Reliability analysis and RAVEN

RAVEN Workshop

PSA 2015 - April 26th 2015, Sun Valley (ID)

Idaho National Laboratory
www.inl.gov

Outline

- Brief introduction on Reliability Analysis
  - Goal Oriented Sampling
  - Concept of Limit Surface
  - Convergence acceleration through ROMs

- Reliability and RAVEN
  - Available Goal Oriented Samplers
  - Performing Reliability Analysis within RAVEN: workflow

- Application examples of Reliability Analysis

- RAVEN examples

Reliability Analysis:
INTRODUCTION

Reliability Analysis: a quick introduction

- Reliability theory describes the probability of a system completing its expected function during an interval of time.
- Failure is called an “event” or “transition”, and the goal is to project, forecast or assess the rate of events for a given population or the probability of an event for an individual.
- Generally reliability period of any object is measured within the durability period of that object.

Main goal

Computing the transition (failure) probability

Reliability Analysis: Goal-oriented sampling

- The evaluation of the transition probability might be computationally challenging, if traditional once-through sampling is employed:
  - Rare events (low probability) require an enormous number of input space explorations:
    - Computing a failure probability of 10^{-6} would require millions of Monte Carlo samples
  - The information entropy contained in the already-evaluated histories is not used to adapt the sampling strategy:
    - System response depends on many variables but often few are really important

Adaptive Sampling based on Reliability (Limit) Surface search

Reliability Analysis: Limit Surface Concept

- The limit surface can be described as the hyper-surface that classifies the input space with respect transition regions:
  - The locus of points that divides the input domain with respect a boolean response (e.g. failure/success)
- Consider

$$\mathrm{F}(x_i) \longleftarrow \text{System characterization}$$

- The goal function is an object that is defined as a part of the system outcome space. In a safety context, the goal function usually represents the success or failure (transition) of the system.

$$C\left(F\left(x_i\right)\right) = \begin{cases} C\left(F\left(x_i\right)\right) = +1 & F\left(x_i\right) \le F\left(x_{TH}\right) \\ C\left(F\left(x_i\right)\right) = -1 & F\left(x_i\right) > F\left(x_{TH}\right) \end{cases}$$

Reliability Analysis: Limit Surface Concept (cont.)

- Let's consider $F(x_{i,L})$ as the transition surface in the output space with respect to the goal function $C\left(F(x_i)\right)$

$$F\left(x_{i,L}\right) = \left\{ F\left(\bar{x}_i\right) \middle| \nabla C\left(F(\bar{x}_i)\right) \right| = \infty \right\}$$

If the system evolution is represented by

$$\bar{x}\left(t, \bar{p}, \bar{x}_i\right)$$

Stochastic Parameters

Initial Conditions

It is possible to identify the set of pairs

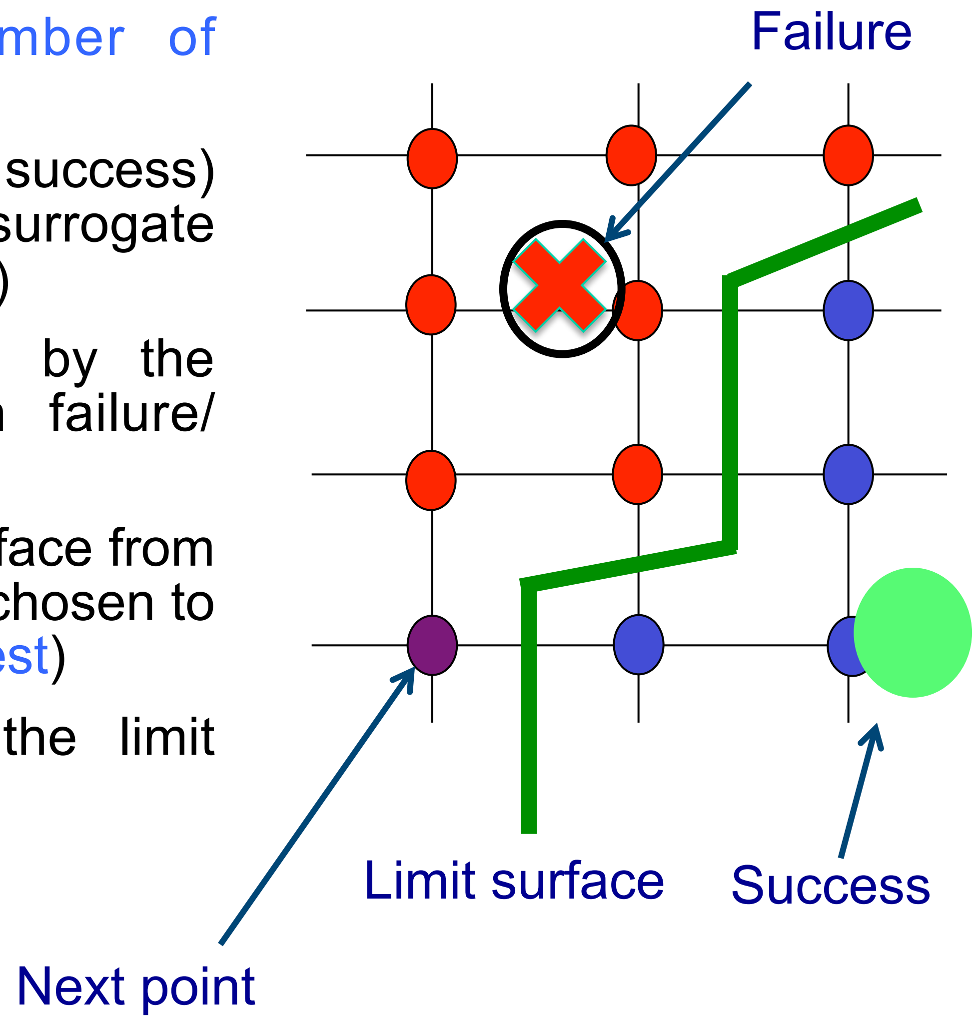
$$\left(\bar{p}, \bar{x}_i\right)_{LS}$$

in the input space, which leads the system outcome to match:

$$F(x_{i,L})$$

Reliability Analysis: Limit Surface Search

1. Training on initial small number of sampled points
2. A fine grid is classified (failure or success) based on an user-specified surrogate model (sampling of the surrogate)
3. The limit surface is identified by the location of transition between failure/ success
4. The furthest point on the limit surface from any other already tested point is chosen to test the classifier (convergence test)
5. Process iterates (2➔4) until the limit surface converges



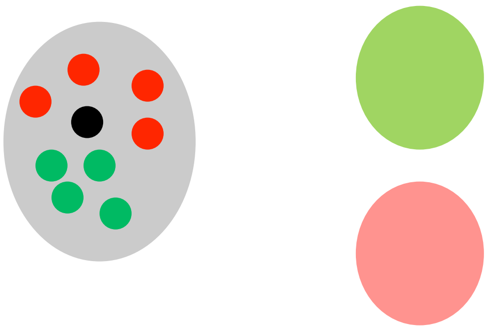
Reliability Analysis: ROMs

- For the Adaptive Sampling based on the Reliability (Limit) Surface search, special types of ROMs called “classifiers” are used:
  - A model (set of equations) that identifies to which category an object belongs in the feature (input) space

- Let’s consider again a set of *N* data points $\left(x_i, F(x_i)\right)$
- Based on the goal function C :

Boolean vector

$$c_i = C\left(F(x_i)\right) \quad i = 1, \ldots, N$$



- Build a surrogate model of type “classifier”
  - Reduced Order Model

$$G(x): x_i \longrightarrow G(x_i) \cong C\left(\mathrm{F}(x_i)\right)$$

- In RAVEN, these ROMs are used as acceleration schemes for goal oriented sampling strategies ➔ they can be used to predict the location of the reliability (limit) surface

Reliability Analysis and RAVEN

Reliability Analysis and RAVEN: Samplers

- RAVEN currently has 2 ways to perform *Reliability Analysis based on Limit Surface Search*:

Adaptive Sampler

Available

All Coupled Codes

Adaptive DET Sampler

Available

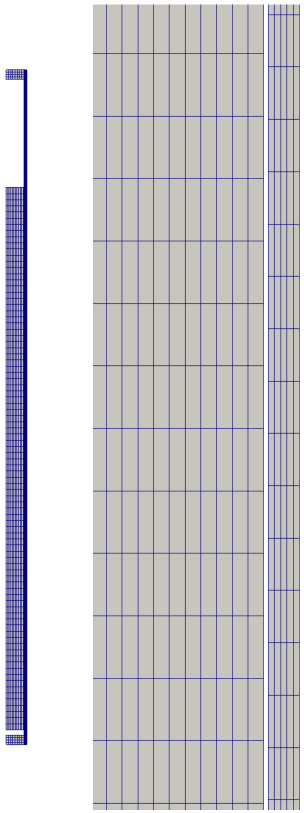
RELAP-7 only

Reliability Analysis: Limit Surface Workflow

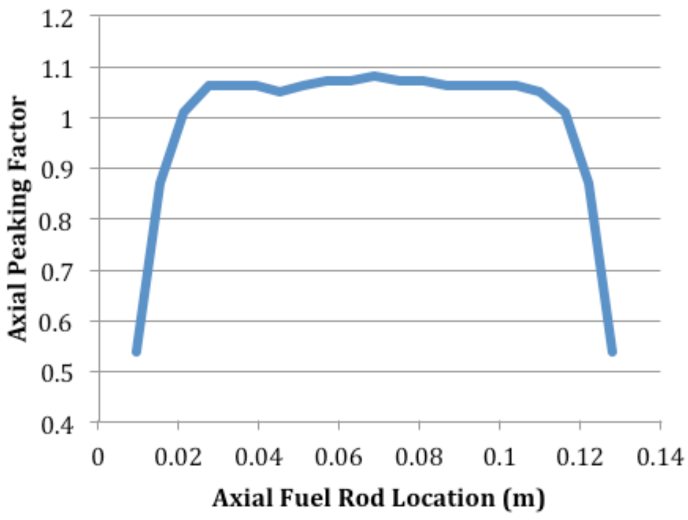
- Define the distributions. Distributions are used to:
  - Weigh the error estimation
  - Generate a probability-weighted metric for grid construction
  - Determine the probability of being inside one of the regions identified by the limit surface
- Define the surrogate model to be used as the accelerator (of type Boolean or discrete)
- Define the adaptive sampler:
  - Convergence control and grid
  - Import the acceleration ROM
  - Associate the distribution with the variables
- Define a step
- Post-process the data to compute the failure probability

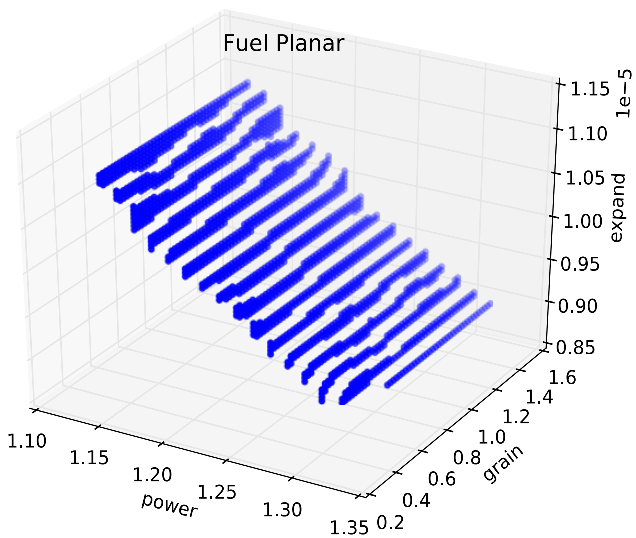
Reliability Analysis: Applications

Reliability Analysis: Applications
Fuel Performance - BISON



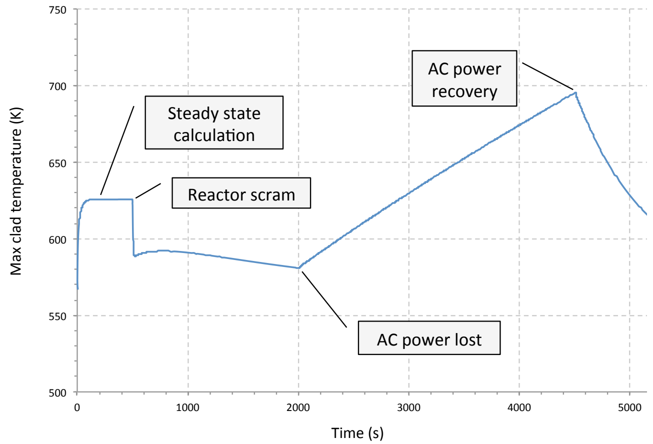
- Smeared pellet stack with hafnium insulator end pellets:
  - 10 fuel pellets as a single, long pellet sandwiched between two insulator pellets.
- 10,000 second ramp from zero power to 25KW/m.
- After the ramp the power was held constant to the end time of 1e8 seconds.



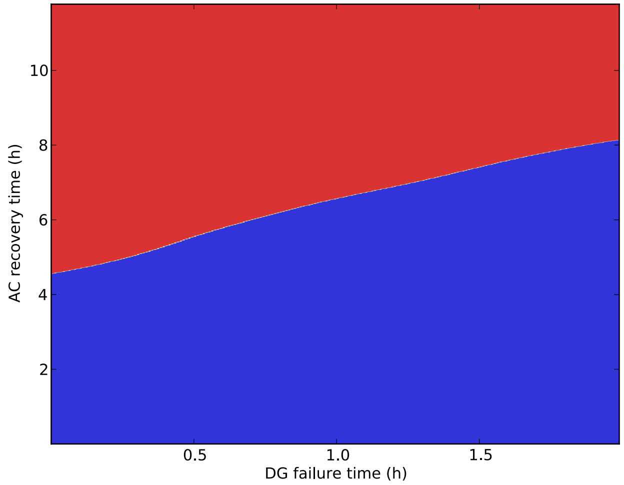


Reliability Analysis: Applications
Pressurized Water Reactor Station Black Out

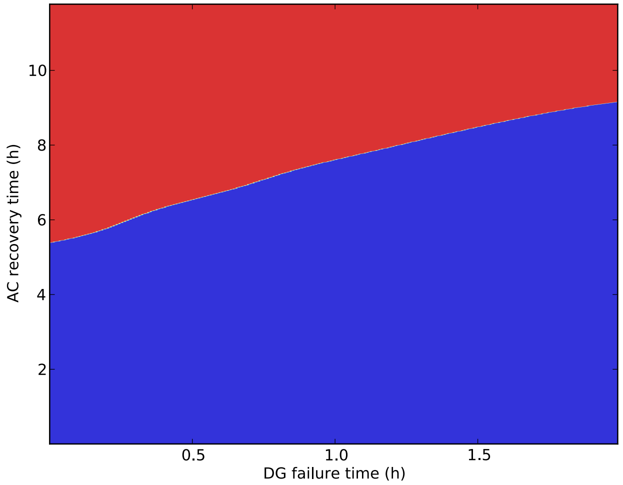
PWR SBO



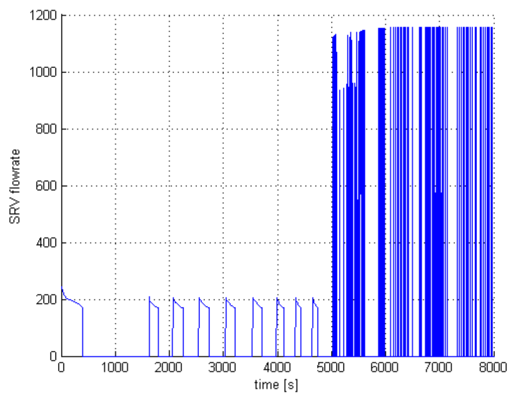
100% Power

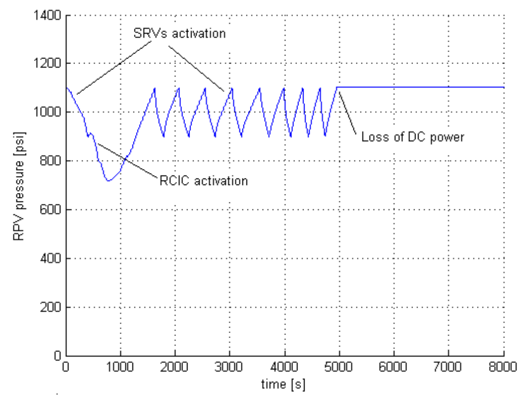


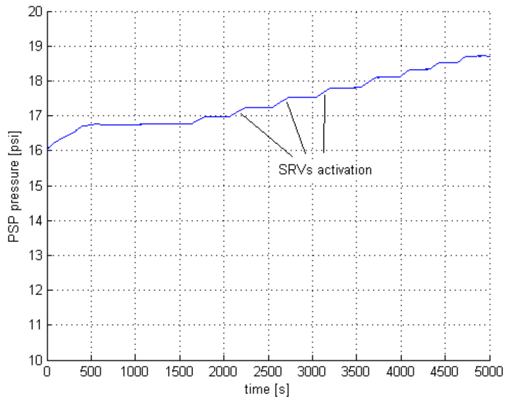
120% Power

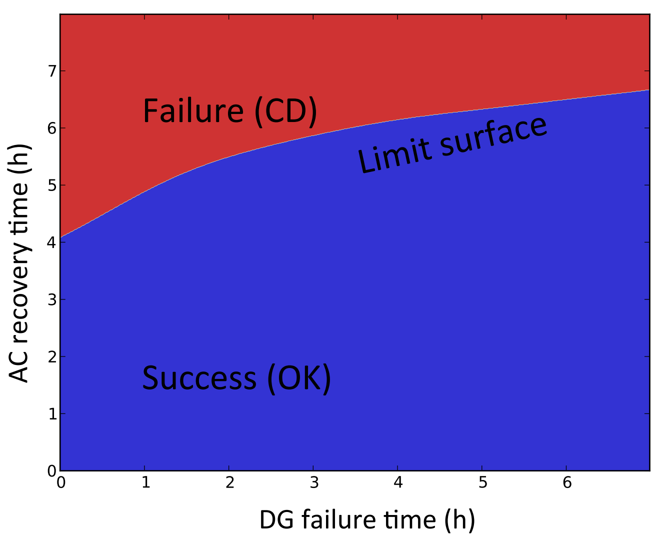


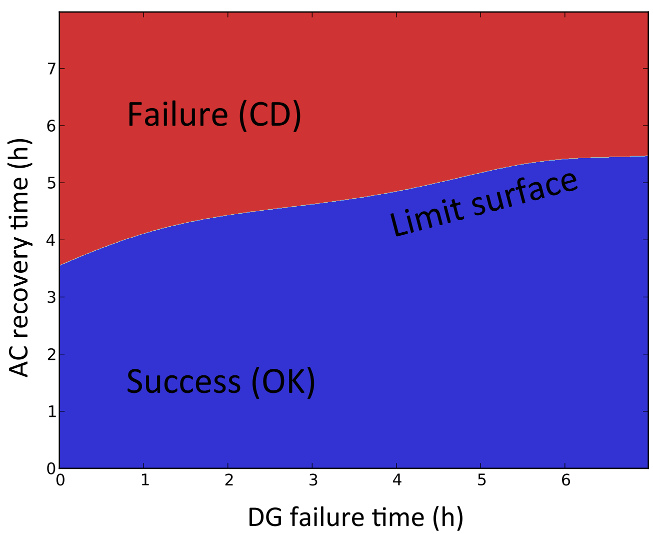
Reliability Analysis: Applications
Boiling Water Reactor Station Black Out







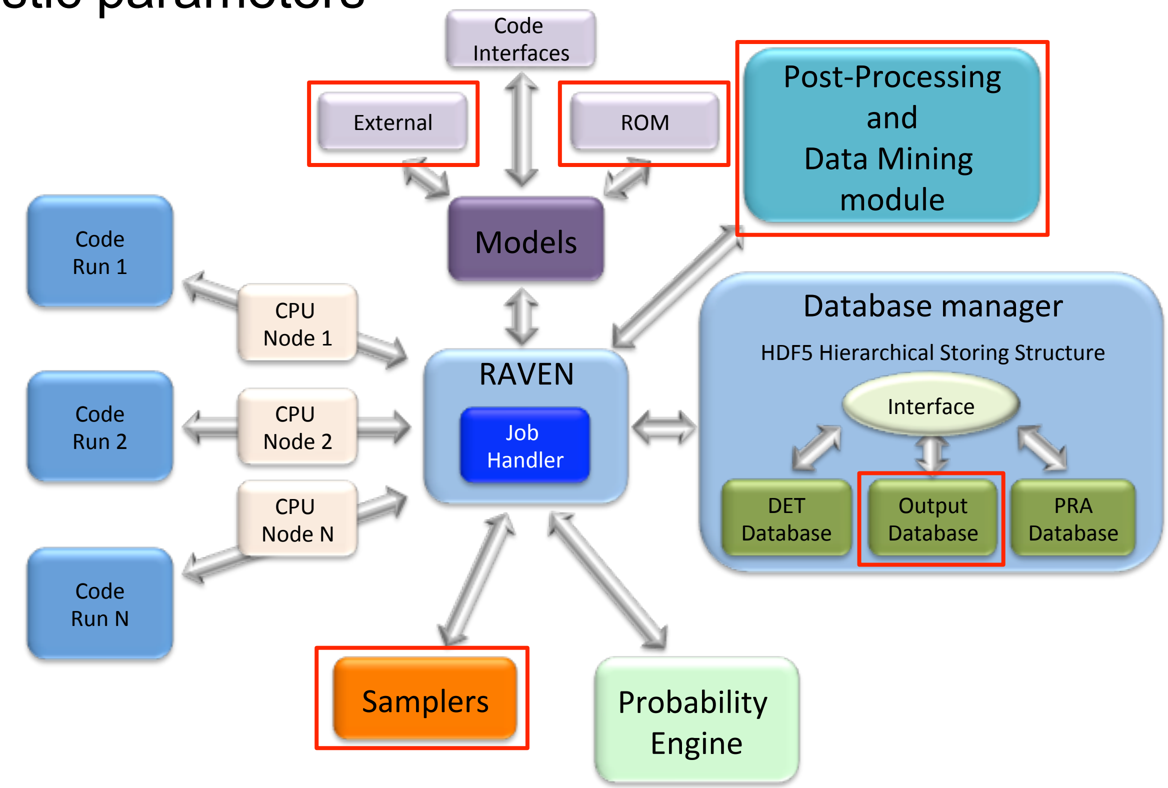




RAVEN examples

Workflow

1. Sample a model and create a database
2. Use generated data to seed the Adaptive Sampling Step
3. Compute failure probability from converged Solution
4. Train a ROM and compute the failure probability changing the stochastic parameters



Exercise 1:
Create a data set to seed the Adaptive Sampler

1 – Sample a Model to seed the Adaptive Sampler

Distributions	Models	Samplers	Databases	DataObjects	Steps

```
<Distributions>
      <Normal name='normal_trunc'>
        <mean>0.5</mean>
        <sigma>0.1</sigma>
        <lowerBound>0.0</lowerBound>
        <upperBound>1.0</upperBound>
    </Normal>
    <Normal name='normal'>
        <mean>2.0</mean>
        <sigma>0.2</sigma>
    </Normal>
    <Uniform name='uniform'>
        <upperBound>4.0</upperBound>
        <lowerBound>1.0</lowerBound>
    </Uniform>
</Distributions>
```

Truncation parameters

Input file name: 1-sample_Grid_seed_adaptive.xml

1 – Sample a Model to seed the Adaptive Sampler

Distributions	Models	Samplers	Databases	DataObjects	Steps

```xml
<Models>
    <ExternalModel name='PythonModule' subType='' ModuleToLoad='workshop_model'>
       <variable>x1</variable>
       <variable>x2</variable>
       <variable>x3</variable>
       <variable>y1</variable>
       <variable>y2</variable>
       <variable>y3</variable>
       <variable>y4</variable>
       <variable>y5</variable>
       <variable>failure</variable>
    </ExternalModel>
</Models>
```

Input variables

Output variables

File name:
workshop_model.py

Boolean Variable

Input file name: 1-sample_Grid_seed_adaptive.xml

1 – Sample a Model to seed the Adaptive Sampler

Distributions	Models	Samplers	Databases	DataObjects	Steps

<variable name>
<distribution>
<grid points>

```
<Samplers>
  <Grid name='Grid_function'>
    <variable name='x1' >
      <distribution>normal_trunc</distribution>
      <grid type='value' lowerBound='0.0' construction='equal' steps='5'>0.2</grid>
    </variable>
    <variable name='x2' >
      <distribution>normal</distribution>
      <grid type='value' lowerBound='1.5' construction='equal' steps='5'>0.2</grid>
    </variable>
    <variable name='x3' >
      <distribution>uniform</distribution>
      <grid type='value' lowerBound='1.0' construction='equal' steps='5'>0.6</grid>
    </variable>
  </Grid>
</Samplers>
```

Input file name: 1-sample_Grid_seed_adaptive.xml

1 – Sample a Model to seed the Adaptive Sampler

Distributions	Models	Samplers	Databases	DataObjects	Steps

```xml
<Databases>
  <HDF5 name='out_db'/>
</Databases>

<DataObjects>
    <TimePointSet name='inputPlaceHolder'>
        <Input>x1,x2,x3</Input>
        <Output>OutputPlaceHolder</Output>
    </TimePointSet>
    <TimePointSet name=’outGRID'>
        <Input>x1,x2,x3</Input>
        <Output>y1,y2,y3,y4,y5,failure</Output>
   </TimePointSet>
</DataObjects>
```

<DataObjects>
<Inputs>
<Output>

Input file name: 1-sample_Grid_seed_adaptive.xml

1 – Sample a Model to seed the Adaptive Sampler

Distributions	Models	Samplers	Databases	DataObjects	Steps

```xml
<Steps>
  <MultiRun name='FirstMRun'>
    <Input   class='DataObjects'       type='TimePoint'     >inputPlaceHolder</Input>
    <Model   class='Models'            type='ExternalModel'>PythonModule</Model>
    <Sampler class='Samplers'          type='Grid'          >Grid_function</Sampler>
    <Output  class='DataObjects'       type='TimePointSet'  >outGRID</Output>
    <Output  class='Databases'         type='HDF5'          >out_db</Output>
    <Output  class='OutStreamManager'  type='Print'         >out_dump</Output>
    <Output  class='OutStreamManager'  type='Plot'          >plotResponse_y3</Output>
    <Output  class='OutStreamManager'  type='Plot'          >plotResponse_y4</Output>
    <Output  class='OutStreamManager'  type='Plot'          >plotResponse_y5</Output>
  </MultiRun>
</Steps>
```

Input file name: 1-sample_Grid_seed_adaptive.xml

Exercise 2:

Use generated data to seed the Adaptive Sampling Step and perform Limit Surface search

2- Use data to seed the Adaptive Sampling Step

Models	Functions	Sampler	Databases	DataObjects	Steps

```
<Models>
   <ExternalModel name='PythonModule' subType=''
     ModuleToLoad='workshop_model'>
       <variable>x1</variable>
       <variable>x2</variable>
       <variable>x3</variable>
       <variable>y2</variable>
       <variable>y3</variable>
       <variable>y4</variable>
       <variable>y5</variable>
       <variable>failure</variable>
    </ExternalModel>
    <ROM  name='AccelerationROM' subType='SciKitLearn'>
      <Features>x1,x2,x3</Features>
      <Target>goalFunction</Target>
      <SKLtype>svm|SVC</SKLtype>
      <kernel>rbf</kernel>
      <C>10.0</C>
    </ROM>
</Models>
```

Kernel Type

Penalty parameter of the error term

```
def run(self,Input):
  …
  if self.y4 < 5.0:
    self.failure= 0.0
  else:
    self.failure= 1.0
```

Transition definition:
0.0 -> success
1.0 -> failure

Input file name: 2-seeded_adaptive_sampling.xml

2- Use data to seed a Goal Oriented Sampling

Models	Functions	Sampler	Databases	DataObjects	Steps

```
<Functions>
    <External name='goalFunction' file= 'goalFunctionWorkshop'>
        <variable>y4</variable>
        <variable>failure</variable>
    </External>
</Functions>
```

Variables that are needed for the evaluation

```
def __residuumSign(self):
    if self.y4 < 5.0: return  1
    else            : return -1
```

Transition definition:
1 -> success
-1 -> failure

Input file name: 2-seeded_adaptive_sampling.xml

2- Use data to seed the Adaptive Sampling Step

Models	Functions	Sampler	Databases	DataObjects	Steps

```
<Samplers>
 <Adaptive name=’workshopAdaptive’>
    <ROM class='Models' type='ROM’>accelerated_ROM</ROM>
    <Function class='Functions' type='External’>goalFunction</Function>
    <TargetEvaluation class='DataObjects' type='TimePointSet'>
       outAdaptive
    </TargetEvaluation>
    <Convergence
       limit='3000'
       forceIteration='False'
       weight='value'
       persistence='25'>
         1e-4
    </Convergence>

    <variable name='x1'><distribution>normal_trunc</distribution></variable>
    <variable name='x2'><distribution>normal</distribution></variable>
    <variable name='x3'><distribution>uniform</distribution></variable>
  </Adaptive>
</Samplers>
```

Max Iterations

Goal function

Force Iteration until limit

Error Weighting: CDF or Value

Tolerance

Input file name: 2-seeded_adaptive_sampling.xml

2- Use data to seed a Goal Oriented Sampling

Models	Functions	Sampler	Databases	DataObjects	Steps

```
<Databases>
    <HDF5 name='out_db_adaptive' directory=’DataBaseStorage/'/>
    <HDF5 name='out_db’ filename='out_db.h5’ />
    <HDF5 name='out_db_ls’ directory='DataBaseStorage/'/>
</Databases>

<DataObjects>
    <TimePointSet name='outAdaptive'>
        <Input>x1,x2,x3</Input>
        <Output>y3,y4,y5,failure</Output>
    </TimePointSet>
    <TimePointSet name=’limitSurface'>
        <Input>x1,x2,x3</Input>
        <Output>goalFunction</Output>
    </TimePointSet>
    <TimePointSet name=’inputPlaceHolder'>
        <Input>x1,x2,x3</Input>
        <Output>OutputPlaceHolder</Output>
    </TimePointSet>
</DataObjects>
```

Limit Surface Container

Input file name: 2-seeded_adaptive_sampling.xml

2- Use data to seed a Goal Oriented Sampling

Models	Functions	Sampler	Databases	DataObjects	Steps

```
<Steps>
 <IOStep name='load_seed_dataset'>
    <Input   class='Databases'    type='HDF5'          >out_db</Input>
    <Output  class='DataObjects' type='TimePointSet'>outAdaptive</Output>
  </IOStep>
 <MultiRun name='GoalOrientedStep'>
    <Input   class=DataObjects' type='TimePoint'    >inputPlaceHolder</Input>
    <Model   class='Models'      type='ExternalModel'>PythonModule</Model>
    <Sampler class='Samplers'    type='Adaptive'     >workshopAdaptive</Sampler>
    <Output  class='DataObjects'type='TimePointSet' >outAdaptive</Output>
    <Output  class='Databases'   type='HDF5'         >out_db_adaptive</Output>
    <SolutionExport class='DataObjects'type='TimePointSet' >
      limitSurface
    </SolutionExport >
  </MultiRun>
  <IOStep name='push_limitsurface_in_database'>
    <Input   class='DataObjects'    type='TimePointSet' >limitSurface</Input>
    <Output  class='Databases'      type='HDF5'          >out_db_ls</Output>
  </IOStep>
</Steps>>
```

Input file name: 2-seeded_adaptive_sampling.xml

Exercise 3:
Compute failure probability from Converged solution

3 –Failure Probability from Converged Solution

Models	Sampler	Databases	DataObjects	Steps

```
<Models>
    <ROM  name='probabilityROm' subType='SciKitLearn'>
        <Features>x1,x2,x3</Features>
        <Target>failure</Target>
        <SKLtype>svm|SVC</SKLtype>
        <C>10.0</C>
        <kernel>rbf</kernel>
    </ROM>
    <ExternalModel name='PythonModule' subType=''
      ModuleToLoad='workshop_model'>
       …
    </ExternalModel>
    <PostProcessor name='computePb' subType='BasicStatistics'>
        <what>expectedValue</what>
        <parameters>failure</paramters>
    </PostProcessor>
</Models>
```

Input file name: 3-compute_failure_pb_from_ls.xml

3 –Failure Probability from Converged Solution

Model	Sampler	Databases	DataObjects	Steps

```
<Samplers>
  <MonteCarlo name='Montecarlo_sampling'>
    <sampler_init> <limit>17576</limit> </sampler_init>
    <variable name='x1'>
      <distribution>normal_trunc</distribution>
    </variable>
    <variable name='x2'>
      <distribution>normal</distribution>
    </variable>
    <variable name='x3'>
      <distribution>uniform</distribution>
    </variable>
  </MonteCarlo>
</Samplers>
```

Input file name: 3-compute_failure_pb_from_ls.xml

3 –Failure Probability from Converged Solution

Model	Sampler	Databases	DataObjects	Steps

```
<Databases>
  <HDF5 name="out_ROM3_db"/>
  <HDF5 name="out_db_montecarlo” directory=“DatabaseStorage/"/>
  <HDF5 name="out_db_adaptive" filename="out_db_adaptive.h5”
directory=“DatabaseStorage/"/>
</Databases>
<DataObjects>
    <TimePointSet name='outAdapt_failure'>
        <Input>x1,x2,x3</Input>
        <Output>failure</Output>
    </TimePointSet>
    <TimePointSet name='outGrid_failure'>
        <Input>x1,x2,x3</Input>
        <Output>failure</Output>
    </TimePointSet>
    <TimePointSet name='outMC_failure'>
        <Input>x1,x2,x3</Input>
        <Output>failure</Output>
     </TimePointSet>
</DaDataObjectstas>
```

Input file name: 3-compute_failure_pb_from_ls.xml

Exercise 4: Compute failure probability from Converged solution changing Distributions

4 – Compute failure probability changing distributions

Distributions

```xml
<Distributions>
      <Normal name='normal_trunc'>
        <mean>0.5</mean>
        <sigma>0.1</sigma>
        <lowerBound>0.0</lowerBound>
        <upperBound>1.0</upperBound>
    </Normal>
    <Normal name='normal'>
        <mean>2.0</mean>
        <sigma>0.2</sigma>
    </Normal>
    <Uniform name='uniform'>
        <upperBound>4.0</upperBound>
        <lowerBound>1.0</lowerBound>
    </Uniform>
</Distributions>
```

```xml
<Distributions>
      <Normal name='normal_trunc'>
        <mean>0.5</mean>
        <sigma>0.2</sigma>
        <lowerBound>0.0</lowerBound>
        <upperBound>1.0</upperBound>
    </Normal>
    <Normal name='normal'>
        <mean>1.5</mean>
        <sigma>0.2</sigma>
    </Normal>
    <Uniform name='uniform'>
        <upperBound>4.0</upperBound>
        <lowerBound>1.5</lowerBound>
    </Uniform>
</Distributions>
```

Input file name: 4-compute_pb_with_rom_and_changed_dists.xml

Thank you

Questions?