Example Development and Evaluation of MPC and RL Controllers

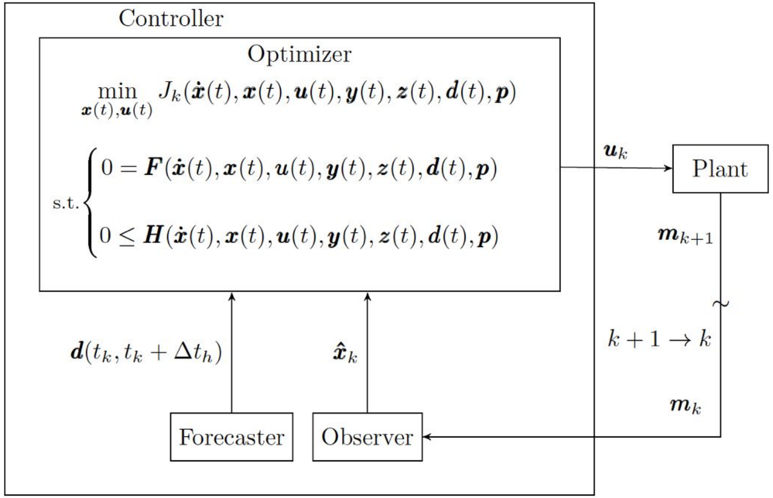
Workshop: Introduction to the BOPTEST Framework for Testing and Benchmarking Advanced Controllers

RLEM

November 12, 2023

Javier Arroyo
javier.arroyo@kuleuven.be

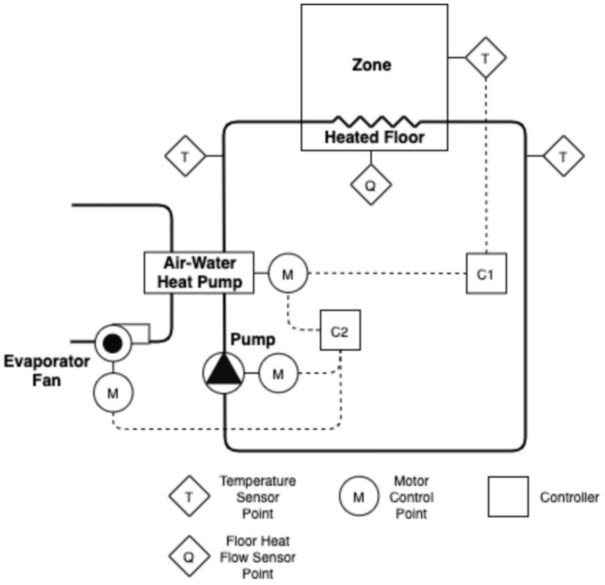
MPC working principle



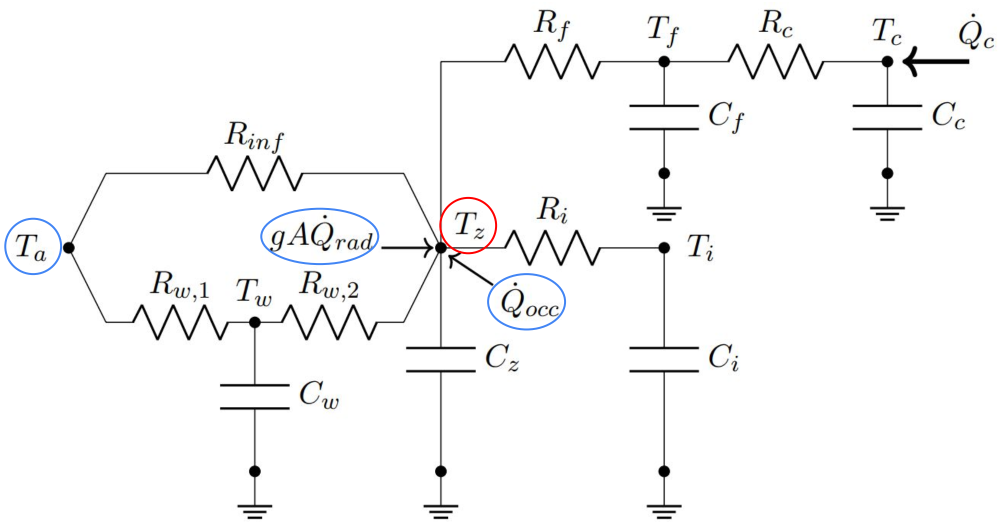
MPC: Optimizes the **future** from **domain knowledge**

$$J_k = \int_{t=t_k}^{t_k+\Delta t_h} \underbrace{l(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{d}(t), \boldsymbol{p})}_{\text{cost} \; + \; \text{w * discomfort}} dt$$

System identification



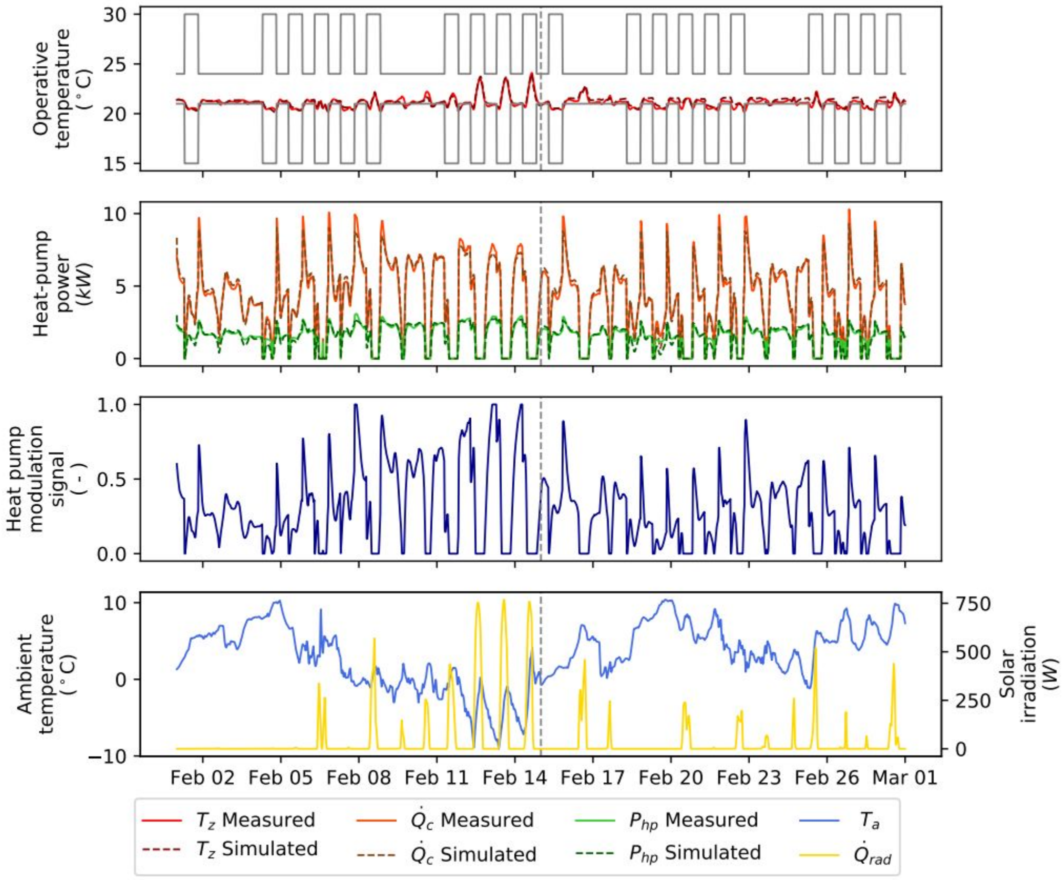
$$\dot{Q}_c = (a_c + b_c(T_c - T_{c,n}) + c_c(T_a - T_{a,n}))k_c u_{hp}$$
$$\dot{Q}_e = (a_e + b_e(T_c - T_{c,n}) + c_e(T_a - T_{a,n}))k_e u_{hp}$$
$$P_{hp} = \dot{Q}_c - \dot{Q}_e$$
$$COP = \frac{\dot{Q}_c}{P_{hp}},$$



System identification

```python
import requests

# url for the BOPTEST service
url = "http://boptest-workshop.net"

# Select test case and get identifier
testcase = "bestest_hydronic_heat_pump"
testid = \
requests.post("{0}/testcases/{1}/select".format(url,testcase)).json()["payload"]["testid"]

# Find all measurements and inputs of this emulator
inputs       = requests.get("{0}/inputs/{1}".format(url, testid)).json()["payload"]
measurements = requests.get("{0}/measurements/{1}".format(url, testid)).json()["payload"]
all_points   = measurements.keys() + inputs.keys()

# Set the emulator in the desired simulation period and initialize
requests.put("{0}/initialize/{1}".format(url, testid),
             json={"start_time":31*24*3600,
             "warmup_period":7*24*3600}).json()["payload"]

# Simulate with baseline control for one month
for _ in range(28*24)
    requests.post("{0}/advance/{1}".format(url, testid),
                  json={}).json()["payload"]

# Gather data
res = requests.put("{0}/results/{1}".format(url, testid),
                   json={"point_names":all_point,
                         "start_time":int(0),
                         "final_time":int(3.1536e7)}).json()["payload"]
```

The **Grey-Box Toolbox** [4] is used to prototype the model and train its parameters



MPC description

- Controlled variable: zone operative temperature
- Control variable: modulation signal for HP compressor frequency
- BOPTEST deterministic forecast
- Prediction horizon: 3, 6, 12, **24**, 48 hours
- Control step: 15, **30**, 60 minutes
- Direct collocation with JModelica
- Unscented Kalman filter

$$\min_{u_{HP}} \int_{t=t_i}^{t_h} (p^{e,\tau}(P_{hp} + P_{fan} + P_{pum}) + w\delta^{T_z})dt$$

$$\dot{T}_z, P_{hp}, P_{fan}, P_{pum} = F(u_{hp}, \dot{Q}_{rad}, \dot{Q}_{occ}, T_a, T_z, T_c, T_f, T_i, T_w)$$

$$\underline{T}_z - \delta^{T_z} \leq T_z \leq \overline{T}_z + \delta^{T_z}$$

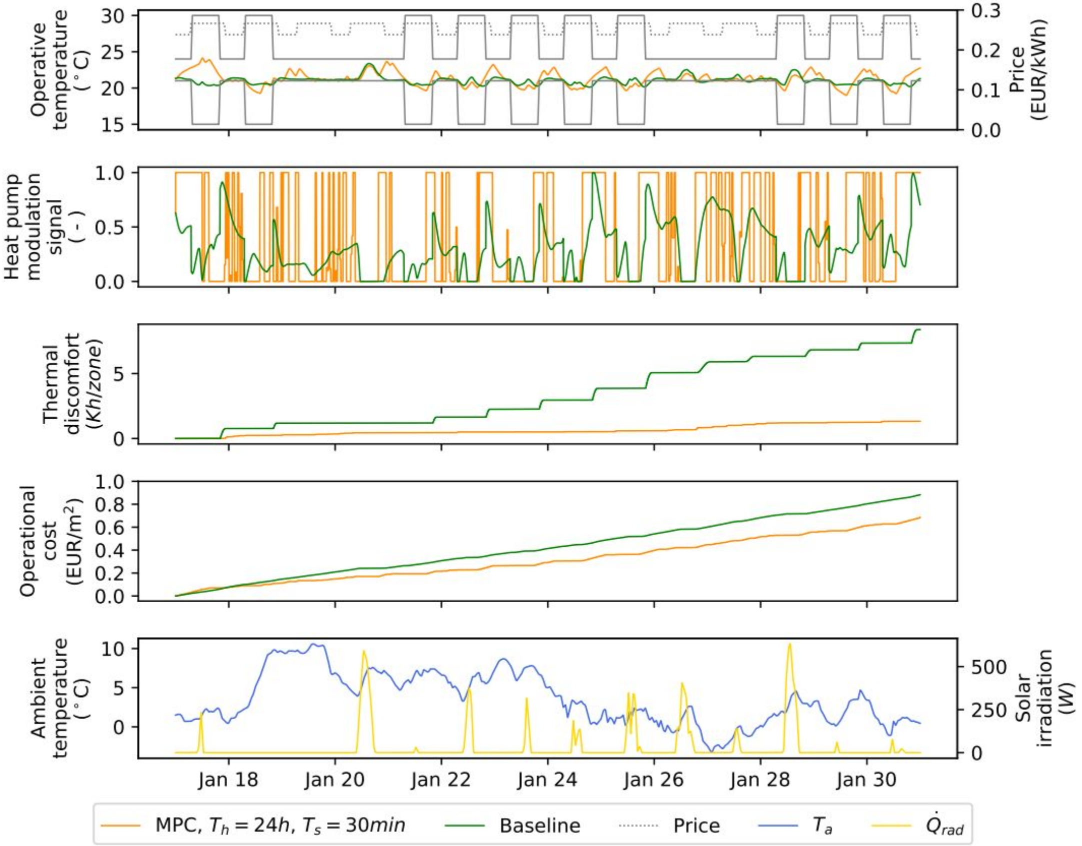
$$\delta^{T_z} \geq 0$$

$$0 \leq u_{hp} \leq 1.$$

MPC results

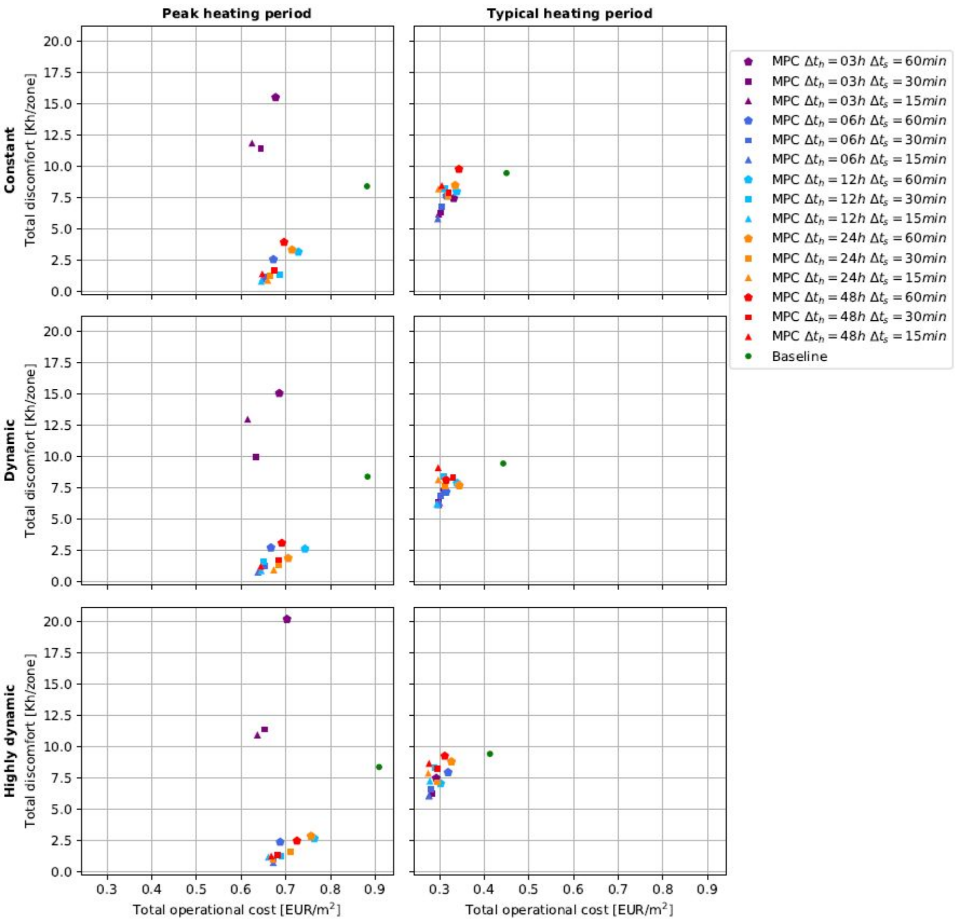
```
# -- Implement your MPC magic --
```

```
# Set step
requests.put("{0}/step/{1}".format(url, testid),
             json={"step":30*60})

# Move to the peak heat testing period with dynamic pricing
y = requests.put("{0}/scenario/{1}".format(url, testid),
                 json={"time_period":"peak_heat_day",
                       "electricity_price":"dynamic"}).json()["payload"]

# Test your MPC magic
while y:
    # Get forecast
    f = requests.put("{0}/forecast".format(self.url),
                     json={"point_names": ["TDryBul","LowerSetp[1]"],
                           "horizon":     int(24*3600),
                           "interval":    int(3600)}).json()["payload"]

    # Compute control signal
    u = mpc.compute_control(y, f)

    # Advance simulation with control signal
    y = requests.post("{0}/advance/{1}".format(url, testid),
                      json=u).json()["payload"]

    # Get KPIs
    kpi = requests.get("{0}/kpi/{1}".format(url, testid)).json()["payload"]
```



MPC results

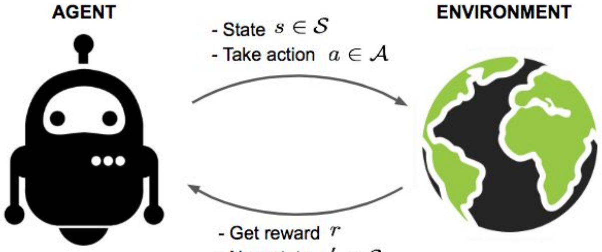
```python
# Get KPIs
kpi = requests.get("{0}/kpi/{1}".format(url, testid)).json()["payload"]
```



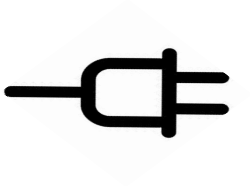
MPC vs. RL

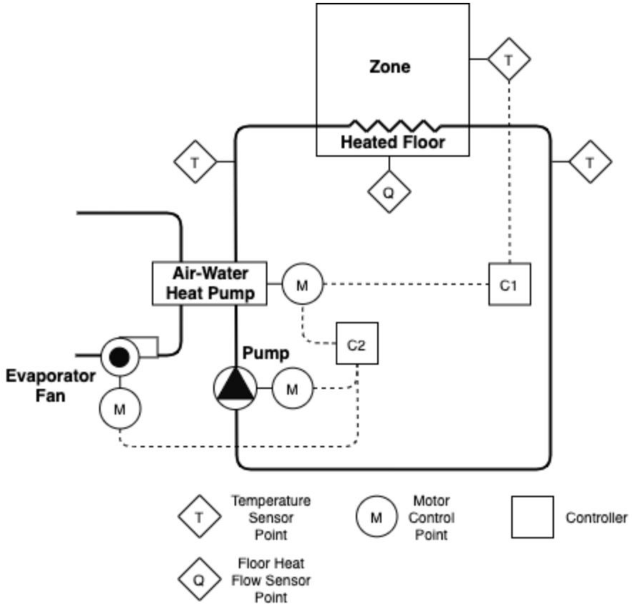
RL be like:

BOPTEST-Gym [7]



https://lilianweng.github.io

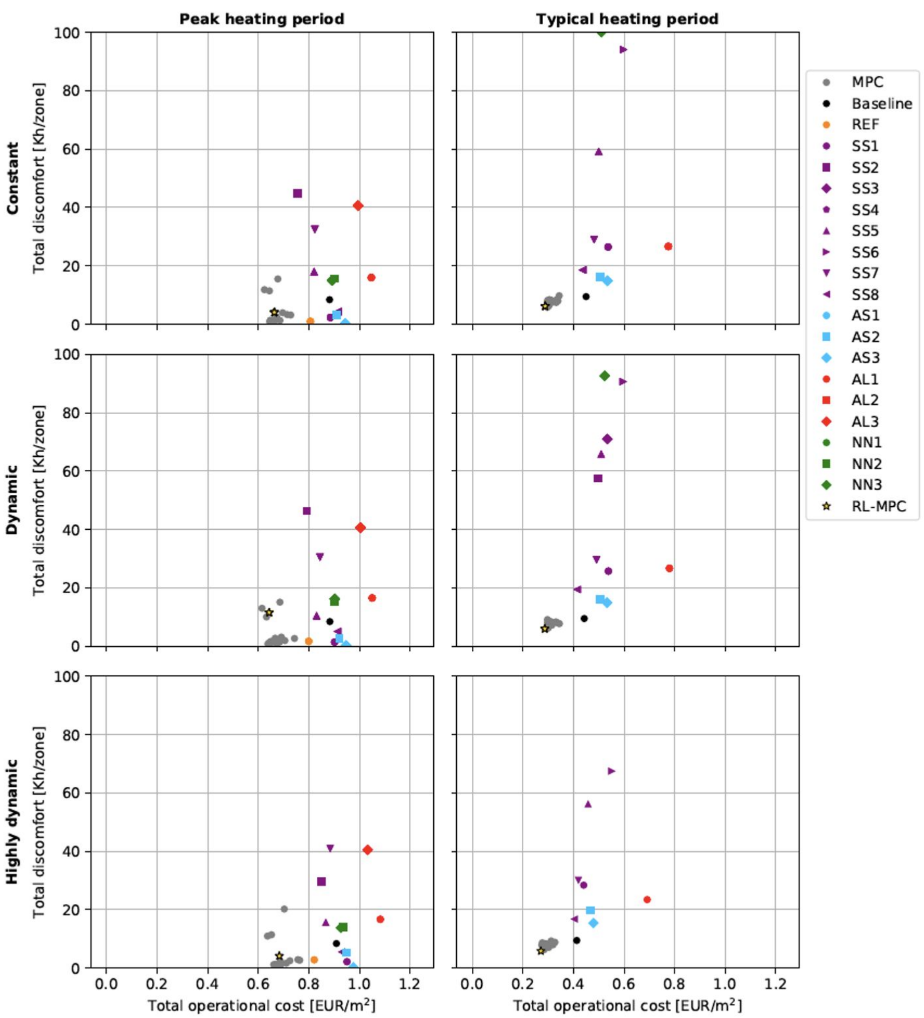




MPC vs. RL

```python
# Get KPIs
kpi = requests.get("{0}/kpi/{1}".format(url, testid)).json()["payload"]
```

	$\mathcal{S}_t$	$\mathcal{S}_m$	$\mathcal{S}_d$														
Case	t_w	T_z	λ	$\underline{T}$	$\overline{T}$	T_a	$\dot{Q}_{rad}$	$\dot{Q}_{occ}$	Δt_s	Δt_h	Δt_r	$\lvert\mathcal{S}\rvert$	$\mathcal{A}$	$\lvert\mathcal{A}\rvert$	Alg	Net	Marker
REF	✓	✓	✓	✓	✓	✓	✓	✓	15m	24h	6h	608	u_{hp}	11	DDQN	2×64	●
SS1	✓	✓	✓	✗	✗	✗	✗	✗	15m	0h	0h	3	u_{hp}	11	DDQN	2×64	●
SS2	✓	✓	✓	✓	✓	✗	✗	✗	15m	0h	0h	5	u_{hp}	11	DDQN	2×64	■
SS3	✓	✓	✓	✓	✓	✗	✗	✗	15m	3h	0h	41	u_{hp}	11	DDQN	2×64	◆
SS4	✓	✓	✓	✓	✓	✓	✓	✓	15m	3h	6h	104	u_{hp}	11	DDQN	2×64	⬟
SS5	✓	✓	✓	✓	✓	✓	✓	✓	15m	6h	6h	176	u_{hp}	11	DDQN	2×64	▲
SS6	✓	✓	✓	✓	✓	✓	✓	✓	15m	12h	6h	320	u_{hp}	11	DDQN	2×64	▶
SS7	✓	✓	✓	✓	✓	✓	✓	✓	30m	24h	6h	308	u_{hp}	11	DDQN	2×64	▼
SS8	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	158	u_{hp}	11	DDQN	2×64	◀
AS1	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	u_{hp}	2	DDQN	2×64	●
AS2	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	T_{set}	11	DDQN	2×64	■
AS3	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	T^1_{set}	11	DDQN	2×64	◆
AL1	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	u_{hp}	∞	SAC	2×64	●
AL2	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	u_{hp}	∞	A2C	1×64	■
AL3	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	u_{hp}	∞	PPO	2×64	◆
NN1	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	u_{hp}	11	DDQN	1×64	●
NN2	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	u_{hp}	11	DDQN	1×32	■
NN3	✓	✓	✓	✓	✓	✓	✓	✓	60m	24h	6h	608	u_{hp}	11	DDQN	2×64^2	◆



References

BOPTEST Github Repository: https://github.com/ibpsa/project1-boptest

BOPTEST OpenAI-Gym Interface: https://github.com/ibpsa/project1-boptest-gym

[1] D. Blum, F. Jorissen, S. Huang, Y. Chen, J. Arroyo, K. Benne, Y. Li, V. Gavan, L. Rivalin, L. Helsen, D. Vrabie, M. Wetter, and M. Sofos. (2019). "Prototyping the BOPTEST framework for simulation-based testing of advanced control strategies in buildings." In *Proc. of the 16th International Conference of IBPSA*, Sep 2 – 4. Rome, Italy.

[2] D. Blum, J. Arroyo, S. Huang, J. Drgona, F. Jorissen, H. T. Walnum, T. Chen, K. Benne, D. Vrabie, M. Wetter, and L. Helsen (2021). "Building Optimization Testing Framework (BOPTEST) for Simulation-Based Benchmarking of Control Strategies in Buildings." *Journal of Building Performance Simulation*, Accepted.

[3] X. Pang, M. A. Piette, and N. Zhou (2017). "Characterizing variations in variable air volume system controls." Energy and Buildings, vol. 135, pp. 166–175.

[4] R. D. Coninck, F. Magnusson, J. Akesson, and L. Helsen (2016). "Toolbox for development and validation of grey-box building models for forecasting and control"Journal of Building Performance Simulation, vol. 9, no. 3, pp. 288–303.

[5] S. Lucia, A. Tatulea-Codrean, C. Schoppmeyer, and S. Engell. Rapid development of modular and sustainable nonlinear model predictive control solutions. Control Engineering Practice, 60:51-62, 2017

[6] Picard, D., Jorissen, F., & Helsen, L. (2015). Methodology for obtaining linear state space building energy simulation models. In *11th international modelica conference* (pp. 51-58).

[7] Arroyo, J., Manna, C., Spiessens, F., and Helsen, L (2021). "An OpenAI-Gym environment for the Building Optimization Testing (BOPTEST) framework." In *Proc. of the 17th International Conference of IBPSA*, Sep. 1 – 3 Bruges, Belgium.

Example Development and Evaluation of Optimal Control

Thank you!

Javier Arroyo, postdoctoral researcher
javier.arroyo@kuleuven.be