Crossing the Streams – Sampler and the TemplateEngine

An uncertainty quantification story

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Objectives

- Understand the stochastic sampling approach to uncertainty quantification in Sampler
- Perform a range of uncertainty quantification calculations with SCALE
- Use the SCALE TemplateEngine to generate repeated pieces of input for both KENO and Sampler inputs

Outline

1. Sampler introduction
2. TemplateEngine introduction
3. Mini-assembly problem specification
4. Uncertainty quantification calculations
    1. Bounding approach
    2. Common sampling
    3. Unique sampling

Oak Ridge National Laboratory

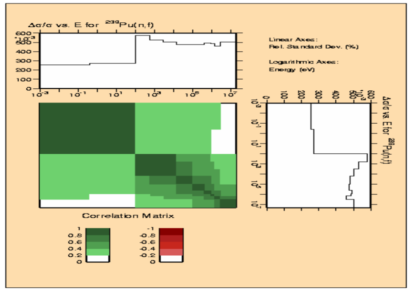
Questions before we get started?

What is Sampler?

- Sampler is a versatile UQ and parametric study tool that can be applied to any SCALE Sequence
- Sampler can perturb any quantity in any SCALE input
- Recent work at ORNL has developed new types of covariance data that allow Sampler UQ to be applied to nearly all SCALE applications
  - Reactor depletion
  - UNF fuel characterization
  - Source term analysis
  - Decay heat calculation
- In SCALE 6.2 releases, CE data in transport *cannot* be perturbed

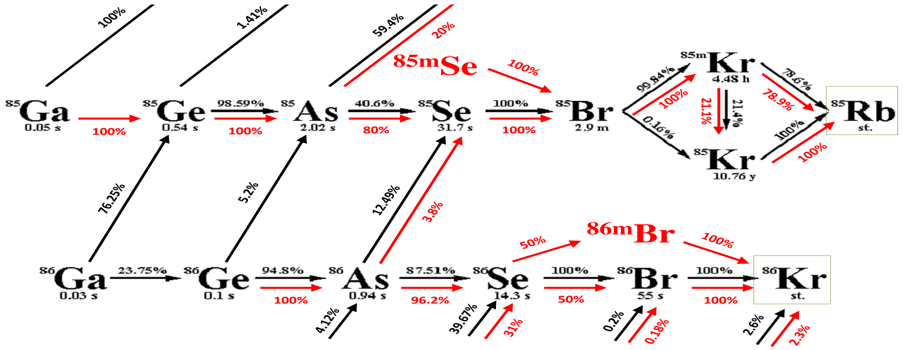
Sources of Sampler Nuclear Data Uncertainty

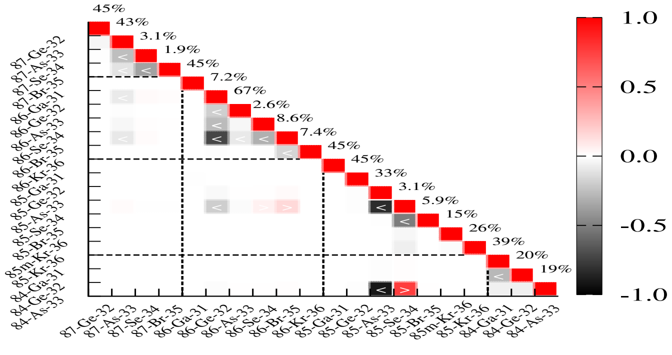
- Cross section covariances:
  - ENDF-VII.1 supplemented by other sources (SCALE cov. library)
- Fission product yield:
  - Standard deviations from ENDF/B-VII; correlations generated by combining independent and cumulative yields
- Decay data:
  - ENDF-VII.1 modified to include branching correlations due to constraint that branch sum=1.0



Pu-239 fission covariance

Mass A=85-87 yields





Yield covariance

OAK RIDGE National Laboratory

Input for nuclear data perturbations is easy

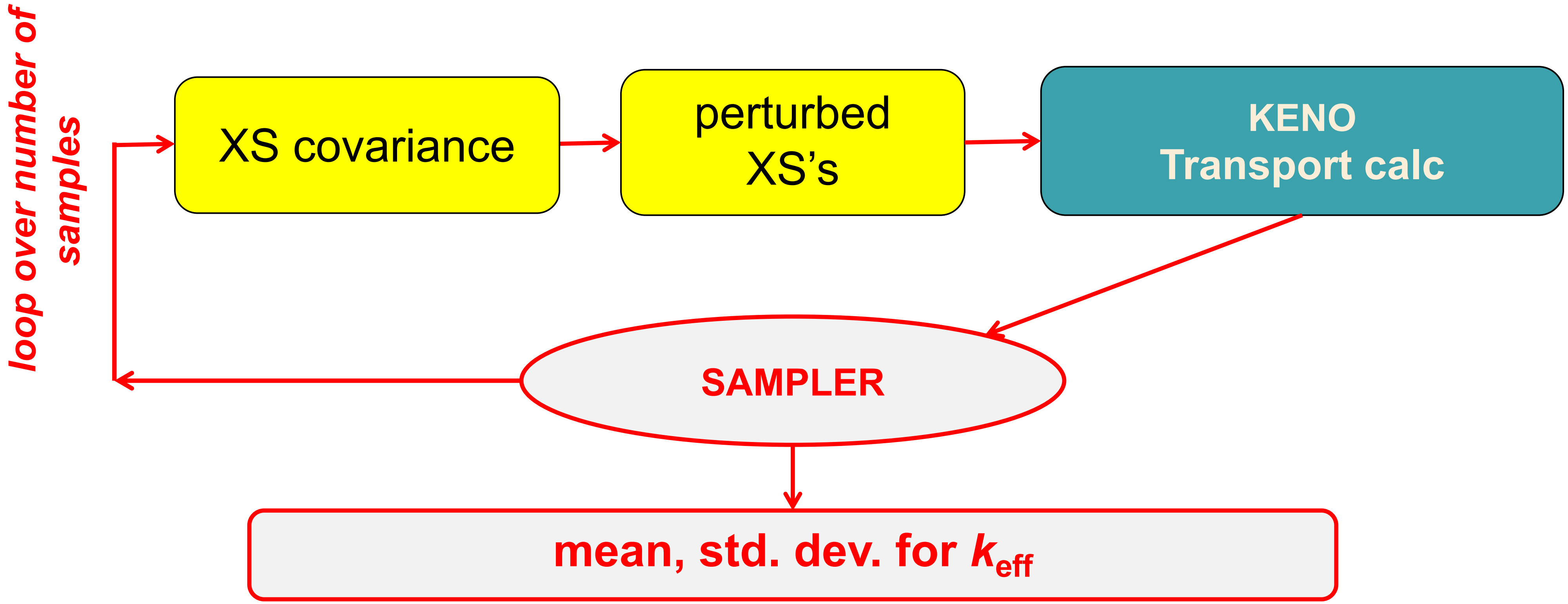
```
read parameters
    n_samples       = 100
    perturb_xs      = yes
    perturb_decay   = yes
    perturb_yields  = yes
end parameters
```

```
=%sampler

read parameters
  n_samples       = 100
  perturb_xs      = yes
  perturb_decay   = yes
  perturb_yields  = yes
end parameters

read case[c1]
  sequence=t-depl  parm=(bonami,addnux=0)
    pincell model
    xn238v7
    read composition
     uo2        10 0.95 900 92235 3.6 92238 96.4   end
     zr-90      20 1 600   end
     h2o        30 den=0.75 0.9991 540   end
    end composition
    read celldata
      latticecell squarepitch pitch=1.2600 30 fuelr=0.4095 10 cladr=0.4750 20 end
    end celldata
    read depletion
      10
    end depletion
    read burndata
      power=25 burn=60 nlib=2 down=30 end
    end burndata
    read model
    read materials
…
```

Nuclear data sampling in KENO



OAK RIDGE National Laboratory

Input placeholders

- Syntax: replace input to be sampled with variable name inside braces with a pound sign prefix in SCALE input #{*variable*}
- Variable name declared in variable block in Sampler input
- Variable names can have:
  - Letters (case sensitive)
  - Numbers
  - Underscores
    - No dashes
  - Must start with a letter
    - Error message for forgetting this constraint is not clear

Placeholders (aka Embedded Input)

SCALE Standard Composition Input

```
read composition
 uranium     1 den=18.742 1 300
       92235 93.7112
       92238 5.2686
       92234 1.0202   end
end composition
```

With Placeholder Variables

```
read composition
 uranium     1 den=18.742 1 300
       92235   #{u235_wo}
       92238   #{u238_wo}
       92234   #{u234_wo}    end
end composition
```

Variable blocks

- Each variable is defined in its own block
  - No limit on number of variable blocks
- Each variable block contains:
  - Variable name
  - Distribution type (more later)
  - Distribution parameters
  - May contain SIREN statement
    - Not discussing SIREN in this workshop due to time constraints
    - Method for substituting variables into input without modifying the SCALE input
  - Cases in which variable is used

Variable block examples

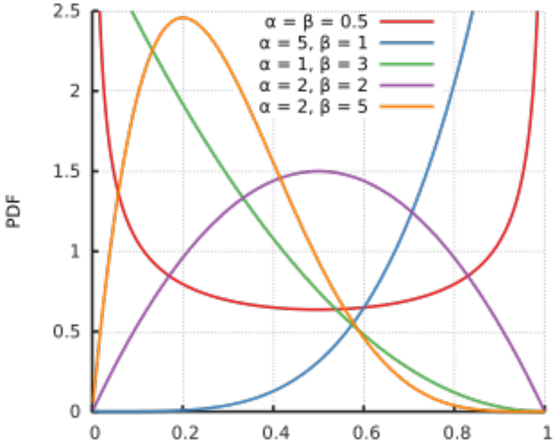
```
read variable[u235_wo]
   distribution=normal
   value=93.7112
   stddev=0.05
   min=93.5
   max=93.9224
   cases= godiva end
end variable
```

```
read variable[u234_wo]
   distribution=uniform
   value=1.0202
   min=1
   max=1.0404
   cases= godiva end
end variable
```

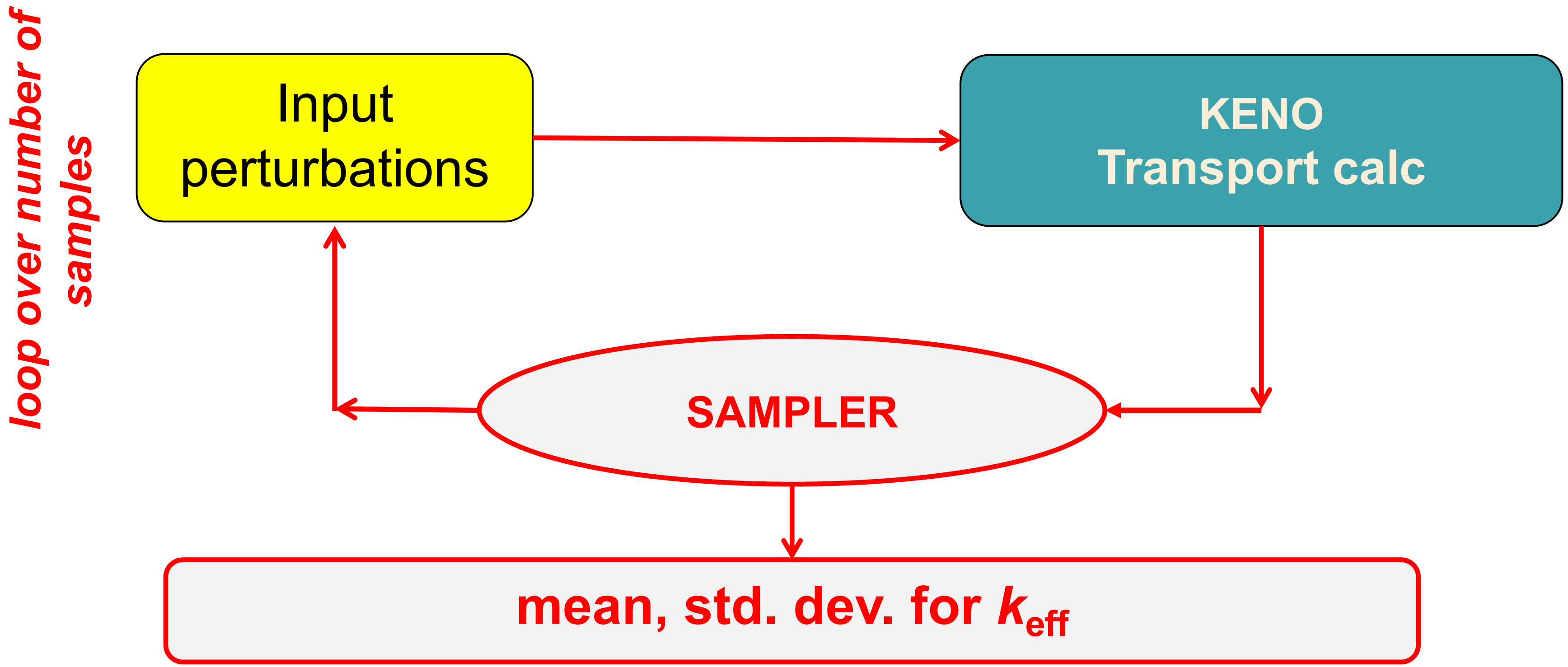
```
read variable[u238_wo]
   distribution=expression
   expression = "100 - u235_wo - u234_wo"
   cases= godiva end
end variable
```

Distributions in Sampler

- 4 options for *distribution*= in a variable block
- Normal
  - Gaussian with specified average and standard deviation
  - Can be truncated
- Uniform
  - Constant probability between max and min
- Beta
  - Tunable distribution with 2 free parameters (α & β)
- Expression



Input sampling in KENO



Questions?

If not, on to the TemplateEngine!

What is the TemplateEngine?

- Tool to expand a template to accelerate model creation
  - Particularly useful for generating repeated copies of a structure
- TemplateEngine introduced in SCALE 6.2
  - Similar TemplateEngine used in UNF-ST&DARDS
  - Uses similar substitution logic to placeholders in Sampler
- Templates can be imported and expanded in-line at time of execution
- Fuel assemblies lend themselves to this approach given repeated structures

Example substitution template

```
' fuelr=#{fuelr=0.47} gapr=#{gapr=0.4875} cladr=#{cladr=0.545} hpitch=#{hpitch=0.82}
unit 1
  zcylinder 101 1 #{fuelr} 365.76  0.0 origin 0 0
  zcylinder 0 1 #{gapr}  372.76  0.0 origin 0 0
  zcylinder 2 1 #{cladr} 383.26 -4.0 origin 0 0
  cuboid 3 1  #{hpitch} -#{hpitch}  #{hpitch} -#{hpitch}  383.26 -4.0
'
unit 2
  zcylinder 102 1 #{fuelr} 365.76  0.0 origin 0 0
  zcylinder 0 1 #{gapr}  372.76  0.0 origin 0 0
  zcylinder 2 1 #{cladr} 383.26 -4.0 origin 0 0
  cuboid 3 1  #{hpitch} -#{hpitch}  #{hpitch} -#{hpitch}  383.26 -4.0
```

Creates:

```
' fuelr=0.47 gapr=0.4875 cladr=0.545 hpitch=0.82
unit 1
  zcylinder 101 1 0.47 365.76  0.0 origin 0 0
  zcylinder 0 1 0.4875  372.76  0.0 origin 0 0
  zcylinder 2 1 0.545 383.26 -4.0 origin 0 0
  cuboid 3 1  0.82 -0.82  0.82 -0.82  383.26 -4.0
'
unit 2
  zcylinder 102 1 0.47 365.76  0.0 origin 0 0
  zcylinder 0 1 0.4875  372.76  0.0 origin 0 0
  zcylinder 2 1 0.545 383.26 -4.0 origin 0 0
  cuboid 3 1  0.82 -0.82  0.82 -0.82  383.26 -4.0
```

Example repetition template

```
#for(i=1; i<=272; i=i+1){
unit #{i+1000}
' bottom end plug
 cylinder 2 1  0.47                     0.97 0.00 origin 0.80 0.80
 cuboid   4 1  1.60 0.0  1.60  0.0  0.97 0.00
unit #{i+2000}
' submerged portion of fuel stack
 cylinder 1 1 0.3946                  73.53  0.0  origin 0.80 0.80
 cylinder 3 1 0.41                    73.53  0.0  origin 0.80 0.80
 cylinder 2 1 0.47                    73.53  0.0  origin 0.80 0.80
 cuboid   4 1 1.60 0.0  1.60 0.0      73.53  0.0
unit #{i+3000}
'fuel above water level
 cylinder 10  1 0.3946                 89.70  73.53  origin 0.80 0.80
 cylinder 30  1 0.41                   89.70  73.53  origin 0.80 0.80
 cylinder 20  1 0.47                   89.70  73.53  origin 0.80 0.80
 cuboid   31  1 1.60 0.0  1.60 0.0     89.70  73.53
unit #{i+4000}
'spring region of fuel rod - in grid plate
 cylinder 30  1 0.41                 96.70   96.45  origin 0.80 0.80
 cylinder 20 1 0.47                 96.70   96.45  origin 0.80 0.80
 cylinder 31 1 0.50                 96.70   96.45  origin 0.80 0.80
 cuboid  5  1 1.60 0.0  1.60 0.0 96.70   96.45
unit #{i+5000}
'top end plug
 cylinder 20 1 0.47                98.20  96.90    origin 0.80 0.80
 cuboid   31 1 1.60 0.0  1.60 0.0  98.20  96.90
}
```

- Creates 272 unique copies of the 5 units for use in an array

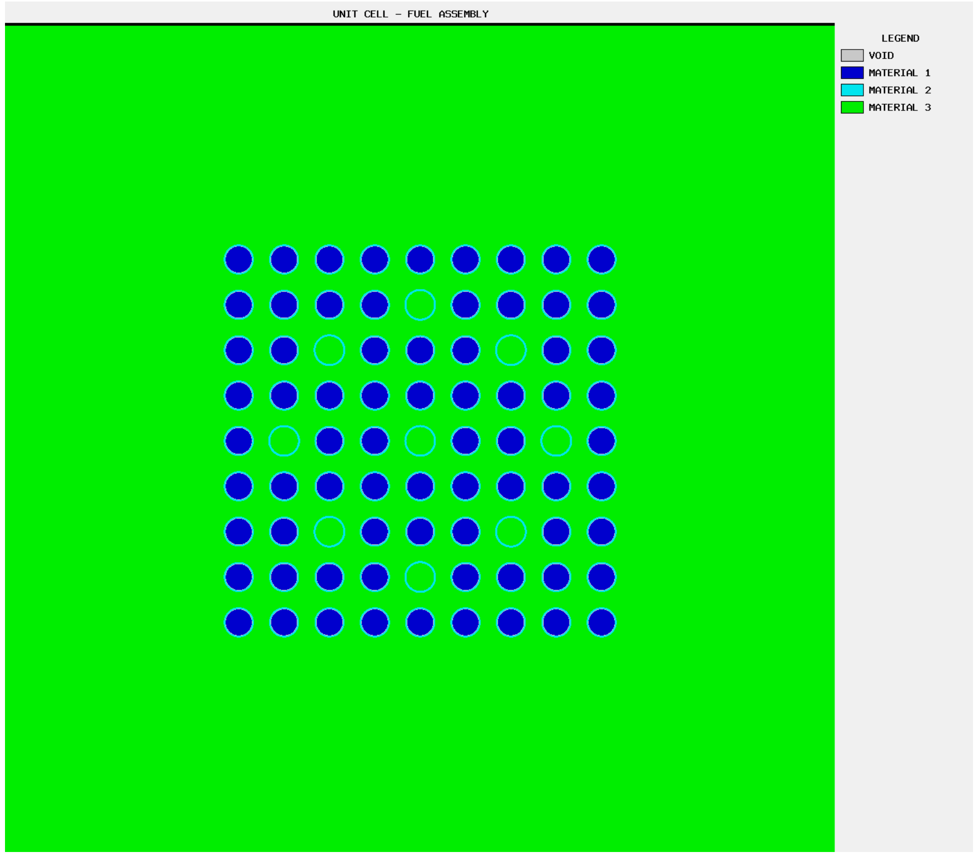
OAK RIDGE National Laboratory

Questions?

If not, on to the problem.

Mini-assembly description

- Problem included in KENO primers
- 9×9 array, including 72 fuel rods and 9 guide tubes
- UO_2 fuel at 3.5 wt% ^{235}U enrichment
- Pitch 1.64 ± 0.02 cm
- MG KENO V.a input provided, named **`nominal.inp`**



UQ discussion

- Examine k_{eff} uncertainty associated with fuel pin pitch
  - Pitch defined as center-to-center spacing within assembly
  - Set in model with rod centered inside cuboid dimensioned ± half pitch
- Three different approaches to quantify the uncertainty
  1. Bounding – run min and max, calculate uncertainty as difference from nominal to higher of the two extreme cases, plus 1.96×uncertainty
  2. Common sampling – run series of random pitch sizes sampled from within the uncertainty range, calculate uncertainty as 1.645×StDev
  3. Unique sampling – run series of random rod placements with each rod positioned using Sampler, calculate uncertainty as 1.645×StDev

Bounding calculations

- Nominal half pitch is 0.82 cm
  - Maximum half pitch is 0.84 cm
  - Minimum half pitch is 0.80 cm
- Input changes are minimal: just change size of cuboids and half-pitch for cross section processing
- $\Delta k = k_{high} - k_{nom} + 1.96\left[\sqrt{\sigma_{high}^2 + \sigma_{nom}^2}\right]$

Note: This equation is for demonstration purposes only. It adds a two-sided **95%** multiplier based on a normal distribution.

Bounding calculations - Results

Case	**Quick Calculation**		**Production Calculation**	
	k_{eff}	σ	k_{eff}	σ
Min (0.80 cm)	0.68073	0.00039	0.68040	0.00009
Nom (0.82 cm)	0.69092	0.00045	0.69064	0.00010
Max (0.84 cm)	0.70017	0.00043	0.70027	0.00010
Δk	0.01047		0.00991	

Bounding calculations - Discussion

- As expected, the assembly is undermoderated so a larger pitch increases reactivity
- A 1% Δk reactivity margin is large
  - Tolerance used here may be ~50% larger than realistic tolerance
  - Linear response, so margin may really be closer to 0.6% Δk
- Only 2 additional, simple calculations are needed
- Can we invest more engineering and computer time and get a smaller uncertainty?

Common sampling

- Generate a number of different assemblies, each with a pitch sampled from the distribution of possible values
- Assume that each assembly is fabricated with a constant pitch
- Run a series of realizations to determine the associated uncertainty
- Problems:
  - Modifying input to work with Sampler
  - More run-time – possible dozens to hundreds of cases needed
  - What distribution to sample from?

Common sampling - Calculations

- Save **`nominal_placeholders.inp`** on your computer
- Provide path inside **`common_Sampler.inp`** on line 10
  - If desired, change number of cases on line 4
- For this exercise, a uniform distribution across the range is used
  - Effect of one normal distribution is also provided
  - Real data could be used if it were available to the analyst
- Run Sampler
  - Sampler creates n_samples perturbed cases and the nominal case
  - Sampler then runs the cases by default

Common sampling - Results

Case	σ_{keff}	Δk
Uniform from 0.80 to 0.84		
10 cases, ±45 pcm	0.00677	0.01114
100 cases, ±45 pcm	0.00576	0.00948
100 cases, ±10 pcm	0.00574	0.00945
Truncated normal, Average=0.82 ± 0.02, Standard Deviation=0.01		
10 cases, ±45 pcm	0.00295	0.00485
100 cases, ±45 pcm	0.00460	0.00756
100 cases, ±10 pcm	0.00458	0.00753

Common sampling - Discussion

- Distribution matters – what can you prove?
- Monte Carlo uncertainty doesn't matter given the magnitude of the reactivity changes caused by the pin pitch changes
- Uniform distribution is no better than bounding approach
  - At least it took a lot more computing time
- Do fuel assemblies really have a uniform pitch?
  - Can we model something more realistic?

Random sampling

- What you've all been waiting for…

1. Create template for KENO fuel rod unit
2. Create template for Sampler fuel rod unit
3. Use TemplateEngine to expand templates
4. Assemble KENO and Sampler inputs
5. Run Sampler to position each rod in the assembly uniquely

Random sampling – KENO template

- Save **`KENO_fuel_rod.tmpl`** on your computer

```
#for(i=1; i<=72; i=i+1){
unit #{i+10}
  zcylinder 1 1 0.47   365.76  0.0 origin 0 0
  zcylinder 0 1 0.4875 372.76  0.0 origin 0 0
  zcylinder 2 1 0.545  383.26 -4.0 origin 0 0
  cuboid 3 1  0.82 -0.82  0.82 -0.82  383.26 -4.0
}
```

Loop over 72 fuel rods

Each rod gets a unique unit number

Normal KENO V.a geometry

Origin must be provided so that it can be changed with Sampler

Random sampling – Sampler template

- Save **`sampler_rod_pos.tmpl`** to your computer
- SIREN statements needed since placeholders can't be passed through the TemplateEngine
  - SIREN: SCALE Input Retrieval Engine
  - Provides path to data to be substituted in Sampler
- Three cylinders (fuel, gap, clad) must remain concentric
  - Same origin substituted for all three via SIREN
- Sample distance and direction of displacement separately
  - X coordinate of origin is displacement × cos(θ)
  - Y coordinate of origin is displacement × sin(θ)

```
#for(i=1; i<=72; i=i+1){
'
' sample displacement distance and theta
'
read variable[displacement_distance_#{i}]
   distribution=uniform
   minimum=-0.02 value=0.0 maximum=0.02
   cases = unique end
end variable
read variable[theta_#{i}]
   distribution=uniform
   minimum=0 value=0 maximum=6.2831853
   cases=unique end
end variable
'
' calculate x and y origins
'
read variable[origin_x_#{i}]
   distribution=expression
   expression="displacement_distance_#{i}*cos(theta_#{i})"
   siren="/csas5/geometry/unit[id='#{i+10}']/zcylinder[1:3]/region_origin[decl='origin']/value[1:5:2]"
   cases=unique end
end variable
read variable[origin_y_#{i}]
   distribution=expression
   expression="displacement_distance_#{i}*sin(theta_#{i})"
   siren="/csas5/geometry/unit[id='#{i+10}']/zcylinder[1:3]/region_origin[decl='origin']/value[2:6:2]"
   cases=unique end
end variable
}
```

Loop over 72 fuel rods

Sample displacement uniformly from -0.02 to +0.02 cm

Sample displacement direction uniformly from 0 to 2π rad

Calculating origin coordinates and SIREN substitutions

TemplateEngine expansion of templates

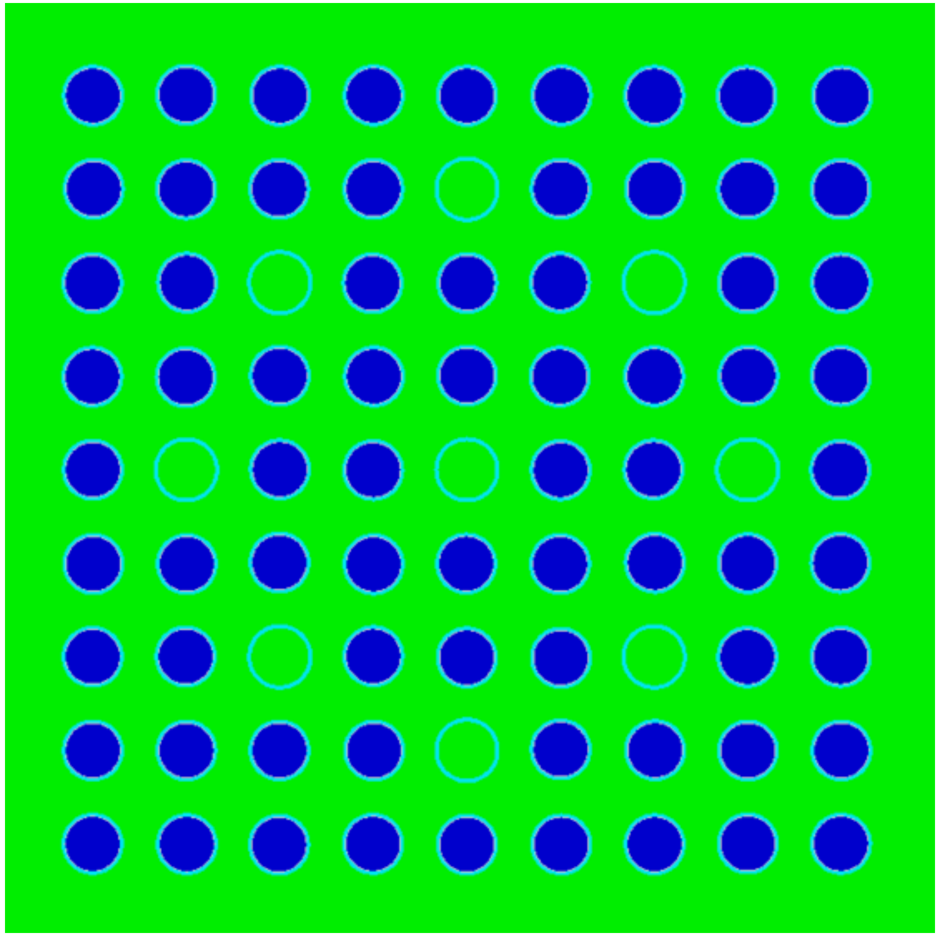
- KENO template
  - /path/to/scale/bin/TemplateEngine KENO_fuel_rod.tmpl > KENO_fuel_rods.part
  - C:\scale6.2.3\bin\TemplateEngine.exe KENO_fuel_rod.tmpl > KENO_fuel_rods.part
- Sampler template
  - /path/to/scale/bin/TemplateEngine sampler_rod_pos.tmpl > sampler_rods.part
  - C:\scale6.2.3\bin\TemplateEngine.exe sampler_rod_pos.tmpl > sampler_rods.part

Assemble KENO and Sampler inputs

- Copy **`nominal_unique.inp`** to your computer
  - KENO skeleton
  - Array filled with unique rod units
  - Insert KENO_fuel_rods.part where indicated around line 24
- Copy **`unique_Sampler.inp`** to your computer
  - Sampler skeleton
  - Provide path to assembled nominal_unique KENO input on line 9
  - Insert sampler_rods.part where indicated around line 15

Run Sampler input to create and run KENO jobs

- Submit **`unique_Sampler.inp`** after assembled
  - Set to create and run 10 perturbed cases and the nominal case
  - Each case creates a png file through the midplane of the assembly
  - These png files can be viewed to see that each rod was moved to unique positions
  - Can also review the 10 perturbed inputs to confirm unique rod positions



Animated GIF of the 10 perturbed pngs

Unique Sampling - Results

Case	σ_{keff}	Δk
Uniform from -0.02 to 0.02		
10 cases, ±45 pcm	0.00033	0.00055
100 cases, ±45 pcm	0.00047	0.00077
100 cases, ±10 pcm	0.00012	0.00020
Truncated normal, Average=0 ± 0.02, Standard Deviation=0.01		
10 cases, ±45 pcm	0.00040	0.00060
100 cases, ±45 pcm	0.00043	0.00071
100 cases, ±10 pcm	0.00012	0.00019

Unique sampling - Discussion

- Essentially no reactivity effect associated with random placement – nearly the same as the KENO uncertainty
  - Can random modeling approach be justified to regulators?
  - What margin needs to be taken?
- Likely a more realistic modeling approach
  - Does this uncertainty matter if uncertainties are considered independently?
- Distribution is less important with these small changes
- This is just an illustration – there are likely other areas this approach could be helpful

Review

- Sampler was originally designed for stochastic sampling with any sequence within SCALE
  - Parametric capability added in SCALE 6.2.2
- Sampler can be used for uncertainty quantification
  - Sample data in static or depletion calculations
  - Sample inputs for uncertainties in compositions and dimensions
- The SCALE TemplateEngine allows for expanding templates to full inputs
- The combination provides a powerful UQ tool

Questions?

Thanks for your attention!