Implementation and Analysis of Offset-Free Explicit Model Predictive Controller on FPGA

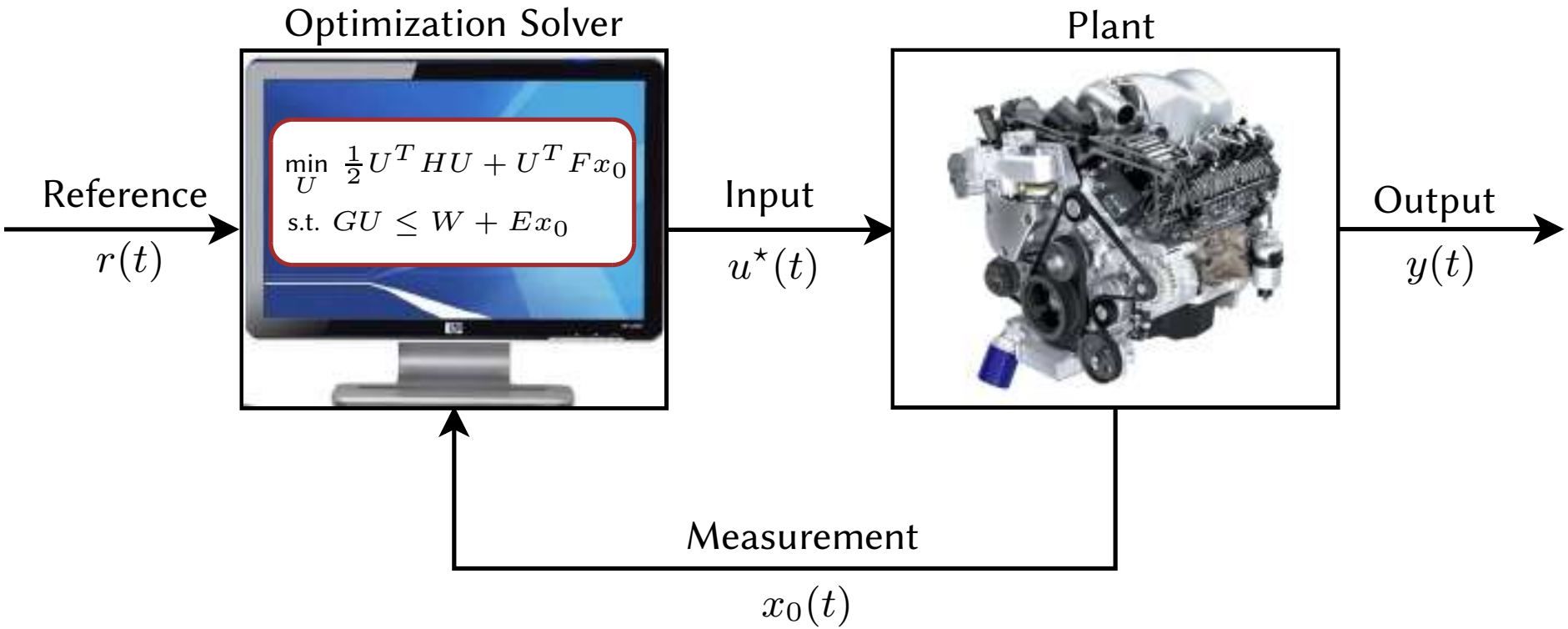
Saket Adhau[1], Kiran Phalke[1], Apeksha Nalawade[1], Deepak Ingole[2], Sayli Patil[1], Dayaram Sonawane[1]

[1]College of Engineering, Pune, India
[2] University of Lyon, IFSTTAR, ENTPE, France

5th Indian Control Conference
IIT Delhi

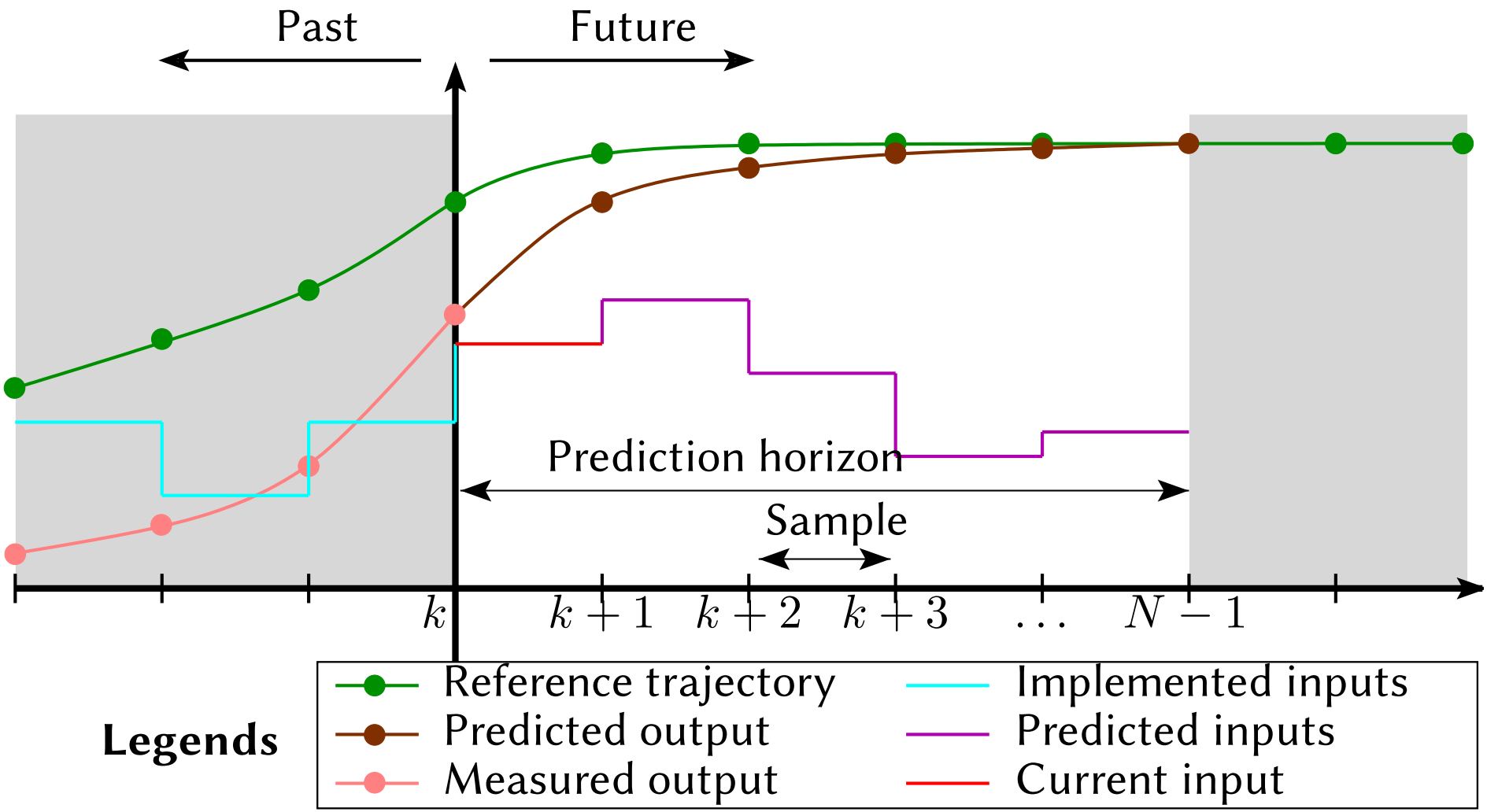
January 10, 2019

Model Predictive Control (MPC)



Use a dynamical model of the process to predict its future evolution and choose the "best" control action

Receding Horizon Implementation



Applications of MPC

Traditional MPC




- Successful in process industries
- Sampling time in minutes
- Powerful computing platforms

Applications of MPC

Traditional MPC




- Successful in process industries
- Sampling time in minutes
- Powerful computing platforms

Embedded MPC




- Successful in embedded world
- Sampling time in ms to ns
- Limited embedded platforms

Embedded Control Systems: Main Requirements

- Model-based, optimal performance, constraints

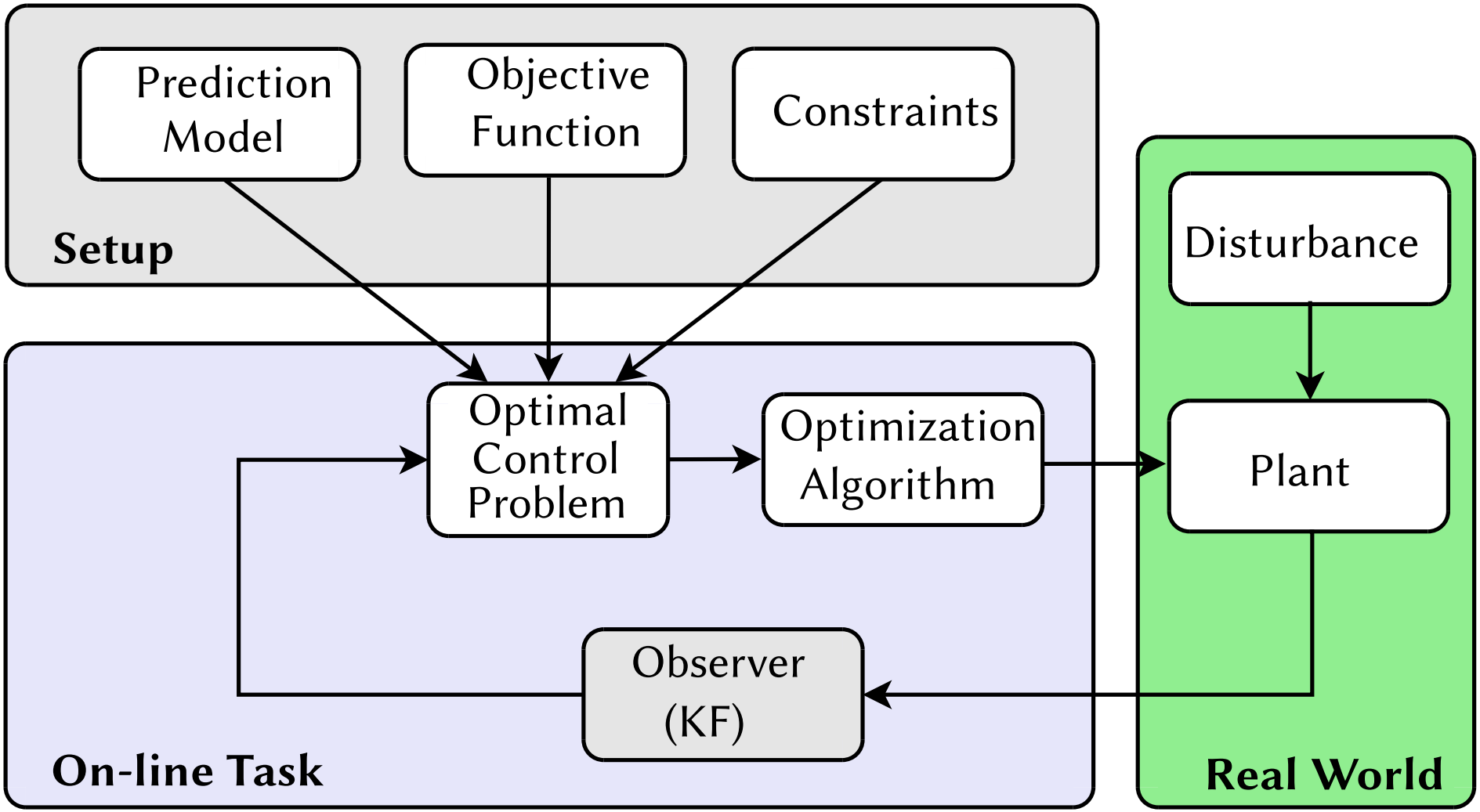
- Control law simple enough for software certification

- Require simple hardware (e.g., cheap and fast FPGA)

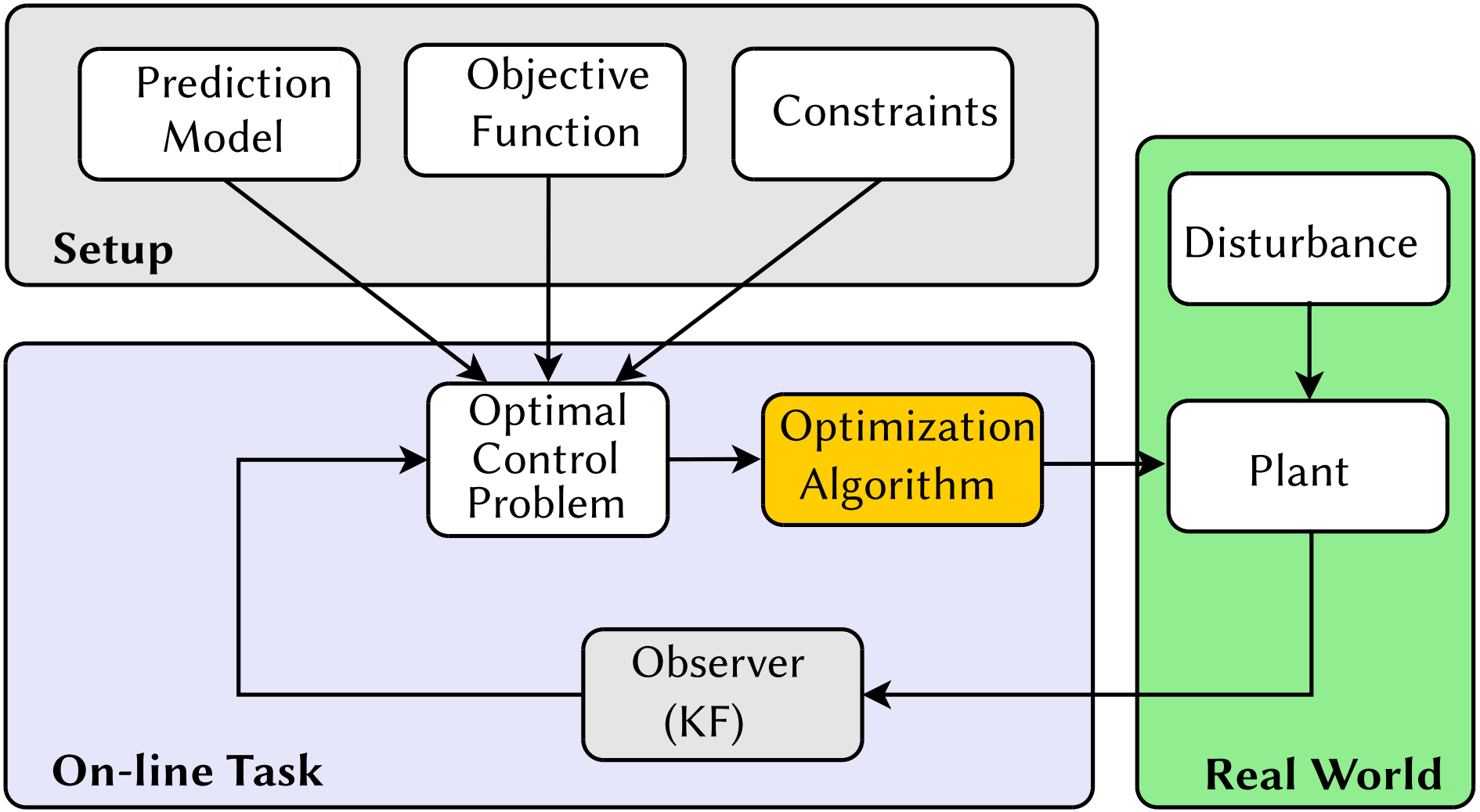
- Speed: fast dynamics $\Rightarrow$short sample time ($< 1\ \mu s$)

- Well defined worst-case execution time, to certify hard real-time system properties

Ingredients of MPC



Ingredients of MPC



Embedded Linear MPC

- Linear MPC requires solving a Quadratic Program (QP)

$$\min_{U} \left\{ \frac{1}{2} U^T H U + x_0{}^T F U \right\} + \cancel{\frac{1}{2} x_0^T Y x_0} \qquad U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\text{s.t. } GU \leq W + Ex_0$$

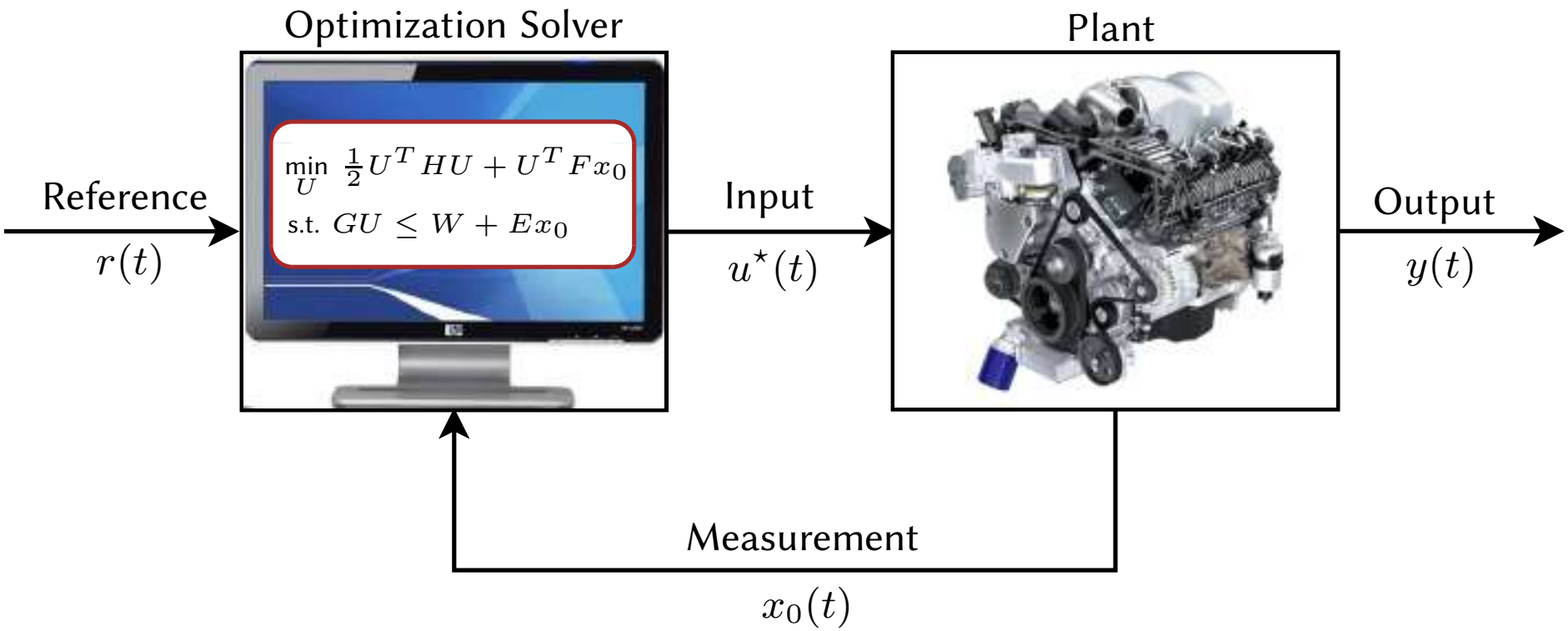
The control law $u(t) = u_0^\star(x_0(t))$ is implicitly defined by the QP

- A variety of algorithms exist to solve the QP on-line given $x_0(t)$
- Explicit MPC: pre-solve the QP off-line for all $x_0(t)$ to find the control law $u_0^\star(x_0(t))$ explicitly via **multi-parametric quadratic programming (mpQP)**

Approach to Solve MPC Problem: On-line

$$\min_{U} \ \frac{1}{2} U^T H U + U^T F x_0$$
$$\text{s.t. } GU \leq W + E x_0$$

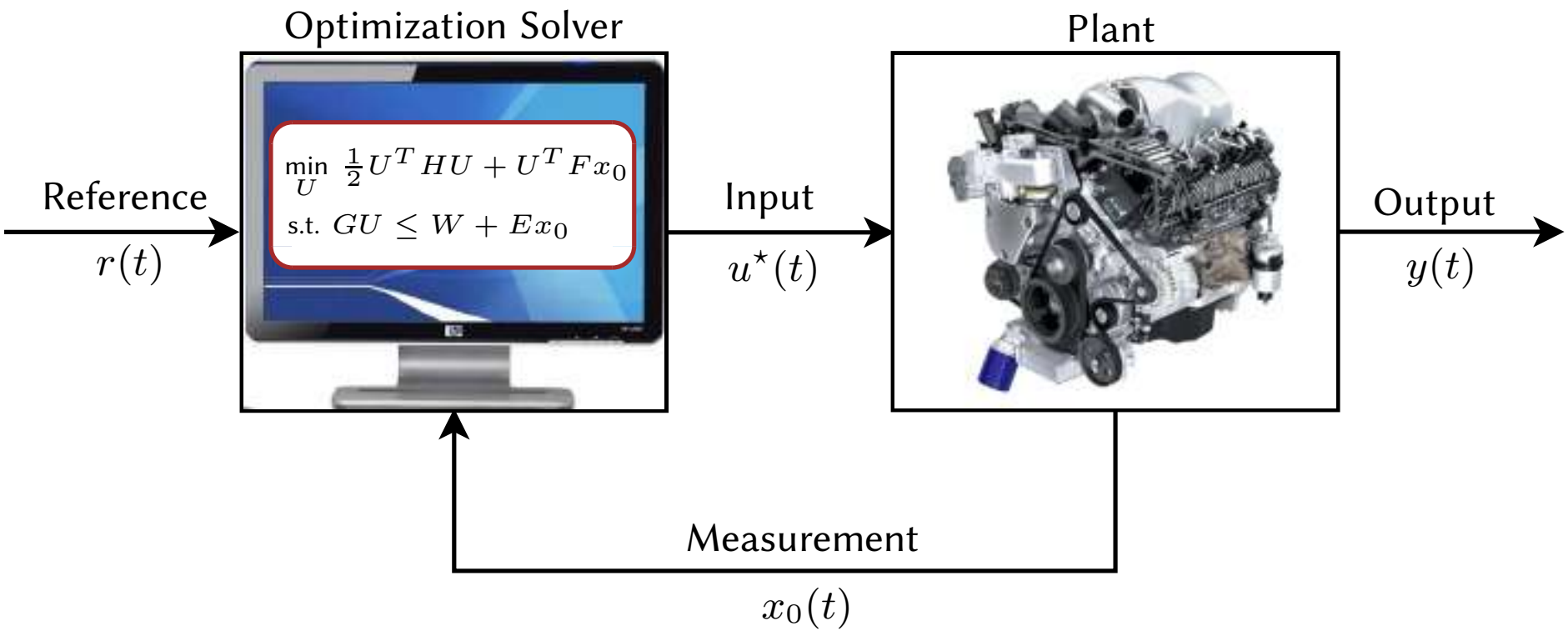
Approach to Solve MPC Problem: On-line



Approach to Solve MPC Problem:Off-line

off-line

on-line



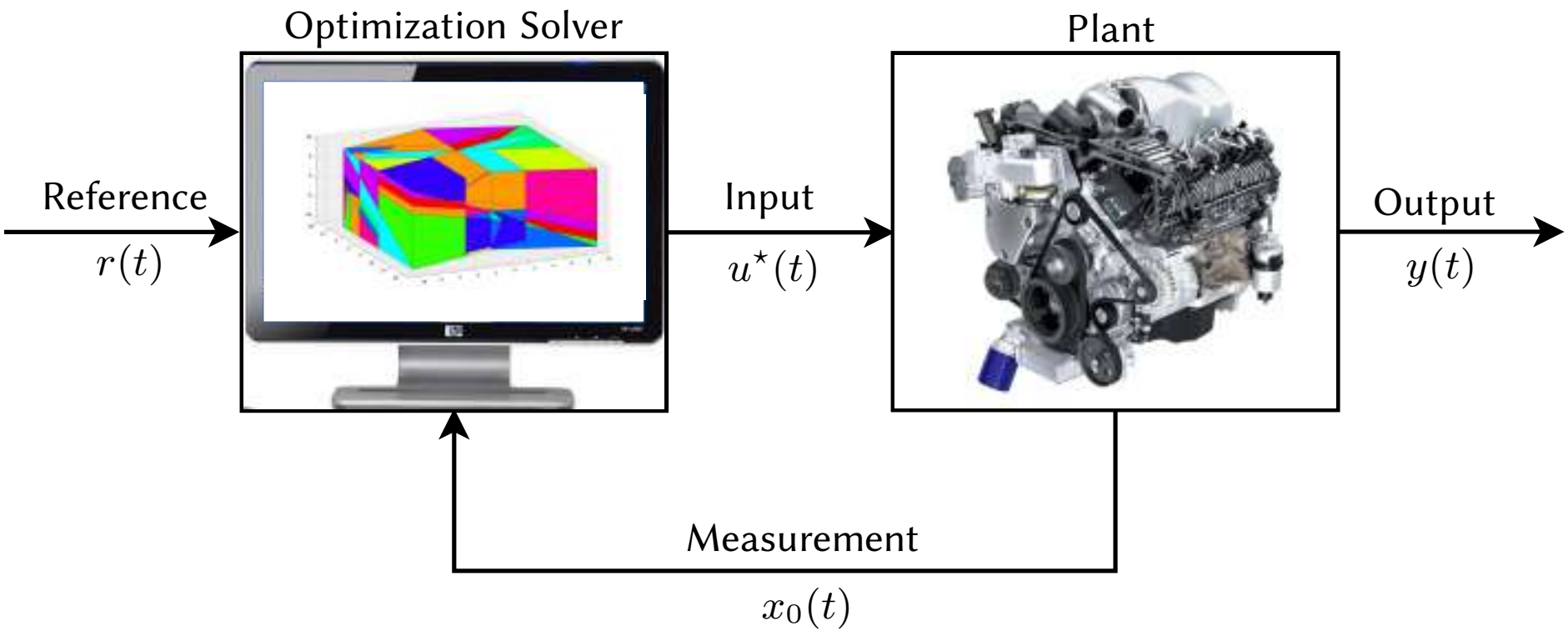
Approach to Solve MPC Problem:Off-line

$$\min_{U} \ \frac{1}{2} U^T H U + U^T F x_0$$
$$\text{s.t. } GU \leq W + Ex_0$$

off-line

---

on-line



Main Drawbacks of On-line Implementation

- ☺ Excellent QP solvers available today

but ...

- ☹ Computation time may be too long
- ☹ Requires relatively expensive hardware
- ☹ Difficult to certify solver code
- ☹ Not suitable of safety critical applications

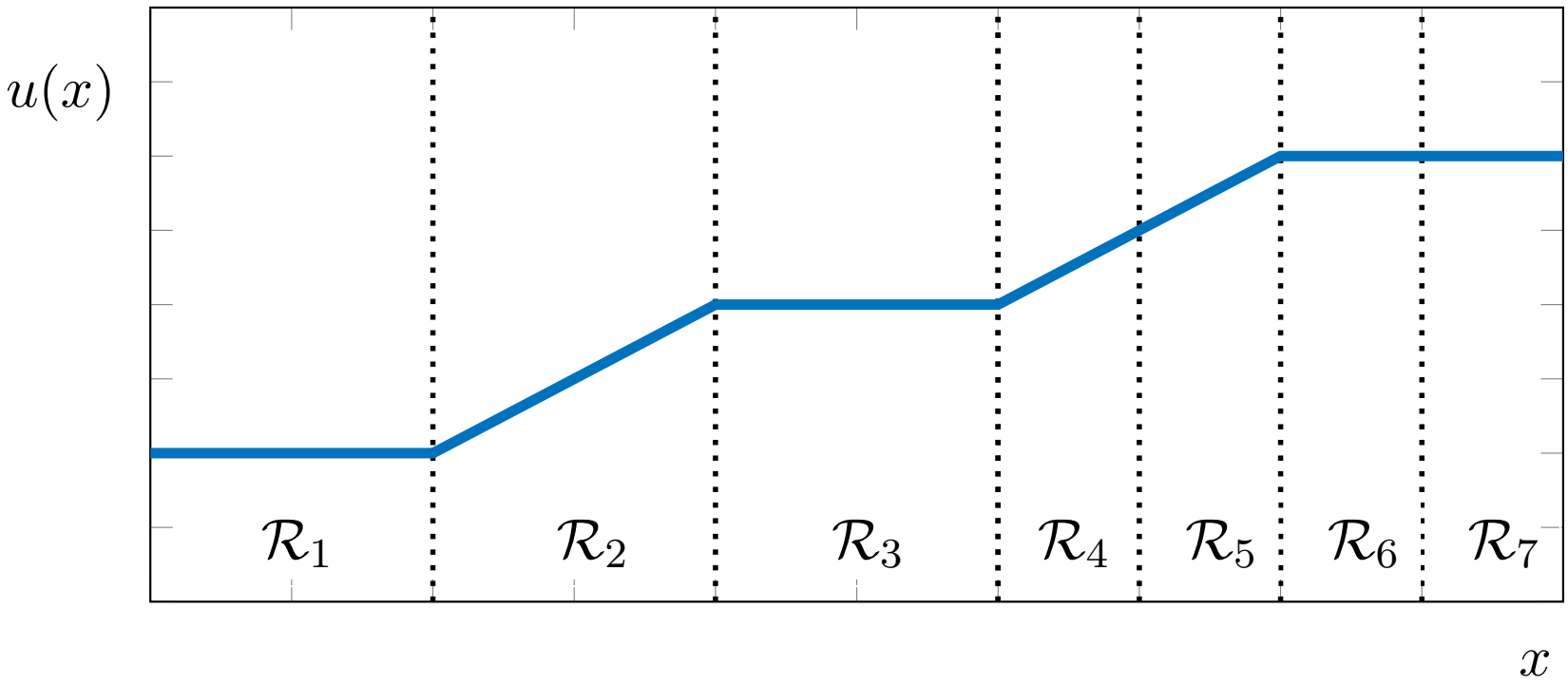
Explicit MPC Control Law

- State-space is divided into polytopic regions
- Control law is a continuous PWA function of states

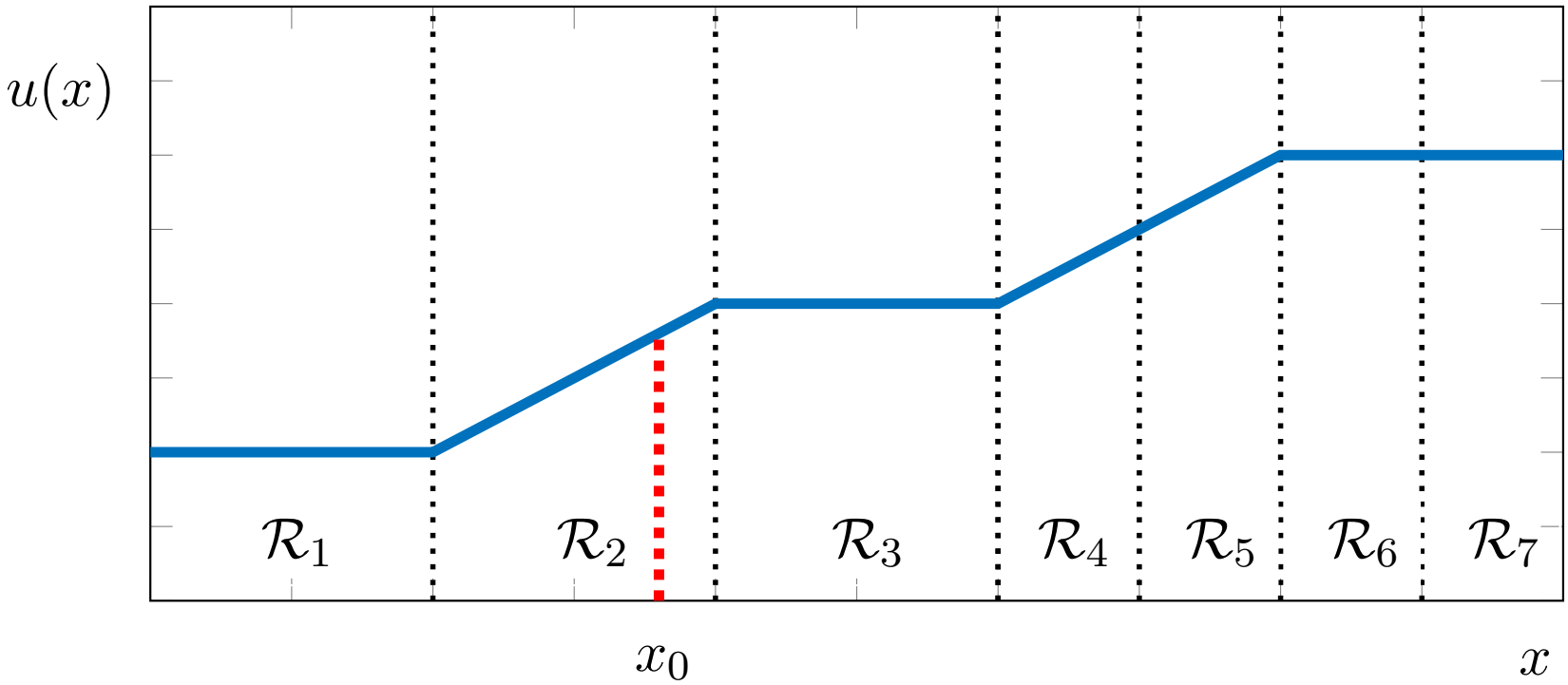
$$\kappa(\tilde{x}_0) = \begin{cases} F_1 x_0 + g_1 & \text{if } x_0 \in \mathcal{R}_1 \\ \vdots & \\ F_M x_0 + g_M & \text{if } x_0 \in \mathcal{R}_M \end{cases}$$

$$\mathcal{R}_i = \{x \in \mathbb{R}^{(n+p)} \mid Z_i x \leq z_i\} \;\; \forall\, i = 1, \ldots, M$$

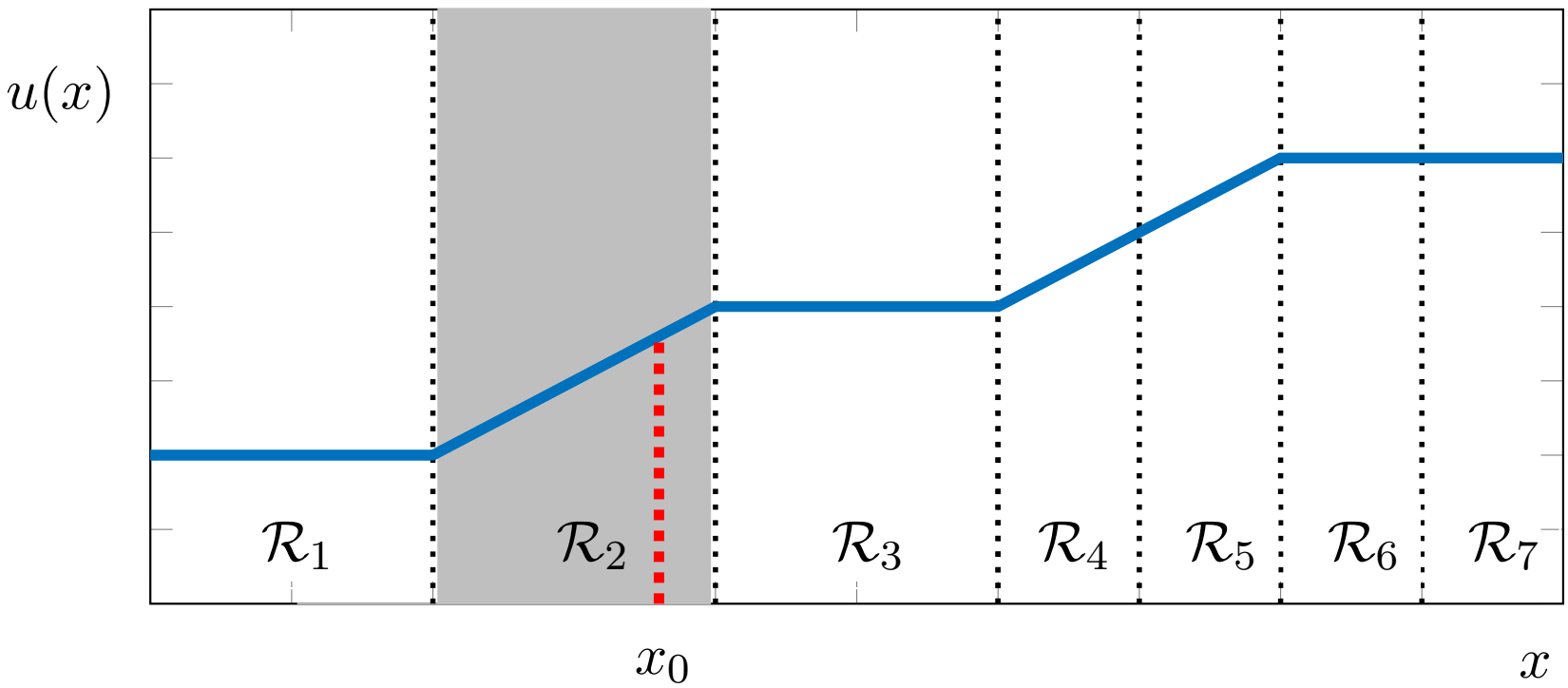
Point Location Algorithm: Sequential Search



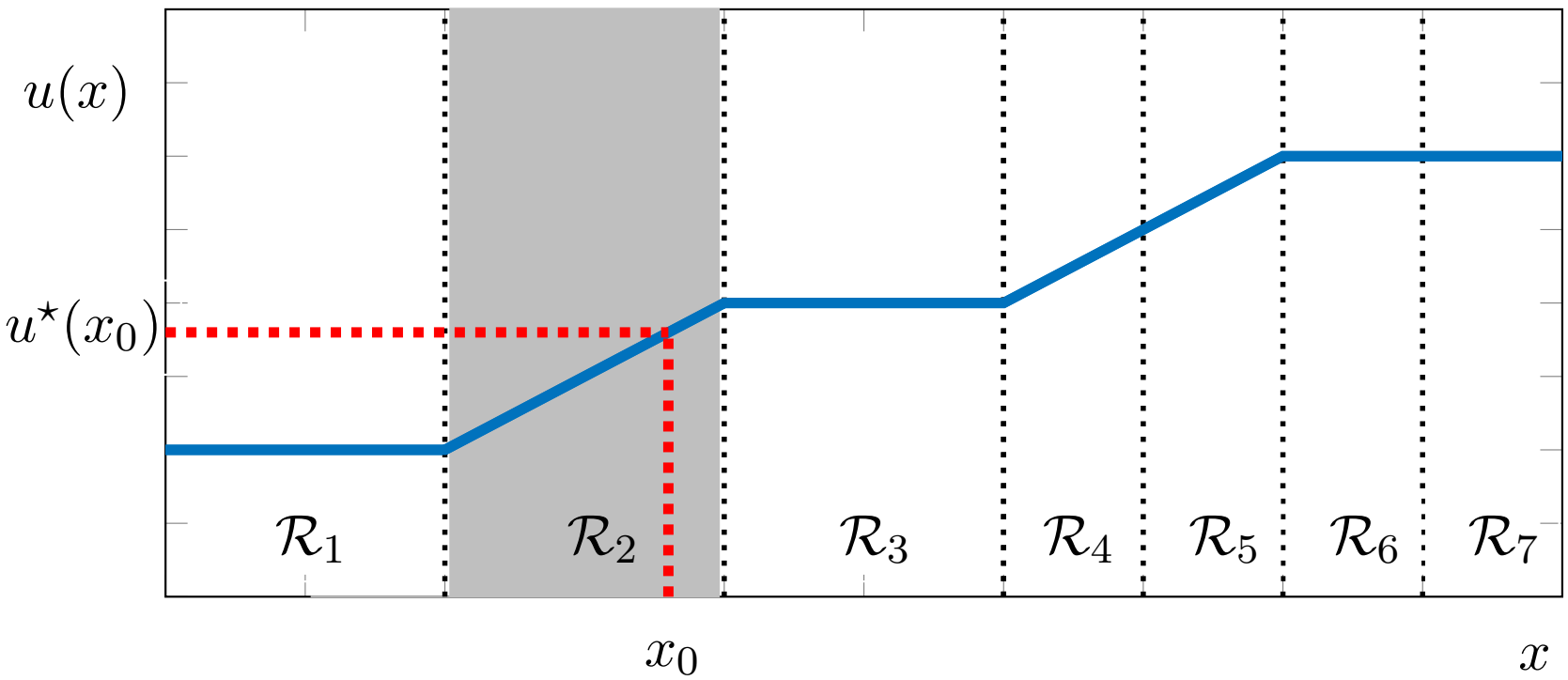
Point Location Algorithm: Sequential Search



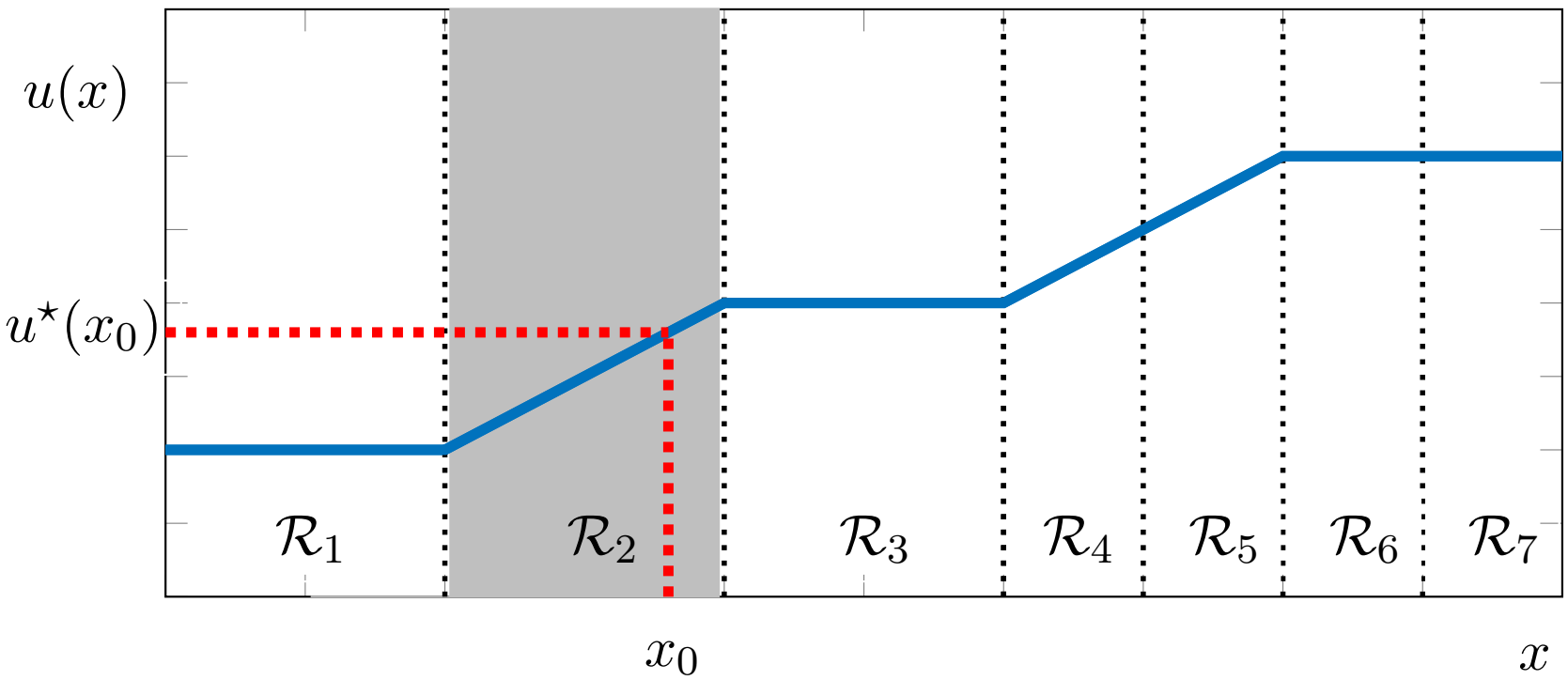
Point Location Algorithm: Sequential Search



Point Location Algorithm: Sequential Search



Point Location Algorithm: Sequential Search



Explicit MPC : Pros and Cons

- Pros :
  - ☺ Easy to implement
  - ☺ Fast on-line evaluation
  - ☺ Division free code
  - ☺ Possible to analyze implementation issues
  - ☺ Tools: Multi-Parametric Toolbox (MPT)
- Cons :
  - ☹ Only for small scale problems
  - ☹ Explicit solutions can be very complex
  - ☹ Reducing complexity requires scarifying performance

Implementation of a Real-time Explicit MPC

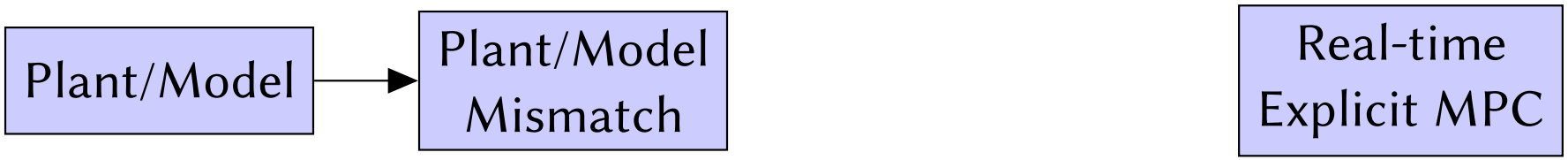
Real-time
Explicit MPC

Implementation of a Real-time Explicit MPC

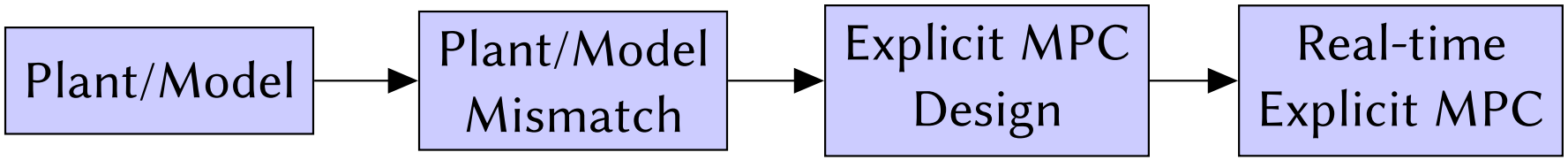
Plant/Model

Real-time
Explicit MPC

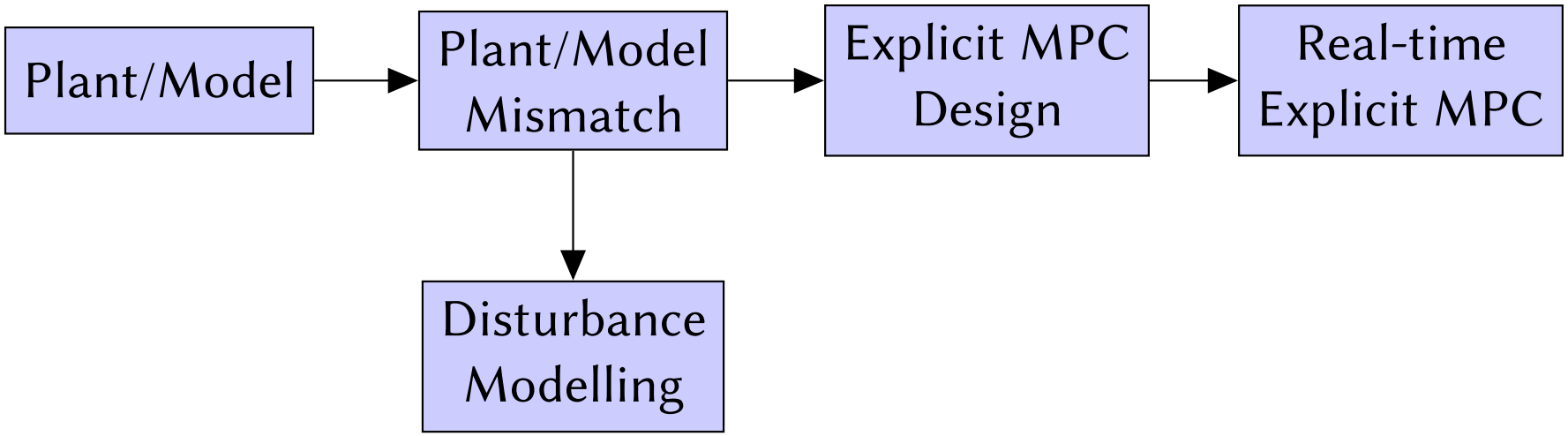
Implementation of a Real-time Explicit MPC



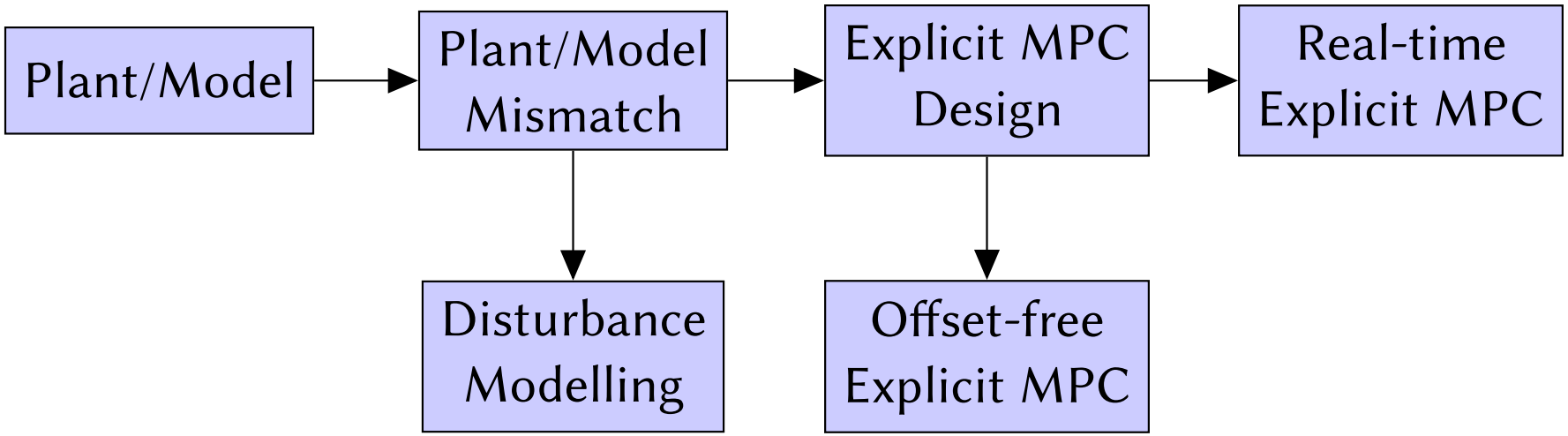
Implementation of a Real-time Explicit MPC



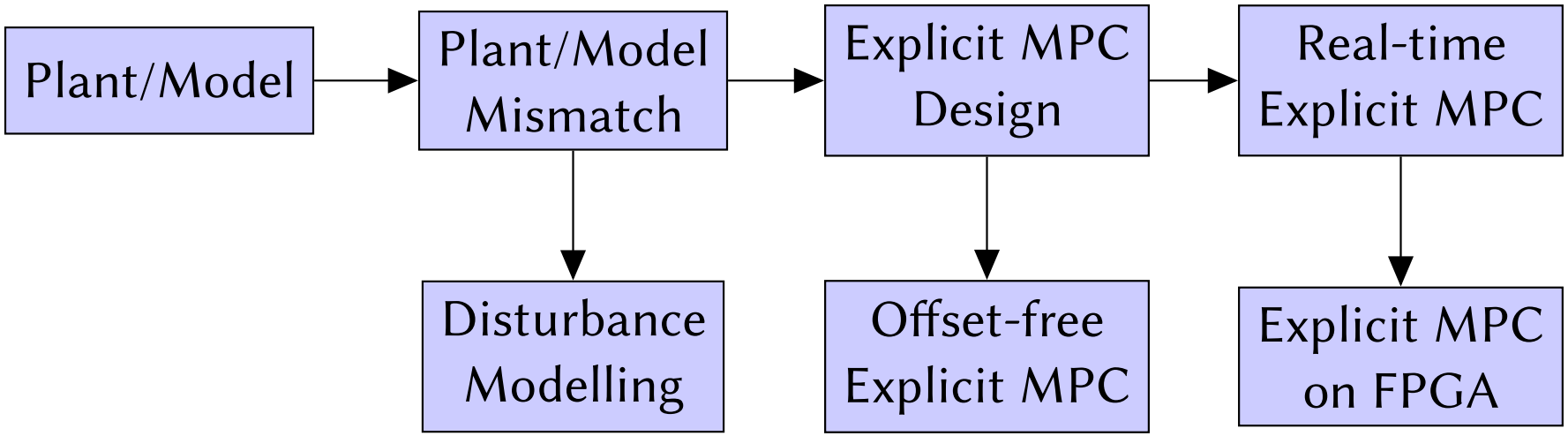
Implementation of a Real-time Explicit MPC



Implementation of a Real-time Explicit MPC



Implementation of a Real-time Explicit MPC

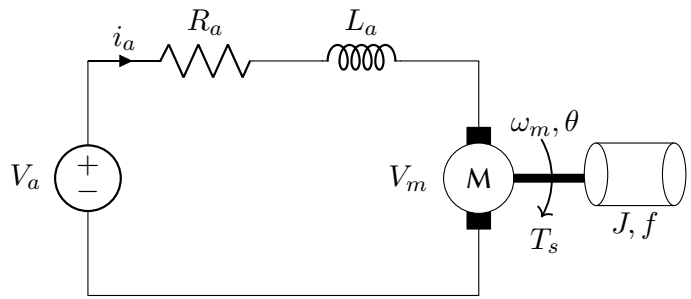


Motor Model

- LTI motor model: 3 state, 1 input

$$\underbrace{\begin{bmatrix} \dot{i_a}(t) \\ \dot{\omega}(t) \\ \dot{\theta}(t) \end{bmatrix}}_{x(t+T_s)} = \underbrace{\begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_b}{L_a} & 0 \\ \frac{K_t}{J} & -\frac{f}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} i_a(t) \\ \omega(t) \\ \theta(t) \end{bmatrix}}_{x(t)} + \underbrace{\begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \end{bmatrix}}_{B} \underbrace{Va(t)}_{u(t)}$$

$$y(t) = \underbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} i_a(t) \\ \omega(t) \\ \theta(t) \end{bmatrix}}_{x(t)}$$



Disturbance Modeling

- Augmented design model

$$x(t+T_s) = Ax(t) + Bu(t)$$
$$d(t+T_s) = d(t)$$
$$y(t) = Cx(t) + d(t)$$

$$\underbrace{\begin{bmatrix} x(t+T_s) \\ d(t+T_s) \end{bmatrix}}_{x_e(t+T_s)} = \underbrace{\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}}_{A_e} \underbrace{\begin{bmatrix} x(t) \\ d(t) \end{bmatrix}}_{x_e(t)} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{B_e} u(t)$$

$$y_e(t) = \underbrace{\begin{bmatrix} C & C_d \end{bmatrix}}_{C_e} \underbrace{\begin{bmatrix} x(t) \\ d(t) \end{bmatrix}}_{x_e(t)}$$

- Luenberger observer

$$\hat{x}_e(t+T_s) = A_e\hat{x}_e(t) + B_eu(t) + L_e(y(t) - \hat{y}_e(t))$$
$$\hat{y}_e(t) = C_e\hat{x}_e(t)$$

Explicit MPC Set-up

$$\min_{U} \sum_{k=0}^{N-1} (y_k - r_k)^T Q (y_k - r_k) + \Delta u_k^T R \Delta u_k$$

$$\begin{aligned}
\text{s.t.} \quad x_{k+1} &= Ax_k + Bu_k \\
d_{k+1} &= d_k \\
y_k &= Cx_k + C_d d_k \\
\Delta u_k &= u_k - u_{k-1} \\
u_{\min} &\leq u_k \leq u_{\max} \\
u_{-1} &= u(t - T_s) \\
x_0 &= \hat{x}_e(t) \\
d_0 &= \hat{d}(t) \\
\forall k &\in \{k = 0, \ldots, N-1\}
\end{aligned}$$

Generation of Explicit Solution

1. Construct the online MPC controller object using the open source **Multi-parametric Toolbox (MPT)**[1]

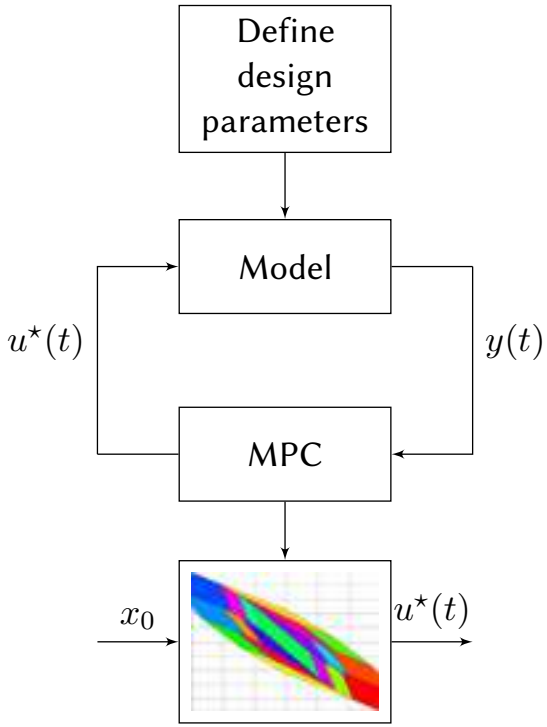
```
ctrl = MPCController(model, N)
u = ctrl.evaluate(x)
```

2. Tune the controller and close the loop

```
loop = ClosedLoop(ctrl, model)
data = loop.simulate(x0, Nsim)
```

3. Export the explicit form in terms of C code

```
expl_ctrl = ctrl.toExplicit()
```



[1] https://www.mpt3.org/

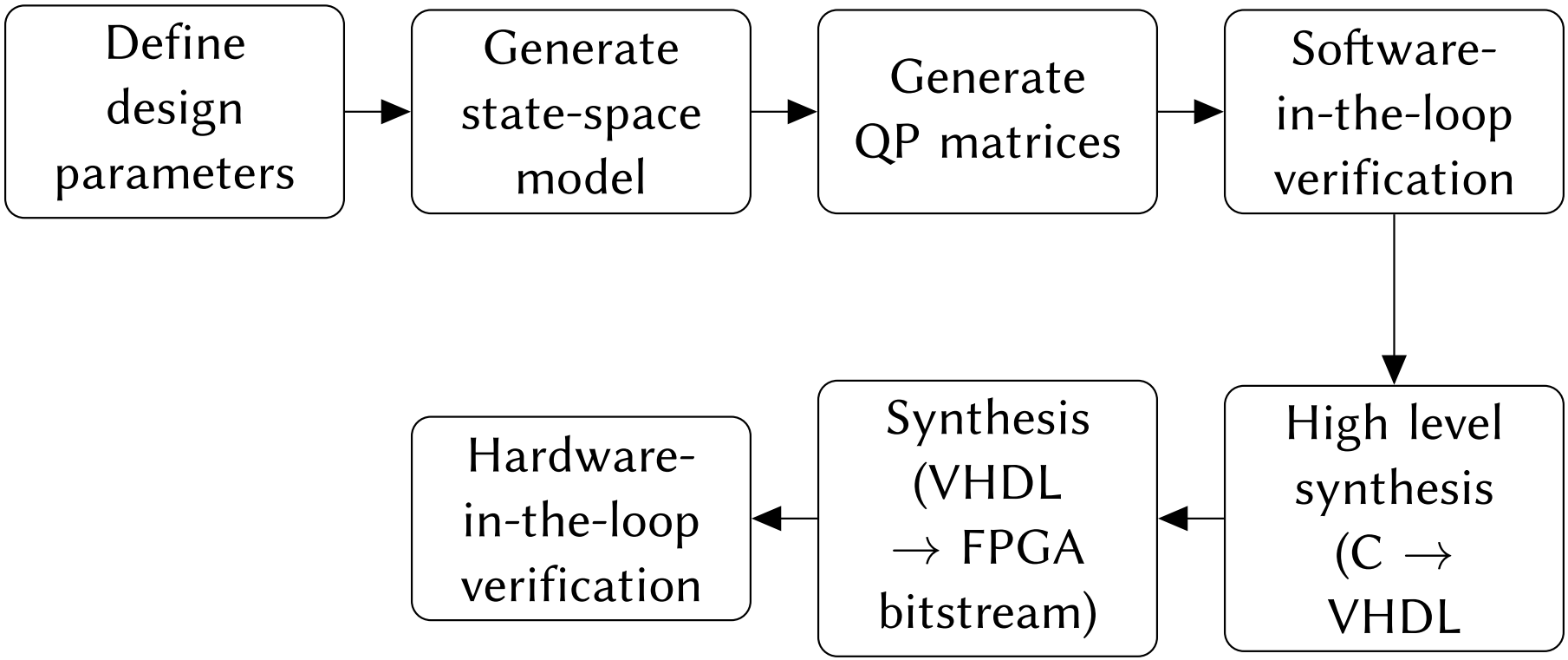
Field Programmable Gate Array (FPGA)

- Contains logic gates, lookup tables, RAM, etc.
- Target FPGA: ZedBoard
- Features
  - Processor: Xilinx's Zynq-7000
  - Cost: 450 USD

- On-board resource:

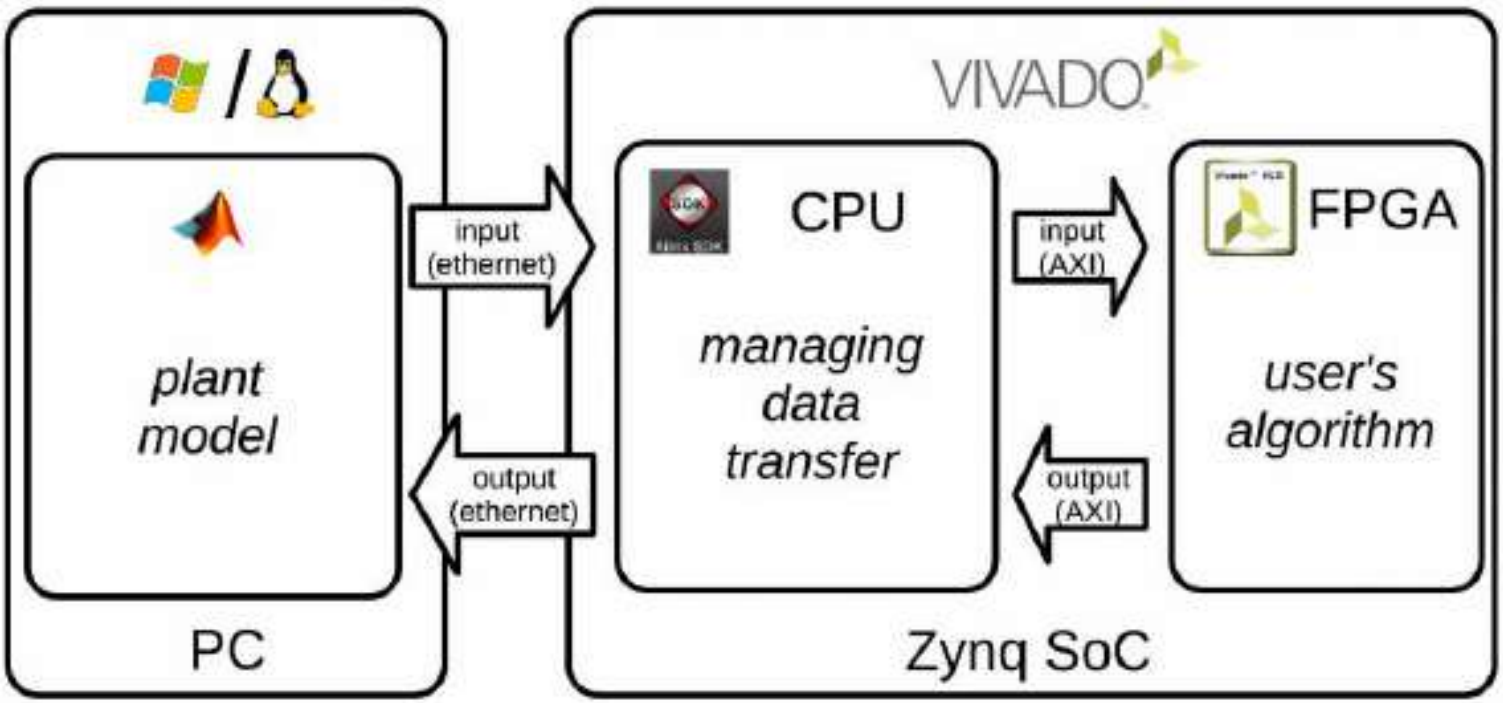
Resource	Available	Size/bits	Total memory (kB)
BRAM	280	18000	627.2
DSP	220	48	1.3
FF	106400	1	13
LUT	53200	64	415.6

FPGA Design Flow



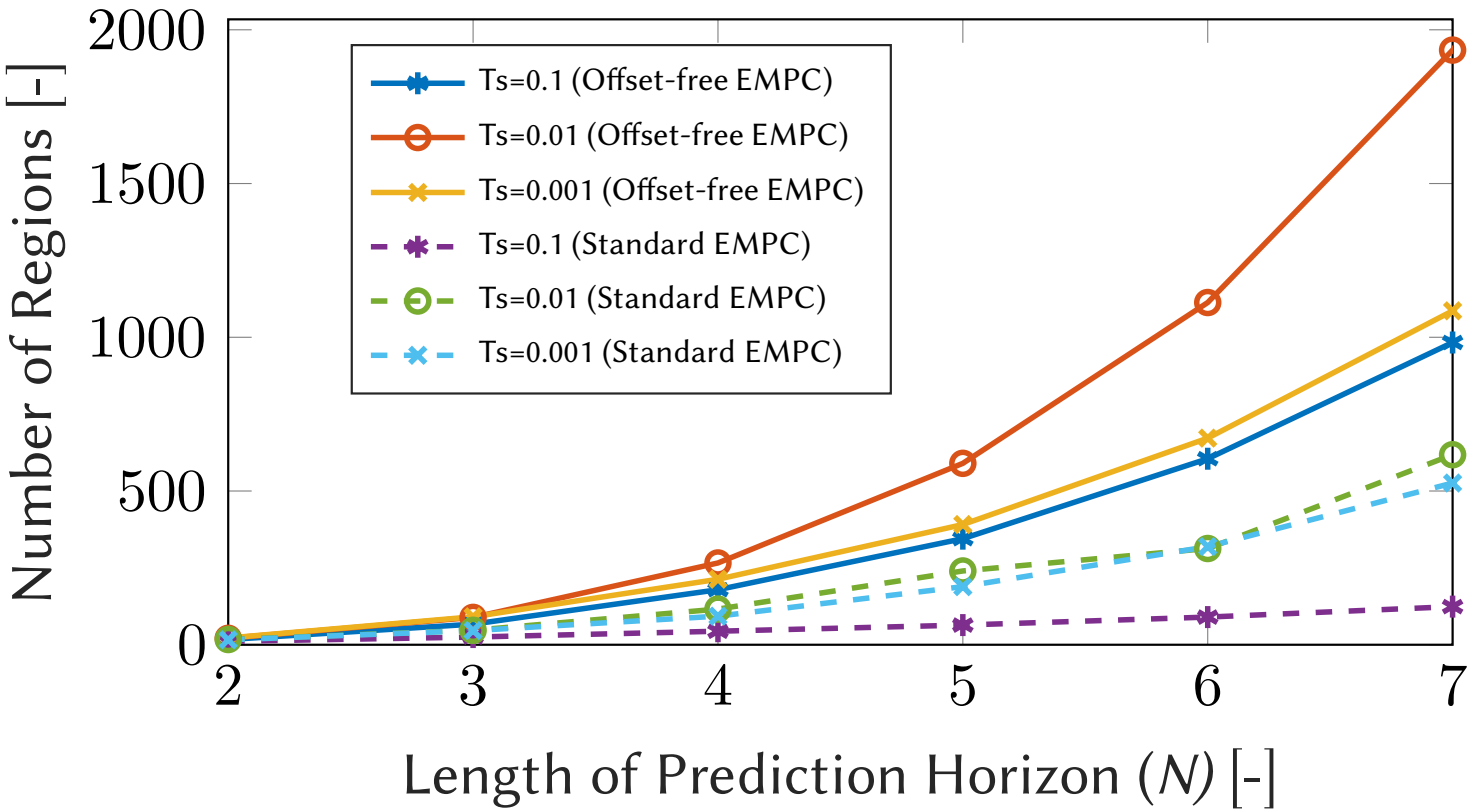
Hardware-in-loop Scheme

- HIL synthesis using MATLAB and Protoip[2]



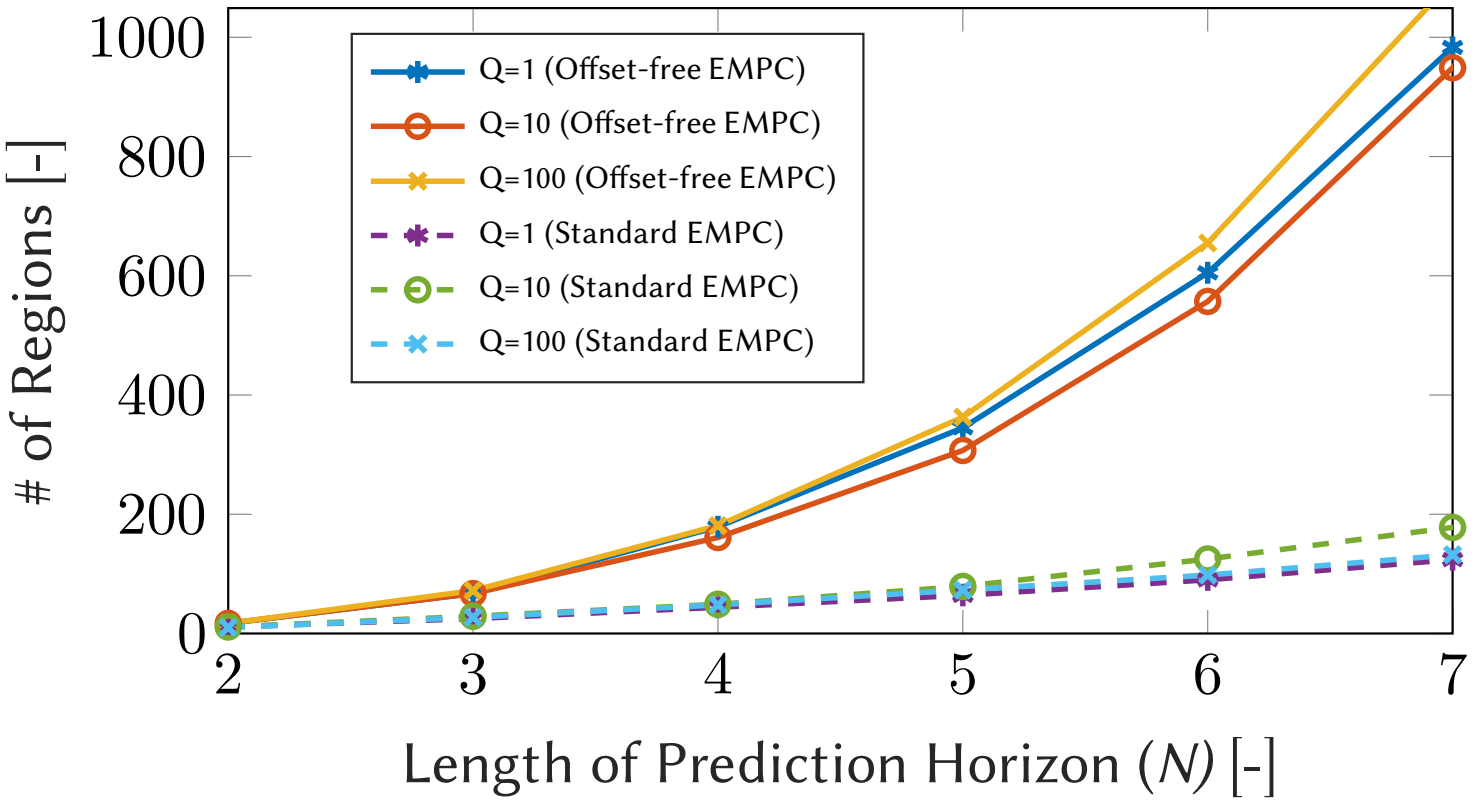
[2] https://github.com/erickerrigan/protoip

Case I: Effect of Sampling Time on number of regions



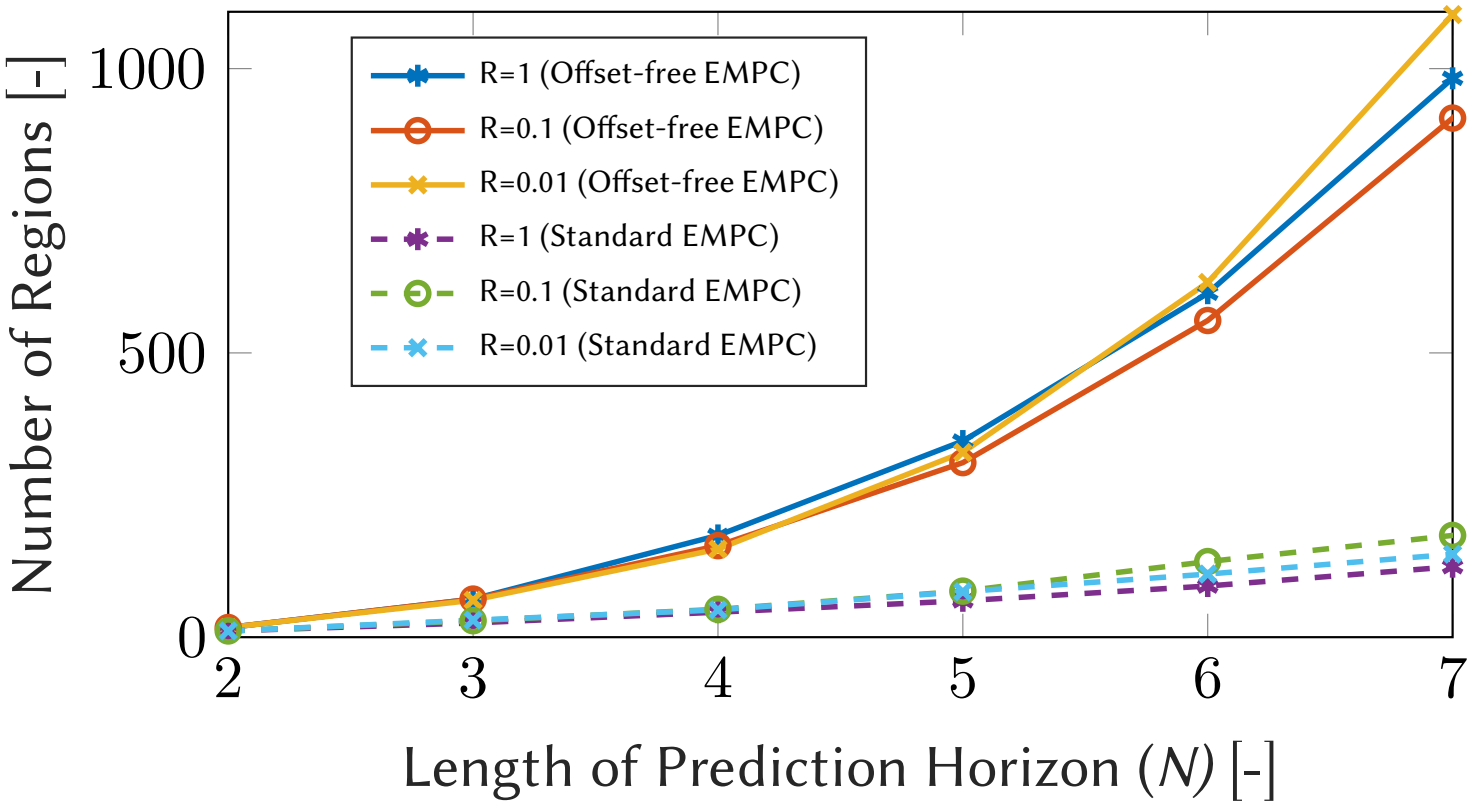
Number of regions increases with change in sampling time and horizon

Case II: Effect of Output Penalty matrix



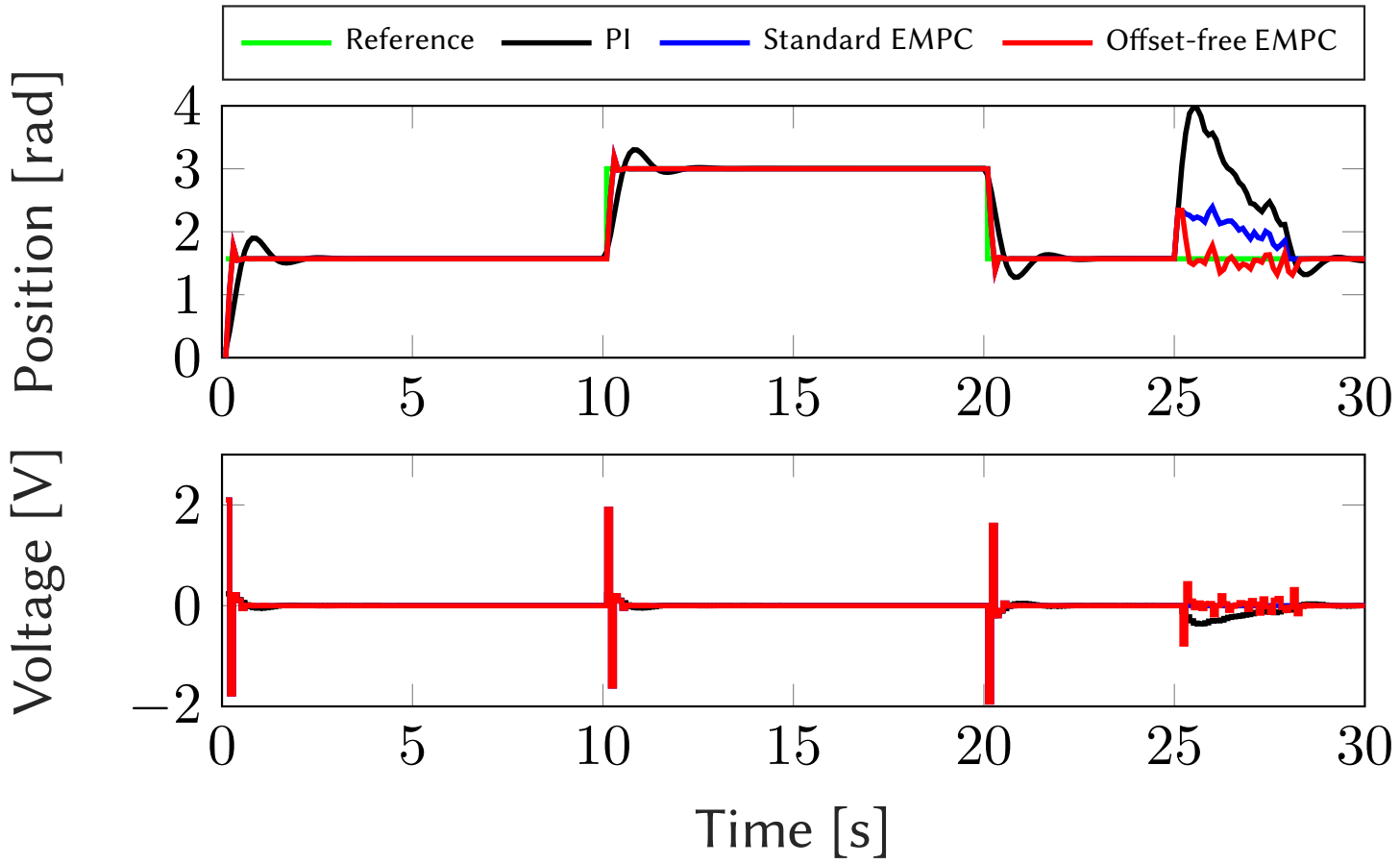
Number of regions increases with increase in output penalty and horizon

Case III: Effect of Input Penalty matrix



Number of regions increases with increase in input penalty and horizon

Controller Performance Comparison



Offset-free EMPC outperforms PI and standard EMPC

Resource Utilization and Memory Calculation

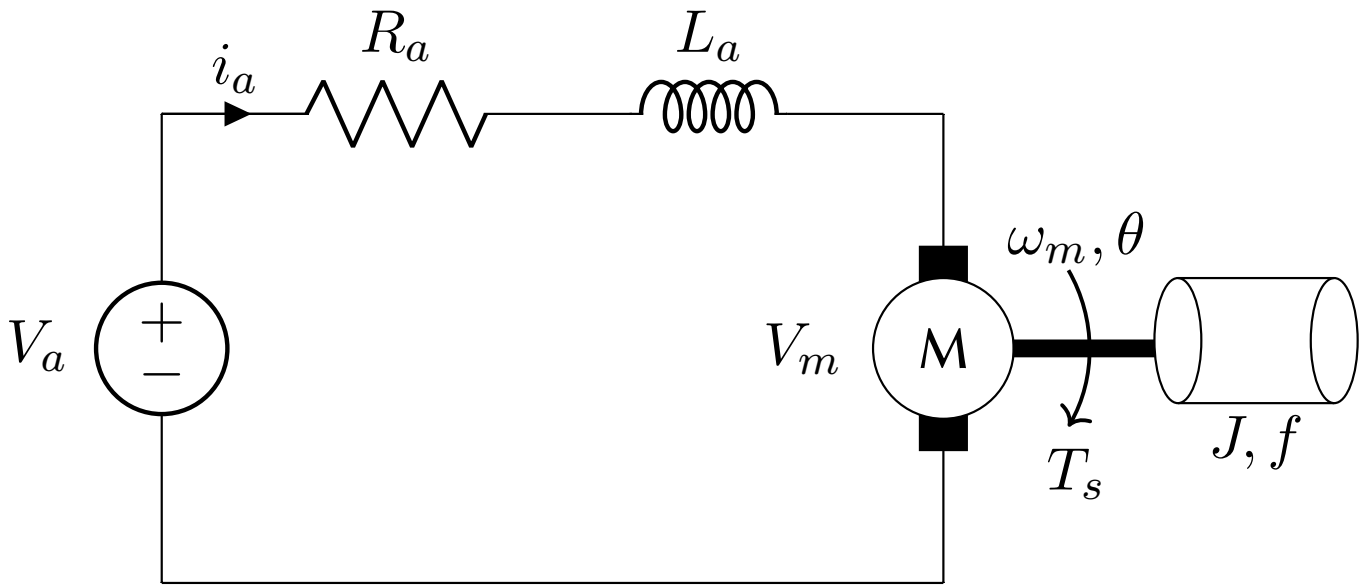
Controller	BRAM	DSP	FF	LUT	Total Memory (kB)
PI	2	24	4166	3543	33
Standard EMPC	21	14	3722	4388	82
Offset-free EMPC	258	14	3453	3157	603

Conclusions

- Explicit MPC is promising solution for embedded applications
- Offset-free explicit MPC can estimate states and help to reject disturbances
- Low-cost FPGA can be used for the control of small scale systems
- Future goal: to apply memory reduction techniques to use low-cost FPGAs for medium scale systems

Back-up Slides

DC Motor Modeling



Mathematical Equations

For a DC Motor, we have set of Differential algebraic equations (DAE) as follows,

$$\begin{aligned}
\frac{di_a(t)}{dt} &= -\frac{R_a}{L_a} i_a(t) - \frac{K_b}{L_a}\omega(t) + \frac{1}{L_a} V_a(t), \\
\frac{d\omega(t)}{dt} &= \frac{K_t}{J} i_a(t) - \frac{f}{J}\omega(t) - \frac{T_l(t)}{J}, \\
\frac{d\theta}{dt} &= \omega(t), \\
0 &= T_s - K_t i_a.
\end{aligned}$$

where, i_a is the Armature Current, L_a is the Armature Inductance, $V_a(t)$ is the Armature Voltage, R_a is the Armature Resistance, K_b is the back emf coefficient , ω is the motor angular speed, J is the inertial constant, T_s is the Shaft Torque, T_l is the load Torque, f is the frictional coefficient, θ is the position of the shaft.

State Space Model

The model has 3 states, 1 input and 3 measured outputs.

$$\begin{bmatrix} \dot{i_a}(t) \\ \dot{\omega}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_b}{L_a} & 0 \\ \frac{K_t}{J} & -\frac{f}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \\ \theta(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \end{bmatrix} Va(t)$$

$$y(t) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \\ \theta(t) \end{bmatrix}$$

Model Parameters

Parameters	**Datasheet Value**	**Estimated Value**
Armature Resistance, R_a in (Ω)	8.4	8.4
Armature Inductance, L_a in (H)	0.00116	0.00116
Inertial Constant, J in (kg m^2)	$4.0e-6$	$3.9466e-05$
Frictional Coefficient, f in (Nm/rad/s)	$0.3900095755e-6$	$3.9001e-11$
Back EMF Constant, K_b in (Nm/A)	0.042	0.0395
Rated Voltage, V in (V)	$0-18V$	$0-18V$
Rated Speed, ω in (rad/s)	450	450
Rated Torque, T_s in (N m)	0.08	0.08

MPC Formulation

When formulating the optimization problem in MPC, it is important to ensure that it can be solved in the one sampling instant. For that reason, the optimization problem is typically formulated into one of two standard forms:

- **Linear programming :** In this formulation, both the objective function and the constraints are linear.
- **Quadratic programming :** In QP formulation, the objective function is quadratic, whereas the constraints have to be linear.

State Space Model

We consider the discrete-time Linear Time-Invariant (LTI) state space model of the type

$$x_{k+1} = Ax_k + Bu_k,$$
$$y_k = Cx_k + Du_k,$$

where,
$x(t) \in \mathbb{R}^{n_x}$ is the system state vector,
$u(t) \in \mathbb{R}^{n_u}$ is the system input vector,
$y(t) \in \mathbb{R}^{n_y}$ is the system output vector,
$A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{n_y \times n_x}$, and $D \in \mathbb{R}^{n_y \times n_u}$, are system matrices.

Prediction

$$
\begin{aligned}
x_1 &= Ax_0 + Bu_0, \\
x_2 &= Ax_1 + Bu_1, \\
&= A(Ax_0 + Bu_0) + Bu_1, \\
&= A^2x_0 + ABu_0 + Bu_1, \\
x_3 &= Ax_2 + Bu_2, \\
&= A(A^2x_0 + ABu_0 + Bu_1) + Bu_2, \\
&= A^3x_0 + A^2Bu_0 + ABu_1 + Bu_2, \\
&\vdots \\
x_k &= A^kx_0 + \sum_{j=0}^{k-1}(A^{k-1-j}B)u_j.
\end{aligned}
$$

Prediction

The state predictions x_{k+1}along the prediction horizon N can be formulated as,

$$X = \Psi x_0 + \Upsilon U,$$

with $\Psi \in \mathbb{R}^{n_x(N+1)\times n_x}$ and $\Upsilon \in \mathbb{R}^{n_x(N+1)\times n_u}$.

Cost Function

Minimizing an objective function for a prediction horizon :

$$\min_{U} J(U, x_0) = x_N^T P x_N + X^T Q X + U^T R U,$$

where the weighting matrices Q and R are diagonal matrices with dimensions $\mathbb{R}^{n_x(N-1)\times n_x(N-1)}$ and $\mathbb{R}^{(n_u N)\times(n_u N)}$.
The cost matrix $P \in \mathbb{R}^{n_x \times n_x}$ is solution of the associated algebraic Riccati equation.
Taking the 2-norm of above eq. we can get,

$$\min_{U} J(U, x_0) = X^T \bar{Q} X + U^T R U.$$

Constraints

Constraints on inputs, outputs and states allows the controller to stay between the specified limits.

Input Constraints

$$u_{\min} \leq u_k \leq u_{\max},$$

State Constraints

$$x_{\min} \leq x_k \leq x_{\max}.$$