mirman '72]

Minimise the finite horizon objective $\sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$ with

$$\ell(x,u) = -\ln(Ax^\alpha - u), \quad A = 5,\ \alpha = 0.34$$

and dynamics $x(k+1) = u(k)$ on $\mathbb{X} = \mathbb{U} = [0, 10]$

x = invested capital; u = investment in next time step

Ax^α = capital after one time step

Example: a macroeconomic model

Consider a classical 1d macroeconomic model

[Brock/Mirman '72]

Minimise the finite horizon objective $\sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$ with

$$\ell(x,u) = -\ln(Ax^\alpha - u), \quad A = 5,\ \alpha = 0.34$$

and dynamics $x(k+1) = u(k)$ on $\mathbb{X} = \mathbb{U} = [0, 10]$

x = invested capital; u = investment in next time step

Ax^α = capital after one time step

$Ax^\alpha - u$ = consumed capital; $\ln(\cdot)$ = utility function

Example: a macroeconomic model

Consider a classical 1d macroeconomic model

[Brock/Mirman '72]

Minimise the finite horizon objective $\sum_{t=0}^{N-1} \ell(x_{\mathbf{u}}(t), \mathbf{u}(t))$ with

$$\ell(x,u) = -\ln(Ax^\alpha - u), \quad A = 5,\ \alpha = 0.34$$

and dynamics $x(k+1) = u(k)$ on $\mathbb{X} = \mathbb{U} = [0, 10]$

x = invested capital; u = investment in next time step

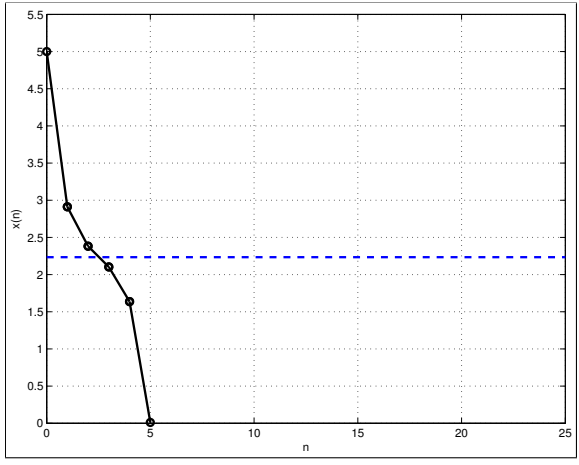
Ax^α = capital after one time step

$Ax^\alpha - u$ = consumed capital; $\ln(\cdot)$ = utility function

On infinite horizon, it is optimal to stay at the equilibrium

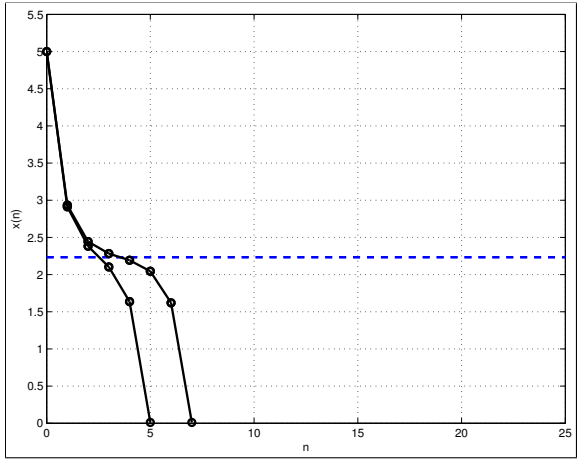
$$x^e \approx 2.2344 \quad \text{with } \ell(x^e, u^e) \approx 1.4673$$

Example: optimal trajectories



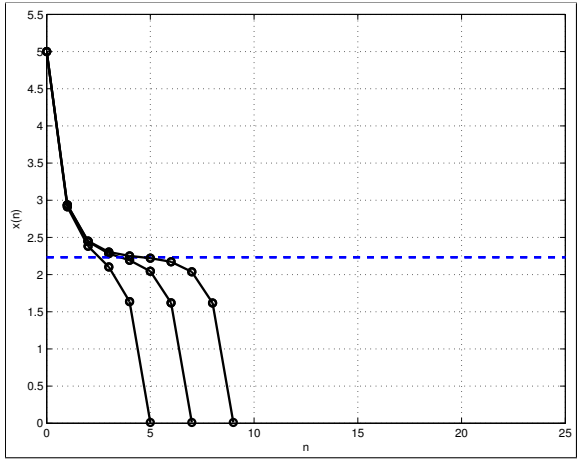
Optimal trajectory for $N = 5$

Example: optimal trajectories



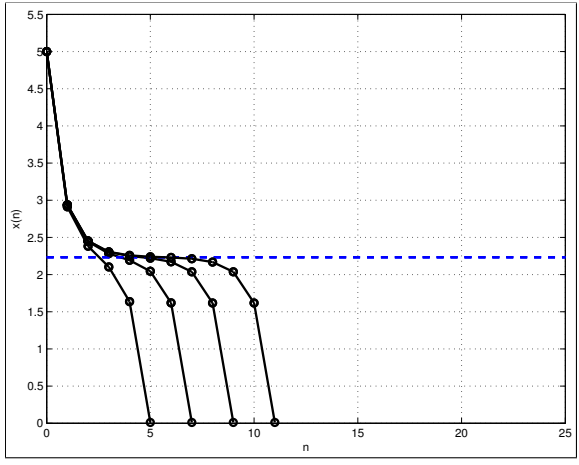
Optimal trajectories for $N = 5, \ldots, 7$

Example: optimal trajectories



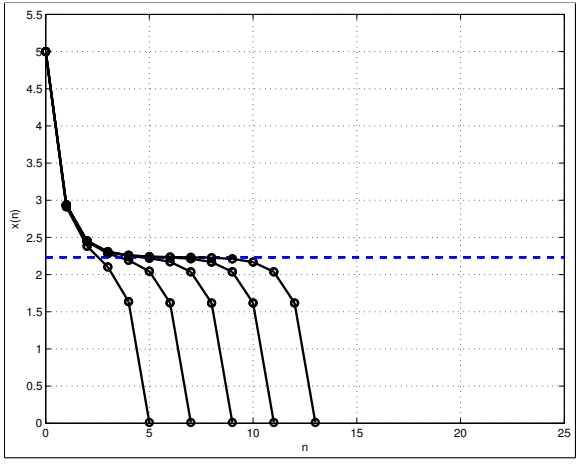
Optimal trajectories for $N = 5, \ldots, 9$

Example: optimal trajectories



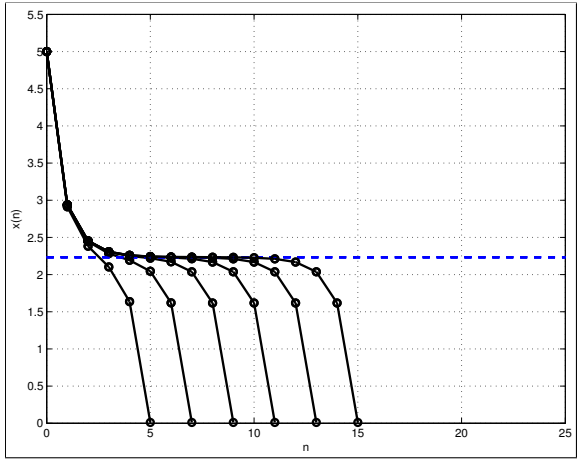
Optimal trajectories for $N = 5, \ldots, 11$

Example: optimal trajectories



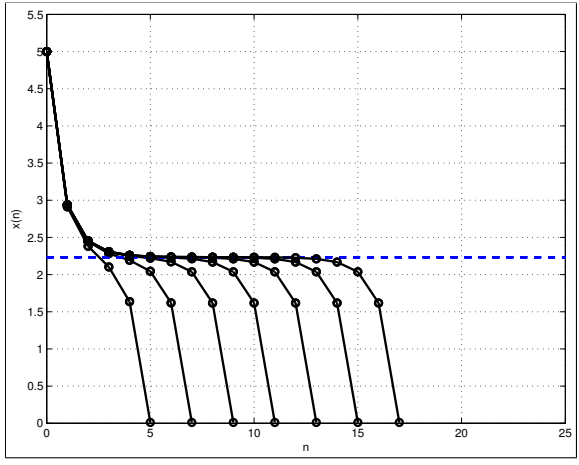
Optimal trajectories for $N = 5, \ldots, 13$

Example: optimal trajectories



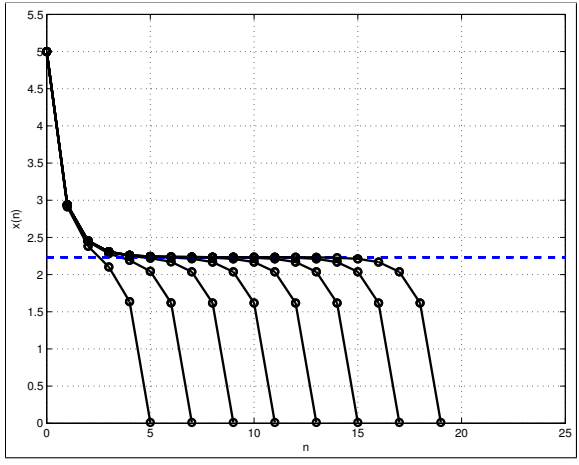
Optimal trajectories for $N = 5, \ldots, 15$

Example: optimal trajectories



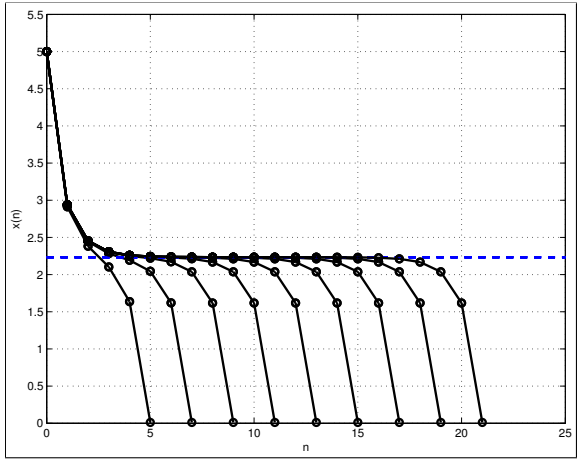
Optimal trajectories for $N = 5, \ldots, 17$

Example: optimal trajectories



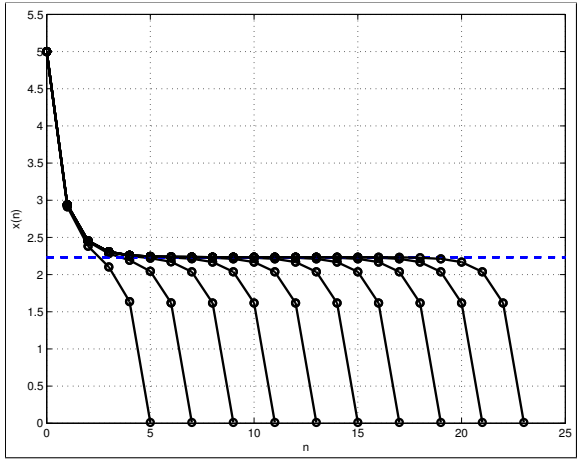
Optimal trajectories for $N = 5, \ldots, 19$

Example: optimal trajectories



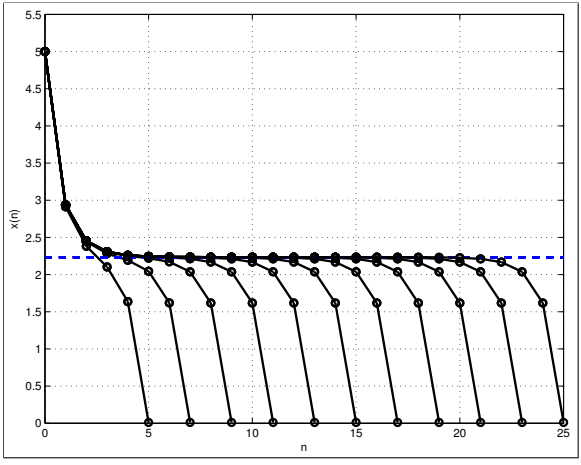
Optimal trajectories for $N = 5, \ldots, 21$

Example: optimal trajectories



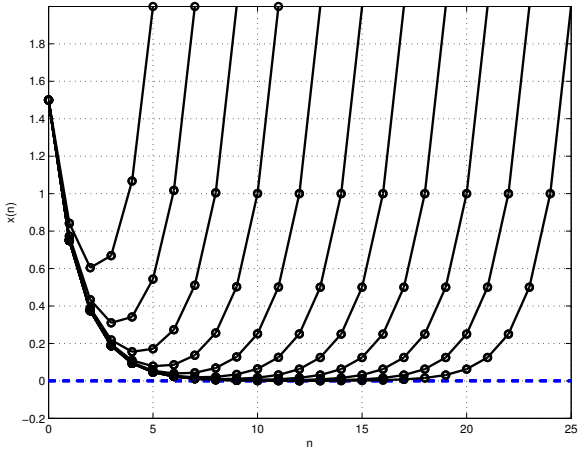
Optimal trajectories for $N = 5, \ldots, 23$

Example: optimal trajectories

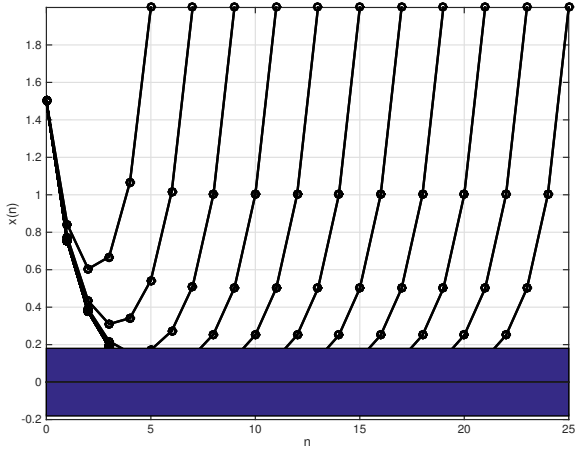


Optimal trajectories for $N = 5, \ldots, 25$

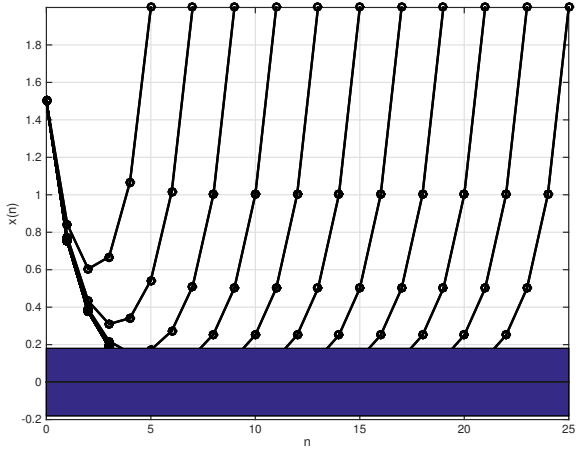
How to formalise the turnpike property?



How to formalise the turnpike property?

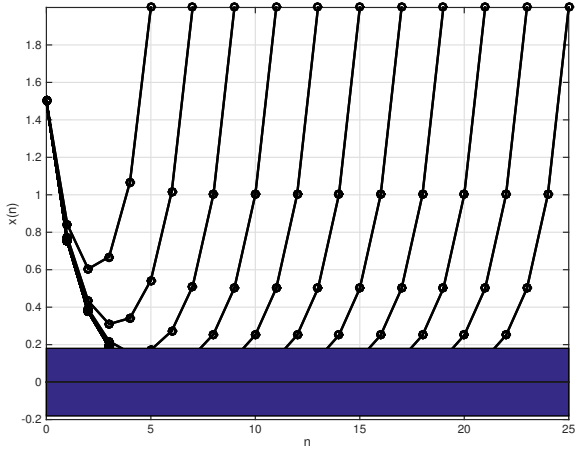


How to formalise the turnpike property?



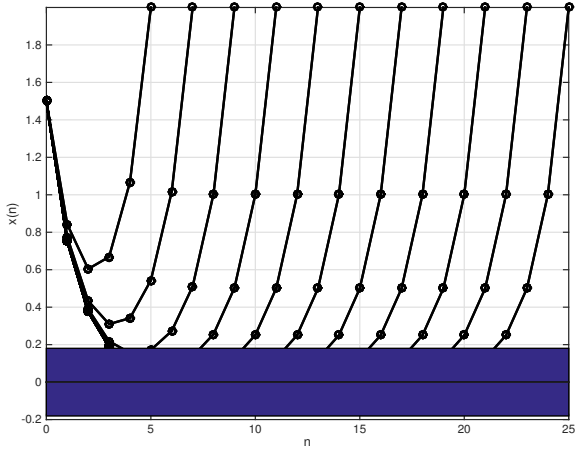
The number of points outside the blue neighbourhood is bounded by a number independent of N

How to formalise the turnpike property?



The number of points outside the blue neighbourhood is bounded by a number independent of N (here: by 8)

How to formalise the turnpike property?



The number of points outside the blue neighbourhood is bounded by a number independent of N (here: by 8)

In continuous time: The Lebesgue measure replaces the counting of points

History

- First described by [Ramsey 1928, von Neumann 1938]

History

- First described by [Ramsey 1928, von Neumann 1938]
- Name "turnpike property" coined by [Dorfman/Samuelson/Solow 1957]

History

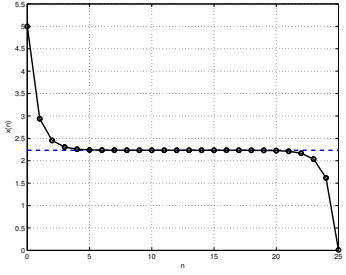
- First described by [Ramsey 1928, von Neumann 1938]
- Name "turnpike property" coined by [Dorfman/Samuelson/Solow 1957]

History

- First described by [Ramsey 1928, von Neumann 1938]
- Name "turnpike property" coined by [Dorfman/Samuelson/Solow 1957]

History

- First described by [Ramsey 1928, von Neumann 1938]
- Name "turnpike property" coined by [Dorfman/Samuelson/Solow 1957]



History

- First described by [Ramsey 1928, von Neumann 1938]
- Name "turnpike property" coined by [Dorfman/Samuelson/Solow 1957]
- Extensively studied in the 1970s in mathematical economy, cf. survey [McKenzie 1983]

History

- First described by [Ramsey 1928, von Neumann 1938]
- Name "turnpike property" coined by [Dorfman/Samuelson/Solow 1957]
- Extensively studied in the 1970s in mathematical economy, cf. survey [McKenzie 1983]
- Renewed interest since about ten years [Zaslavski '14ff, Faulwasser et al. '15ff, Trélat/Porretta/Zuazua et al. '15ff, Gugat et al. '16ff, Schaller et al. '19f, Breiten/Pfeiffer '20, ...]

History

- First described by [Ramsey 1928, von Neumann 1938]
- Name "turnpike property" coined by [Dorfman/Samuelson/Solow 1957]
- Extensively studied in the 1970s in mathematical economy, cf. survey [McKenzie 1983]
- Renewed interest since about ten years [Zaslavski '14ff, Faulwasser et al. '15ff, Trélat/Porretta/Zuazua et al. '15ff, Gugat et al. '16ff, Schaller et al. '19f, Breiten/Pfeiffer '20, ...]
- Selected applications:
  - synthesis of optimal trajectories [Anderson/Kokotovic '87]
  - learning in neural ODEs [Esteve-Yagüe/Geshkovski/Pighin/Zuazua '21ff, Püttschneider/Faulwasser '24f]
  - optimization based estimation [Schiller/Gr./Müller '24f]
  - model predictive control [will be explained in a few minutes]

Tracking type problems

For a stabilizable equilibrium x^s with control u^s, i.e., $g(x^s, u^s) = x^s$, the tracking cost

$$\ell(x^s, u^s) := \|x - x^s\| + \mu\|u - u^s\|$$

for $\mu \geq 0$ defines an optimal control problem with turnpike property at $x^e = x^s$

Tracking type problems

For a stabilizable equilibrium x^s with control u^s, i.e., $g(x^s, u^s) = x^s$, the tracking cost

$$\ell(x^s, u^s) := \|x - x^s\| + \mu\|u - u^s\|$$

for $\mu \geq 0$ defines an optimal control problem with turnpike property at $x^e = x^s$

However, the class of problems exhibiting the turnpike property is much larger

Tracking type problems

For a stabilizable equilibrium x^s with control u^s, i.e., $g(x^s, u^s) = x^s$, the tracking cost

$$\ell(x^s, u^s) := \|x - x^s\| + \mu\|u - u^s\|$$

for $\mu \geq 0$ defines an optimal control problem with turnpike property at $x^e = x^s$

However, the class of problems exhibiting the turnpike property is much larger

Strict Dissipativity is a key to understanding how large it is

Strict dissipativity

The optimal control problem is called strictly dissipative on X, if there exists a storage function $\lambda : X \to \mathbb{R}$, bounded from below, and $\alpha \in \mathcal{K}$ such that for all $x \in X$, $u \in U$ with $g(x,u) \in X$:

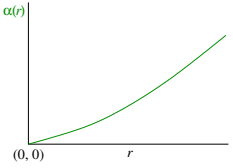
$$\lambda(g(x,u)) \leq \lambda(x) + \ell(x,u) - \ell(x^e,u^e) - \alpha(\|x - x^e\|)$$

Strict dissipativity

The optimal control problem is called strictly dissipative on X, if there exists a storage function $\lambda : X \to \mathbb{R}$, bounded from below, and $\alpha \in \mathcal{K}$ such that for all $x \in X$, $u \in U$ with $g(x,u) \in X$:

$$\lambda(g(x,u)) \leq \lambda(x) + \ell(x,u) - \ell(x^e,u^e) - \alpha(\|x - x^e\|)$$

$\alpha \in \mathcal{K}$: $\alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, continuous, strictly increasing, $\alpha(0) = 0$



Strict dissipativity

The optimal control problem is called strictly dissipative on X, if there exists a storage function $\lambda : X \to \mathbb{R}$, bounded from below, and $\alpha \in \mathcal{K}$ such that for all $x \in X$, $u \in U$ with $g(x,u) \in X$:

$$\lambda(g(x,u)) \leq \lambda(x) + \ell(x,u) - \ell(x^e,u^e) - \alpha(\|x - x^e\|)$$

Facts (for stabilizable systems):

- Strict dissipativity implies the turnpike property [Gr. '13]

Strict dissipativity

The optimal control problem is called strictly dissipative on X, if there exists a storage function $\lambda : X \to \mathbb{R}$, bounded from below, and $\alpha \in \mathcal{K}$ such that for all $x \in X$, $u \in U$ with $g(x,u) \in X$:

$$\lambda(g(x,u)) \leq \lambda(x) + \ell(x,u) - \ell(x^e,u^e) - \alpha(\|x - x^e\|)$$

Facts (for stabilizable systems):

- Strict dissipativity implies the turnpike property [Gr. '13]
- Under a controllability condition, it is equivalent to a robust version of the turnpike property [Gr./Müller '16]

Strict dissipativity

The optimal control problem is called strictly dissipative on X, if there exists a storage function $\lambda : X \to \mathbb{R}$, bounded from below, and $\alpha \in \mathcal{K}$ such that for all $x \in X$, $u \in U$ with $g(x,u) \in X$:

$$\lambda(g(x,u)) \leq \lambda(x) + \ell(x,u) - \ell(x^e,u^e) - \alpha(\|x - x^e\|)$$

Facts (for stabilizable systems):

- Strict dissipativity implies the turnpike property [Gr. '13]
- Under a controllability condition, it is equivalent to a robust version of the turnpike property [Gr./Müller '16]

  (this is analogous to "asymptotic stability $\Leftrightarrow$ existence of a Lyapunov function")

Strict dissipativity

The optimal control problem is called strictly dissipative on X, if there exists a storage function $\lambda : X \to \mathbb{R}$, bounded from below, and $\alpha \in \mathcal{K}$ such that for all $x \in X$, $u \in U$ with $g(x,u) \in X$:

$$\lambda(g(x,u)) \leq \lambda(x) + \ell(x,u) - \ell(x^e,u^e) - \alpha(\|x - x^e\|)$$

Facts (for stabilizable systems):

- Strict dissipativity implies the turnpike property [Gr. '13]
- Under a controllability condition, it is equivalent to a robust version of the turnpike property [Gr./Müller '16]

  (this is analogous to "asymptotic stability $\Leftrightarrow$ existence of a Lyapunov function")
- For linear-quadratic problems, strict dissipativity is equivalent to classical systems theoretic properties like detectability or weaker variants thereof [Gr./Guglielmi '18, '20]

Strict dissipativity

The optimal control problem is called strictly dissipative on X, if there exists a storage function $\lambda : X \to \mathbb{R}$, bounded from below, and $\alpha \in \mathcal{K}$ such that for all $x \in X$, $u \in U$ with $g(x,u) \in X$:

$$\lambda(g(x,u)) \leq \lambda(x) + \ell(x,u) - \ell(x^e,u^e) - \alpha(\|x - x^e\|)$$

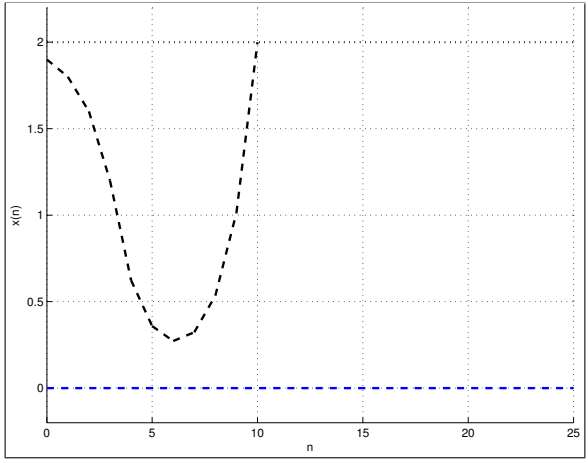
Facts (for stabilizable systems):

- Strict dissipativity implies the turnpike property [Gr. '13]
- Under a controllability condition, it is equivalent to a robust version of the turnpike property [Gr./Müller '16]

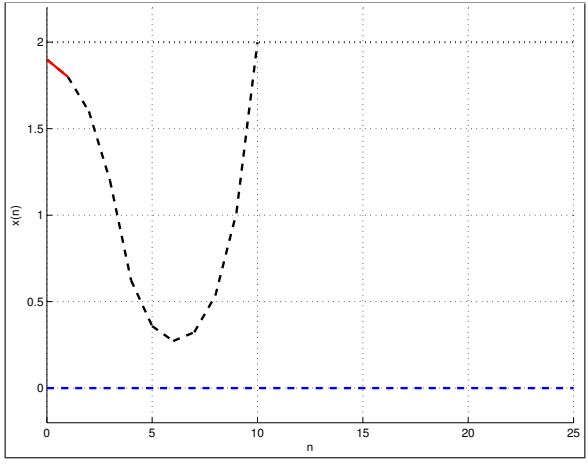
  (this is analogous to "asymptotic stability $\Leftrightarrow$ existence of a Lyapunov function")
- For linear-quadratic problems, strict dissipativity is equivalent to classical systems theoretic properties like detectability or weaker variants thereof [Gr./Guglielmi '18, '20]
- Tracking type problems are strictly dissipative with $\lambda \equiv 0$

Main Performance Result

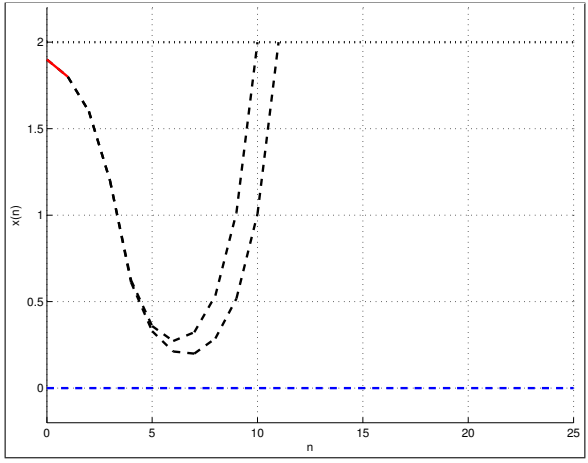
MPC in presence of the turnpike property



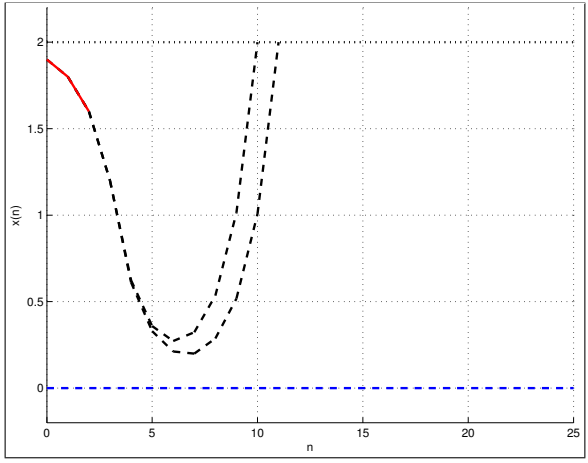
MPC in presence of the turnpike property



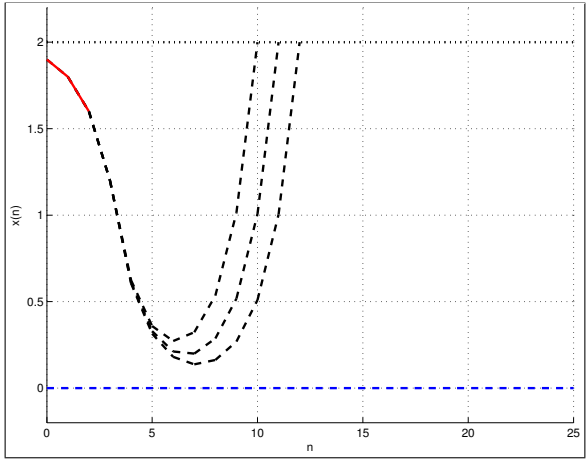
MPC in presence of the turnpike property



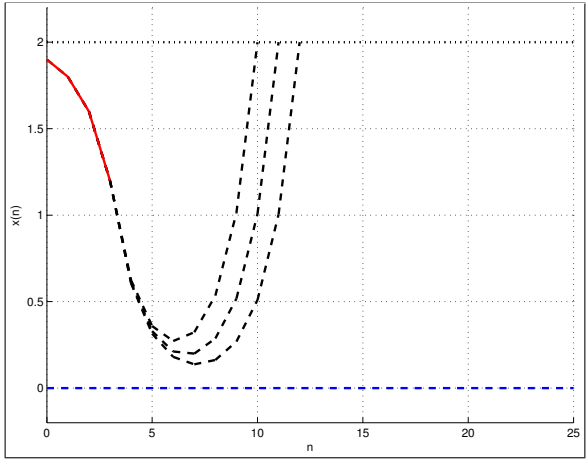
MPC in presence of the turnpike property



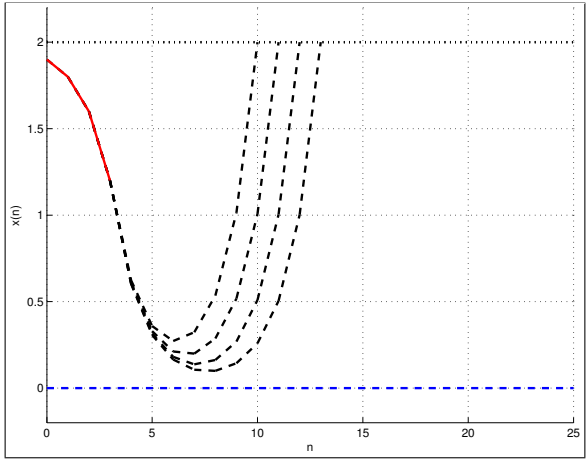
MPC in presence of the turnpike property



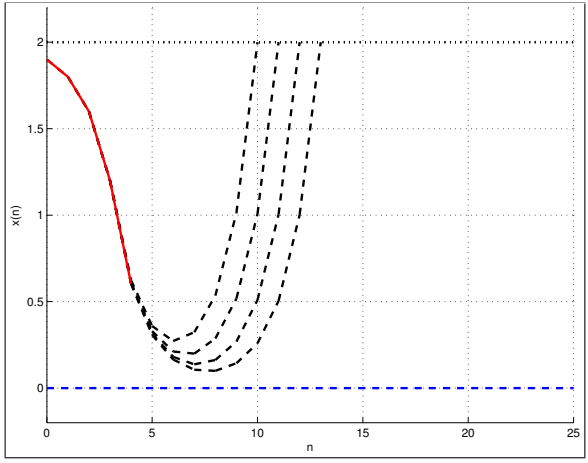
MPC in presence of the turnpike property



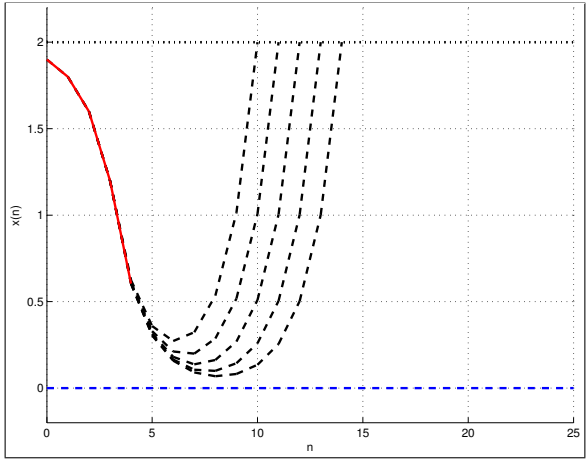
MPC in presence of the turnpike property



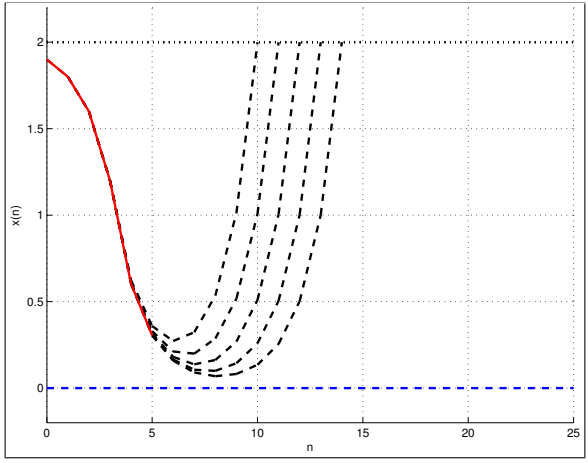
MPC in presence of the turnpike property



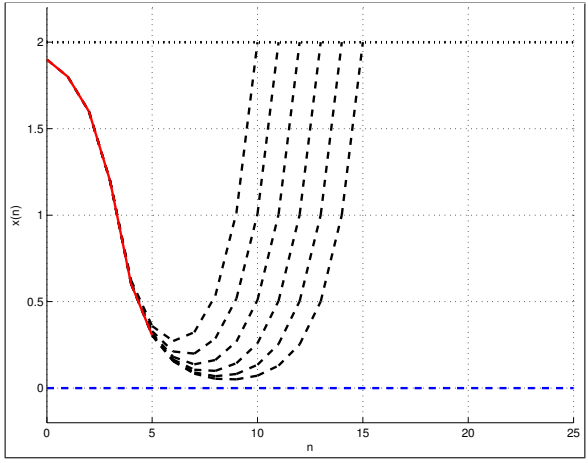
MPC in presence of the turnpike property



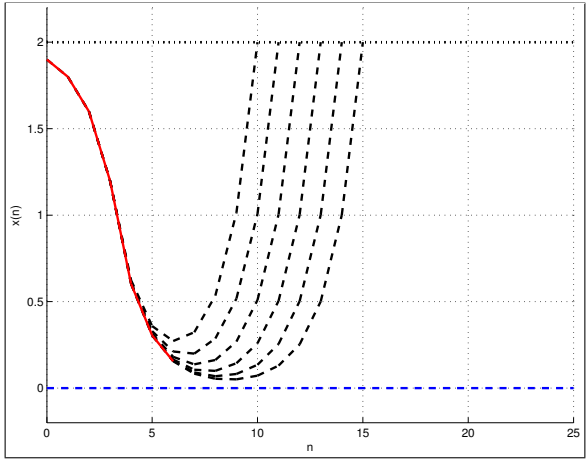
MPC in presence of the turnpike property



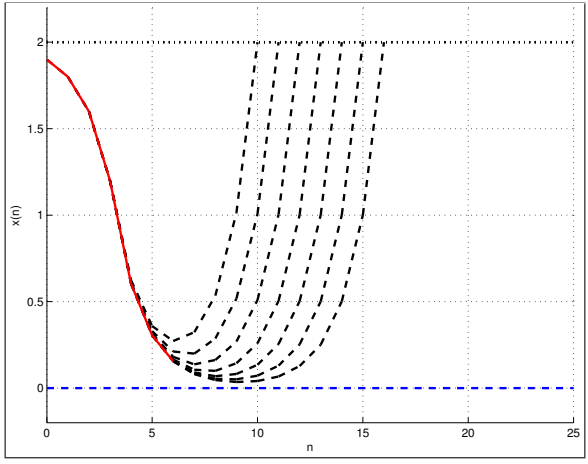
MPC in presence of the turnpike property



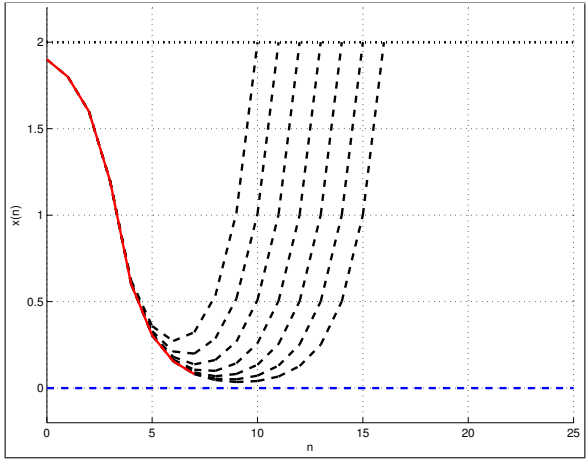
MPC in presence of the turnpike property



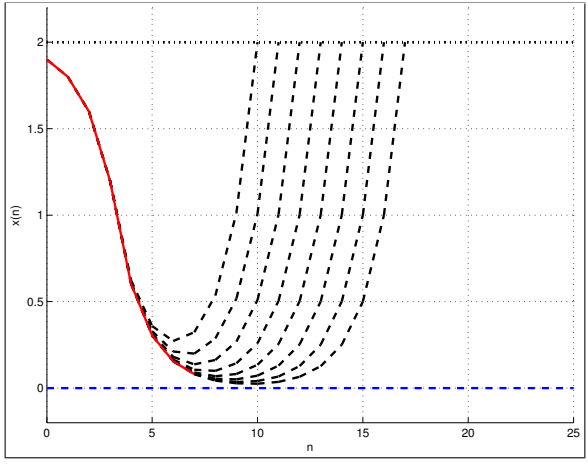
MPC in presence of the turnpike property



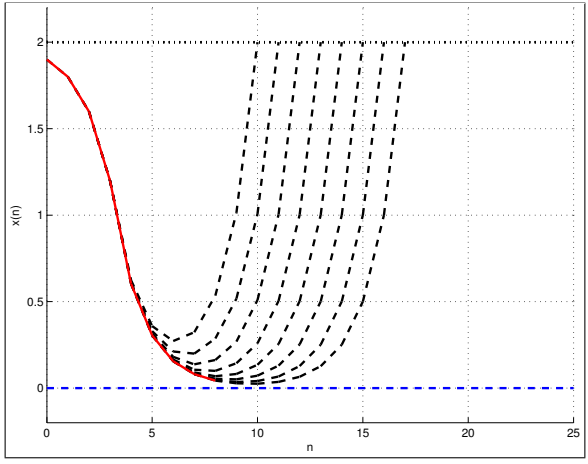
MPC in presence of the turnpike property



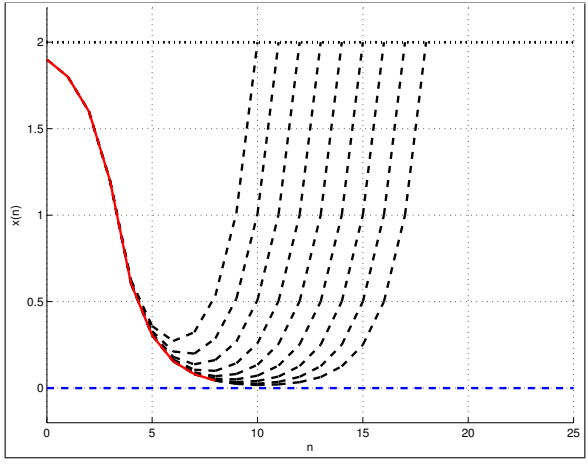
MPC in presence of the turnpike property



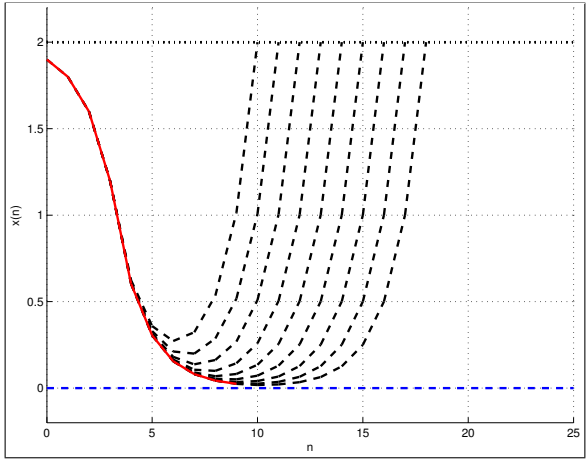
MPC in presence of the turnpike property



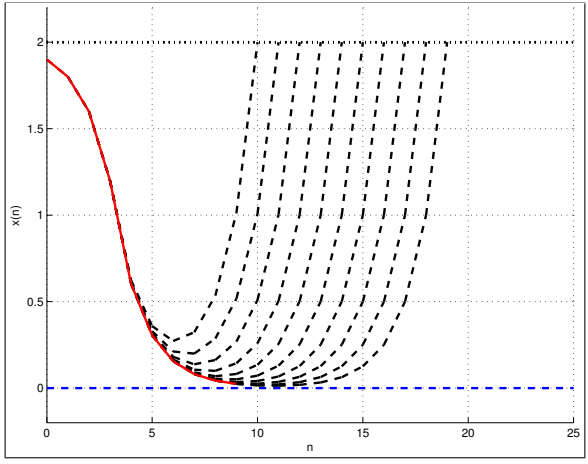
MPC in presence of the turnpike property



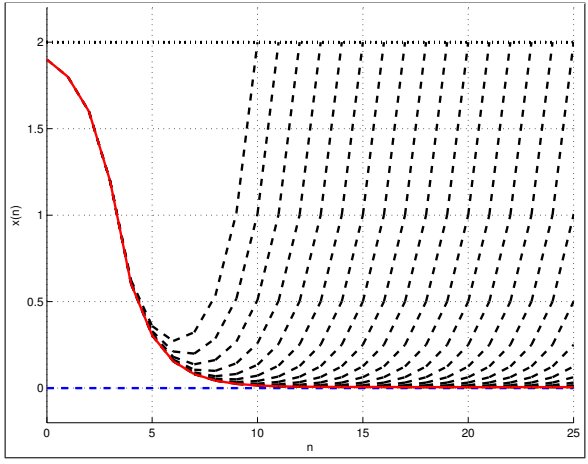
MPC in presence of the turnpike property



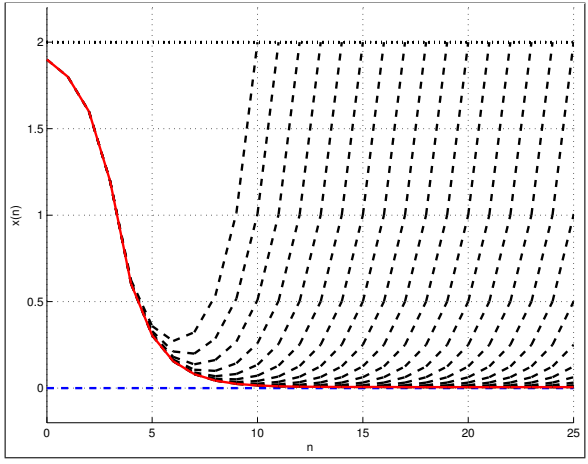
MPC in presence of the turnpike property



MPC in presence of the turnpike property



MPC in presence of the turnpike property



This behaviour allows to prove the following performance theorem

MPC via strict dissipativity — Theorem

Theorem: [Gr./Stieler '14] Consider an optimal control problem which is strictly dissipative at an equilibrium x^e

MPC via strict dissipativity — Theorem

Theorem: [Gr./Stieler '14] Consider an optimal control problem which is strictly dissipative at an equilibrium x^e

Assume moreover that the optimal value functions V_N are equicontinuous at x^e for all $N \in \mathbb{N} \cup \{\infty\}$

MPC via strict dissipativity — Theorem

Theorem: [Gr./Stieler '14] Consider an optimal control problem which is strictly dissipative at an equilibrium x^e

Assume moreover that the optimal value functions V_N are equicontinuous at x^e for all $N \in \mathbb{N} \cup \{\infty\}$

Then the MPC closed loop is

- semiglobally practically asymptotically stable
- approximately transient optimal
- approximately averaged optimal

MPC via strict dissipativity — Theorem

Theorem: [Gr./Stieler '14] Consider an optimal control problem which is strictly dissipative at an equilibrium x^e

Assume moreover that the optimal value functions V_N are equicontinuous at x^e for all $N \in \mathbb{N} \cup \{\infty\}$

Then the MPC closed loop is

- semiglobally practically asymptotically stable
- approximately transient optimal
- approximately averaged optimal

We explain the first two properties graphically

Illustration of transient optimality

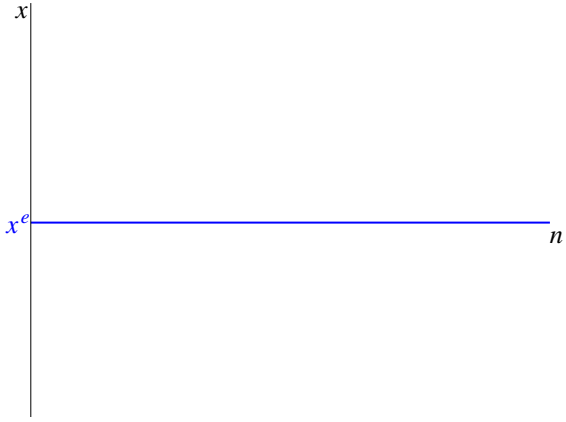


Illustration of transient optimality

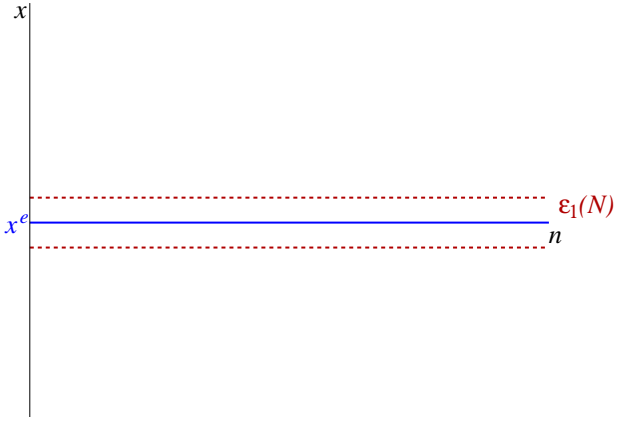
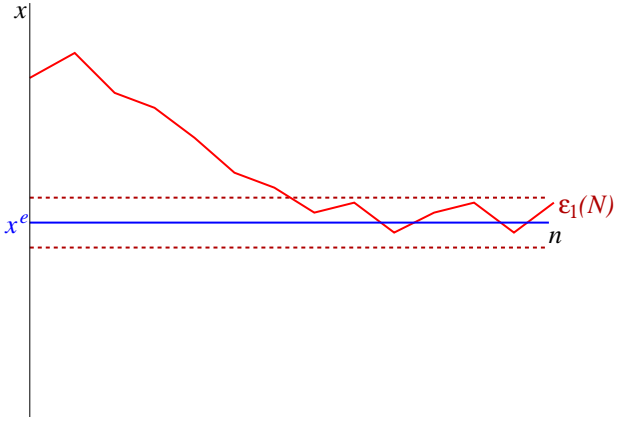


Illustration of transient optimality

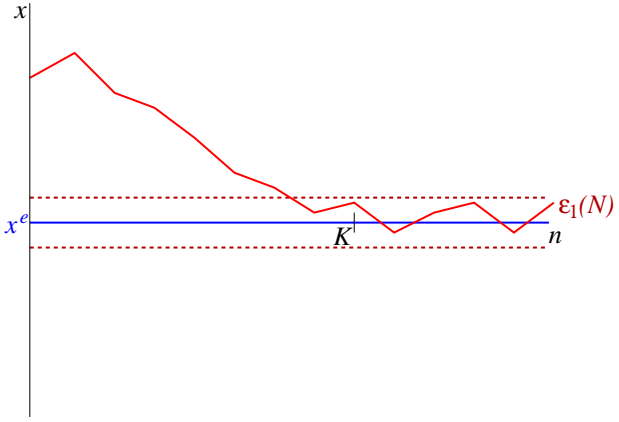


Practical asymptotic stability:

$x_{MPC}(n)$ converges to the $\varepsilon_1(N)$-ball around x^e

(with $\varepsilon_1(N) \to 0$ as $N \to \infty$)

Illustration of transient optimality

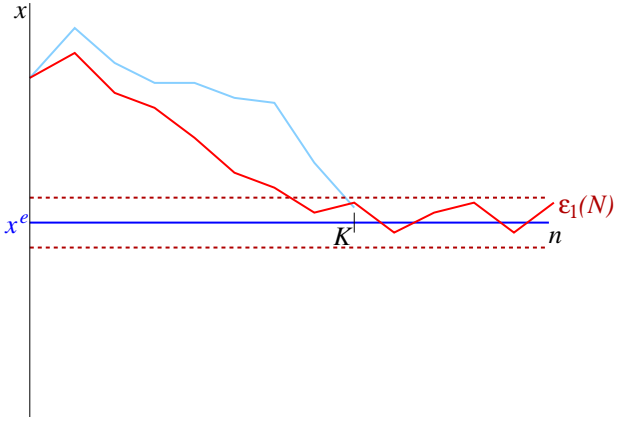


Practical asymptotic stability:

$x_{MPC}(n)$ converges to the $\varepsilon_1(N)$-ball around x^e

(with $\varepsilon_1(N) \to 0$ as $N \to \infty$)

Illustration of transient optimality

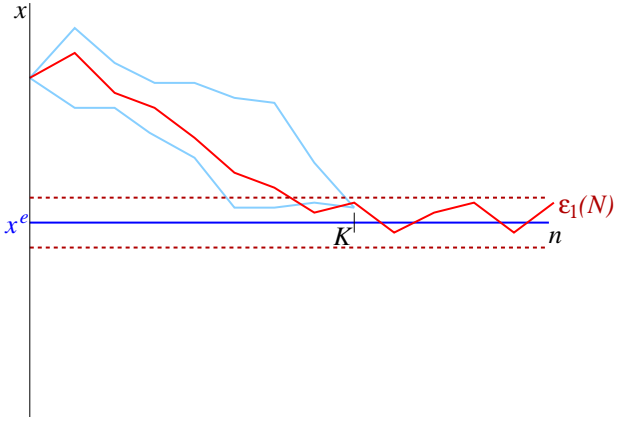


Practical asymptotic stability:

$x_{MPC}(n)$ converges to the $\varepsilon_1(N)$-ball around x^e

(with $\varepsilon_1(N) \to 0$ as $N \to \infty$)

Illustration of transient optimality

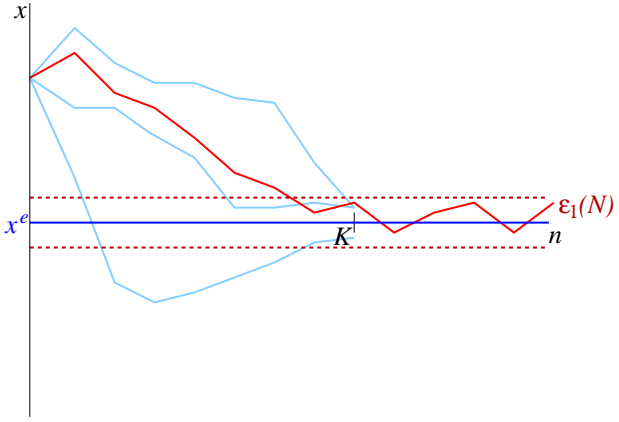


Practical asymptotic stability:

$x_{MPC}(n)$ converges to the $\varepsilon_1(N)$-ball around x^e

(with $\varepsilon_1(N) \to 0$ as $N \to \infty$)

Illustration of transient optimality

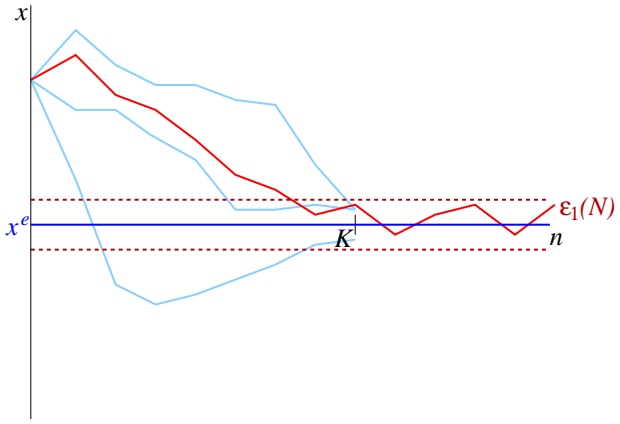


Practical asymptotic stability:

$x_{MPC}(n)$ converges to the $\varepsilon_1(N)$-ball around x^e

(with $\varepsilon_1(N) \to 0$ as $N \to \infty$)

Illustration of transient optimality



Transient optimality:

cost of all other trajectories reaching the ball at time K is higher than that of $x_{MPC}(n)$ up to an error $K\varepsilon_1(N) + \varepsilon_2(K)$

(with $\varepsilon_1(N), \varepsilon_2(K) \to 0$ as $N, K \to \infty$)

Use case

Use case

—

optimal startup of a combined cycle power plant

Combined Cycle Power Plants

C. Stadler/Bwag, wikipedia

Combined cycle power plants generate electrical energy via a combination of gas and steam turbines

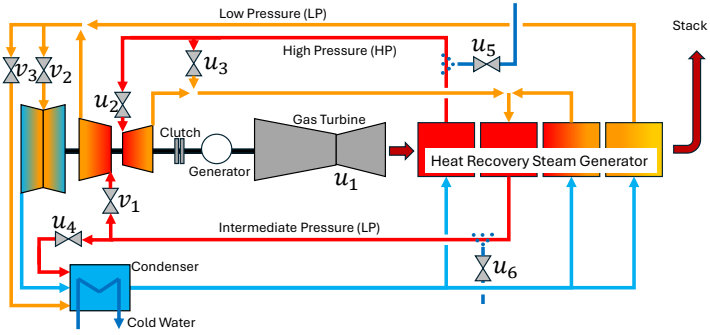
Combined Cycle Power Plants

C. Stadler/Bwag, wikipedia

Combined cycle power plants generate electrical energy via a combination of gas and steam turbines

The exhaust gas from the gas turbine produces steam for a steam turbine

Combined Cycle Power Plants

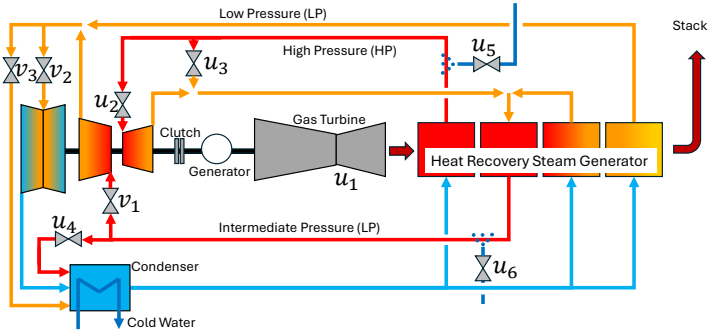


u_1: exhaust gas temperature	u_4: IP bypass valve	v_1: IP turb. control valve
u_2: HP turbine control valve	u_5: HP spray valve	v_2: LP turb. control valve
u_3: HP bypass valve	u_6: RH spray valve	v_3: LP bypass

Combined cycle power plants generate electrical energy via a combination of gas and steam turbines

The exhaust gas from the gas turbine produces steam for a steam turbine

Combined Cycle Power Plants



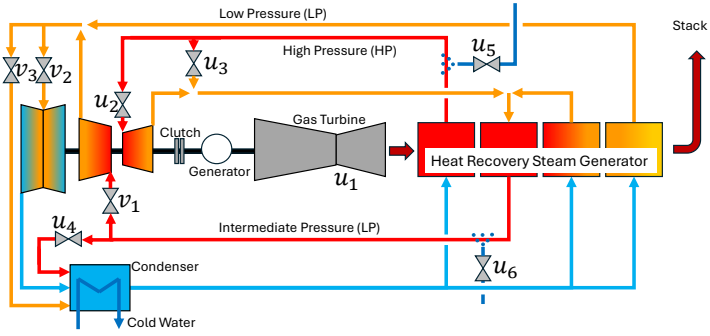
u_1: exhaust gas temperature	u_4: IP bypass valve	v_1: IP turb. control valve
u_2: HP turbine control valve	u_5: HP spray valve	v_2: LP turb. control valve
u_3: HP bypass valve	u_6: RH spray valve	v_3: LP bypass

Combined cycle power plants generate electrical energy via a combination of gas and steam turbines

The exhaust gas from the gas turbine produces steam for a steam turbine

This way, up to 64% efficiency can be reached (and even more if the remaining heat is used for district heating)

Combined Cycle Power Plants



u_1: exhaust gas temperature	u_4: IP bypass valve	v_1: IP turb. control valve
u_2: HP turbine control valve	u_5: HP spray valve	v_2: LP turb. control valve
u_3: HP bypass valve	u_6: RH spray valve	v_3: LP bypass

Combined cycle power plants generate electrical energy via a combination of gas and steam turbines

The exhaust gas from the gas turbine produces steam for a steam turbine

This way, up to 64% efficiency can be reached (and even more if the remaining heat is used for district heating)

Goal: Develop control strategies for flexible, fast, and economic start-up

Control Objective and Constraints

Objective:

- reach a desired load level with minimal fuel consumption $\rightsquigarrow$ economic MPC

Control Objective and Constraints

Objective:

- reach a desired load level with minimal fuel consumption $\rightsquigarrow$ economic MPC
- practical constraint: electricity is traded one day ahead
  $\rightsquigarrow$ optimal reference is computed in advance and tracked by MPC

Control Objective and Constraints

Objective:

- reach a desired load level with minimal fuel consumption $\rightsquigarrow$ economic MPC
- practical constraint: electricity is traded one day ahead
  $\rightsquigarrow$ optimal reference is computed in advance and tracked by MPC

Constraints:

- ensure safe plant operation, e.g., avoid too high pressures

Control Objective and Constraints

Objective:

- reach a desired load level with minimal fuel consumption $\rightsquigarrow$ economic MPC
- practical constraint: electricity is traded one day ahead
  $\rightsquigarrow$ optimal reference is computed in advance and tracked by MPC

Constraints:

- ensure safe plant operation, e.g., avoid too high pressures
- anticipate the behaviour of the underlying low level control system

Control Objective and Constraints

Objective:

- reach a desired load level with minimal fuel consumption $\rightsquigarrow$ economic MPC
- practical constraint: electricity is traded one day ahead
  $\rightsquigarrow$ optimal reference is computed in advance and tracked by MPC

Constraints:

- ensure safe plant operation, e.g., avoid too high pressures
- anticipate the behaviour of the underlying low level control system
- avoid excessive wear and tear: if steam temperatures rise much faster than wall temperatures, high temperature gradients occur, causing thermal stress, which should be limited

Control Objective and Constraints

Objective:

- reach a desired load level with minimal fuel consumption $\rightsquigarrow$ economic MPC
- practical constraint: electricity is traded one day ahead
  $\rightsquigarrow$ optimal reference is computed in advance and tracked by MPC

Constraints:

- ensure safe plant operation, e.g., avoid too high pressures
- anticipate the behaviour of the underlying low level control system
- avoid excessive wear and tear: if steam temperatures rise much faster than wall temperatures, high temperature gradients occur, causing thermal stress, which should be limited

Possible measures against high temperature gradients:

- injection of water to cool down the steam

Control Objective and Constraints

Objective:

- reach a desired load level with minimal fuel consumption $\rightsquigarrow$ economic MPC
- practical constraint: electricity is traded one day ahead
  $\rightsquigarrow$ optimal reference is computed in advance and tracked by MPC

Constraints:

- ensure safe plant operation, e.g., avoid too high pressures
- anticipate the behaviour of the underlying low level control system
- avoid excessive wear and tear: if steam temperatures rise much faster than wall temperatures, high temperature gradients occur, causing thermal stress, which should be limited

Possible measures against high temperature gradients:

- injection of water to cool down the steam
- slower ramp up of gas turbine

Model

The model is derived from

- conservation of mass

Model

The model is derived from

- conservation of mass

$$\dot{m} = \sum_{i \in I_m} \dot{m}_i \qquad [kg \cdot s^{-1}]$$

m: system mass $[kg]$
m_i: mass of medium entering (> 0) / leaving (< 0) the system $[kg]$
I_m: set of indices for medium entering/leaving the system

Model

The model is derived from

- conservation of mass
- conservation of energy

Model

The model is derived from

- conservation of mass
- conservation of energy

$$\frac{d}{dt}(mu) = \sum_{i \in I_Q} \dot{Q}_i - \sum_{i \in I_W} \dot{W}_i + \sum_{i \in I_m} h_{m_i} \dot{m}_i \qquad [W]$$

m: system mass $[kg]$
u: specific internal energy of system $[J \cdot kg^{-1}]$
Q_i: heat transfer into (> 0) / out (< 0) of the system $[J]$
W_i: work performed by (> 0) / on (< 0) the system $[J]$
m_i: mass of medium entering (> 0) / leaving (< 0) the system $[kg]$
h_i: specific enthalpy of medium entering / leaving the system $[J \cdot kg^{-1}]$
I_Q: set of indices for heat transferred into/out of the system
I_W: set of indices for work performed by/on the system
I_m: set of indices for medium entering/leaving the system

Model

The model is derived from

- conservation of mass
- conservation of energy
- convective and conductive heat-transfer

Model

The model is derived from

- conservation of mass
- conservation of energy
- convective and conductive heat-transfer

Convective heat transfer, e.g., steam $\leftrightarrow$ wall:

$$\dot{Q} = \alpha \cdot A \cdot (T - T_f) \qquad [W]$$

Q: heat transfer $[J]$
α: heat transfer coefficient $[W \cdot m^{-2} \cdot K^{-1}]$
A: contact area between fluid and material $[m^2]$
T: material temperature $[K]$
T_f: fluid temperature $[K]$

Model

The model is derived from

- conservation of mass
- conservation of energy
- convective and conductive heat-transfer

Conductive heat transfer, inside material:

$$\dot{Q} = \lambda \cdot A \cdot \frac{dT}{dx} \qquad [W]$$

Q: heat transfer $[J]$
λ: conductivity $[W \cdot m^{-1} \cdot K^{-1}]$
A: cross-sectional area $[m^2]$
T: material temperature $[K]$
x: location $[m]$

Model

The model is derived from

- conservation of mass
- conservation of energy
- convective and conductive heat-transfer
- kinetic energy of rotation bodies (shaft)

Model

The model is derived from

- conservation of mass
- conservation of energy
- convective and conductive heat-transfer
- kinetic energy of rotation bodies (shaft)
- work performed by turbines

Model

The model is derived from

- conservation of mass
- conservation of energy
- convective and conductive heat-transfer
- kinetic energy of rotation bodies (shaft)
- work performed by turbines

After reduction: $\sim$ 30 differential states, 300 algebraic states, and 430 constraints

Model

The model is derived from

- conservation of mass
- conservation of energy
- convective and conductive heat-transfer
- kinetic energy of rotation bodies (shaft)
- work performed by turbines

After reduction: $\sim$ 30 differential states, 300 algebraic states, and 430 constraints

The economic optimization objective is

$$\int_{t_0}^{\infty} \mu(t) \cdot \varphi(t) dt$$

with $\mu(t) = 1$ if the desired load level is not yet reached and $\mu(t) = 0$ otherwise, $\varphi(t) =$ fuel consumption at time t

Designing the Turnpike

- As a rule of thump, the prediction horizon should be long enough that the optimizer can find the way to the turnpike

Designing the Turnpike

- As a rule of thump, the prediction horizon should be long enough that the optimizer can find the way to the turnpike
- In order to reach the turnpike, the prediction horizon for the economic optimization is about 1 hour (at least)

Designing the Turnpike

- As a rule of thump, the prediction horizon should be long enough that the optimizer can find the way to the turnpike
- In order to reach the turnpike, the prediction horizon for the economic optimization is about 1 hour (at least). This leads to optimization times that are prohibitive for online implementation

Designing the Turnpike

- As a rule of thump, the prediction horizon should be long enough that the optimizer can find the way to the turnpike
- In order to reach the turnpike, the prediction horizon for the economic optimization is about 1 hour (at least). This leads to optimization times that are prohibitive for online implementation

  $\rightsquigarrow$ even if no day ahead reference is desired, it is beneficial to compute the economically optimal trajectory in advance and track it by tracking MPC in the online phase

Designing the Turnpike

- As a rule of thump, the prediction horizon should be long enough that the optimizer can find the way to the turnpike
- In order to reach the turnpike, the prediction horizon for the economic optimization is about 1 hour (at least). This leads to optimization times that are prohibitive for online implementation
  - $\rightsquigarrow$ even if no day ahead reference is desired, it is beneficial to compute the economically optimal trajectory in advance and track it by tracking MPC in the online phase
- For tracking MPC, the reference trajectory is the turnpike

Designing the Turnpike

- As a rule of thump, the prediction horizon should be long enough that the optimizer can find the way to the turnpike
- In order to reach the turnpike, the prediction horizon for the economic optimization is about 1 hour (at least). This leads to optimization times that are prohibitive for online implementation

  $\rightsquigarrow$ even if no day ahead reference is desired, it is beneficial to compute the economically optimal trajectory in advance and track it by tracking MPC in the online phase

- For tracking MPC, the reference trajectory is the turnpike. If the system state is sufficiently close, a prediction horizon of a few minutes is suitable

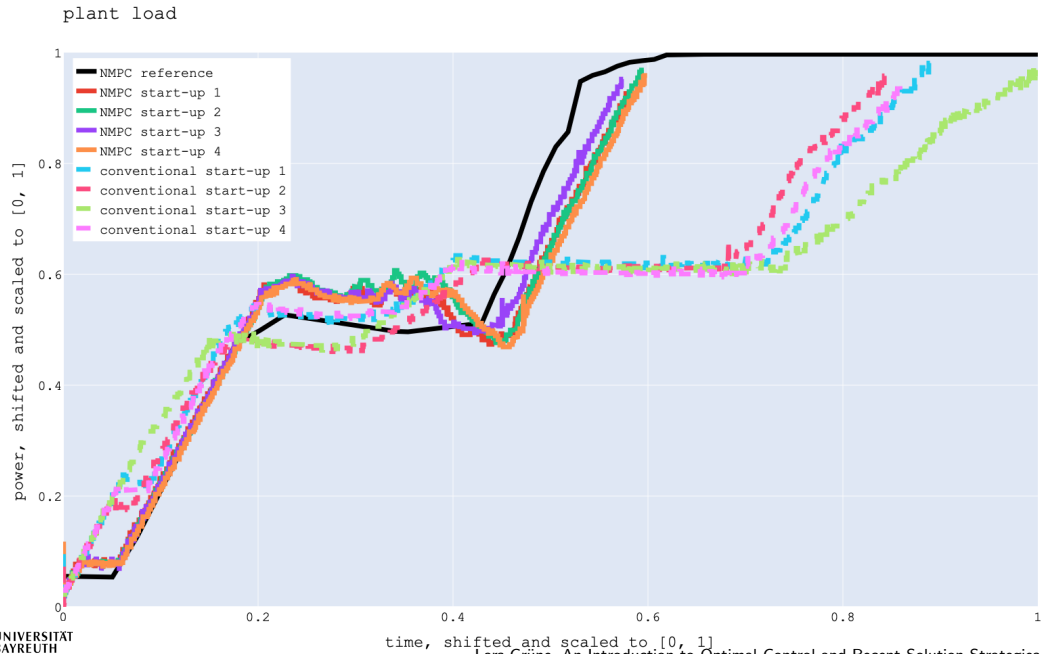
Designing the Turnpike

- As a rule of thump, the prediction horizon should be long enough that the optimizer can find the way to the turnpike
- In order to reach the turnpike, the prediction horizon for the economic optimization is about 1 hour (at least). This leads to optimization times that are prohibitive for online implementation

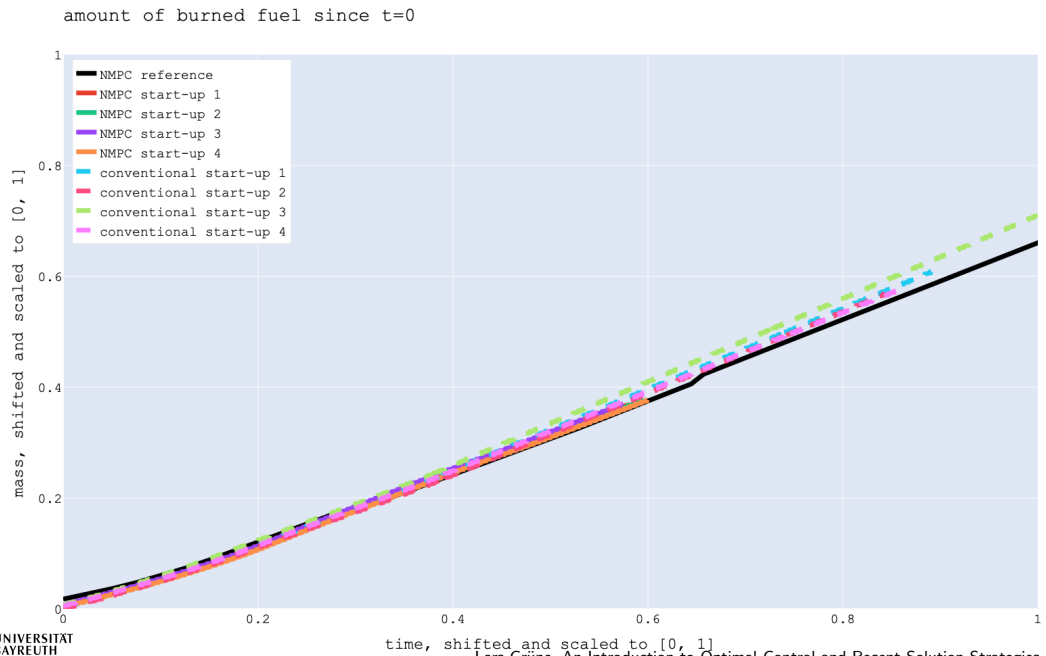
  $\rightsquigarrow$ even if no day ahead reference is desired, it is beneficial to compute the economically optimal trajectory in advance and track it by tracking MPC in the online phase

- For tracking MPC, the reference trajectory is the turnpike. If the system state is sufficiently close, a prediction horizon of a few minutes is suitable
- Since in the online phase the initial state at the beginning of the start up is unknown, a library of optimal references with different initial states is computed in advance

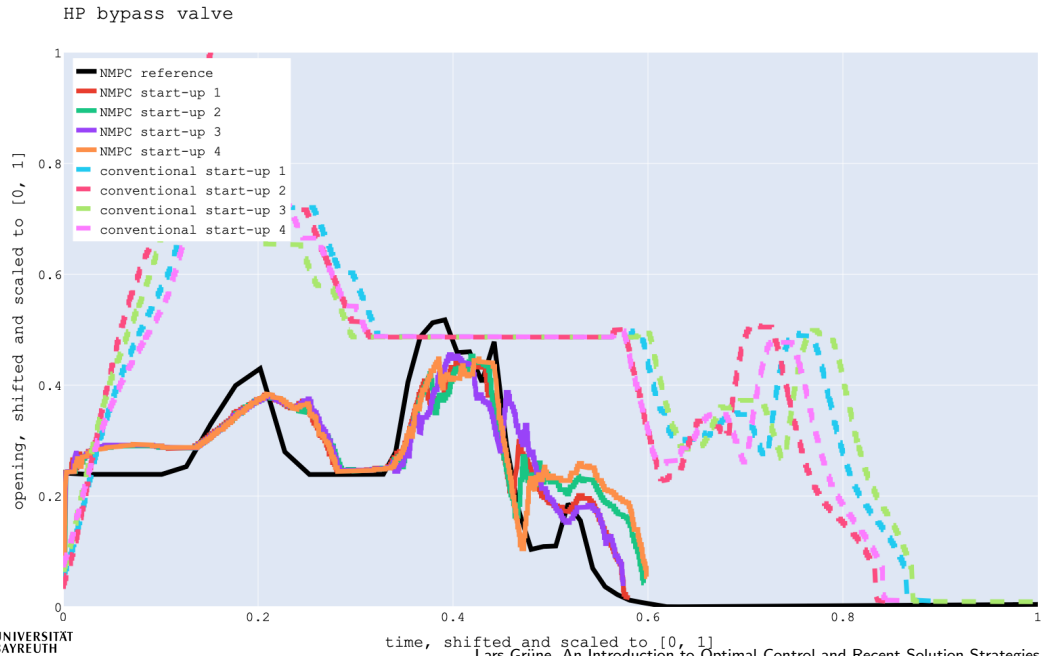
Start-up runs on the plant



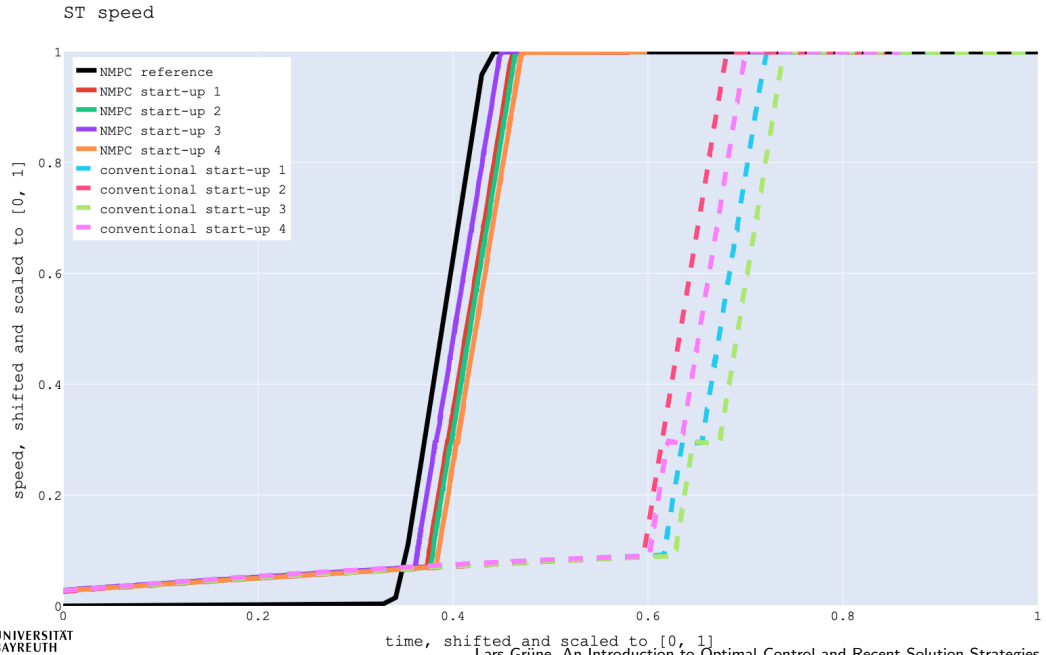
Start-up runs on the plant



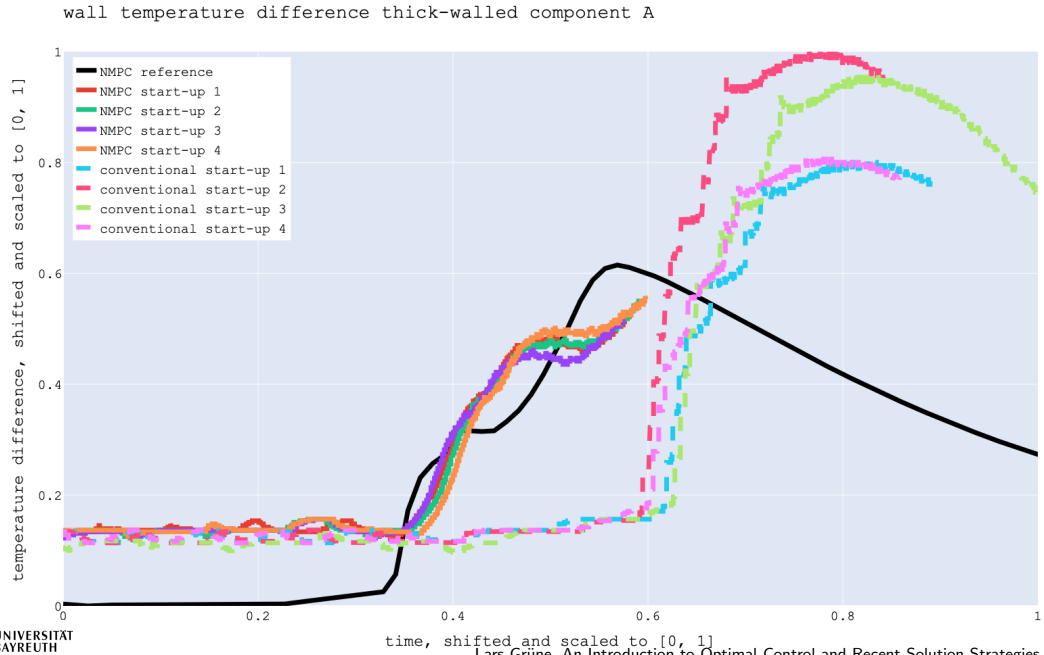
Start-up runs on the plant



Start-up runs on the plant



Start-up runs on the plant



Summary of Part 4

- Model Predictive Control (MPC) synthesizes closed-loop solutions from open-loop optimal controls and trajectories

Summary of Part 4

- Model Predictive Control (MPC) synthesizes closed-loop solutions from open-loop optimal controls and trajectories
- It generates a feedback law by online optimization

Summary of Part 4

- Model Predictive Control (MPC) synthesizes closed-loop solutions from open-loop optimal controls and trajectories
- It generates a feedback law by online optimization
- If the turnpike property and dissipativity holds, near-optimal performance can be shown

Summary of Part 4

- Model Predictive Control (MPC) synthesizes closed-loop solutions from open-loop optimal controls and trajectories
- It generates a feedback law by online optimization
- If the turnpike property and dissipativity holds, near-optimal performance can be shown
- MPC is highly successful in industrial applications

Outline of the course

Part 1: Optimal Control Problems — An Introduction

Part 2: Solution Concepts

Part 3: Numerical Solution Methods

Part 4: Model Predictive Control

Part 5: Deep Neural Networks for High-Dimensional Problems

Purpose of this part

Identify structural properties of optimal control problems, under which deep neural networks can indeed avoid the curse of dimensionality

Use ideas from distributed optimal control to describe these properties

Give rigorous estimates for the approximation error

Contents of this part

Part 5: Deep Neural Networks for High-Dimensional Problems

- Reminder: Deep Reinforcement Learning
- Separable functions
- Distributed optimal control
- Decaying Sensitivity

Deep Reinforcement Learning

Deep Reinforcement Learning

Recall the optimal value function

$$V(x) = \inf_{\mathbf{u}} J_\infty(x, \mathbf{u})$$

Deep Reinforcement Learning

Recall the optimal value function

$$V(x) = \inf_{\mathbf{u}} J_\infty(x, \mathbf{u})$$

In Deep Reinforcement Learning, deep neural networks (DNNs) are used for storing V

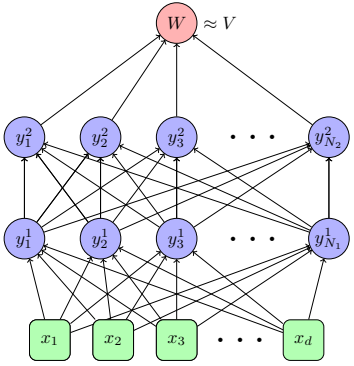
Deep Reinforcement Learning

Recall the optimal value function

$$V(x) = \inf_{\mathbf{u}} J_\infty(x, \mathbf{u})$$

In Deep Reinforcement Learning, deep neural networks (DNNs) are used for storing V (or a derivate thereof, called Q)

Deep neural network with 2 hidden layers



$$W(x;\theta) = a \cdot y^2 + c$$

$$y_k^2 = \sigma^2(w_k^2 \cdot y^1 + b_k^2)$$

$$y_k^1 = \sigma^1(w_k^1 \cdot x + b_k^1)$$

w_k^1, w_k^2, a = vectors of weights, " $\cdot$ " = scalar product

b_k^1, b_k^2, c = scalar parameters, $\sigma^1, \sigma^2 : \mathbb{R} \to \mathbb{R}$ = activation functions

Examples: $\sigma(r) = r$, $\sigma(r) = \max\{r, 0\}$, $\sigma(r) = \ln(e^r + 1)$, $\sigma(r) = \frac{1}{1+e^{-r}}$

θ = vector of all parameters $(w_k^\ell, b_k^\ell, a, c)$

$W(x;\theta^*) \approx V(x)$, approximating function for "trained" θ^*

Deep Reinforcement Learning

The network is then "trained" such that the function $W(x;\theta^*)$ with optimized parameters θ^* satisfies the Hamilton-Jacobi-Bellman equation

$$\inf_{u\in U}\{\ell(x,u)+DW(x;\theta^*)f(x,u)\}\approx 0$$

or the Bellman equation

$$\inf_{u\in U}\{\ell(x,u)+W(g(x,u);\theta^*)-W(x;\theta^*)\}\approx 0$$

as good as possible

Deep Reinforcement Learning

The network is then "trained" such that the function $W(x;\theta^*)$ with optimized parameters θ^* satisfies the Hamilton-Jacobi-Bellman equation

$$\inf_{u\in U}\{\ell(x,u)+DW(x;\theta^*)f(x,u)\}\approx 0$$

or the Bellman equation

$$\inf_{u\in U}\{\ell(x,u)+W(g(x,u);\theta^*)-W(x;\theta^*)\}\approx 0$$

as good as possible

Training DNNs is an interesting problem in itself, which we briefly discussed in Part 3

Deep Reinforcement Learning

The network is then "trained" such that the function $W(x;\theta^*)$ with optimized parameters θ^* satisfies the Hamilton-Jacobi-Bellman equation

$$\inf_{u\in U}\{\ell(x,u)+DW(x;\theta^*)f(x,u)\}\approx 0$$

or the Bellman equation

$$\inf_{u\in U}\{\ell(x,u)+W(g(x,u);\theta^*)-W(x;\theta^*)\}\approx 0$$

as good as possible

Training DNNs is an interesting problem in itself, which we briefly discussed in Part 3

Here, the following necessary property will be discussed:

Can DNNs provide approximations $W(\cdot\,;\theta^*)\approx V$ for large space dimensions d?

Universal approximation theorem

Let $K = [-\kappa, \kappa]^n$, $\kappa > 0$ fixed, $d \in \mathbb{N}$ varying

Universal approximation theorem

Let $K = [-\kappa, \kappa]^n$, $\kappa > 0$ fixed, $d \in \mathbb{N}$ varying, $\quad \|V\|_{\infty,K} := \max_{x \in K} |V(x)|$

Universal approximation theorem

Let $K = [-\kappa, \kappa]^n$, $\kappa > 0$ fixed, $d \in \mathbb{N}$ varying, $\|V\|_{\infty,K} := \max_{x \in K} |V(x)|$, and

$$\mathcal{W}_1^n := \left\{ V \in C^1(K, \mathbb{R}) \;\middle|\; \sum_{i=1}^{n} \left\| \frac{\partial}{\partial x_i} V \right\|_{\infty,K} \leq 1 \right\}$$

Universal approximation theorem

Let $K = [-\kappa, \kappa]^n$, $\kappa > 0$ fixed, $d \in \mathbb{N}$ varying, $\|V\|_{\infty,K} := \max_{x \in K} |V(x)|$, and

$$\mathcal{W}_1^n := \left\{ V \in C^1(K, \mathbb{R}) \;\middle|\; \sum_{i=1}^{n} \left\| \frac{\partial}{\partial x_i} V \right\|_{\infty,K} \leq 1 \right\}$$

Theorem [Cybenko '89, Mhaskar '96, Poggio et al. '17]:
Let $\sigma^1 : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable and not polynomial. Then, for any $\varepsilon > 0$, a neural network with one hidden layer provides an approximation

$$\inf_{\theta \in \mathbb{R}^P} \|W(x;\theta) - V(x)\|_{\infty,K} \leq \varepsilon$$

for any $V \in \mathcal{W}_1^n$ with a minimal number of neurons

$$N = \mathcal{O}\left(\varepsilon^{-n}\right)$$

Quiz

Assume the required number of neurons for achieving the accuracy $\varepsilon > 0$ is $\mathcal{O}\left(\varepsilon^{-n}\right)$.

By how much does the number increase if we want to reduce the accuracy ε to $\varepsilon/5$?

- By a factor of 5
- By a factor of n^5
- By a factor of 5^n

Quiz

Assume the required number of neurons for achieving the accuracy $\varepsilon > 0$ is $\mathcal{O}\left(\varepsilon^{-n}\right)$.

By how much does the number increase if we want to reduce the accuracy ε to $\varepsilon/5$?

- By a factor of 5
- By a factor of n^5
- By a factor of 5^n

What is worse?

Quiz

Assume the required number of neurons for achieving the accuracy $\varepsilon > 0$ is $\mathcal{O}\left(\varepsilon^{-n}\right)$.

By how much does the number increase if we want to reduce the accuracy ε to $\varepsilon/5$?

- By a factor of 5
- By a factor of n^5
- By a factor of 5^n

What is worse?

E.g., for $n = 10$: $\quad 5^{10} \approx 10^7 > 10^5$

Separable functions

What are separable functions and why are they beneficial?

Separable function:

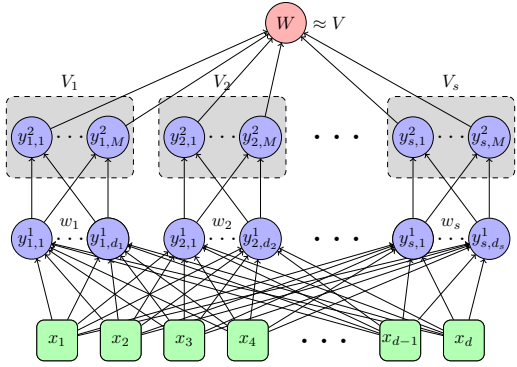
$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$

What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$

We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d



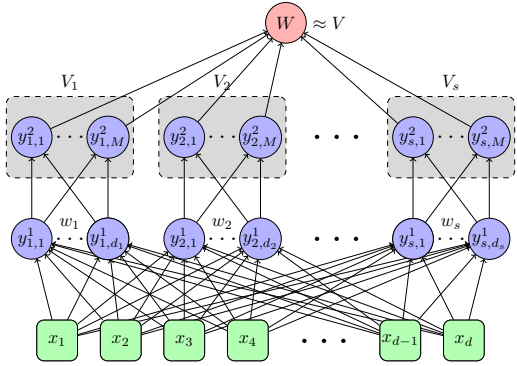
What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$

We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d

Applying the universal approximation theorem separately in each grey block, we can prove:



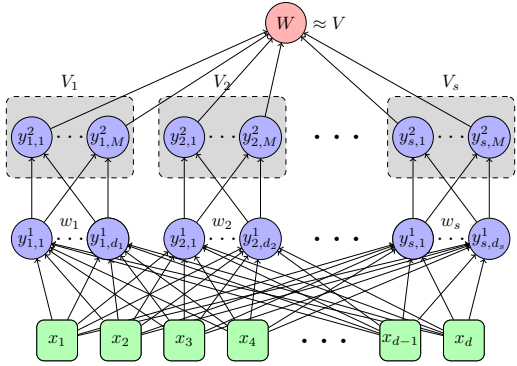
What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$

We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d

Applying the universal approximation theorem separately in each grey block, we can prove:



Theorem [Gr. 21]: Functions $V(x) = \sum_{j=1}^{s} V_j(w_j)$ with $V_j \in \mathcal{W}_1^{d_j}$ and $d_j \leq d_{\max}$ independent of d can be approximated on K with any accuracy $\varepsilon > 0$ with a number of neurons growing only polynomially in d

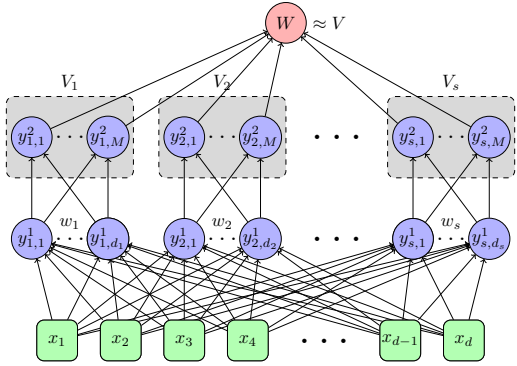
What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$

We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d

Applying the universal approximation theorem separately in each grey block, we can prove:



Theorem [Gr. 21]: Functions $V(x) = \sum_{j=1}^{s} V_j(w_j)$ with $V_j \in \mathcal{W}_1^{d_j}$ and $d_j \leq d_{\max}$ independent of d can be approximated on K with any accuracy $\varepsilon > 0$ with a number of neurons growing only polynomially in d

More precisely, the number of required neurons is $\mathcal{O}\left(\varepsilon^{-d_{\max}}\right) \mathcal{O}\left(d^{d_{\max}+1}\right)$

Distributed optimal control

Setting for distributed control

We assume that the system can be decomposed into s subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}, u_i), \quad i = 1, \ldots, s, \qquad z_{-i} = \begin{pmatrix} z_1 \\ \vdots \\ z_{i-1} \\ z_{i+1} \\ \vdots \\ z_s \end{pmatrix}$$

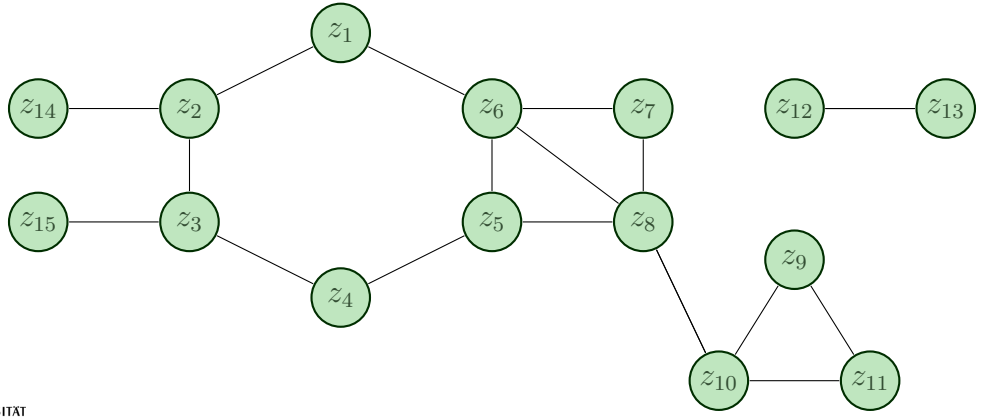
with $z_i \in \mathbb{R}^{n_i}$

Setting for distributed control

We assume that the system can be decomposed into s subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}, u_i), \quad i = 1, \dots, s, \qquad z_{-i} = \begin{pmatrix} z_1 \\ \vdots \\ z_{i-1} \\ z_{i+1} \\ \vdots \\ z_s \end{pmatrix}$$

with $z_i \in \mathbb{R}^{n_i}$, whose interconnection is expressed via an undirected graph

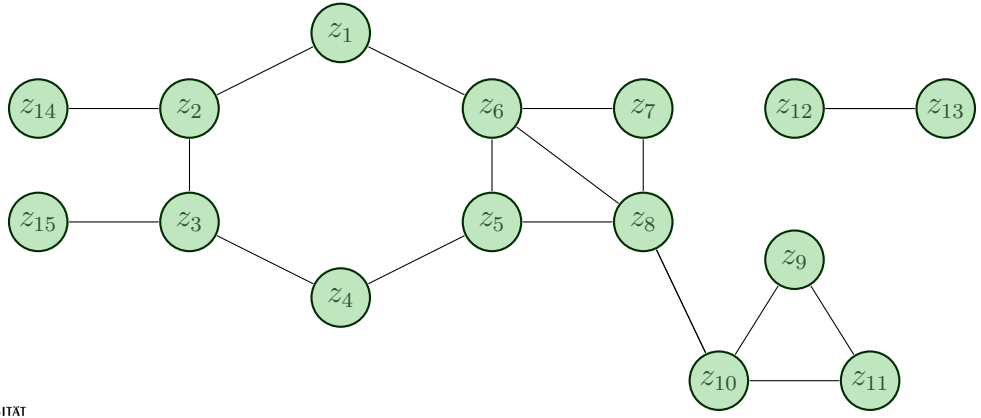


Setting for distributed control

We assume that the system can be decomposed into s subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}, u_i), \quad i = 1, \ldots, s, \qquad z_{-i} = \begin{pmatrix} z_1 \\ \vdots \\ z_{i-1} \\ z_{i+1} \\ \vdots \\ z_s \end{pmatrix}$$

with $z_i \in \mathbb{R}^{n_i}$, whose interconnection is expressed via an undirected graph: Whenever f_i depends on z_j, then there is an edge from z_i to z_j

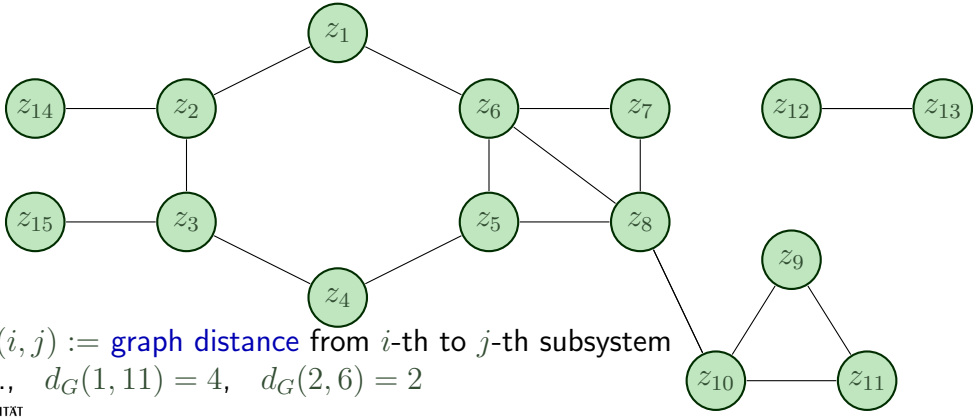


Setting for distributed control

We assume that the system can be decomposed into s subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}, u_i), \quad i = 1, \ldots, s, \qquad z_{-i} = \begin{pmatrix} z_1 \\ \vdots \\ z_{i-1} \\ z_{i+1} \\ \vdots \\ z_s \end{pmatrix}$$

with $z_i \in \mathbb{R}^{n_i}$, whose interconnection is expressed via an undirected graph: Whenever f_i depends on z_j, then there is an edge from z_i to z_j



$d_G(i,j) :=$ graph distance from i-th to j-th subsystem

e.g., $d_G(1,11) = 4$, $d_G(2,6) = 2$

Setting for distributed control

$$\dot{z}_i = f_i(z_i, z_{-i}, u_i), \quad i = 1, \ldots, s, \qquad z_{-i} = \begin{pmatrix} z_1 \\ \vdots \\ z_{i-1} \\ z_{i+1} \\ \vdots \\ z_s \end{pmatrix}$$

Likewise, we assume that the cost $\ell(x, u)$ can be written as

$$\ell(x, u) = \sum_{i=1}^{s} \ell_i(z_i, z_{-i}, u_i)$$

Setting for distributed control

$$\dot{z}_i = f_i(z_i, z_{-i}, u_i), \quad i = 1, \ldots, s, \qquad z_{-i} = \begin{pmatrix} z_1 \\ \vdots \\ z_{i-1} \\ z_{i+1} \\ \vdots \\ z_s \end{pmatrix}$$

Likewise, we assume that the cost $\ell(x, u)$ can be written as

$$\ell(x, u) = \sum_{i=1}^{s} \ell_i(z_i, z_{-i}, u_i)$$

The graph structure is then determined by the f_i and the ℓ_i together

Setting for distributed control

$$\dot{z}_i = f_i(z_i, z_{-i}, u_i), \quad i = 1, \ldots, s, \qquad z_{-i} = \begin{pmatrix} z_1 \\ \vdots \\ z_{i-1} \\ z_{i+1} \\ \vdots \\ z_s \end{pmatrix}$$

Likewise, we assume that the cost $\ell(x, u)$ can be written as

$$\ell(x, u) = \sum_{i=1}^{s} \ell_i(z_i, z_{-i}, u_i)$$

The graph structure is then determined by the f_i and the ℓ_i together

Question: How does the state of a subsystem influence the optimal trajectory of another subsystem far away?

Decaying sensitivity

Convoy example

Example: Convoy of vehicles

Convoy example

Example: Convoy of vehicles

It is known that a perturbation in the first vehicle (e.g., a braking manoeuvre) may amplify while propagating through the convoy

Convoy example

Example: Convoy of vehicles

It is known that a perturbation in the first vehicle (e.g., a braking manoeuvre) may amplify while propagating through the convoy

However, the perturbation will decrease quickly, if the vehicles are controlled optimally

Convoy example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

Convoy example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

Convoy example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty (x_1(t) - x_{ref}(t))^2 + \sum_{i=1}^{N-1} (x_{i+1}(t) - x_i(t) - L)^2 + \gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2 dt$$

Convoy example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty \underbrace{(x_1(t) - x_{ref}(t))^2}_{\text{reference for 1st vehicle}} + \sum_{i=1}^{N-1} (x_{i+1}(t) - x_i(t) - L)^2 + \gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2 dt$$

Convoy example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty \underbrace{(x_1(t) - x_{ref}(t))^2}_{\text{reference for 1st vehicle}} + \sum_{i=1}^{N-1} \underbrace{(x_{i+1}(t) - x_i(t) - L)^2}_{\text{desired distance}} + \gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2 dt$$

Convoy example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

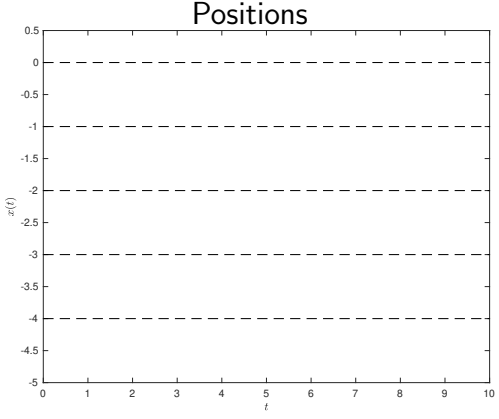
$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty \underbrace{(x_1(t) - x_{ref}(t))^2}_{\text{reference for 1st vehicle}} + \sum_{i=1}^{N-1} \underbrace{(x_{i+1}(t) - x_i(t) - L)^2}_{\text{desired distance}} + \underbrace{\gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2}_{\text{regularization terms}} dt$$

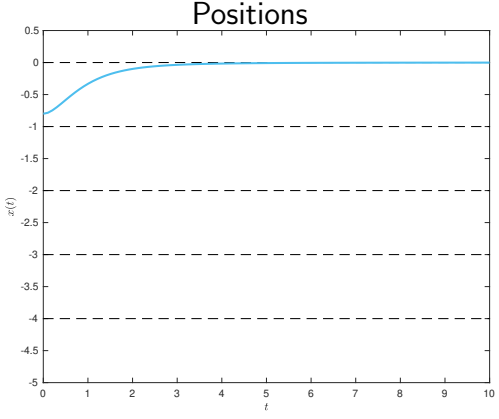
Convoy example

Optimal solutions for $N = 100$ vehicles, shown $i = 1, \ldots, 5$, $x_{ref} \equiv 0$, $v_{ref} \equiv 0$



Convoy example

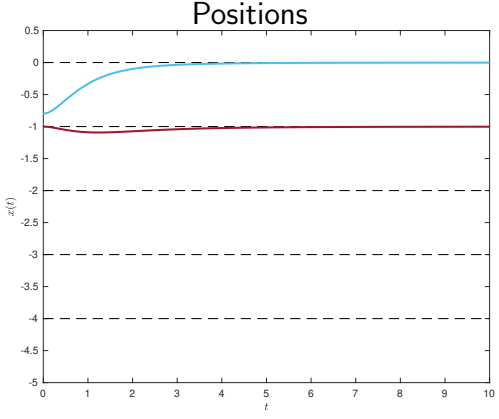
Optimal solutions for $N = 100$ vehicles, shown $i = 1, \ldots, 5$, $x_{ref} \equiv 0$, $v_{ref} \equiv 0$



$\rightsquigarrow$ The sensitivity decays with the distance from the perturbation

Convoy example

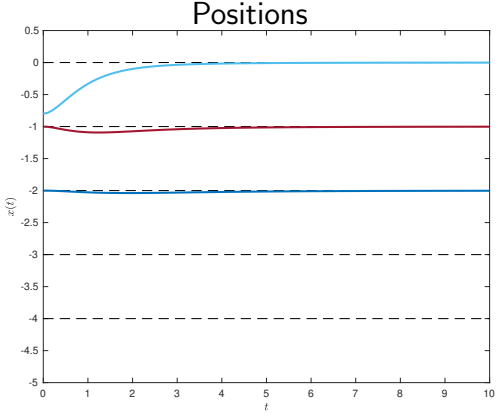
Optimal solutions for $N = 100$ vehicles, shown $i = 1, \ldots, 5$, $x_{ref} \equiv 0$, $v_{ref} \equiv 0$



$\rightsquigarrow$ The sensitivity decays with the distance from the perturbation

Convoy example

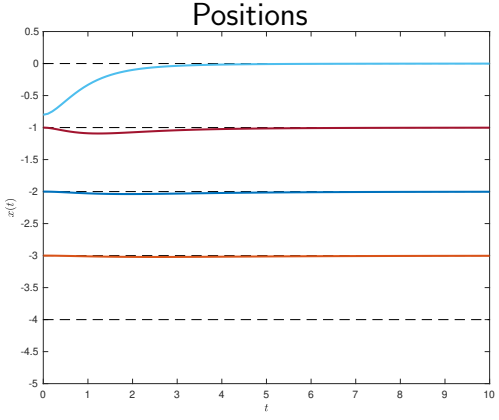
Optimal solutions for $N = 100$ vehicles, shown $i = 1, \ldots, 5$, $x_{ref} \equiv 0$, $v_{ref} \equiv 0$



$\rightsquigarrow$ The sensitivity decays with the distance from the perturbation

Convoy example

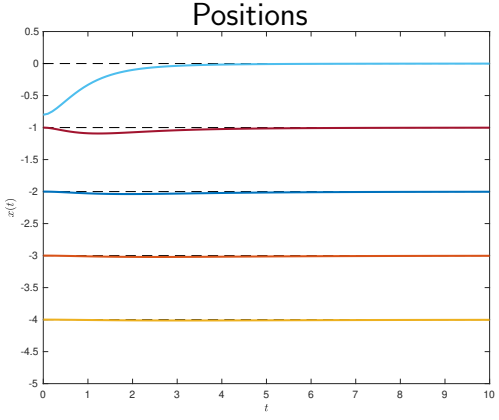
Optimal solutions for $N = 100$ vehicles, shown $i = 1, \ldots, 5$, $x_{ref} \equiv 0$, $v_{ref} \equiv 0$



$\rightsquigarrow$ The sensitivity decays with the distance from the perturbation

Convoy example

Optimal solutions for $N = 100$ vehicles, shown $i = 1, \ldots, 5$, $x_{ref} \equiv 0$, $v_{ref} \equiv 0$



$\rightsquigarrow$ The sensitivity decays with the distance from the perturbation

Decaying sensitivity for LQ optimal control

Consider linear quadratic optimal control problems, i.e., with linear dynamics $f(x,u) = Ax + Bu$ and with quadratic cost $\ell(x,u) = x^T Q x + u^T R x$

Then the optimal value function and the optimal feedback are quadratic and linear, respectively, i.e., $V(x) = x^T P x$, $u^\star = F(x) = Kx$

Decaying sensitivity for LQ optimal control

Consider linear quadratic optimal control problems, i.e., with linear dynamics $f(x,u) = Ax + Bu$ and with quadratic cost $\ell(x,u) = x^TQx + u^TRx$

Then the optimal value function and the optimal feedback are quadratic and linear, respectively, i.e., $V(x) = x^TPx, \quad u^\star = F(x) = Kx$

Theorem [Shin/Lin/Qu/Wierman/Anitescu '23]: Under suitable uniform stabilizability and detectability assumptions the inequality

$$\|K_{ij}\| \leq C\rho^{d_G(i,j)}$$

holds for a $0 < \rho < 1$, with K_{ij} being the block in K mapping z_j to u_i

Decaying sensitivity for LQ optimal control

Consider linear quadratic optimal control problems, i.e., with linear dynamics $f(x,u) = Ax + Bu$ and with quadratic cost $\ell(x,u) = x^TQx + u^TRx$

Then the optimal value function and the optimal feedback are quadratic and linear, respectively, i.e., $V(x) = x^TPx$, $u^\star = F(x) = Kx$

Theorem [Shin/Lin/Qu/Wierman/Anitescu '23]: Under suitable uniform stabilizability and detectability assumptions the inequality

$$\|K_{ij}\| \leq C\rho^{d_G(i,j)}$$

holds for a $0 < \rho < 1$, with K_{ij} being the block in K mapping z_j to u_i

(proved in discrete time but expected to hold also in continuous time)

Decaying sensitivity for LQ optimal control

Consider linear quadratic optimal control problems, i.e., with linear dynamics $f(x,u) = Ax + Bu$ and with quadratic cost $\ell(x,u) = x^TQx + u^TRx$

Then the optimal value function and the optimal feedback are quadratic and linear, respectively, i.e., $V(x) = x^TPx$, $u^\star = F(x) = Kx$

Theorem [Shin/Lin/Qu/Wierman/Anitescu '23]: Under suitable uniform stabilizability and detectability assumptions the inequality

$$\|K_{ij}\| \leq C\rho^{d_G(i,j)}$$

holds for a $0 < \rho < 1$, with K_{ij} being the block in K mapping z_j to u_i

(proved in discrete time but expected to hold also in continuous time)

Corollary [Sperl/Gr./Saluzzi/Kalise '23]: On suitable graphs this implies

$$\|P_{ij}\| \leq C\rho^{d_G(i,j)}$$

Decaying sensitivity for LQ optimal control

Consider linear quadratic optimal control problems, i.e., with linear dynamics $f(x,u) = Ax + Bu$ and with quadratic cost $\ell(x,u) = x^TQx + u^TRx$

Then the optimal value function and the optimal feedback are quadratic and linear, respectively, i.e., $V(x) = x^TPx$, $u^\star = F(x) = Kx$

Theorem [Shin/Lin/Qu/Wierman/Anitescu '23]: Under suitable uniform stabilizability and detectability assumptions the inequality

$$\|K_{ij}\| \leq C\rho^{d_G(i,j)}$$

holds for a $0 < \rho < 1$, with K_{ij} being the block in K mapping z_j to u_i

(proved in discrete time but expected to hold also in continuous time)

Corollary [Sperl/Gr./Saluzzi/Kalise '23]: On suitable graphs this implies

$$\|P_{ij}\| \leq C\rho^{d_G(i,j)}$$

"Exponentially decaying sensitivity"

Construction of the separable approximations

Structure of the separable approximation

We approximate V by a separable function

$$V(x) \approx \sum_{k=1}^{s} \Psi_k^l(z_{i_{k,1}}, z_{i_{k,2}}, \ldots, z_{i_{k,l}})$$

Structure of the separable approximation

We approximate V by a separable function

$$V(x) \approx \sum_{k=1}^{s} \Psi_k^l(\underbrace{z_{i_{k,1}}, z_{i_{k,2}}, \ldots, z_{i_{k,l}}}_{=w_k})$$

Structure of the separable approximation

We approximate V by a separable function

$$V(x) \approx \sum_{k=1}^{s} \Psi_k^l(\underbrace{z_{i_{k,1}}, z_{i_{k,2}}, \ldots, z_{i_{k,l}}}_{=w_k})$$

Then each w_k has dimension $d_k = n_{i_{k,1}} + n_{i_{k,2}} + \ldots + n_{i_{k,l}}$

Structure of the separable approximation

We approximate V by a separable function

$$V(x) \approx \sum_{k=1}^{s} \Psi_k^l(\underbrace{z_{i_{k,1}}, z_{i_{k,2}}, \ldots, z_{i_{k,l}}}_{=w_k})$$

Then each w_k has dimension $d_k = n_{i_{k,1}} + n_{i_{k,2}} + \ldots + n_{i_{k,l}}$

- $\rightsquigarrow$ complexity theorem applies if n_j and l bounded independent of d
- $\rightsquigarrow$ approximation by polynomially growing NNs possible

Structure of the separable approximation

We approximate V by a separable function

$$V(x) \approx \sum_{k=1}^{s} \Psi_k^l(\underbrace{z_{i_{k,1}}, z_{i_{k,2}}, \ldots, z_{i_{k,l}}}_{=w_k})$$

Then each w_k has dimension $d_k = n_{i_{k,1}} + n_{i_{k,2}} + \ldots + n_{i_{k,l}}$

- $\rightsquigarrow$ complexity theorem applies if n_j and l bounded independent of d
- $\rightsquigarrow$ approximation by polynomially growing NNs possible

Note: It is in general unrealistic to expect

$$V(x) = \sum_{k=1}^{s} \Psi_k^l(z_{i_{k,1}}, z_{i_{k,2}}, \ldots, z_{i_{k,l}}) \qquad \text{or} \qquad V(x) \approx \sum_{k=1}^{s} \Psi_k^1(z_{i_{k,1}})$$

Separable approximations for $V(x) = x^T P x$

Assume for a moment that $d_G(i,j) = |i-j|$ (as in the convoy example)

Separable approximations for $V(x) = x^T P x$

Assume for a moment that $d_G(i,j) = |i-j|$ (as in the convoy example)

Then the decay of $\|P_{ij}\|$ implies

$$V(0,\ldots,0,z_k,z_{k+1},\ldots,z_s) - V(0,\ldots,0,0,z_{k+1},\ldots,z_s)$$

$$\approx V(0,\ldots,0,z_k,z_{k+1},\ldots,z_{k+l},0,\ldots,0) - V(0,\ldots,0,0,z_{k+1},\ldots,z_{k+l},0,\ldots,0)$$

Separable approximations for $V(x) = x^T P x$

Assume for a moment that $d_G(i,j) = |i-j|$ (as in the convoy example)

Then the decay of $\|P_{ij}\|$ implies

$$V(0,\dots,0,z_k,z_{k+1},\dots,z_s) - V(0,\dots,0,0,z_{k+1},\dots,z_s)$$

$$\approx \underbrace{V(0,\dots,0,z_k,z_{k+1},\dots,z_{k+l},0,\dots,0) - V(0,\dots,0,0,z_{k+1},\dots,z_{k+l},0,\dots,0)}_{=: \Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}})}$$

Separable approximations for $V(x) = x^T P x$

Assume for a moment that $d_G(i,j) = |i-j|$ (as in the convoy example)

Then the decay of $\|P_{ij}\|$ implies

$$V(0,\dots,0,z_k,z_{k+1},\dots,z_s) - V(0,\dots,0,0,z_{k+1},\dots,z_s)$$

$$\approx \underbrace{V(0,\dots,0,z_k,z_{k+1},\dots,z_{k+l},0,\dots,0) - V(0,\dots,0,0,z_{k+1},\dots,z_{k+l},0,\dots,0)}_{=:\,\Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}})}$$

This implies

$$V(x) \approx V(0) + \sum_{k=1}^{s} \Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}}),$$

Separable approximations for $V(x) = x^T P x$

Assume for a moment that $d_G(i,j) = |i-j|$ (as in the convoy example)

Then the decay of $\|P_{ij}\|$ implies

$$V(0,\dots,0,z_k,z_{k+1},\dots,z_s) - V(0,\dots,0,0,z_{k+1},\dots,z_s)$$

$$\approx \underbrace{V(0,\dots,0,z_k,z_{k+1},\dots,z_{k+l},0,\dots,0) - V(0,\dots,0,0,z_{k+1},\dots,z_{k+l},0,\dots,0)}_{=:\,\Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}})}$$

This implies

$$V(x) \approx V(0) + \sum_{k=1}^{s} \Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}}),$$

Theorem [Sperl/Saluzzi/Kalise/Gr. '25]: When the sensitivity decays exponentially

$$\left| V(x) - V(0) - \sum_{k=1}^{s} \Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}}) \right| \le c\rho^l \quad \text{on } L_2 \text{ balls}$$

Separable approximations for $V(x) = x^T P x$

Assume for a moment that $d_G(i,j) = |i-j|$ (as in the convoy example)
Then the decay of $\|P_{ij}\|$ implies

$$V(0,\dots,0,z_k,z_{k+1},\dots,z_s) - V(0,\dots,0,0,z_{k+1},\dots,z_s)$$

$$\approx \underbrace{V(0,\dots,0,z_k,z_{k+1},\dots,z_{k+l},0,\dots,0) - V(0,\dots,0,0,z_{k+1},\dots,z_{k+l},0,\dots,0)}_{=:\ \Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}})}$$

This implies

$$V(x) \approx V(0) + \sum_{k=1}^{s} \Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}}),$$

Theorem [Sperl/Saluzzi/Kalise/Gr. '25]: When the sensitivity decays exponentially

$$\left| V(x) - V(0) - \sum_{k=1}^{s} \Psi_k^l(z_k,\dots,z_{\min\{k+l,s\}}) \right| \le c\rho^l \quad \text{on } L_2 \text{ balls}$$

If $d_G(z_i,z_j) \neq |i-j|$, then for any node the number of nodes with graph distance l must grow slower than ρ^{-l}

Separable approximations for nonlinear V

For nonlinear problems, the optimal value function is not of the form $V(x) = x^T P x$

Separable approximations for nonlinear V

For nonlinear problems, the optimal value function is not of the form $V(x) = x^T P x$

In this case, decaying sensitivity can be expressed via the Lipschitz constant L_{ij} of

$$z_j \mapsto V(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_s) - V(z_1, \ldots, z_{i-1}, 0, z_{i+1}, \ldots, z_s) \qquad (*)$$

Separable approximations for nonlinear V

For nonlinear problems, the optimal value function is not of the form $V(x) = x^T P x$

In this case, decaying sensitivity can be expressed via the Lipschitz constant L_{ij} of

$$z_j \mapsto V(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_s) - V(z_1, \ldots, z_{i-1}, 0, z_{i+1}, \ldots, z_s) \qquad (*)$$

Theorem [Sperl/Saluzzi/Kalise/Gr. '25]: If $L_{ij} \leq C\rho^{d_G(i,j)}\|z_i\|$ then

$$\left| V(x) - V(0) - \sum_{k=1}^{q} \Psi_k^l(z_k, \ldots, z_{\min\{k+l,s\}}) \right| \leq c\rho^l \quad \text{on } L_2 \text{ balls}$$

Separable approximations for nonlinear V

For nonlinear problems, the optimal value function is not of the form $V(x) = x^T P x$

In this case, decaying sensitivity can be expressed via the Lipschitz constant L_{ij} of

$$z_j \mapsto V(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_s) - V(z_1, \ldots, z_{i-1}, 0, z_{i+1}, \ldots, z_s) \qquad (*)$$

Theorem [Sperl/Saluzzi/Kalise/Gr. '25]: If $L_{ij} \leq C\rho^{d_G(i,j)}\|z_i\|$ then

$$\left| V(x) - V(0) - \sum_{k=1}^{q} \Psi_k^l(z_k, \ldots, z_{\min\{k+l,s\}}) \right| \leq c\rho^l \quad \text{on } L_2 \text{ balls}$$

Moreover, estimates for slower-than-exponentially decaying sensitivity are possible

Separable approximations for nonlinear V

For nonlinear problems, the optimal value function is not of the form $V(x) = x^T P x$

In this case, decaying sensitivity can be expressed via the Lipschitz constant L_{ij} of

$$z_j \mapsto V(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_s) - V(z_1, \ldots, z_{i-1}, 0, z_{i+1}, \ldots, z_s) \qquad (*)$$

Theorem [Sperl/Saluzzi/Kalise/Gr. '25]: If $L_{ij} \leq C\rho^{d_G(i,j)}\|z_i\|$ then

$$\left| V(x) - V(0) - \sum_{k=1}^{q} \Psi_k^l(z_k, \ldots, z_{\min\{k+l,s\}}) \right| \leq c\rho^l \quad \text{on } L_2 \text{ balls}$$

Moreover, estimates for slower-than-exponentially decaying sensitivity are possible

The important open question is: When do the Lipschitz constants L_{ij} of $(*)$ satisfy this inequality?

Separable approximations for nonlinear V

For nonlinear problems, the optimal value function is not of the form $V(x) = x^T P x$

In this case, decaying sensitivity can be expressed via the Lipschitz constant L_{ij} of

$$z_j \mapsto V(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_s) - V(z_1, \ldots, z_{i-1}, 0, z_{i+1}, \ldots, z_s) \qquad (*)$$

Theorem [Sperl/Saluzzi/Kalise/Gr. '25]: If $L_{ij} \leq C\rho^{d_G(i,j)}\|z_i\|$ then

$$\left| V(x) - V(0) - \sum_{k=1}^{q} \Psi_k^l(z_k, \ldots, z_{\min\{k+l,s\}}) \right| \leq c\rho^l \quad \text{on } L_2 \text{ balls}$$

Moreover, estimates for slower-than-exponentially decaying sensitivity are possible

The important open question is: When do the Lipschitz constants L_{ij} of $(*)$ satisfy this inequality? This is subject of current research

Numerical test

We test the approach on the linear-quadratic problem with

$$\dot{x} = Ax + u, \qquad \ell(x,u) = \|x\|_2^2 + \|u\|_2^2,$$

and $A \in \mathbb{R}^{200\times 200}$ a randomly generated banded matrix

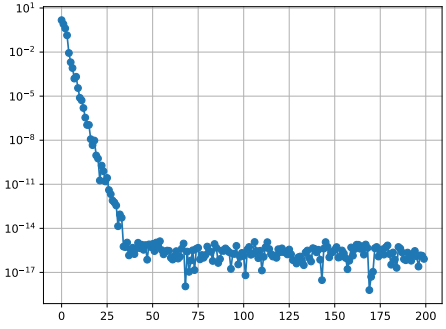
Numerical test

We test the approach on the linear-quadratic problem with

$$\dot{x} = Ax + u, \qquad \ell(x,u) = \|x\|_2^2 + \|u\|_2^2,$$

and $A \in \mathbb{R}^{200\times 200}$ a randomly generated banded matrix

This induces exponentially decaying sensitivity of P



$\|P_{j1}\|$, $j = 1, \ldots, 200$

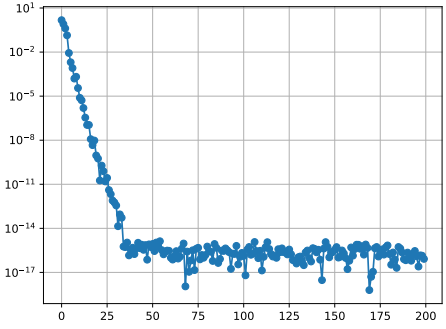
Numerical test

We test the approach on the linear-quadratic problem with

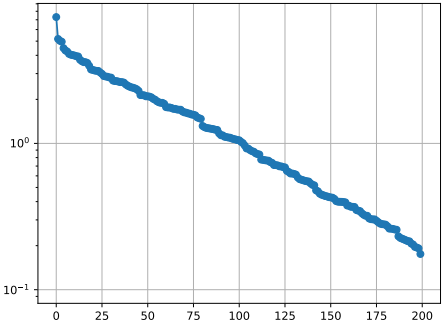
$$\dot{x} = Ax + u, \qquad \ell(x,u) = \|x\|_2^2 + \|u\|_2^2,$$

and $A \in \mathbb{R}^{200\times 200}$ a randomly generated banded matrix

This induces exponentially decaying sensitivity of P



$\|P_{j1}\|$, $j = 1, \ldots, 200$

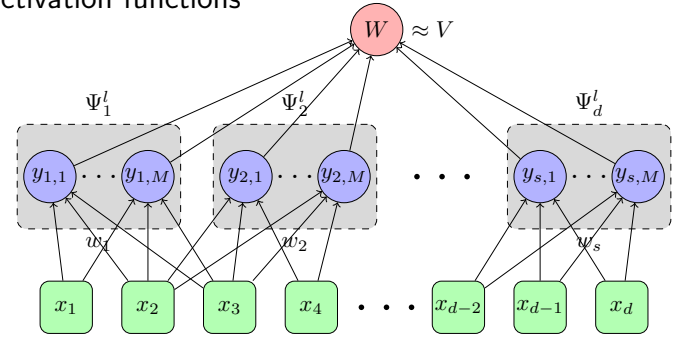


singular values of P

Numerical test

Separable network structure with

- $d_j = 1 \quad \Rightarrow \quad z_k = x_k$
- $l = 10$ (number of inputs for each Ψ_k^l)
- $M = 16$ (number of neurons for each Ψ_k^l)
- sigmoid activation functions

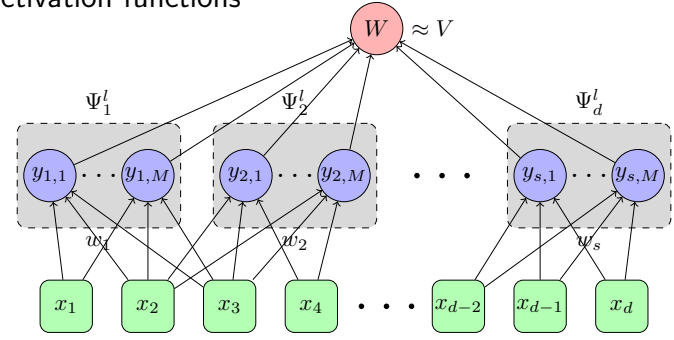


Numerical test

Separable network structure with

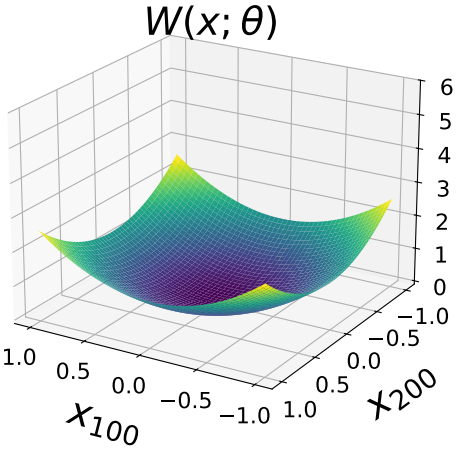
- $d_j = 1 \quad \Rightarrow \quad z_k = x_k$
- $l = 10$ (number of inputs for each Ψ_k^l)
- $M = 16$ (number of neurons for each Ψ_k^l)
- sigmoid activation functions

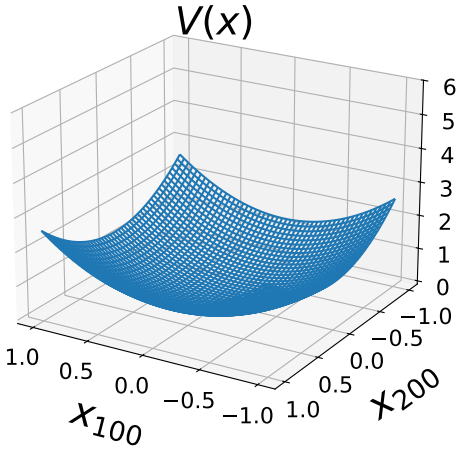
In order to avoid effects due to incomplete learning in reinforcement learning (exploration vs. exploitation), we use supervised learning to learn V



Numerical results

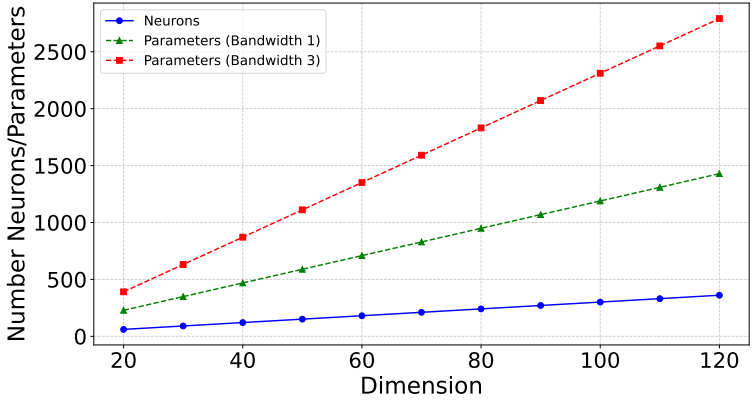
For the 200-dim problem we reach a mean-square error $< 10^{-3}$ in the test data





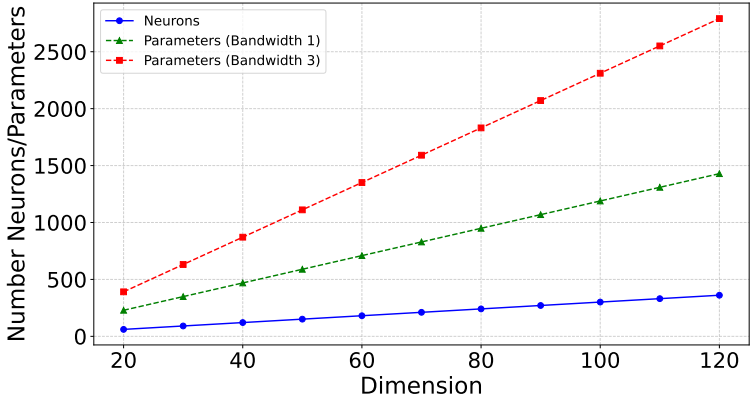
Numerical results

We have numerically evaluated the minimal number of neurons and parameters in the network for achieving a mean-square error in the test data of 10^{-3} for varying dimensions



Numerical results

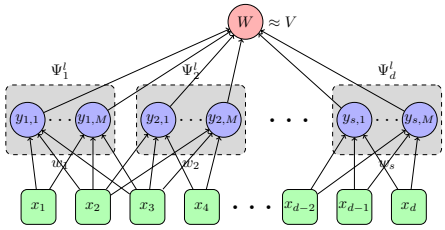
We have numerically evaluated the minimal number of neurons and parameters in the network for achieving a mean-square error in the test data of 10^{-3} for varying dimensions



Interestingly, the growth is linear (and not polynomial with degree ≥ 2)

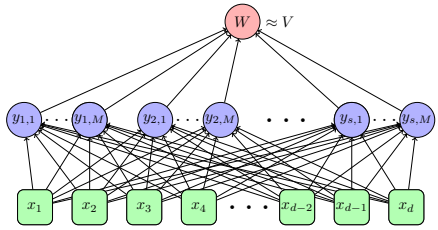
Separable vs. fully connected networks

Obviously, one can always embed a separable network into a fully connected network



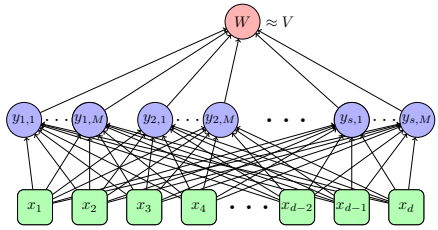
Separable vs. fully connected networks

Obviously, one can always embed a separable network into a fully connected network



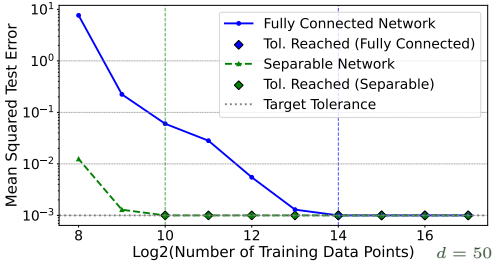
Separable vs. fully connected networks

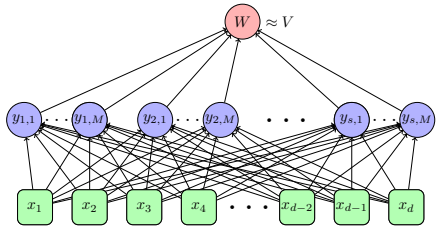
Obviously, one can always embed a separable network into a fully connected network. Is there any benefit in using the separable structure?



Separable vs. fully connected networks

Obviously, one can always embed a separable network into a fully connected network. Is there any benefit in using the separable structure?

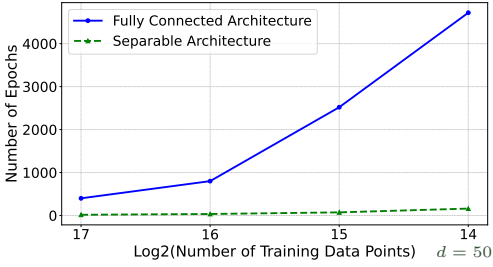


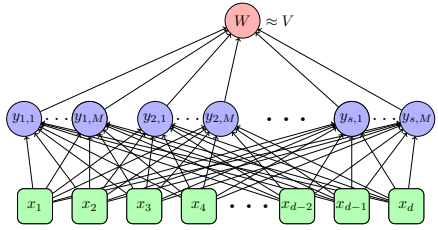


Yes: the amount of data and the number of training epochs needed for training is significantly smaller

Separable vs. fully connected networks

Obviously, one can always embed a separable network into a fully connected network. Is there any benefit in using the separable structure?

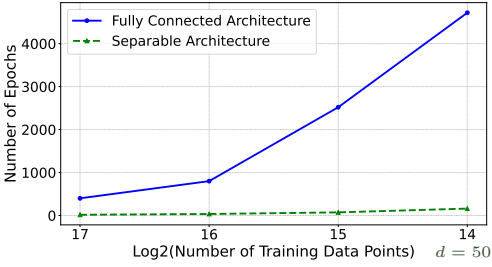




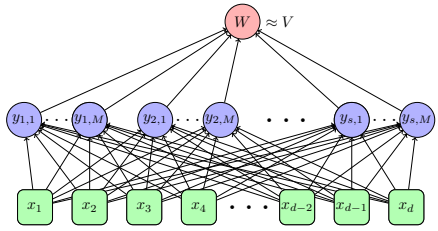
Yes: the amount of data and the number of training epochs needed for training is significantly smaller

Separable vs. fully connected networks

Obviously, one can always embed a separable network into a fully connected network. Is there any benefit in using the separable structure?



Yes: the amount of data and the number of training epochs needed for training is significantly smaller



We conjecture that this can be explained using anchored decompositions [Sobol '69, Kuo/Sloan/Wasilikowski/Woźniakowski '09, Rieger/Wendland '24]

Summary of Part 5

- Using Deep Neural Networks for approximating optimal value function does in general not remove the curse of dimensionality

Summary of Part 5

- Using Deep Neural Networks for approximating optimal value function does in general not remove the curse of dimensionality
- However, the existence of separable approximations of optimal value functions allows for a curse-of-dimensionality-free approximation via deep neural networks

Summary of Part 5

- Using Deep Neural Networks for approximating optimal value function does in general not remove the curse of dimensionality
- However, the existence of separable approximations of optimal value functions allows for a curse-of-dimensionality-free approximation via deep neural networks
- This existence can be established via decaying sensitivity

Summary of Part 5

- Using Deep Neural Networks for approximating optimal value function does in general not remove the curse of dimensionality
- However, the existence of separable approximations of optimal value functions allows for a curse-of-dimensionality-free approximation via deep neural networks
- This existence can be established via decaying sensitivity
- Open questions:
  - Exponential sensitivity for nonlinear problems
  - Analysis of sampling efficiency

Literature, Part 1–3:

J.A.E. Andersson, J. Gillis, G. Horn, J.B. Rawlings, M. Diehl. "CasADi: a software framework for nonlinear optimization and optimal control". In: Mathematical Programming Computation 11 (2019), pp. 1–36

M. Bardi and I. Capuzzo Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, Birkhäuser, 1997

A. Barto and R. S. Sutton, *Reinforcement Learning: An Introduction*, MIT Press, 2nd ed., 2018

D. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, Boston, MA, 4th edition, 2017 (Vol. I) and 2012 (Vol. II)

D. Bertsekas, *A course in Reinforcement Learning*, Athena Scientific, Boston, MA, 2023, https://web.mit.edu/dimitrib/www/RLCOURSECOMPLETE.pdf

H.G. Bock and K.-J. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: Proc. 9th IFAC World Congress. 1984

V.G. Boltyanskii, R.V. Gamkrelidze, L.S. Pontryagin. "On the theory of optimal processes". In: Doklady Akademii Nauk SSSR 110 (1956), pp. 7–10

B. Chachuat. *Nonlinear and Dynamic Optimization—From Theory to Practice*, EPFL, 2009. https://infoscience.epfl.ch/server/api/core/bitstreams/28c307a4-ced8-4df6-bf43-fd02005ed74d/content

Literature, Part 1–3 (continued):

T. Englert, A. Völz, F. Mesmer, S. Rhein, K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: Optimization and Engineering 20.3 (2019), pp. 769–809

M. Falcone and R. Ferretti, *Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations*, SIAM, 2013

T. Faulwasser and C.M. Kellett, "On continuous-time infinite horizon optimal control—Dissipativity, stability, and transversality". In: Automatica 134 (2021), 109907

A.T. Fuller. "Relay control systems optimized for various performance criteria". In: Automation and remote control, Proc. first world congress IFAC Moscow. Vol. 1. 1960, pp. 510–519

L. Grüne, *Artificial Intelligence Methods in Control*, Lecture Notes, University of Bayreuth, 2024 (2nd edition), https://num.math.uni-bayreuth.de/de/team/lars-gruene/skripten/

L. Grüne and W. Semmler, "Using dynamic programming with adaptive grid scheme for optimal control problems in economics". In: Journal of Economic Dynamics and Control, 28 (2004), pp. 2427–2456

H. Halkin, "Necessary conditions for optimal control problems with infinite horizons". In: Econometrica: Journal of the Econometric Society 42.2 (1974), pp. 267–272

Literature, Part 1–3 (continued):

R.F. Hartl, S.P. Sethi, R.G. Vickson. "A survey of the maximum principles for optimal control problems with state constraints". In: SIAM Review 37.2 (1995), pp. 181–218

R. Kamalapurkar, P. Walters, J. Rosenfeld, W. Dixon, *Reinforcement Learning for Optimal Feedback Control*, Springer, 2018

D. Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*, Princeton University Press, 2012.
http://liberzon.csl.illinois.edu/teaching/cvoc.pdf

E.R. Pinch. *Optimal Control and the Calculus of Variations*, Oxford University Press, 1995

L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, E.F. Mishchenko, *The Mathematical Theory of Optimal Processes*, John Wiley & Sons Inc., New York, 1962

H.J. Sussmann and J.C. Willems, "300 years of optimal control: from the Brachystochrone to the Maximum Principle". In: IEEE Control Systems 17.3 (1997), pp. 32–44

Literature, Part 4:

T. Faulwasser. L. Grüne, "Turnpike Properties in Optimal Control: An Overview of Discrete-Time and Continuous-Time Results". In: Emmanuel Trélat, Enrique Zuazua (eds), Handbook of Numerical Analysis: Numerical Control. Part A, North-Holland, 367–400, 2022

T. Faulwasser, L. Grüne, M.A. Müller, "Economic Nonlinear Model Predictive Control". In: Foundations and Trends® in Systems and Control, 5, 1–98, 2018

L. Grüne, "Approximation properties of receding horizon optimal control". In: Jahresbericht der Deutschen Mathematiker Vereinigung, 118, 3–37, 2016

L. Grüne, J. Pannek, *Nonlinear Model Predictive Control*, Springer, 2nd ed. 2017

L. Grüne, "Dynamic programming, optimal control and model predictive control". In: Saša V. Raković, William S. Levine (eds), *Handbook of Model Predictive Control*, Birkhäuser, 29–52, 2018

L. Grüne, "Dissipativity and optimal control: Examining the Turnpike Phenomenon". In: IEEE Control Systems Magazine, 42, 74–87, 2022

L. Grüne, M. Höger, K. Link, "Cost design for predictive controllers: theoretical considerations and application to the start-up of a combined cycle power plant". Preprint, 2025, https://dx.doi.org/doi:10.15495/EPub_UBT_00008521

Literature, Part 5:

L. Grüne, "Computing Lyapunov functions using deep neural networks". In: Journal of Computational Dynamics 8 (2021), 131–152

L. Grüne and M. Sperl, "Examples for existence and non-existence of separable control Lyapunov functions". In: Proc. NOLCOS 2022, IFAC-PapersOnLine 56 (2023), 19–24

C. Rieger and H. Wendland, "On the approximability and curse of dimensionality of certain classes of high-dimensional functions". In: SIAM Journal on Numerical Analysis 62 (2024), 842–871

S. Shin, Y. Lin, G. Qu, A. Wierman, and M. Anitescu, "Near-Optimal Distributed Linear-Quadratic Regulator for Networked Systems". In: SIAM Journal on Control and Optimization 61 (2023), 1113–1135

M. Sperl, J. Mysliwitz and L. Grüne, "On the existence and neural network representation of separable control Lyapunov functions". In: Automatica, 182 (2025), No. 112517

M. Sperl, L. Saluzzi, L. Grüne, and D. Kalise, "Separable approximations of optimal value functions under a decaying sensitivity assumption". In: Proc. CDC 2023, 2790–2795

M. Sperl, L. Saluzzi, D. Kalise, and L. Grüne, "Separable approximations of optimal value functions and their representation by neural networks". Preprint, 2025, https://dx.doi.org/doi:10.48550/arXiv.2502.08559