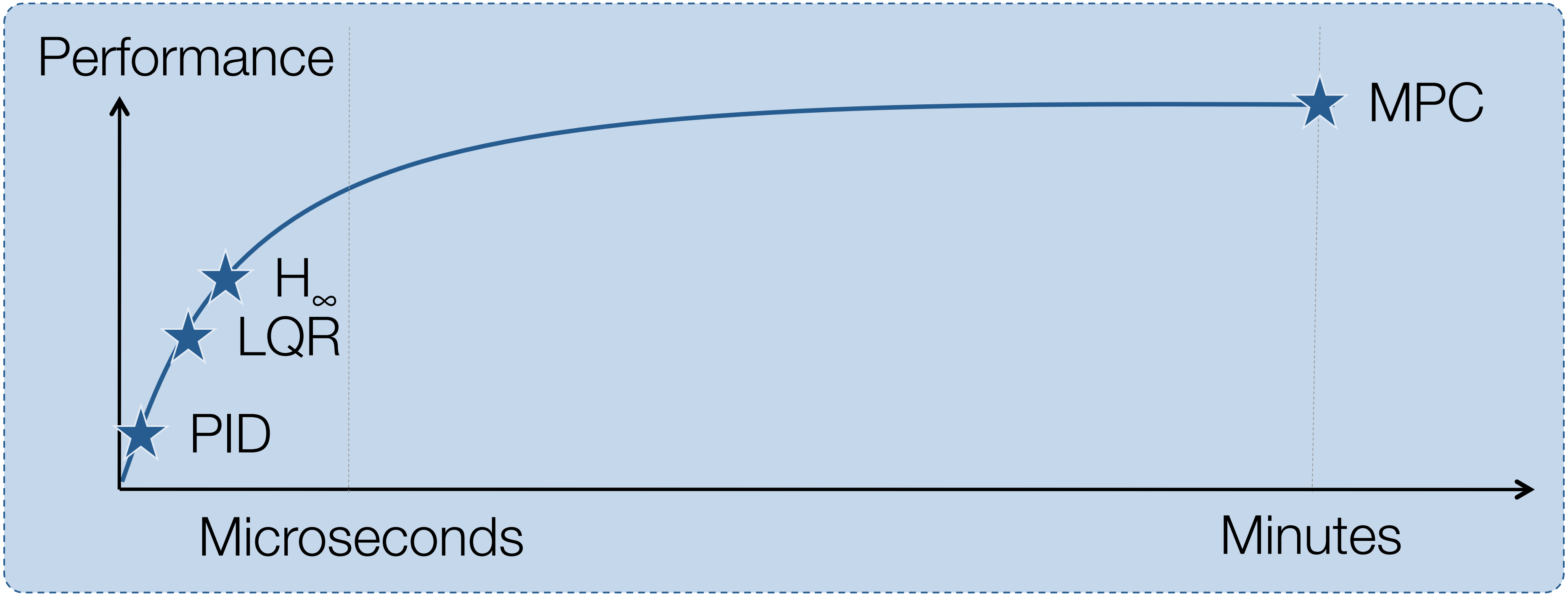
Real-Time Model Predictive Control

Colin Jones and Melanie Zeilinger

Automatic Control Laboratory, EPFL

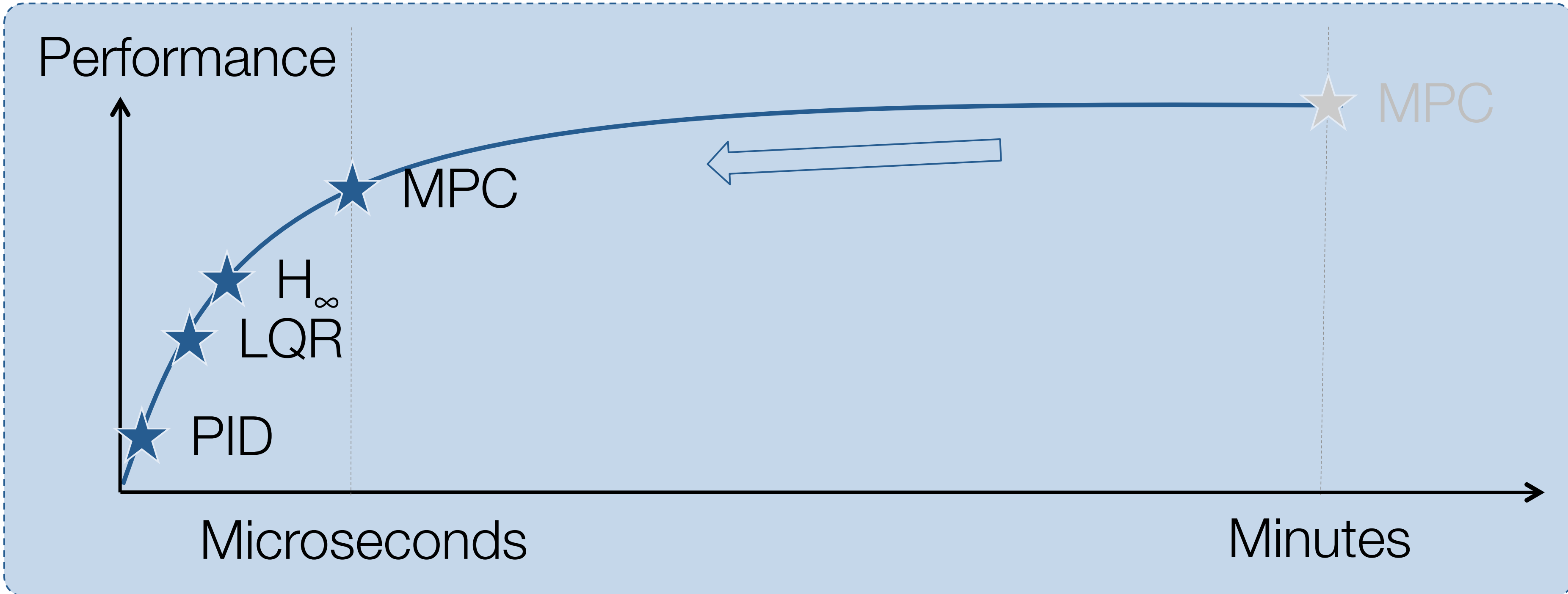
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Model Predictive Control : Computationally Challenged?



- MPC is an optimization-in-the-loop control law
  - Automatic translation from complex specification to controller
    - Reduces design and verification cost; manual synthesis errors
  - Optimizing at every sample => High performance control law
- The Myth : Suitable only for large-scale, high-cost systems

Model Predictive Control : Extreme speed



This workshop:

- Control extremely fast systems on low-cost hardware
- Critical requirement : Fast, real-time optimization

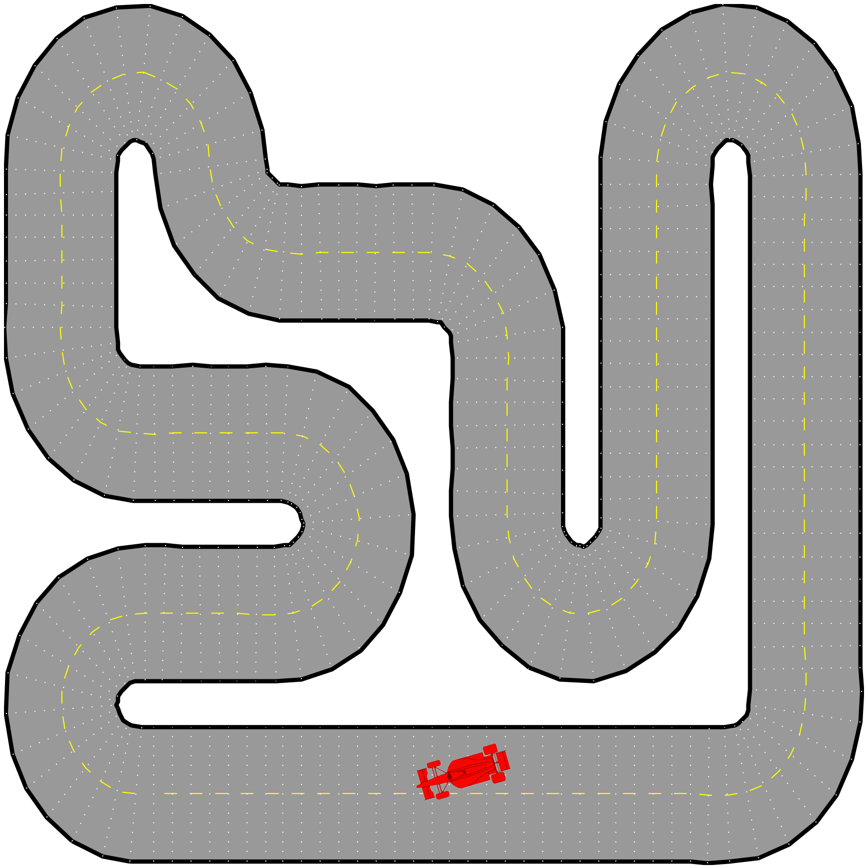
Outline : Introduction

- The intuition
- The math
- Automatic real-time synthesis : The goals and challenges

Optimization-based control: Conceptual Example

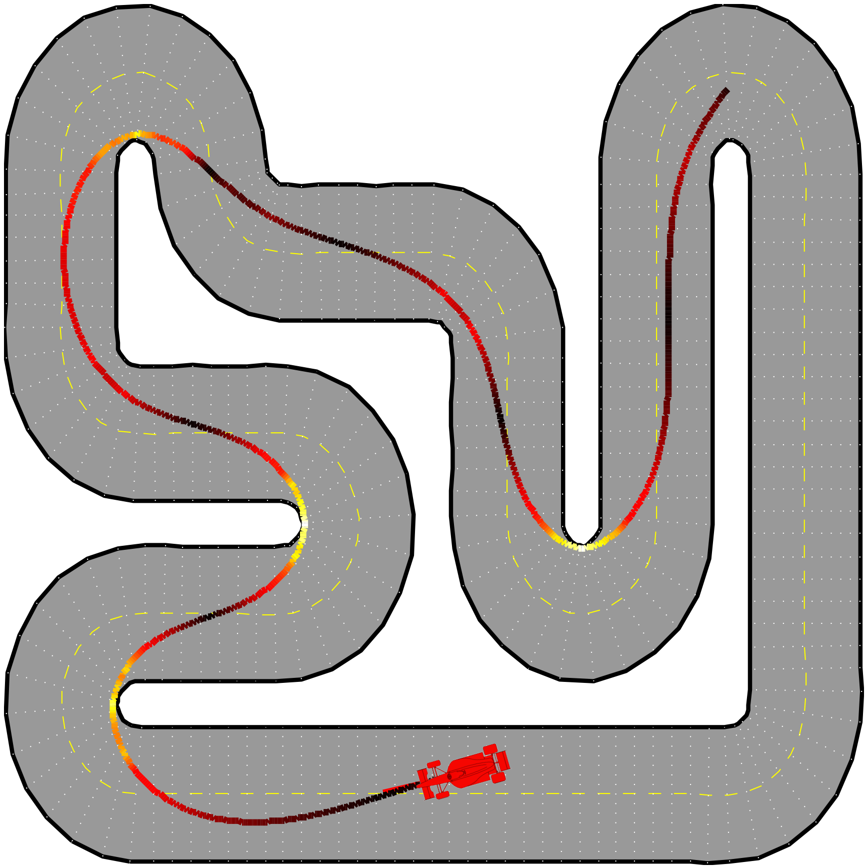
Constraints:
- Stay on road
- Don't skid
- Limited acceleration

Intuitive approach:
- Look forward and plan path based on
  - Road conditions
  - Upcoming corners
  - Abilities of car
  - etc



Optimization-based control: Conceptual Example

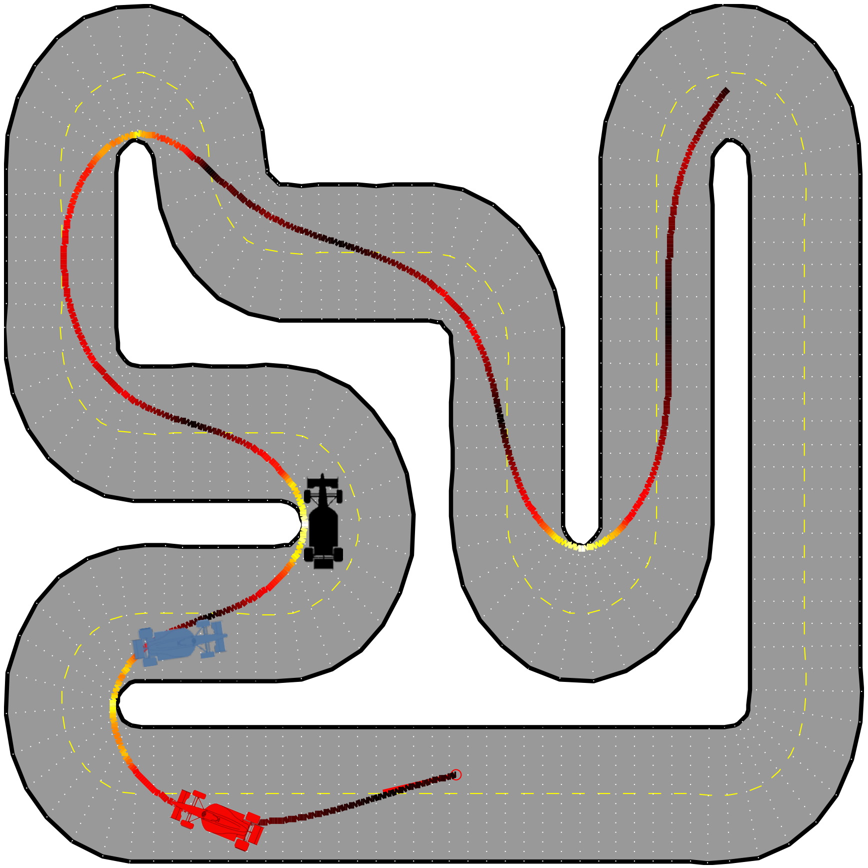
minimize(circuit time)
while avoid other cars
stay on road
…



- Solve optimization problem to compute minimum-time path
- What happens if something unexpected happens?
  - We didn't see a car around the corner!
  - Must introduce *feedback*

Optimization-based control: Conceptual Example

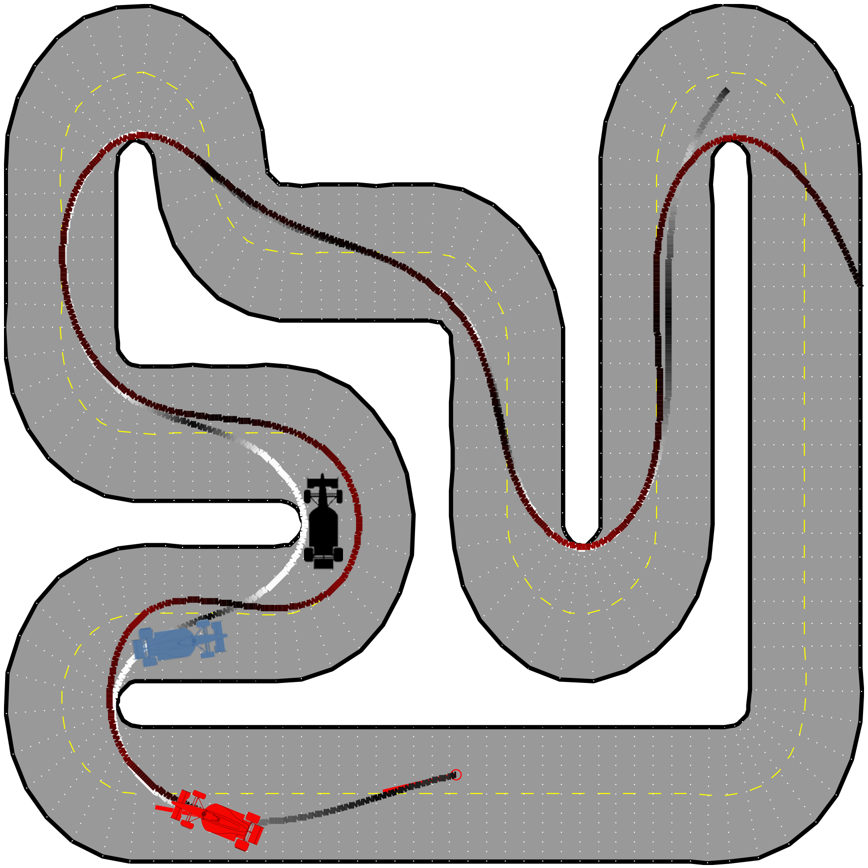
minimize(circuit time)
while avoid other cars
stay on road
…



- Solve optimization problem to compute minimum-time path
- What happens if something unexpected happens?
  - We didn't see a car around the corner!
  - Must introduce *feedback*

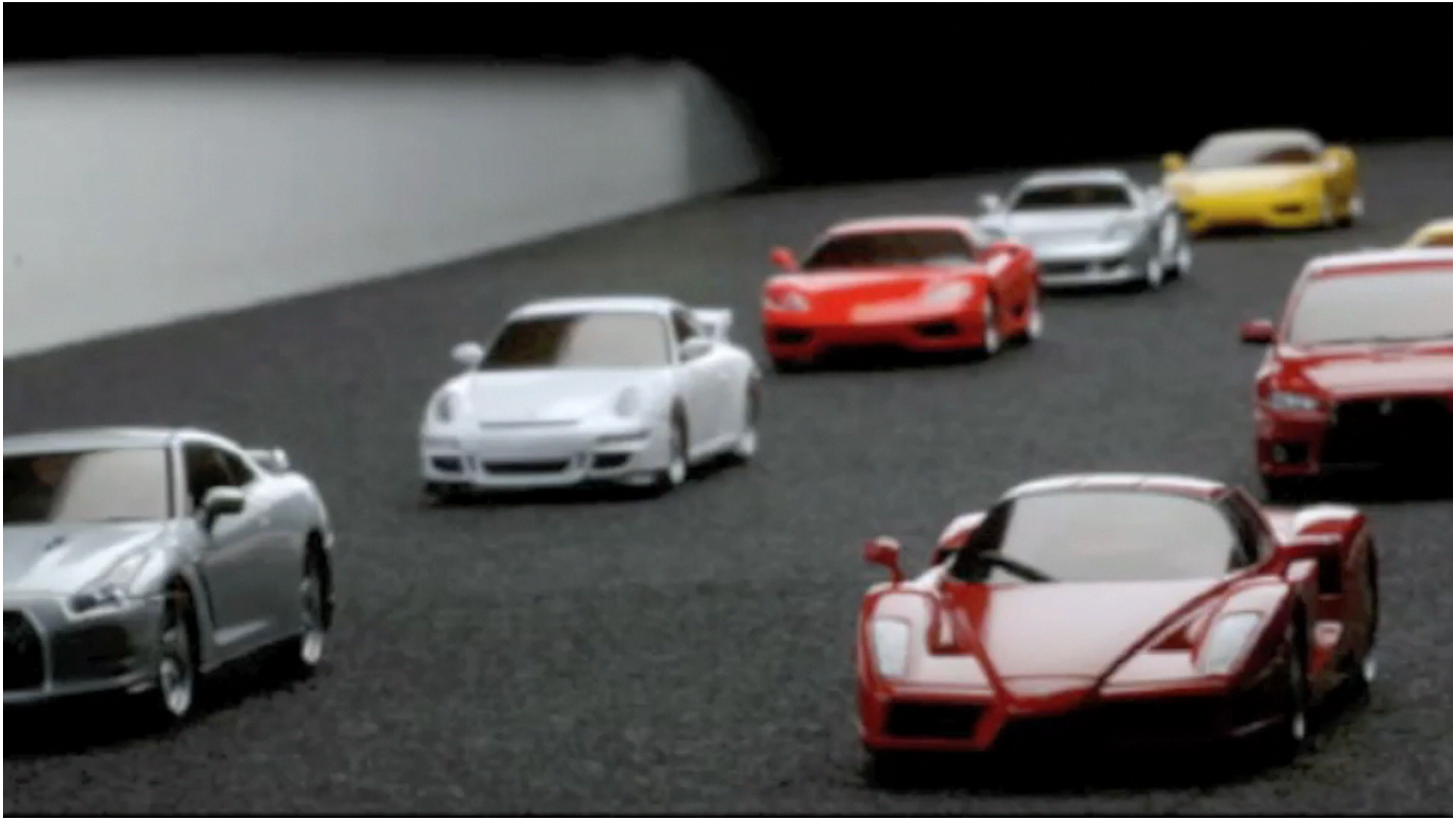
Optimization-based control: Conceptual Example

minimize(circuit time)
while avoid other cars
stay on road
…



- Solve optimization problem to compute minimum-time path
- What happens if something unexpected happens?
  - We didn't see a car around the corner!
  - Must introduce *feedback*

Putting it all together : OrcaRacer



Outline : Introduction

- The intuition
- The math
- Automatic real-time synthesis : The goals and challenges

Receding horizon control : Mathematical formulation

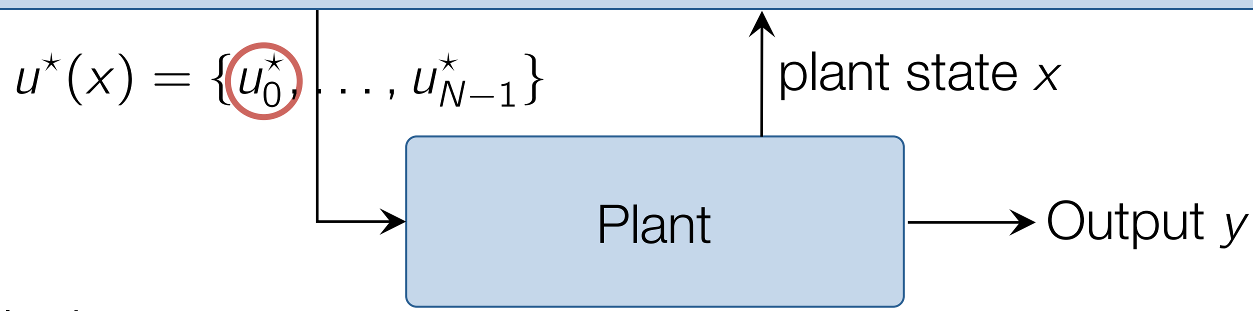
$$
\begin{aligned}
u^\star(x) := \text{argmin} \quad & x_N^T Q_f x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\
\text{s.t.} \quad & x_0 = x && \text{measurement} \\
& x_{i+1} = A x_i + B u_i && \text{system model} \\
& C x_i + D u_i \leq b && \text{constraints} \\
& R \succ 0, Q \succ 0 && \text{performance weights}
\end{aligned}
$$

Cost function measures distance from origin

- Also possible to express much more complex constraints / objectives

Receding horizon control : Mathematical formulation

$$
\begin{aligned}
u^\star(x) := \text{argmin} \quad & x_N^T Q_f x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\
\text{s.t.} \quad & x_0 = x && \text{measurement} \\
& x_{i+1} = A x_i + B u_i && \text{system model} \\
& C x_i + D u_i \leq b && \text{constraints} \\
& R \succ 0, Q \succ 0 && \text{performance weights}
\end{aligned}
$$



Each sample time:

1. Measure / estimate state
2. Solve optimization problem for entire planning window
3. Implement only the *first* control action

Outline : Introduction

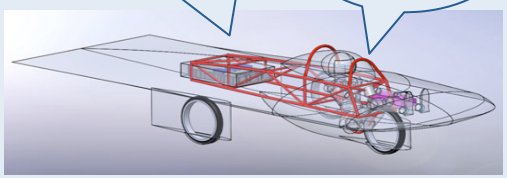
- The intuition
- The math
- Automatic real-time synthesis : The goals and challenges

Control : The Holy Grail

Formal specification

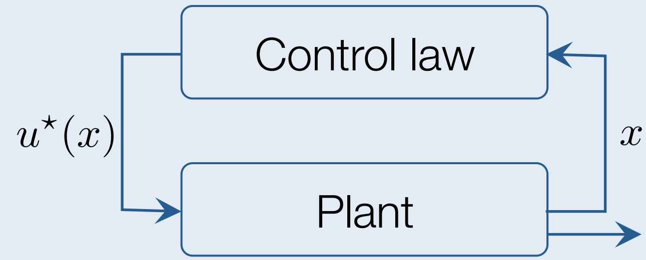
Maximize fuel economy

Don't slip



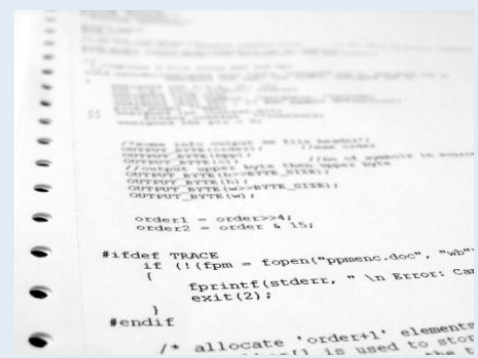
Translate

Control law



Synthesize

Software



Implement

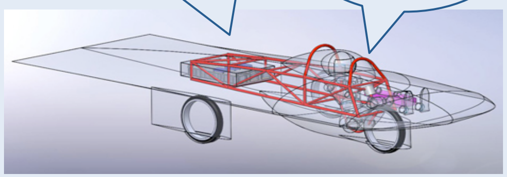
Verified system

The challenge : Real-time constraints

Formal specification

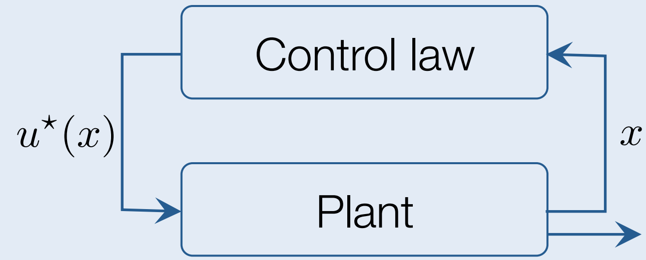
Maximize fuel economy

Don't slip



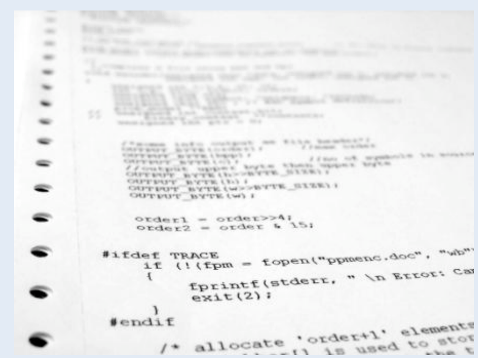
Translate

Control law



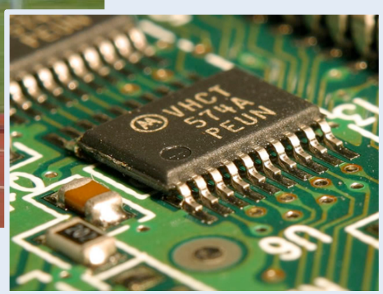
Synthesize

Software



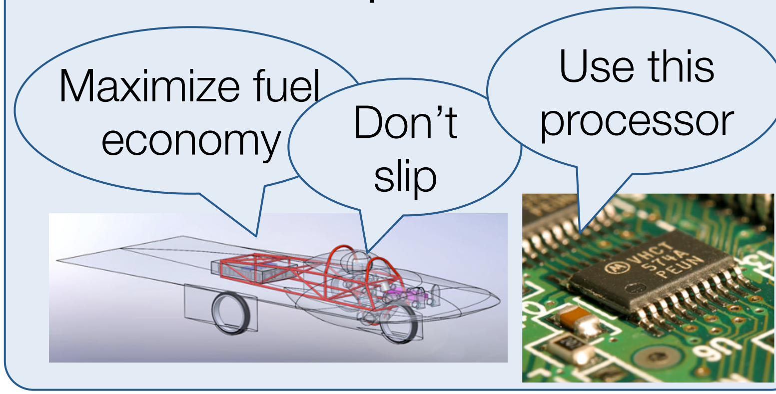
Implement

~~Verified~~ controller



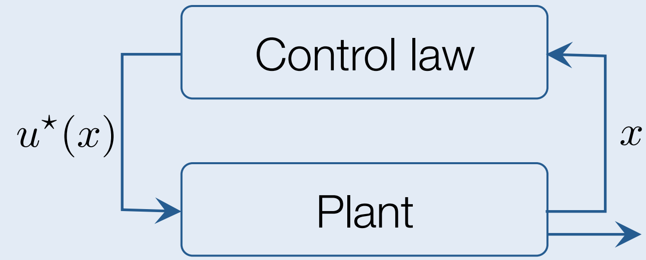
Key principle : Computation is part of the spec

Formal specification



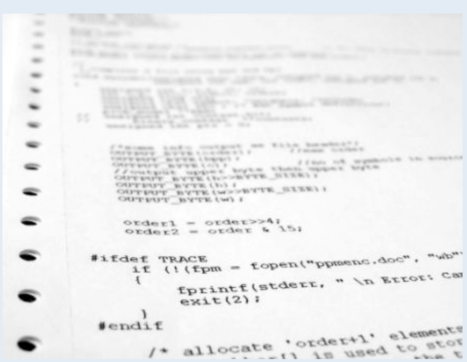
Translate

Control law



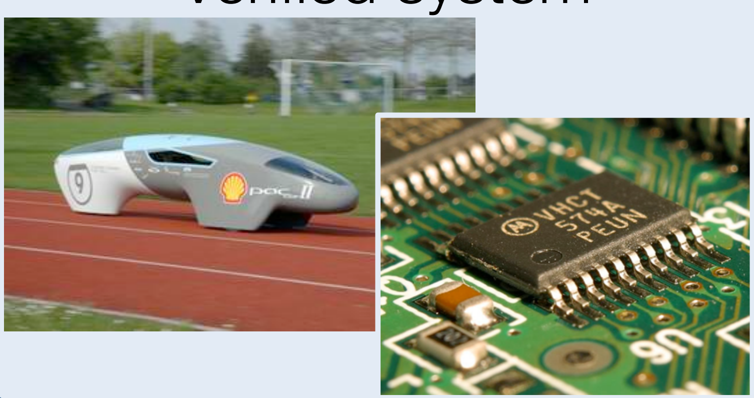
Synthesize

Software



Implement

Verified system



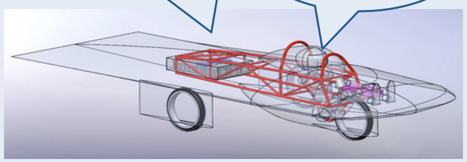
Key principle : Computation is part of the spec

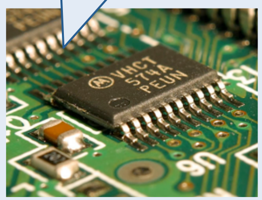
Formal specification

Maximize fuel economy

Don't slip

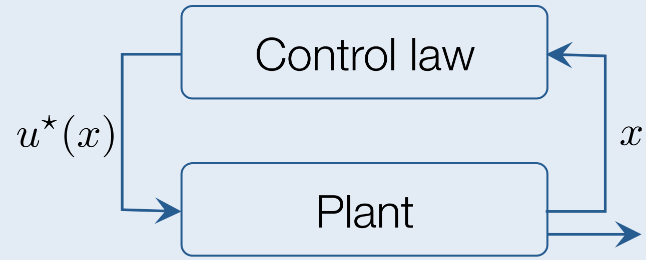
Use this processor





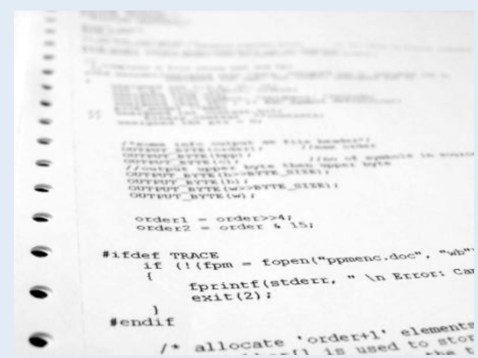
Translate

Control law



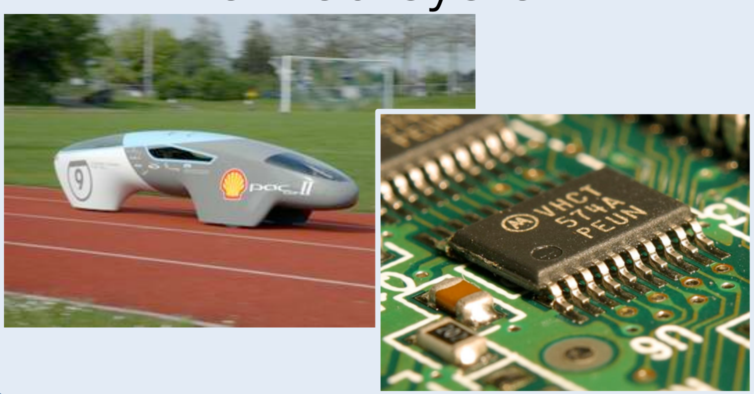
Synthesize

Software



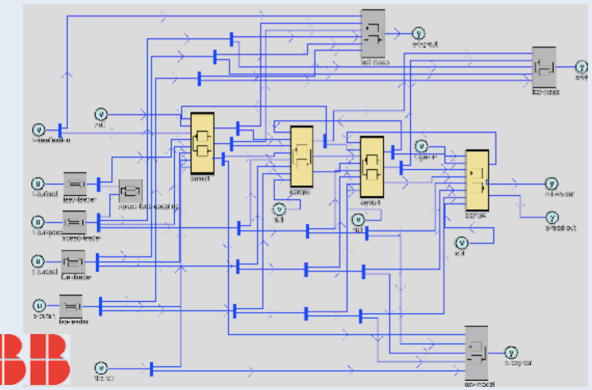
Implement

Verified system



Translate formal spec => MPC control law

Formal specification



Translate

Control law

$$u^\star(x) = \underset{u_i}{\operatorname{argmin}} \sum_{i=0}^{N-1} l(x_i, u_i)$$
$$\text{s.t. } x_{i+1} = f(x_i, u_i)$$
$$(x_i, u_i) \in \mathcal{X} \times \mathcal{U}$$
$$x_0 = x$$

Control problem compilers:

- HYSDEL *Kvasnica, et al*
- MPT *Kvasnica, et al*
- OPTIMICA *Modelon*
- ACADO *Houska, et al*
- …

MPC control law:

- Optimal trajectory subject to system limitations
- Re-optimize at each sample

  $\Rightarrow$ Feedback control

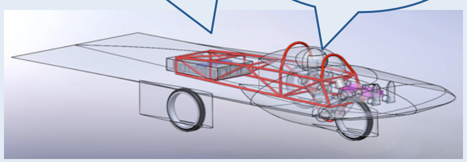
Key principle : Computation is part of the spec

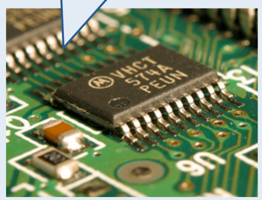
Formal specification

Maximize fuel economy

Don't slip

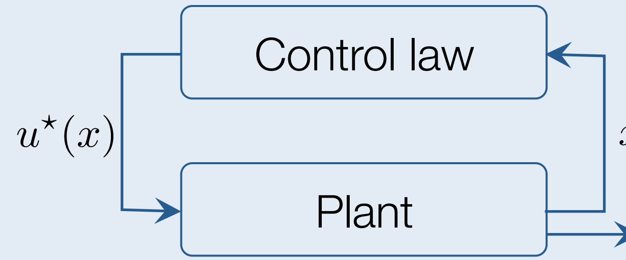
Use this processor





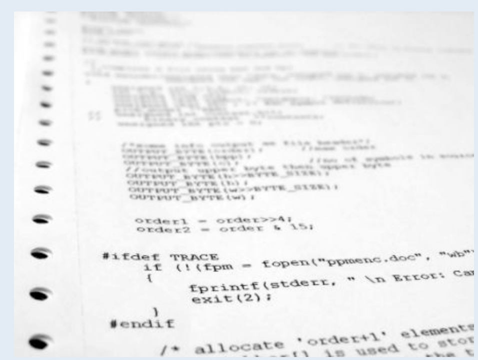
Translate

Control law



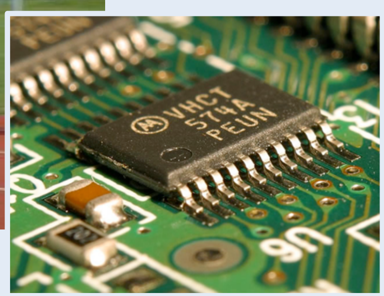
Synthesize

Software



Implement

Verified system



MPC : The fine print

'Classic' MPC has no guarantee of stability or constraint satisfaction

but…

Theory is now reasonably well-developed

- Automatic augmentation of problem for range of systems ensuring
  - (Robust) constraint satisfaction
  - Stability
- Covered in first part of the lecture

$$J^\star(x_0) = \min_{u_i} V_N(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i)$$

$$\text{s.t. } x_{i+1} = f(x_i, u_i)$$

$$(x_i, u_i) \in \mathcal{X} \times \mathcal{U}$$

$$x_N \in \mathcal{X}_N$$

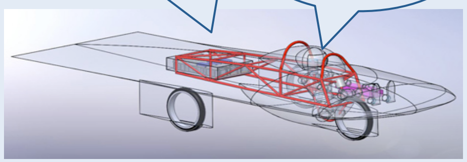
Key principle : Computation is part of the spec

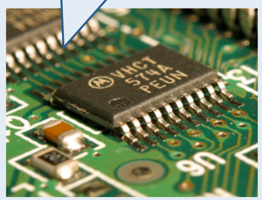
Formal specification

Maximize fuel economy

Don't slip

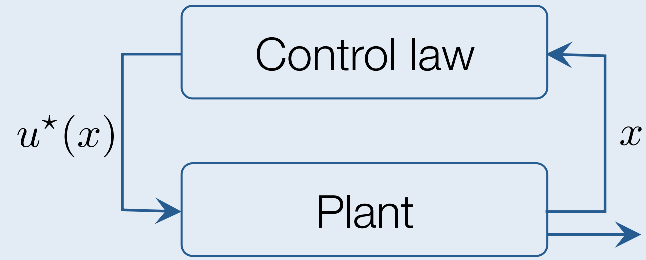
Use this processor





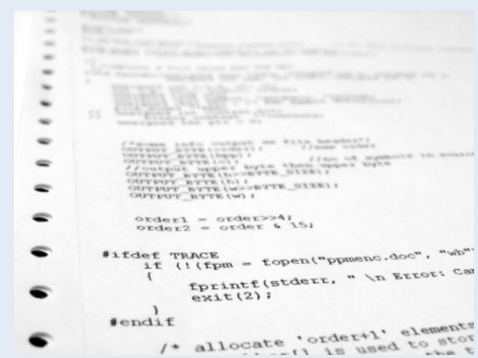
Translate

Control law



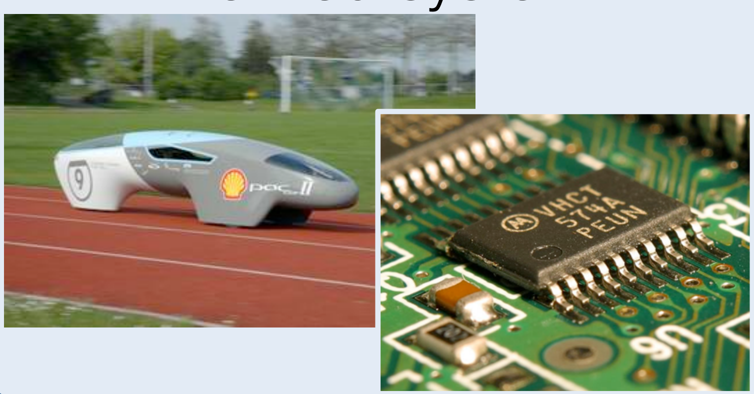
Synthesize

Software

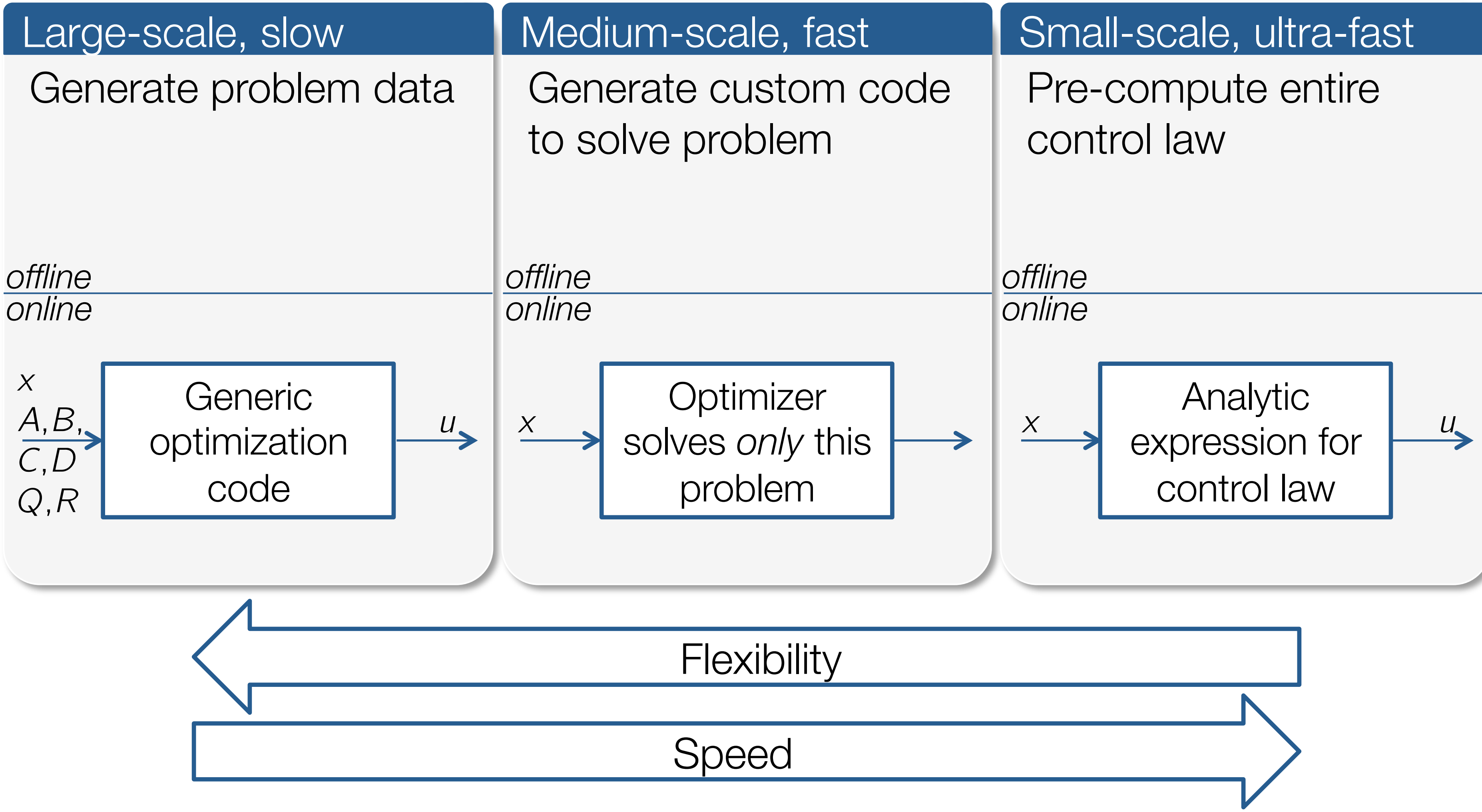


Implement

Verified system



MPC Controller Synthesis



- Ideal approach is problem specific

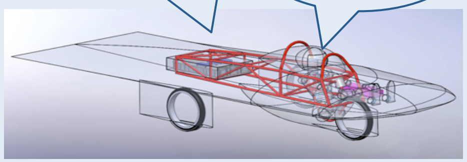
Key principle : Computation is part of the spec

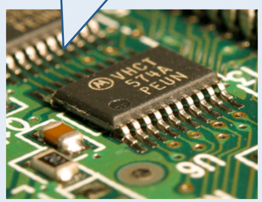
Formal specification

Maximize fuel economy

Don't slip

Use this processor





Translate

Control law

Control law

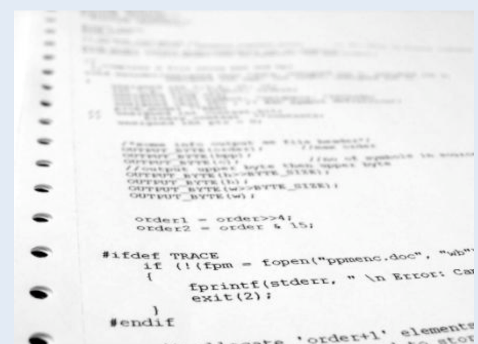
$u^\star(x)$

x

Plant

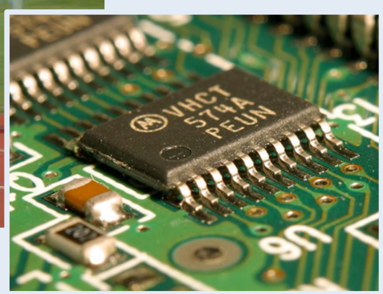
Synthesize

Software



Implement

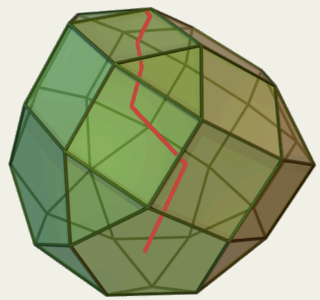
Verified system

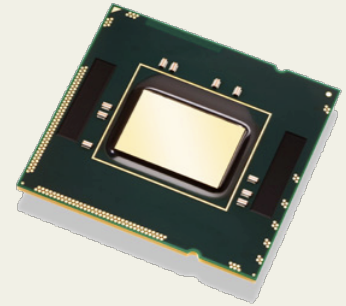


Real-time synthesis : Complexity as a specification

MPC problem + Computational method + Embedded Processor = Real time!

$$J^\star(x_0) = \min_{u_i} V_N(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i)$$
$$\text{s.t. } x_{i+1} = f(x_i, u_i)$$
$$(x_i, u_i) \in \mathcal{X} \times \mathcal{U}$$
$$x_N \in \mathcal{X}_N$$





- Hardware platform bounds computation *time* and *storage*
- Complexity is a function of the problem
  - Uncertain and difficult to estimate and/or bound apriori
- Active area of research : Real-time MPC
  - Sub-optimal, but stabilizing controller in specified time
- 3rd lecture this afternoon