Recent progress in tokamak plasma control based on reinforcement learning

JinSu KIM

asdwlstn@snu.ac.kr

Department of Nuclear Engineering, Seoul National University, Seoul, Korea

PLARE

Plasma Laboratory for Advanced REsearch

2023.04.07

SEOUL NATIONAL UNIVERSITY

Contents

- Introduction
- Environment for RL
- RL algorithms
- Experimental results
- Discussion
- Summary

Introduction

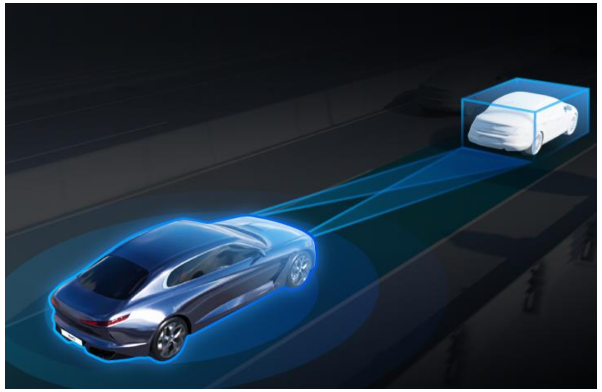
▪ Deep Learning and Reinforcement Learning

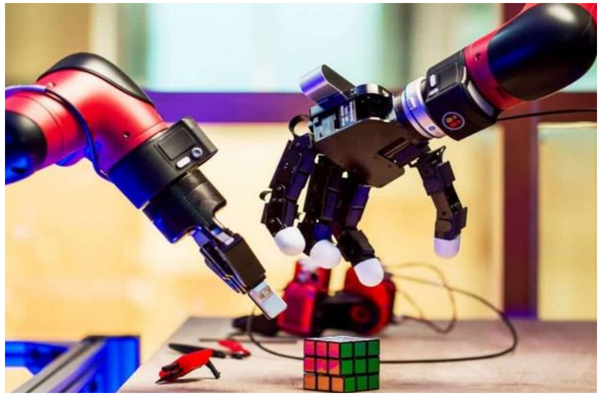
❑ <u>Current progress on deep learning and reinforcement learning</u>

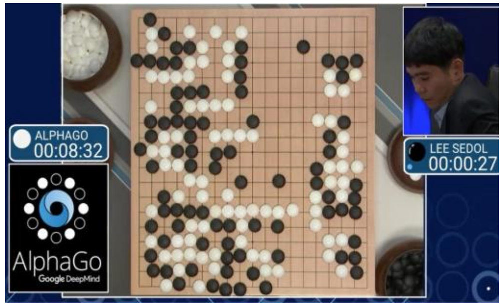
- Machine Learning framework and application: Vision / Language / Robotics / Medical …. → New paradigm
- GPU computation and Neural network (Forward-Backward algorithm): Caffe, TensorFlow, Pytorch
- Fusion with Deep Learning framework and Reinforcement Learning: ChatGPT, DQN, GPT, Dreamer, …
- What is the next step?

Transportation? Trading? Marketing? **Fusion**

DeepMind







ChatGPT

Introduction

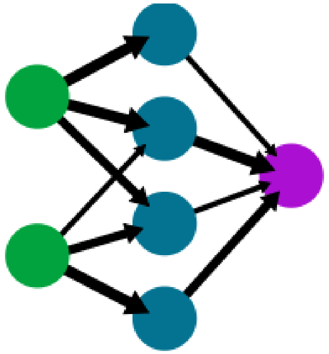
▪ Purpose of the research

❑ <u>Tokamak plasma autonomous operation based on reinforcement learning</u>

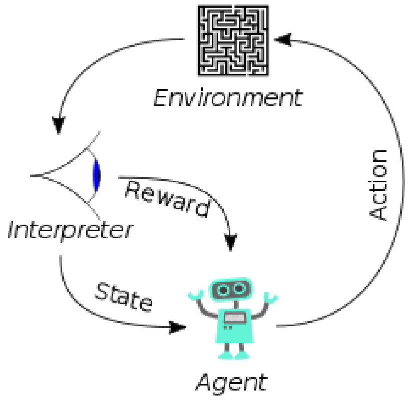
- Concept: **Control** by **Reinforcement Learning** + **Virtual KSTAR environment** + Stability (Optional)
- Under the virtual KSTAR environment, we can find out the **near-optimal** way to approach the ideal operation scenario
- How to implement the virtual KSTAR environment

  ➡ **Neural network:** A computing system for **approximating the mapping function** based on forward-backward algorithm
- How to control the virtual KSTAR environment

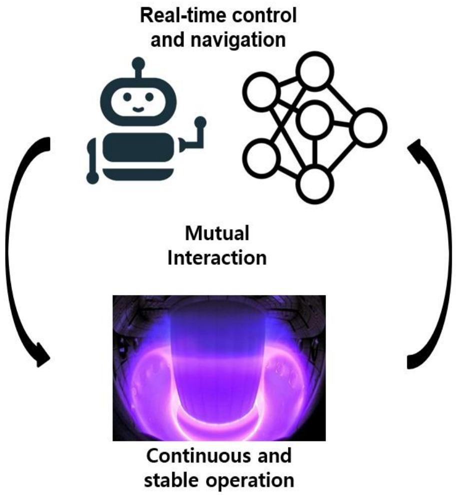
  ➡ **Reinforcement learning:** One of the machine learning concept for **dynamic decision making**



Neural Network
(Function approximator, Simulator)



Reinforcement Learning
(Decision maker / Feedback controller)



Introduction

▪ Basic of deep neural network

❑ Neuro-computing and artificial neural network

- A computing modeling tools as structures comprised of densely interconnected adaptive simple processing elements
- Easy to model the nonlinear and complex relation between input and output
- Relatively low inductive bias: better generalization in large dataset
- Main processes in neural network
  - **Forward propagation:** A computation and storage of intermediate variables including outputs and loss.
  - **Backward propagation:** A calculation of the gradient of neural network from output to input using chain rules.
- Universal approximation theorem : Neural network can represent a wide variety of functions when given appropriate weights.

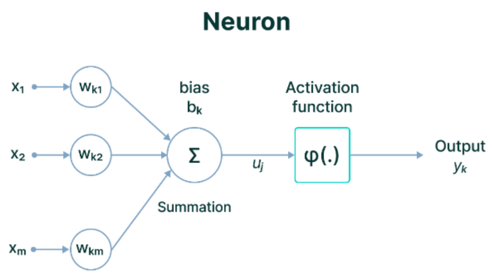
Universal approximation theorem — Let $C(X, Y)$ denote the set of continuous functions from X to Y. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x.

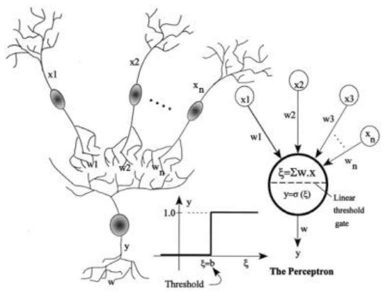
Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m), \varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$$

where

$$g(x) = C \cdot (\sigma \circ (A \cdot x + b))$$



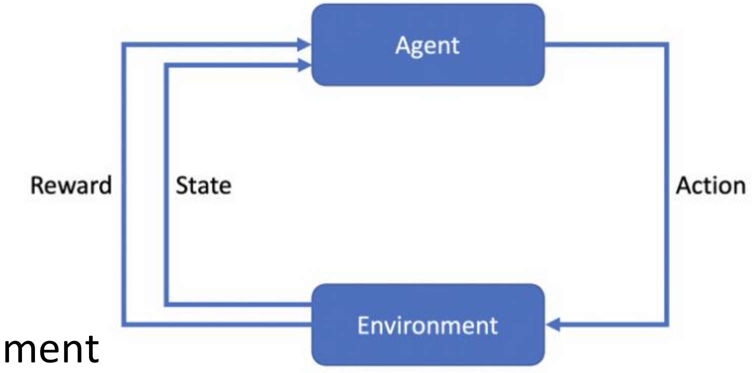


Introduction

- **Basic of reinforcement learning**

  - **Basic components**

    - **Agent**: Object that takes decisions based on the rewards and punishment
    - **Environment**: Physical world in which the agent interacts
    - **Reward**: Feedback from the environment
    - **Action**: Mechanism by which the agent transitions between states of the environment
    - **State**: Current situation of the agent (Information about the world)



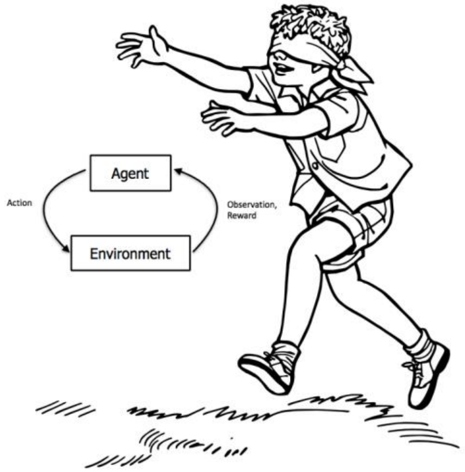
  - **Learning process for the case of policy gradient method**

    - **Objective**: To find the optimal policy which achieves the optimal reward
    - **Reward function**:

$$J(\theta) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} \pi_{\theta}(a|s) Q^{\pi}(s,a)$$

    - **How to optimize**:

$$\boxed{\begin{aligned}\nabla_{\theta} J(\theta) &= \nabla_{\theta} \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} Q^{\pi}(s,a) \pi_{\theta}(a|s) \\ &\propto \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} Q^{\pi}(s,a) \nabla_{\theta} \pi_{\theta}(a|s)\end{aligned}} \rightarrow \boxed{\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi}[Q^{\pi}(s,a) \nabla_{\theta} \ln \pi_{\theta}(a|s)]}$$



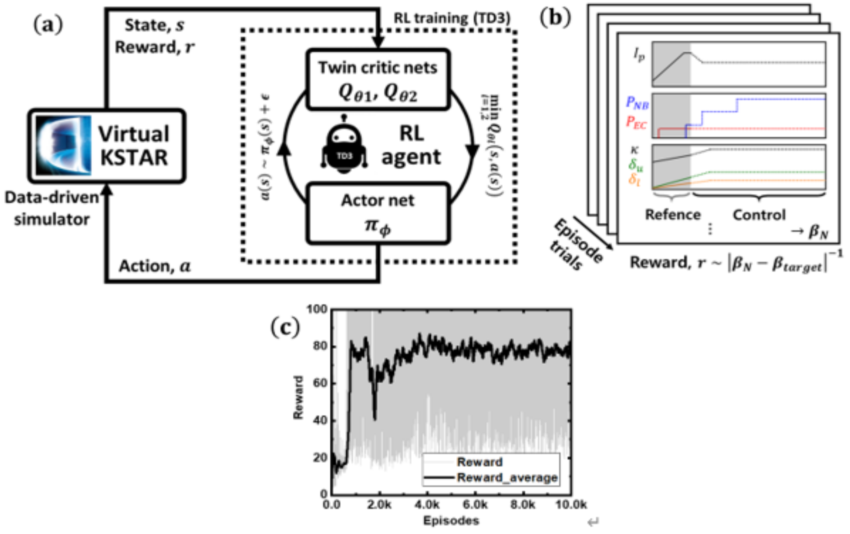
PLARE

Introduction

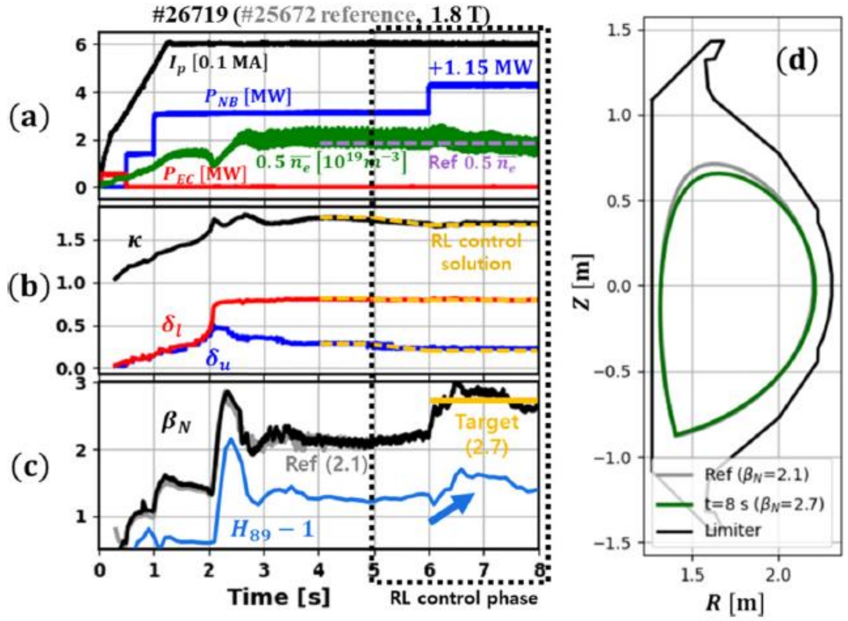
- **Related work : 0D parameter control using RL**

❑ **Feedforward beta control in the KSTAR tokamak (Jaemin Seo et al, Nucl.Fusion 61, 2021)**

- **Deep Neural Network(LSTM)** based simulator for virtual KSTAR environment
- **Feedforward control** for **KSTAR 0D parameter** using **Deep Reinforcement Learning**
- Not only validation under the virtual simulator but also the real experiment proceeded in this paper
- The RL agent cannot avoid MHD instabilities, which cannot be predicted by the LSTM network



Jaemin Seo *et al* 2021 *Nucl. Fusion* 61 106010

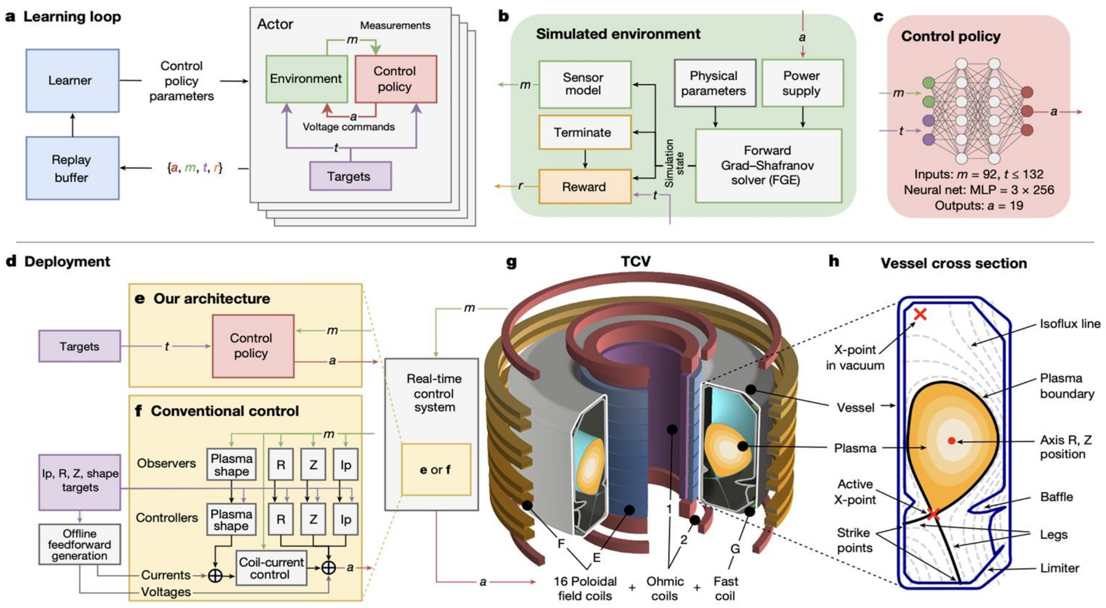


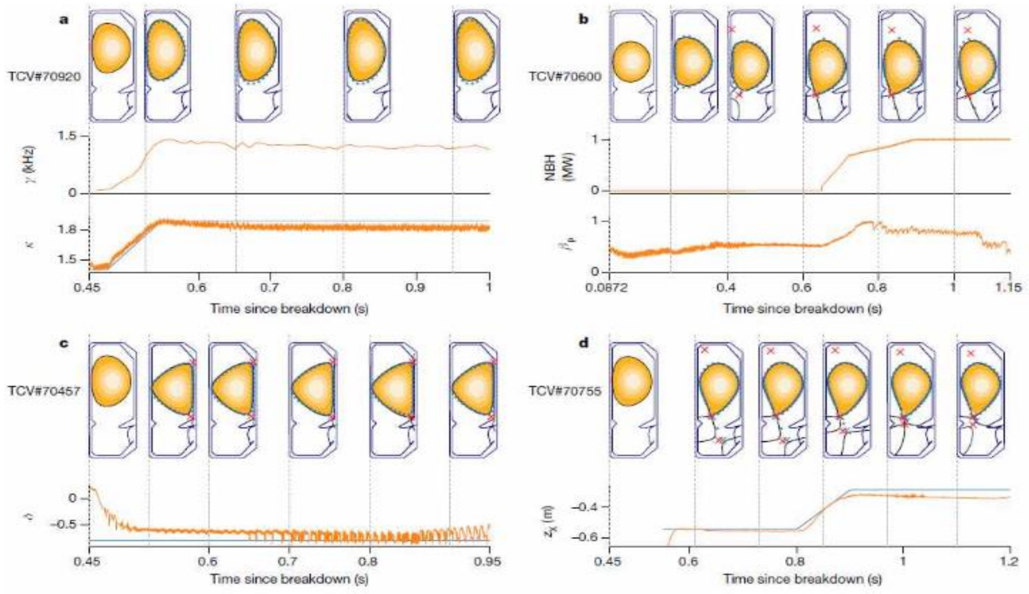
Introduction

- **Related work : plasma shape control using RL**

❑ **<u>Magnetic control of tokamak plasmas through deep reinforcement learning (Google Deepmind, Nature, 2022)</u>**

- Plasma boundary shape control by modifying the PF coil current + TF coil current using reinforcement learning
- **Policy gradient method : MPO (Maximum a Posteriori Optimization)**
- **Nonlinear Feedback** controller + **Policy-based method :** Plasma boundary shape control





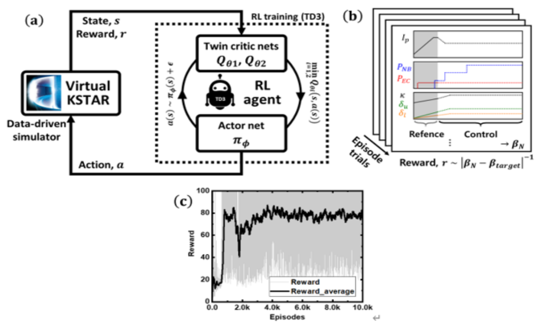
Google Deepmind, Nature 2022

PLARE

Introduction

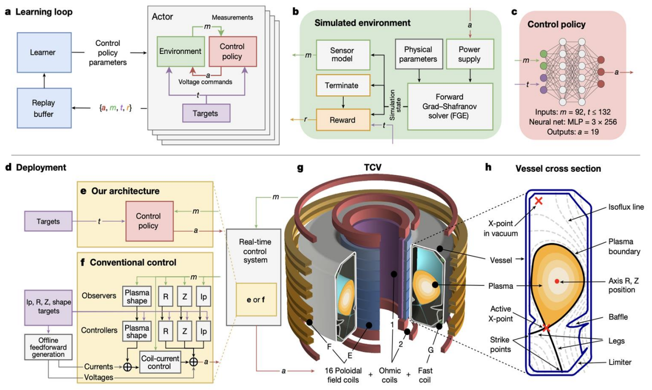
- **The final aim of the research: Tokamak plasma autonomous operation**

0D parameter control



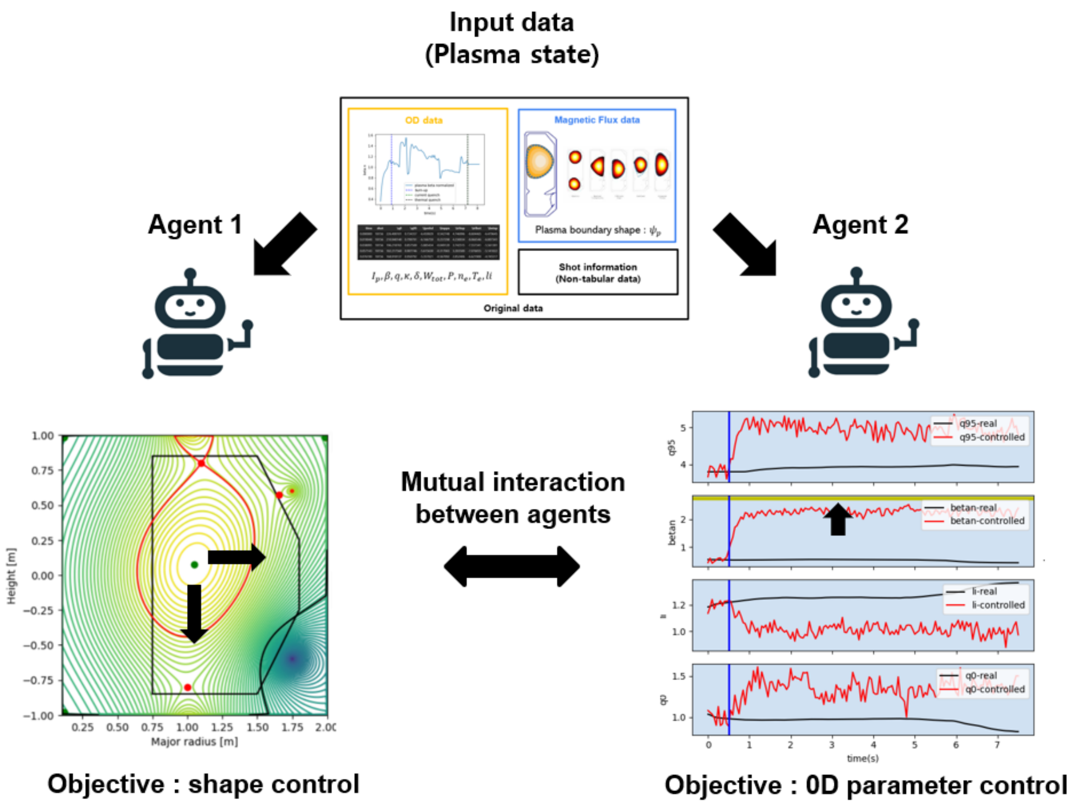
Jaemin Seo *et al* 2021 *Nucl. Fusion* 61 106010

Shape control



Google Deepmind, Nature 2022

Multi-agent reinforcement learning



Introduction

* **Control variable**: **input signal** that **we can control** (조작량)

* **Controlled variable**: **target signal** that **has to be control** (제어량)

- **Research scheme: 3-stage development**

❑ **Stage 1: 0D parameter control**

- Purpose: To control 0D parameters for achieving the high performance in virtual KSTAR environment
- Control variables: NBI heating, EC heating, I_p, B_c, Shape parameters ($\kappa, \delta, \epsilon, R_{geo}, a_{minor}$)
- Controlled variables: 0D parameter (β_n, q_{95}, li, q_0)

❑ **Stage 2: shape parameter control**

- Purpose: To control shape parameters using PF coils and Heating resources
- Control variables: PF coil current, NBI heating, EC heating, I_p, B_c
- Controlled variables: shape parameter ($\kappa, \delta, \epsilon, R_{geo}, a_{minor}$)

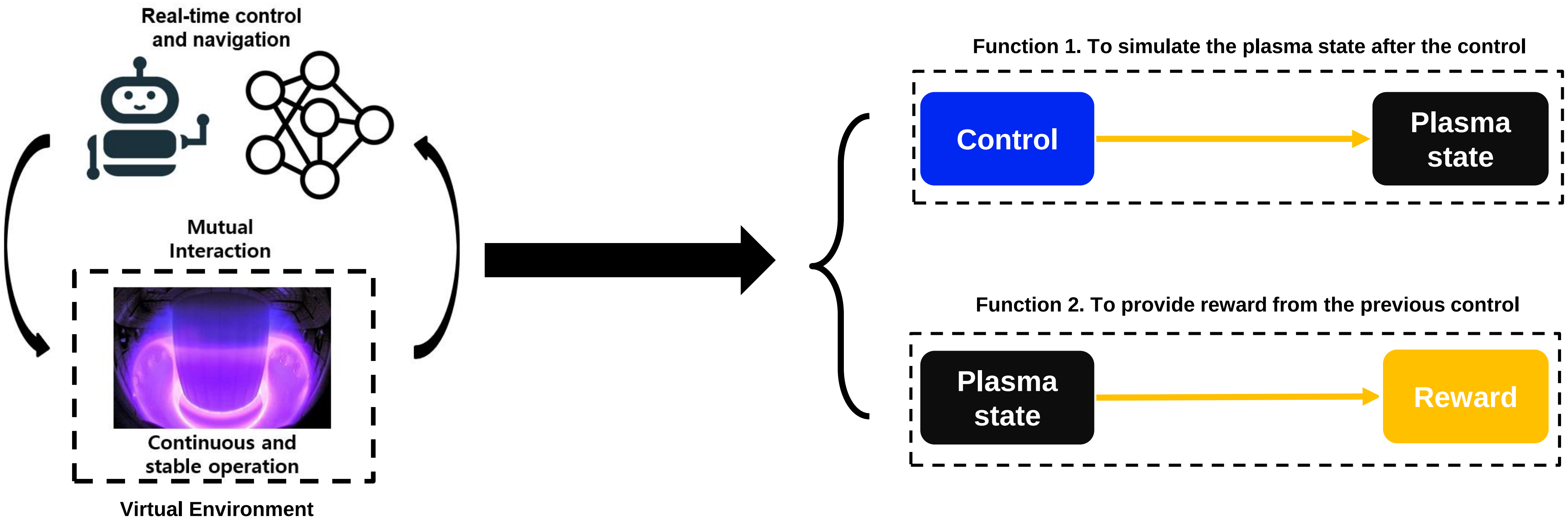
❑ **Stage 3: 0D parameter + shape parameter control**

- Purpose: To control the (1) **plasma boundary shape** and (2) **beta-N** for achieving the high performance
- Control variables: PF coil current, NBI heating, EC heating, I_p, B_c + Additional control variables
- Controlled variables: 0D parameter (β_n, q_{95}, li, q_0) + shape parameter ($\kappa, \delta, \epsilon, R_{geo}, a_{minor}$)

Environment for RL

- **Implementation of the environment for virtual tokamak operation**

❑ <u>The rule of virtual KSTAR environment</u>



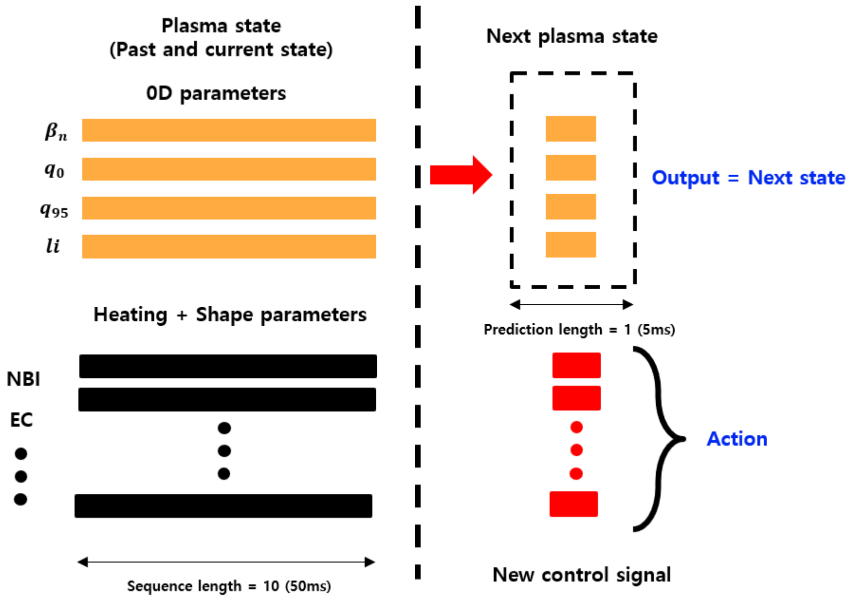
- Reinforcement Learning generally needs **reward** for learning the optimal policy
- To make the agent **understand the dynamics** of the environment, information of the **target's next state** is needed.

Environment for RL - implementation

▪ Function 1. Simulation of the plasma state

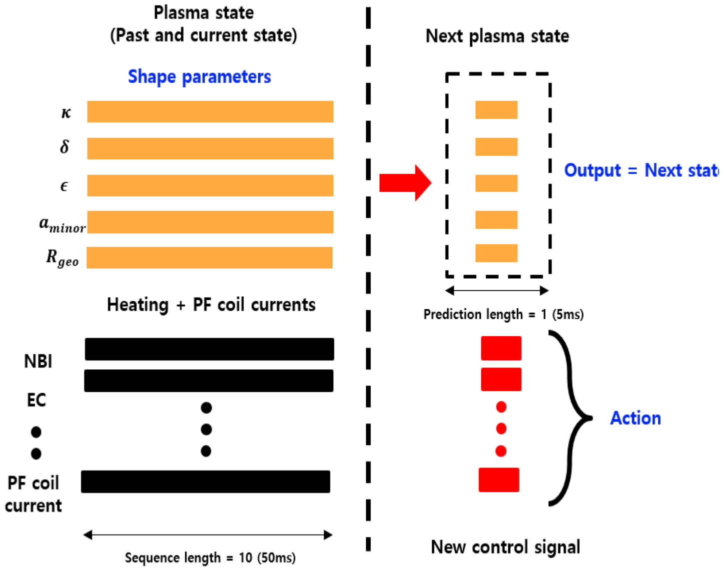
❑ <u>Neural network based simulator for predicting the next plasma state</u>

- Data architecture for 0D parameter predictor

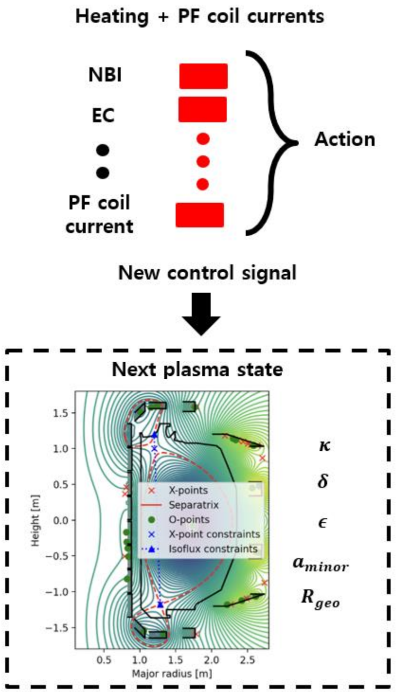


Time series forecasting / Auto-regressive method

- Data architecture for shape parameter predictor



Method 1. Time series forecasting /
Auto-regressive method



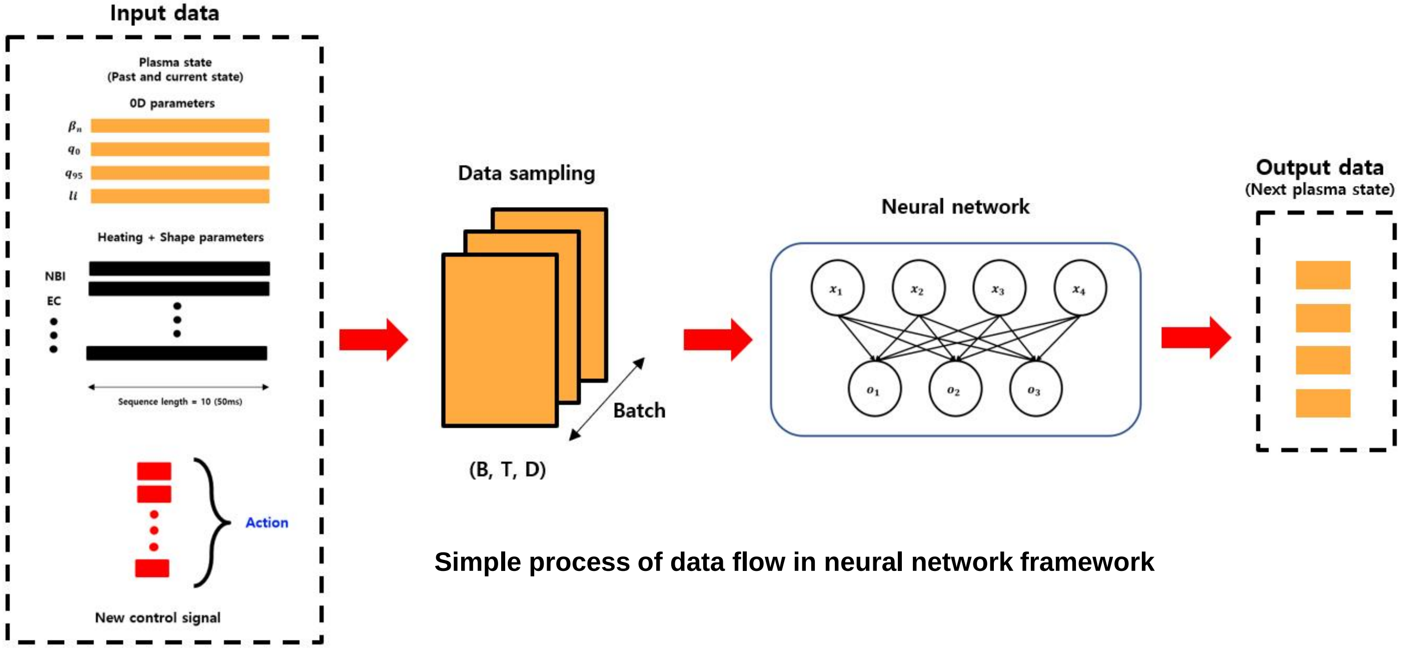
Method 2. Solving GS equation numerically

Environment for RL - implementation

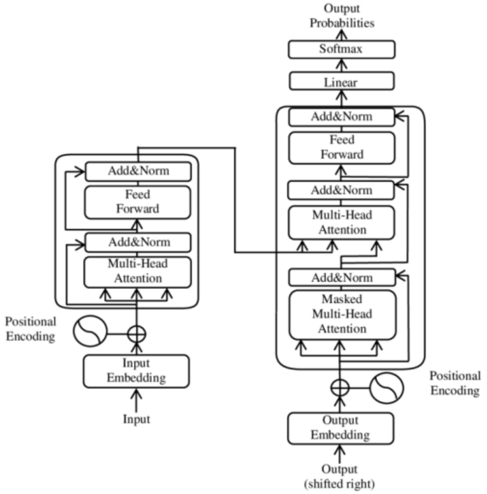
▪ Function 1. Simulation of the plasma state

❑ <u>Neural network based simulator for predicting the next plasma state</u>

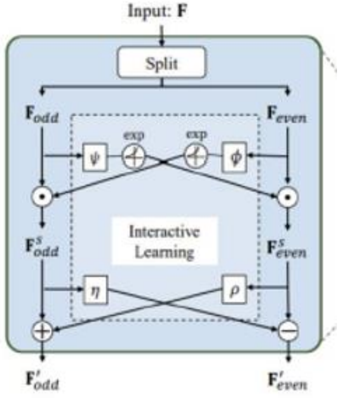
- Conventional methods: Consistent but high computational cost, limit on calibrating the experimental data error
- Data-driven methods: **Validation** of **experimental data error** in training process + Faster due to **end-to-end** computing.
- Since the dynamics of the tokamak plasma is hard to understand, data-driven approach is needed.
- **Attention mechanism based neural networks** are used: Transformer(Ashish Vaswani et al, 2017), SCINet(Minhao Liu et al, 2022)



Simple process of data flow in neural network framework



Transformer architecture, 2017



SCINet architecture, 2022

Environment for RL - implementation

- ## Function 2. reward engineering based on the plasma state

❑ <u>**Reward engineering for 0D parameter control**</u>

- **Reward**: key component for agent to **provide the feedback** for learning the optimal policy
- **A good reward function** would **increase the output reward** when the agent **approach to the target values**, but would **decrease the reward** when the **agent provides wrong actions**.
- The reward must be affected by the **current plasma state** and the **target values**.
- The **range of the reward function** can **affect the training process** of the agent.
  - Case 1. the range is too large: it can be difficult for the agent to learn the policy since it discern subtle differences between actions.
  - Case 2. the range is too small: it may not provide enough guidance for the agent to learn an optimal policy.
- The **range of the reward function** can also **affect the stability** of the learning.
- Tanh function : the range of the output value is bounded + slope increase near the original point (target value = actual value)

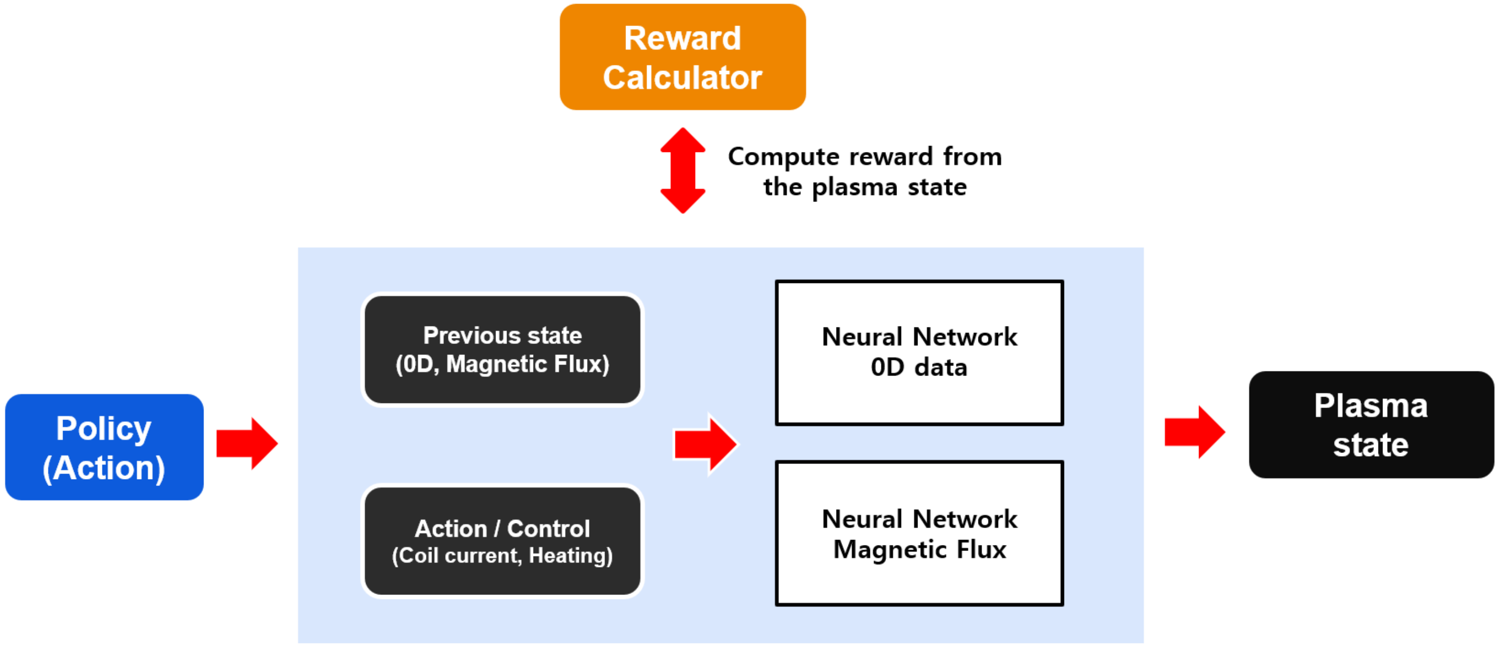
$$Reward = \tanh\left(\frac{1}{\boxed{target\ value \times 0.001} + \sqrt{(target\ value - actual\ value)^2}}\right)$$

1. **Ignore NaN**
2. **Consider target value scale**

Environment for RL - implementation

- **Integrated RL environment for virtual KSTAR plasma**

❑ **Integration with neural network and reward function**



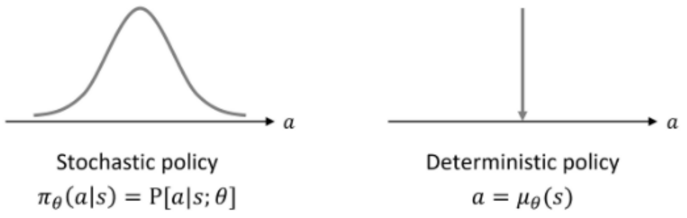
- Integration: **Neural networks** for 0D parameter prediction and boundary shape prediction + **Reward calculator**
- Action: **control parameters** including NBI, EC heating, and shape parameters / PF coil currents
- State: **plasma 0D parameters** $(\beta_n, q_{95}, li, q_0)$ / **plasma shape parameters** $(\kappa, \delta, \epsilon, R_{geo}, a_{minor})$

RL algorithms : finding the optimal policy for 0D parameter control

- ## Deep Deterministic Policy Gradient (DDPG)

❑ <u>**Actor-Critic based deterministic policy gradient algorithm**</u>

- **Policy :** A way of behaving for achieving the objective
- **Action** can be determined by the **current state** of the environment
- **Stochastic vs Deterministic**
- Using **replay buffer** to manage the trajectories (=past dataset) efficiently for training process.
- **Ornstein-Uhlenbeck process:** temporal correlated noise for exploration
- Useful for continuous action space, but sensitive to hyper-parameters

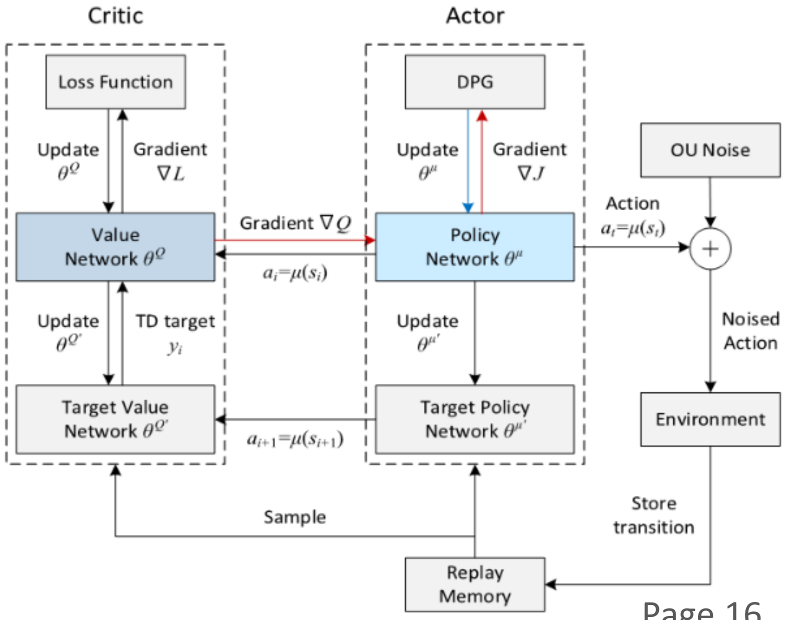


Policy Gradient Theorem $\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi}\left[\nabla_\theta \log \pi_\theta(a|s) \cdot Q^\pi(s,a)\right]$

Deterministic Policy Gradient Theorem $\nabla_\theta J(\theta) = \mathbb{E}_s\left[\nabla_\theta \mu_\theta(s) \cdot \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]$

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$



PLARE

RL algorithms : finding the optimal policy for 0D parameter control

▪ Off-policy Soft Actor-Critic (SAC)

❑ <u>Off-policy algorithm for learning a maximum entropy policy</u>

- **Actor-Critic** based algorithm + **Maximum entropy** objective **+ Double Q Learning**
- **Maximum entropy objective:** exploration ↑+ capture near optimal easily

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \boxed{\alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))}\right]$$

- **Double Q-Learning:** 2 separate critic networks used for **preventing over-optimistic value estimates** in action-value function.
- **Final objective**: **maximization** of the **expected long-term reward** + **long-term entropy**
- This algorithm is **one of the most efficient RL algorithms** in real-world robotics

Objective function for DDPG

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a \boxed{Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_t}}$$

$$\nabla_\phi \alpha \log(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)) + (\nabla_{\mathbf{a}_t}\alpha\log(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)) - \nabla_{\mathbf{a}_t}Q(\mathbf{s}_t,\mathbf{a}_t))\nabla_\phi f_\phi(\epsilon_t;\mathbf{s}_t)$$

Action-Value function + Entropy term

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]$$

Optimal policy: Maximization of the reward + Maximization of the entropy term

$$\pi_{\text{new}} = \arg\min_{\pi'\in\Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|\mathbf{s}_t)\,\middle\|\,\frac{\exp\left(\frac{1}{\alpha}Q^{\pi_{\text{old}}}(\mathbf{s}_t,\,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)}\right)$$

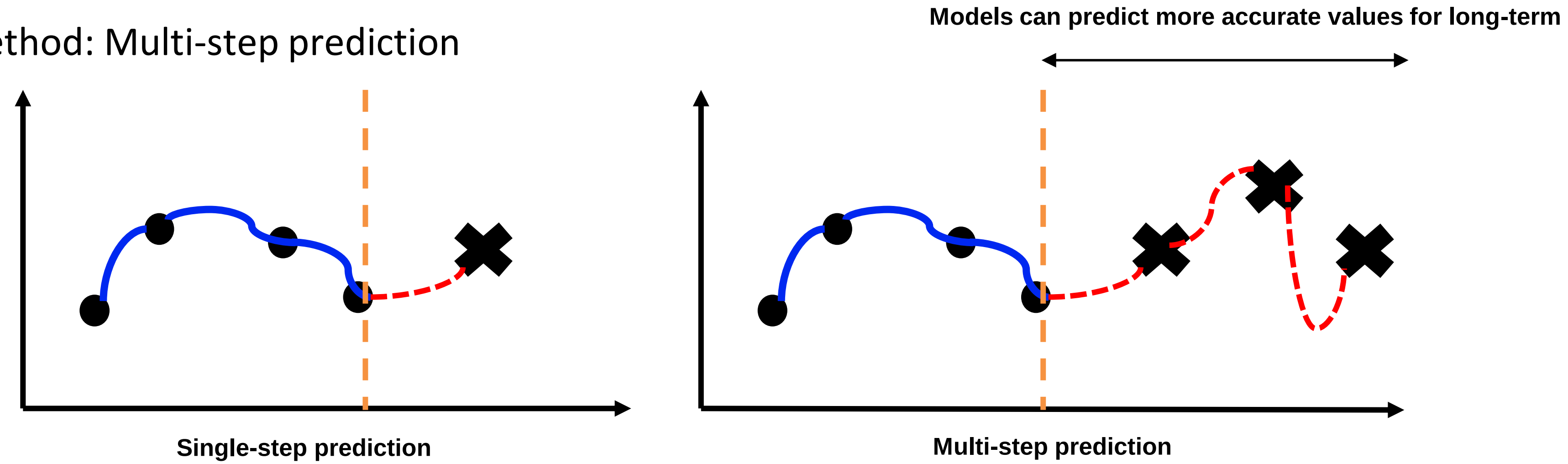
PLARE

Experimental results

- **Validation for virtual KSTAR environment**

❑ **Dataset and model setup**

- # of KSTAR experiments: 8882 (15138 ~ 31996)
- Training samples: 300,668
- Validation method: Multi-step prediction



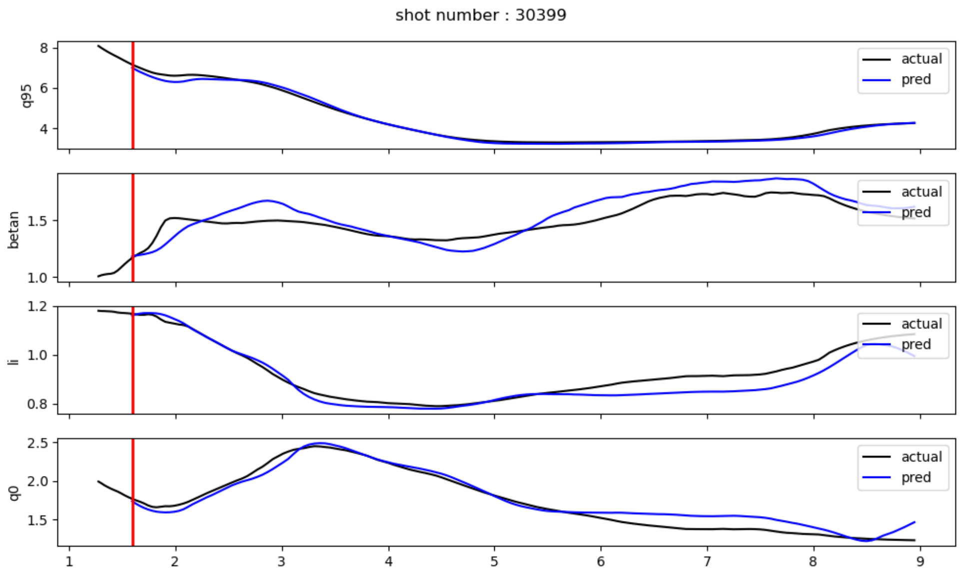
- Model / # of parameters: Transformer, 2,657,947
- Gaussian noise added for each input data in the training process: Robustness for new data + accuracy increasement

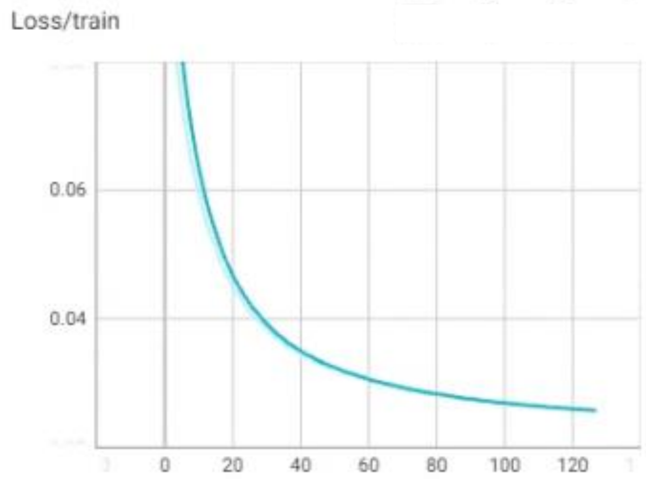
Experimental results

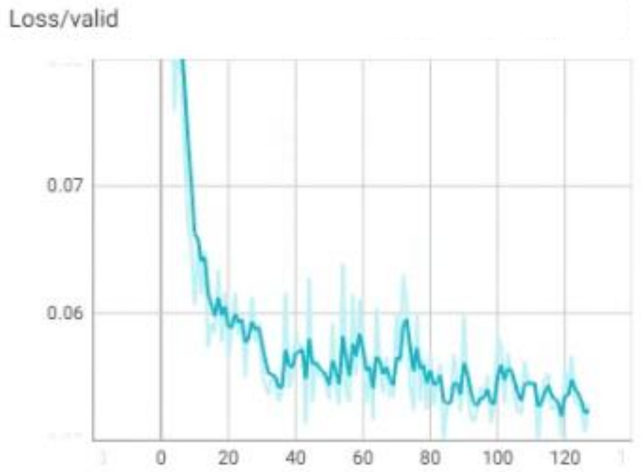
▪ Validation for virtual KSTAR environment

❑ Model performance for simulating the experiment in an auto-regressive way

- KSTAR shot # 30399 was used: initial 0D parameters + control values (NBI, EC, Ip, Bc, shape parameters)
- Input sequence length: 10 data points / 500ms
- Selected model: Transformer model
- Evaluation of test dataset: MSE : 0.064 RMSE : 0.241 MAE : 0.138 R2 : 0.870





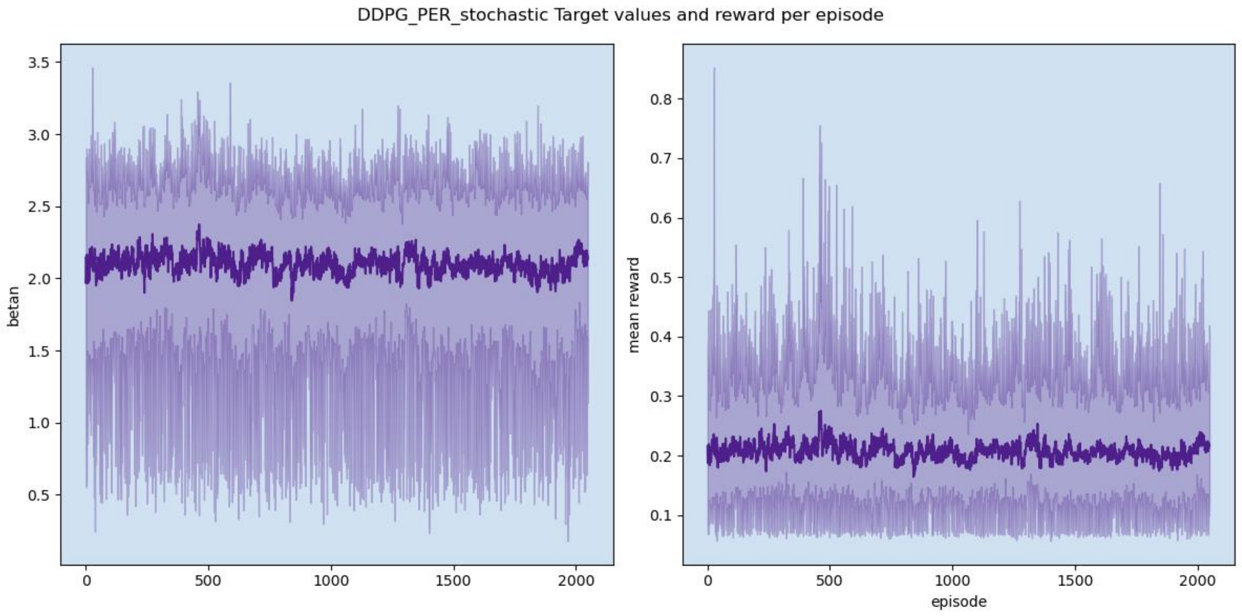


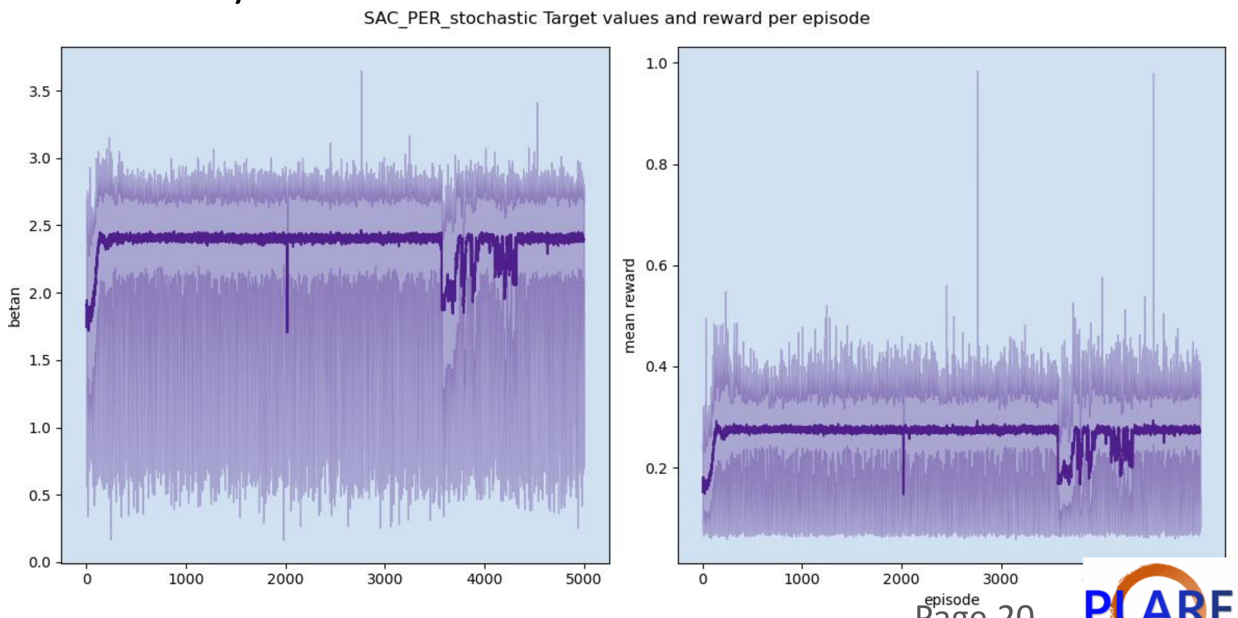
Experimental results

▪ 0D parameter control in virtual KSTAR environment

❑ <u>Training curve of DDPG and SAC</u>

- Target parameter: beta-n, 2.75
- Total episodes for training: 5000 episodes * different initial data used for each episode
- Memory buffer: Prioritized experience replay
- Stochastic environment: Gaussian noise added due to the stochasticity of real tokamak environment



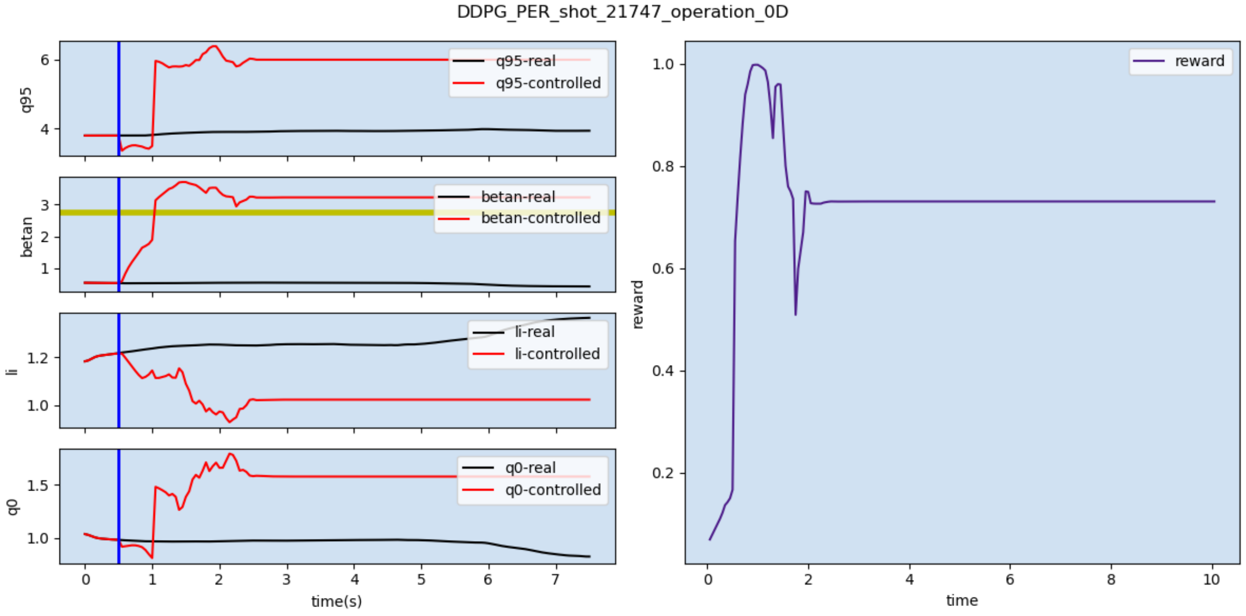


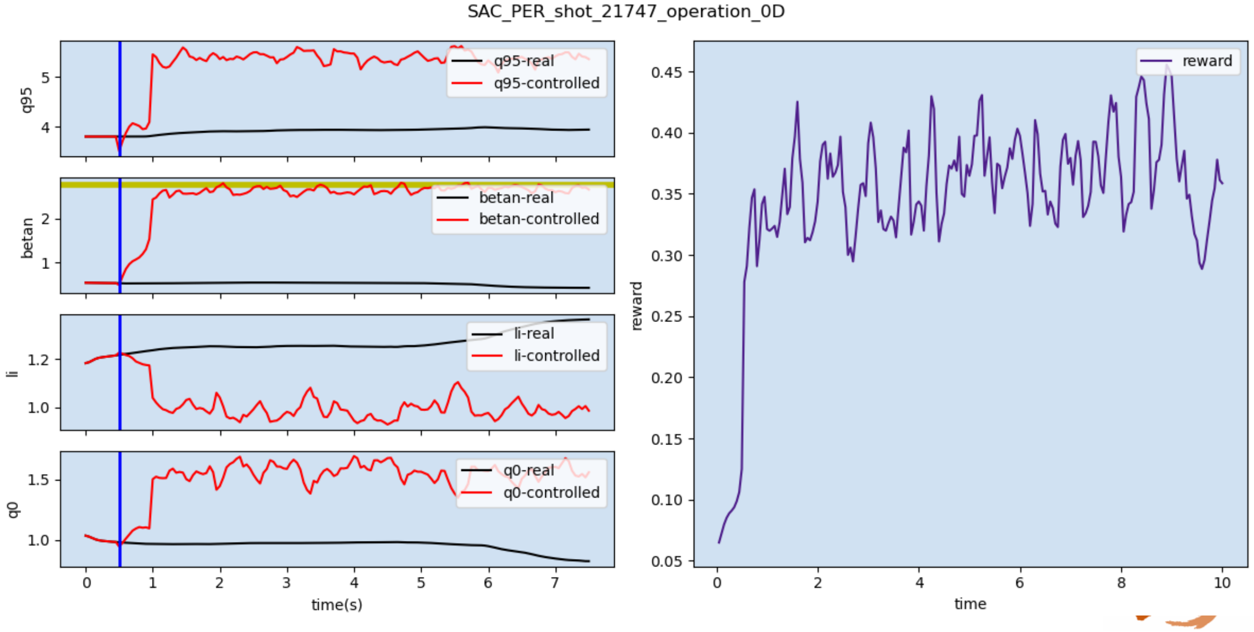
Experimental results

- **0D parameter control in a virtual KSTAR environment (deterministic)**

❑ **<u>Comparison between RL control and real experiment data</u>**

- KSTAR shot experiment: 21747 * initial data for 0D parameters and control parameters are given
- Initial 0D parameters and control parameters are equal for both DDPG and SAC.
- SAC and DDPG can control its controlled value(=betan) to be approached to the target value(=2.75), but both have different control strategies.

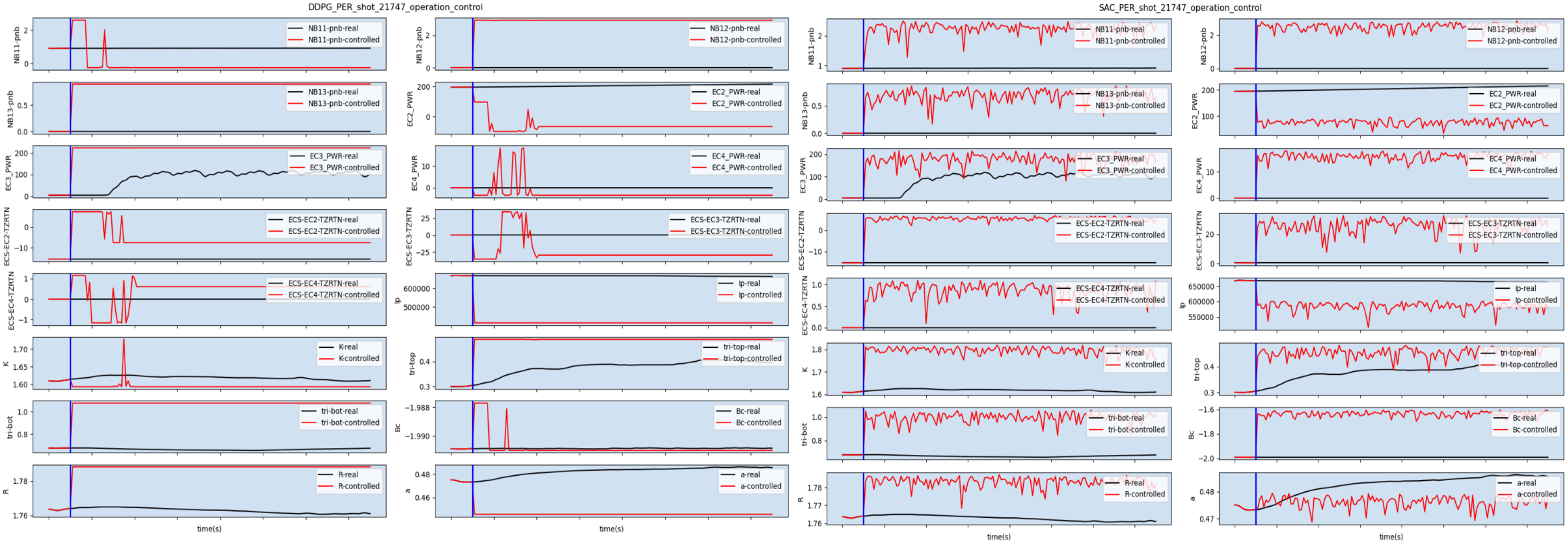




Experimental results

- **0D parameter control in a virtual KSTAR environment (deterministic)**

❑ **<u>Comparison between RL control and real experiment data (Control value)</u>**

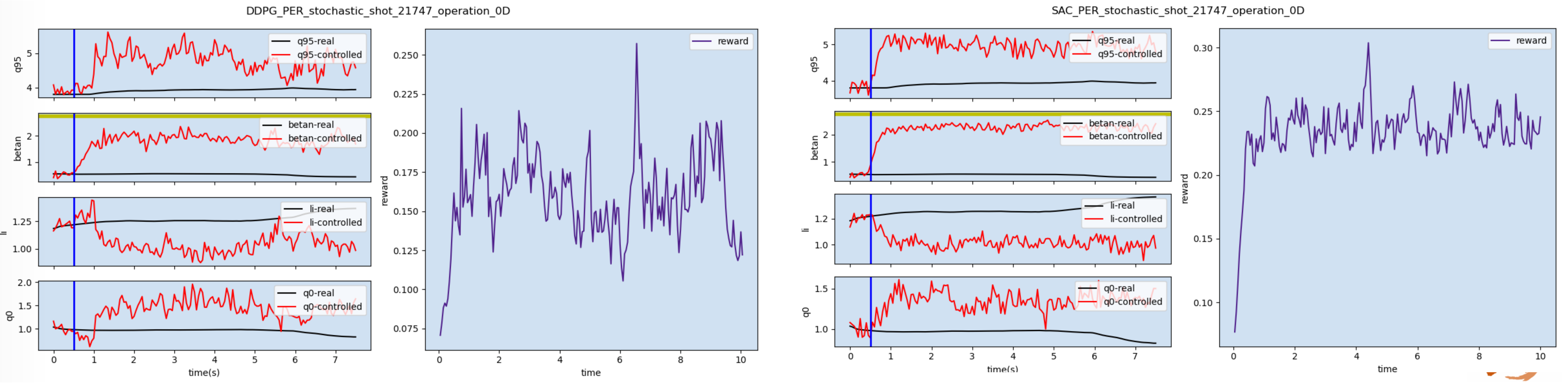


Experimental results

- **0D parameter control in a virtual KSTAR environment (stochastic)**

❑ **<u>Comparison between RL control and real experiment data</u>**

- KSTAR shot experiment: 21747 * initial data for 0D parameters and control parameters are given
- Initial 0D parameters and control parameters are equal for both DDPG and SAC.
- Both can not control the controlled parameter to be approached to the target value, and even the controlled parameter can not increase over 2.2.

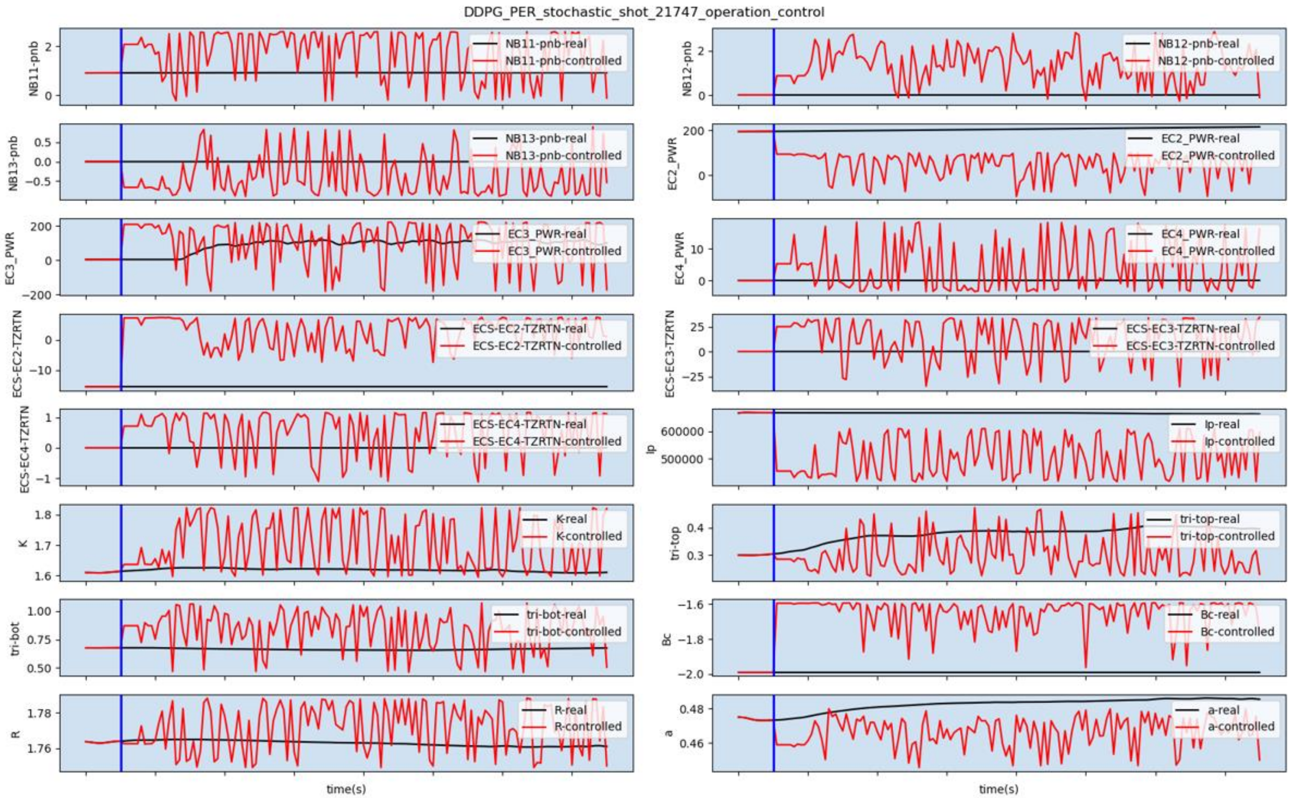


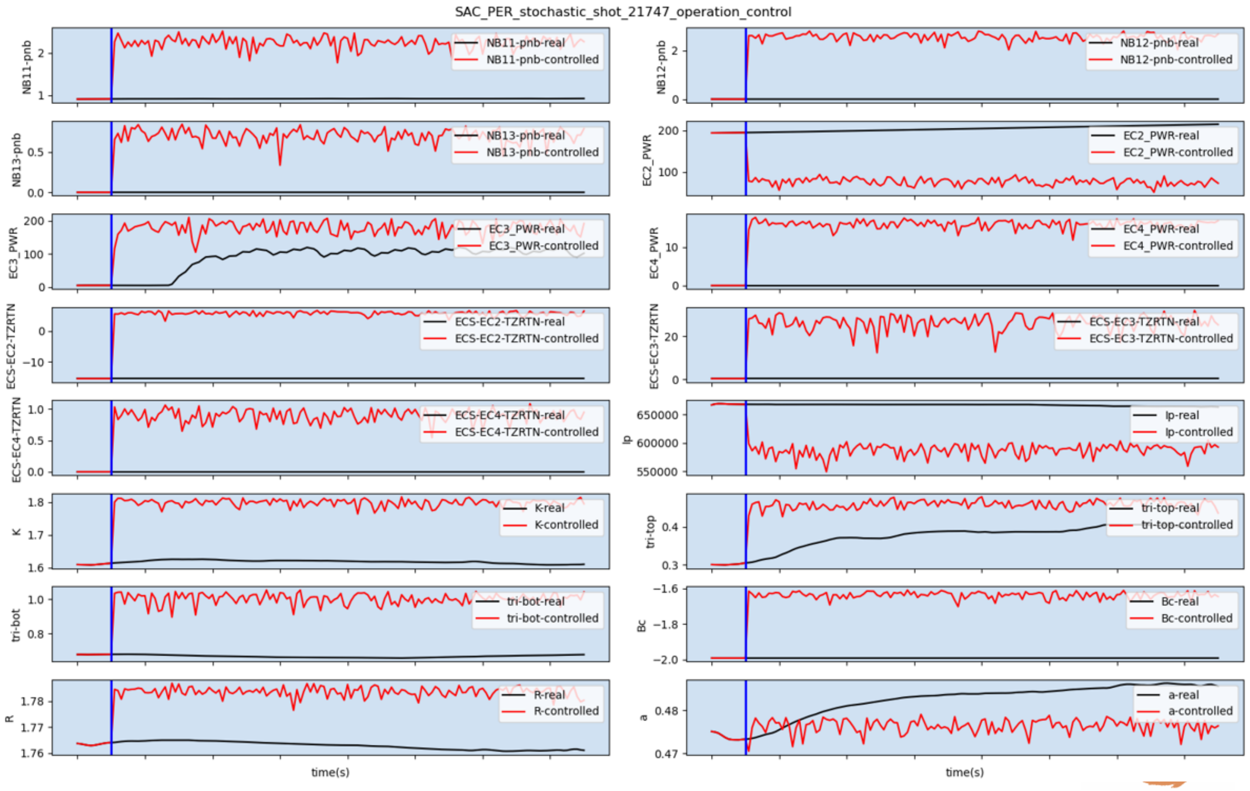
Experimental results

- **0D parameter control in a virtual KSTAR environment (stochastic)**

❑ **<u>Comparison between RL control and real experiment data (Control value)</u>**

- Severe noise in action (control values) is observed in DDPG case.
- Since DDPG is based on a deterministic policy gradient algorithm, it cannot find the optimal policy if the environment is stochastic.





Discussion

- **Issue 1. Is the KSTAR environment non-stationary?**
  - **Stationary**: the statistics or attributes are not changeable over time.
  - **Data distribution shift** causes **non-stationarity**: the statistics of the variables change over time
  - **Non-stationarity** makes the prediction of the 0D parameters **difficult** because there is **no consideration** for **data distribution shift** in the training process.
  - How to **check stationarity** in time series data (global): Augmented Dickey-Fuller test (ADF)
  - How to **detect change points** in timer series data (locally): Maximum Mean Discrepancy (MMD)

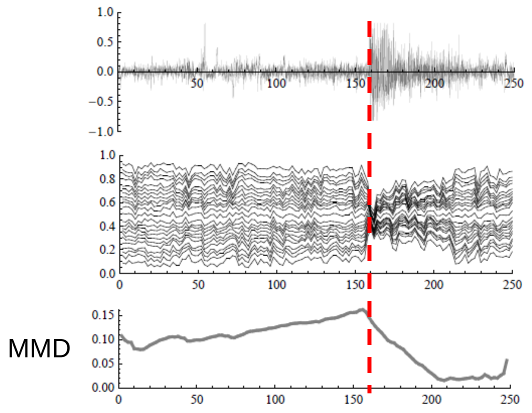


Figure 5: Top: 250 seconds long part of an EEG recording. Middle: Ordinal pattern distributions. Bottom: Resulting MMD values.

Mathieu Sinn et al, AUAI, 2012

Detecting non-stationarity is fine…

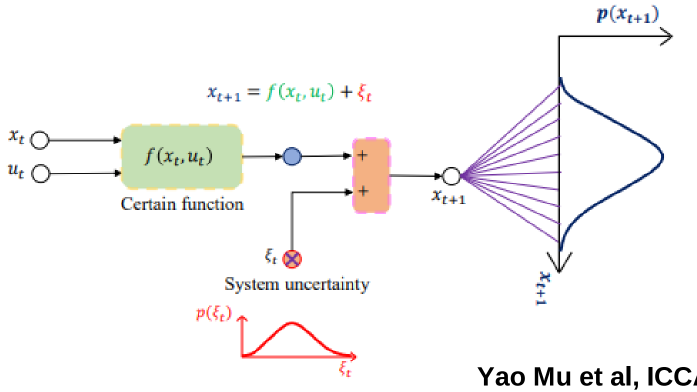
But how can we design this non-stationary environment?

Does the network really learn the non-stationarity of the tokamak environment?

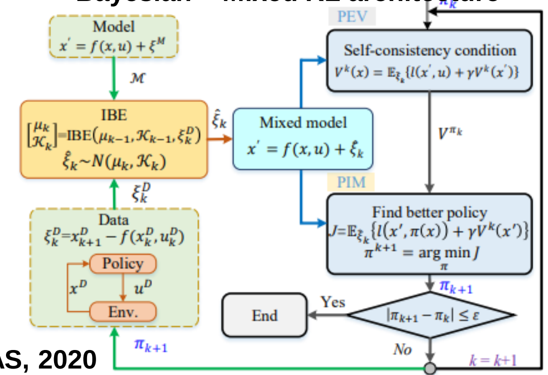
Discussion

- **Issue 2. Is the KSTAR environment stochastic?**
  - **Stochasticity:** the property of being a random process
  - **Stationary process = stochastic process** in which unconditional joint probability distribution **does not change** over time.
  - **Non-stationarity $\neq$ Stochasticity**
  - Since there are not enough measurements or variables which represent the KSTAR plasma state, the virtual KSTAR environment acts as a **partially observable system**: POMDP (**Partially Observable Markov Decision Process**)
  - Latent variables which affect the next state of the plasma: noise/stochasticity
  - Virtual KSTAR environment = Stochastic environment

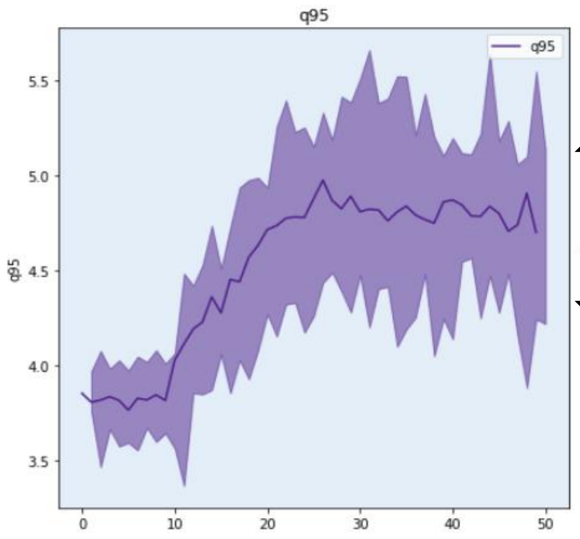
How to learn optimal policy in a stochastic environment?



Bayesian + Mixed RL architecture



Yao Mu et al, ICCAS, 2020

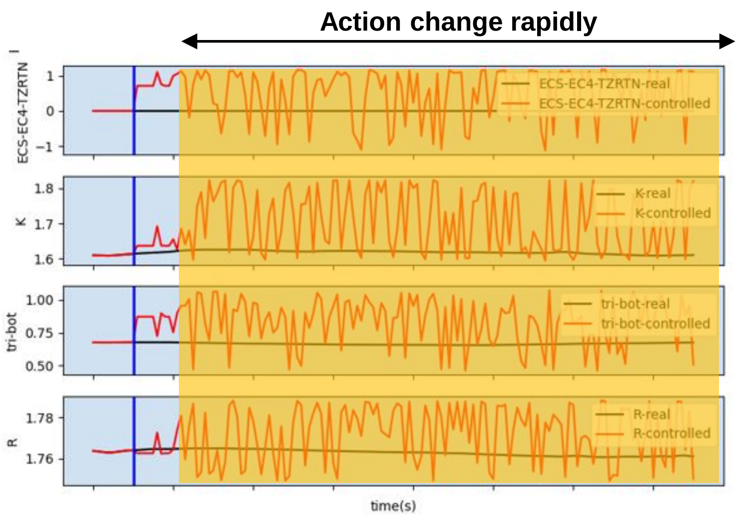


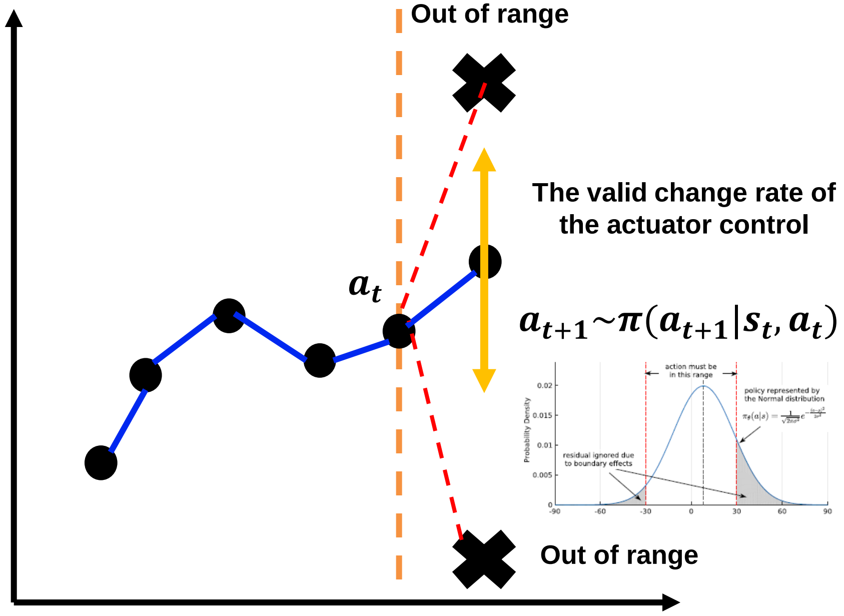
Stochasticity causes a totally different result of the tokamak operation

PLARE

Discussion

- **Issue 3. Can the RL agent reflect the realistic control of the actuator in KSTAR?**
  - The actions (= control values, NBI, EC, Ip, Bc, ..) are bound by physical constraints in a real KSTAR environment, since the change rate of the actuator control is limited.
  - Method 1. **Gaussian policy**-based model can **have bias** induced by the **bounded action range**: Beta distribution?
  - Method 2. The state transition probability may not satisfy the Markov Decision Process: the current RL algorithms can not guarantee their convergence in this situation.





Action space is change over time

Method 1. Action range clipping process

Method 2. Define action as a change rate of the actuator

$$a_{t+1} \sim \pi(a_{t+1}|s_t, a_t)$$
$$u_{t+1} = u_t + a_t,\ s_{t+1} = f(s_t, u_t)$$

 PLARE

Summary

- In this research, 0D parameter control using reinforcement learning under the neural network-based virtual KSTAR environment proceeded.
- However, there are 3 main issues in this research.
  - ✓ Stochasticity should be considered in a virtual KSTAR environment.
  - ✓ Non-stationarity also should be considered in a virtual KSTAR environment.
  - ✓ The controller should reflect the realistic control that actuators can afford
- RL technique which aims for a stochastic environment is needed.
- Since the shape parameters are controlled values originally, the next stage of the research about plasma shape control using reinforcement learning has to proceed.

Thank You

Current status of the research

- **Research scheme: 3-stage development**

❑ **Stage 1: 0D parameter control**

- Data collection : complete
- Code implementation : complete
- Experimental performance : complete

❑ **Stage 2: shape parameter control**

- Data collection : complete
- **Code implementation : proceeding (GS solver / Reconstruction)** → **Numerical vs Data-driven**
- Experimental performance : not yet

Some improvement is needed!

❑ **Stage 3: 0D parameter + shape parameter control**

- Code implementation : not yet
- Experimental performance : not yet

Multi-Agent Reinforcement Learning