INL/MIS-23-75442-Revision-0

Scientific Machine Learning using MOOSE

July 2023

Peter German

Changing the World's Energy Future

Idaho National Laboratory

INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC

INL/MIS-23-75442-Revision-0

Scientific Machine Learning using MOOSE

Peter German

July 2023

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517





July 27, 2023

Peter German,

Dewen Yushu,

Vasileios Kyriakopoulos,

Mauricio Tano

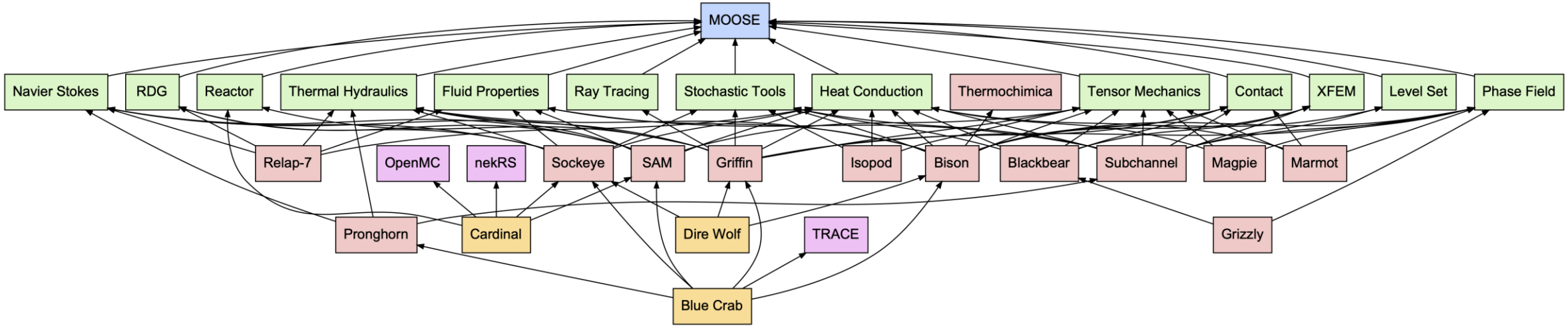
Advances in Scientific Machine Learning in MOOSE

Battelle Energy Alliance manages INL for the
U.S. Department of Energy's Office of Nuclear Energy

Idaho National Laboratory

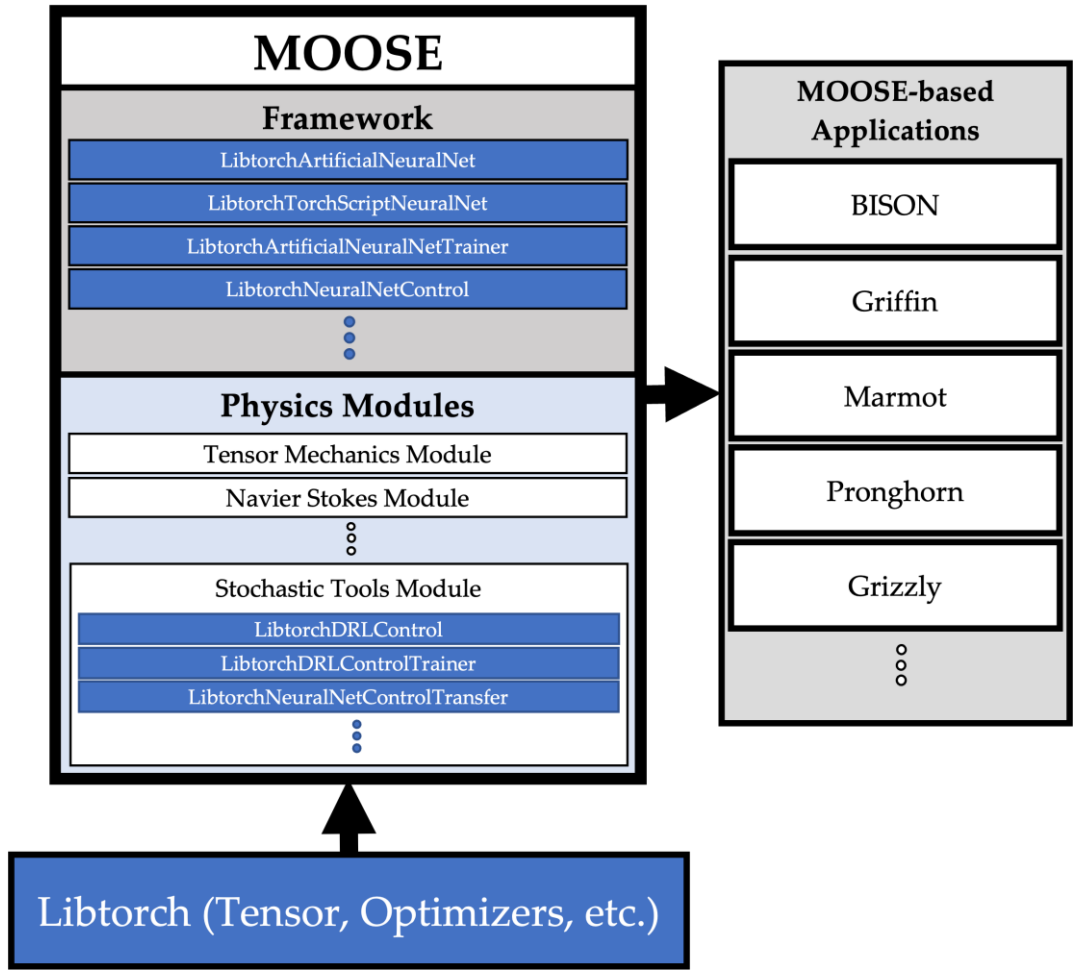
Multiphysics Object-Oriented Simulation Environment (MOOSE)

- MOOSE [1] has a modular structure:
  - Framework
  - Physics modules: Fluid dynamics, elasticity, heat conduction
  - Applications: Nuclear fuel performance (BISON), reactor physics (Griffin)
- LGPL 2.1 license (very permissive)
- https://github.com/idaholab/moose



How can we harvest machine learning in MOOSE?

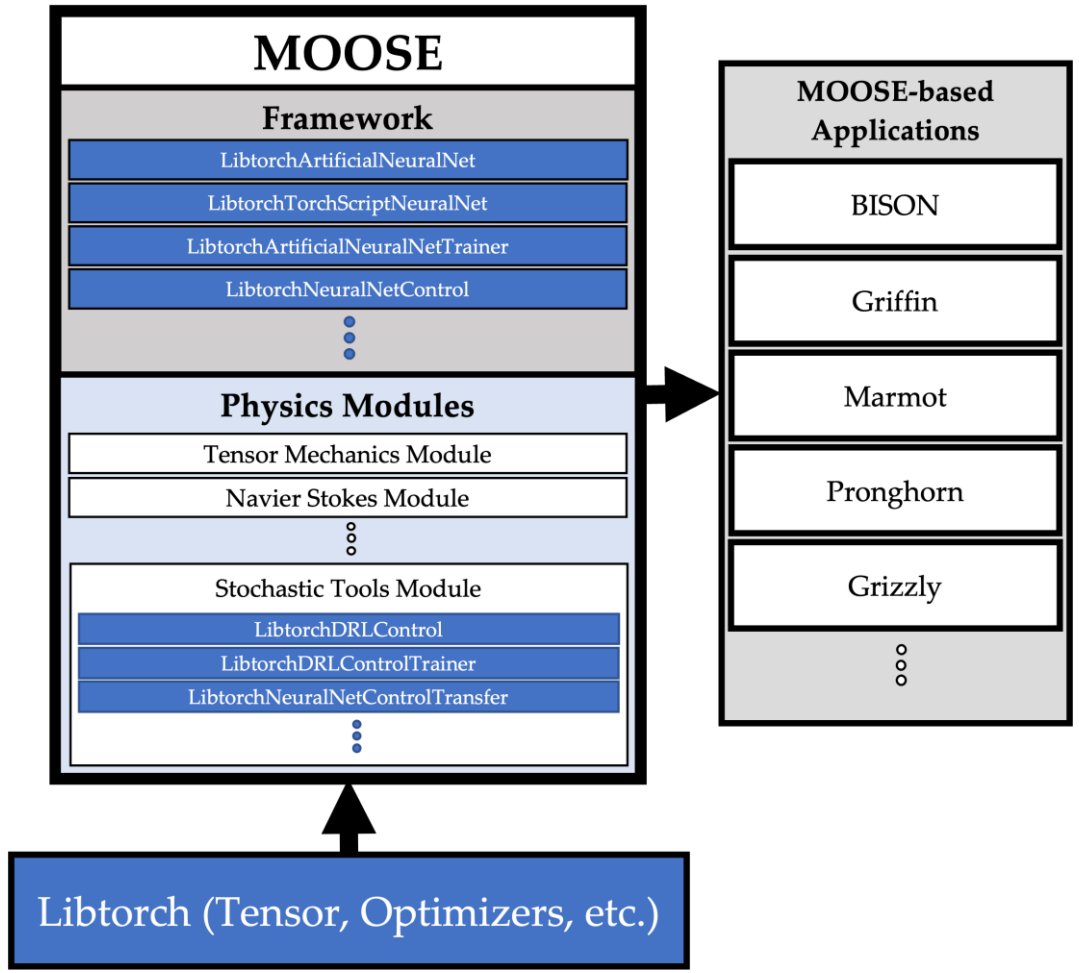
- High-fidelity simulations can be accelerated or (partially) replaced by machine learning models
- Creation of machine-learning-based (ML) controllers for complex multiphysics systems—which is extremely challenging using traditional model-based control—relies on the control system and all existing physics modules in MOOSE
- Could be used to train fast surrogates on the fly to act as predictors in transient problems
- Could be used to train fast surrogates to accelerate sensitivity analysis and uncertainty quantification on complex multiphysics problems



Linking with Libtorch (C++ front end of Pytorch [2])

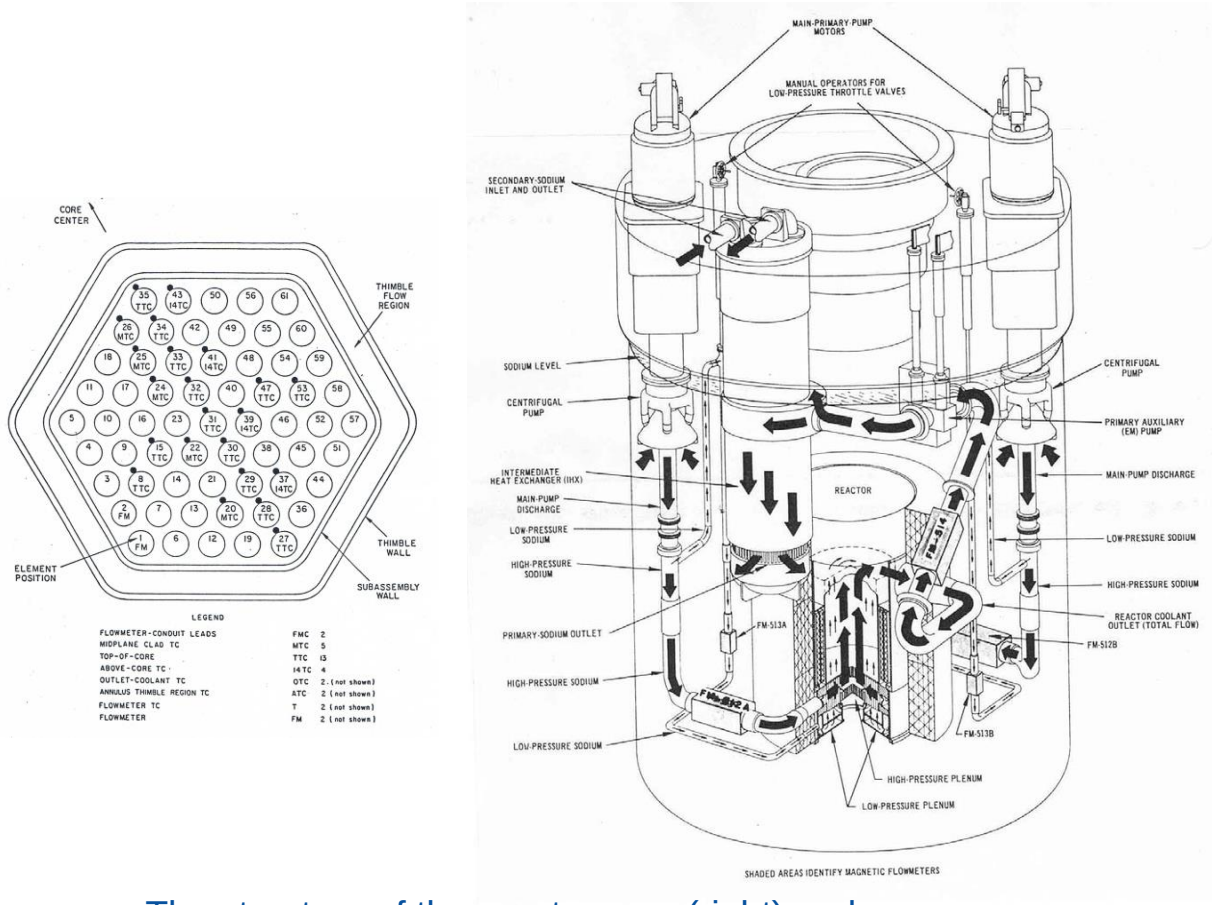
New capabilities

- Framework:
  - General Neural Net (NN) interfaces
  - Enables using networks trained using the python API by reading TorchScript files
  - Implements basic neural net-based controllers
- Stochastic Tools Module
  - Uses NNs for surrogate generation (for sensitivity and uncertainty studies)
  - Enables the use of Proximal Policy Optimization-based Deep Reinforcement Learning [3]



Surrogate generation for flow in nuclear fuel assemblies

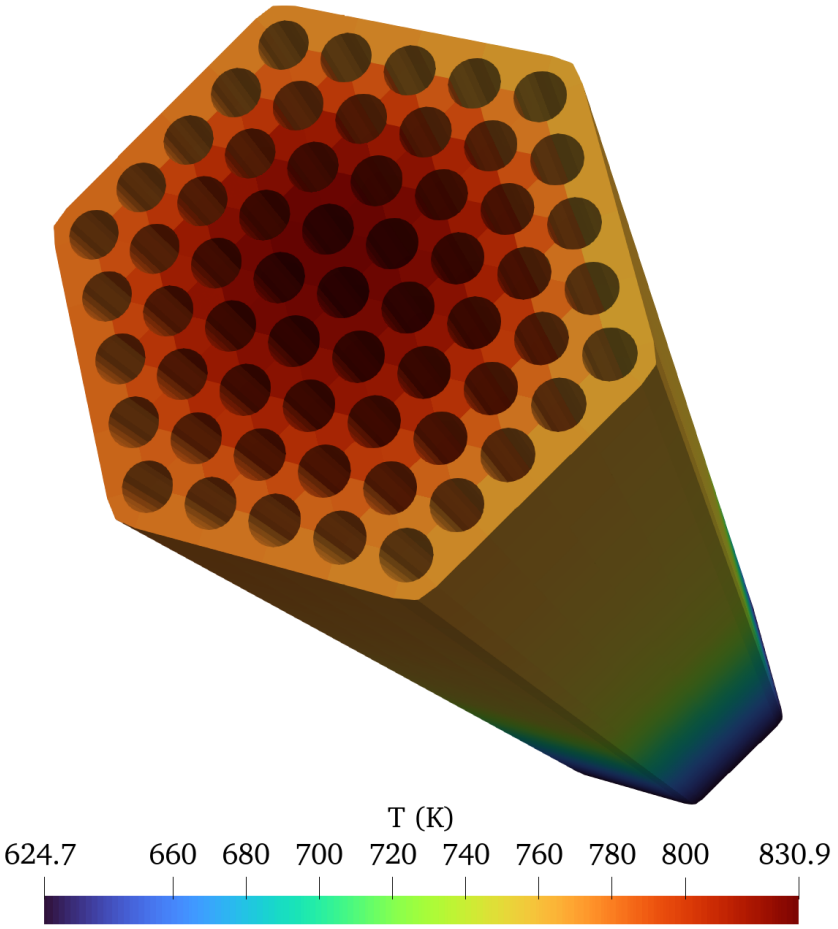
- Experimental Breeder Reactor II (EBR-II) [4] assembly flow simulations
- Quantities of interest:
  - Minimum and maximum temperatures
- Simulation tool: Pronghorn-Subchannel [5]
- Depend on many input parameters:
  - Power tilt factor
  - Power magnitude
  - Inlet mass flow rate
  - Turbulent mixing factor
- Goal: Carry out global sensitivity analysis for the quantities of interest (compute Sobol indices)
- Assumes that the input parameters can change in a +/-15% interval around the expected value



The structure of the reactor core (right) and a fuel assembly (left) of EBR-II [4]

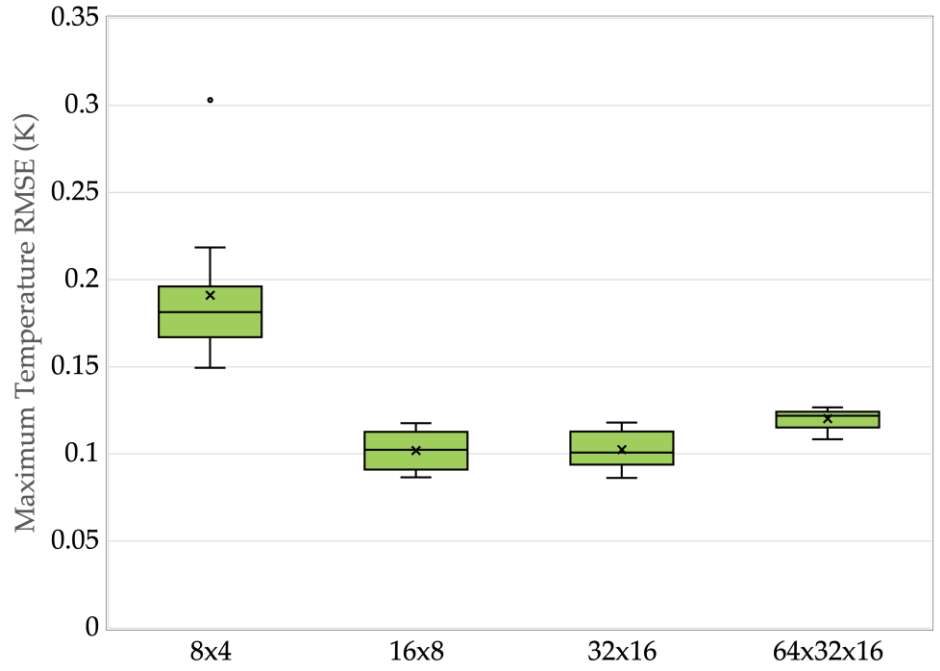
Surrogate generation for flow in nuclear fuel assemblies

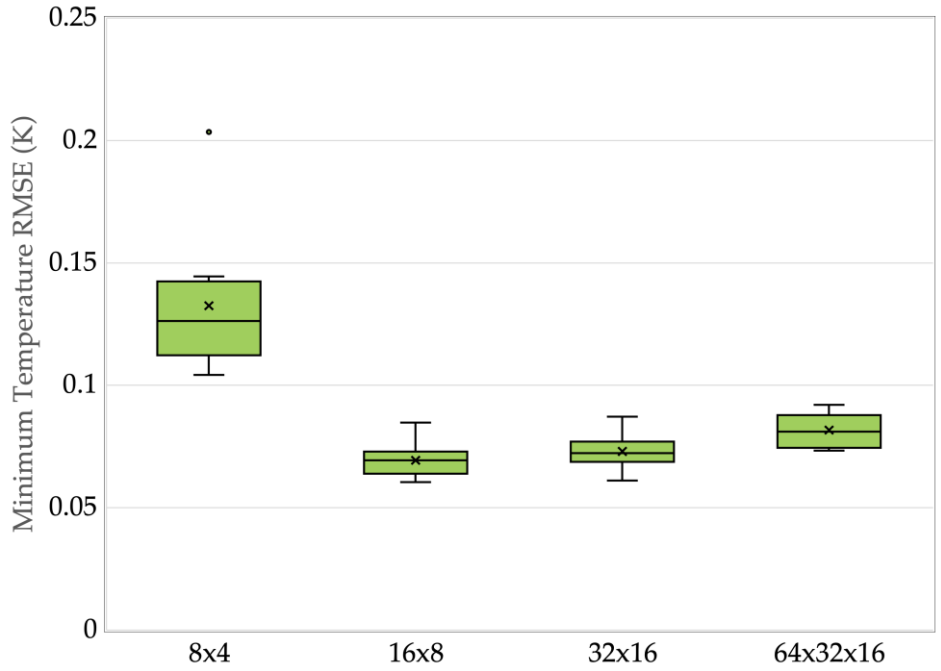
- Issues:
  - For acceptable statistics, one would require a large number of samples
  - One run is relatively expensive (~10^6 degrees of freedom)
- Solution:
  - Build NN-based surrogate → carry out sensitivity study with surrogates
- Using the Stochastic Tools Module
  - 2,000 samples for training, data normalization
  - Train neural networks with different architectures
  - Use three-fold cross-validation for approximating test accuracy (minimize approximation and overfitting errors)



Surrogate generation for flow in nuclear fuel assemblies

- Root Mean Squared Error (RMSE) for different architectures (based on 10 repeated three-fold tests):

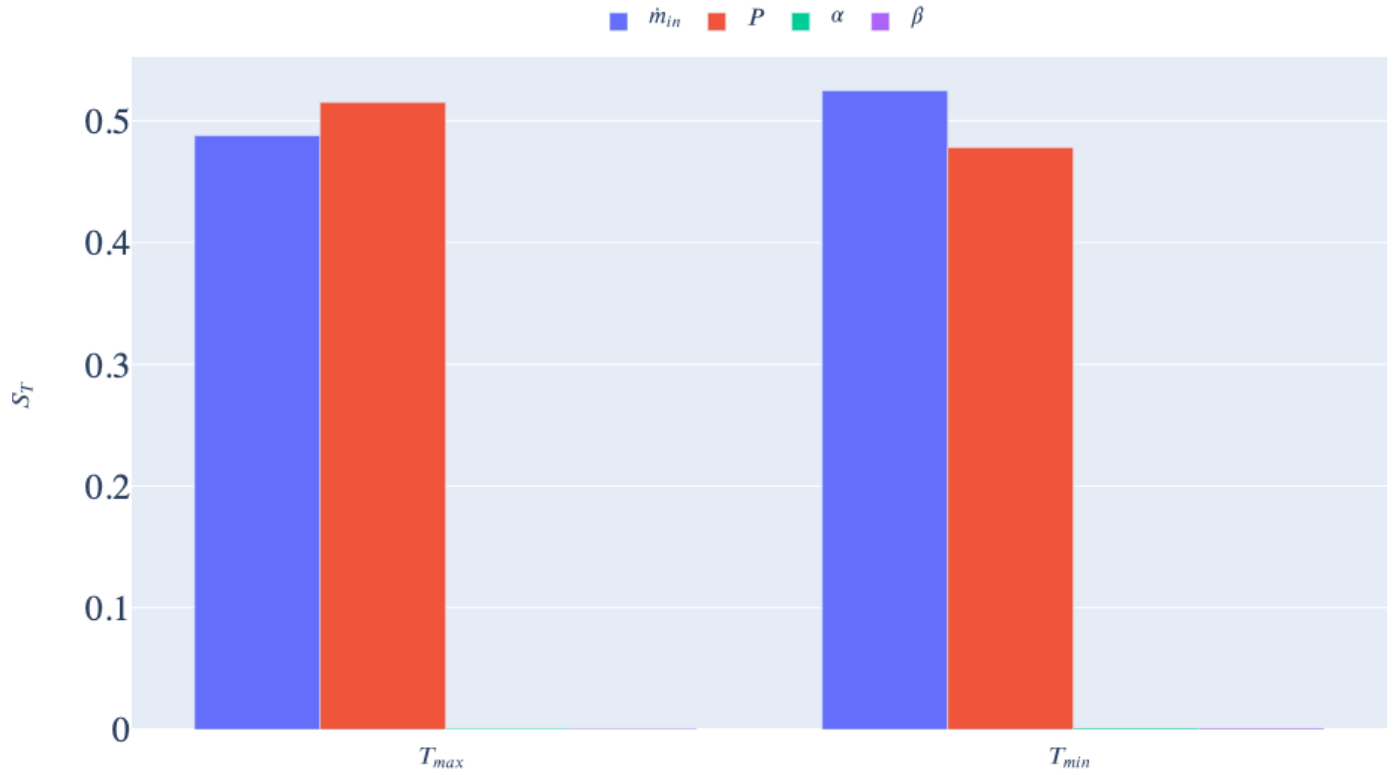




- Simple solution surface, 16 x 8 neural network generalizes well with good accuracy
- Beyond 16 x 8 some overfitting is visible

Surrogate generation for flow in nuclear fuel assemblies

- Total Sobol indices generated using 10^6 samples



- Maximum and minimum temperatures are most sensitive to the inlet mass flow rate and the power

Surrogate generation for flow in nuclear fuel assemblies

- Streamlined input file (plug and play with MOOSE-based models)

```
[Distributions]
  [alpha]
    type = Uniform
    lower_bound = '${fparse 1.8012*0.85}'
    upper_bound = '${fparse 1.8012*1.15}'
  []
  [beta]
    type = Uniform
    lower_bound = '${fparse 0.006*0.85}'
    upper_bound = '${fparse 0.006*1.15}'
  []
  [mass_in]
    type = Uniform
    lower_bound = '${fparse 2.45*0.85}'
    upper_bound = '${fparse 2.45*1.15}'
  []
  [power]
    type = Uniform
    lower_bound = '${fparse 486200*0.85}'
    upper_bound = '${fparse 486200*1.15}'
  []
[]
```

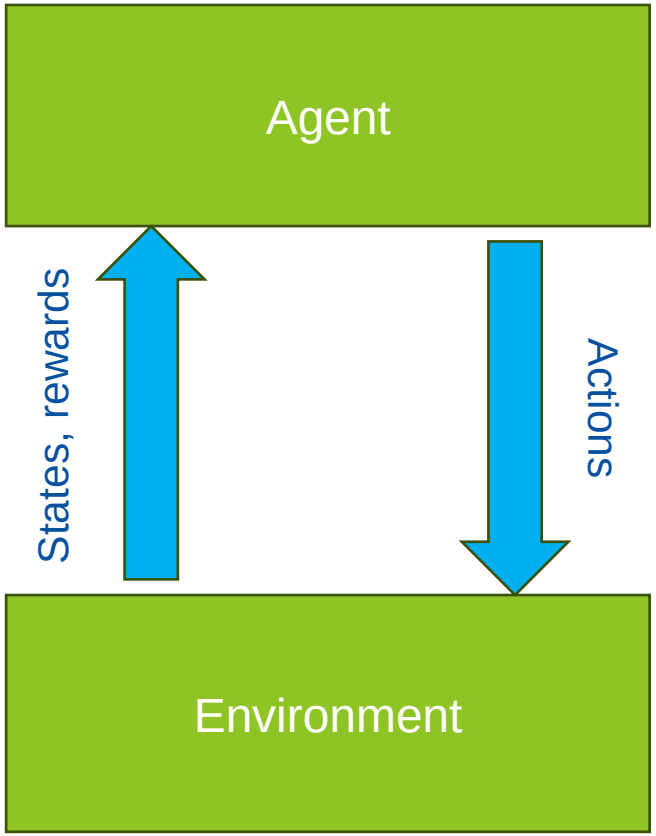
```
[Trainers]
  [nn_max]
    type = LibtorchANNTrainer
    sampler = sample
    response = "results_train/T_max_out:value"
    num_neurons_per_layer = '16 8'
    activation_function = 'relu relu'
    nn_filename = 'weights_max.pt'
    read_from_file = false
    num_epochs = 15000
    num_batches = 20
    learning_rate = 5e-5
    print_epoch_loss = 10
  []
[]
```

```
[Surrogates]
  [nn_max]
    type = LibtorchANNSurrogate
    filename = "train_nn_trainer_nn_max.rd"
  []
[]
```

```
[Samplers]
  [sample]
    type = MonteCarlo
    distributions = 'mass_in power alpha beta'
    num_rows = 200000000
    seed = 0
  []
[]
```

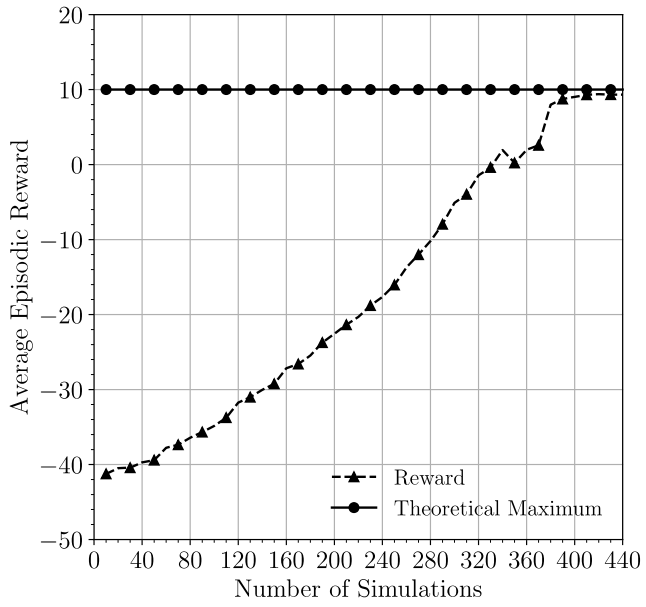
Process control with Deep Reinforcement Learning

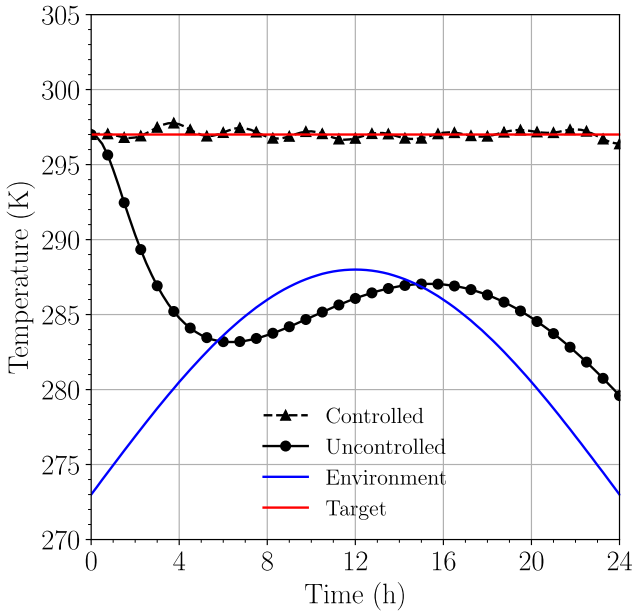
- Proximal Policy Optimization [3]:
  - Agent uses 2 neural networks:
    - Actor (controller)
    - Critic (value-estimator)
- Try to maximize the reward
- Probabilistic action (policy), help with exploration and the reduction of overfitting
- Exercise the environment and see if the actions resulted in higher rewards (if we have an advantage)
- Use the advantage and the probability of the actions to update the controller
- Use clipping for the update to be conservative

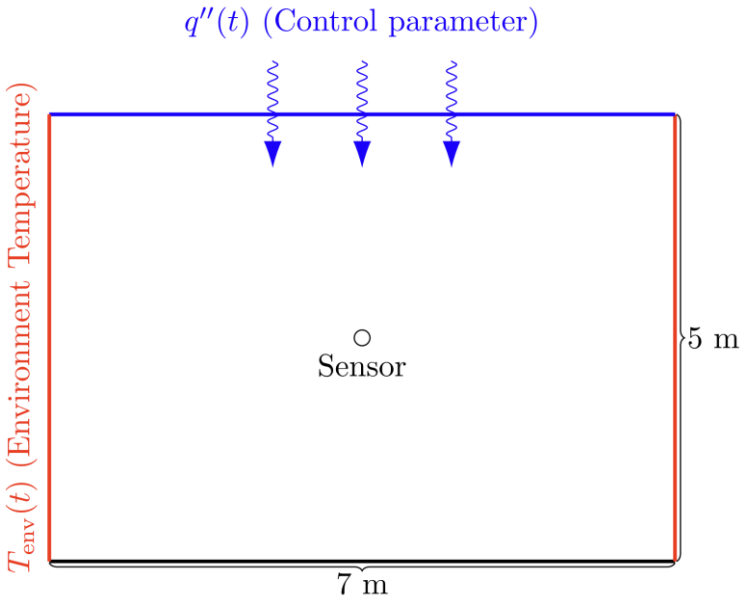


Process control with Deep Reinforcement Learning

- Simple heat conduction problem
  - Try to keep the temperature at the sensor constant
  - Using the heat flux on the top boundary
  - Side boundaries follow the environmental temperature







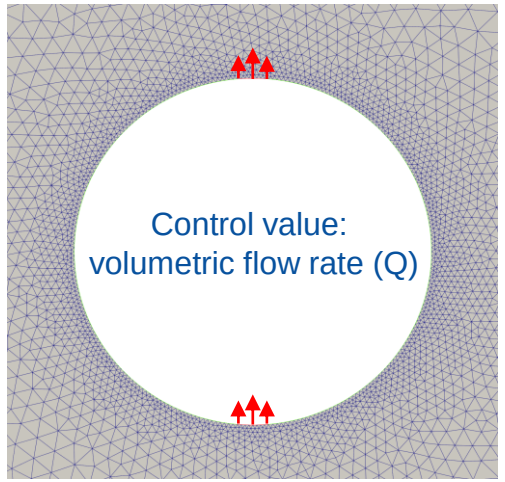
Process control with Deep Reinforcement Learning

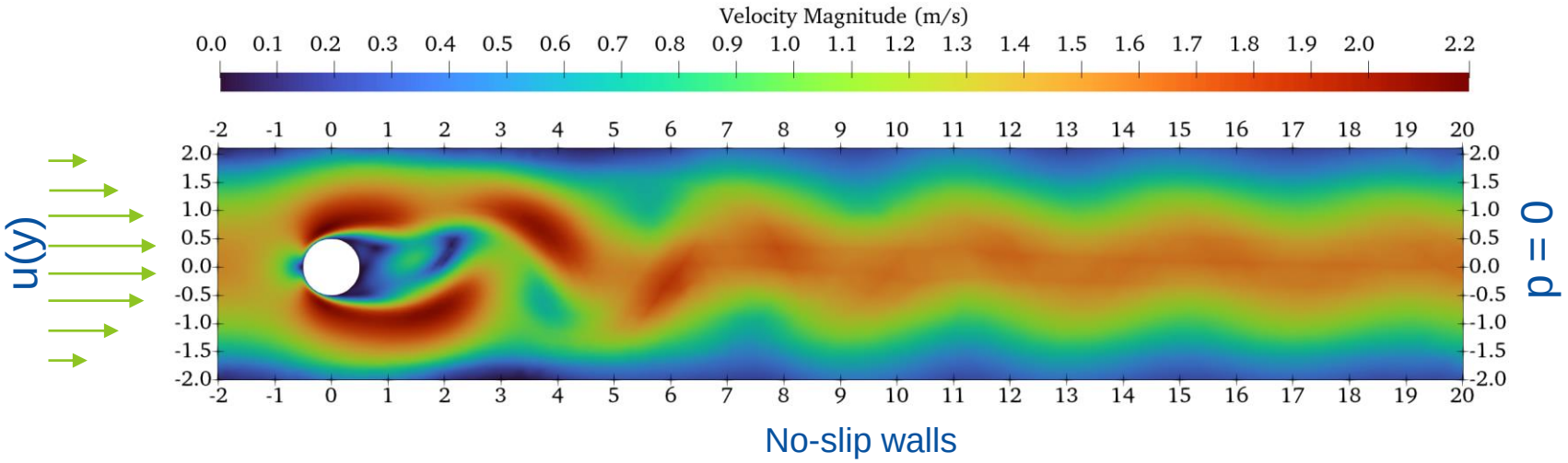
```
[Trainers]
  [nn_trainer]
    type = LibtorchDRLControlTrainer
    response = 'results/center_temp results/env_temp'
    control = 'results/top_flux'
    log_probability = 'results/log_prob_top_flux'
    reward = 'results/reward'

    num_epochs = 1000
    update_frequency = 10
    decay_factor = 0.0

    loss_print_frequency = 10

    critic_learning_rate = 0.0001
    num_critic_neurons_per_layer = '64 27'

    control_learning_rate = 0.0005
    num_control_neurons_per_layer = '16 6'

    # keep consistent with LibtorchNeuralNetControl
    input_timesteps = 2
    response_scaling_factors = '0.03 0.03'
    response_shift_factors = '290 290'
    action_standard_deviations = '0.02'

    standardize_advantage = true

    read_from_file = false
  []
[]
```

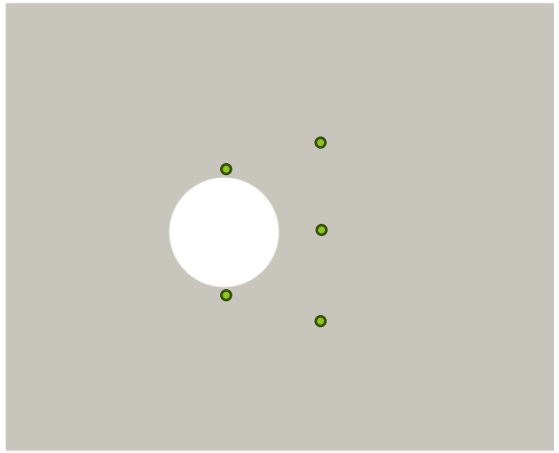
```
[Controls]
  [src_control]
    type = LibtorchDRLControl
    parameters = "BCs/top_flux/value"
    responses = 'center_temp_tend env_temp'

    # keep consistent with LibtorchDRLControlTrainer
    input_timesteps = 2
    response_scaling_factors = '0.03 0.03'
    response_shift_factors = '290 290'
    action_standard_deviations = '0.02'
    action_scaling_factors = 200

    execute_on = 'TIMESTEP_BEGIN'
  []
[]
```

Flow control with Deep Reinforcement Learning

- Benchmark case from [6], uncontrolled case validated using data from [7]
- Fluid dynamics solver: MOOSE Navier Stokes Module
- Mesh: GMSH (~9,300 elements)
- Goal: Minimize the drag without considerable increase in the lift using the small jets on the surface of the body
- Sensors: Five sensors recording pressure and velocity

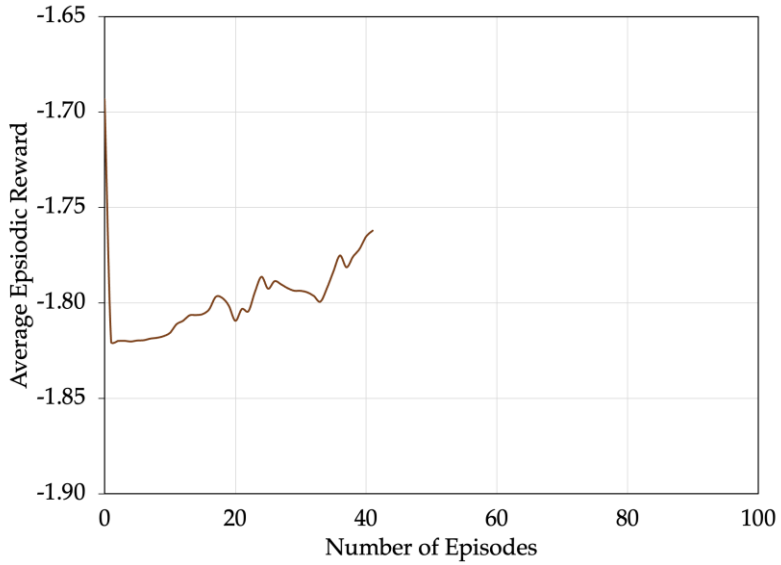


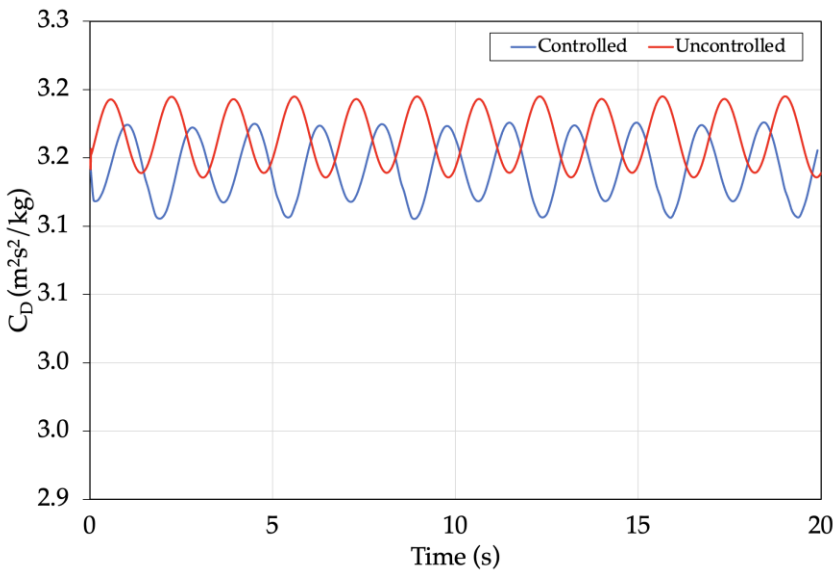




Sensor locations

Flow control with Deep Reinforcement Learning





- Reward function: $r = -\langle C_D \rangle_T - 0.2\langle C_L \rangle_T$ (average quantities over one period)
- Tricks (adapted from [6]):
  - Relax the change in the control value in the Q by $Q^n = Q^{n-1} + 0.2(a - Q^{n-1})$, where a is the action determined by the controller
  - Action a changes every 25 time steps (no quick changes in a)
  - Limit the maximum control value to $0.06\,Q^*$, where Q^* is the volumetric flow rate intercepting the cylinder

Summary

- New Libtorch interface is readily available in MOOSE and any MOOSE-based application
- Can be used for:
  - Surrogate generation for Uncertainty Quantification and Sensitivity Analysis
    - Example provided for EBR-II fuel assembly computations
  - Reinforcement learning for physics controllers
    - Example provided for control aimed at reducing drag
- Future (ongoing) work:
  - Reinforcement learning for additive manufacturing
  - Advanced material models for plasticity

References

[1] Lindsay, A. D., et al. 2020. “2.0-MOOSE: Enabling massively parallel multiphysics simulation.” *SoftwareX* 20: 101202. https://doi.org/10.1016/j.softx.2022.101202.

[2] Paszke, A., et al. 2019. “Pytorch: An imperative style, high-performance deep learning library.” *Advances in Neural Information Processing Systems* 32. https://doi.org/10.48550/arXiv.1912.01703

[3] Schulman, J., et al. 2017. “Proximal Policy Optimization Algorithms.” *https://doi.org/10.48550/arXiv.1707.06347*.

[4] Lentz, G. L., H. W. Buschman, and R. N. Smith. 1985. “EBR-II: twenty years of operating experience.” CONF-850722-1, Argonne National Laboratory. https://www.osti.gov/servlets/purl/5434129.

[5] Kyriakopoulos, V., M. E. Tano, and J. C. Ragusa. 2022. “Development of a Single-Phase, Transient, Subchannel Code, within the MOOSE Multi-Physics Computational Framework.” *Energies* 15(11): 3948. https://doi.org/10.3390/en15113948.

[6] Rabault, J., et al. 2019. "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control." *Journal of Fluid Mechanics* 865: 281–302. https://doi.org/10.1017/jfm.2019.62.

[7] Schäfer, M., S. Turek, F. Durst, E. Krause, and R. Rannacher. 1996. “Benchmark computations of laminar flow around a cylinder.” In Hirschel, E.H. (eds) Flow Simulation with High-Performance Computers II. Vieweg+Teubner Verlag. https://doi.org/10.1007/978-3-322-89849-4_39.

Idaho National Laboratory

Battelle Energy Alliance manages INL for the U.S. Department of Energy's Office of Nuclear Energy. INL is the nation's center for nuclear energy research and development, and also performs research in each of DOE's strategic goal areas: energy, national security, science and the environment.

Surrogate generation for flow in nuclear fuel assemblies

- Sobol expansion:

$$Y = f(X_1, X_2, \dots, X_k) = f_0 + \sum_{i}^{k} f_i(X_i) + \sum_{i} \sum_{j<i} f_{ij}(X_i, X_j) + \cdots + f_{12\dots k}(X_1, X_2, \dots, X_k)$$

- If each term has a zero mean
  - The functions are pair-wise orthogonal
  - The terms can be determined using conditional expectations from low to high orders
  - The total variance can be expressed as:

$$\mathrm{Var}(f) = \sum_{i}^{k} \mathrm{Var}(f_i) + \sum_{i} \sum_{j<i} \mathrm{Var}(f_{ij}) + \cdots + \mathrm{Var}(f_{12\dots k})$$

- Sobol indices:
  - First order: $S_i = \frac{\mathrm{Var}(f_i)}{\mathrm{Var}(f)}$
  - Total: $S_{T,i} = S_i + \sum_{j}^{k} S_{ij} + \cdots + S_{ij\dots k}$
- Compute these using Saltelli's Monte Carlo methods

Saltelli, Andrea, et al. *Sensitivity analysis in practice: a guide to assessing scientific models*. Vol. 1. New York: Wiley, 2004.