CONTROLLABILITY

👤 Sébastien Boisgérault

CONTROL ENGINEERING WITH PYTHON

- 📖 Documents (GitHub)

- ©️ License CC BY 4.0

- 🏛️ Mines ParisTech, PSL University

SYMBOLS

🐍	Code	🔍	Worked Example
📈	Graph	🧩	Exercise
🏷️	Definition	💻	Numerical Method
💎	Theorem	🧮	Analytical Method
📝	Remark	🧠	Theory
ℹ️	Information	🗝️	Hint
⚠️	Warning	🔓	Solution
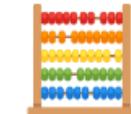
🐍 IMPORTS

```
from numpy import *

from numpy.linalg import *

from numpy.testing import *

from scipy.integrate import *

from scipy.linalg import *

from matplotlib.pyplot import *
```
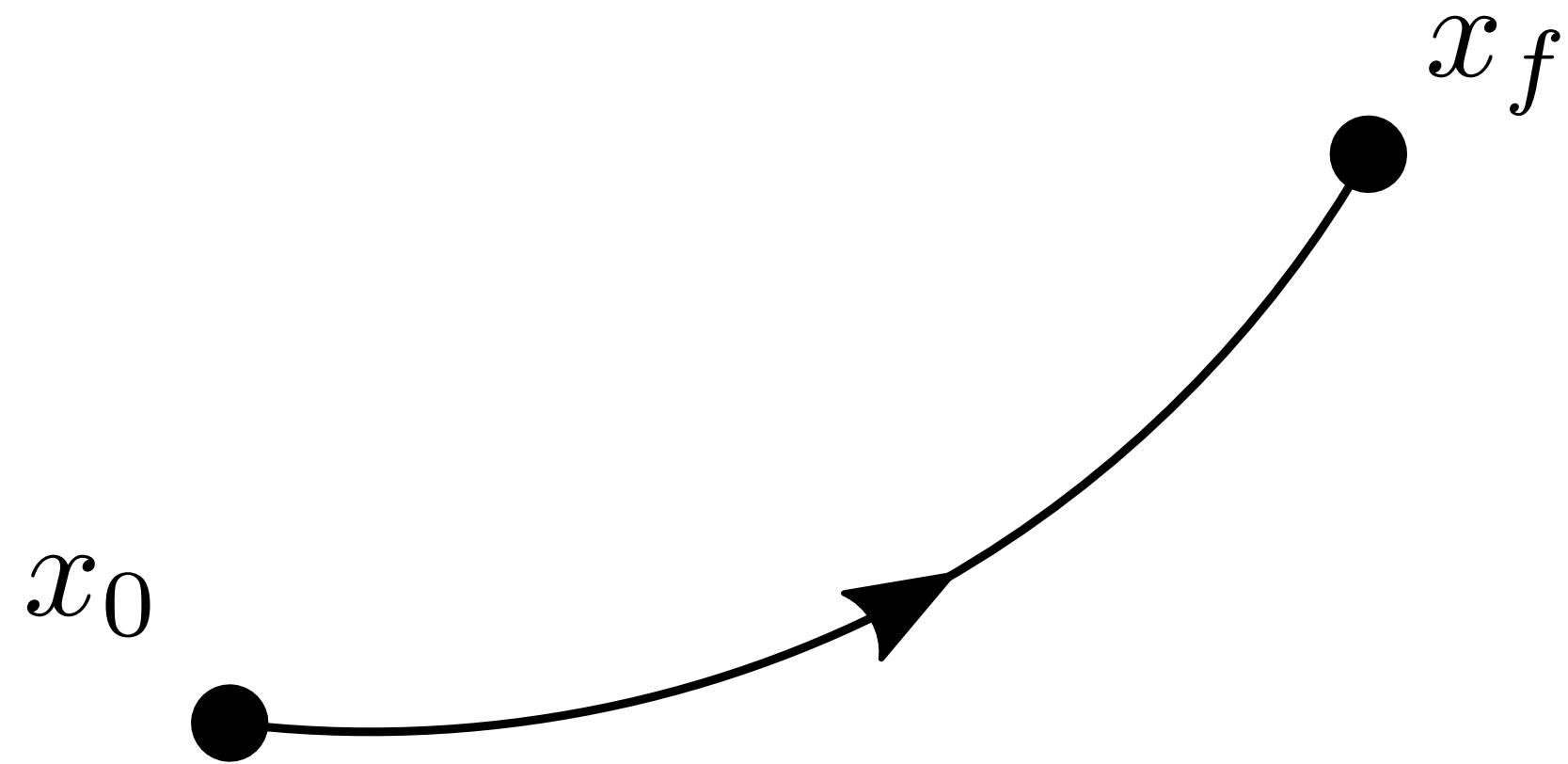
🏷️ CONTROLLABILITY

The system $\dot{x} = f(x, u)$ is **controllable** if

- for any $t_0 \in \mathbb{R}$, $x_0 \in \mathbb{R}^n$ and $x_f \in \mathbb{R}^n$,
- there are $t_f > t_0$ and $u : [t_0, t_f] \to \mathbb{R}^m$ such that
- the solution $x(t)$ such that $x(t_0) = x_0$ satisfies

$$x(t_f) = x_f.$$

x_0

x_f

🏷️ ADMISSIBLE TRAJECTORY

Let x_r be a reference trajectory of the system state:

$$x_r(t), \ t \in [t_0, t_f].$$

It is **admissible** if there is a function $u_r(t), t \in [t_0, t_f]$ such that the solution x of the IVP

$$\dot{x} = f(x, u_r), \ x(t_0) = x_r(t_0)$$

is the function x_r.

🔍 CAR

The position d (in meters) of a car of mass m (in kg) on a straight road is governed by

$$m\ddot{d} = u$$

where u the force (in Newtons) generated by its motor.

The car is initially at the origin of a road and motionless.

We would like to cross the end of the road (location $d_f > 0$) at time $t_f > 0$ and speed v_f.

Numerical values:

- $m = 1500\,\mathrm{kg}$,
- $t_f = 10\,\mathrm{s}$, $d_f = 100\,\mathrm{m}$ and $v_f = 100\,\mathrm{km/h}$.

STEP 1 – TRAJECTORY PLANNING

We search for a **reference trajectory** for the state

$$x_r(t) = (d_r(t), \dot{d}_r(t))$$

such that:

- $d_r(0) = 0, \dot{d}_r(0) = 0,$
- $d_r(t_f) = x_f, \dot{d}_r(t_f) = v_f.$

STEP 2 – ADMISSIBILITY

We check that this reference trajectory is **admissible**, i.e. that we can find a control $u_r(t)$ such that the solution of the IVP is $x(t) = x_r(t)$ when $x(0) = x_r(t)$.

ADMISSIBLE TRAJECTORY

Here, if d_r is smooth and if we apply the control
$$u(t) = m\ddot{d}_r(t),$$

$$m\frac{d^2}{dt^2}(d - d_r) = 0,$$

$$(d - d_r)(0) = 0, \ \frac{d}{dt}(d - d_r)(0) = 0.$$

Thus, $d(t) = d_r(t)$ – and thus $\dot{d}(t) = \dot{d}_r(t)$ – for every $t \geq 0$.

REFERENCE TRAJECTORY

We can find d_r as a third-order polynomial in t

$$d_r(t) = \alpha t^3 + \beta t^2 + \gamma t + \delta$$

with

$$\alpha = \frac{v_f}{t_f^2} - 2\frac{d_f}{t_f^3}, \ \beta = 3\frac{d_f}{t_f^2} - \frac{v_f}{tf}, \ \gamma = 0, \ \delta = 0.$$

(equivalently, with $u(t)$ as an affine function of t).

🐍 CONSTANTS

```python
m = 1500.0
xf = 100.0
vf = 100.0 * 1000 / 3600 # m/s
tf = 10.0
alpha = vf/tf**2 - 2*xf/tf**3
beta = 3*xf/tf**2 - vf/tf
```

🐍 STATE & INPUT EVOLUTION

```python
def x(t):
    return alpha * t**3 + beta * t**2
def d2_x(t):
    return 6 * alpha * t + 2 * beta
def u(t):
    return m * d2_x(t)
```

🐍 AUTOMATIC DIFFERENTIATION

Alternatively, use the autograd package:

```python
import autograd.numpy as numpy

from autograd import elementwise_grad as egrad

def x(t):

    return alpha * t**3 + beta * t**2

d2_x = egrad(egrad(x))

def u(t):

    return m * d2_x(t)
```

🐍 💻 SIMULATION
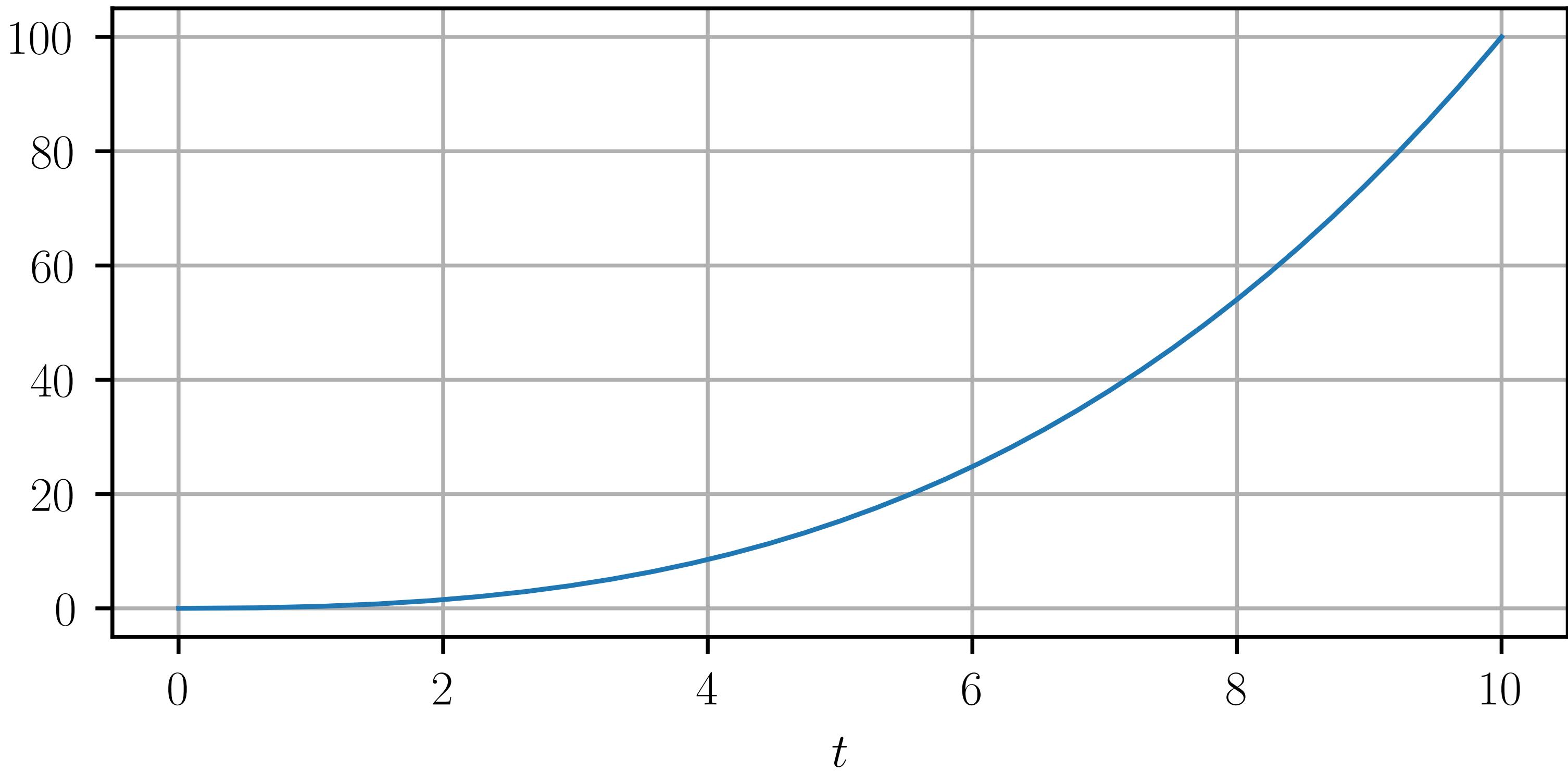
```python
y0 = [0.0, 0.0]
def fun(t, y):
    x, d_x = y
    d2_x = u(t) / m
    return [d_x, d2_x]
result = solve_ivp(
  fun, [0.0, tf], y0, dense_output=True
)
```

📊 GRAPH OF THE DISTANCE
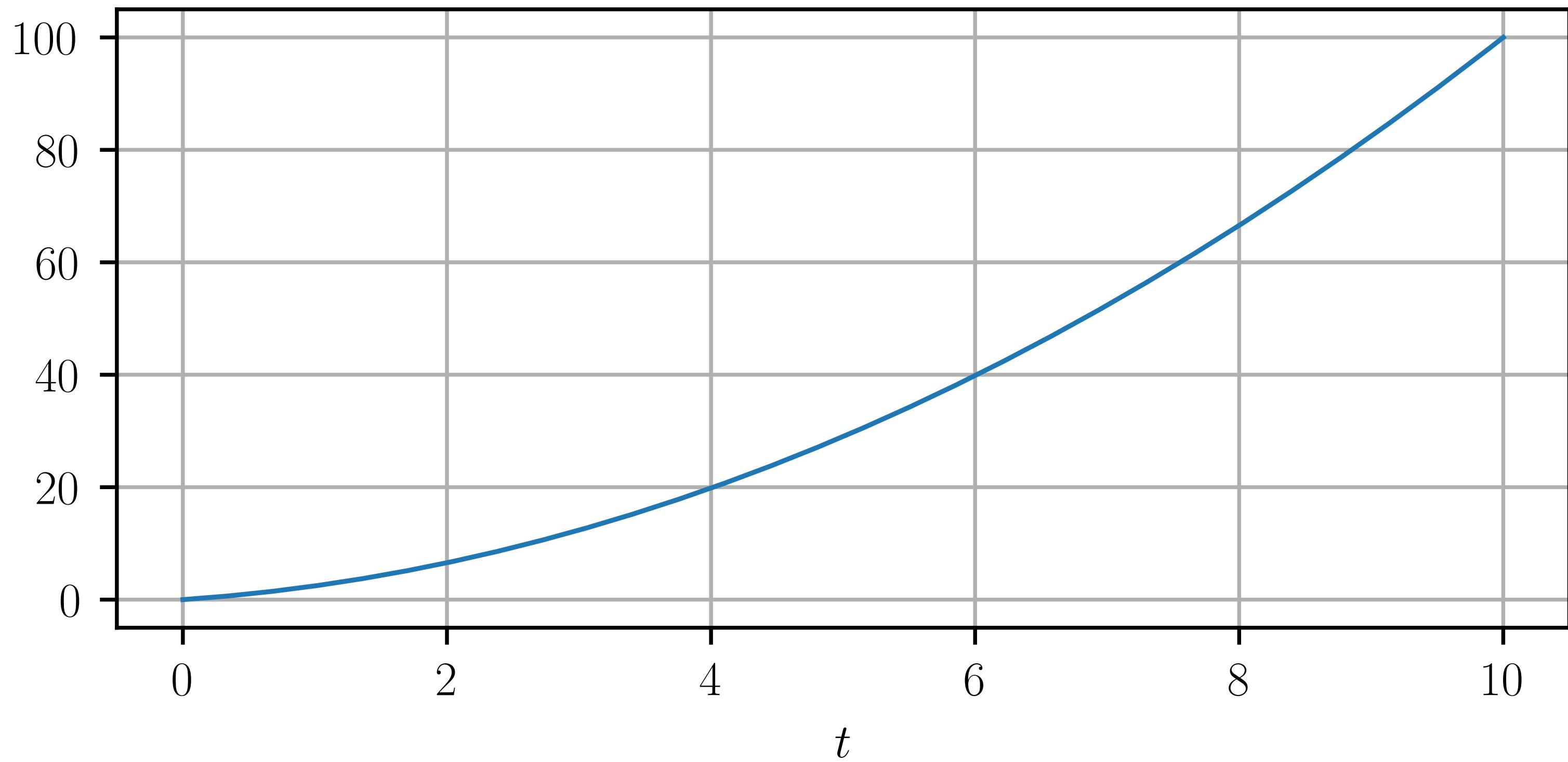
```
figure()
t = linspace(0, tf, 1000)
xt = result["sol"](t)[0]
plot(t, xt)
grid(True); xlabel("$t$"); title("$d(t)$")
```

$d(t)$

📊 GRAPH OF THE VELOCITY

```python
figure()
vt = result["sol"](t)[1]
plot(t, 3.6 * vt)
grid(True); xlabel("$t$")
title(r"$\dot{d}(t)$ km/h")
```

$\dot{d}(t)$ km/h

🧩 NON-ADMISSIBLE TRAJECTORY

Let $\dot{x} = Ax + Bu$ with $x \in \mathbb{R}^2, u \in \mathbb{R}$,

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

1. 🧠

Find a smooth reference trajectory $x_r(t), t \in [0, 1]$ which is not admissible.

🔓 **NON-ADMISSIBLE TRAJECTORY**

1. 🔓

The first line of the vector equation $\dot{x} = Ax + Bu$ is

$$\dot{x}_1 = x_2.$$

Any trajectory x_r that does not satisfy this equation is not admissible; for example

$$x_r(t) := \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$
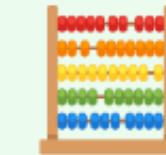
🧩 PENDULUM

Consider the pendulum with dynamics:

$$m\ell^2\ddot{\theta} + b\dot{\theta} + mg\ell \sin\theta = u$$

1. 🧠 🧮
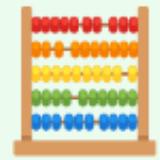
Find a smooth reference trajectory that leads the pendulum from the bottom configuration

$$\theta(0) = 0, \ \dot{\theta}(0) = 0$$

to the top configuration

$$\theta(t_f) = \pi, \ \dot{\theta}(t_f) = 0.$$

2. 🧠 🧮

Show that the reference trajectory is admissible and compute the corresponding input $u_r(t)$.

3. 💻 🧠

Simulate the result and visualize the solution.

What should theoretically happen at $t = t_f$ if $u(t) = 0$ is applied when $t \geq t_f$? What does happen in reality ? Why ? How can we mitigate this issue?

Numerical Values:

$$m = 1.0, \ l = 1.0, \ b = 0.1, \ g = 9.81, \ t_f = 10.$$

🔓 **PENDULUM**

1. 🔓

Since the initial and final conditions form a set of 4 equations, we search for an admissible trajectory defined by a 3rd-order polynomial

$$\theta_r(t) := at_f^3 + bt_f^2 + ct_f + d$$

with unknown coefficients a, b, c, d.

The initial conditions $\theta_r(0) = 0$ and $\dot{\theta}_r(0) = 0$ are equivalent to

$$b = 0, \ c = 0.$$

The final condition $\theta_r(t_f) = \pi$ is equivalent to

$$at_f^3 + bt_f^2 = \pi$$

and $\dot{\theta}_r(t_f) = 0$ to

$$3at_f^2 + 2bt_f = 0.$$

The latter equation can be transformed into

$$b = -\frac{3}{2}at_f$$

and the former becomes

$$at_f^3 - \frac{3}{2}at_f^3 = \pi \iff a = -\frac{2\pi}{t_f^3}$$

and thus $b = \frac{3\pi}{t_f^2}$.

Finally, the following trajectory $(\theta_r(t), \dot{\theta}_t(t))$ is smooth and meets the required initial and final conditions:

$$\theta_r(t) = -2\pi \left(\frac{t}{t_f} \right)^3 + 3\pi \left(\frac{t}{t_f} \right)^2$$

$$\dot{\theta}_r(t) = -\frac{6\pi}{t_f} \left(\frac{t}{t_f} \right)^2 + \frac{6\pi}{t_f} \left(\frac{t}{t_f} \right)$$

2. 🔓

By construction our reference trajectory meets the initial condition. There is a unique input that makes the solution follows this reference; its is defined by

$$u_r(t) := m\ell^2 \ddot{\theta}_r(t) + b\dot{\theta}_r(t) + mg\ell \sin \theta_r(t)$$

where

$$\ddot{\theta}_r(t) = -\frac{12\pi}{t_f^2}\left(\frac{t}{t_f}\right) + \frac{6\pi}{t_f^2}.$$

3. 🔓

```python
m = 1.0
l = 1.0
b = 0.1
g = 9.81
t0, tf = 0.0, 10.0
```

```python
def theta_r(t):
    s = t/tf
    return -2*pi*s**3 + 3*pi*s**2
def dtheta_r(t):
    s = t/tf
    return -6*pi/tf*s**2 + 6*pi/tf*s
```
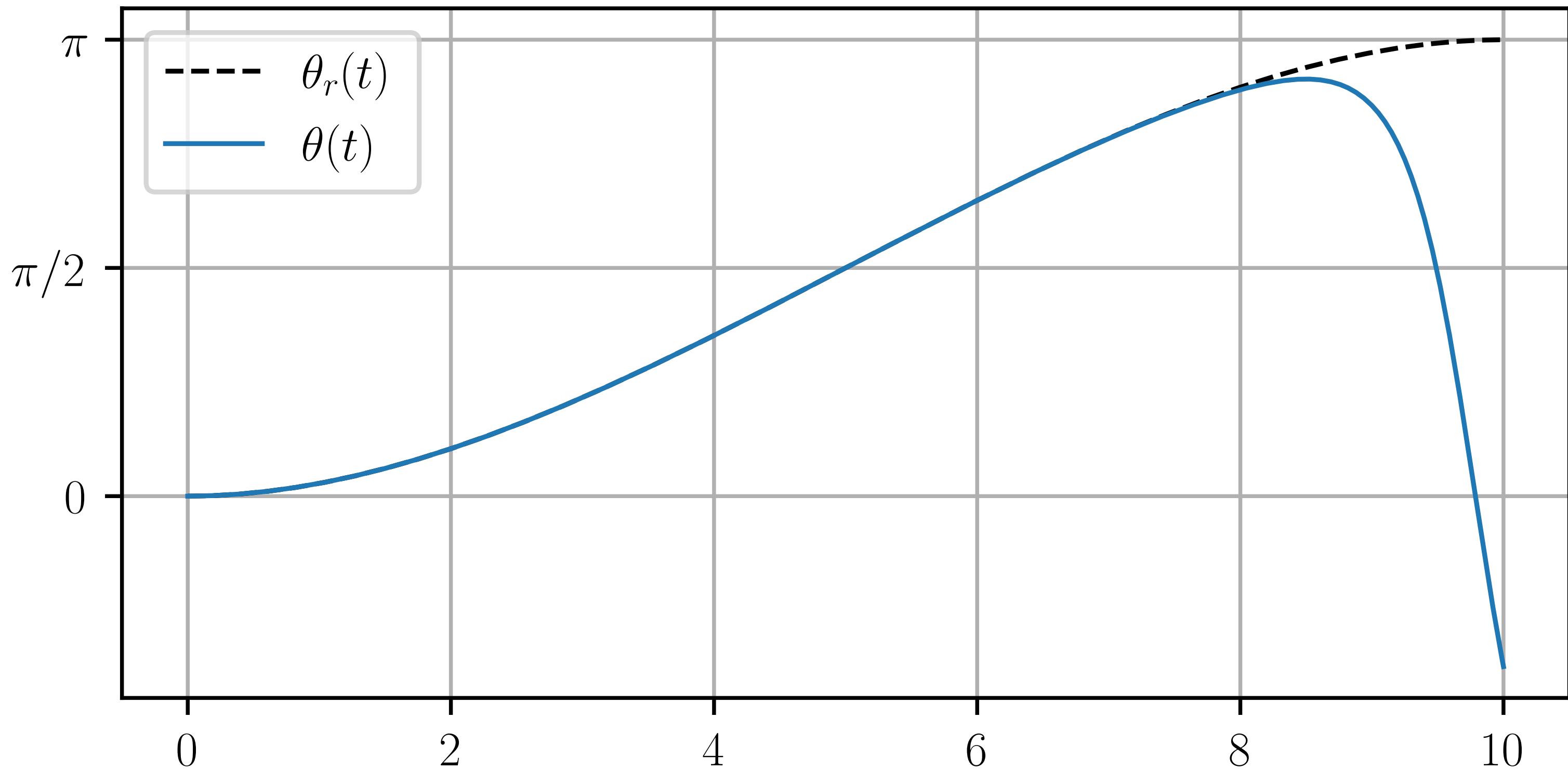
```python
def d2theta_r(t):
    s = t/tf
    return -12*pi/(tf*tf)*s + 6*pi/(tf*tf)
def u_r(t):
    return (m*l*l*d2theta_r(t) +
            b*dtheta_r(t) +
            m*g*l*sin(theta_r(t)))
```

```python
def fun(t, theta_dtheta):
    theta, dtheta = theta_dtheta
    d2theta = ((-b*dtheta - m*g*l*sin(theta) + u_r(t))
               / (m * l * l))
    return dtheta, d2theta
t_span = [t0, tf]
y0 = [0.0, 0.0]
```

```python
r = solve_ivp(fun, t_span, y0, dense_output=True)
t = linspace(t0, tf, 1000)
solt = r["sol"](t)
thetat, dthetat = solt
```

```
figure()
plot(t, theta_r(t), "k--", label=r"$\theta_r(t)$")
plot(t, thetat, label=r"$\theta(t)$")
yticks([0, 0.5*pi, pi], ["$0$", r"$\pi/2$", r"$\pi$"])
title(r"Simulation of $\theta(t)$")
grid(True); legend()
```

Simulation of $\theta(t)$

Theoretically, we should have $\theta(t_f) = \pi$ and $\dot{\theta}(t_f) = 0$, but the simulation is quite far from that.
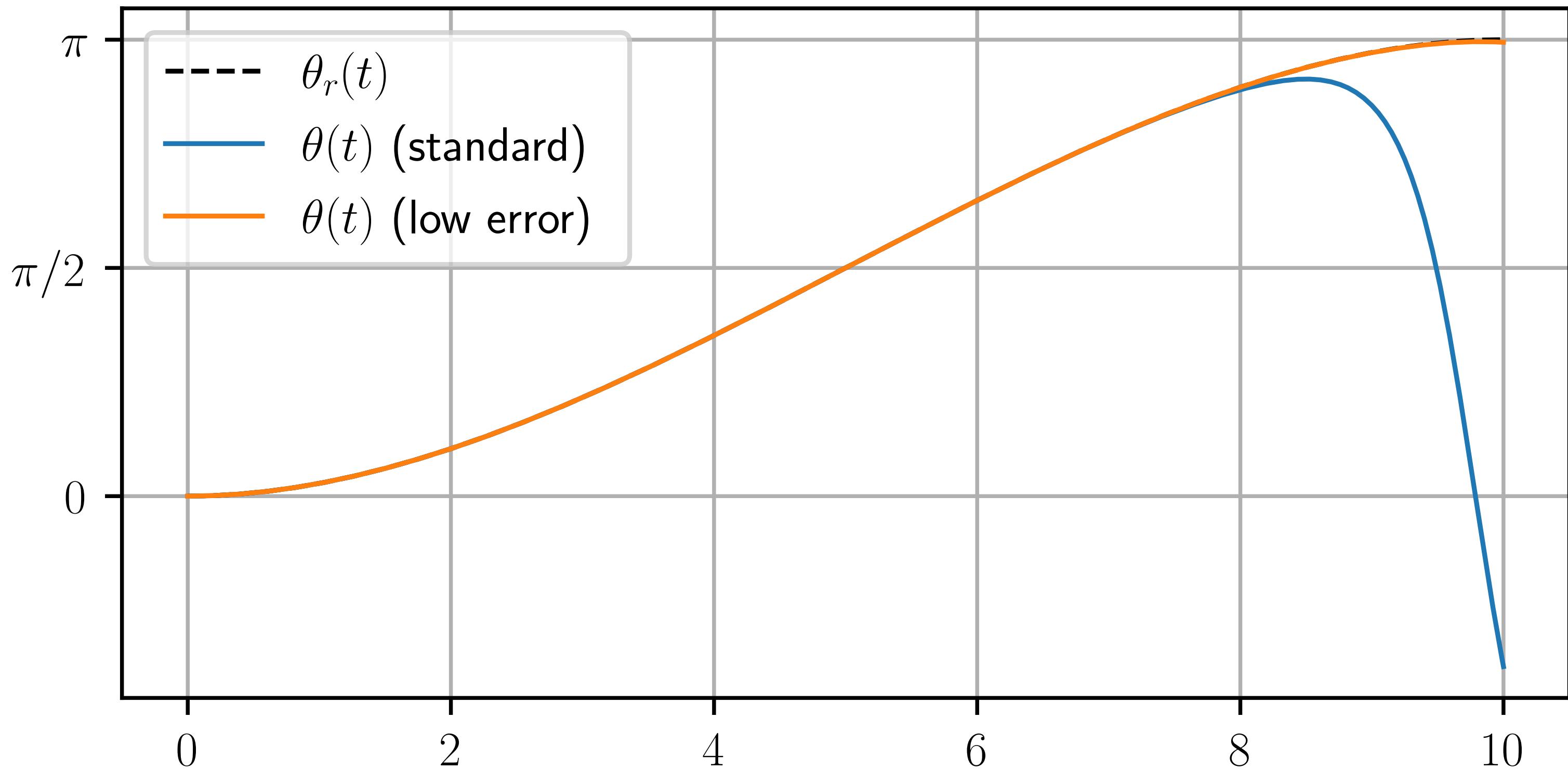
Since the top equilibrium is unstable, small (numerical) errors in the state may cause large deviations of the trajectory.

We may reduce the simulation error thresholds to alleviate this problem.

```
rhp = solve_ivp(fun, t_span, y0,
                  dense_output=True,
                  rtol=1e-6, atol=1e-9)
t = linspace(t0, tf, 1000)
solt_hp = rhp["sol"](t)
thetat_hp, dthetat_hp = solt_hp
```

```python
figure()
plot(t, theta_r(t), "k--", label=r"$\theta_r(t)$")
plot(t, thetat, label=r"$\theta(t)$ (standard)")
plot(t, thetat_hp, label=r"$\theta(t)$ (low error)")
yticks([0, 0.5*pi, pi], ["$0$", r"$\pi/2$", r"$\pi$"])
title(r"Simulation of $\theta(t)$")
grid(True); legend()
```

Simulation of $\theta(t)$

💎 CONTROLLABILITY OF LTI SYSTEMS

A system $\dot{x} = Ax + Bu$ is controllable iff

- from the origin $x_0 = 0$ at $t_0 = 0$,

- we can reach any state $x_f \in \mathbb{R}^n$.

🏷️ 💎 KALMAN CRITERION

The system $\dot{x} = Ax + Bu$ ($x \in \mathbb{R}^n$) is controllable iff:

$$\text{rank}\ \left[B, AB, \ldots, A^{n-1}B\right] = n$$

$[B, \ldots, A^{n-1}B]$ is the **controllability matrix**.

🔍 CONTROLLABILITY MATRIX

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

🐍 COMPUTATION

```python
def KCM(A, B):
    n = shape(A)[0]
    mp = matrix_power
    cs = column_stack
    return cs([mp(A, k) @ B for k in range(n)])
```

🐍 LTI SYSTEM

```python
n = 3
A = zeros((n, n))
for i in range(0, n-1):
    A[i,i+1] = 1.0


B = zeros((n, 1))
B[n-1, 0] = 1.0
```

🐍 RANK CONDITION

```
C = KCM(A, B)


C_expected = [[0, 0, 1], [0, 1, 0], [1, 0, 0]]
assert_almost_equal(C, C_expected)


assert matrix_rank(C) == n
```

⚠️ WARNINGS

- This implementation of KCM is not optimized: fewer computations are achievable using the identity:

$$A^n B = A \times (A^{n-1} B).$$

- Rank computations are subject to (catastrophic) numerical errors; sensitivity analysis or symbolic computations may alleviate the problem.

🧩 FULLY ACTUATED SYSTEM
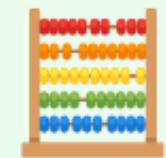
Consider $\dot{x} = Ax + Bu$ with

- $x \in \mathbb{R}^n$,

- $u \in \mathbb{R}^m$ and,

- $\text{rank } B = n$.

1. 🧠
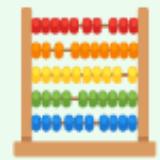
Show that $m \geq n$.

2. 🧮

Is the system controllable ?

3. 🧠 🧮

Given x_0, x_f and $t_f > 0$, show that any smooth trajectory that leads from x_0 at time t_0 to x_f at time t_f is admissible.

🔓 **FULLY ACTUATED SYSTEM**

1. 🔓

The shape of B is $n \times m$, thus

$$\operatorname{rank} B \leq \min(n, m).$$

By assumption $\operatorname{rank} B = n$, thus $n \leq m$.

2. 🔓

The system is controllable since

$$\operatorname{rank}[B, AB, \ldots] \leq \operatorname{rank} B = n.$$

3. 🔓

Since $\text{rank } B = n$, matrix B contains a $n \times n$ invertible matrix R. Let's assume for the sake of simplicity that R is made of the first n columns of B and let

$$S := \begin{bmatrix} R^{-1} \\ 0 \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

Then by construction $B \times S = I \in \mathbb{R}^{n \times n}$.

If we define u as a function of some auxiliary control $v \in \mathbb{R}^n$ by

$$u(t) = Sv(t),$$

then

$$\dot{x} = Ax + v.$$

To join x_0 at t_0 and x_f at t_f, we can apply the control

$$v(t) := -Ax_r(t) + \dot{x}_r(t)$$

where

$$x_r(t) := x_0 + \frac{t - t_0}{t_f - t_0}(x_f - x_0).$$

Indeed, that leads to

$$\dot{x}(t) = Ax(t) - Ax_r(t) + \dot{x}_r(t), \; x(t_0) = x_r(t_0)$$
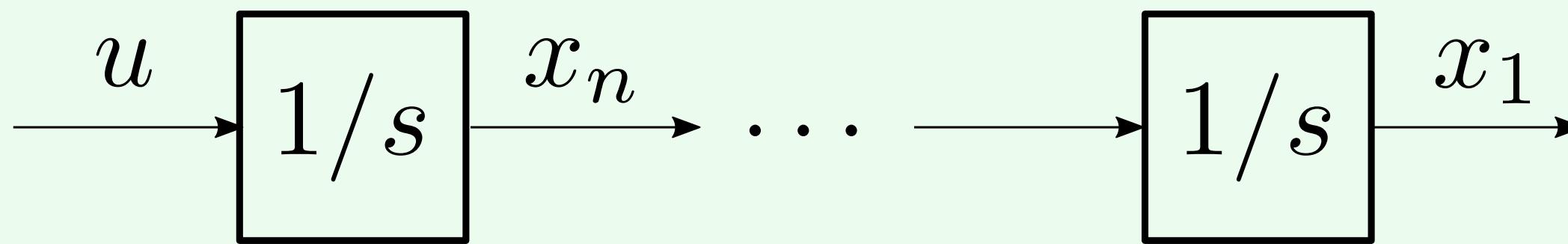
or if we denote $e := x - x_r$,

$$\dot{e}(t) = Ae(t), \; e(t_0) = 0$$

and thus yields $x(t) = x_r(t)$ for any $t \geq t_0$.

🧩 INTEGRATOR CHAIN

$$u \longrightarrow \boxed{1/s} \xrightarrow{x_n} \cdots \longrightarrow \boxed{1/s} \xrightarrow{x_1}$$

$$\dot{x}_n = u, \ \dot{x}_{n-1} = x_n, \ \cdots, \dot{x}_1 = x_2.$$

1. 🧠 🧮

Show that the system is controllable

🔓 **INTEGRATOR CHAIN**

1. 🔓

We have

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$
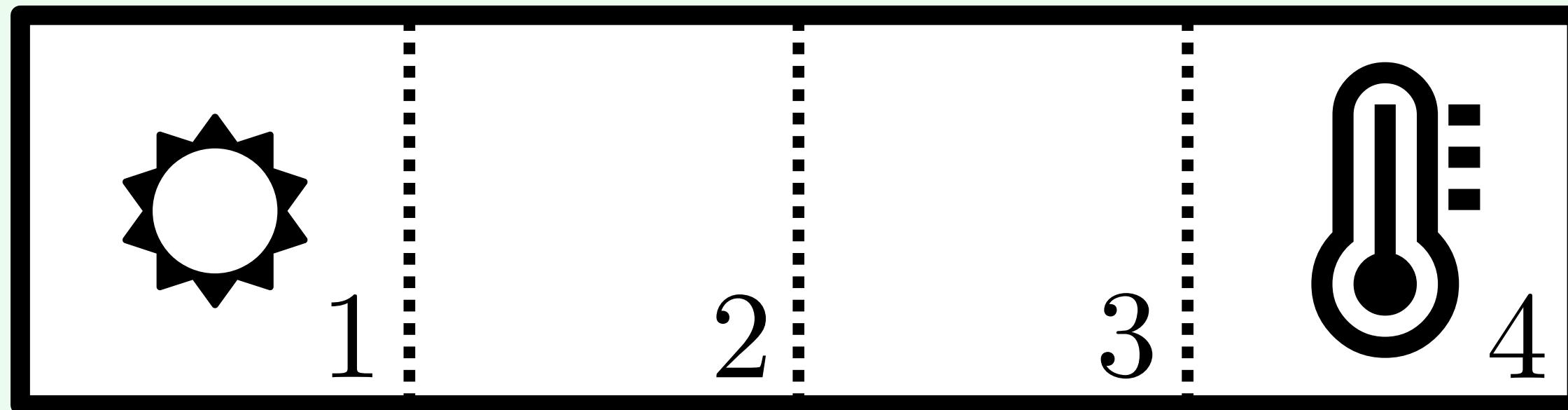
Thus,

$$B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \; AB = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \ldots, A^{n-1}B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

The controllability matrix has full rank and the system is controllable.

🧩 HEAT EQUATION
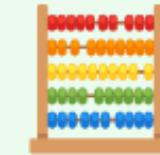
- $dT_1/dt = u + (T_2 - T_1)$
- $dT_2/dt = (T_1 - T_2) + (T_3 - T_2)$
- $dT_3/dt = (T_2 - T_3) + (T_4 - T_3)$
- $dT_4/dt = (T_3 - T_4)$

1. 🧠 🧮

Show that the system is controllable.

2. 🧠 🧮

Assume now that the four cells are organized as a square.

- Is the system still controllable?

- Why?

- How could you solve this problem?

🔓 **HEAT EQUATION**

1. 🔓

We have

$$
A = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.
$$

$$AB = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \; A^2 B = \begin{bmatrix} 2 \\ -3 \\ 1 \\ 0 \end{bmatrix}, \; A^3 B = \begin{bmatrix} -5 \\ 9 \\ -5 \\ 1 \end{bmatrix}.$$

The controllability matrix

$$\begin{bmatrix} 1 & -1 & 2 & -5 \\ 0 & 1 & -3 & 9 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is full rank, thus the system is controllable.

2. 🔓

If the system is organized as a square

$$A = \begin{bmatrix} -2 & 1 & 1 & 0 \\ 1 & -2 & 0 & 1 \\ 1 & 0 & -2 & 1 \\ 0 & 1 & 1 & -2 \end{bmatrix}, \ B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The controllability matrix is

$$
\begin{bmatrix}
1 & -2 & 6 & -20 \\
0 & 1 & -4 & 16 \\
0 & 1 & -4 & 16 \\
0 & 0 & 2 & -12
\end{bmatrix}
$$

It is **not** full rank since the second and the third rows are equal. Thus the system is not controllable.

The lack of controllability is due to symmetry.

If the initial temperature in cell 2 and 3 are equal, since they play symmetric role in the system, their temperature will be equal for any subsequent time.

Hence, it will be impossible to reach a state with different temperature in cell 2 and 3, no matter what the input is.

To break this symmetry and restore controllability, we may for example try to add a second independent heat source sink in cell 2.

This leads to

$$A = \begin{bmatrix} -2 & 1 & 1 & 0 \\ 1 & -2 & 0 & 1 \\ 1 & 0 & -2 & 1 \\ 0 & 1 & 1 & -2 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

and the controllability matrix

$$\begin{bmatrix} 1 & 0 & -2 & 1 & 6 & -4 & -20 & 16 \\ 0 & 1 & 1 & -2 & -4 & 6 & 16 & -20 \\ 0 & 0 & 1 & 0 & -4 & 2 & 16 & -12 \\ 0 & 0 & 0 & 1 & 2 & -4 & -12 & 16 \end{bmatrix}$$

which has a full-rank.