# Advanced Reliability and Safety Analysis

RAVEN Workshop

# *Outline*

1. Dynamic Probabilistic Risk/Safety Analysis
   – Classic vs. Dynamic PRA

2. Adaptive Sampling
   – Limit Surface

# Reliability and Safety Analysis

- Safety Analysis: determine the likelihood a system will not perform its functions in a specified time period

- Reliability Analysis: determine the ability of a system to perform its functions in a specified time period

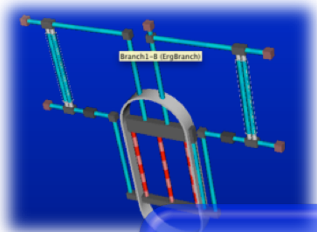
$$\mathrm{Re}\,liability = 1 - probability\_failure$$

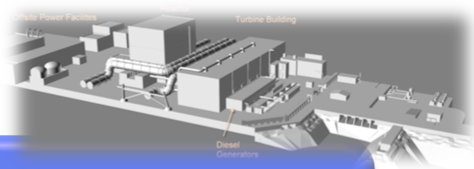
# Dynamic Probabilistic Risk/Safety Analysis

- Classic Probabilistic Risk/Safety Analysis
  - Static boolean logic structures
    - Event-Trees / Fault-Trees
  - Accident sequence set by the analyst
  - Timing of events partially considered

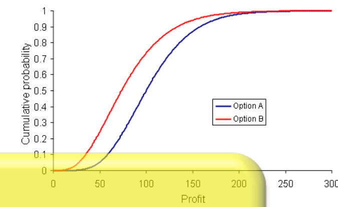
- Dynamic Probabilistic Risk/Safety Analysis
  - Direct use of system simulation codes
  - Coupled with stochastic analysis tools (e.g., RAVEN)
  - Accident sequence set by the system control logic
  - Timing of events fully considered

# Dynamic Probabilistic Risk/Safety Analysis





| Parameters | Distribution |
|---|---|
| Wave height (m) | Exponential |
| Wave impact time (h) | Uniform |
| Diesel recovery time (h) | Weibull |
| Off-site grid recovery time[a] (h) | Lognormal |
| Off-site grid recovery time[b] (h) | Lognormal |
| Batteries failure time (h) | Triangular |
| Batteries recovery time (h) | Lognormal |




**Deterministic modeling**

Plant modeling

External event modeling

**Stochastic modeling**

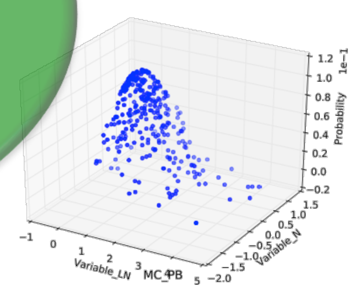
Identify uncertain parameters and associate a pdf

**Stochastic analysis**

Sample the pdfs

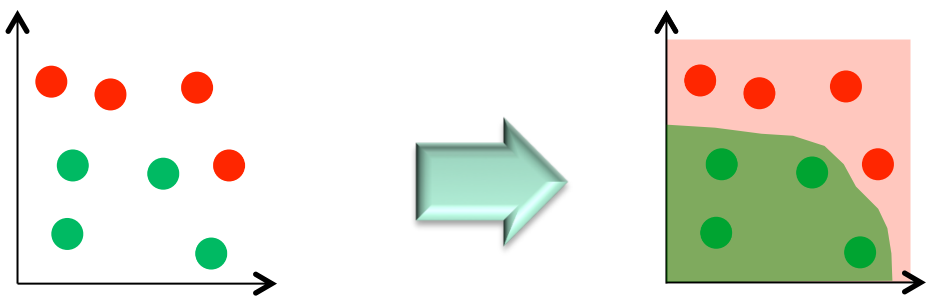
Run N times system simulation code(s)

Evaluate desired FOMs



# ROMs: Classifiers

- ROMs as mathematical models designed approximate an input-output relationship
- Classifiers: output variable is boolean
  - i.e., 0 or 1
  - e.g., OK or Failure

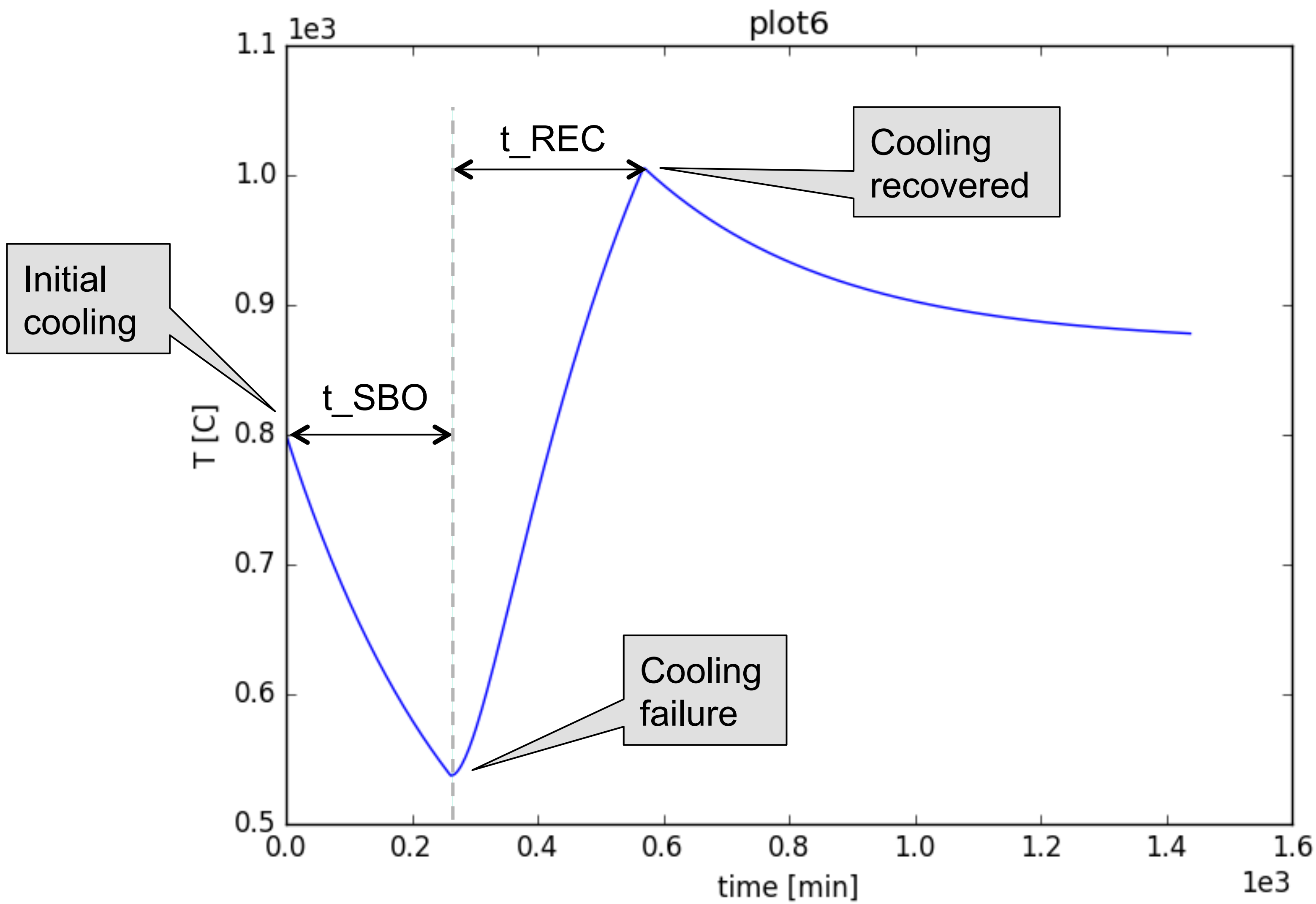


# *Exercise 1: Dynamic PRA Analysis*

# Excercise1: Model Sampling

- Model
  - Simplified PWR model
- Scenario
  - Station Blackout Analysis
- Steps
  - Sample the model response *N* times
  - Determine failure probability

# Excercise1: Model Sampling



# *Excercise1: Model Sampling*

| Distributions | Models | Samplers | Databases | DataObjects | Steps |
|---|---|---|---|---|---|

```
<Steps>
  <MultiRun name="MCRun">
    <Input   class="DataObjects" type="PointSet"       >inputPlaceHolder</Input>
    <Model   class="Models"      type="ExternalModel"  >PythonModule</Model>
    <Sampler class="Samplers"    type="MonteCarlo"     >MC</Sampler>
    <Output  class="DataObjects" type="PointSet"       >outPS</Output>
    <Output  class="DataObjects" type="HistorySet"     >outHS</Output>
  </MultiRun>
  ...
  <PostProcess name="PP">
    <Input   class="DataObjects"  type="PointSet"      >outPS</Input>
    <Model   class="Models"       type="PostProcessor" >integral</Model>
    <Output  class="Files"        type=""              >integral.xml</Output>
  </PostProcess>
</Steps>
```

Input file name: dynamicPRA.xml

# *Exercise 2: Limit Surface Analysis*

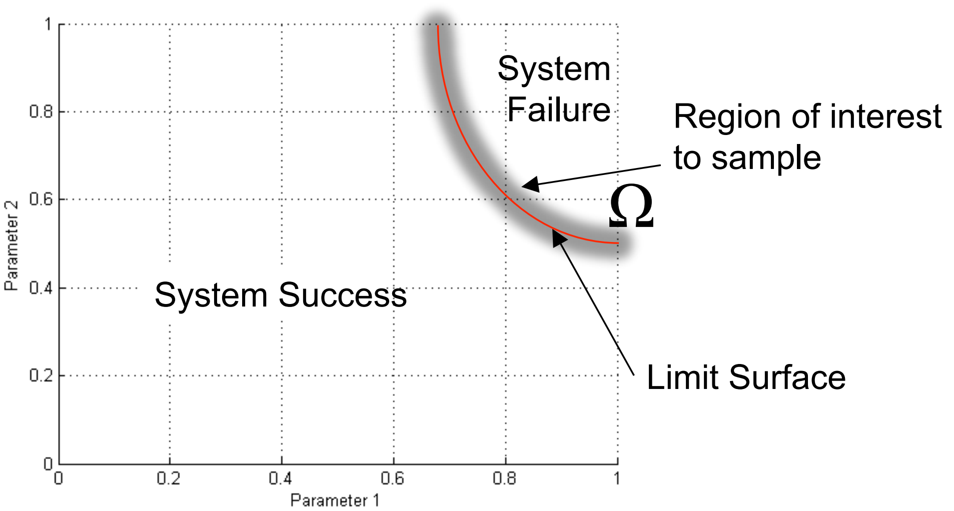
# Adaptive Sampling

- Computational costs might be impractical
  - Computationally expensive codes
  - Many simulation runs are often required

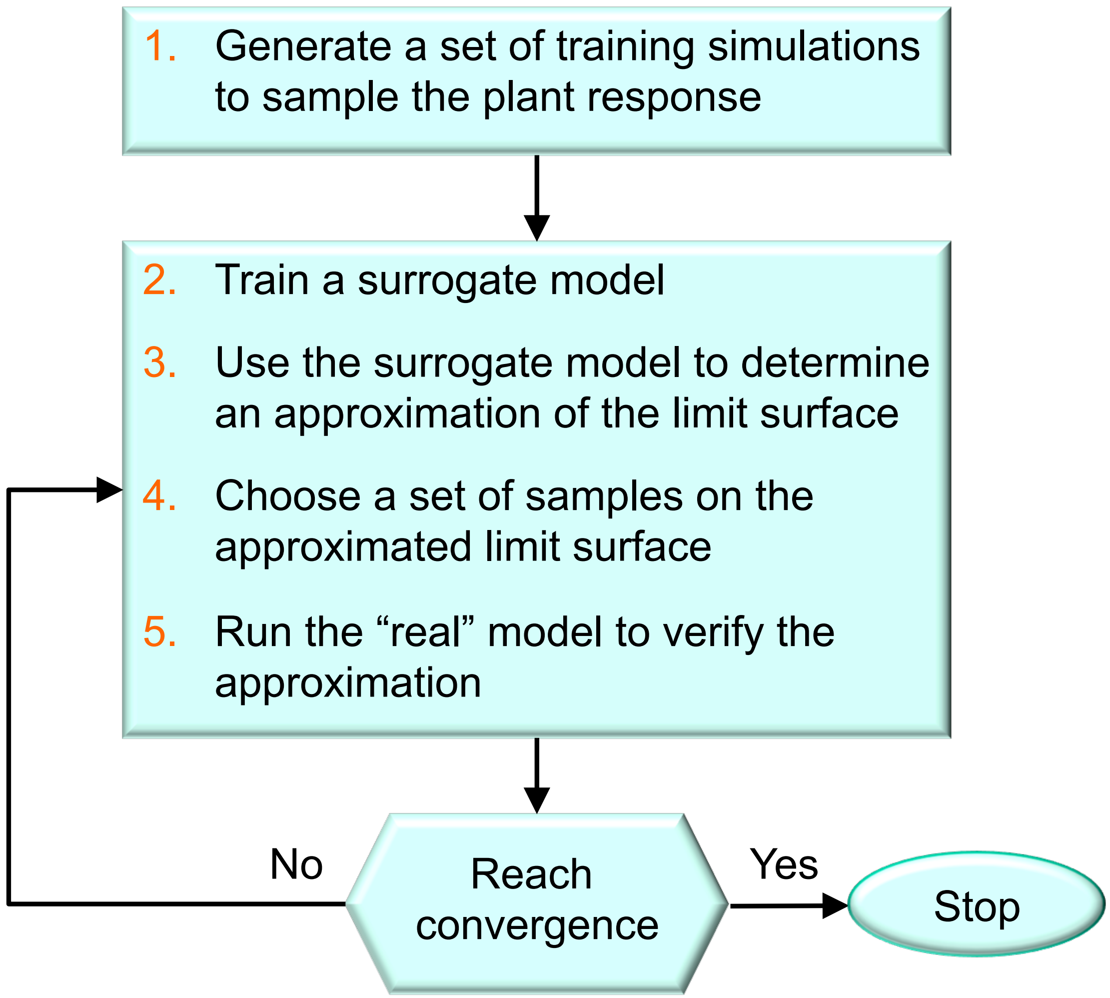
- Reduced Order Modeling can be a solution
  - Objective: reduce the complexity of the problem

- Search of the limit surface: boundaries in the input space between failure and success

$$p_{failure} = \int_{\Omega} pdf(\omega)\, d\omega$$



# Reliability Analysis: Limit Surface Search



# Reliability Analysis: Limit Surface Workflow

1. Define the distributions:
   - Weigh the error estimation
   - Generate a probability-weighted metric for grid construction
   - Determine the probability of being inside one of the regions identified by the limit surface

1. Define the surrogate model (acceleration ROM) to be used as the adaptive sampler

2. Define the adaptive sampler:
   - Convergence control and grid
   - Import the acceleration ROM
   - Associate the distribution with the variables

3. Define the adaptive sampling step

# Exercise 2: Limit Surface Analysis

| Distributions | Models | Samplers | Functions | DataObjects | Steps |
|---|---|---|---|---|---|

```
<Models>
  ...
  <ROM name="ROM" subType="SciKitLearn">
    <Features>tREC,tSBO</Features>
    <Target>goalFunction</Target>
    <SKLtype>svm|SVC</SKLtype>
    <kernel>rbf</kernel>
    <gamma>0.25</gamma>
    <C>25</C>
  </ROM>
</Models>
```

Input file name: runLS.xml

# Exercise 2: Limit Surface Analysis

| Distributions | Models | Samplers | Functions | DataObjects | Steps |
| --- | --- | --- | --- | --- | --- |

```xml
<Functions>
  <External name="goalFunction" file="goalFunction">
  <variable>outcome</variable>
  </External>
</Functions>
```

```python
def __residuumSign(self):
  if self.outcome == 0:
    return -1
  else: return 1
```

File: goalFunction.py

Input file name: runLS.xml

# Exercise 2: Limit Surface Analysis

| Distributions | Models | Samplers | Functions | DataObjects | Steps |
|---|---|---|---|---|---|

```
<Samplers>
  ...
  <LimitSurfaceSearch name="Adaptive">
    <ROM       class="Models"     type="ROM"      >ROM</ROM>
    <Function  class="Functions"  type="External" >goalFunction</Function>
    <TargetEvaluation class="DataObjects" type="PointSet" >outPS
                      </TargetEvaluation>
    <Convergence limit="250" forceIteration="False" weight="value"
                 persistence="10">1.0e-5</Convergence>
    <variable name="tREC">
      <distribution>tREC_dist</distribution>
    </variable>
    <variable name="tSBO">
      <distribution>tSBO_dist</distribution>
    </variable>
  </LimitSurfaceSearch>
</Samplers>
```

Input file name: runLS.xml

# Exercise 2: Limit Surface Analysis

| Distributions | Models | Samplers | Functions | DataObjects | Steps |
|---|---|---|---|---|---|

```
<Steps>
  <MultiRun name="MCRun" pauseAtEnd="True">
    <Input   class="DataObjects" type="PointSet"      >inputPlaceHolder</Input>
    <Model   class="Models"      type="ExternalModel" >PythonModule</Model>
    <Sampler class="Samplers"    type="MonteCarlo"    >MC</Sampler>
    <Output  class="DataObjects" type="PointSet"      >outPS</Output>
  </MultiRun>
  <MultiRun name="adaptive" pauseAtEnd="True">
    <Input   class="DataObjects" type="PointSet"      >inputPlaceHolder</Input>
    <Model   class="Models"      type="ExternalModel" >PythonModule</Model>
    <Sampler class="Samplers"    type="LimitSurfaceSearch">Adaptive</Sampler>
    <SolutionExport class="DataObjects" type="PointSet" >limitSurface
    </SolutionExport>
    <Output  class="DataObjects" type="PointSet"      >outPS</Output>
  </MultiRun>
</Steps>
```

Input file name: runLS.xml

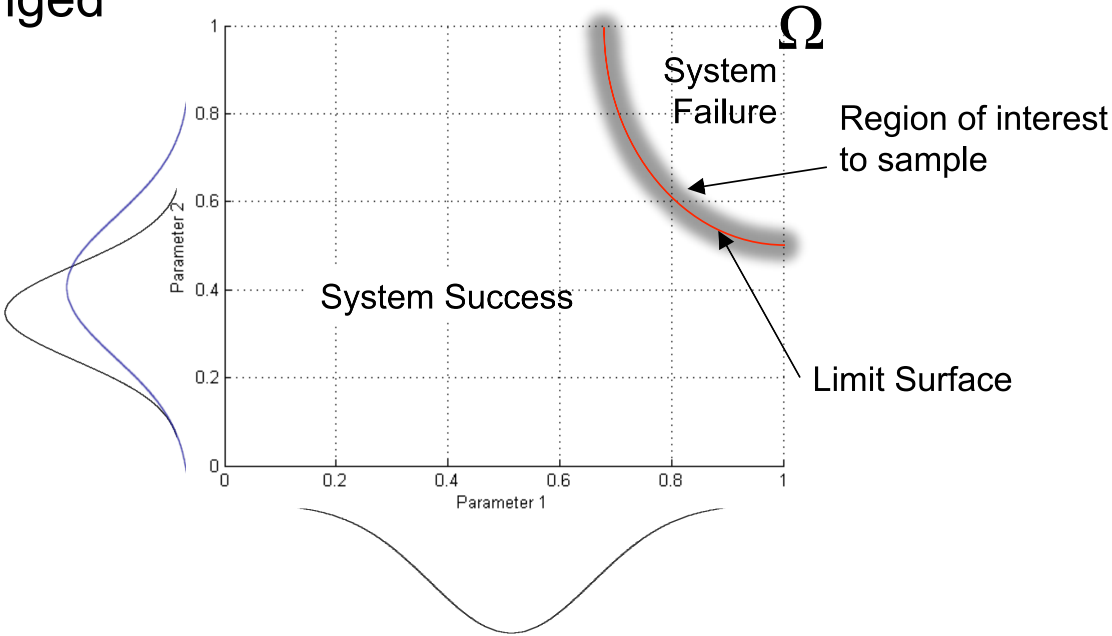
Idaho National Laboratory

# *Exercise 3: Failure Probability Calculation from LS*

# Exercise 3: Failure Probability Calculation From LS

- Limit surface has a pure deterministic information
- Probabilistic information is contained in the integral within the limit surface weighted by the set of distributions
- For a new set of distributions the integral needs to be recalculated but the limit surface is unchanged

$$p_{failure} = \int_{\Omega} pdf(\omega)\, d\omega$$



# Exercise 3: Failure Probability Calculation From LS

| Distributions | Models | Samplers | Databases | DataObjects | Steps |
|---|---|---|---|---|---|

```
<Models>
  <PostProcessor name="LS_Integral" subType="LimitSurfaceIntegral">
    <tolerance>0.001</tolerance>
    <integralType>MonteCarlo</integralType>
    <target>goalFunction</target>
    <variable name="tREC">
      <distribution>tREC_dist</distribution>
    </variable>
    <variable name="tSBO">
      <distribution>tSBO_dist</distribution>
    </variable>
  </PostProcessor>
</Models>
```

Input file name: pbFromLS.xml

# Exercise 3: Failure Probability Calculation From LS

| Distributions | Models | Samplers | Databases | DataObjects | Steps |
|---|---|---|---|---|---|

```
<Steps>
  <IOStep name="extract_data_adapt">
    <Input   class="Databases"   type="HDF5"           >out_db_adaptive</Input>
    <Output  class="DataObjects” type="PointSet"       >outAdapt_failure</Output>
  </IOStep>
  <PostProcess name="integral">
    <Input   class="DataObjects” type="PointSet"       >outAdapt_failure</Input>
    <Model   class="Models"      type="PostProcessor" >LS_Integral</Model>
    <Output  class="DataObjects” type="PointSet"       >LSintegral_PS</Output>
    <Output  class="Files"       type=""               >integral.csv</Output>
  </PostProcess>
</Steps>
```

Input file name: pbFromLS.xml