INL/CON-23-73756-Revision-0

Adaptive, Active Learning, and Multifidelity Monte Carlo Methods in the MOOSE Stochastic Tools Module

July 2023

Changing the World's Energy Future

Som LakshmiNarasimha Dhulipala, Zachary M Prince, Peter German

INL/CON-23-73756-Revision-0

Adaptive, Active Learning, and Multifidelity Monte Carlo Methods in the MOOSE Stochastic Tools Module

Som LakshmiNarasimha Dhulipala, Zachary M Prince, 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

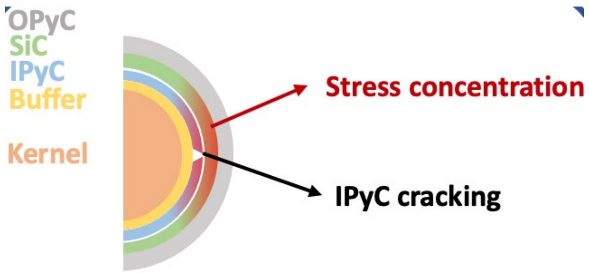
Adaptive, Active Learning, and Multifidelity Monte Carlo Methods in the MOOSE Stochastic Tools Module

Zach Prince, Som Dhulipala, Peter German

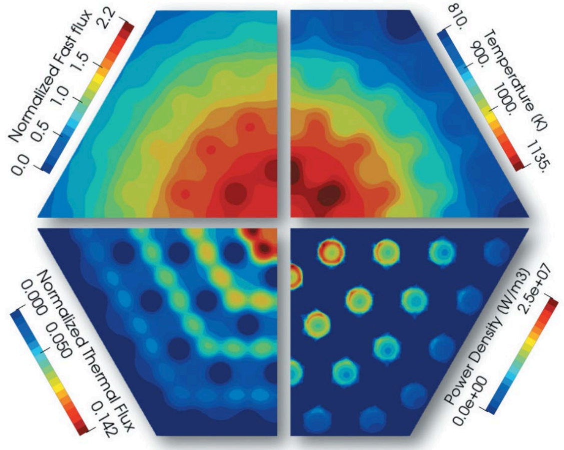
17th US National Congress on Computational Mechanics

Idaho National Laboratory

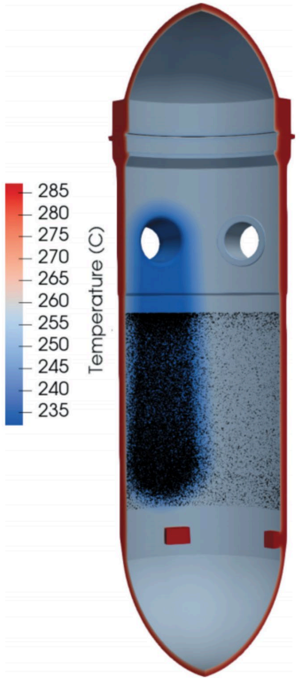
Motivation



TRISO advanced nuclear fuel model (Jiang et al. 2021)



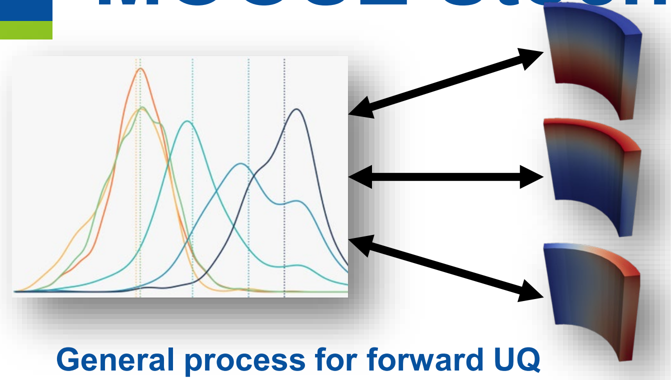
Reactor core simulation of a heat-pipe microreactor (Matthews et al. 2021)



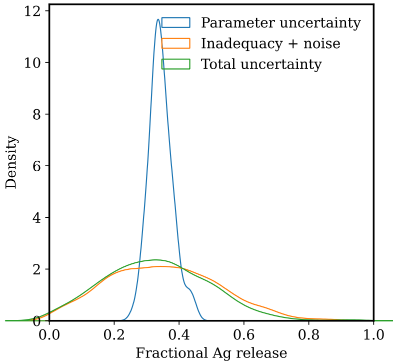
Flaws simulation in a reactor pressure vessel (Spencer et al. 2021)

- In safety-critical fields such as nuclear engineering, failure characterization of advanced reactor technologies considering different sources of uncertainties is important
- Often, failure characterization of advanced reactor technologies and their design optimization involves solving a rare events problem
- Examples: TRISO nuclear fuel, heat-pipe microreactor thermal stresses, reactor pressure vessel embrittlement
- We discuss **recent developments to MOOSE stochastic tools module (STM)** for efficient characterization of rare events

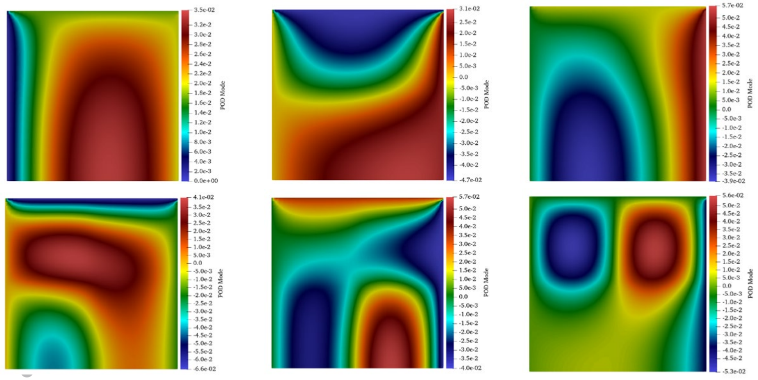
MOOSE Stochastic Tools Module



General process for forward UQ



UQ on modeling TRSIO Particle Ag release



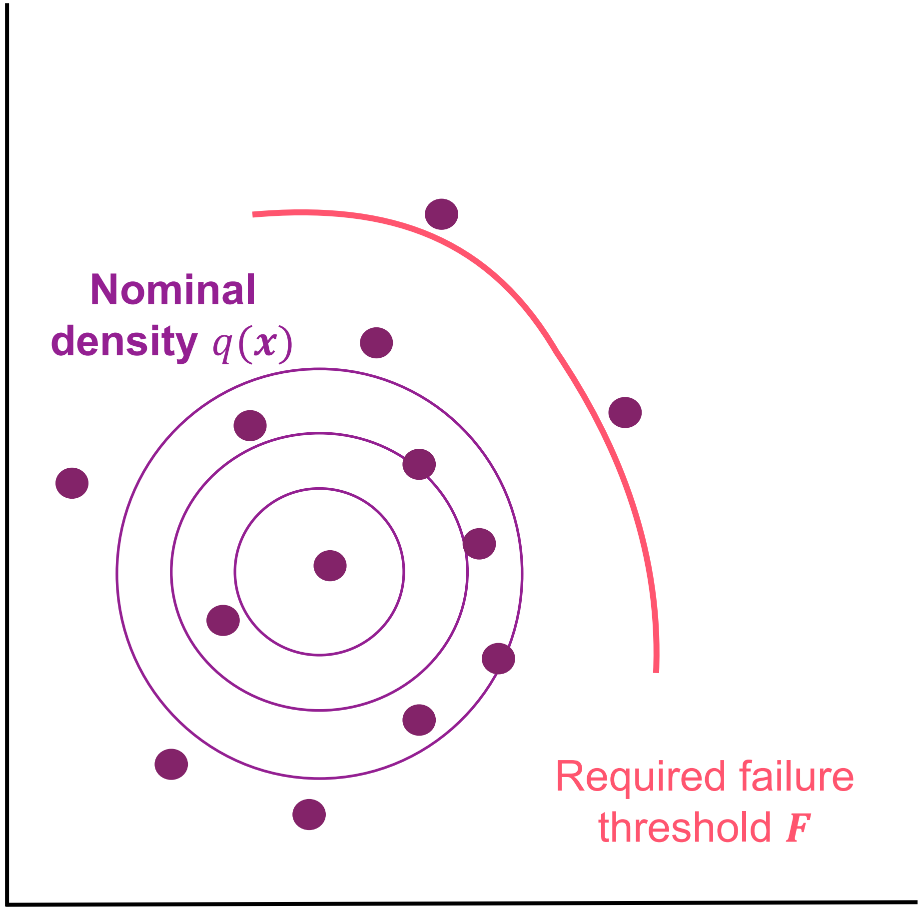
POD modes of a 2D heat conduction problem

- Provide a **MOOSE interface** for performing stochastic analysis on MOOSE-based models.
- Sample parameters, run applications, and gather data that is both **efficient** (memory and runtime) and **scalable**.
- Perform UQ and sensitivity analysis with **distributed data** with **advanced variance reduction** methods
- **Parallel Scalable Inverse Bayesian UQ** for parameter and model error estimation
- Train meta-models to develop fast-evaluating **surrogates** of the high-fidelity multiphysics model
  - Harness advanced machine learning capabilities through the C++ front end of Pytorch [1]
  - Use active learning models for building surrogates
- Provide a **pluggable** interface for these surrogates.
- Use **POD (Proper Orthogonal Decomposition)-based dimensionality reduction** methods to build mappings between solution variables and latent (low-dimensional) spaces

Sampling for rare events estimation: Monte Carlo

$$P_f = \int_{\widetilde{F}(\mathbf{X})>\mathcal{F}} q(\mathbf{X})\, d\mathbf{X} \qquad P_f \approx \hat{P}_f = \frac{1}{N_m} \sum \mathbf{I}(\widetilde{F}(\mathbf{X}) > \mathcal{F})$$

(*F: Failure threshold; F(**X**): Model output; q(**X**): input distributions*)



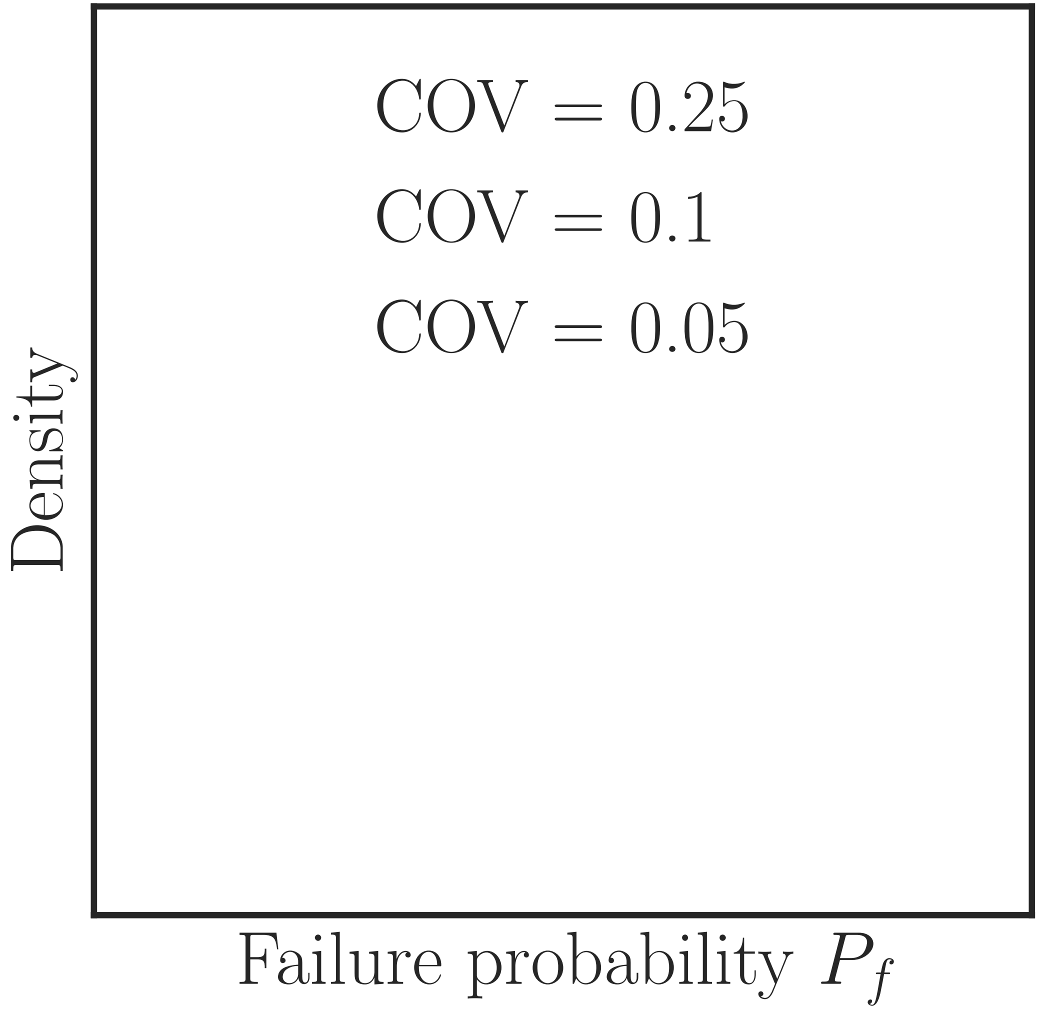
MOOSE STM

```
[Samplers]
  [sample]
    type = MonteCarlo
    num_rows = 100 # Number of Monte
    distributions = 'normal_kernel_r
    execute_on = 'PRE_MULTIAPP_SETUP'
  []
[]
```

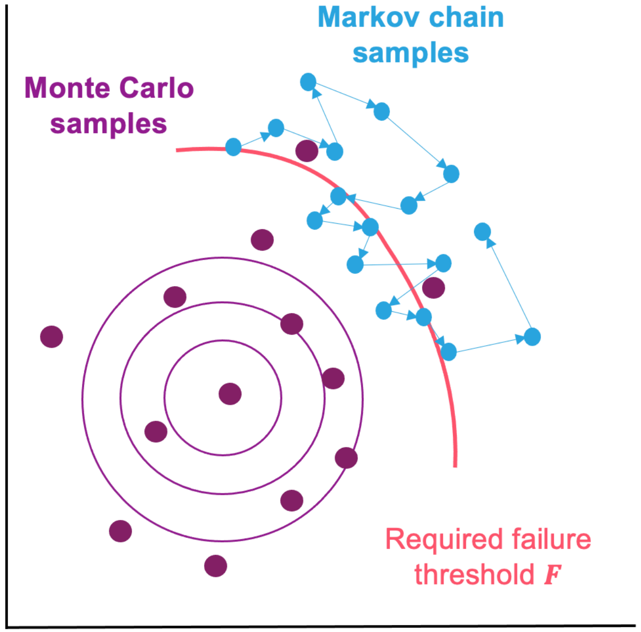
Computationally intractable

- For 2D TRISO fuel model with a Pf of 1E-4, Monte Carlo requires ~500 hours on 1000 processors (10% coefficient of variation)
- If the Pf is 1E-5, Monte Carlo requires ~5000 hours

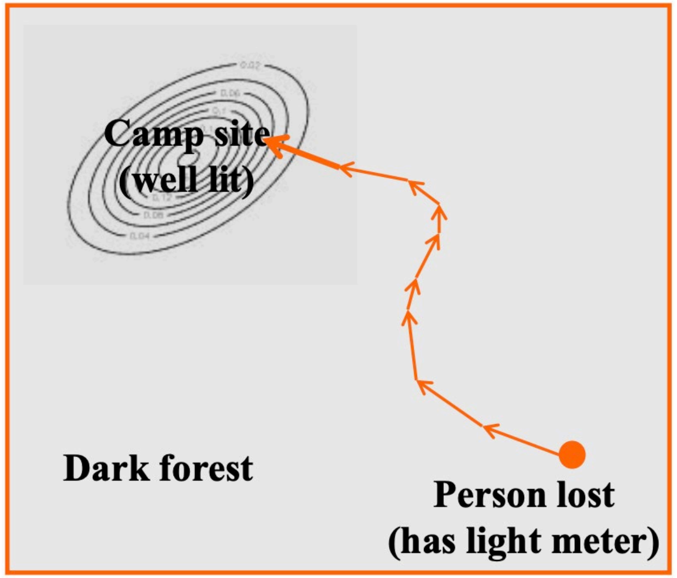
Coefficient of variation of failure probability



Principle of a Markov chain

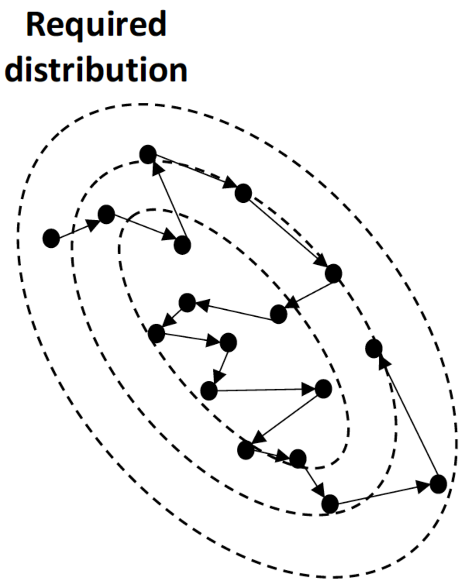


MCMC analogy

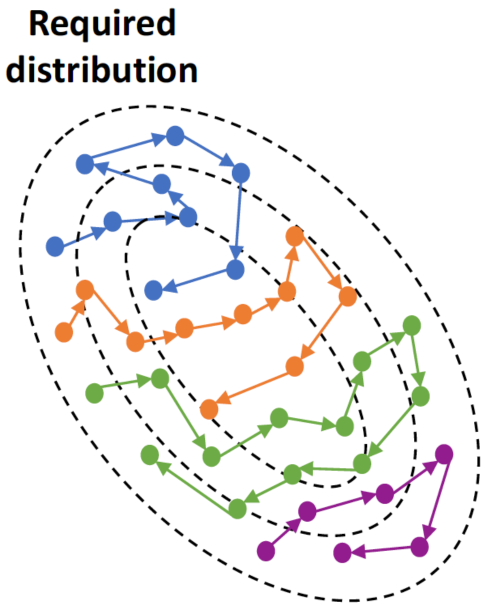


- Conditional samples: $q(\boldsymbol{X}|F(\boldsymbol{X}) > F\ )$
- **Dark forest:** Parameter space
- **Well lit camp site:** Required distribution to be sampled from
- **Light meter:** Acceptance ratio (or transition operator)
- Metropolis-Hastings: popular

Serial and parallel Markov chains



A Markov chain



Multiple Markov chains
(shown in different colors)

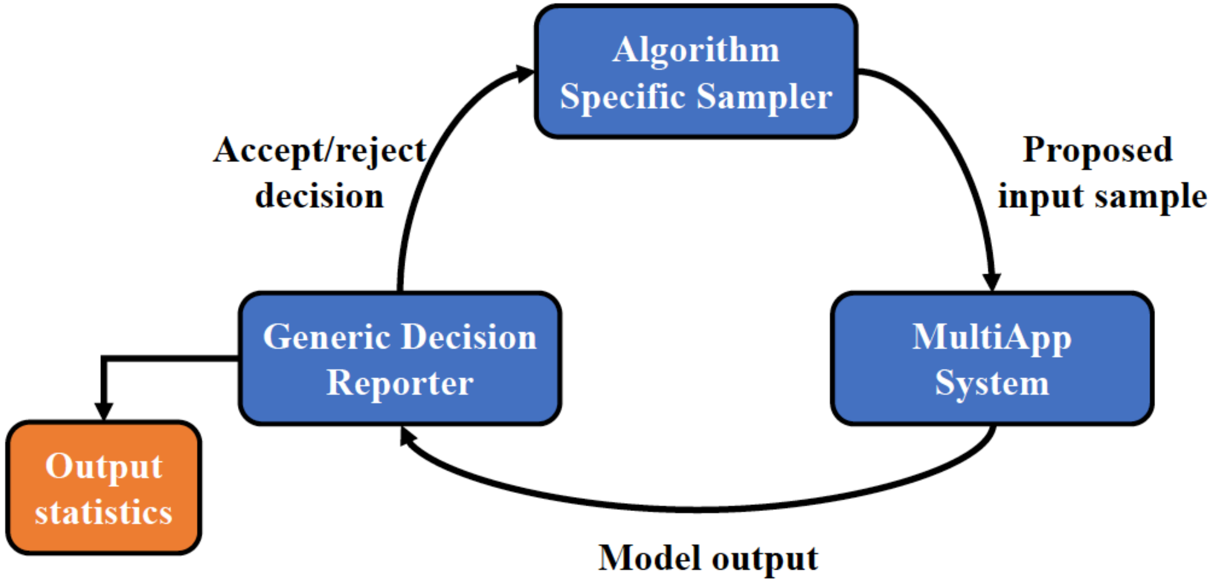
Serial: Single Markov chain
Executed on several processors sample by sample

Parallel: Multiple Markov chains
Executed on several sets of processors

Important: Parallelization is only achieved across chains but not within a chain.

Samples within a Markov chain are history dependent

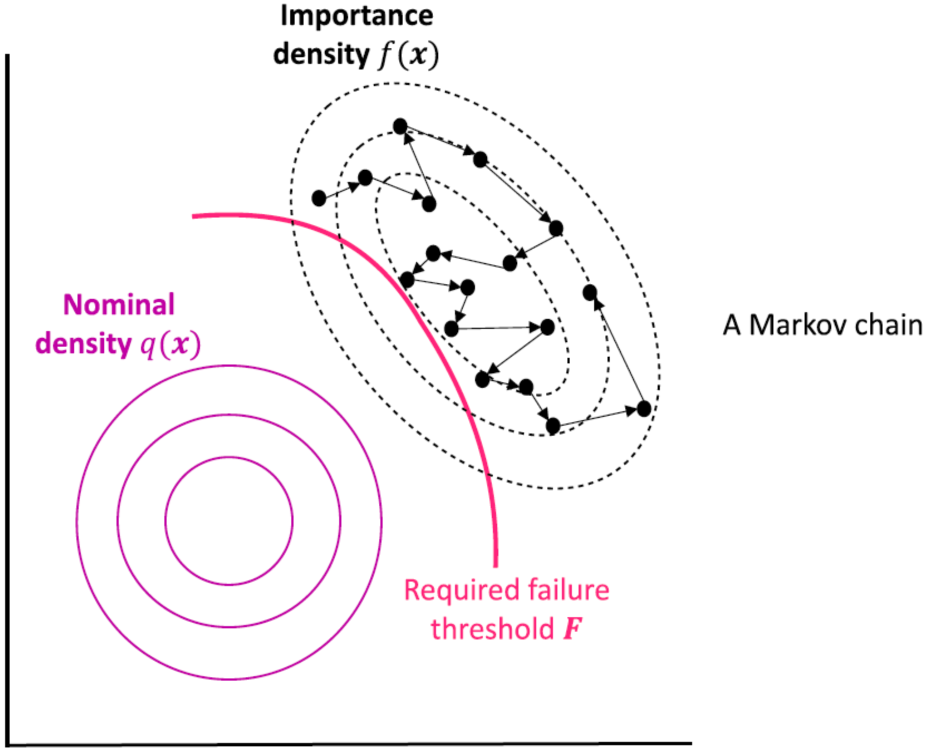
MOOSE STM design for simulating Markov chains



- Three steps for simulating Markov chains:
  1. Propose a random input sample based on the previous sample
  2. Evaluate the FE model
  3. Accept/reject sample based on FE model output

- **Sampler:** Proposes a random input sample centered around a previously accepted sample. All parameters transformed to standard Normal space.
- **MultiApp:** Evaluate FE model based on the sampler input. Parallelization
- **Reporter:** Consume the subapp output and decide on whether to accept/reject the sampler input proposal
- **Statistics Reporter**

Adaptive importance sampler: theory



$$\hat{P}_f^{\text{AIS}} = \frac{1}{N} \sum_{i=1}^{N} I(F_i(\boldsymbol{x}) \geq \mathcal{F}) \; \frac{q(\boldsymbol{x})}{f(\boldsymbol{x})}$$

$$\text{Var}(\hat{P}_f^{\text{AIS}}) = \frac{1}{N} \left\{ \frac{1}{N} \sum_{i=1}^{N} \left[I(F_i(\boldsymbol{x}) \geq \mathcal{F}) \; \frac{q(\boldsymbol{x})}{f(\boldsymbol{x})} \right]^2 - (\hat{P}_f^{\text{AIS}})^2 \right\}$$

Proposed by Au and Beck (1999)

- Consists of two phases: training phase and sampling phase
- Transformation to standard Normal space
- **Training phase:** Use Markov chain to sample such that the model always fails
- Construct the importance density [f(x)] using the Markov chain samples
- **Sampling phase:** Sample from the importance density
- Evaluate failure probability and variance using the required equations using samples from the second phase

Adaptive importance sampler: MOOSE STM

Sampler block

```
[Samplers]
  [sample]
    type = AdaptiveImportance
    distributions = 'mu1 mu2'
    proposal_std = '1.0 1.0'
    output_limit = 0.45
    num_samples_train = 30
    std_factor = 0.9
    initial_values = '-0.103 1.239'
    inputs_reporter = 'adaptive_MC/inputs'
  []
[]
```

MultiApp block

```
[MultiApps]
  [sub]
    type = SamplerFullSolveMultiApp
    input_files = sub.i
    sampler = sample
  []
[]
```

Reporter block

```
[Reporters]
  [constant]
    type = StochasticReporter
  []
  [adaptive_MC]
    type = AdaptiveMonteCarloDecision
    output_value = constant/reporter_transfe
    inputs = 'inputs'
    sampler = sample
  []
[]
```

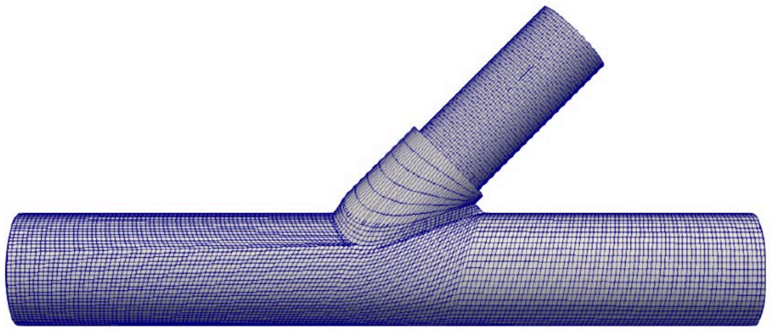
Executioner block (total samples)

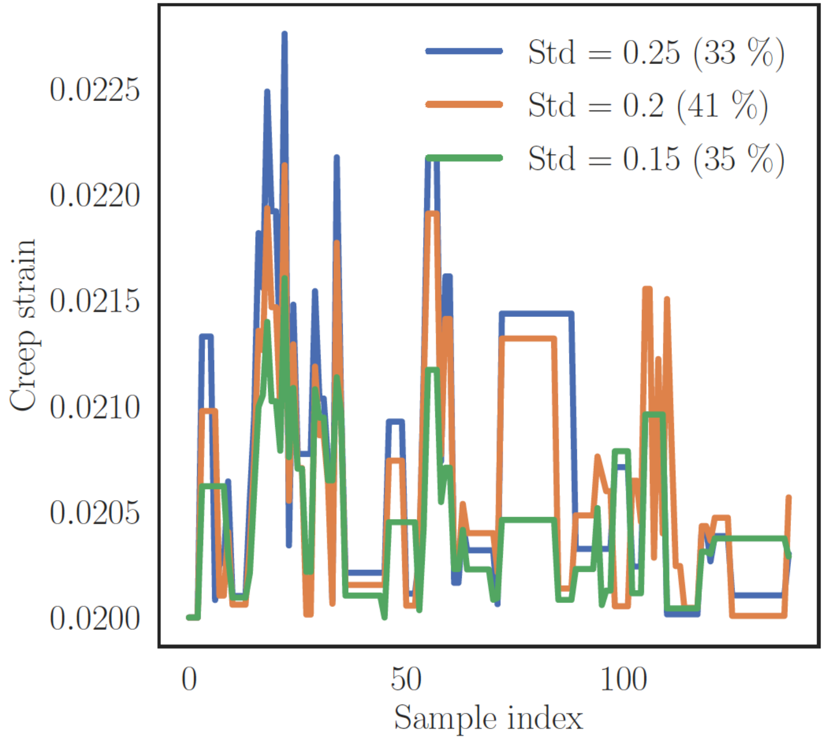
```
[Executioner]
  type = Transient
  num_steps = 60
[]
```

Output JSON file


```
{
    "adaptive_MC": {
        "inputs": [
            [
                -0.22230051785267368
            ],
            [
                1.1535106629452407
            ]
        ],
        "output_required": [
            1.0
        ]
    },
    "constant": {
        "reporter_transfer:average:value": [
            0.45366656803460575
        ],
        "reporter_transfer:converged": [
            true
        ]
    },
    "time": 33.0,
    "time_step": 33
},
```

High temperature creep response of a nuclear alloy

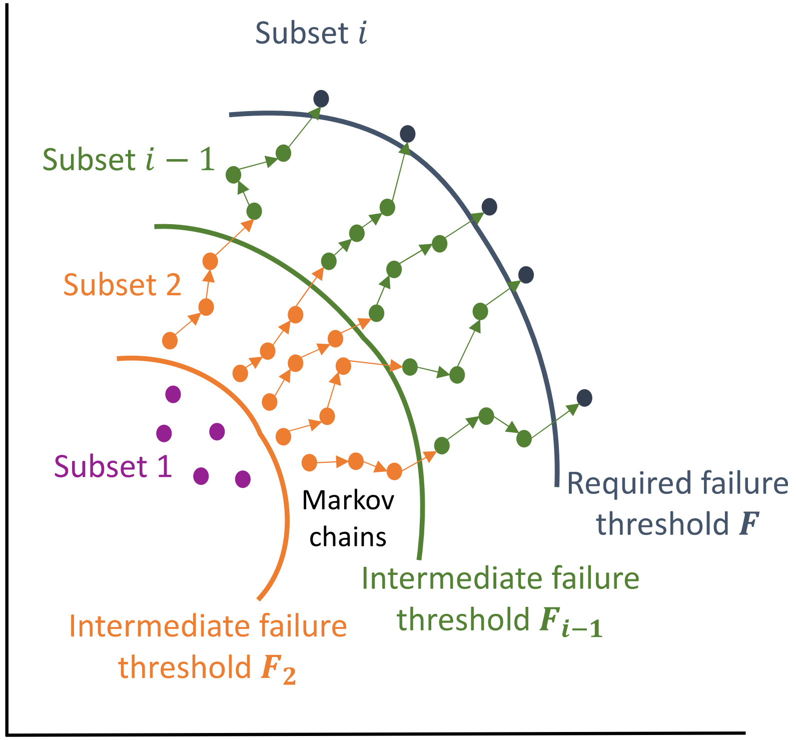




- A pipe component in a nuclear reactor exposed to high temperatures. High temperature creep simulations.
- Computational model is expensive, and the creep model has uncertain parameters.
- We want to sample the input parameters such that the creep strain exceeds 2%.
- Regular Monte Carlo takes many model evaluations under random material params so that the 2% creep strain is exceeded sufficiently
- Adaptive Importance Sampler samples from the failure region

(example courtesy of Lynn Munday, Ben Spencer)

Parallel subset simulation: theory



$$P_f = P(F_1)\prod_{i=2}^{N} P(F_i|F_{i-1})$$

Proposed by Au and Beck (2001)

- AIS limitations: Serial execution; starting sample
- PSS: Expresses small failure probabilities as a product of larger conditional probabilities (of the order 0.1)
- Creates intermediate failure thresholds before the required failure threshold
- An intermediate failure threshold is defined as the (1-x) percentile value of the samples in previous conditional level
- First conditional level: Direct Monte Carlo
- Subsequent conditional levels: Markov Chain Monte Carlo
- **Uses 100s of Markov chains which can be executed in parallel**

Parallel subset simulation: MOOSE usage

Sampler block

```
[Samplers]
  [sample]
    type = ParallelSubsetSimulation
    distributions = 'mu1 mu2'
    num_samplessub = 20
    num_parallel_chains = 2
    output_reporter = 'constant/reporter_transfer
    inputs_reporter = 'adaptive_MC/inputs'
  []
[]
```

Reporter block

```
[Reporters]
  [constant]
    type = StochasticReporter
    outputs = none
  []
  [adaptive_MC]
    type = AdaptiveMonteCarloDecision
    output_value = constant/reporter_transfer
    inputs = 'inputs'
    sampler = sample
  []
[]
```

Output JSON file

```
{
  "adaptive_MC": {
      "inputs": [
          [
              0.4756234757587354,
              0.23574626132198995
          ],
          [
              1.6069412882448921,
              1.7647270585523014
          ]
      ],
      "output_required": [
          1.0145830285171529,
          0.9745897531875222
      ]
  },
  "time": 13.0,
  "time_step": 13
},
```

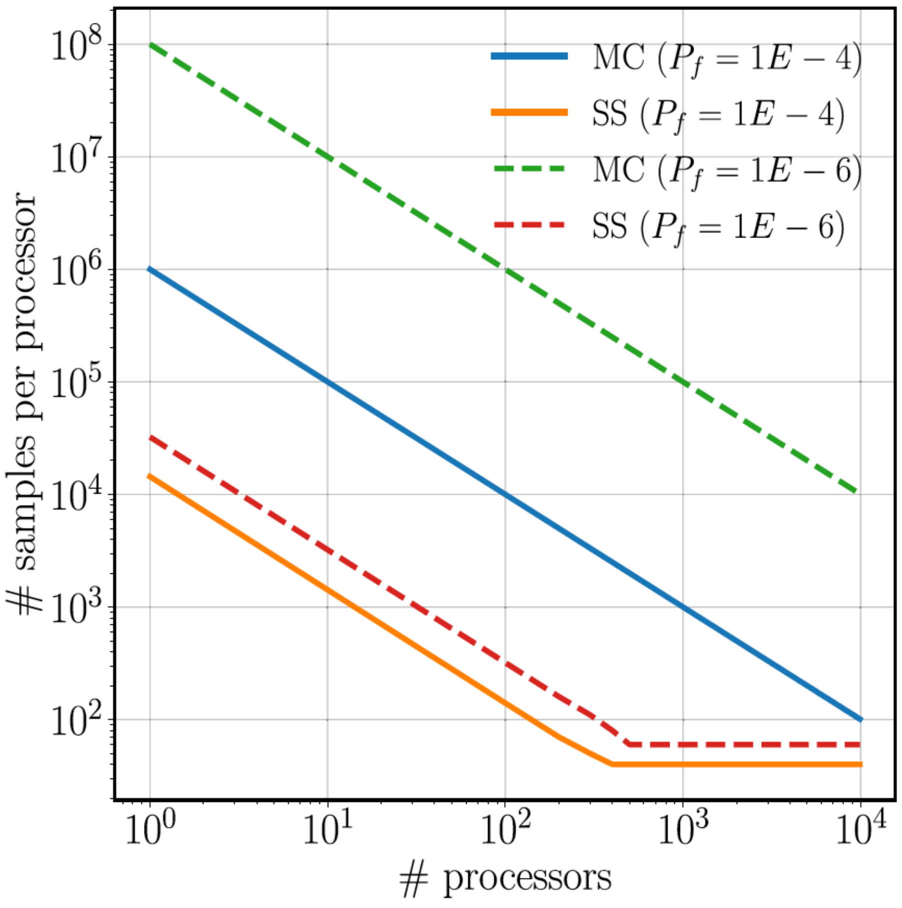
MultiApp block

```
[MultiApps]
  [sub]
    type = SamplerFullSolveMultiApp
    input_files = sub.i
    sampler = sample
  []
[]
```

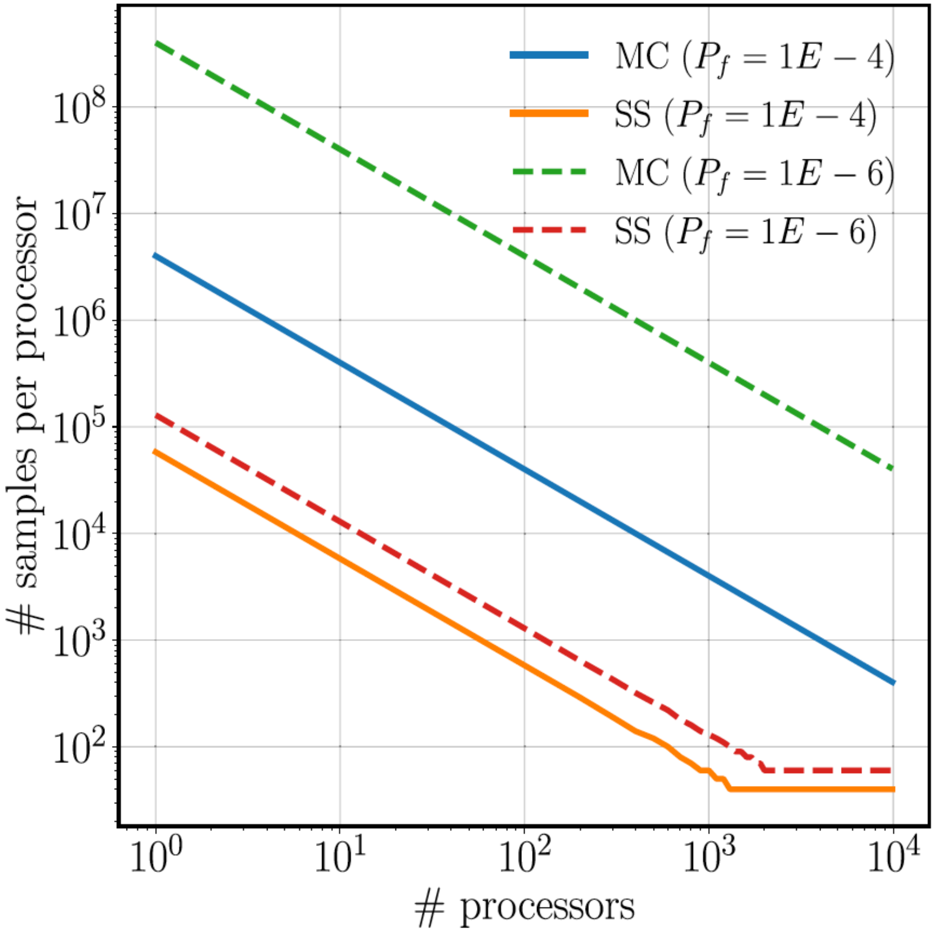
Executioner block (total samples; 3 subsets with 20 samples each)

```
[Executioner]
  type = Transient
  num_steps = 60
[]
```

Parallel scalability compared to Monte Carlo



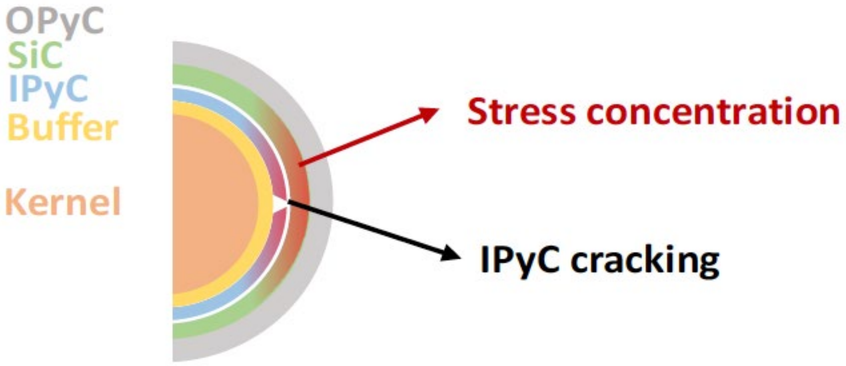
Coefficient of variation: 10%

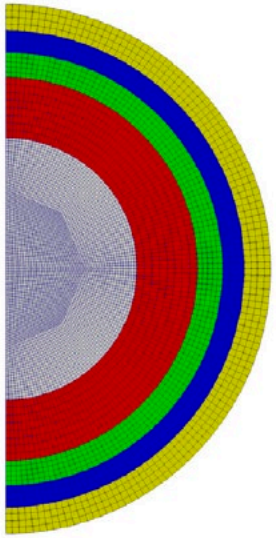


Coefficient of variation: 5%

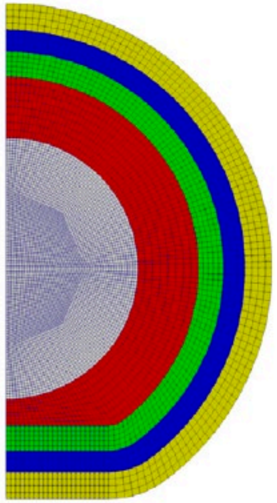
(MC: Monte Carlo; SS: subset simulation)

TRISO fuel failure analysis (2D models)





Spherical



Aspherical

- The 1-D models approximate stresses in the SiC layer based on modification factors
- These factors are calibrated by running evals of the 2-D model
- 2-D model explicitly models cracking in IPyC layer and stress conc. in SiC layer
- More accurate, but mesh density dependent. Therefore, **computationally expensive (~30 mins)**
- Same output: SiC stress – strength (> 0 failure)
- All other inputs for the four models are same as the 1D models

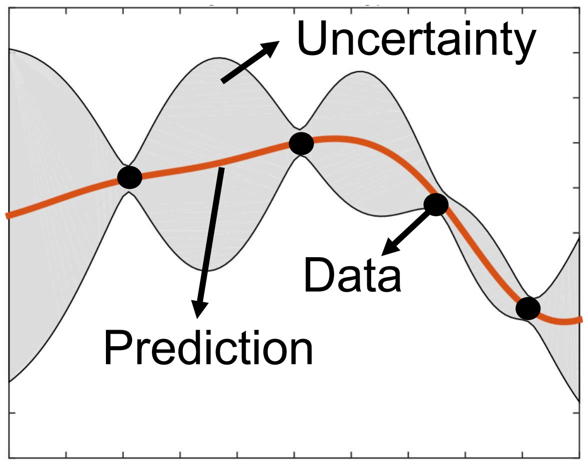
TRISO fuel failure analysis (2D models)

	Weibull	Subset simulation
Model 1	Prob. fail = 0.99E-4	Prob. fail = 1.19E-4 COV = 0.055 N/proc. = 40
Model 2	Prob. fail = 5.86E-5	Prob. fail = 3.4E-5 COV = 0.06 N/proc. = 40
Model 3	Prob. fail = 2.42E-3	Prob. fail = 2.49E-3 COV = 0.045 N/proc. = 40
Model 4	Prob. fail = 3.84E-5	Prob. fail = 3.75E-5 COV = 0.067 N/proc. = 40

COV: Coefficient of variation; N = number of model evals

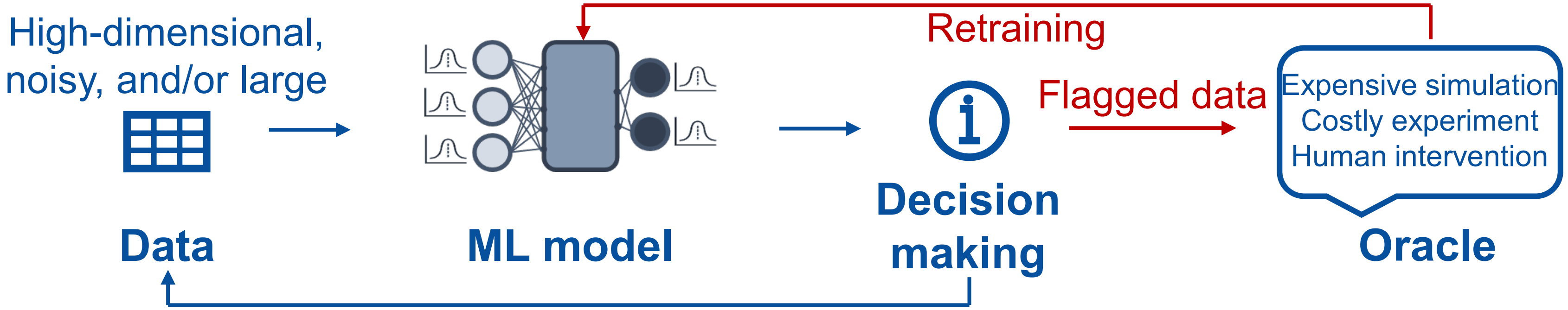
- Monte Carlo computationally infeasible
- Reference result is from Weibull theory; closed form. TRISO failure modes restricted to Weibull failure modes.
- Subset simulation results compare well with Weibull theory
- Subset simulation approximately takes 25-30 hours on 1000 procs
- Subset simulation can consider failure modes beyond Weibull (e.g., debonding and kernel migration)

Active learning concept

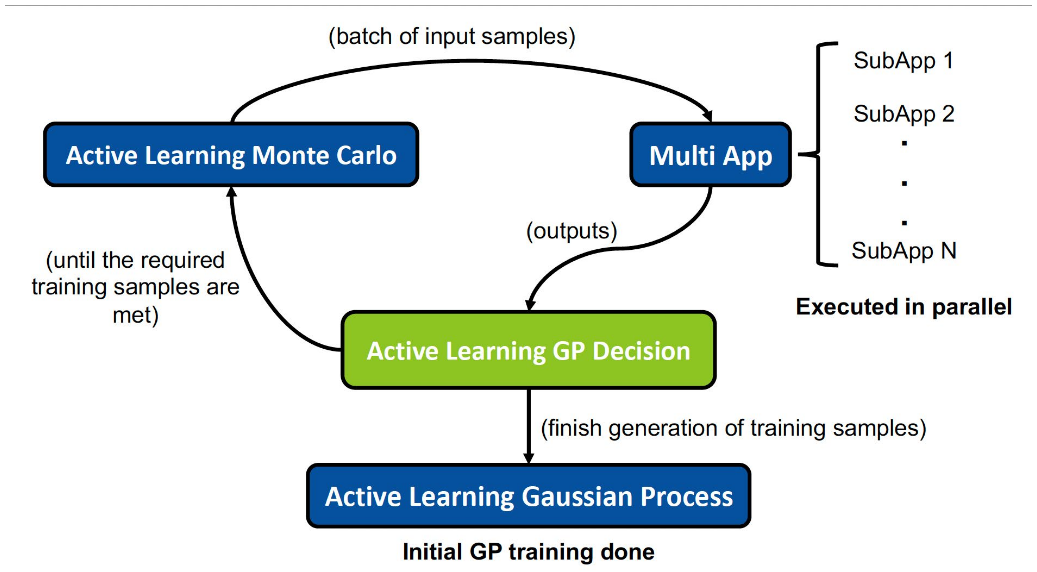


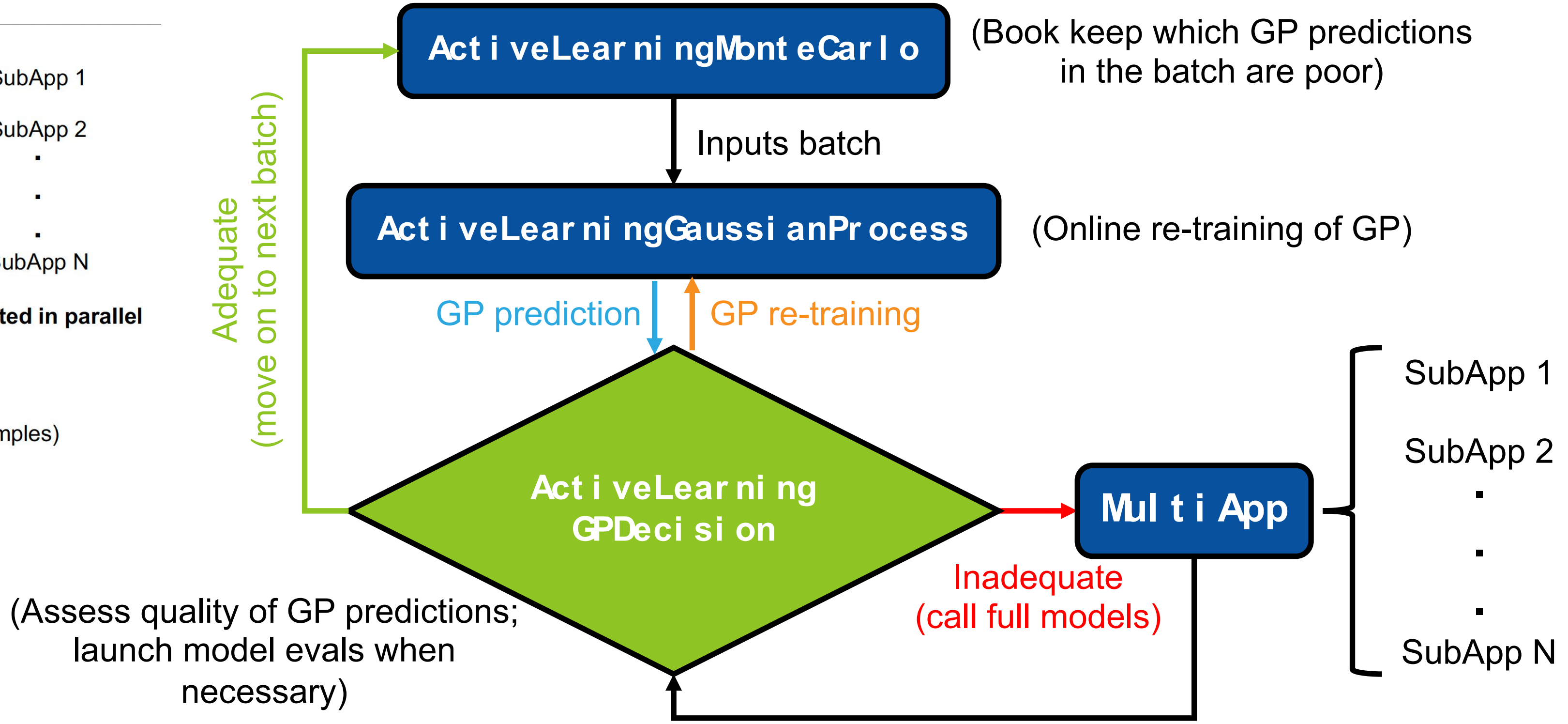
(Credit: Cornell University)

Gaussian Process: Surrogates that know when they're wrong.

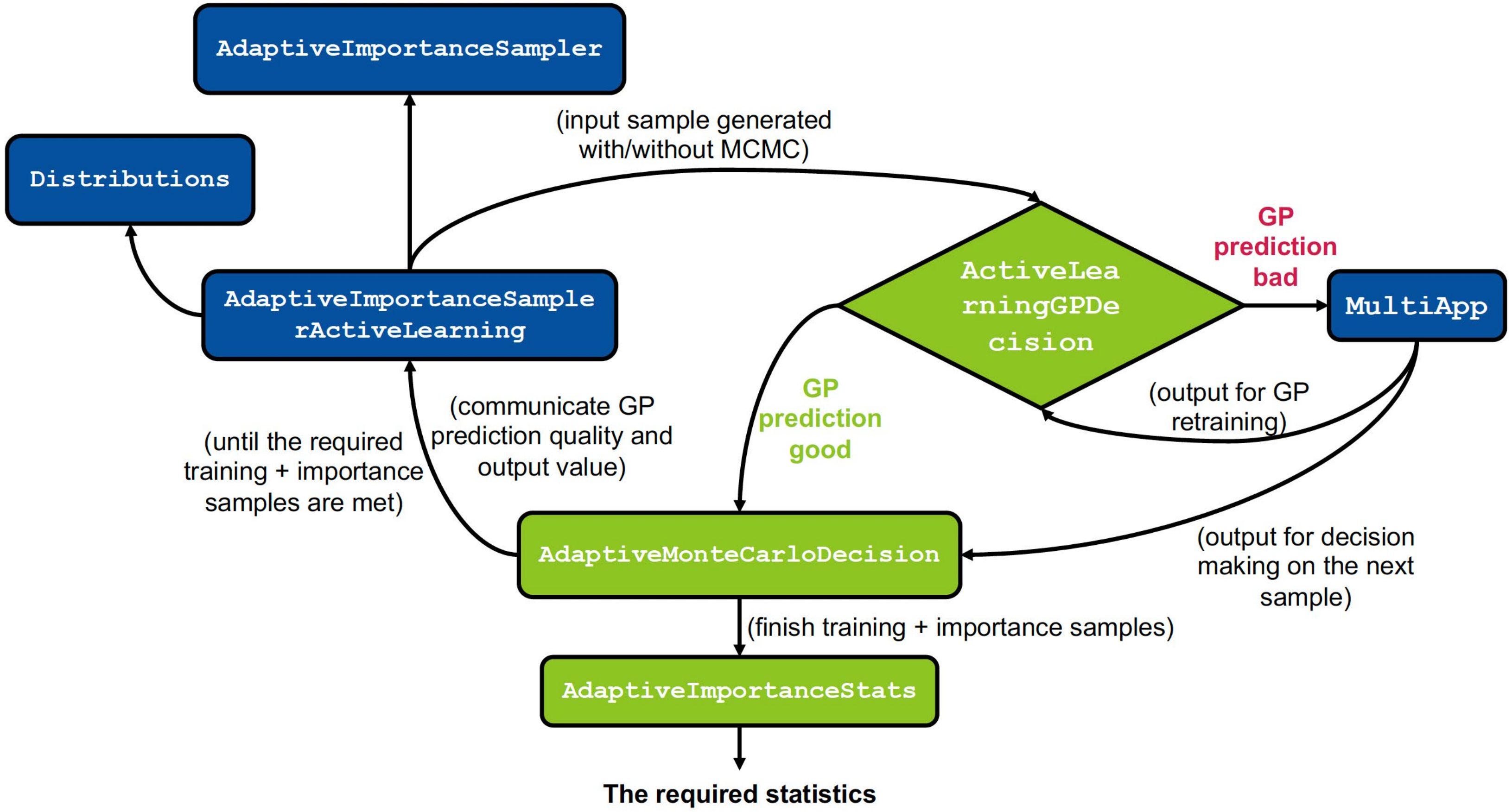


Parallel active learning in MOOSE STM

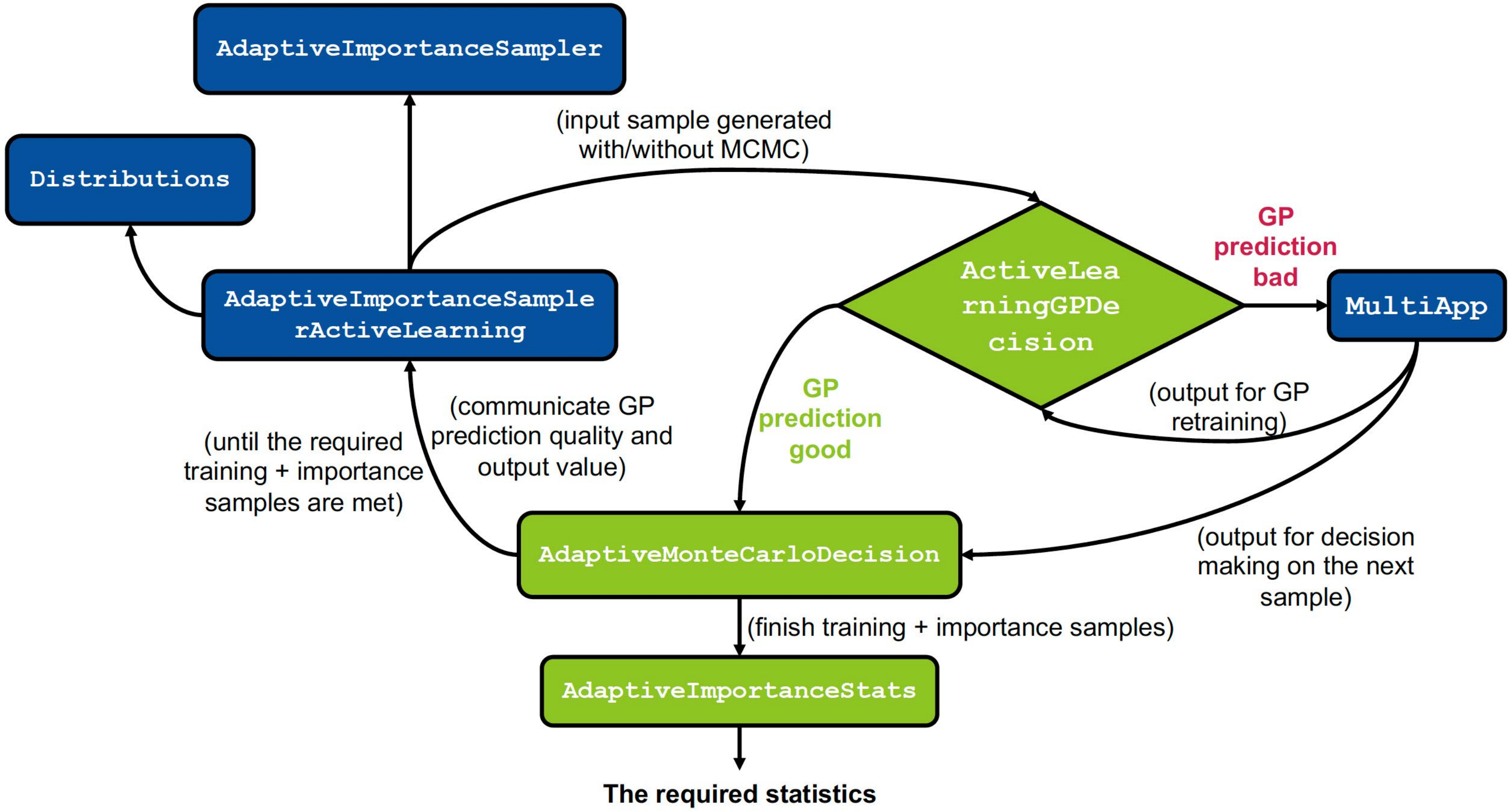




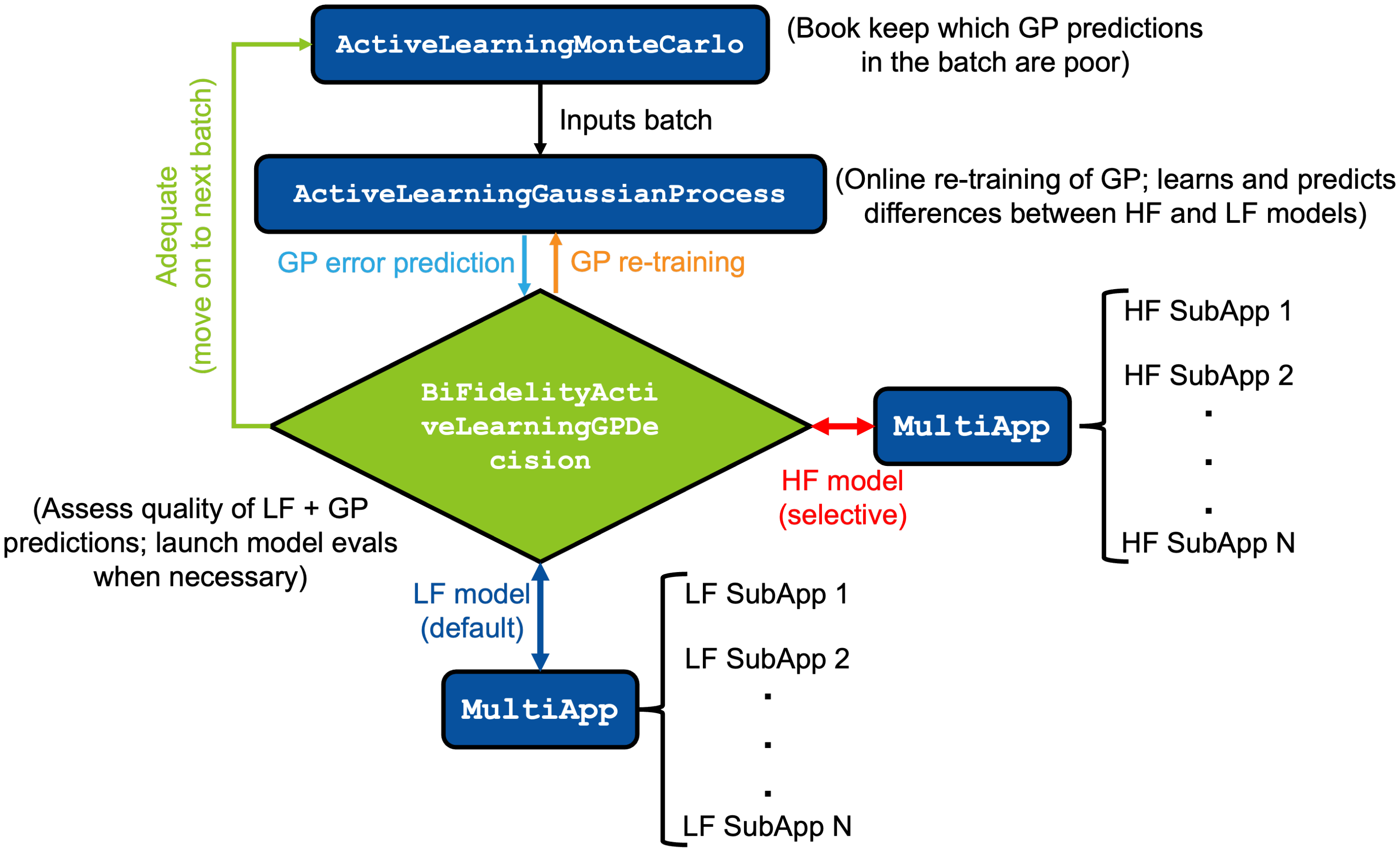
Active learning in importance sampling in MOOSE STM



Active learning in importance sampling in MOOSE STM



Bi-fidelity active learning in MOOSE STM



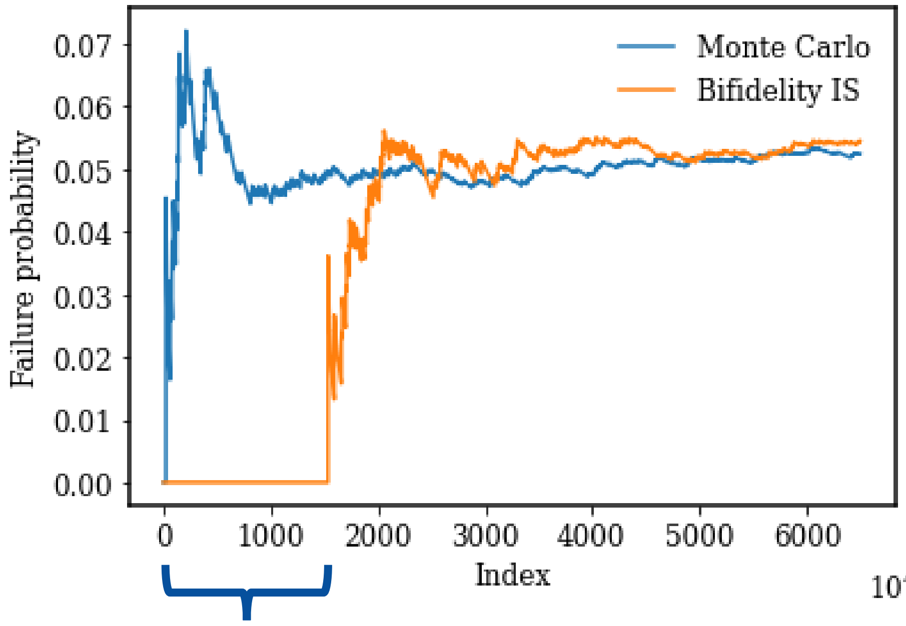
Application: Bi-fidelity active learning in importance sampling

$$k\frac{d^2T}{dx^2} = T_b$$

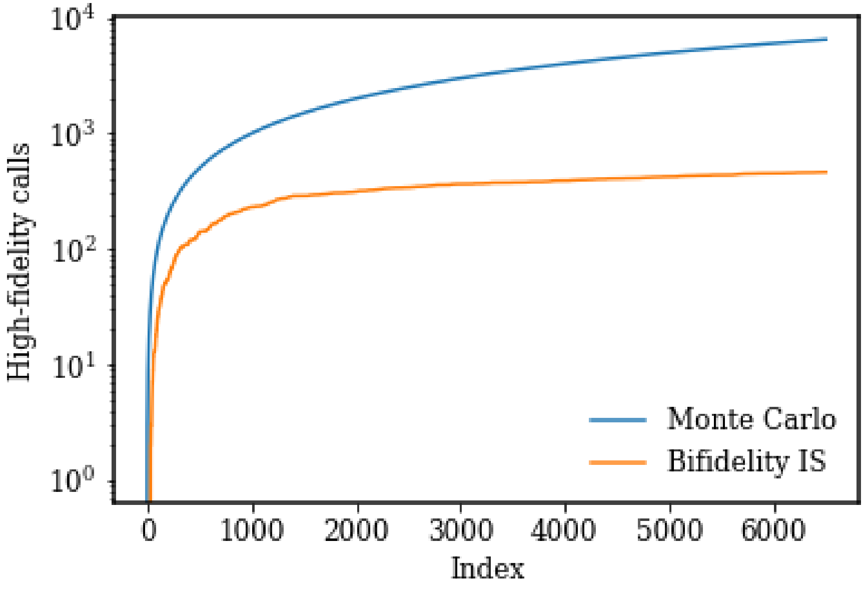
Stochastic parameters
k, T_b, Dirichlet BC

Bi-fidelity models
High-fidelity: 100 intervals mesh
Low-fidelity: 5 intervals mesh

Failure threshold is 350.0



Learning the importance region



Ongoing and future developments in MOOSE STM

- Development of **multiple output Gaussian processes** and linear and non-linear **dimensionality reduction** methods (funded by DOE AMMT)
- Development of a framework for **Bayesian optimization** for reactor design optimization (funded by DOE NEUP)
- Parallelizable inverse UQ capabilities using **Bayesian methods** (funded by DOE NEAMS)
- Parallelizable **active learning learning with MCMC** sampling methods (funded by LDRD and DOE NEAMS)

Thank you!
Questions?