Data-driven and Learning-based Control

Optimal Control

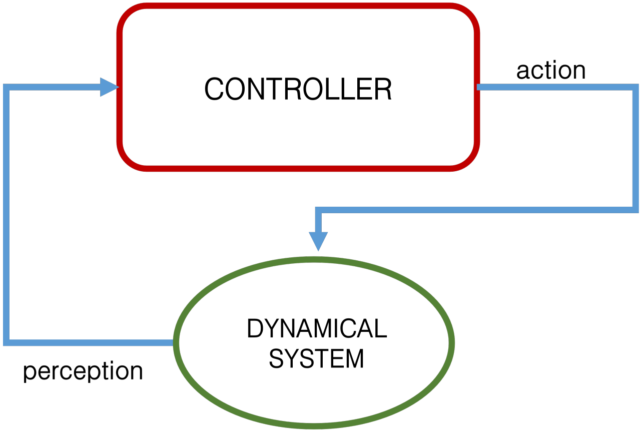
Erica Salvato

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Table of Contents

Schematic of an AS

1 A brief recap



Dynamical Systems

- Definition
- Overview of the different kinds of dynamical systems:
  - continuous time (CT)
  - discrete-time (DT)
  - from CT to DT
  - linear
  - non-linear
- Equilibrium state
- Linearization around an equilibrium

Controller

1 A brief recap

- Definition
- Control problem
- Overview of the different kinds of controllers:
  - open-loop
  - closed-loop
- PID controller
- Introduction to the different types of controller synthesis procedures:
  - model-based
  - data-driven
  - learning-based

Table of Contents

2 Optimal Control

Optimal control problem

Consider the DT dynamical system

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right)$$

where

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional control input
- $k \in \mathbb{Z}_0^+$ the time-step index
- $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the state transition function

Optimal control problem

Consider the DT dynamical system

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right)$$

where

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional control input
- $k \in \mathbb{Z}_0^+$ the time-step index
- $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the state transition function

Suppose that at $k = 0$ the system initial state is $x^{(0)} = x_{\text{ini}} \in \mathcal{X}$.

Optimal control problem

Consider the DT dynamical system

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right)$$

where

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional control input
- $k \in \mathbb{Z}_0^+$ the time-step index
- $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the state transition function

Suppose that at $k = 0$ the system initial state is $x^{(0)} = x_{\text{ini}} \in \mathcal{X}$.

We would like to design a state feedback controller π able to return $x^{(k)} = x_{\text{des}} \in \mathcal{X}$ when $k = H$.

Optimal control problem

Consider the DT dynamical system

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right)$$

where

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional control input
- $k \in \mathbb{Z}_0^+$ the time-step index
- $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$ the state transition function

Suppose that at $k = 0$ the system initial state is $x^{(0)} = x_{\text{ini}} \in \mathcal{X}$.

We would like to design a state feedback controller π able to return $x^{(k)} = x_{\text{des}} \in \mathcal{X}$ when $k = H$.

What if we also want the control strategy to be **the optimal one**?

Optimal control problem

Consider the DT dynamical system

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right)$$

where

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional control input
- $k \in \mathbb{Z}_0^+$ the time-step index
- $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the state transition function

Suppose that at $k = 0$ the system initial state is $x^{(0)} = x_{\text{ini}} \in \mathcal{X}$.

We would like to design a state feedback controller π able to return $x^{(k)} = x_{\text{des}} \in \mathcal{X}$ when $k = H$.

What if we also want the control strategy to be **the optimal one**?

What does optimal control strategy mean? And optimal with respect to what?

Optimal control problem

- First of all we need to define a task

Task

A task is an objective to be achieved by a controller operating on a system

Optimal control problem

- First of all we need to define a task

Task

A task is an objective to be achieved by a controller operating on a system

- Given a task, it is also necessary to define a **performance index** capable of capturing the effectiveness of the control input, returned by the controller, in achieving the task

Optimal control problem

- First of all we need to define a task

Task

A task is an objective to be achieved by a controller operating on a system

- Given a task, it is also necessary to define a **performance index** capable of capturing the effectiveness of the control input, returned by the controller, in achieving the task

Optimal control problem

- First of all we need to define a task

Task

A task is an objective to be achieved by a controller operating on a system

- Given a task, it is also necessary to define a **performance index** capable of capturing the effectiveness of the control input, returned by the controller, in achieving the task
- Furthermore, if we need to satisfy certain physical **constraints** on the state or control input, we must formally define them
- Then the **optimal controller** $\pi^*\left(x^{(k)}\right)$ is the one which is able to produce $\forall k = \{0,\ 1,\ \ldots,\ H-1\}$ the optimal control inputs $u^{*(0)},\ u^{*(1)},\ \ldots,\ u^{*(H-1)}$ such that the performance index is maximized and the task is performed

Formal definition of an optimal control problem

Define:

- $x^{(k)} \in \mathcal{X}$ the state of a dynamical system at the k-th time instant
- $\pi\left(x^{(k)}\right)$ the state-feedback controller such that $u^{(k)} = \pi\left(x^{(k)}\right) \in \mathcal{U}\ \forall k$ is the control input
- $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{X}$ the state transition function s.t.:

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right)$$

Formal definition of an optimal control problem

Define also:

- $x^{(0)} = x_{\text{ini}} \in \mathcal{X}$ the *initial state*
- $x_{\text{des}} \in \mathcal{X}$ the *desired state*
- $h : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ an **objective function** s.t.:

$$J = \sum_{k=0}^{H-1} h\left(x^{(k)}, u^{(k)}\right)$$

  is a **performance index** that evaluates the state-action pair in a time horizon of H steps

Formal definition of an optimal control problem

Compute the optimal controller $\pi^* \left(x^{(k)}\right)$ by solving the following optimal control problem:

$$\pi^* = \arg\max_{\pi} J \qquad \text{if } h = \text{Objective reward function}$$

$$\pi^* = \arg\min_{\pi} J \qquad \text{if } h = \text{Objective cost function}$$

s.t.:

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right)$$

$$x^{(0)} = x_{\text{ini}}$$

$$x^{(H)} = x_{\text{des}}$$

Performance index

The general definition of J for optimal control problems is the following:

$$J = x^{(H)^\top} Q_H x^{(H)} + \sum_{k=0}^{H-1} x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

What does it mean?

Performance index

The general definition of J for optimal control problems is the following:

$$J = x^{(H)^\top} Q_H x^{(H)} + \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{H}-\mathbf{1}} \mathbf{x}^{(\mathbf{k})^\top} \mathbf{Q} \mathbf{x}^{(\mathbf{k})} + \mathbf{u}^{(\mathbf{k})^\top} \mathbf{R} \mathbf{u}^{(\mathbf{k})}$$

What does it mean?

- We want to steer the state of the system toward zero ($x_{\text{des}} = 0$) by applying the minimum possible control

Performance index

The general definition of J for optimal control problems is the following:

$$J = \mathbf{x^{(H)}}^\top \mathbf{Q_H} \mathbf{x^{(H)}} + \sum_{k=0}^{H-1} x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

What does it mean?

- We want to steer the state of the system toward zero ($x_{\text{des}} = 0$) by applying the minimum possible control
- We want at $k = H$ the state of the system to be as close to 0 as possible

Performance index

The general definition of J for optimal control problems is the following:

$$J = x^{(H)^\top} Q_H x^{(H)} + \sum_{k=0}^{H-1} x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

What does it mean?

- We want to steer the state of the system toward zero ($x_{\text{des}} = 0$) by applying the minimum possible control
- We want at $k = H$ the state of the system to be as close to 0 as possible

Here the **task** is to steer the state of the system toward zero in H steps by also applying the minimum control effort

Performance index

The general definition of J for optimal control problems is the following:

$$J = {x^{(H)}}^\top Q_H x^{(H)} + \sum_{k=0}^{H-1} {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)}$$

What does it mean?

- We want to steer the state of the system toward zero ($x_{\text{des}} = 0$) by applying the minimum possible control
- We want at $k = H$ the state of the system to be as close to 0 as possible

Here the **task** is to steer the state of the system toward zero in H steps by also applying the minimum control effort

Depending on the task we want the dynamic system to perform, we can manage J appropriately

Examples of tasks and performance indices

- **Minimum time control problem**
  Consider a task in which we want to drive the system from an initial state x_{ini} to a desired state x_{des} in the minimum number of steps

Examples of tasks and performance indices

- **Minimum time control problem**
  Consider a task in which we want to drive the system from an initial state x_{ini} to a desired state x_{des} in the minimum number of steps

  Find:

$$\pi^* = \arg\min_{\pi} H$$

  s.t.:

$$x^{(k+1)} = f\left(x^{(k)}, \pi\left(x^{(k)}\right)\right)$$

$$x^{(0)} = x_{\text{ini}}$$

$$x^{(H)} = x_{\text{des}}$$

Examples of tasks and performance indices

- **Minimum energy control problem**
  Consider a task in which we want to drive the system from an initial state x_{ini} to a desired state x_{des} minimizing the energy consumption

Examples of tasks and performance indices

- **Minimum energy control problem**
  Consider a task in which we want to drive the system from an initial state x_{ini} to a desired state x_{des} minimizing the energy consumption

  Find:

$$\pi^* = \arg\min_{\pi} \sum_{k=0}^{H-1} u^{(k)^\top} R u^{(k)}$$

  s.t.:

$$x^{(k+1)} = f\left(x^{(k)}, \pi\left(x^{(k)}\right)\right)$$

$$x^{(0)} = x_{\text{ini}}$$

$$x^{(H)} = x_{\text{des}}$$

Examples of tasks and performance indices

- **Tracking optimal control**
  Consider a task in which we want to drive the system from an initial state x_{ini} to a desired state x_{des} following a nominal trajectory $\tilde{x}^{(0)}, \tilde{x}^{(1)}, \ldots, \tilde{x}^{(H)}$ where $\tilde{x}^{(0)} = x_{\text{ini}}$ and $\tilde{x}^{(H)} = x_{\text{des}}$

Examples of tasks and performance indices

- **Tracking optimal control**
  Consider a task in which we want to drive the system from an initial state x_{ini} to a desired state x_{des} following a nominal trajectory $\tilde{x}^{(0)}, \tilde{x}^{(1)}, \ldots, \tilde{x}^{(H)}$ where $\tilde{x}^{(0)} = x_{\text{ini}}$ and $\tilde{x}^{(H)} = x_{\text{des}}$

  Find:

$$\pi^* = \arg\min_{\pi} \sum_{k=0}^{H-1} \left(x^{(k)} - \tilde{x}^{(k)}\right)^\top Q \left(x^{(k)} - \tilde{x}^{(k)}\right)$$

  s.t.:

$$x^{(k+1)} = f\left(x^{(k)}, \pi\left(x^{(k)}\right)\right)$$

$$x^{(0)} = x_{\text{ini}}$$

$$x^{(H)} = x_{\text{des}}$$

Two fundamental questions

Before we start introducing possible ideas useful in solving optimal control problems, we should pose two fundamental questions.

1. Is it always possible to **access the state** measure?

2. Is it always possible to **reach the desired state** in the horizon H?

Two fundamental questions

Before we start introducing possible ideas useful in solving optimal control problems, we should pose two fundamental questions.

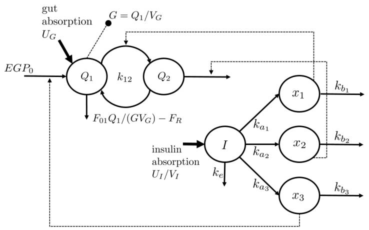
1. Is it always possible to **access the state** measure?

   Roughly speaking, this means that there are sensors, equal in number to the number of state components in the system (n), able to provide real-time measurements of each state component.

Can we always access the state measure?

Consider a glucoregulatory system:

$$
\begin{aligned}
\dot{Q}_1(t) &= -F_{01} - g_1 Q_1 + k_{12} Q_2 - F_R + EGP_0(1 - g_3) + U_G(t) \\
\dot{Q}_2(t) &= g_1 Q_1 - (k_{12} + g_2) Q_2 \\
\dot{S}_1(t) &= u(t) + u_b - \frac{S_1}{t_{maxI}} \\
\dot{S}_2(t) &= \frac{S_1 - S_2}{t_{maxI}} \\
\dot{I}(t) &= \frac{S_2}{t_{maxI} V_I} - k_e I \\
\dot{g}_i(t) &= -k_{a_i} g_i + k_{b_i} I \quad (i = 1, 2, 3) \\
y(t) = G(t) &= \frac{Q_1(t)}{V_G}
\end{aligned}
$$

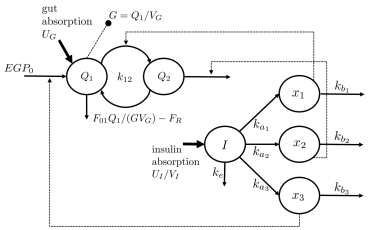


- The plasma insulin I is not feasible to measure in real-time
- There is a glucose compartment Q_2 which cannot be measured by sensors

Can we always access the state measure?

Consider a glucoregulatory system:

$$\dot{Q}_1(t) = -F_{01} - g_1 Q_1 + k_{12} Q_2 - F_R + EGP_0(1 - g_3) + U_G(t)$$
$$\dot{Q}_2(t) = g_1 Q_1 - (k_{12} + g_2) Q_2$$
$$\dot{S}_1(t) = u(t) + u_b - \frac{S_1}{t_{maxI}}$$
$$\dot{S}_2(t) = \frac{S_1 - S_2}{t_{maxI}}$$
$$\dot{I}(t) = \frac{S_2}{t_{maxI} V_I} - k_e I$$
$$\dot{g}_i(t) = -k_{a_i} g_i + k_{b_i} I \quad (i = 1, 2, 3)$$
$$y(t) = G(t) = \frac{Q_1(t)}{V_G}$$



Not fully accessible state

Can we always access the state measure?

However, in order to work with the previously defined optimal control problem, we need the state measures.

How can we manage systems with non-accessible state measures??

Can we always access the state measure?

However, in order to work with the previously defined optimal control problem, we need the state measures.

How can we manage systems with non-accessible state measures??

- Add sensors able to provide measures of the non-accessible parts
- Being able to estimate state measurements directly from output measurements

Can we always access the state measure?

However, in order to work with the previously defined optimal control problem, we need the state measures.

How can we manage systems with non-accessible state measures??

- Add sensors able to provide measures of the non-accessible parts

- Being able to estimate state measurements directly from output measurements

$$\downarrow$$

State observer

Two fundamental questions

Before we start introducing possible ideas useful in solving optimal control problems, we should pose two fundamental questions.

1. Is it always possible to **access the state** measure?
2. Is it always possible to **reach the desired state** in the horizon H?

Two fundamental questions

Before we start introducing possible ideas useful in solving optimal control problems, we should pose two fundamental questions.

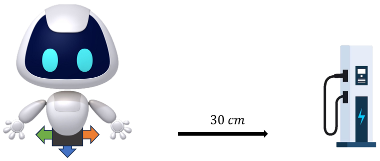
1. Is it always possible to **access the state** measure?

2. Is it always possible to **reach the desired state** in the horizon H?

   In simple terms, this means that it is possible to design a controller π such that the resulting sequence of control inputs $u^{(0)}, u^{(1)}, \ldots, u^{(H-1)}$ is able to move the state of the system from an initial state to a desired final state in H steps without exceeding the physical limits imposed by $\mathcal{U}$

Can we always reach the desired state?

Consider a problem where we want to control the robot to reach its charging station

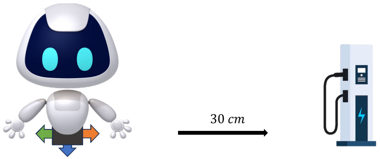


- The charging station is 30 cm away from the robot
- The battery level of the robot is such that the maximum speed it can reach is $v_{\max} = 10\,\mathrm{cm\,s^{-1}}$
- The maximum speed is chosen in order to have a battery autonomy of at most 5 s

Is it possible to design an optimal controller to reach the charging station in 2 s?

Can we always reach the desired state?

Consider a problem where we want to control the robot to reach its charging station



- The charging station is 30 cm away from the robot
- The battery level of the robot is such that the maximum speed it can reach is $v_{\text{max}} = 10\,\text{cm}\,\text{s}^{-1}$
- The maximum speed is chosen in order to have a battery autonomy of at most 5 s

Is it possible to design an optimal controller to reach the charging station in 2 s?

Even assuming the optimal control imposes a constant velocity of $10\,\text{cm}\,\text{s}^{-1}$ for two seconds, it can move at the most of 20 cm

The final state is not reachable for the robot

Can we always reach the desired state?

How can we handle these scenarios?

Can we always reach the desired state?

How can we handle these scenarios?

- Increasing the H horizon
- Manually adjusting physical constraints

Can we always reach the desired state?

How can we handle these scenarios?

- Increasing the H horizon
- Manually adjusting physical constraints $\rightarrow$ **not always possible**

The linear systems case

In the particular case of linear systems, there are two well-defined routines to check:

1. Whether it is possible to always reconstruct the state values from the output values when the system has non-accessible states

2. Whether we can reach all the states belonging to $\mathcal{X}$ in n (state-size) steps from the initial one

The linear systems case

In the particular case of linear systems, there are two well-defined routines to check:

1. Whether it is possible to always reconstruct the state values from the output values when the system has non-accessible states

$$\downarrow$$

OBSERVABILITY CHECK

2. Whether we can reach all the states belonging to $\mathcal{X}$ in n (state-size) steps from the initial one

$$\downarrow$$

REACHABILITY CHECK

The linear systems case: Observability check

Consider a linear discrete-time time-invariant dynamical system:

$$x^{(k+1)} = Ax^{(k)} + Bu^{(k)}, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m}$$
$$y^{(k)} = Cx^{(k)} + Du^{(k)}, \qquad C \in \mathbb{R}^{p\times n},\ D \in \mathbb{R}^{p\times m}.$$

where:

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional input
- $y^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^p$ the p-dimensional output
- $k \in \mathbb{Z}_0^+$ the time-step index

We know (basic course in automatic control) that the output movement is:

$$y^{(k)} = CA^k x^{(0)} + \sum_{j=0}^{k-1} \left(CA^{k-(j+1)} Bu^{(j)} \right) + Du^{(k)}$$

The linear systems case: Observability check

Consider a linear discrete-time time-invariant dynamical system:

$$\begin{aligned} x^{(k+1)} &= Ax^{(k)} + Bu^{(k)}, \qquad && A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m} \\ y^{(k)} &= Cx^{(k)} + Du^{(k)}, \qquad && C \in \mathbb{R}^{p\times n},\ D \in \mathbb{R}^{p\times m}. \end{aligned}$$

where:

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional input
- $y^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^p$ the p-dimensional output
- $k \in \mathbb{Z}_0^+$ the time-step index

We know (basic course in automatic control) that the output movement is:

$$y^{(k)} = CA^k x^{(0)} + \sum_{j=0}^{k-1} \left(CA^{k-(j+1)} Bu^{(j)} \right) + Du^{(k)}$$

zero-input response   zero-state response

The linear systems case: Observability check

Sufficient condition for non-full observability

A sufficient condition for a system to be **non-fully observable** is that at least one state variable does not influence the output (directly or indirectly)

Observing the zero-input response $CA^k x^{(0)}$ we need to check $\mathbf{A}$ and $\mathbf{C}$ matrices in order to estimate the state from the output.

Example:

$$\begin{cases} x^{(k+1)} &= \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} x^{(k)} \qquad \text{indirect} \\ \\ y^{(k)} &= \begin{bmatrix} C_{11} & 0 \end{bmatrix} x^{(k)} \qquad \text{direct} \end{cases}$$

The linear systems case: Observability check

Necessary and sufficient condition for full observability

If n is the state size of the dynamical system. Necessary and sufficient condition for a system to be **fully observable** is that

$$\operatorname{rank}\left\{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}\right\} = n$$

The linear systems case: Reachability check

Consider a linear discrete-time time-invariant dynamical system:

$$x^{(k+1)} = Ax^{(k)} + Bu^{(k)}, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m}$$
$$y^{(k)} = Cx^{(k)} + Du^{(k)}, \qquad C \in \mathbb{R}^{p\times n},\ D \in \mathbb{R}^{p\times m}.$$

where:

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional input
- $y^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^p$ the p-dimensional output
- $k \in \mathbb{Z}_0^+$ the time-step index

We know (basic course in automatic control) that the state movement is:

$$x^{(k)} = A^k x^{(0)} + \sum_{j=0}^{k-1} A^{k-(j+1)} B u^{(j)}$$

The linear systems case: Reachability check

Consider a linear discrete-time time-invariant dynamical system:

$$\begin{aligned} x^{(k+1)} &= Ax^{(k)} + Bu^{(k)}, \qquad && A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m} \\ y^{(k)} &= Cx^{(k)} + Du^{(k)}, \qquad && C \in \mathbb{R}^{p\times n},\ D \in \mathbb{R}^{p\times m}. \end{aligned}$$

where:

- $x^{(k)} \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional input
- $y^{(k)} \in \mathcal{U} \subseteq \mathbb{R}^p$ the p-dimensional output
- $k \in \mathbb{Z}_0^+$ the time-step index

We know (basic course in automatic control) that the state movement is:

$$x^{(k)} = A^k x^{(0)} + \sum_{j=0}^{k-1} A^{k-(j+1)} Bu^{(j)}$$

zero-input response   zero-state response

The linear systems case: Reachability check

Suppose that at $k = 0$ we are in the null state, the reachability of a desired state in H steps requires analyzing the zero-state response $\sum_{j=0}^{k-1} A^{k-(j+1)} B u^{(j)}$ which depends on the **A** and **B** matrices

Necessary and sufficient condition for full reachability

If n is the state size of the dynamical system. Necessary and sufficient condition for a system to be **fully reachable** is that

$$\text{rank}\left\{\begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}\right\} = n$$

This means that all the states $x^{(k)} \in \mathcal{X}$ are reachable in at most n steps.

The linear systems case: Reachability check

What if we wanted to verify that a state x_{des} is reachable in H steps starting from an initial condition $x^{(0)} = x_{\text{ini}}$?

1. Check that the H-steps reachability matrix has rank H

$$\text{rank}\left\{\begin{bmatrix} B & AB & \dots & A^{H-1}B \end{bmatrix}\right\} = H$$

   It ensures that there exists a subset of $\mathcal{X}$ containing states reachable in H steps

2. Solve the following:

$$x_{\text{des}} - A^H x_{\text{ini}} = \sum_{j=0}^{H-1} A^{H-(j+1)} B u^{(j)} = \begin{bmatrix} B & AB & \dots & A^{H-1}B \end{bmatrix} \begin{bmatrix} u^{(H-1)} \\ u^{(H-2)} \\ \vdots \\ u^{(0)} \end{bmatrix}$$

   When does it admit solutions?

The linear systems case: Reachability check

What if we wanted to verify that a state x_{des} is reachable in H steps starting from an initial condition $x^{(0)} = x_{\text{ini}}$?

1. Check that the H-steps reachability matrix has rank H

$$\text{rank}\left\{\begin{bmatrix} B & AB & \dots & A^{H-1}B \end{bmatrix}\right\} = H$$

   It ensures that there exists a subset of $\mathcal{X}$ containing states reachable in H steps

2. Solve the following:

$$x_{\text{des}} - A^H x_{\text{ini}} = \sum_{j=0}^{H-1} A^{H-(j+1)} B u^{(j)} = \begin{bmatrix} B & AB & \dots & A^{H-1}B \end{bmatrix} \begin{bmatrix} u^{(H-1)} \\ u^{(H-2)} \\ \vdots \\ u^{(0)} \end{bmatrix}$$

   When does it admit solutions? When $x_{\text{des}} - A^H x_{\text{ini}} \in \text{Im}\left(\begin{bmatrix} B & AB & \dots & A^{H-1}B \end{bmatrix}\right)$

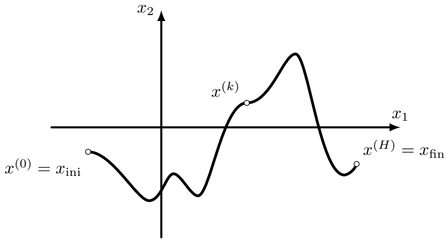
Table of Contents

3 Bellman’s principle

Bellman’s principle of optimality

Consider any of the previously stated optimal control problems.

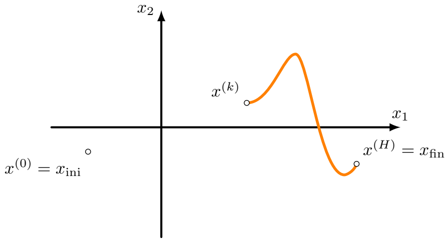
Suppose that the optimal solution leads to the following optimal trajectory



Bellman's principle of optimality

Consider any of the previously stated optimal control problems.

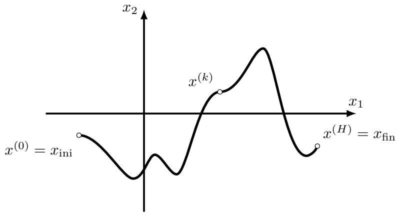
Suppose that the optimal solution leads to the following optimal trajectory



If we consider repeating the task but from an initial state $x^{(k)}$ belonging to the optimal trajectory. How will the sub-trajectory starting from $x^{(k)}$ be considered?

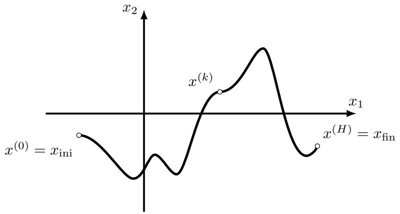
Bellman’s principle of optimality

If some trajectory connects the initial $x^{(0)} = x_{\text{ini}}$ and terminal $x^{(H)} = x_{\text{fin}}$ points and is optimal in the sense of some cost functional, then the sub-trajectory, connecting any intermediate point $x^{(k)},\ k > 0$ of the same trajectory with the same terminal point $x^{(H)}$, should also be optimal.



Bellman's principle of optimality

If some trajectory connects the initial $x^{(0)} = x_{\text{ini}}$ and terminal $x^{(H)} = x_{\text{fin}}$ points and is optimal in the sense of some cost functional, then the sub-trajectory, connecting any intermediate point $x^{(k)}$, $k > 0$ of the same trajectory with the same terminal point $x^{(H)}$, should also be optimal.



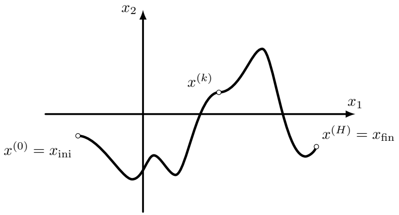
Bellman's principle

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.

Applying Bellman’s principle

Consider again any of the previously stated optimal control problems.

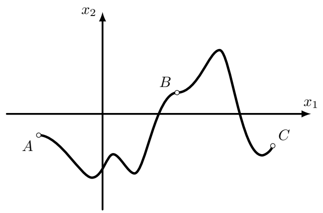
Suppose again that the optimal solution leads to the following optimal trajectory where $0 < k < H$



Applying Bellman's principle

Consider again any of the previously stated optimal control problems.

Suppose again that the optimal solution leads to the following optimal trajectory where $0 < k < H$



Assume that this optimal trajectory is the solution of a multistage problem with additive performance index structure:

$$A \to B \implies J_{AB}$$

$$B \to C \implies J_{BC}$$

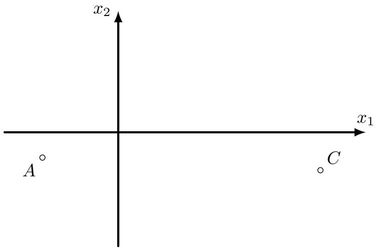
The optimal value of the performance index is:

$$J^*_{AC} = J_{AB} + J_{BC}$$

Applying Bellman’s principle

What if we don’t know the optimal trajectory?

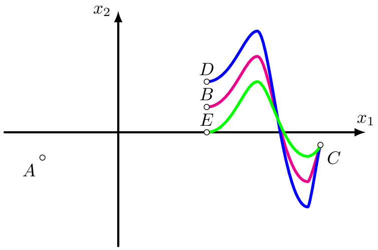
We want to find the optimal trajectory from A to C



Applying Bellman’s principle

What if we don’t know the optimal trajectory?

We want to find the optimal trajectory from A to C



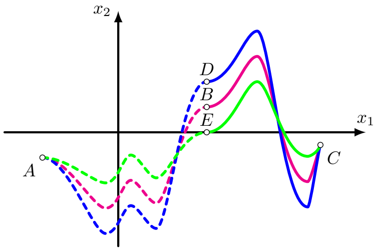
Assume that we know the optimal trajectories:

$E \to C \implies J^*_{EC}$ $\qquad D \to C \implies J^*_{DC}$ $\qquad B \to C \implies J^*_{BC}$

Applying Bellman's principle

What if we don't know the optimal trajectory?

We want to find the optimal trajectory from A to C



Assume that we know the optimal trajectories:

$E \to C \implies J^*_{EC}$ $\qquad D \to C \implies J^*_{DC}$ $\qquad B \to C \implies J^*_{BC}$

Following Bellman's principle, the optimal trajectory is obtained by comparing:

$$J_{AE} + J^*_{EC} \qquad J_{AD} + J^*_{DC} \qquad J_{AB} + J^*_{BC}$$

Table of Contents

Dynamic programming

It is a recursive method of solving optimal control problems by decomposing them into smaller optimal sub-problems

Given the optimal control problem of finding the optimal controller $\pi^*\left(x^{(k)}\right)$ solution of:

$$\pi^* = \arg\min_{\pi} x^{(H)^\top} Q_H x^{(H)} + \sum_{k=0}^{H-1} x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

s.t.:

$$x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right),\ x^{(0)} = x_{\text{ini}},\ x^{(H)} = 0$$

Dynamic programming

Denote by

$$J_{x^{(0)}} = {x^{(H)}}^\top Q_H x^{(H)} + \sum_{k=0}^{H-1} {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)}$$

the value of the performance index starting at $x^{(0)}$, and by

$$J^*_{x^{(H)}} = {x^{(H)}}^\top Q_H x^{(H)}$$

the optimal value of the performance index at the final state

Dynamic programming

DP algorithm

For every initial state $x^{(0)}$ the value of the performance index $J^*_{x^{(0)}}$ can be obtained starting from the optimal value of the performance index at the final state $J^*_{x^{(H)}}$ and proceeding **backward** in time from $k = H - 1$ to $k = 0$

$$J^*_{x^{(H-1)}} = J^*_{x^{(H)}} + \min \; {x^{(H-1)}}^\top Q x^{(H-1)} + {u^{(H-1)}}^\top R u^{(H-1)}$$

$$J^*_{x^{(k)}} = J^*_{x^{(k+1)}} + \min \; {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)}.$$

DP algorithm: optimal control sequence

The optimal control policy π^* is the one that produces the sequence of optimal control inputs $u^{(0)^*}, \ldots, u^{(H-1)^*}$ such that the optimal value of the performance index $J^*_{x^{(0)}}$ is achieved.

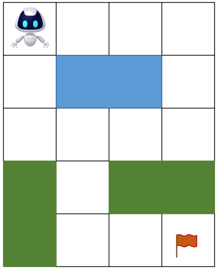
The sequence of optimal control inputs $u^{(0)^*}, \ldots, u^{(H-1)^*}$ can be computed **forward** in time from $k = 0$ to $k = H - 1$

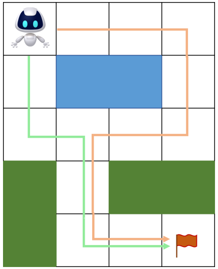
$$u^{(0)^*} = \underset{u^{(0)}=\pi\left(x^{(0)}\right)}{\arg\min} \; J^*_{x^{(1)}} + x^{(0)^\top} Q x^{(0)} + u^{(0)^\top} R u^{(0)}$$

$$u^{(k)^*} = \underset{u^{(k)}=\pi\left(x^{(k)}\right)}{\arg\min} \; J^*_{x^{(k+1)}} + x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

Dynamic programming example

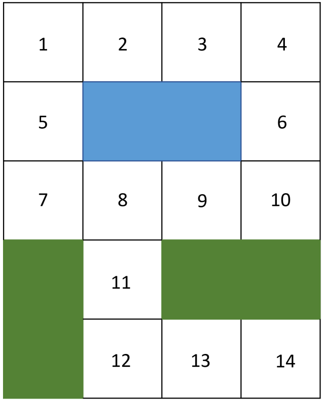
Let's consider a mobile robot that, starting from an initial point x_{ini} in a map, has to reach a goal x_{des} avoiding some obstacles in H steps.





Suppose the robot can move $\uparrow \downarrow \rightarrow \leftarrow$ STOP and set $H = 7$

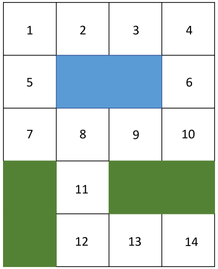
Dynamic programming example

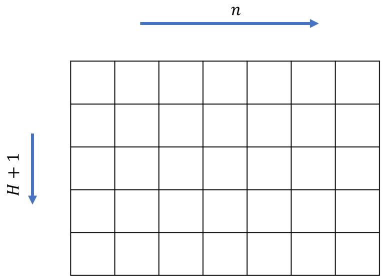


- Indexing admitted states in n values
- **State:** $x \in \mathcal{X} = \{1, 2, \dots, 14\}$
- **Control input:** $u \in \mathcal{U} = \{\uparrow \downarrow \rightarrow \leftarrow \text{ STOP}\}$
- **State transition function:**
  $x^{(k+1)} = x^{(k)} + u^{(k)}$ if $1 \leq x^{(k)} \leq 14$
  $x^{(k+1)} = x^{(k)}$ otherwise
- **Performance evaluation:**
  ${x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)} = 1$ if $x^{(k+1)} \in \{1, 2, \dots, 14\}$
  ${x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)} = \infty$ if $x^{(k+1)} \notin \{1, 2, \dots, 14\}$

Dynamic programming example

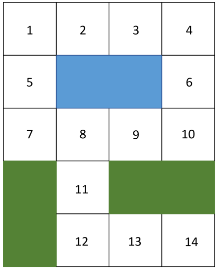
- Create a table that includes the cost values ($H + 1$ rows, n columns)





$n = 14, H = 7$

Dynamic programming example

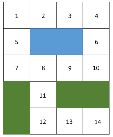


- Create a table that includes the cost values ($H+1$ rows, n columns)
- Start with the bottom-up approach:
  - assume that the cost at the $H+1$-th step is $\boxed{\infty}\boxed{\infty}\boxed{\dots}\boxed{\infty}\boxed{0}$
  - for each k-th step $\in \{1,\dots,H\}$ and for each i-th state index, apply each action among $\{\uparrow\downarrow\rightarrow\leftarrow \text{ STOP}\}$:
    1. if the action leads to a feasible state $j \in \{1,\dots,14\}$ the cost will assume the value of the table element $(k+1,\, j)+1$
    2. if the action leads to an unfeasible state $j \notin \{1,\dots,14\}$ the cost will assume the value ∞

    The $(k,\, i)$ table element will be set as the minimum values of the costs thus obtained and the corresponding action is stored

Dynamic programming example

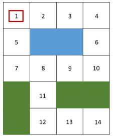
4 Dynamic programming



∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example

4 Dynamic programming

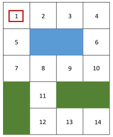


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = \infty, J_{\uparrow} = \infty, J_{\rightarrow} = (8,2) + 1, J_{\downarrow} = (8,5) + 1, J_{\text{stop}} = (8,1) = \infty$$

Dynamic programming example

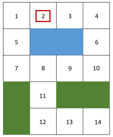
4 Dynamic programming



∞													
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min\{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = \infty$$

Dynamic programming example

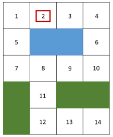


∞													
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = (8,1) + 1, J_{\uparrow} = \infty, J_{\rightarrow} = (8,3) + 1, J_{\downarrow} = \infty, J_{\text{stop}} = (8,2) = \infty$$

Dynamic programming example

4 Dynamic programming

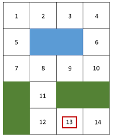


∞	∞												
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min\{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = \infty$$

Dynamic programming example

4 Dynamic programming

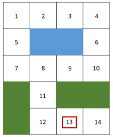


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞		
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = (8, 12) + 1, J_{\uparrow} = \infty, J_{\rightarrow} = (8, 14) + 1 = 0 + 1 = 1, J_{\downarrow} = \infty,$$
$$J_{\text{stop}} = (8, 13) = \infty$$

Dynamic programming example

4 Dynamic programming

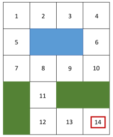


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min\{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = 1$$

Dynamic programming example

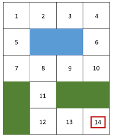
4 Dynamic programming



∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = (8, 13) + 1, J_{\uparrow} = \infty, J_{\rightarrow} = \infty, J_{\downarrow} = \infty, J_{\text{stop}} = (8, 14) = 0$$

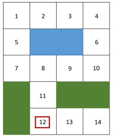
Dynamic programming example



∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min\{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = 0$$

Dynamic programming example

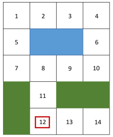


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞			
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = \infty, J_{\uparrow} = (7, 11) + 1, J_{\rightarrow} = (7, 13) + 1 = 2, J_{\downarrow} = \infty, J_{\text{stop}} = (7, 12) = \infty$$

Dynamic programming example

4 Dynamic programming

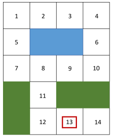


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$		
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min\{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = 2$$

Dynamic programming example

4 Dynamic programming

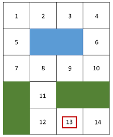


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$		
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = (7,12) + 1, J_{\uparrow} = \infty, J_{\rightarrow} = (7,14) + 1 = 1, J_{\downarrow} = \infty, J_{\text{stop}} = (7,12) = \infty$$

Dynamic programming example

4 Dynamic programming

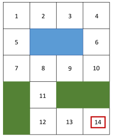


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min \{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = 1$$

Dynamic programming example

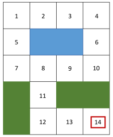
4 Dynamic programming



∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = (7, 13) + 1 = 2, J_{\uparrow} = \infty, J_{\rightarrow} = \infty, J_{\downarrow} = \infty, J_{\text{stop}} = (7, 14) = 0$$

Dynamic programming example

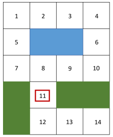


∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min\{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = 0$$

Dynamic programming example

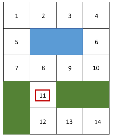
4 Dynamic programming



∞	∞	∞	∞	∞	∞	∞	∞	∞	∞				
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$J_{\leftarrow} = \infty, J_{\uparrow} = (6, 8) + 1, J_{\rightarrow} = \infty, J_{\downarrow} = (6, 12) + 1 = 3, J_{\text{stop}} = (6, 11)$$

Dynamic programming example



∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$			
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$$\min\{J_{\leftarrow}, J_{\uparrow}, J_{\rightarrow}, J_{\downarrow}, J_{\text{STOP}}\} = 3$$

Dynamic programming example

4 Dynamic programming

∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

proceeding in this way we obtain...

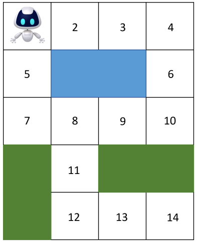
Dynamic programming example

4 Dynamic programming

$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

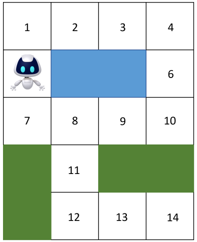
Dynamic programming example

- Indexing admitted states in n values
- Create a table that includes the cost values ($H + 1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach



1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example

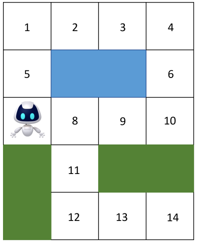


- Indexing admitted states in n values
- Create a table that includes the cost values ($H + 1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach

1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example

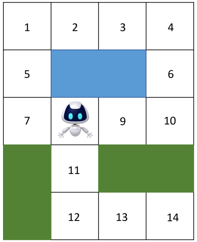
- Indexing admitted states in n values
- Create a table that includes the cost values ($H+1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach



1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7\downarrow$	∞	∞	∞	$6\downarrow$	$7\downarrow$	$5\rightarrow$	$4\downarrow$	$5\leftarrow$	$6\leftarrow$	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	$6\downarrow$	∞	$5\rightarrow$	$4\downarrow$	$5\leftarrow$	$6\leftarrow$	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5\rightarrow$	$4\downarrow$	$5\leftarrow$	∞	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4\downarrow$	∞	∞	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example

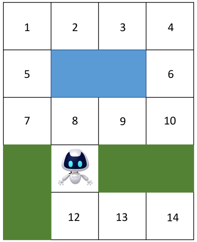
- Indexing admitted states in n values
- Create a table that includes the cost values ($H + 1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach



1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example

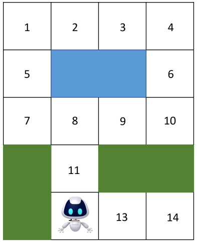
- Indexing admitted states in n values
- Create a table that includes the cost values ($H+1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach



1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7\downarrow$	∞	∞	∞	$6\downarrow$	$7\downarrow$	$5\rightarrow$	$4\downarrow$	$5\leftarrow$	$6\leftarrow$	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	$6\downarrow$	∞	$5\rightarrow$	$4\downarrow$	$5\leftarrow$	$6\leftarrow$	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5\rightarrow$	$4\downarrow$	$5\leftarrow$	∞	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4\downarrow$	∞	∞	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3\downarrow$	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2\rightarrow$	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1\rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

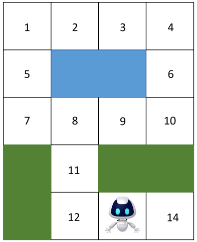
Dynamic programming example

- Indexing admitted states in n values
- Create a table that includes the cost values ($H + 1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach



1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example



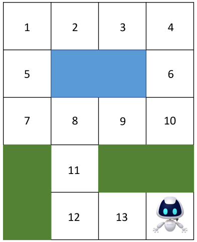
- Indexing admitted states in n values
- Create a table that includes the cost values ($H+1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach

1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example

4 Dynamic programming

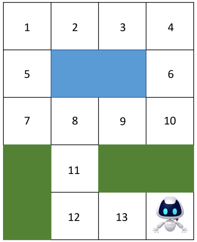
- Indexing admitted states in n values
- Create a table that includes the cost values ($H + 1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach



1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming example

4 Dynamic programming



- Indexing admitted states in n values
- Create a table that includes the cost values ($H + 1$ rows, n columns)
- Perform the bottom-up approach
- Perform the forward approach: select the optimal control inputs from the table of the bottom-up approach

1	2	3	4	5	6	7	8	9	10	11	12	13	14
$7 \downarrow$	∞	∞	∞	$6 \downarrow$	$7 \downarrow$	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	$6 \downarrow$	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	$6 \leftarrow$	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	$5 \rightarrow$	$4 \downarrow$	$5 \leftarrow$	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	$4 \downarrow$	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$3 \downarrow$	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$2 \rightarrow$	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$1 \rightarrow$	0 STOP
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dynamic programming limits

1. Consider the formula

$$J^*_{x^{(k)}} = J^*_{x^{(k+1)}} + \min\ {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)}$$

It is used to compute backward in time the optimal value of the performance index.

Dynamic programming limits

1. Consider the formula

$$J^*_{x^{(k)}} = J^*_{x^{(k+1)}} + \min\ x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

   It is used to compute backward in time the optimal value of the performance index.

However $\min\ x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$ means that we should try all the feasible $u^{(k)} \in \mathcal{U}$ and select that one minimizing that expression.

Dynamic programming limits

1. Consider the formula

$$J^*_{x^{(k)}} = J^*_{x^{(k+1)}} + \min\; {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)}$$

   It is used to compute backward in time the optimal value of the performance index.

However $\min\; {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)}$ means that we should try all the feasible $u^{(k)} \in \mathcal{U}$ and select that one minimizing that expression.

Therefore:

- Even if it guarantees convergence to the optimal solution
- It is susceptible to the curse of dimensionality, meaning that the number of sub-problems can increase exponentially with the size of the problem or the number of variables.

Dynamic programming limits

2. Consider the backward procedure:

$$J^*_{x^{(H-1)}} = J^*_{x^{(H)}} + \min \, {x^{(H-1)}}^\top Q x^{(H-1)} + {u^{(H-1)}}^\top R u^{(H-1)}$$

$$\dots$$

$$J^*_{x^{(k)}} = J^*_{x^{(k+1)}} + \min \, {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)}.$$

It starts from the assumption that $J^*_{x^{(H)}}$ is known.
In several cases this assumption is realistic since $J^*_{x^{(H)}} = 0$.

What if we don't know it?

Table of Contents

5 Bellman’s equations

Value function

A **value function** evaluates the benefit of being in a given state in terms of the overall task achievement.

Therefore, given a controller π, it can be defined as the value of the performance index computed in the state $x^{(k)}$ following π

$$V_\pi\left(x^{(k)}\right) = x^{(k)\top} Q x^{(k)} + \pi\left(x^{(k)}\right)^\top R \pi\left(x^{(k)}\right) + V_\pi\left(x^{(k+1)}\right)$$

Value function

5 Bellman's equations

A **value function** evaluates the benefit of being in a given state in terms of the overall task achievement.

Therefore, given a controller π, it can be defined as the value of the performance index computed in the state $x^{(k)}$ following π

$$V_\pi\left(x^{(k)}\right) = x^{(k)\top} Q x^{(k)} + \pi\left(x^{(k)}\right)^\top R \pi\left(x^{(k)}\right) + V_\pi\left(x^{(k+1)}\right)$$

Suppose that we are working with an objective cost function (optimal control problem expressed as a minimization problem), then the **optimal value function** is $\forall x \in \mathcal{X}$:

$$V_\pi^* = \min_\pi \left[x^\top Q x + \pi\left(x\right)^\top R \pi\left(x\right) + V_\pi\left(x\right)\right]$$

Bellman's equations

Starting from the optimal value function

$$V_\pi^* = \min_\pi \left[\boldsymbol{x}^\top Q \boldsymbol{x} + \pi\left(\boldsymbol{x}\right)^\top R \pi\left(\boldsymbol{x}\right) + V_\pi\left(\boldsymbol{x}\right) \right]$$

we can also define the **optimal controller**

$$\pi^* = \arg\min_\pi V_\pi^*$$

Bellman’s equations

Starting from the optimal value function

$$V_\pi^* = \min_\pi \left[\boldsymbol{x}^\top Q \boldsymbol{x} + \pi\left(\boldsymbol{x}\right)^\top R \pi\left(\boldsymbol{x}\right) + V_\pi\left(\boldsymbol{x}\right) \right]$$

we can also define the **optimal controller**

$$\pi^* = \arg\min_\pi V_\pi^*$$

BELLMAN’S OPTIMALITY EQUATION

Table of Contents

6 Value and policy iteration

Relaxation schemes

Bellman's equations can be exploited to develop *forward-in-time* methods for solving optimal control problems:

1. **Value iteration**
2. **Policy iteration**

Both methods are initialized with a guessed solution and iteratively improve it to converge to the optimal one.

Value iteration

Consider the optimal control problem of finding:

$$\pi^* = \arg\min_{\pi} \sum_{k=0}^{H-1} x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

$$\text{s.t.} \qquad x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right), \; x^{(0)} = x_{\text{ini}}, \; x^{(H)} = 0$$

Value iteration searches for the **optimal value function**, which consists of the maximal returns from every state or from every state-action pair.

The optimal value function is used to compute an optimal policy.

Value iteration

The algorithm steps are:

Initialization. Select a guess $V_i = V_0$

Value improvement. $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$

$$V_{i+1}(x) = \min_{u \in \mathcal{U}} \left(x^\top Q x + u^\top R u + V_i\left(f(x,u)\right) \right)$$

Terminal condition. Value improvement is repeated until $V_{i+1} = V_i$.

It has been proven that value iteration asymptotically converges to V_π^* as $i \to \infty$.

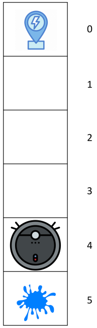
An **optimal policy** can be computed from V_π^* applying

$$\pi^* = \arg\min_{\pi} V_\pi^*$$

Example

6 Value and policy iteration

Consider a robot vacuum cleaner that needs to clean a patch on the floor and also needs to recharge the batteries.



- **State:** $x \in \mathcal{X} = \{0, 1, 2, 3, 4, 5\}$
- **Control input:** $u \in \mathcal{U} = \{-1, 1\}$
- **State transition function:**
$$x^{(k+1)} = x^{(k)} + u^{(k)} \qquad \text{if } 1 \leq x^{(k)} \leq 4$$
$$x^{(k+1)} = x^{(k)} \qquad \text{if } x^{(k)} = 0 \text{ or } x^{(k)} = 5$$
- **Performance evaluation:**
$$x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)} = -5 \quad \text{if } x^{(k)} = 4 \text{ and } u^{(k)} = 1$$
$$x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)} = -1 \quad \text{if } x^{(k)} = 1 \text{ and } u^{(k)} = -1$$
$$x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)} = 0 \text{ otherwise}$$

Example

6 Value and policy iteration

Initialization.

$V_0 =$

0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0

Example

Initialization.

$V_0 =$

0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0

Value improvement. $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$

$$V_{i+1}(x) = \min_{u \in \mathcal{U}} \left(x^\top Q x + u^\top R u + V_i(f(x,u)) \right)$$

$V_0 =$

0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0

$V_1 =$

0	**1**	**2**	**3**	**4**	**5**
0	-1	0	0	-5	0
	$\leftarrow$			$\rightarrow$	

Example

6 Value and policy iteration

Initialization.

$V_0 =$

0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0

Value improvement. $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$

$$V_{i+1}(x) = \min_{u \in \mathcal{U}} \left(x^\top Q x + u^\top R u + V_i(f(x,u)) \right)$$

$V_1 =$

0	**1**	**2**	**3**	**4**	**5**
0	-1	0	0	-5	0

$V_2 =$

0	**1**	**2**	**3**	**4**	**5**
0	-1	-1	-5	-5	0
	$\leftarrow$	$\leftarrow$	$\rightarrow$	$\rightarrow$	

Example

6 Value and policy iteration

Initialization.

$V_0 =$

0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0

Value improvement. $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$

$$V_{i+1}(x) = \min_{u \in \mathcal{U}} \left(x^\top Q x + u^\top R u + V_i(f(x,u)) \right)$$

$V_2 =$

0	**1**	**2**	**3**	**4**	**5**
0	-1	-1	-5	-5	0

$V_3 =$

0	**1**	**2**	**3**	**4**	**5**
0	-1	-5	-5	-5	0
	$\leftarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	

Example

Initialization.

$V_0 =$

0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0

Value improvement. $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$

$$V_{i+1}(x) = \min_{u \in \mathcal{U}} \left(x^\top Q x + u^\top R u + V_i(f(x,u)) \right)$$

$V_2 =$

0	**1**	**2**	**3**	**4**	**5**
0	-1	-5	-5	-5	0

$V_3 =$

0	**1**	**2**	**3**	**4**	**5**
0	-5	-5	-5	-5	0
	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	

Example

Initialization.

$V_0 =$

0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0

Value improvement. $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$

$$V_{i+1}(x) = \min_{u \in \mathcal{U}} \left(x^\top Q x + u^\top R u + V_i(f(x, u)) \right)$$

$V_3 =$

0	**1**	**2**	**3**	**4**	**5**
0	-5	-5	-5	-5	0

$V_4 =$

0	**1**	**2**	**3**	**4**	**5**
0	-5	-5	-5	-5	0
	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	

Example

6 Value and policy iteration

Initialization.

$$V_0 = \begin{array}{|c|c|c|c|c|c|}\hline \mathbf{0} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Value improvement. $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$

$$V_{i+1}(x) = \min_{u \in \mathcal{U}} \left(x^\top Q x + u^\top R u + V_i(f(x,u)) \right)$$

$$V_3 = \begin{array}{|c|c|c|c|c|c|}\hline \mathbf{0} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \hline 0 & -5 & -5 & -5 & -5 & 0 \\ \hline \end{array}$$

$$V_4 = \begin{array}{|c|c|c|c|c|c|}\hline \mathbf{0} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \hline 0 & -5 & -5 & -5 & -5 & 0 \\ \hline & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \\ \hline \end{array}$$

Terminal condition $\rightarrow$ $\quad V_4 = V_\pi^*$

Policy iteration

Consider the optimal control problem of finding:

$$\pi^* = \arg\min_{\pi} \sum_{k=0}^{H-1} x^{(k)^\top} Q x^{(k)} + u^{(k)^\top} R u^{(k)}$$

$$\text{s.t.} \qquad x^{(k+1)} = f\left(x^{(k)}, u^{(k)}\right),\ x^{(0)} = x_{\text{ini}},\ x^{(H)} = 0$$

Policy iteration evaluates **controllers** by constructing their **value functions** (instead of the optimal value function), and **uses these value functions to find new, improved controllers**.

Policy iteration

The algorithm steps are:

Initialization. Select a guess $\pi_i = \pi_0$

Policy evaluation (PE). Determine the value of the current policy

- Select a guess $V_j = V_0$
- Repeat until $V_{j+1} = V_j$
  $\forall x \in \mathcal{X}, \quad V_{j+1}(x) = x^\top Q x + \pi_i(x)^\top R \pi_i(x) + V_j(f(x, \pi_i(x)))$
- $V_{\pi_i} = V_{j+1} = V_j$

Policy improvement (PI). Determine an improved policy

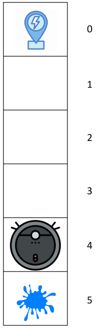
$$\pi_{i+1} = \arg\min_{\pi} V_{\pi_i}$$

Terminal condition. PE and PI are repeated until $\pi_{i+1} = \pi_i$.

It has been proven that the sequence of V_π asymptotically converges to V_π^* as $i \to \infty$. Simultaneously, an **optimal policy** π^* is obtained.

Example

Try to solve the previously defined control problem involving a robot vacuum cleaner with policy iteration.



- **State:** $x \in \mathcal{X} = \{0, 1, 2, 3, 4, 5\}$
- **Control input:** $u \in \mathcal{U} = \{-1, 1\}$
- **State transition function:**
$$\begin{aligned} x^{(k+1)} &= x^{(k)} + u^{(k)} && \text{if } 1 \leq x^{(k)} \leq 4 \\ x^{(k+1)} &= x^{(k)} && \text{if } x^{(k)} = 0 \text{ or } x^{(k)} = 5 \end{aligned}$$
- **Performance evaluation:**
$$\begin{aligned} {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)} &= -5 && \text{if } x^{(k)} = 4 \text{ and } u^{(k)} = 1 \\ {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)} &= -1 && \text{if } x^{(k)} = 1 \text{ and } u^{(k)} = -1 \\ {x^{(k)}}^\top Q x^{(k)} + {u^{(k)}}^\top R u^{(k)} &= 0 \text{ otherwise} \end{aligned}$$

Questions’ time!

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE