Tale of two cities

Automatic Control

Can we design a steering policy such that the car follows the centerline?

Optimization

minimize or maximize \( \{ \text{objective function} \} \)

subject to constraints on the decision variables.

subject to constraints on the decision variables.

subject to constraints on the decision variables.
In the first city:

dynamics: the variables of interest and evolve over time:

- heading \( \rightarrow h(t) \)
- position \( \rightarrow p(t) \)
- steering \( \rightarrow a(t) \)

-a notion of "desirables"

- a notion of variable to be chosen.

follow the center line

actuation:

- steering wheel?

decision making:

- a human or maybe an algorithm/software

sensing:

- maybe a way to monitor how good we are doing.
you have probably seen other issues that can be relevant for a **dynamic system**:
(a system that has dynamic variables!)

- **Stability**: the ability of the system to stay around an equilibrium if perturbed away from it (this is sometimes called internal stability)
- **BIBO Stable**: if the system accepts inputs & these inputs are bounded, the output should stay bounded

- **Robustness**: the ability of the system to preserve & maintain a certain "quality" even if model/input are uncertain
There is also a notion of performance that shows up in typically time-dependent objectives, such as rise time, settling time, max overshoot, ... these are often directly or indirectly relate to "desirable".

So in a nutshell, control is about:

\[
\begin{align*}
\text{effecting the behavior of} \\
\text{a dynamic system by} \\
\text{manipulating the inputs to} \\
\text{the system (allowable)}
\end{align*}
\]

"Automatic" \(\rightarrow\) via an algorithm/software/hardware as opposed to humans.
Another distinction is the way that this control signal can be generated:

- open loop: pre-computed/no sensing required
- closed loop: online computation on sensed data/feedback

Moreover, the control signal can be

- single channel → just steering
- multi-channel → aileron, spoilers, elevators

In classical control we typically deal with single-channel design (may one at the time)

We will see that we need linear algebra to discuss the multivariable control problems.
So the control signals can be represented as one vector:

\[ U(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad m \text{- channels} \]

In fact you have already seen how we can embed this in the operation of the system:

\[ U(t) \xrightarrow{\text{Dynamic System } x(t)} \]

\[ x(t) = f(x(t), u(t), w(t)) \]

Time evolution of this state

\[ \text{exogenous } W(t) \]

\[ \text{output } y(t) \]

A multi-dimensional variable that captures the "state" of the system, e.g., \( x(t) \in \mathbb{R}^n \) for each \( t \)
or if the system evolves in discrete time \( k=0, 1, 2, \ldots \), then

\[
x(k+1) = f(x(k), u(k), w(k))
\]

we might also add the notion of "output" - what is observed - as \( y(t) \) (continuous time) or \( y(k) \) (discrete time)

So together they can form a dynamic system with inputs and outputs:

\[
\begin{align*}
W(t) & \rightarrow \begin{cases} \dot{x} = f(x, u, w) \\ y = h(x, u, w) \end{cases} \rightarrow y(t) \\
u(t) & \rightarrow \end{cases}
\]

or

\[
\begin{align*}
W(k) & \rightarrow \begin{cases} x(k+1) = f(x(k), u(k), w(k)) \rightarrow y(k) \\
u(k) & \rightarrow \end{cases}
\]

\[
y(k+1) = h(x(k), u(k), w(k)) 
\]
How about city #2: optimization?

Well, at least in principle optimization is a bit more clean to introduce:

\[ \min_{\mathbb{Z}} J(z) \]

Subject to constraints on \( z \)

Since we are minimizing \( J(z) \), \( J(z) \) should belong to a space that is, e.g., real #s, not a vector space often is in a vector space

If this vector space is finite-dimensional we call the optimization problem finite-dimensional optimization or mathematical programming.
If \( z \) lives in an infinite dimensional vector space, we call the optimization problem infinite dimensional. Interestingly, mathematical programming / finite dimensional optimization has more recent origins.

Infinite dimensional optimization goes back to foundations of mathematical physics, e.g., Euler-Lagrange equations, action principle, etc. As we will actually see, explore this connection to control soon — in this case, "\( z \)" belongs to the vector space of functions (of time).
So for our purpose let us say that

optimization

\[ \min f(x) \]

constraints on \( x \)

\( x \in \mathbb{R}^n \)

optimal control

\[ \min J(z(t)) \]

constraints on \( z(t) \)

\( z(t) \) a continuous function

Why are these two problems distinct (we will also explore their similarities!)

It really depends on whether "\( x \)" is static or has a time variation embedded.
Let us say we are at $X(0)$, and we want to go to $X(f)$. We have steps 1, 2, 3, and 4.

If we only think one step ahead, we will go:

$$a \rightarrow d \rightarrow g \rightarrow h \rightarrow z$$

Total cost = 9.

But there is a better route:

$$a \rightarrow b \rightarrow e \rightarrow h \rightarrow i$$

Total cost = 8.
The problem is that an initially "cheap" route puts you in a state that will be costly to come out!

\[ \begin{align*}
1 & \rightarrow 100 \\
1 & \rightarrow 200 \\
10 & \rightarrow 1 \\
& \rightarrow 2 \\
\end{align*} \]

So a greedy approach to sequential decision-making might not be a good idea.

But what is a good idea?

Going back in time!
We start looking at the problem in terms of *tail problems*.

- Tail problem of length 0: we are at $x(f) = i$

- Tail problems of length 1:

  ![Diagram](image)

So if we had happened to be in states $h$ or $f$, there would be no ambiguity about what to do "optimally" and cost-to-go.

- Tail problem of length 2

  ![Diagram](image)
tail problem of length 3

\[ b \xrightarrow{2} c \quad d \xrightarrow{3} e \quad 4 \]

\[ 5 \xrightarrow{1} e \quad 4 \]

\[ 7 \xrightarrow{2} 9 \quad 6 \]

tail problem of length 4

\[ a \xrightarrow{3} b \quad 5 \]

\[ 8 \xrightarrow{1} d \quad 7 \]

so cost-to-go, from 'a' is 8 \iff optimal total cost.

\[ a \rightarrow d \rightarrow e \rightarrow h \rightarrow i \]

\[ a \rightarrow b \rightarrow e \rightarrow h \rightarrow i \]

\[ 1 \rightarrow 3 \rightarrow 2 \rightarrow 2 \]

\[ 2 \rightarrow 1 \rightarrow 2 \rightarrow 2 \]
Principle of Optimality

If \( U(t) \) is optimal over \([t_0, t_f]\)

it has to be optimal on \([t, t_f]\)

for all \( t_0 \leq t \leq t_f \) (starting from \( x(t) \))

we will denote the cost-to-go from state \( x \) as

\[ J(x) \]

optimal cost if we were starting from state \( x \).