Introduction to the Workshop

Mustafa K. Jaradat , Ph.D.

4/21/2024

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

Idaho National Laboratory

Purpose and Assumptions

The Purpose of this training session is to

- Demonstrate the application of MOOSE-based tools for coupled neutronics and thermal-hydraulics modeling of liquid fuel molten salt reactors (MSRs).
- Developing a multiphysics simulation for a simplified model of the Molten Salt Reactor Experiment (MSRE).
- Coupling neutronics based on the Griffin neutronics code, and thermal hydraulics based on the Pronghorn code.
- Steady-state and a flow driven transient will be modeled with demonstration of the delayed neutron precursor impacts.

Presenters: Mustafa K. Jaradat, Mauricio Tano Retamales, Ting Fei

We assume that attendees are familiar with:

- the discretization of the linearized Boltzmann transport equation
  - Energy (multigroup approximation)
  - Space (CFEM, DFEM)
  - Angle (Discrete ordinates method, diffusion approximation)
- The MOOSE framework
- Paraview application

MSR Workshop Agenda

	Time	Topic	Presenter
1	1:00 – 1:30 pm (30 min):	Introduction to MSRE	M. Jaradat
2	1:30 – 2:00 pm (30 min):	Introduction to cross section preparation & mesh generation for MSRE	T. Fei
3	2:00 – 2:20 pm (20 min):	Steady-state Heterogeneous neutronics model of MSRE with Griffin	T. Fei
4	2:20 – 3:00 pm (40 min):	Thermal-Hydraulics model of the MSRE with Pronghorn (Exercise 1)	M. Retamales
	3:00 – 3:10 pm (10 min):	Coffee Break	
5	3:10 – 3:55 pm (45 min):	Multiphysics steady-state model of MSRE with Griffin & Pronghorn (Exercise 2)	M. Jaradat
6	3:55 – 4:40 pm (45 min):	Multiphysics transient model of MSRE with Griffin & Pronghorn (Exercise 3)	M. Jaradat
7	4:40 – 5:00 pm (20 min):	Demonstration of species transport (Griffin, Pronghorn, Thermochimica) (Exercise 4)	M. Retamales

Resources

- MOOSE framework website – https://mooseframework.inl.gov
- Griffin application website – https://griffin-docs.hpcondemand.inl.gov/
- Griffin application forum – https://griffin-discourse.hpcondemand.inl.gov
- Pronghorn application website https://pronghorn-dev.hpc.inl.gov/site/index.html

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.

Introduction to Molten Salt Reactors Modeling

Mustafa K. Jaradat , Ph.D.

4/21/2024

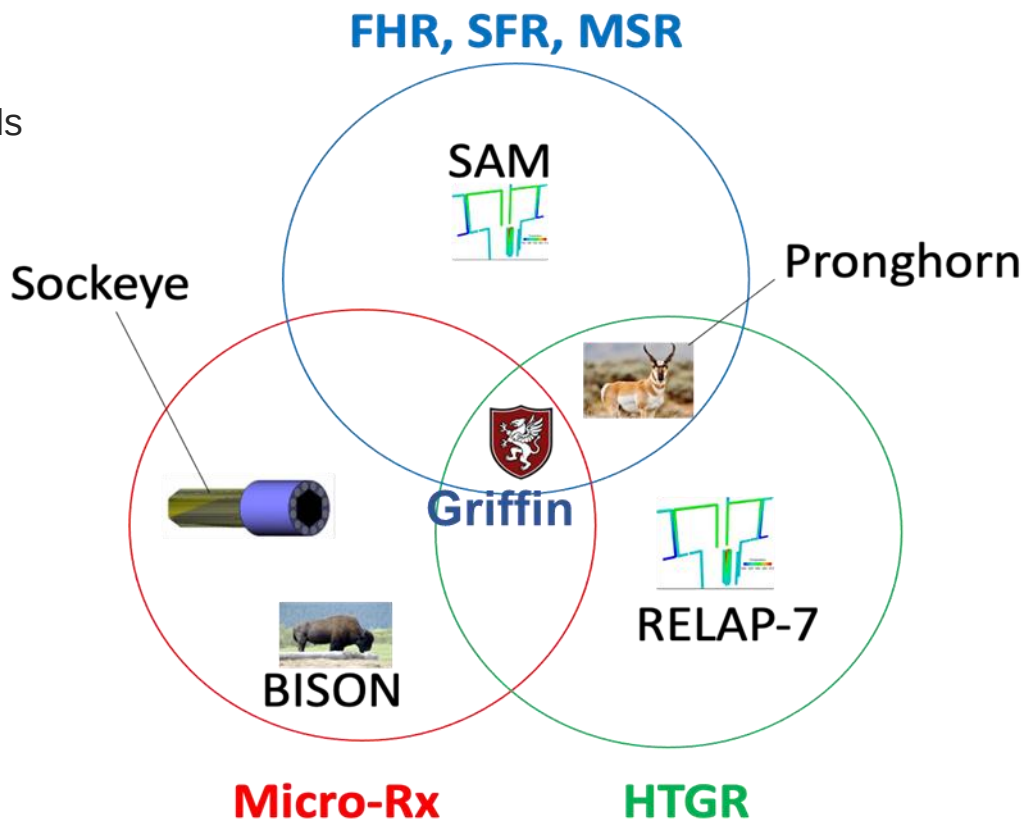
Battelle Energy Alliance manages INL for the
U.S. Department of Energy's Office of Nuclear Energy

Idaho National Laboratory

MOOSE-Based Applications: Griffin-Pronghorn-SAM

- Griffin a MOOSE-based application for
  - A generalized tool for reactor physics/ neutronics (non-LWR reactors)
  - Multiphysics-oriented: Provides native coupling to all MOOSE-based tools
  - Flexible and Extendable
    - Regular and unstructured geometries
    - Easy addition of functionality (e.g., advection term for MSR)

- Pronghorn is a MOOSE-based application for
  - Coarse-mesh tool for thermal-hydraulics calculations.
  - Solves multidimensional compressible/incompressible Euler equations
  - Handle porous and non-porous flow configurations.
  - Model the transport of DNPs with advection through the system.



Part of the U.S. NRC's Comprehensive Reactor Analysis Bundle (CRAB)

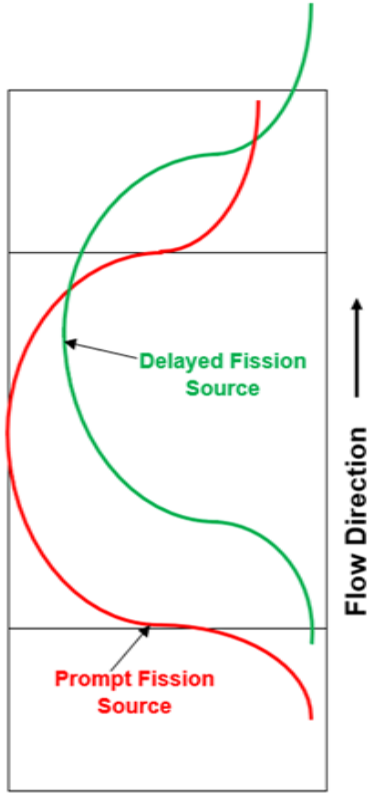
Flowing Fuel Governing Equations of / Steady State

- Multigroup neutron diffusion equation of **stationary fuel** is given as**:**

$$-\nabla \cdot D_g \nabla \phi_g + \Sigma_{tg}\phi_g = \sum_{g'=1}^{G} \Sigma_{sg'g}\phi_{g'} + \frac{\chi_g}{k_{eff}} \sum_{g'=1}^{G} \nu\Sigma_{fg'}\phi_{g'}$$

Total fission neutrons fraction

$$\chi_g = \frac{\chi_{pg} \sum_{g'=1}^{G} \nu_p \Sigma_{fg'}\phi_{g'} + \sum_{k=1}^{K} \chi_{dkg} \sum_{g'=1}^{G} \nu_{dk} \Sigma_{fg'}\phi_{g'}}{\sum_{g'=1}^{G} \nu\Sigma_{fg'}\phi_{g'}}$$



- Multigroup neutron diffusion equation and delayed neutron precursor equation of **flowing fuel** are given as:

$$-\nabla \cdot D_g \nabla \phi_g + \Sigma_{tg}\phi_g = \sum_{g'=1}^{G} \Sigma_{sg'g}\phi_{g'} + \underbrace{\frac{\chi_{pg}}{k_{eff}} \sum_{g'=1}^{G} \nu_p \Sigma_{fg'}\phi_{g'}}_{\text{Prompt Fission Source}} + \underbrace{\sum_{k=1}^{K} \chi_{dkg} \lambda_k C_k}_{\text{Delayed Fission Source}}$$

$$\nabla \cdot [\vec{u}(\vec{r}) C_k(\vec{r})] + \lambda_k C_k(\vec{r}) = \frac{1}{k_{eff}} \psi_k(\vec{r})$$

$$\psi_k(\vec{r}) = \sum_{g'=1}^{G} \nu_{dk} \Sigma_{fg'}(\vec{r}) \phi_{g'}(\vec{r})$$

$$C_k(r,0) = \frac{\int_{A_{out}} dAu(r,H) C_k(r,H) e^{-\lambda_k \tau}}{\int_{A_{in}} dAu(r,0)}$$

Flowing Fuel Governing Equations of / Time Dependent

- **Neutron flux:**

$$\frac{1}{v_g}\frac{\partial}{\partial t}\psi_g(\vec{r},\Omega,t)+\Omega\cdot\nabla\psi_g(\vec{r},\Omega,t)+\Sigma_{tg}(\vec{r},t)\psi_g(\vec{r},\Omega,t)=\sum_{g'=1}^{G}\Sigma_{sg'\to g}(\vec{r},\Omega'\cdot\Omega,t)\psi_{g'}(\vec{r},\Omega',t)$$

$$+\frac{1}{k_{eff}}\underbrace{\frac{\chi_{pg}(\vec{r},t)}{4\pi}\sum_{g'=1}^{G}\nu_p\Sigma_{fg'}(\vec{r},t)\phi_{g'}(\vec{r},t)}_{\text{Prompt Fission Source}}+\underbrace{\frac{1}{4\pi}\sum_{k=1}^{K}\chi_{dkg}(\vec{r},t)\lambda_k C_k(\vec{r},t)}_{\text{Delayed Fission Source}},\quad g=1,2,\cdots,G$$

- **DNPC**

$$\frac{\partial}{\partial t}C_k(\vec{r},t)+\underbrace{\nabla\cdot[\vec{u}(\vec{r},t)C_k(\vec{r},t)]}_{\text{Precursor Drift}}+\lambda_k C_k(\vec{r},t)=\frac{1}{k_{eff}}\sum_{g'=1}^{G}\nu_{dk}\Sigma_{fg'}(\vec{r},t)\phi_{g'}(\vec{r},t),\quad k=1,2,\cdots,K$$

- **Nuclide depletion:**

$$\frac{\partial N_i}{\partial t}+\underbrace{\nabla\cdot[\vec{u}N_i]}_{\text{Nuclide Drift}}+\left(\lambda_i+\underbrace{\lambda_i^R}_{\text{Reprocessing}}\right)N_i+N_i\sum_{g'=1}^{G}\sigma_{a,g'}^{i}\phi_{g'}=\sum_{j=1}^{Niso}\gamma_{ij}'\sum_{g'=1}^{G}N_j\sigma_{g'}^{j}\phi_{g'}+\sum_{j=1}^{Niso}\gamma_{ij}\lambda_j N_j+\underbrace{F_i}_{\text{Refueling}}$$

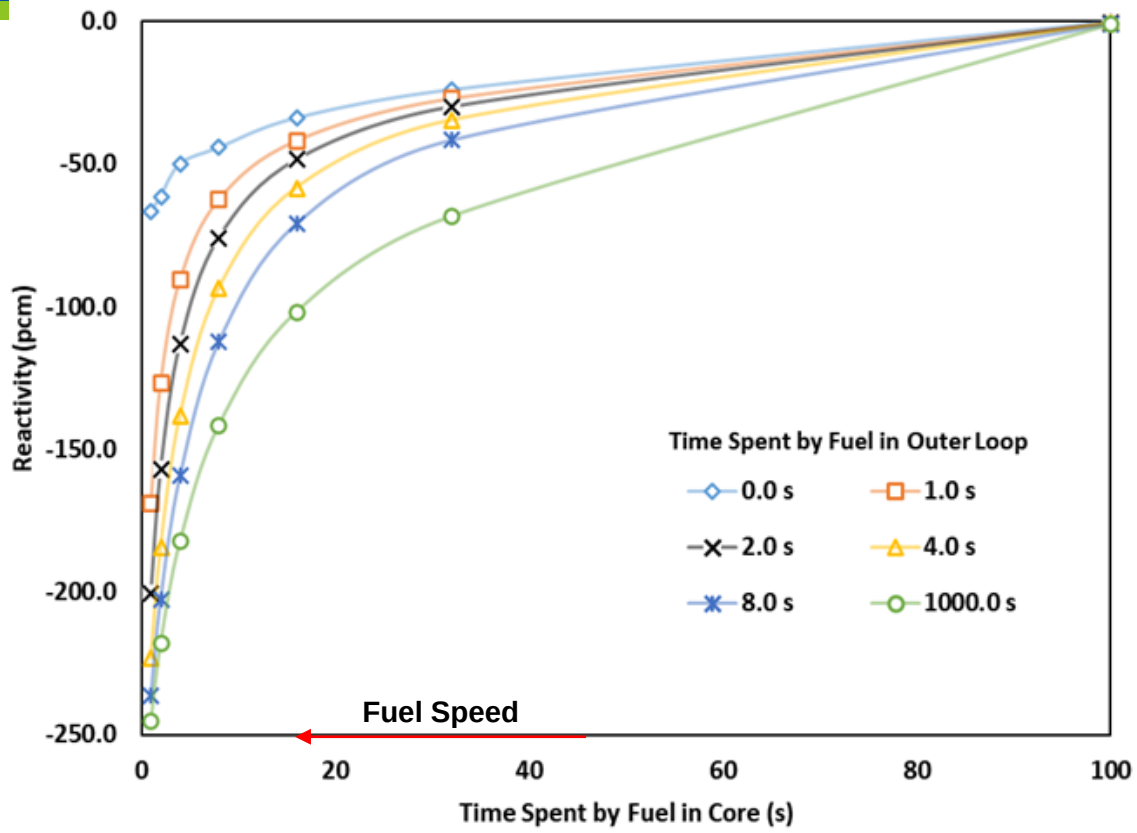
- **Decay Heat:**

$$\frac{\partial}{\partial t}h_k+\underbrace{\nabla\cdot[\vec{u}h_k]}_{\text{Decay Heat Precursor Drift}}+\lambda_k h_k=f_k\sum_{g=1}^{G}\kappa_f\Sigma_{fg}\phi_g$$

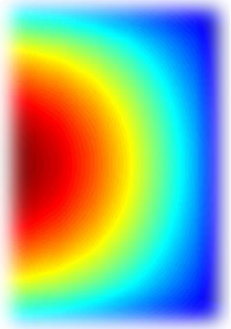
- **Nuclide Distribution:**

$$\frac{\partial}{\partial t}N_i(\vec{r},t)+\underbrace{\nabla\cdot[\vec{u}(\vec{r},t)N_i(\vec{r},t)]}_{\text{Nuclide Drift}}=S_i(\vec{r},t)$$

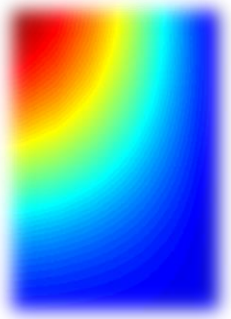
Reactivity Loss with Fuel Flow



$$\beta_{circ} = \frac{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \sum_{k=1}^{K} \chi_{dkg} \lambda_k C_{k,i}}{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \chi_{pg,i} \sum_{g'=1}^{G} \nu_p \Sigma_{fg',i} \phi_{g',i} + \sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \sum_{k=1}^{K} \chi_{dkg,i} \lambda_k C_{k,i}}$$



$$\cancel{\nabla \cdot [\vec{u} C_k]} + \lambda_k C_k = \sum_{g'=1}^{G} \nu_{dk} \Sigma_{fg'} \phi_{g'}$$



$$\beta_{loss} = \beta_{eff} - \beta_{circ} = \rho_{loss}$$

$$\rho_{loss} = \frac{1}{k_{eff}^{circ}} - \frac{1}{k_{eff}^{static}} = \beta_{loss}$$

$$\beta_{eff} = \frac{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \sum_{k=1}^{K} \chi_{dkg} \sum_{g'=1}^{G} \nu_{dk} \Sigma_{fg'} \phi_{g',i}}{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \chi_g \sum_{g'=1}^{G} \nu \Sigma_{fg'} \phi_{g',i}}$$

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.

MSRE Problem Description

Mustafa K. Jaradat , Ph.D.

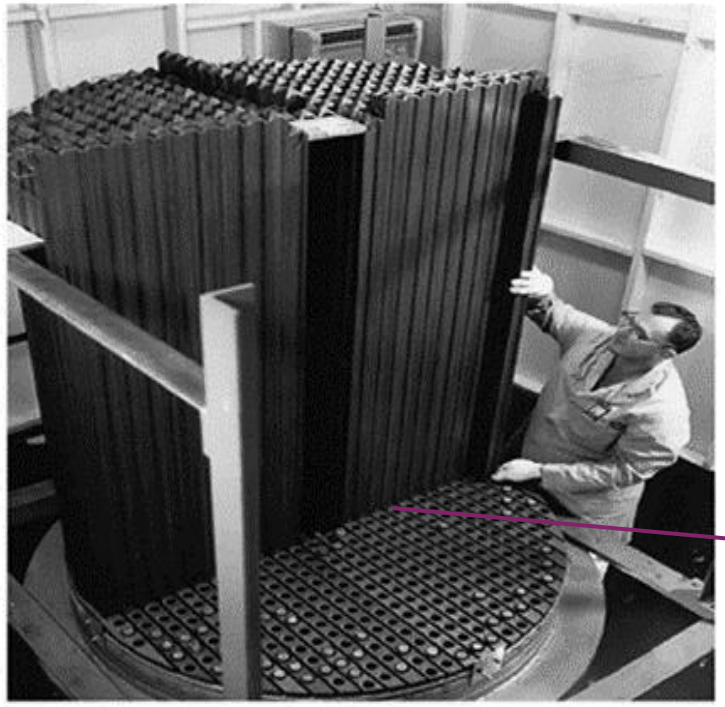
4/21/2024

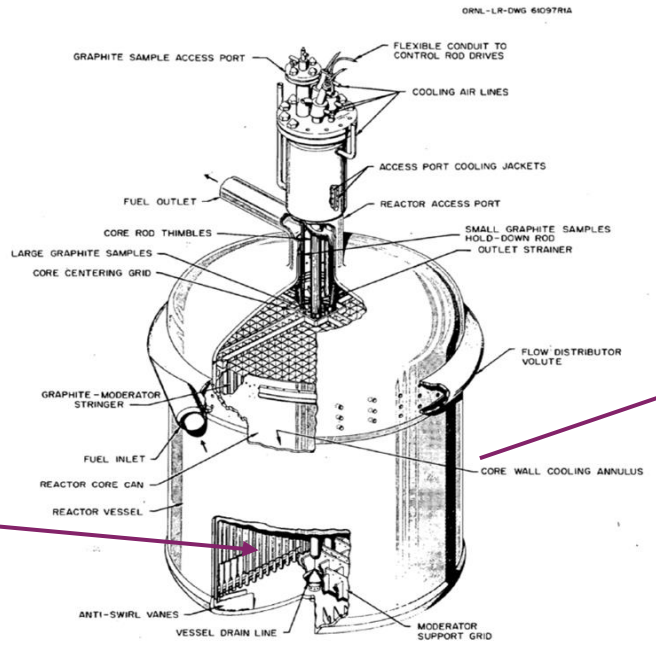
Battelle Energy Alliance manages INL for the
U.S. Department of Energy's Office of Nuclear Energy

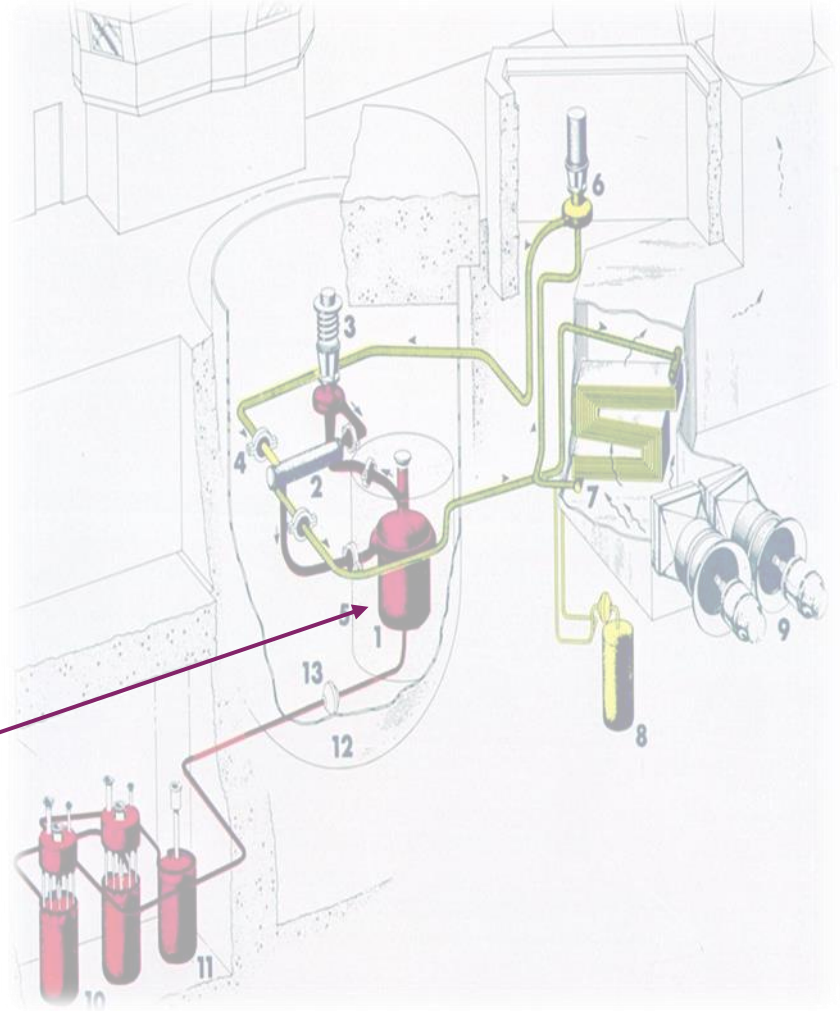
Idaho National Laboratory

Molten Salt Reactor Experiment (MSRE)

- The Molten Salt Reactor Experiment (MSRE) was built in 1964 at ORNL
  - Thermal power of 10 MW.
  - Utilized a thermal neutron spectrum.
  - Liquid fuel salt flow into graphite moderator channels.
  - Operated with U-235 fuel then it was replaced with U-233 fuel.





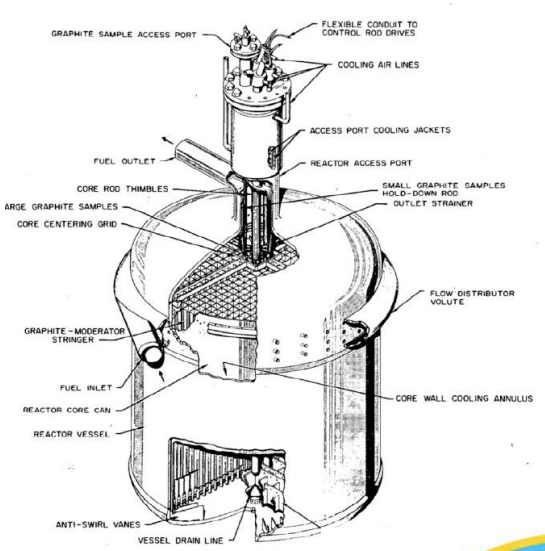


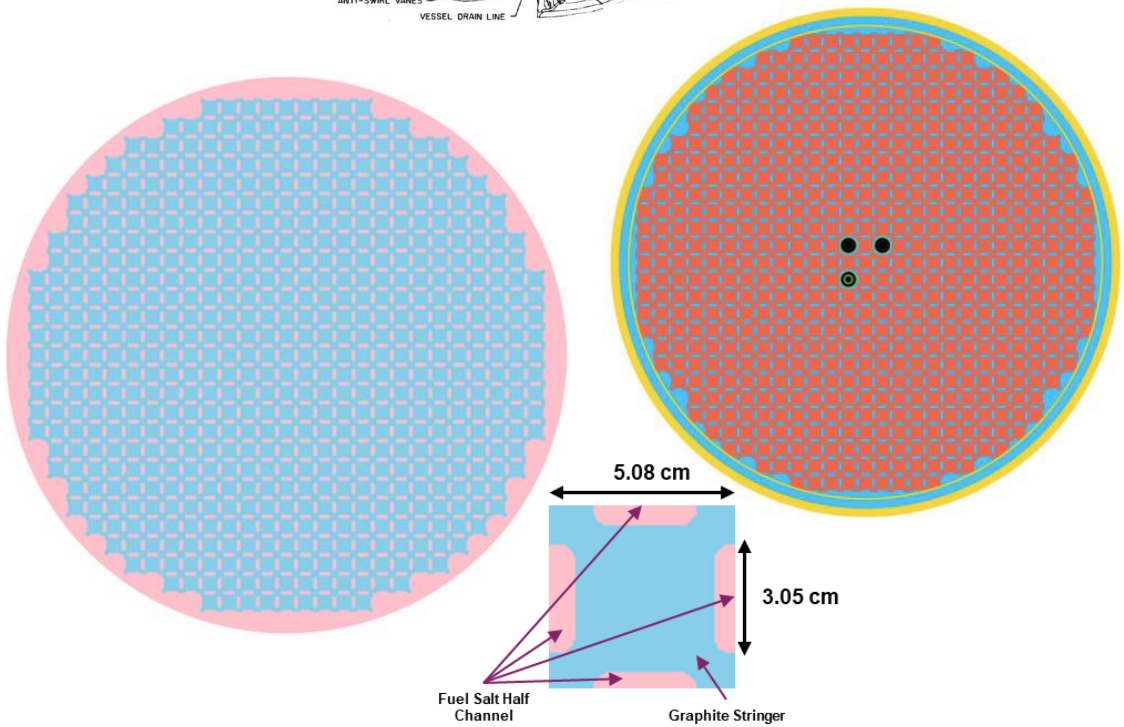
MSRE Modeling

- **MSRE Specifications**
  - Core geometry and material prosperities.
  - Thermo-physical properties of fuel salt and graphite. (ORNL work on experience, database, correlations).

- **Multigroup Cross Sections**
  - Utilized OpenMC code to generate multigroup xs and delayed neutron data.
  - Converted OpenMC-Outputs into readable format by griffin (ISOXML).
  - OpenMC-XS-Griffin converter is available under Griffin code.

- **Simplified MSRE Model with Griffin/Pronghorn**
  - 2D model in RZ geometry.
  - Fuel and graphite in homogeneous mixture.
  - Steady-state and transient verification tests.





MSRE Specifications

MSRE Specifications & Thermal Properties

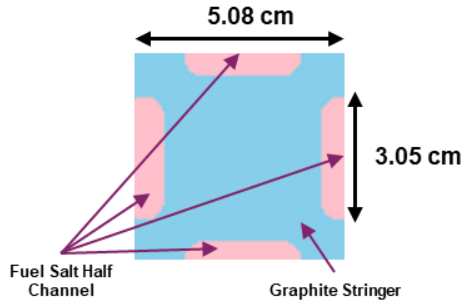
Thermal power	10 MWth
Fuel Salt	$LiF\text{-}BeF_2\text{-}ZrF_4\text{-}UF_4$
Molar composition	65.0%-29.1%-5.0%-0.9%
Enrichment	33.0%
Fuel inlet/ Outlet temperature	908 K / 936 K
Core height / Core diameter	1.63 m / 1.39 m
Fuel salt density [kg/m3] [@ 922K]	ρ(T)= 2263-0.4798x(T(K)-923.0) [2263.5]
Fuel salt dynamic viscosity [Pa·s]	μ=0.263371
Fuel salt thermal conductivity [W/m·K]	k=1.4
Fuel salt specific heat [J/kg·K]	C_p= 1868.0
Graphite density [kg/m3]	ρ_g=1860.0
Graphite thermal conductivity [W/m·K]	k_g=40.1
Graphite specific heat [J/kg·K]	$C_{p,g}$= 1757.3

Fuel Salt composition

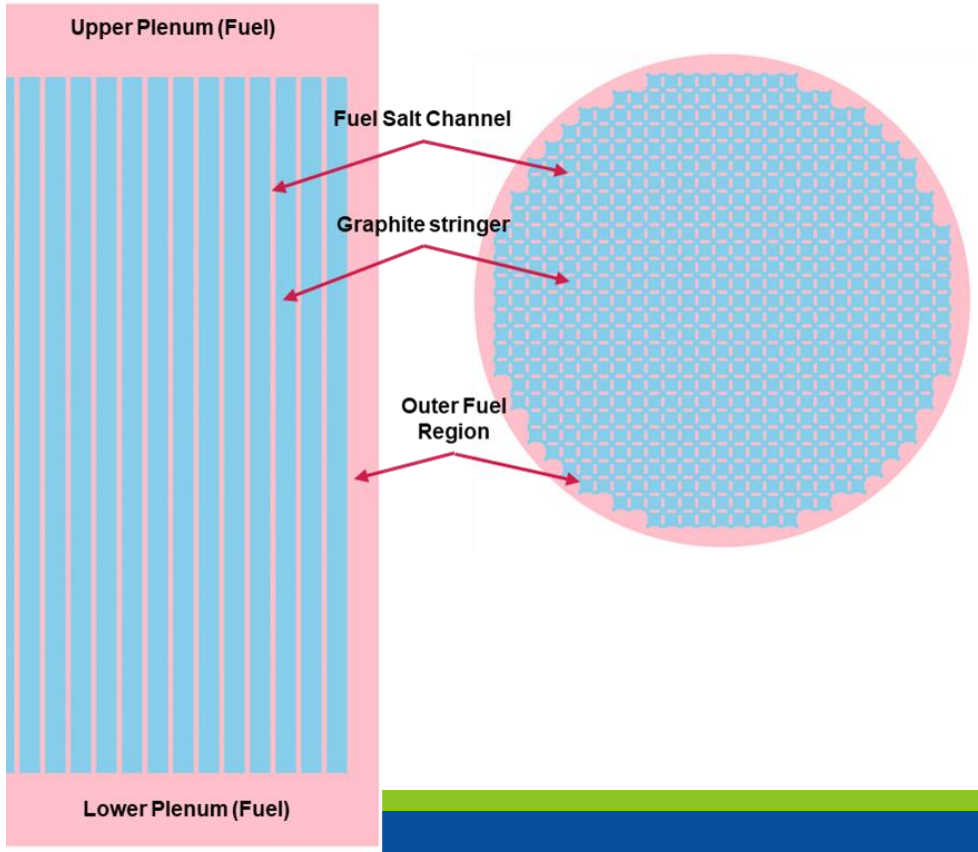
Isotope	Atom Fraction
Li-7	2.63371E-01
F-19	5.94814E-01
Be-9	1.17909E-01
Zr-90	1.04234E-02
Zr-91	2.27310E-03
Zr-92	3.47447E-03
Zr-94	3.52107E-03
Zr-96	5.67260E-04
U-235	1.20340E-03
U-238	2.44327E-03

MSRE Cross Sections Generation Model

- The MSRE lattice is made of vertical graphite stringers
- The fuel salt flows through a rectangular channel (3.05 cm x1.016 cm with round corners of radius 0.508 cm) in the sides of the stringers.



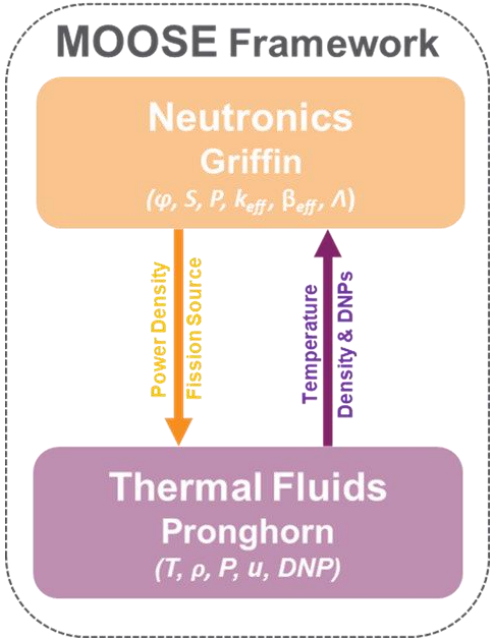
- 3D core model for MSRE was developed with OpenMC code:
  - MG XS generation.
  - Model verification.
- XS were generated in 16 groups energy structure and 6 delayed precursor groups.



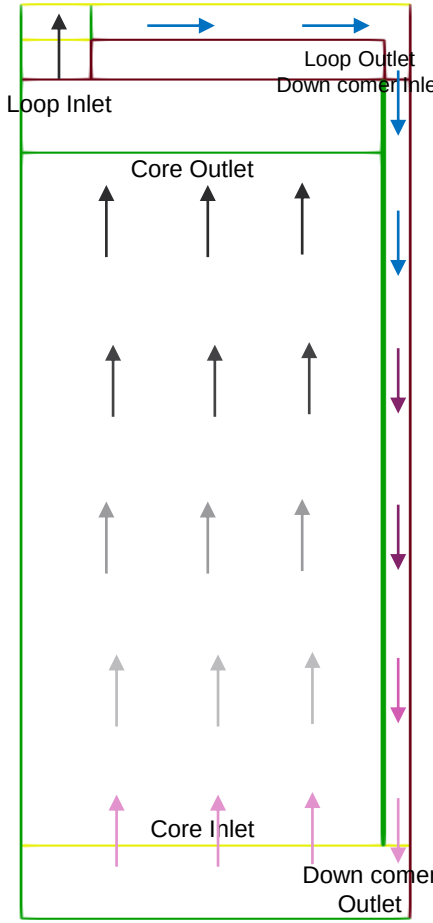
Core Height	197.3
Outer Core Diameter (Stringers +Outer Fuel)	148.7
Inner Core Diameter (Stringers)	139.58
No. Graphite Stringers	593
Size Stringers	5.08 cm x 5.08 cm x 163.0 cm

MSRE Multiphysics Model

- **A Multiphysics** model of the **MSRE** was developed in **RZ** geometry including the following components:
  - **Neutronics core model of core: Griffin**
  - **Thermal hydraulics core & loop models: Pronghorn**



- **Three feedback mechanisms**:
  - **Temperature**: Fuel Salt and Graphite.
  - **Density**: concentration of the salt nuclides due to salt expansion.
  - **Velocity**: delayed neutron precursors distributions in core & outer loop.

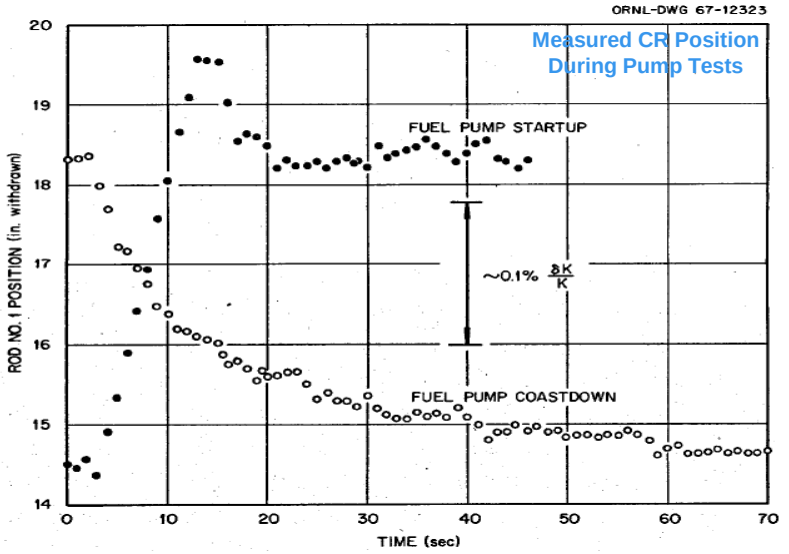


Test Results

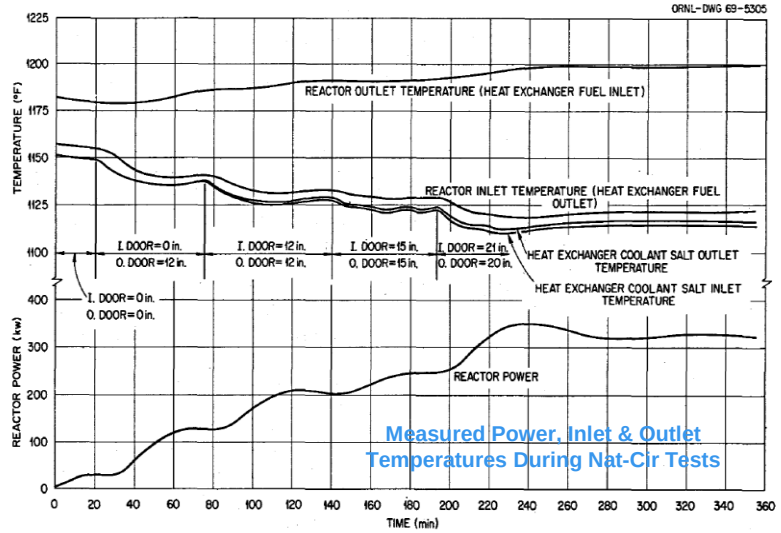
Transient Tests

- **Validation Tests**
  - Pump Startup Test.
  - Pump Coast-Down Test.
  - Natural Circulation Test.

- **Variety of Verification Tests**
  - Pump Driven Transients
    - Loss of flow / coast down
    - Pump over speed / startup
  - Temperature Driven Transients
    - Loss of heat sink
    - Fuel salt over cooling
  - Reactivity Driven Transients
    - Control rod withdrawal
    - Fissile nuclide injection
    - Fertile nuclide injection
  - Localized Transients
    - Channel blockage/unblockage
    - Control rod withdrawal

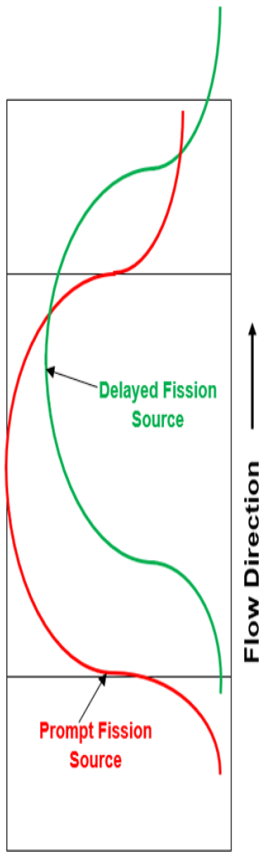


Measured CR Position During Pump Tests



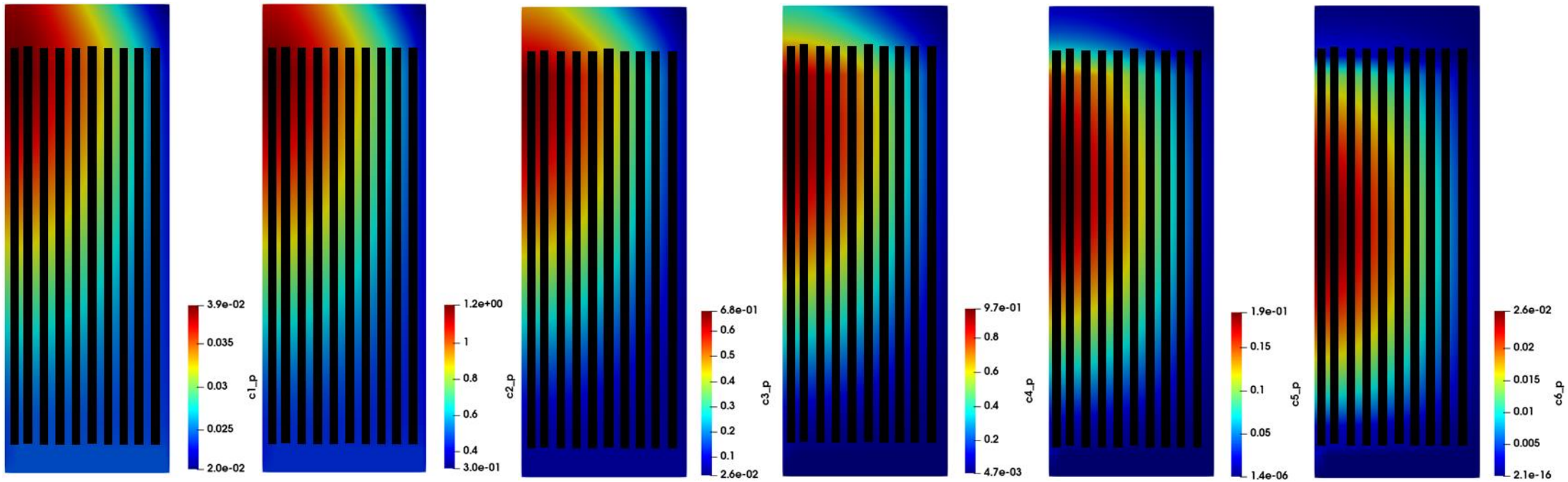
Measured Power, Inlet & Outlet Temperatures During Nat-Cir Tests

Delayed Neutron Precursors Steady State Solution



Long Lived → **Short Lived**

$\lambda_1 = 0.013\ s^{-1}$	$\lambda_2 = 0.033\ s^{-1}$	$\lambda_3 = 0.121\ s^{-1}$	$\lambda_4 = 0.303\ s^{-1}$	$\lambda_5 = 0.849\ s^{-1}$	$\lambda_6 = 2.853\ s^{-1}$
$T_{1/2\_1} = 51.976\ s$	$T_{1/2\_2} = 21.172\ s$	$T_{1/2\_3} = 5.739\ s$	$T_{1/2\_4} = 2.289\ s$	$T_{1/2\_5} = 0.816\ s$	$T_{1/2\_6} = 0.243\ s$



Flowing	$\beta_1 = 0.000006$	$\beta_2 = 0.000374$	$\beta_3 = 0.000535$	$\beta_4 = 0.001846$	$\beta_5 = 0.001048$	$\beta_6 = 0.000481$
Stationary	$\beta_1 = 0.000237$	$\beta_2 = 0.001218$	$\beta_3 = 0.001163$	$\beta_4 = 0.002592$	$\beta_5 = 0.001068$	$\beta_6 = 0.000447$
Losses	$\beta_1 = 0.000231$	$\beta_2 = 0.000844$	$\beta_3 = 0.000628$	$\beta_4 = 0.000746$	$\beta_5 = 0.000020$	$\beta_6 = 0.000034$

Calculated total reactivity losses due to fuel salt flow is 240 pcm.

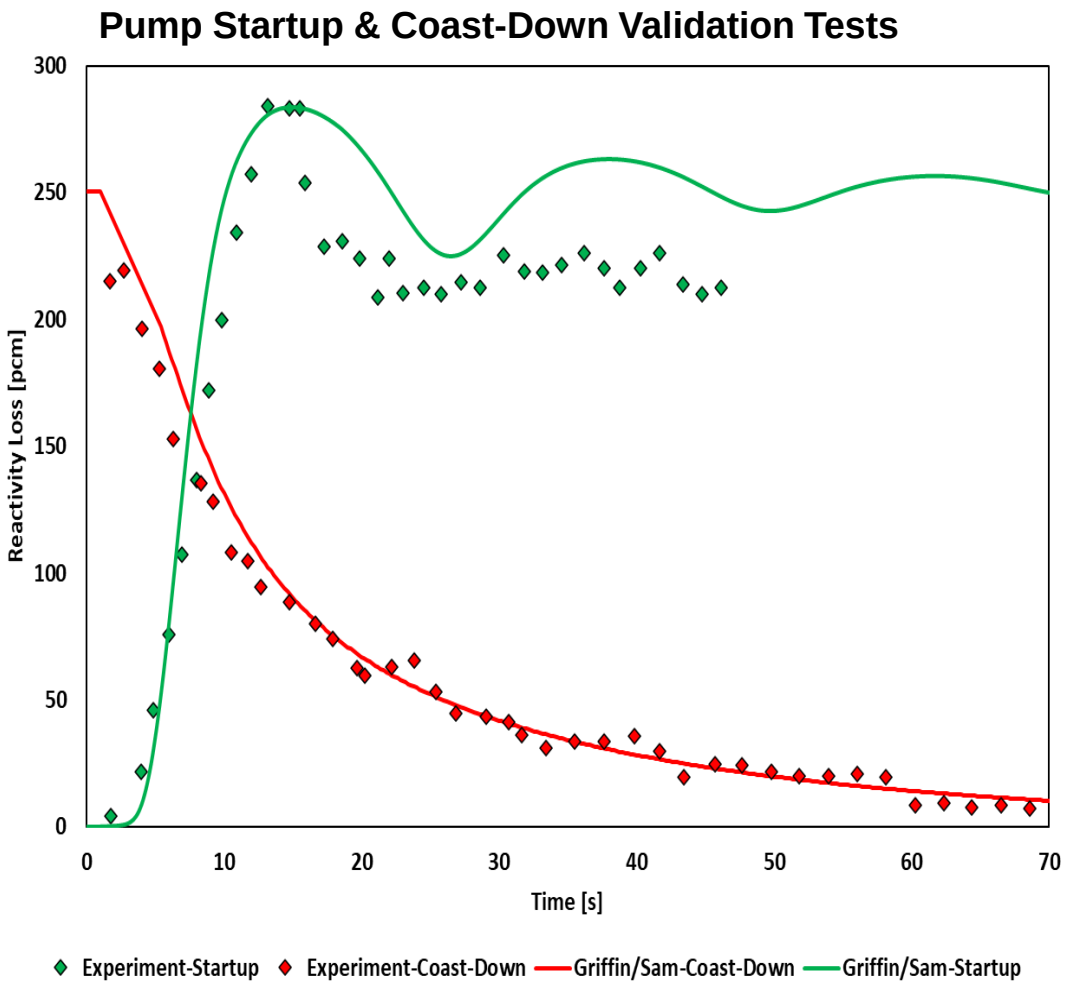
M. Jaradat, J. Ortensi, Thermal spectrum molten salt-fueled reactor reference plant model, Idaho National Laboratory, INL/RPT-23-72875, 2023.

Transient Validation Tests / Pump Transients

- **Pump Startup** & **Coast-Down** transient experiments were performed to demonstrate the impact of the delayed neutrons flow on reactivity.

- During the zero-power experiment:
  - Initial equilibrium conditions were established with minimal flow rate.
  - Reactor was at low or zero power level.
  - No temperature changes of the fuel salt and graphite.
  - **DNP losses (Fuel Salt Velocity)** is only feedback mechanism.
  - Criticality was maintained at zero power by adjusting control rod positions.
  - Reactivity inserted by control rods compensates reactivity changes due to redistribution of the DNPs in the core region.

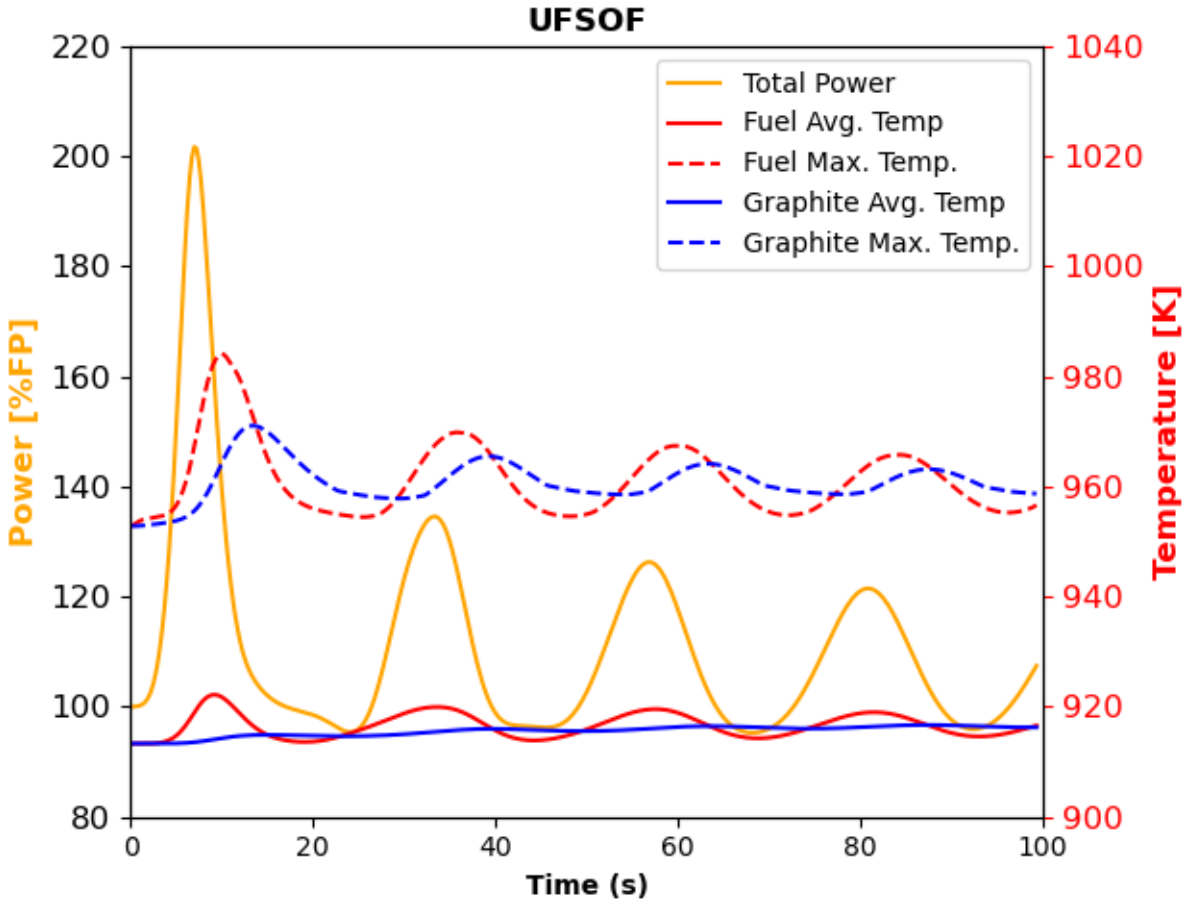
- Multiphysics transient simulations were performed to validate the developed model of MSRE against experimental data.
  - Initial equilibrium conditions were established.
  - Minimal flow (**0.01% Nominal Value**).
  - Zero power state (**0.01% Full Power**).
  - Both pump transient scenarios were performed with the same simulation.
  - Mass flow rate was adjusted similar to the measured values.

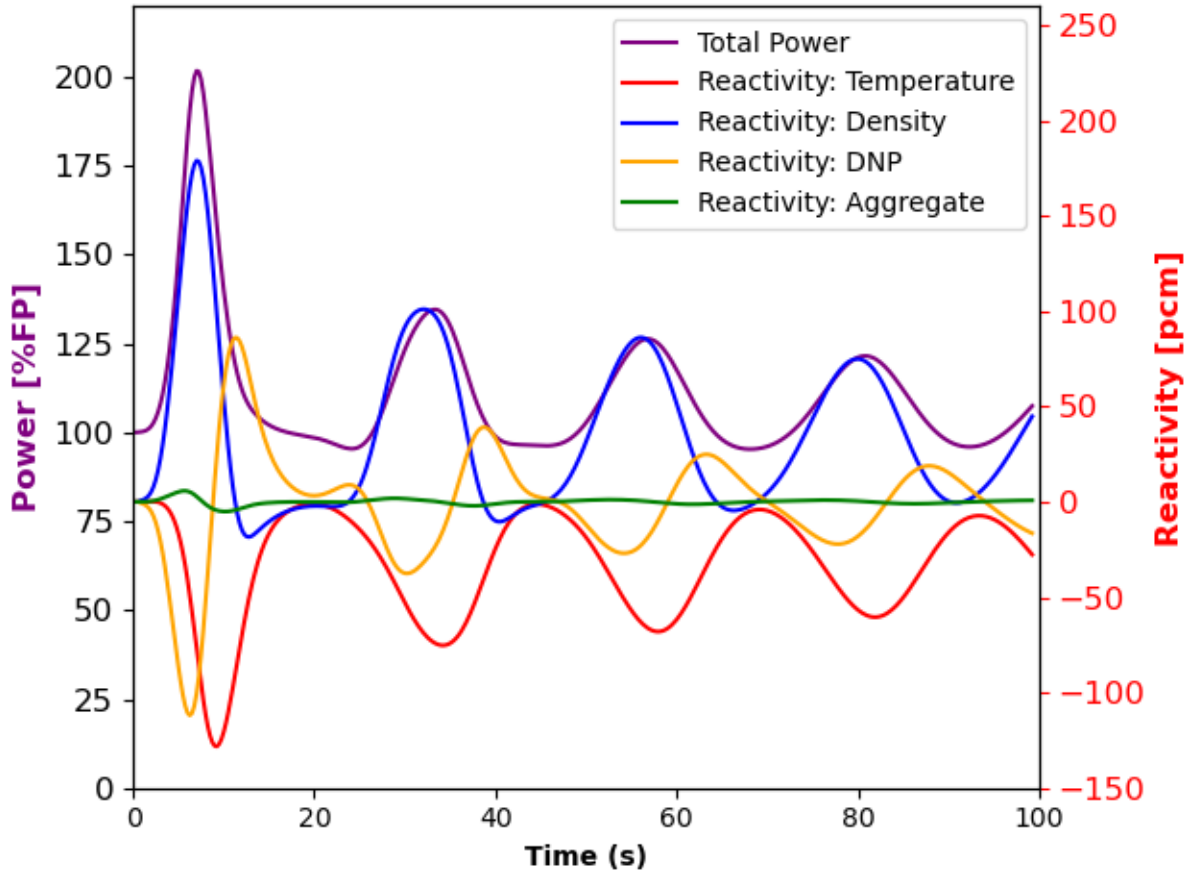


M. Jaradat, J. Ortensi, Thermal spectrum molten salt-fueled reactor reference plant model, Idaho National Laboratory, INL/RPT-23-72875, 2023.

Unprotected Fuel Salt Over Fueling

0.2 kg of U-235 in 4.0 s

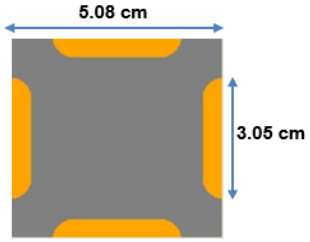




Cross Section Generation

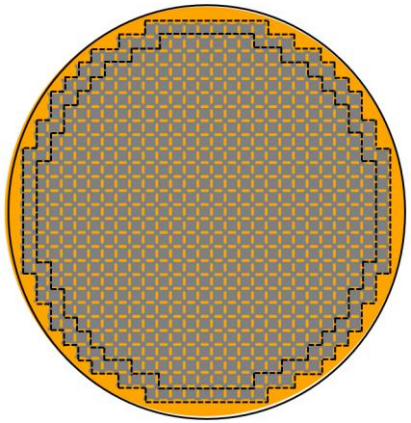
Selection of XS Generation Model

OpenMC Models for Multigroup Cross Sections Generation.

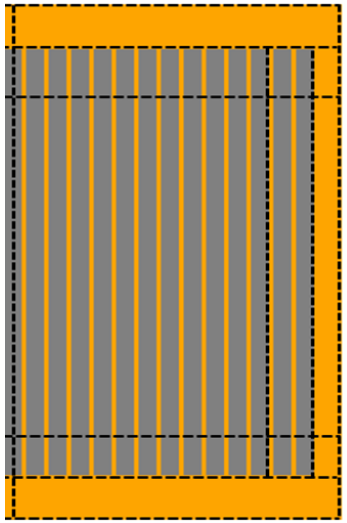


Inner Region Unit Cell

Outer Region Unit Cell



Radial Layout of 2-D and 3-D Models



Axial layout of 3-D Model

Comparison of Eigenvalues and Leakage Fractions Obtained with Different Cross Section Models

XS Model	Multiplication Factor	Diff. (pcm)	Leakage Fraction	Diff. (%)
OpenMC	1.06252 ± 0.00011	-	0.3362	-
3-D	1.06232	-19.6	0.3362	0.007
2-D	1.06487	235.9	0.3352	-0.294
Unit Cell	1.06993	741.4	0.3327	-1.035

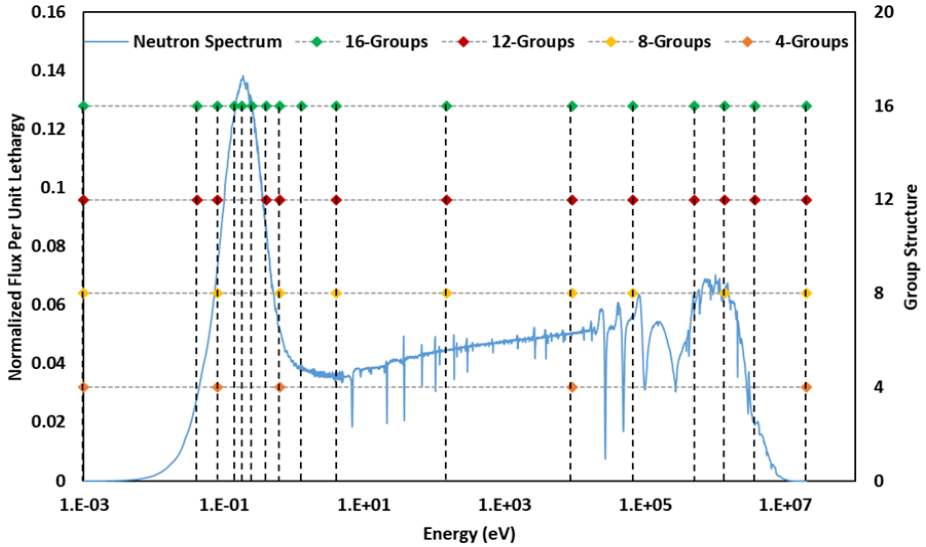
Reactivity Errors Analysis (pcm)

XS Model	Energy Range	Reaction/Leakage				
		Fission	Absorption	Leakage	Scattering	Sum
3-D	**Fast**	-0.1	0.0	-132.7	6.0	-126.8
	Resonance	-36.3	-34.0	67.3	0.0	-3.0
	Thermal	36.4	24.4	49.4	0.0	110.2
	Total	0.0	-9.6	-16.0	6.0	-19.6
2-D	**Fast**	1.2	2.0	-187.7	-2.1	-186.6
	Resonance	177.2	142.3	388.6	0.0	708.1
	Thermal	-178.3	-87.3	-19.9	0.0	-285.5
	Total	0.0	57.0	181.0	-2.1	235.9
Unit Cell	**Fast**	7.1	24.3	-9.3	-26.1	-3.9
	Resonance	233.0	245.7	601.3	0.0	1079.9
	Thermal	-240.1	-117.9	23.5	0.0	-334.6
	Total	0.0	152.0	615.4	-26.1	741.4

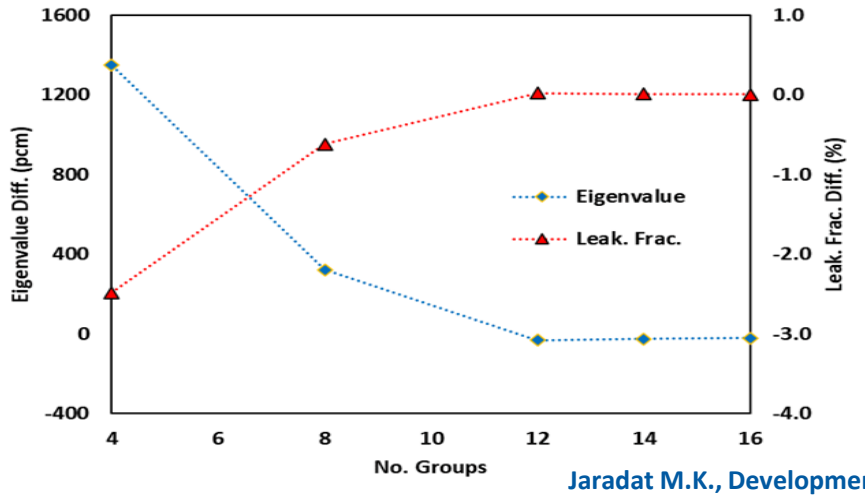
Jaradat M.K., Development of Neutronics Analysis Capabilities for Application to Flowing Fuel Molten Salt Reactors, Ph.D. thesis, University of Michigan (2021)

Selection of Energy Group Structure

Neutron spectrum in fuel salt and selected energy group structures.



Trends of eigenvalue and leakage fraction errors



Selected Energy Group Structures for MSRE Analysis

16-Groups	14-Groups	12-Groups	8-Groups	4-Groups
2.000E+07	2.000E+07	2.000E+07	2.000E+07	2.000E+07
3.679E+06	3.679E+06	3.679E+06		
1.353E+06	1.353E+06	1.353E+06	1.353E+06	
5.000E+05	5.000E+05	5.000E+05		
6.734E+04	6.734E+04	6.734E+04	6.734E+04	
9.118E+03	9.118E+03	9.118E+03	9.118E+03	9.118E+03
1.487E+02	1.487E+02	1.487E+02	1.487E+02	
4.000E+00	4.000E+00	4.000E+00	4.000E+00	
1.300E+00	1.300E+00			
6.250E-01		6.250E-01	6.250E-01	6.250E-01
4.000E-01	4.000E-01	4.000E-01		
2.500E-01	2.500E-01			
1.800E-01				
1.400E-01	1.400E-01			
8.000E-02	8.000E-02	8.000E-02	8.000E-02	8.000E-02
4.200E-02	4.200E-02	4.200E-02		
1.000E-03	1.000E-03	1.000E-03	1.000E-03	1.000E-03

Jaradat M.K., Development of Neutronics Analysis Capabilities for Application to Flowing Fuel Molten Salt Reactors, Ph.D. thesis, University of Michigan (2021)

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.

MSRE Cross section generation

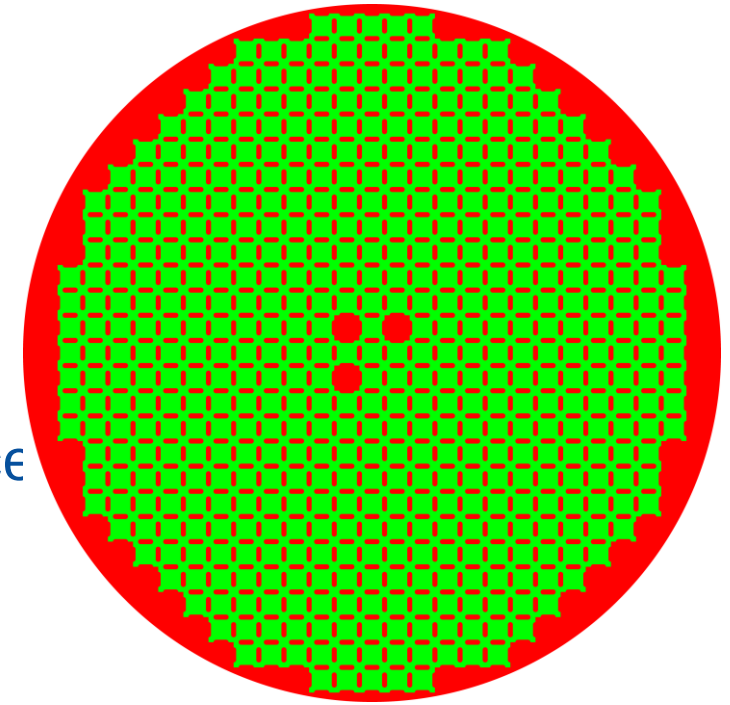
T. FEI, S. K. LEE, K. MO, Y. CAO, C. LEE

Argonne National Laboratory

Openmc mgxs Introduction

- Using OpenMC MGXS module to automatically create multi-group tallies.
  - Manual: https://docs.openmc.org/en/stable/pythonapi/mgxs.html
- Information required
  - Energy structure
  - Cells/Universes/Materials of interest
  - Reaction type: total, transport, nu-transport, absorption, reduced absorption, capture, fission, nu-fission, kappa-fission, scatter, nu-scatter, scatter matrix, nu-scatter matrix, multiplicity matrix, nu-fission matrix, scatter probability matrix, consistent scatter matrix, consistent nu-scatter matrix, chi, chi-prompt, inverse-velocity, prompt-nu-fission, prompt-nu-fission matrix, current, diffusion-coefficient, nu-diffusion-coefficient
  - macro or micro

MSRE Cross section generation (I)



- Step 1: create mgxs.Library
  - domains are the cells in the fuel lattice and control rod lattice
  - geometry is the openmc.Geometry object
  - scatt_order = 1
  - domain_type is cell
  - by_nuclide is True

```
lib = mgxs.Library(geometry)
lib.energy_groups = mgxs.EnergyGroups(groups)
lib.mgxs_types = ["total", "absorption", "nu-fission", "fission", "chi",
                  "consistent nu-scatter matrix", "consistent scatter matrix",
                  "kappa-fission"]
lib.correction = None
lib.legendre_order = scatt_order
lib.domain_type = domain_type
lib.domains = domains
lib.by_nuclide = by_nuclide

lib.build_library()
```

MSRE Cross section generation (II)

- Step 2: add the tallies to the openmc tallies object

```
mgxs_lib = PrepareLibrary(talcl,'cell',1,energy,geom,True)
tals = omc.Tallies()
mgxs_lib.add_to_tallies_file(tals,merge=True)
tals.export_to_xml()
```

- Step 3: run openmc
- Step 4: load openmc statepoint file back into the magx.Library and use external scripts to convert the cross section into isoxml format
  - convert to ISOTXS, then use Griffin utility program to convert ISOTXS to isoxml.
  - direct conversion to isoxml using Griffin utility program.

MSRE Mesh generation

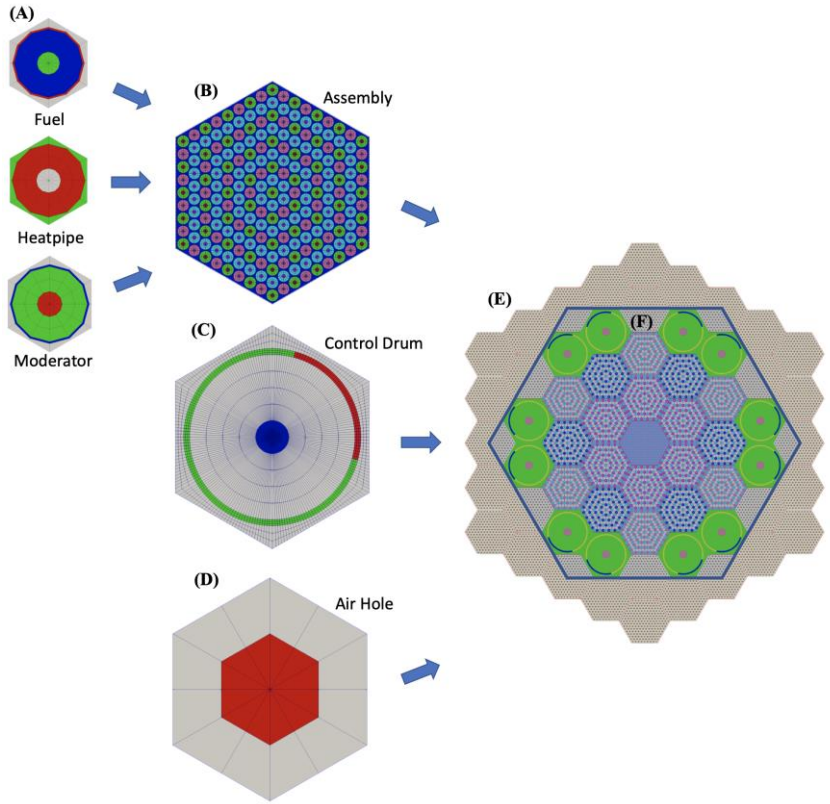
T. FEI, S. K. LEE, K. MO, Y. CAO, C. LEE

Argonne National Laboratory

mesh system & MOOSE Reactor Module

Open Source Meshing Capabilities

- MOOSE mesh system is using MOOSE programmatic interfaces for “online generation” mesh generation
  - Manual: https://mooseframework.inl.gov/syntax/Mesh/
- MOOSE Reactor Module is an open-source reactor meshing capability that consists of a series of new mesh generators to enable meshing of reactor cores provides
  - Manual: https://mooseframework.inl.gov/modules/reactor/
  - Rapidly build up Hexagonal and Cartesian geometries with concise input. Can leverage advanced routines for more unique geometries.

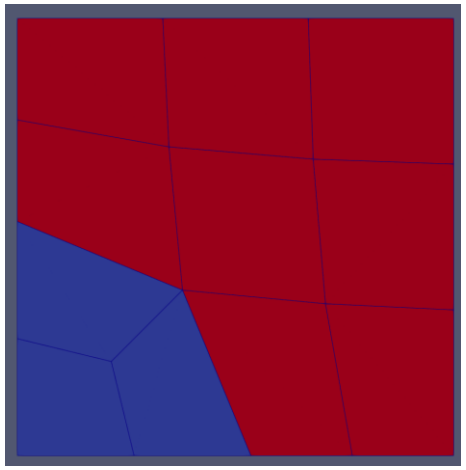


Empire heat-pipe cooled microreactor mesh with rotating control drums

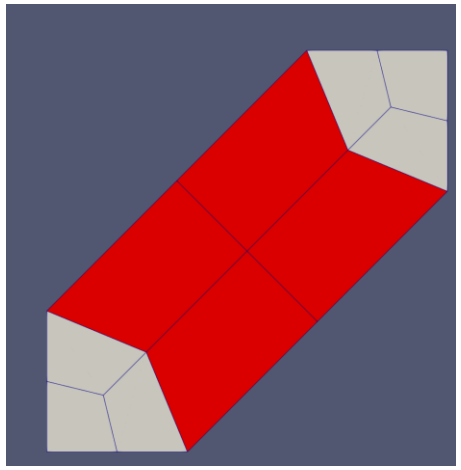
MSRE mesh: Lattice

- MSRE heterogeneous model was created through stitching, transforming, deleting, and filling primitive components
- The MSRE mesh was generated using the combination of multiple MOOSE mesh generator objects, such as
    - ConcentricCircleMeshGenerator / PolygonConcentricCircleMeshGenerator
    - FillBetweenSidesetsGenerator
    - PlaneDeletionGenerator
    - StitchedMeshGenerator
- High flexibility of MOOSE mesh generation allows more than one way to generate the MSRE mesh
    - Alternative method: ParsedCurveGenerator + XYDelaunayGenerator

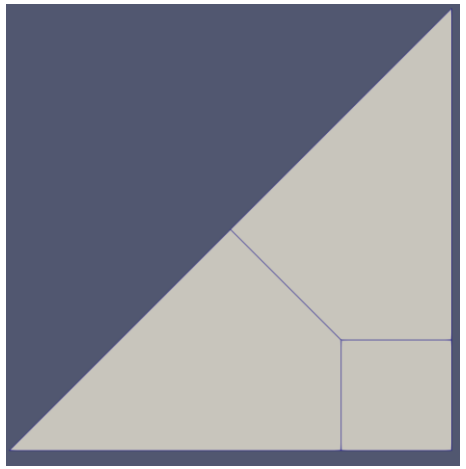
Circle + Depletion



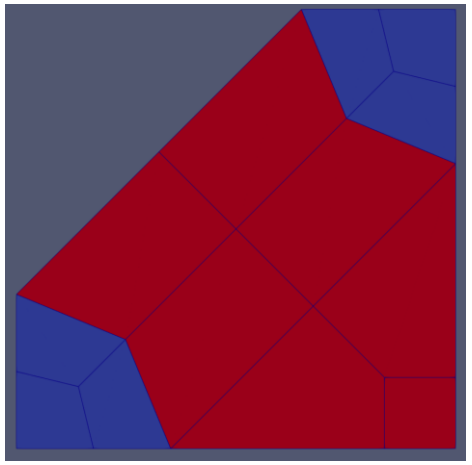
Fill between circles



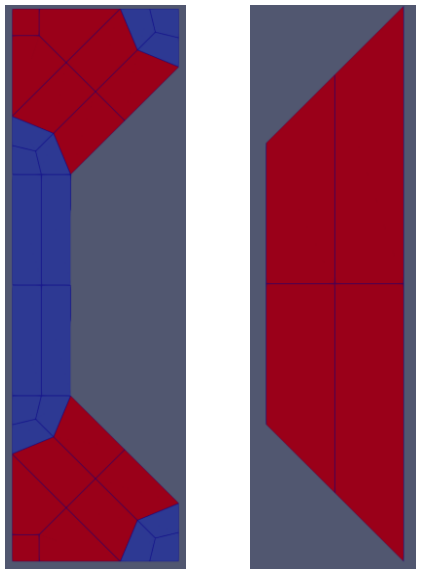
Polygon



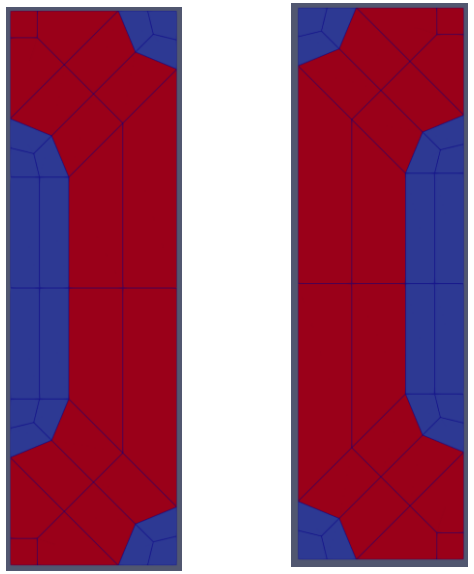
Stitch polygon and circles



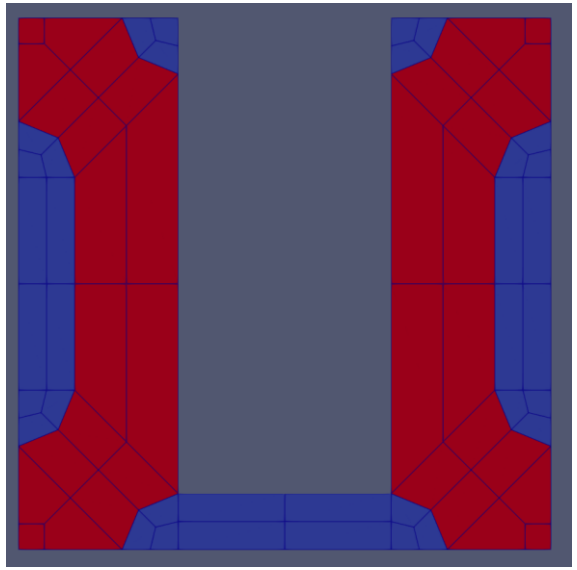
Fill area between two corners



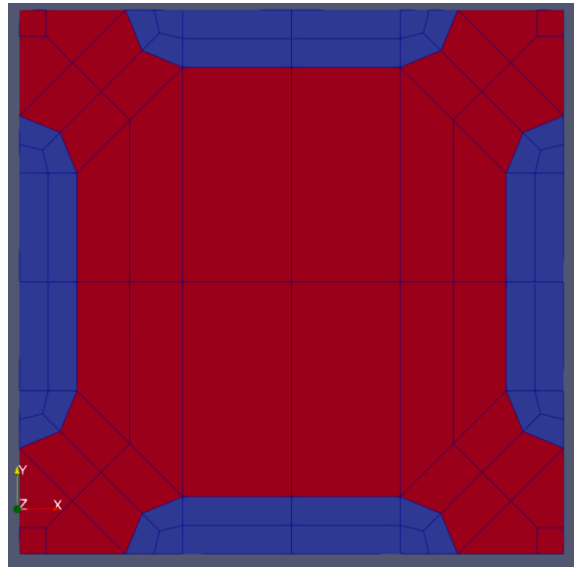
Mirror left side



Fill between left and right sides

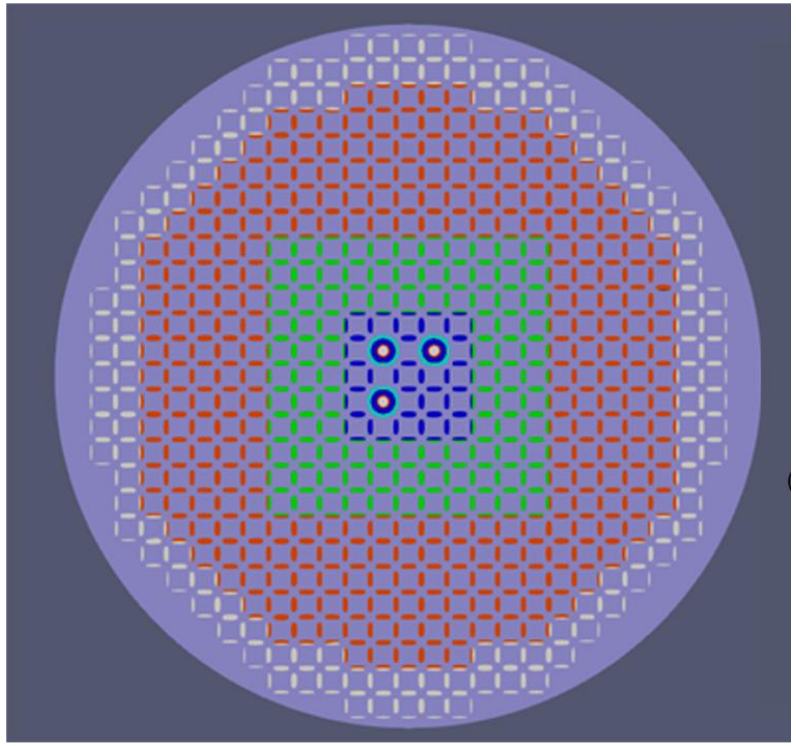


Final fuel lattice



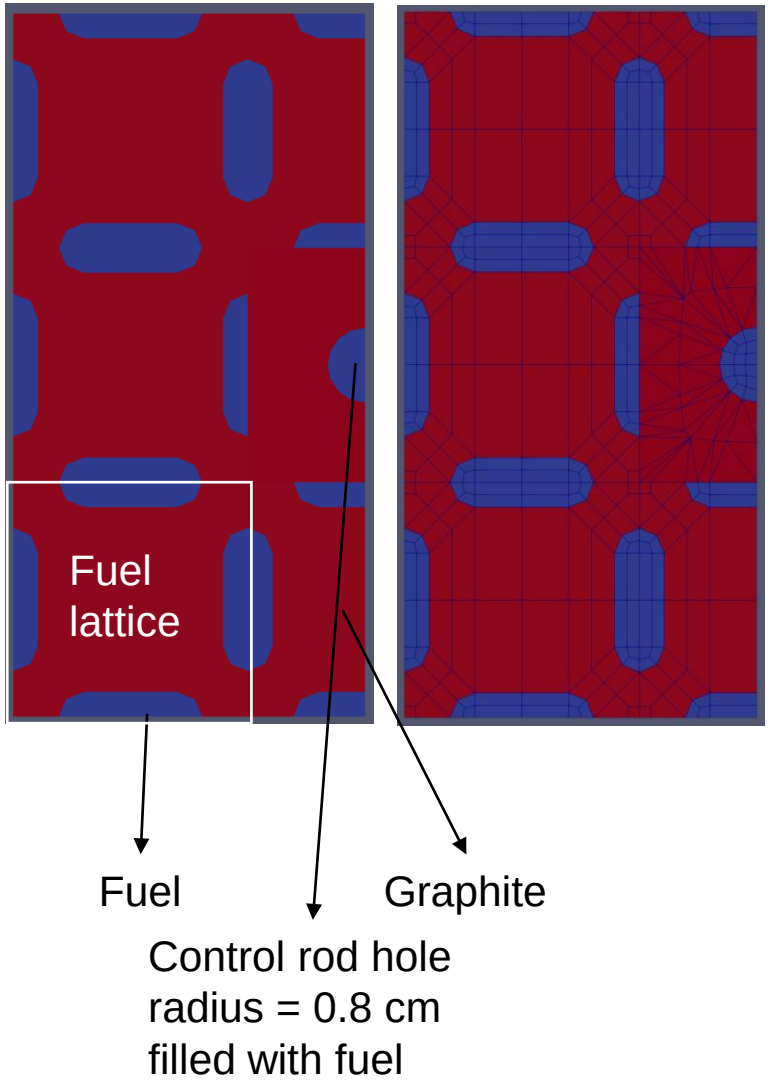
MSRE MESH: Core

- Core mesh was created as follow:
  - PatternedMeshGenerator to generate core lattices
  - Remove the central lattices
  - Create control rod lattice based on its surrounding boundary
  - Stitch control rod lattices together to generate the central control rod region
  - Stitch the central control rod region back to the core lattices.
  - PeripheralRingMeshGenerator to add the outer boundary
  - Extrude and scale.



Practice problem

- Create a model with fuel lattice surrounding a control rod (withdrawn).
- Ideas:
  - Copy the fuel lattice generators
  - Create a PatternedMeshGenerator with fuel lattices surround a dummy lattice
  - Delete the dummy lattice and keep the boundary
  - Create control rod lattice
  - Connect control rod lattice with the fuel lattices.
  - Extrude and scale



MSRE Heterogenous Model

T. FEI, S. K. LEE, K. MO, Y. CAO, C. LEE

Argonne National Laboratory

Griffin calculation

- Information required
  - Cross section
  - Mesh
  - Boundary condition (Vacuum/Reflective)
  - Solver
  - Material assignment

Material Assignment

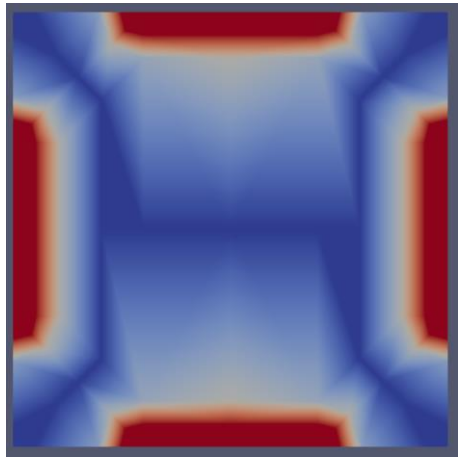
```
[core_fuel_salt]
  type        = MixedNeutronicsMaterial
  block       = '1979 1980 1981 1982 2 3 4 5 6 201 211'
  # block       = '1 2 3 4 5 6 201 211'
  isotopes    = 'pseudo_Be09_A
                 pseudo_F_19_A
                 pseudo_Li06_A
                 pseudo_Li07_A
                 pseudo_U_35_A
                 pseudo_U_38_A
                 pseudo_Zr90_A
                 pseudo_Zr91_A
                 pseudo_Zr92_A
                 pseudo_Zr94_A
                 pseudo_Zr96_A'
  densities   = '9.84823E-03
                 4.94437E-02
                 2.55699E-06
                 2.19199E-02
                 8.38376E-05
                 1.85984E-04
                 8.67636E-04
                 1.89210E-04
                 2.89212E-04
                 2.93091E-04
                 4.72182E-05'
  material_id = 1
[]
```

Griffin calculation

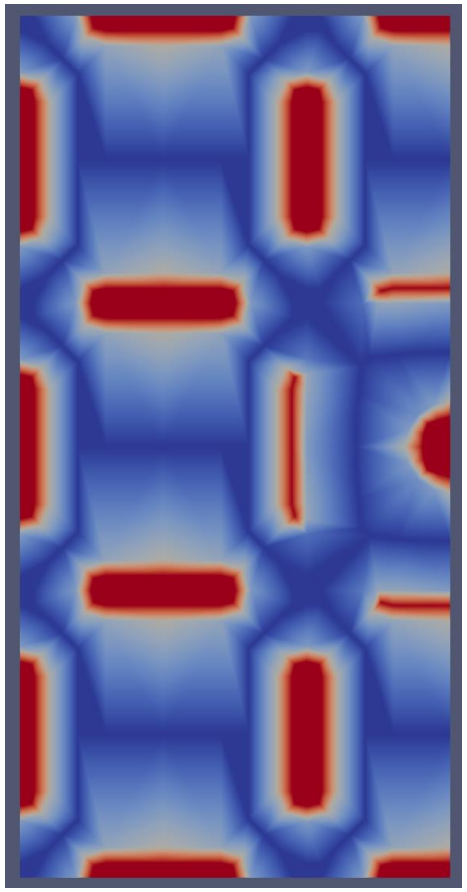
- Diffusion calculation for all cases (using 20-group cross section)
  - Lattice calculation: axial vacuum BC and radial reflective BC
  - Practice problem: reflective BC for both axial and radial boundary
  - Full core: vacuum BC for both axial and radial boundary
  - Difference between OpenMC and Griffin full calculation is -357 pcm.

Code	k-eff	diff [pcm]
OpenMC (full core)	0.97038	-
Griffin (fuel lattice)	1.40948	-
Griffin (practice)	1.60598	-
Griffin (full core)	0.96703	-357

Power distribution

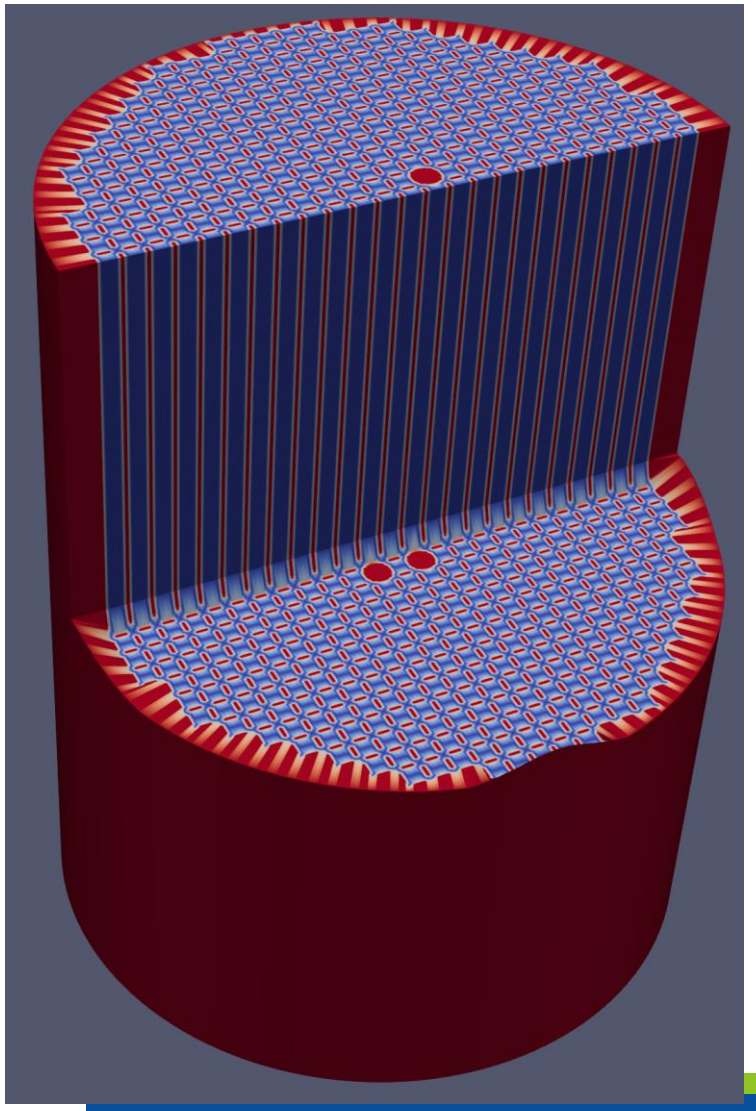


Fuel lattice



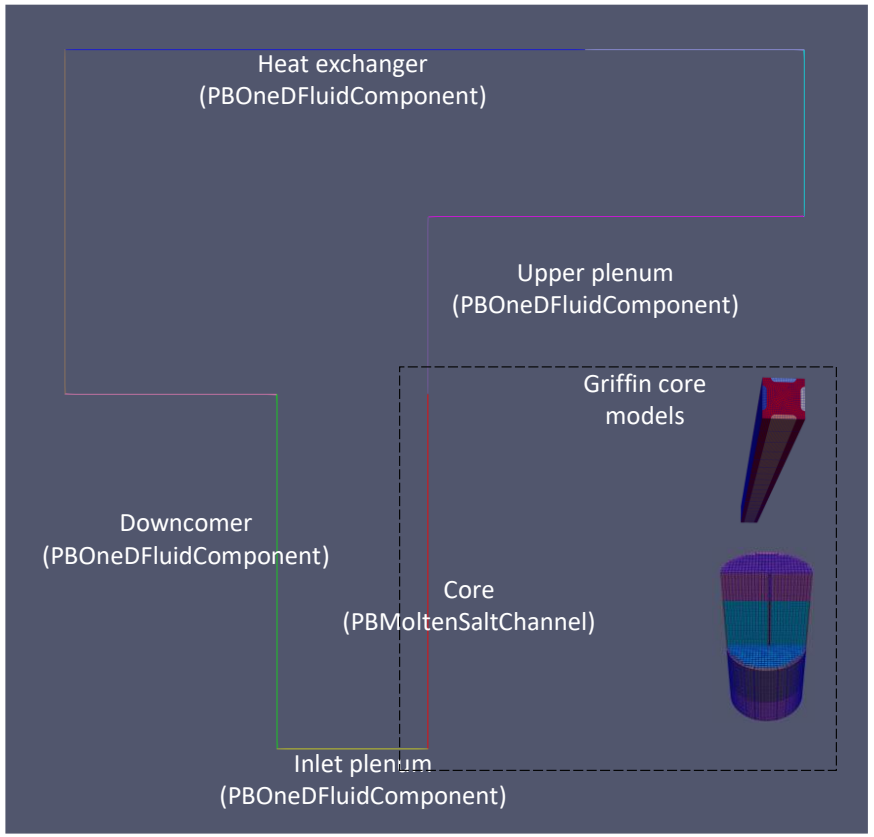
Practice problem

Core



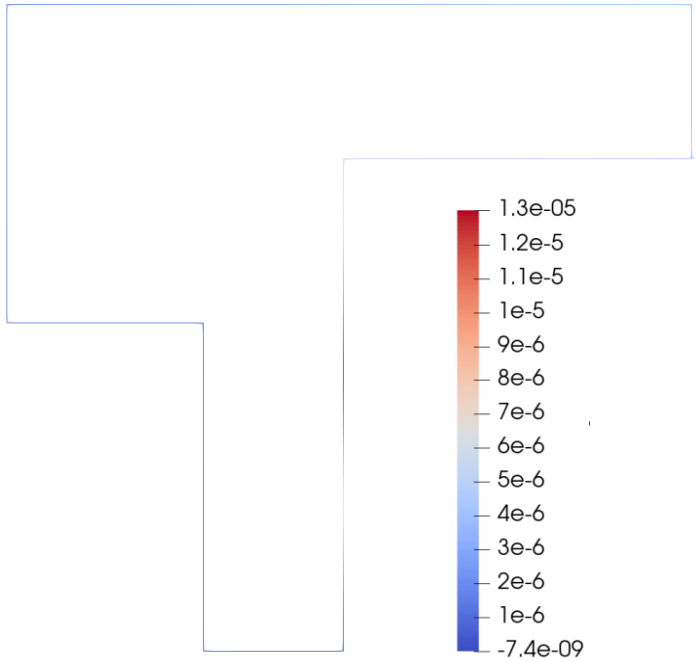
Griffin-SAM coupling (I)

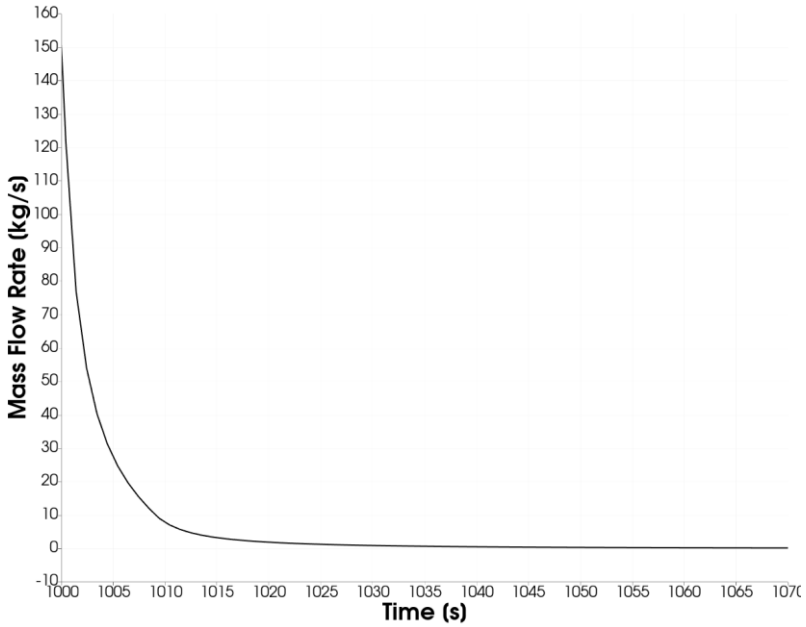
- Griffin-SAM coupled model was developed to simulate the pump startup and coastdown transient.
- MultiApp and Transfer systems are used to transfer information between SAM and Griffin.
  - Nearest node transfer in z-direction
- Precursor concentration from the core region (including core, upper and lower plenum) is transferred from SAM to Griffin.
- k-eff and power profile are transferred from Griffin to SAM.

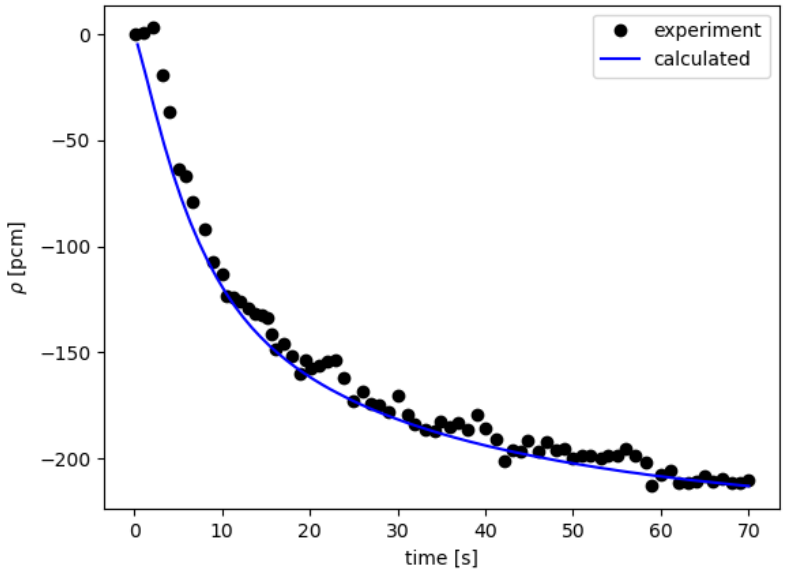


Griffin-SAM coupling (II)

- SAM is the main app that drives Griffin eigenvalue calculation at each time step.
  - Single-/Multi-Channel approach in SAM







MSRE Simple Meshing

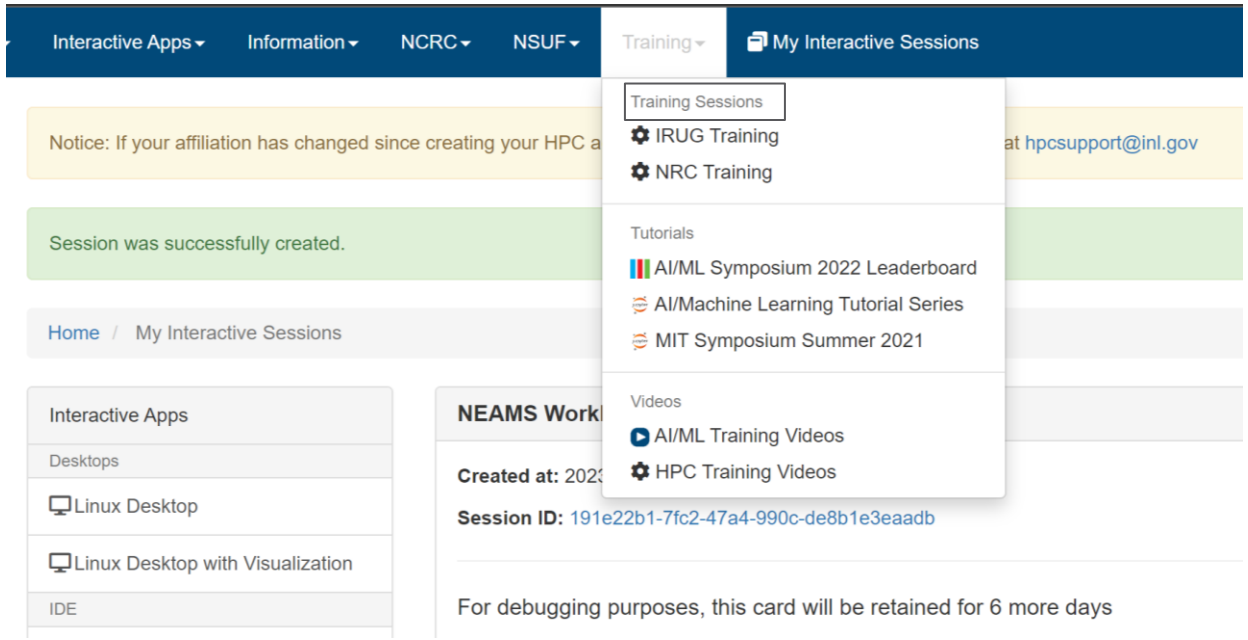
Mauricio E. Tano, Ph.D.

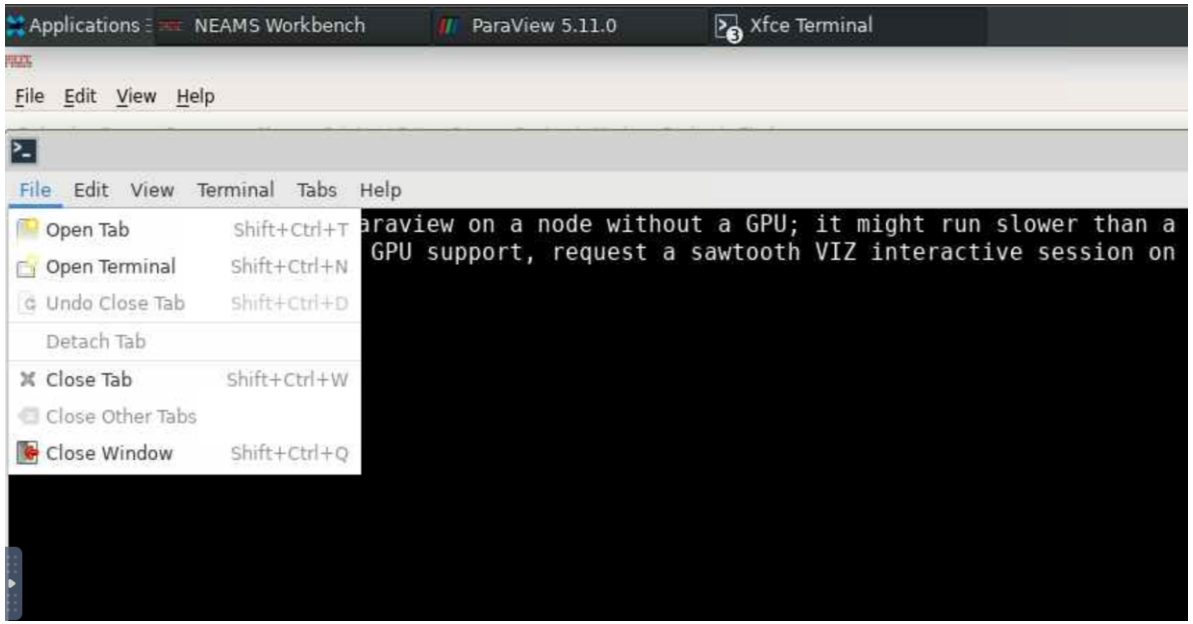
4/21/2024

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

Idaho National Laboratory

HPC-OnDemand & General Setup

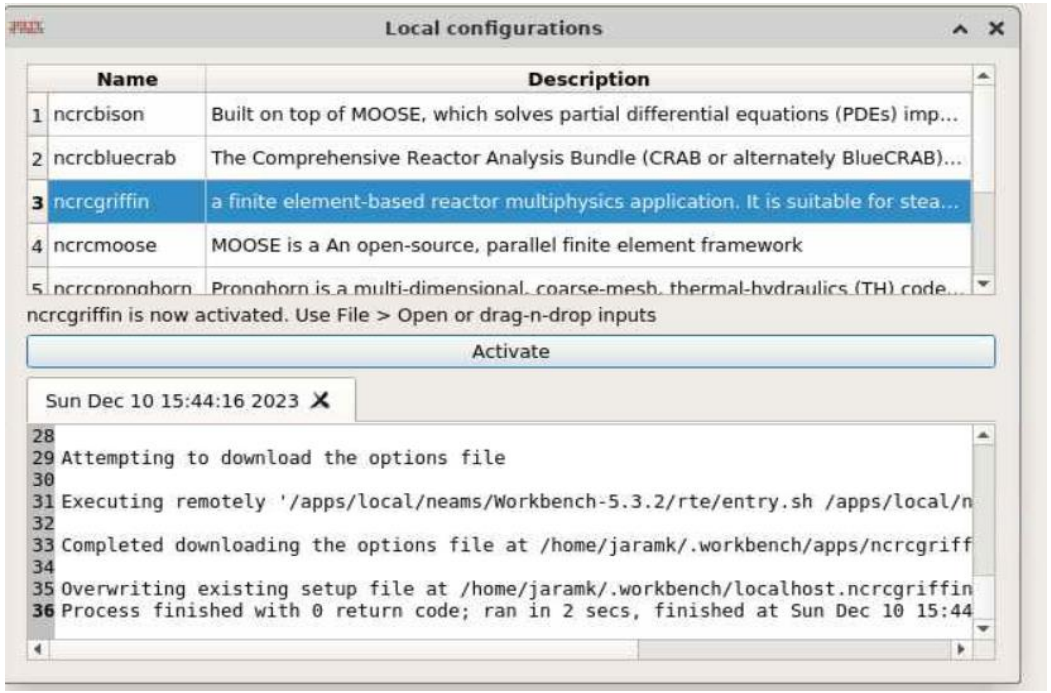


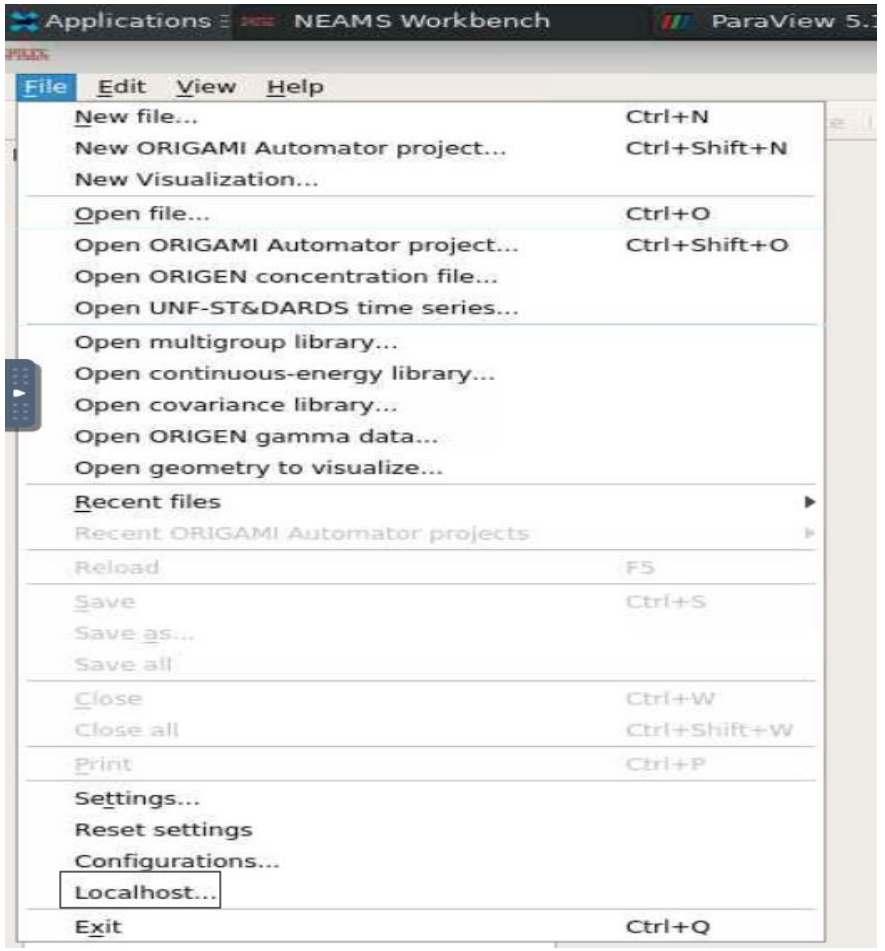


- Open new terminal:
  - Copy the workshop input/output folder
    - cp -r /projects/physor_molten_salt_training_2024/ .
  - **If not loaded you need to** Load the following modules
    - module load paraview
    - module load neams-workbench && Workbench

HPC-OnDemand & General Setup

- On workbench tab:
  - File/Localhost
  - Select ncrcgriffin and click Activate
  - Select ncrcpronghorn and click Activate
  - File/ Open file/ NRC_WS_121123





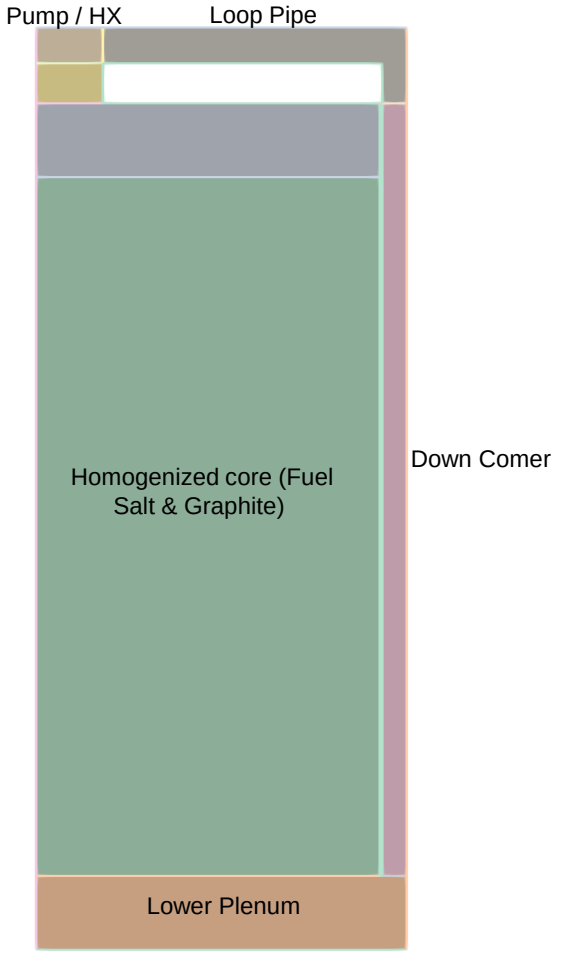
MSRE Mesh / RZ

MSRE Mesh

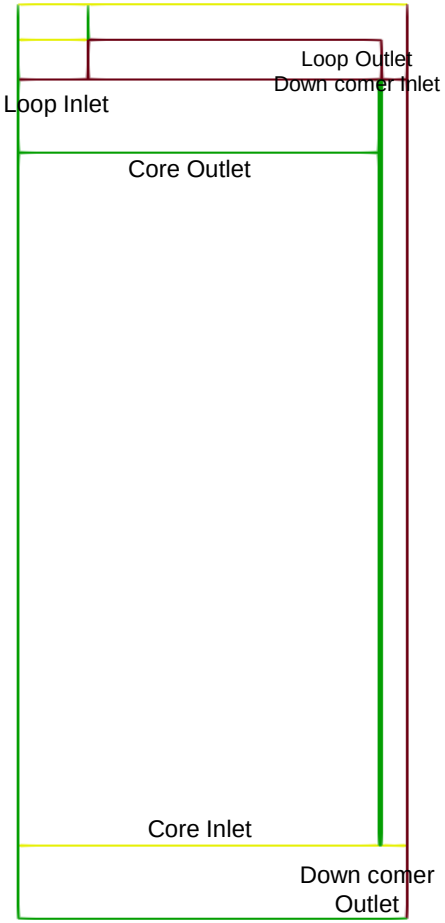
- Files:
  - Input: mesh_msre.i
  - Output: mesh_msre_in.e

```
[Mesh]
  coord_type     = 'RZ'
  type           = MeshGeneratorMesh
  block_id       = '1 2 3 4 6 7 8 9'
  block_name     = 'core  lower_plenum  upper_plenum  down_comer  core_barrel riser pump elbow'
  uniform_refine = 1

  [cartesian_mesh]
    type         = CartesianMeshGenerator
    dim          = 2
    dx           = '${fuel_pipe_R}  ${core_internal_R}  ${core_outer_gap}  ${core_barrel_thickne
    ix           = '4                8                   2                  1
    dy           = '0.1715   0.100   0.100   0.246   0.246   0.246
                    0.246   0.246   0.100   0.100   0.1715 ${piping_height} ${height_pump}'
    iy           = '6  4  4  10  10  10  10  10  4  4  6  4 4'
    subdomain_id = ' 2  2  2  2  2
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     3  3  3  6  4
                     7  10 10 10 9
                     8  9  9  9  9'
  []
```



Defined blocks



Defined Sidsets

MSRE Mesh

```
[Mesh]
  coord_type     = 'RZ'
  type           = MeshGeneratorMesh
  block_id       = '1 2 3 4 6 7 8 9'
  block_name     = 'core  lower_plenum  upper_plenum  down_comer  core_barrel riser pump elbow'
  uniform_refine = 1

  [cartesian_mesh]
    type         = CartesianMeshGenerator
    dim          = 2
    dx           = '${fuel_pipe_R}  ${core_internal_R}  ${core_outer_gap}  ${core_barrel_thickne
    ix           = '4               8                   2                 1
    dy           = '0.1715   0.100   0.100   0.246   0.246   0.246
                    0.246   0.246   0.100   0.100   0.1715 ${piping_height} ${height_pump}'
    iy           = '6  4  4  10  10  10  10  10  4  4  6  4 4'
    subdomain_id = ' 2  2  2  2  2
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     1  1  1  6  4
                     3  3  3  6  4
                     7  10 10 10 9
                     8  9  9  9  9'
  []
```

```
  [loop_boundary]
    type           = SideSetsBetweenSubdomainsGenerator
    primary_block  = '9 7 3'
    paired_block   = '10'
    input          = cartesian_mesh
    new_boundary   = loop_boundary
  []
  [core_in]
    type           = SideSetsBetweenSubdomainsGenerator
    primary_block  = '1'
    paired_block   = '2'
    input          = loop_boundary
    new_boundary   = core_in
  []
  [core_out]
    type           = SideSetsBetweenSubdomainsGenerator
    primary_block  = '1'
    paired_block   = '3'
    input          = core_in
    new_boundary   = core_out
  []
  [block_delete]
    type           = BlockDeletionGenerator
    input          = core_out
    block          = '10'
  []
```

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.

MOOSE-based Thermal-Hydraulics Tools

Description, Status, and Needs

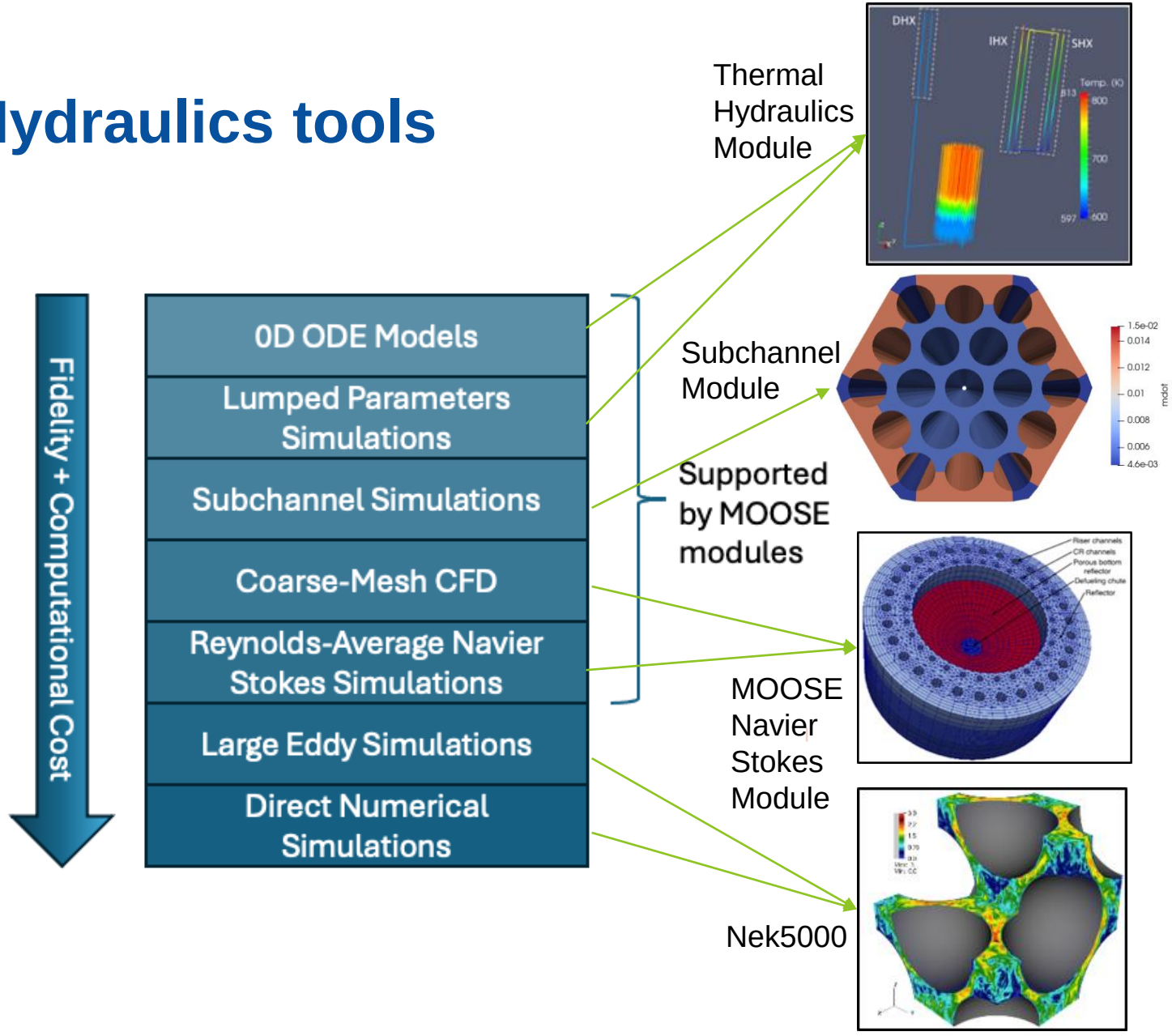
Mauricio E. Tano, Ph.D.

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

INL Idaho National Laboratory

Moose-Based Thermal-Hydraulics tools

- MOOSE provides versatile, general-purpose thermal-hydraulics applications.
- These applications solve for:
  - mass, momentum, and energy conservation
  - in multicomponent, multiphase flows
  - using incompressible, weakly-compressible, or fully compressible formulations
  - for steady-state or transients
  - in lumped parameters and/or multidimensional (1, 2, or full 3D) geometries.

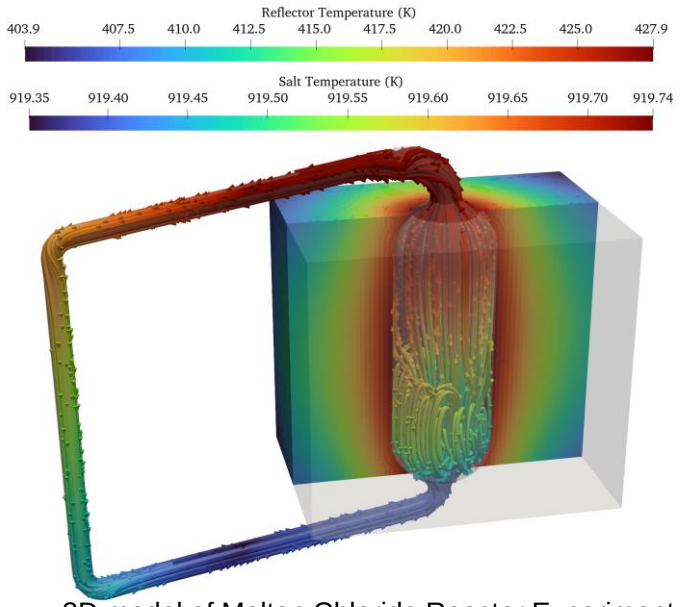


Key MOOSE-based Thermal-Hydraulics Tools

Module	Scale	Flow-Formulation	Dimension	Typical Element Count	Typical Runtime	Typical Simulations
Navier-Stokes Module / Pronghorn	Coarse-Mesh CFD Reynolds-Average Navier Stokes (RANS) Simulations	• Incompressible, Weakly-Compressible, or Fully-Compressible • Single- or multi-phase • Single- or multi-component flow	• Typically, 2D, 2D axisymmetric, or 3D • Can also be used in 1D	10,000	1 minute	• Flow through nuclear reactor core or plena • 3D multi-phase flow in pipes • Natural convection flow in open cavities
Subchannel Module	Subchannel Scale	• Incompressible or Weakly-Compressible • Single-phase • Single- or multi-component flow	• Typically, 3D • Can be used in 1D and 2D	100,000	10 seconds	• Flow development through nuclear reactor fuel assembly • Thermal-hydraulics analysis of nuclear reactor assembly blockage • Natural convection cooling in nuclear reactors low-flow assemblies
Thermal-Hydraulics Module	Lumped-Parameters Simulations	• Compressible • Single-phase	1D, 0D	100	10 seconds	• Heat extraction unit from nuclear reactor core • Thermal loops with significant compressibility effects

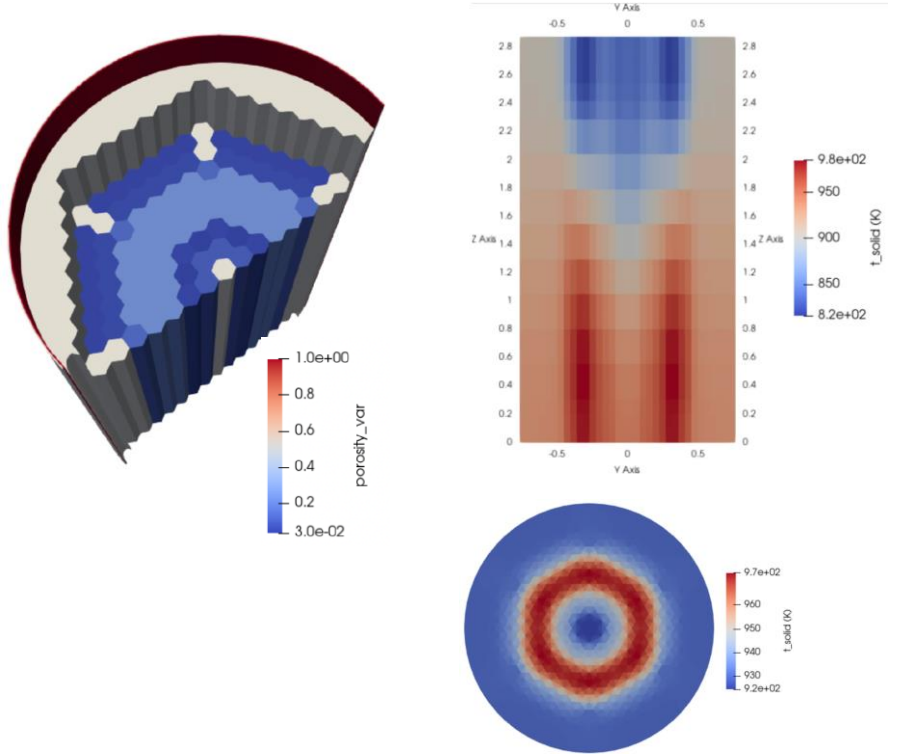
Pronghorn + MOOSE NS Module

- Numerical solvers:
  - Stabilized Finite Elements: 2017 –
  - Finite Volumes: 2021 –
- Fluid types:
  - Incompressible
  - Weakly-compressible
  - Compressible
- Flow regimes:
  - Laminar
  - Turbulent
- Flow types:
  - Free-flow
  - Porous media flow
  - Two-phase flow
- Validation:
  - Hundreds of regression test on canonical flows
  - 17 completed ERCOFTAC cases for diverse flow conditions



3D model of Molten Chloride Reactor Experiment
P. German; M. Tano
Model: FV, WCNS, Turbulent, Free-flow

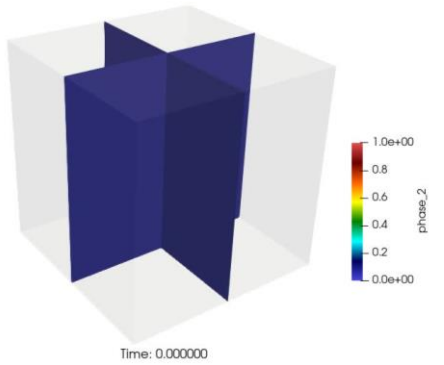
Melt Pool Simulation
A. Lindsay
Model: ALE, INS, Laminar, Free-flow

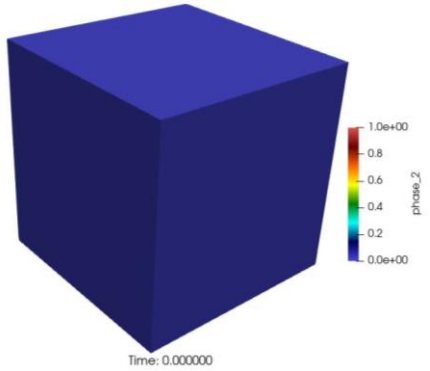


3D Model of High-Temperature Test Facility
V. Kyriakopoulos, M. Tano, P. Balestra, S. Schunert
Model: FV, WCNS, Laminar, Porous

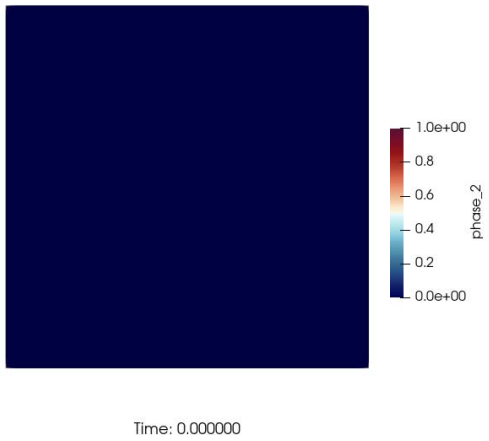
Contact Angle Equilibration in Molten Salt – Gas permeation
M. Tano; V. Prithivirajan
Model: DG-FE, INS, Laminar, Two-phase

Examples MOOSE Navier-Stokes Validation Cases

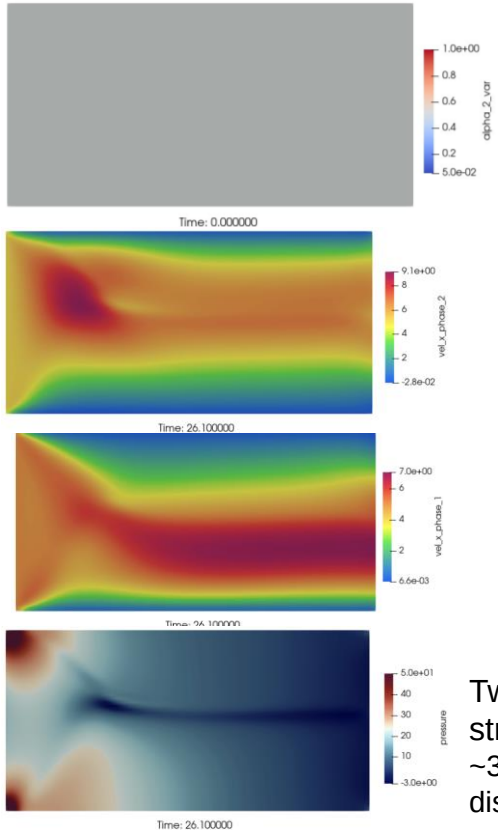




Two-phase Rayleigh-Benard Convection
~1.4% error in void fraction distribution vs experiments

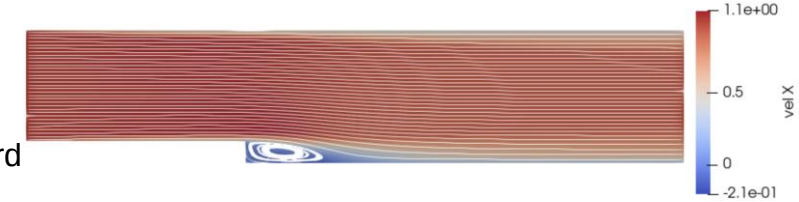


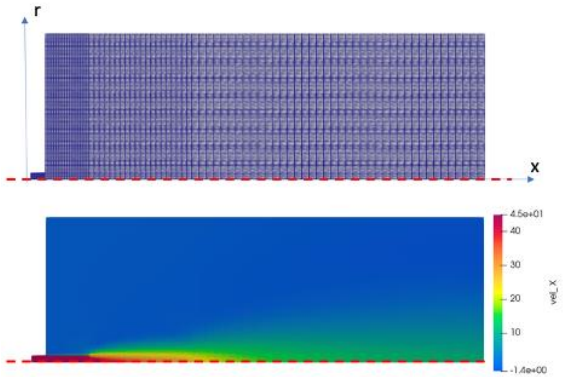
Two-phase Kelvin-Helmholtz Convection
~2.3% error in void fraction distribution vs experiments

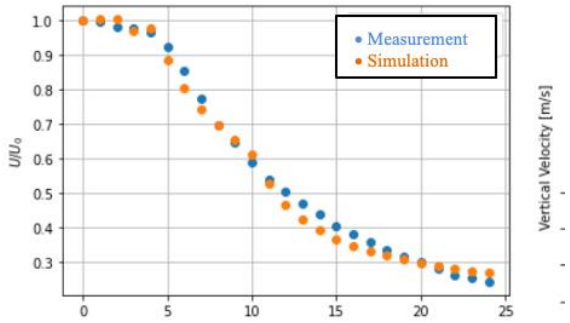


Two-phase flow stratification in channel
~3.2% error in velocity distribution vs experiments

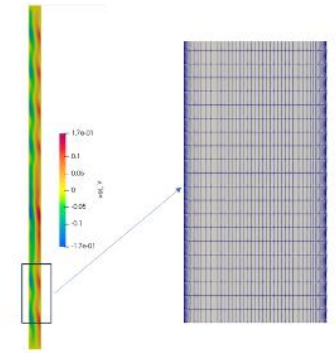
Single-phase backward facing step test
~4.1% error in velocity and pressure distribution vs experiments

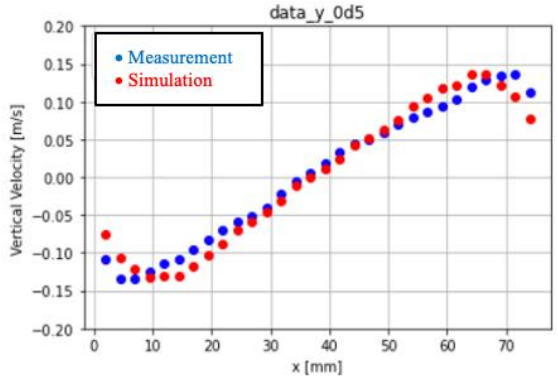






Air jet expansion in a a channel
~4.1% error in velocity distribution vs experiments

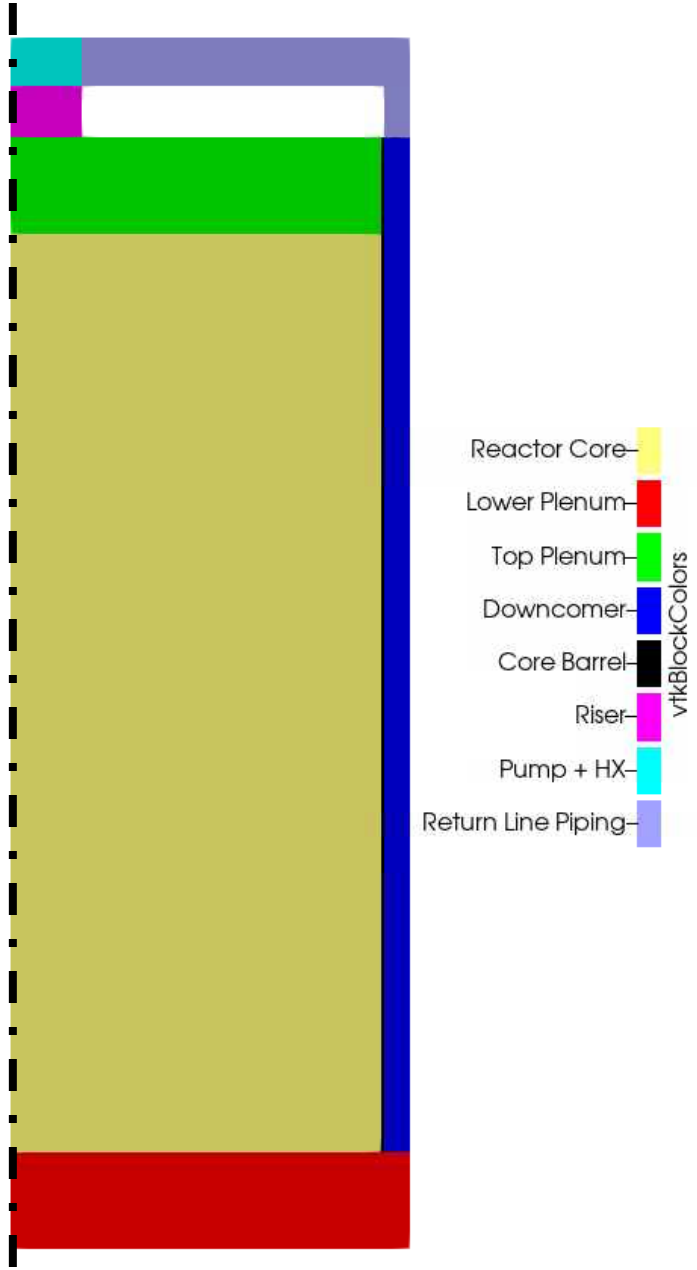




High-Rayleigh natural convection in tall cavity
~5.2% error in temperature velocity distribution vs experiments

Key Model Features

- Axisymmetric model
- Following the circulation of the molten salt, the model includes:
  1. Reactor Core
  2. Top Plenum
  3. Riser
  4. Pump + HX
  5. Return Line Piping
  6. Downcomer
  7. Bottom plenum
- The solid core barrel is included in the model.
  - Conjugated heat transfer is implemented between the reactor core and downcomer with the core barrel



Thermal-Hydraulics Formulation in Pronghorn

- General porous media model in Pronghorn:

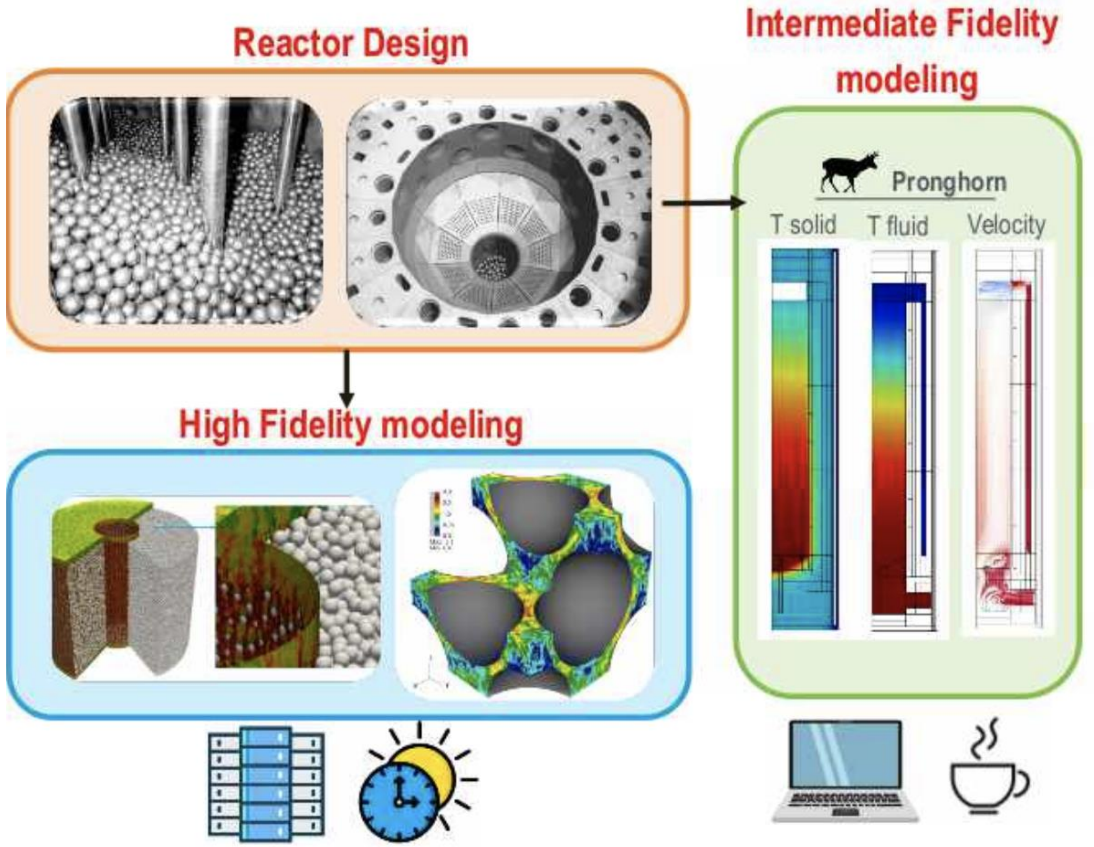
$$\gamma \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\gamma^{-1} \rho \boldsymbol{u}\boldsymbol{u}) = -\gamma \nabla p + \gamma \rho \boldsymbol{g} + \nabla \cdot \left(\mu (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)\right) - \rho W$$

$$\gamma \frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho h \boldsymbol{u}) = \nabla \cdot \left(\kappa_{eff} \nabla T\right) - \alpha_{ls}(T - Ts) + q'''_f$$

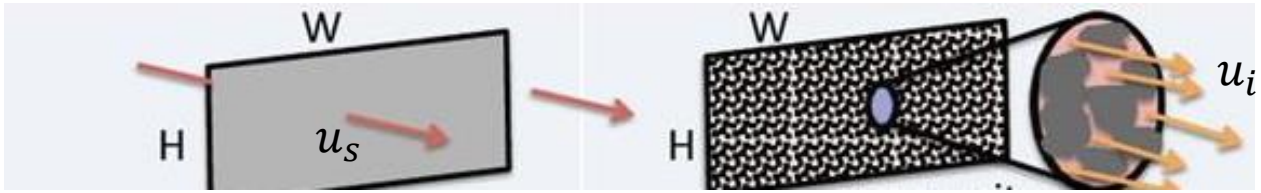
- Standard notation; γ is the porosity and W is the porous friction term, e.g., the Darcy-Forchheimer coefficient given by:

$$W \leftarrow \frac{150\nu}{D^2} \frac{(1-\gamma)^2}{\gamma^2} \boldsymbol{u} + \frac{1.75}{D} \frac{1-\gamma}{\gamma} |\boldsymbol{u}| \boldsymbol{u}$$



Solution Variables

- Solving for:
  - Superficial Velocity (linear momentum conservation)
  - Pressure (mass conservation)
  - Fluid Temperature (energy conservation in the fluid)
  - Solid Temperature (energy conservation in the solid)
- Note that we solve for superficial vs. interstitial velocity, i.e., $u_s = u_i \gamma$




```
[Variables]
  [superficial_vel_x]
    type              = PINSFVSuperficialVelocityVariable
    initial_condition = 1e-8
    block             = ${fluid_blocks}
  []
  [superficial_vel_y]
    type              = PINSFVSuperficialVelocityVariable
    initial_condition = 1e-8
    block             = ${fluid_blocks}
  []
  [pressure]
    type              = INSFVPressureVariable
    initial_condition = ${p_outlet}
    block             = ${fluid_blocks}
  []
  [T_fluid]
    type              = INSFVEnergyVariable
    initial_condition = ${T_Salt_initial}
    block             = ${fluid_blocks}
  []
  [T_solid]
    type              = INSFVEnergyVariable
    initial_condition = ${T_Salt_initial}
    block             = ${solid_blocks}
  []
[]
```

Fluid Properties

- Using coded thermophysical properties for MSRE salt
- A significant number of common nuclear coolants are supported in Pronghorn and the open-source MOOSE-NS module

```
[FluidProperties]
  [fluid_properties_obj]
    type                = SimpleFluidProperties
    density0            = 2705.8554    # kg/m^3
    thermal_expansion   = 0.000177319 # K^{-1}
    cp                  = 1868.0       # J/kg·K
    viscosity           = 0.008268     # Pa-s11
    thermal_conductivity = 1.4         # W/m·K
  []
[]
```

FluidProperties

Fluid Properties App	
AddFluidPropertiesAction	Add a UserObject object to the simulation.
BrineFluidProperties	Fluid properties for brine
CO2FluidProperties	Fluid properties for carbon dioxide (CO2) using the Span & Wagner EOS
CaloricallyImperfectGas	Fluid properties for an ideal gas with imperfect caloric behavior.
FlibeFluidProperties	Fluid properties for flibe
FlinakFluidProperties	Fluid properties for flinak
HeliumFluidProperties	Fluid properties for helium
HydrogenFluidProperties	Fluid properties for Hydrogen (H2)
IdealGasFluidProperties	Fluid properties for an ideal gas
IdealRealGasMixtureFluidProperties	Class for fluid properties of an arbitrary vapor mixture
LeadBismuthFluidProperties	Fluid properties for Lead Bismuth eutectic 2LiF-BeF2
LeadFluidProperties	Fluid properties for Lead
MethaneFluidProperties	Fluid properties for methane (CH4)
NaClFluidProperties	Fluid properties for NaCl
NaKFluidProperties	Fluid properties for NaK
NitrogenFluidProperties	Fluid properties for Nitrogen (N2)
SalineMoltenSaltFluidProperties	Molten salt fluid properties using Saline
SimpleFluidProperties	Fluid properties for a simple fluid with a constant bulk density

From: https://mooseframework.inl.gov/syntax/index.html

Fuel Salt Thermal-Hydraulics Solution 1/2

```
[NavierStokesFV]
  # Basic settings - weakly-compressible, turbulent flow with buoyancy
  block                           = ${fluid_blocks}
  compressibility                 = 'weakly-compressible'
  porous_medium_treatment         = true
  add_energy_equation             = true
  gravity                         = '0.0 -9.81 0.0'

  # Variable naming
  velocity_variable               = 'superficial_vel_x superficial_vel_y'
  pressure_variable               = 'pressure'
  fluid_temperature_variable      = 'T_fluid'

  # Numerical schemes
  pressure_face_interpolation     = average
  momentum_advection_interpolation = upwind
  mass_advection_interpolation    = upwind
  energy_advection_interpolation  = upwind
  velocity_interpolation          = rc

  # Porous & Friction treatement
  use_friction_correction         = true
  friction_types                  = 'darcy forchheimer'
  friction_coeffs                 = 'Darcy_coefficient Forchheimer_coefficient'
  consistent_scaling              = 100.0
  porosity_smoothing_layers       = 2
  turbulence_handling             = 'mixing-length'
```

Flow formulation

Solve Variables

Numerical Discretization

Porous Media Treatment for Reactor Core
The friction coefficients are defined as materials

Fuel Salt Thermal-Hydraulics Solution 2/2

```
# fluid properties
density                 = 'rho'
dynamic_viscosity       = 'mu'
thermal_conductivity    = 'kappa'
specific_heat           = 'cp'

# Energy source-sink
external_heat_source    = 'power_density_fuel'

# Boundary Conditions
wall_boundaries         = 'left      top      bottom   right    loop_boundary '
momentum_wall_types     = 'symmetry  slip     noslip   noslip   noslip'
energy_wall_types       = 'heatflux  heatflux heatflux heatflux heatflux'
energy_wall_function    = '0         0        0        0        0'

# Constrain Pressure
pin_pressure            = true
pinned_pressure_value   = ${p_outlet}
pinned_pressure_point   = '0.0 2.13859 0.0'
pinned_pressure_type    = point-value-uo

# Passive Scalar -- solved separetely to integrate porosity jumps
add_scalar_equation     = false

#Scaling -- used mainly for nonlinear solves
momentum_scaling        = 1e-3
mass_scaling            = 10
```

Thermophysical properties
Defined by materials using the fluid_properties_obj

External heat source
Defined by auxiliary variables

Boundary conditions

Pressure pin
Needed to uniquely define the pressure in a closed-loop system

Scaling parameters
Needed to improve conditioning of the Jacobian in nonlinear solves

Extra Kernels for the solid domain

```
[energy_storage]
  type                  = PINSFVEnergyTimeDerivative
  variable              = T_solid
  rho                   = rho_s
  cp                    = cp_s
  is_solid              = true
[]
[solid_energy_diffusion_core]
  type                  = PINSFVEnergyAnisotropicDiffusion
  variable              = T_solid
  kappa                 = 'effective_thermal_conductivity'
  effective_diffusivity = true
  porosity              = 1
[]
[heat_source]
  type                  = FVCoupledForce
  variable              = T_solid
  v                     = power_density_graph
  block                 = 'core'
[]
```

Time Derivative of specific Solid Enthalpy (defining $h = \rho c_p T$)

Specific Solid Enthalpy Diffusion

External heart source
Source is defined as Auxiliary Variables

Extra kernels for pump + HX modeling

```
[pump_x]
  type                = INSFVBodyForce
  variable            = superficial_vel_x
  functor             = ${pump_force}
  block               = 'pump'
  momentum_component  = 'x'
  rhie_chow_user_object = 'pins_rhie_chow_interpolator'
[]
[pump_y]
  type                = INSFVBodyForce
  variable            = superficial_vel_y
  functor             = ${pump_force}
  block               = 'pump'
  momentum_component  = 'y'
  rhie_chow_user_object = 'pins_rhie_chow_interpolator'
[]
[convection_fluid_hx]
  type                = NSFVEnergyAmbientConvection
  variable            = T_fluid
  T_ambient           = ${T_inlet_hx}
  alpha               = ${vol_hx}
  block               = 'pump'
[]
```

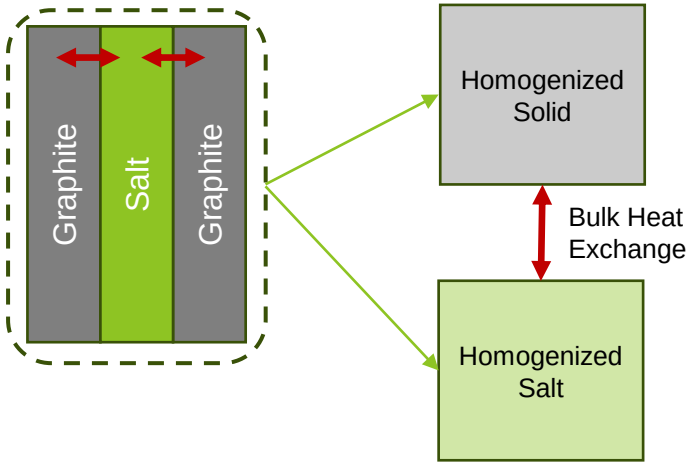
Volumetric force ($\vec{F} = [F_x, F_y]$) for the pump in the ‘x’ and ‘y’ direction

Heat exchange with external temperature ($q''' = \alpha(T - T_{ambient})$)

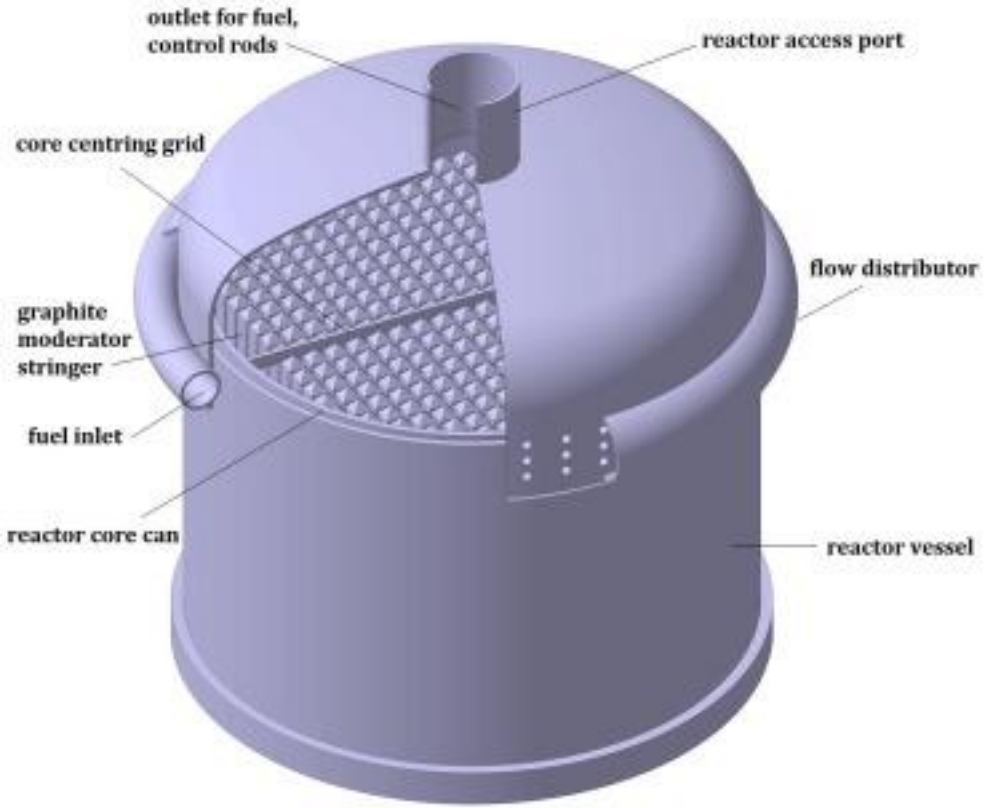
```
T_inlet_hx          = 904.55            # Salt inlet temperature (K)
```

Liquid/Solid Heat exchanges in the MSRE

- The MSRE core is modeled as a porous media, where the liquid and solid domains are homogenized
  - Bulk heat exchange occurs between the liquid and solid phases



- Conjugated heat transfer occurs between the downcomer, after the flow distributor, and the reactor core through the reactor core can or core barrel



From: Tadepalli, S. C., Gupta, A., & Umasankari, K. (2017). Neutronic analysis of MSRE and its study for validation of ARCH code. *Nuclear Engineering and Design*, *320*, 1-8.

Modeling bulk heat transfer at reactor core

```
[convection_core]
  type                = PINSFVEnergyAmbientConvection
  variable            = T_solid
  T_fluid             = T_fluid
  T_solid             = T_solid
  is_solid            = true
  h_solid_fluid       = ${bulk_htc}
  block               = 'core'
[]
[convection_core_completmeent]
  type                = PINSFVEnergyAmbientConvection
  variable            = T_fluid
  T_fluid             = T_fluid
  T_solid             = T_solid
  is_solid            = false
  h_solid_fluid       = ${bulk_htc}
  block               = 'core'
[]
```

Bulk heat transfer **from liquid** salt **to solid** core

Bulk heat transfer **from solid** core **to liquid** salt

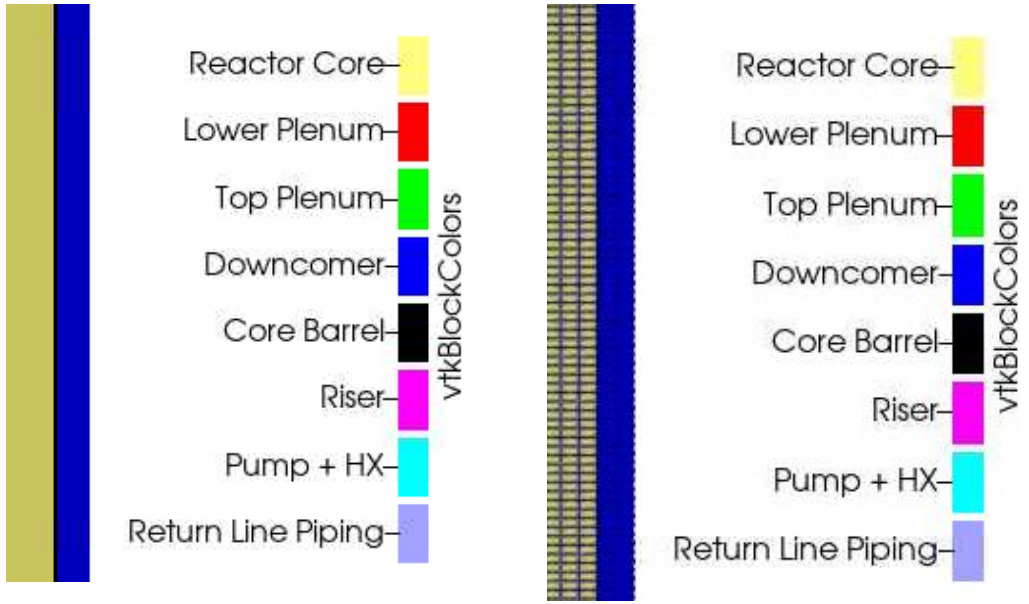
```
bulk_htc            = 20000.0          # (W/(m3.K)) core bulk volumetric heat exchange coefficient (already callibrated)
```

Modeling conjugated heat transfer trough core barrel

```
[FVInterfaceKernels]
  # Conjugated heat transfer with core barrel
  [convection]
    type                = FVConvectionCorrelationInterface
    variable1           = T_fluid
    variable2           = T_solid
    boundary            = 'core_barrel'
    h                   = ${bulk_htc}
    T_solid             = T_solid
    T_fluid             = T_fluid
    subdomain1          = 'core down_comer lower_plenum upper_plenum'
    subdomain2          = 'core_barrel'
    wall_cell_is_bulk   = true
  []
[]
```

Conjugated heat transfer through reactor core barrel

Exchange occurs between the liquid salt temperature, defined on the the left between the core + lower_plenum + upper_plenum and on the right in the down_comer, and the solid temperature defined on the core barrel

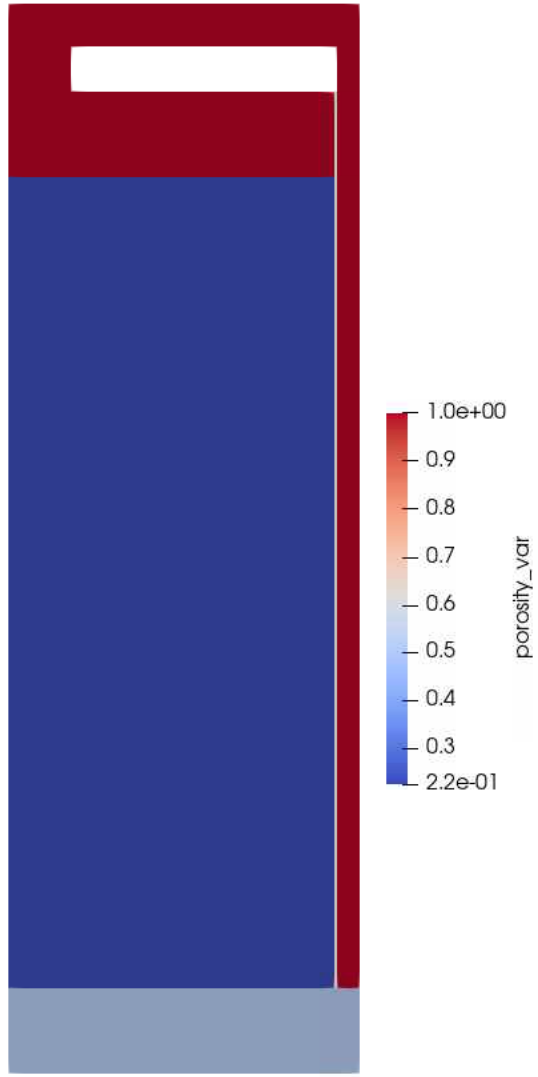


Defining materials

```
# ------------------------------------------------------------------
# Setting up material porosities at fluid blocks
# ------------------------------------------------------------------
[porosity]
  type                  = ADPiecewiseByBlockFunctorMaterial
  prop_name             = 'porosity'
  subdomain_to_prop_value = 'core             ${core_porosity}
                             lower_plenum     ${lower_plenum_porosity}
                             upper_plenum     ${upper_plenum_porosity}
                             down_comer       ${down_comer_porosity}
                             riser            ${riser_porosity}
                             pump             ${pump_porosity}
                             elbow            ${elbow_porosity}
                             core_barrel      0'
[]
# ------------------------------------------------------------------
# Setting up hydraulic diameters at fluid blocks
# ------------------------------------------------------------------
[hydraulic_diameter]
  type                  = PiecewiseByBlockFunctorMaterial
  prop_name             = 'characteristic_length'
  subdomain_to_prop_value = 'core             ${D_H_fuel_channel}
                             lower_plenum     ${D_H_plena}
                             upper_plenum     ${D_H_plena}
                             down_comer       ${D_H_downcomer}
                             riser            ${D_H_pipe}
                             pump             ${D_H_pipe}
                             elbow            ${D_H_pipe}'
  block                 = ${fluid_blocks}
[]
```

Setting up **porosity** for each block in the model

Setting up the **hydraulic diameter** for each block in the model



Setting up materials for fluid and solid properties

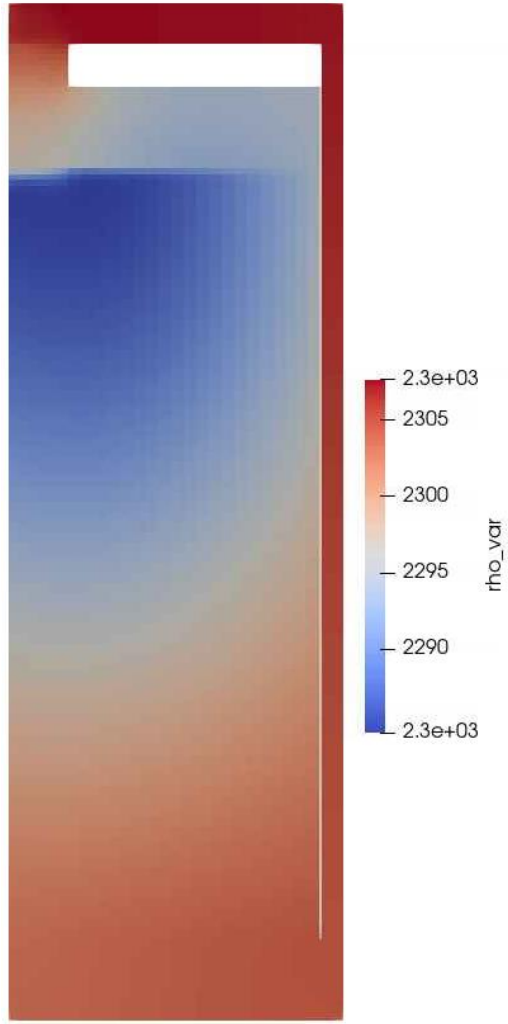
```
# ------------------------------------------------------------------
# Setting up Fluid & Solid properties
# ------------------------------------------------------------------
[fluid_props_to_mat_props]
  type                  = GeneralFunctorFluidProps
  pressure              = 'pressure'
  T_fluid               = 'T_fluid'
  speed                 = 'speed'
  characteristic_length = characteristic_length
  block                 = ${fluid_blocks}
[]
[core_moderator]
  type                  = ADGenericFunctorMaterial
  prop_names            = 'rho_s   cp_s   k_s'
  prop_values           = '${rho_graph} ${cp_graph} ${k_graph}'
  block                 = 'core'
[]
[core_barrel_steel]
  type                  = ADGenericFunctorMaterial
  prop_names            = 'rho_s   cp_s   k_s'
  prop_values           = '${rho_steel} ${cp_steel} ${k_steel}'
  block                 = 'core_barrel'
[]
[effective_fluid_thermal_conductivity]
  type                  = ADGenericVectorFunctorMaterial
  prop_names            = 'kappa'
  prop_values           = 'k k k'
  block                 = ${fluid_blocks}
[]
[effective_solid_thermal_conductivity]
  type                  = ADGenericVectorFunctorMaterial
  prop_names            = 'effective_thermal_conductivity'
  prop_values           = 'k_s k_s k_s'
  block                 =  ${solid_blocks}
[]
```

Setting up properties for the fuel salt
Note: this properties are taken from fluid_properties_obj and define temperature-dependent thermophysical properties

Setting up properties for the graphite moderator

Setting up properties for the steel core barrel

Defining effective thermal conductivity for liquid and solid phases in porous media
Note: these properties could be anisotropic



Defining friction coefficients

```
# Drag correlations per block
[isotropic_drag_core]
  type                  = FunctorChurchillDragCoefficients
  multipliers           = '100000 100 100000'
  block                 = 'core'
[]
[drag_lower_plenum]
  type                  = FunctorChurchillDragCoefficients
  multipliers           = '10 1 10'
  block                 = 'upper_plenum'
[]
[drag_upper_plenum]
  type                  = FunctorChurchillDragCoefficients
  multipliers           = '1 1 1'
  block                 = 'lower_plenum'
[]
[drag_downcomer]
  type                  = FunctorChurchillDragCoefficients
  multipliers           = '1 1 1'
  block                 = 'down_comer'
[]
[drag_piping]
  type                  = FunctorChurchillDragCoefficients
  multipliers           = '0 0 0'
  block                 = 'riser pump elbow'
[]
```

Drag coefficients for the core
Note: blocking non-physical flow in the radial direction

Drag coefficients of lower plenum
Note: accounts for lower plenum internal structures

Drag coefficients of upper plenum
Note: single channel

Drag coefficients of downcomer
Note: single channel

Drag coefficients of piping
Note: pressure drop not modeled in detail due to return loop approximation

Churchill Darcy friction factor

$$f/8 = \left[\left(\frac{8}{\mathrm{Re}}\right)^{12} + \frac{1}{(\Theta_1 + \Theta_2)^{1.5}}\right]^{\frac{1}{12}}$$

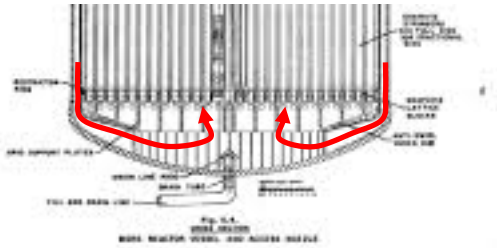
where

$$\Theta_1 = \left[-2.457 \ln\left(\left(\frac{7}{\mathrm{Re}}\right)^{0.9} + 0.27\frac{\varepsilon}{D}\right)\right]^{16}$$

$$\Theta_2 = \left(\frac{37530}{\mathrm{Re}}\right)^{16}$$

From: https://en.wikipedia.org/wiki/Darcy_friction_factor_formulae

MSRE lower plenum



Postprocessors

Multiple postprocessors to analyze MSRE performance

```
[pressure_outlet]
  type                = SideAverageValue
  variable            = pressure
  boundary            = 'pump_inlet'
[]
[pressure_inlet]
  type                = SideAverageValue
  variable            = 'pressure'
  boundary            = 'downcomer_outlet'
[]
[pressure_core_delta]
  type                = ParsedPostprocessor
  function            = 'pressure_inlet - pressure_outlet'
  pp_names            = 'pressure_inlet pressure_outlet'
  execute_on          = 'initial timestep_end'
[]
[T_inlet]
  type                = SideAverageValue
  variable            = 'T_fluid'
  boundary            = 'downcomer_outlet'
[]
[T_outlet]
  type                = SideAverageValue
  variable            = 'T_fluid'
  boundary            = 'riser_inlet'
[]
[T_core_inlet]
  type                = SideAverageValue
  variable            = 'T_fluid'
  boundary            = 'core_in'
[]
[T_core_outlet]
  type                = SideAverageValue
  variable            = 'T_fluid'
  boundary            = 'core_out'
[]
```

```
[v_core_inlet]
  type                = SideAverageValue
  variable            = 'superficial_vel_y'
  boundary            = 'core_in'
[]
[v_core_outlet]
  type                = SideAverageValue
  variable            = 'superficial_vel_y'
  boundary            = 'core_out'
[]
[T_core_delta]
  type                = ParsedPostprocessor
  function            = 'T_core_outlet - T_core_inlet'
  pp_names            = 'T_core_outlet T_core_inlet'
  execute_on          = 'initial timestep_end'
[]
[area_pp_downcomer_inlet]
  type                = AreaPostprocessor
  boundary            = 'downcomer_inlet'
  execute_on          = 'INITIAL'
[]
[vfr_downcomer]
  type                = VolumetricFlowRate
  vel_x               = superficial_vel_x
  vel_y               = superficial_vel_y
  advected_quantity   = 1.0
  boundary            = 'downcomer_inlet'
[]
[vfr_pump]
  type                = VolumetricFlowRate
  vel_x               = superficial_vel_x
  vel_y               = superficial_vel_y
  advected_quantity   = 1.0
  boundary            = 'pump_outlet'
[]
[mfr_core_inlet]
  type                = VolumetricFlowRate
  vel_x               = superficial_vel_x
  vel_y               = superficial_vel_y
  advected_quantity   = rho
  boundary            = 'downcomer_outlet'
[]
```

```
[mfr_core_outlet]
  type                = VolumetricFlowRate
  vel_x               = superficial_vel_x
  vel_y               = superficial_vel_y
  advected_quantity   = rho
  boundary            = 'pump_inlet'
[]
[core_vol]
  type                = VolumePostprocessor
  block               = 'core'
  execute_on          = 'initial timestep_end'
[]
[loop_vol]
  type                = VolumePostprocessor
  block               = ${non_solid_blocks}
  execute_on          = 'initial timestep_end'
[]
[Tmax_fuel]
  type                = ElementExtremeValue
  value_type          = max
  variable            = T_fluid
  block               = ${fluid_blocks}
  execute_on          = 'initial timestep_end'
[]
[Tavg_fuel]
  type                = ElementAverageValue
  variable            = T_fluid
  block               = ${fluid_blocks}
  execute_on          = 'initial timestep_end'
[]
[Tmax_core_fuel]
  type                = ElementExtremeValue
  value_type          = max
  variable            = T_fluid
  block               = 'core'
  execute_on          = 'initial timestep_end'
[]
[Tavg_core_fuel]
  type                = ElementAverageValue
  variable            = T_fluid
  block               = 'core'
  execute_on          = 'initial timestep_end'
[]
```

```
[Tavg_core_fuel]
  type                = ElementAverageValue
  variable            = T_fluid
  block               = 'core'
  execute_on          = 'initial timestep_end'
[]
[Tmax_mod]
  type                = ElementExtremeValue
  value_type          = max
  variable            = T_solid
  block               = 'core'
  execute_on          = 'initial timestep_end'
[]
[Tavg_mod]
  type                = ElementAverageValue
  variable            = T_solid
  block               = 'core'
  execute_on          = 'initial timestep_end'
[]
[power_total]
  type                = ElementIntegralVariablePostprocessor
  variable            = power_density
  execute_on          = 'initial timestep_end'
[]
[power_avg]
  type                = ElementAverageValue
  variable            = power_density
  execute_on          = 'initial timestep_end'
[]
[power_fuel_total]
  type                = ElementIntegralVariablePostprocessor
  variable            = power_density_fuel
  execute_on          = 'initial timestep_end'
[]
[power_ghrap_total]
  type                = ElementIntegralVariablePostprocessor
  variable            = power_density_graph
  execute_on          = 'initial timestep_end'
[]
[power_total_2]
  type                = ParsedPostprocessor
  function            = 'power_ghrap_total + power_fuel_total'
  pp_names            = 'power_ghrap_total power_fuel_total'
  execute_on          = 'initial timestep_end'
[]
```

Next Steps

- Understand auxiliary variables and auxiliary kernels
- Running the simulation: `mpirun -n 12 blue_crab-opt -i msre_ph_ss.i`

- Analyze the simulation integrating neutron precursors
  - The equation solved for the delayed neutron precursor concentration of type i is:

$$\underbrace{\frac{\partial c_i}{\partial t}}_{\text{Time derivative}} + \underbrace{\nabla \cdot \left(\frac{\boldsymbol{u}}{\gamma} c_i\right)}_{\text{Convection in Porous Media}} - \underbrace{\nabla \cdot \left[\left(D_{C_i} + \frac{\nu_t}{Sc_t}\right) \nabla c_i\right]}_{\text{Molecular and Turbulent Diffusion}} = \underbrace{y_i f}_{\text{Fission Yield Source}} - \underbrace{\lambda c_i}_{\text{Natural Decay Sink}}$$

  - Note how this implementation is performed in `msre_ph_ss_dnp.i`
  - Run the simulation with neutron precursors: `mpirun -n 12 blue_crab-opt -i msre_ph_ss_dnp.i`

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.

Basic Griffin Inputs

Mustafa K. Jaradat , Ph.D.

4/21/2024

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

Idaho National Laboratory

System Overview

- MOOSE syntax for all systems is block based
  - Includes extensive parameter definition to customize based on user needs
  - Check application documentation for available parameters

- MOOSE systems and their function
  - **Mesh** – mesh operations (build, read, modify, … mesh)
  - **Primal/Primary** -- generates variables and physics kernels that represent our PDEs
  - **Auxiliary** -- generates variables and kernels that store or compute data not in the primal system like power density, temperatures, etc.
  - **Material** -- used to assign the values of the coefficients in the PDEs (e.g., cross sections for Griffin)
  - **Output** -- storage of solutions
  - **Action** -- used for input simplification and special problem setup
  - **MultiApps & Transfers** -- used for multiphysics coupling between different application to transfer variables among each other.

MSRE Mesh

- Mesh System
  - Mesh operations (build, read, modify, … mesh)
- The example below:
  - Uses FileMeshGenerator to read an Exodus formatted mesh file
  - Loads a variable named material_id from the mesh file
  - Sets the coordinate type of the problem

```
# ============================================
# GEOMETRY AND MESH
# ============================================
[Mesh]
  [fmg]
    type     = FileMeshGenerator
    file     = '../../mesh_msre_in.e'
  []
  coord_type = 'RZ'
[]
```

Primal System

- Primal System
  - In Griffin we use an action to setup the primal system since setting up neutron transport problems requires many kernels
  - In the example we use the TransportSystems action to setup a diffusion problem
    - Particle type
    - Equation type
    - No. neutron energy groups
    - Boundary conditions
    - No. delayed neutron precursor groups
    - Quadratic Lagrange family
    - Jacobian parameters for improved convergence

```
# ============================================================
# TRANSPORT SYSTEM
# ============================================================
[TransportSystems]
  particle                          = neutron
  equation_type                     = eigenvalue
  G                                 = 16
  ReflectingBoundary                = 'left'
  VacuumBoundary                    = 'bottom right top loop
  [transport]
    scheme                          = CFEM-Diffusion
    family                          = LAGRANGE
    order                           = FIRST
    n_delay_groups                  = 6
    assemble_scattering_jacobian = true
    assemble_fission_jacobian    = true
    external_dnp_variable        = 'dnp'
    fission_source_aux              = true
  []
[]
```

Auxiliary & Material Systems

- Auxiliary System
  - Add DNPCs Aux
    - Add an aux variable named ‘c1’
    - Define mesh block(s) where to evaluate the tally
    - Add an aux kernel named ‘build_dnp’ and store it in variable name ‘dnp’
    - Group all DNPCs ‘c1 c2 c3 c4 c5 c6’ and store it in ‘dnp’ victor variable
    - ‘dnp’ is assigned ‘external_dnp_variable’ in the Primal System

```
# ================================================================
# AUXVARIABLES AND AUXKERNELS
# ================================================================
[AuxVariables]
  [c1]
    order             = CONSTANT
    family            = MONOMIAL
    block             = ${salt_blocks}
    initial_condition = 0.000
  []
[]
[AuxKernels]
  [build_dnp]
    type                = BuildArrayVariableAux
    variable            = dnp
    component_variables = 'c1 c2 c3 c4 c5 c6'
    execute_on          = 'initial timestep_begin final'
  []
[]
```

```
# ================================================================
# MATERIALS
# ================================================================
[Materials]
  [activeCore]
    type            = CoupledFeedbackNeutronicsMaterial
    isotopes        = '    C12     U235     U238      BE9      LI7     F19
                         ZR90     ZR91     ZR92     ZR94     ZR96'
    densities       = ' ad_C12  ad_U235  ad_U238   ad_Be9   ad_Li7   ad_F9
                       ad_Zr90  ad_Zr91  ad_Zr92  ad_Zr94  ad_Zr96'
    material_id     = 1
    block           = 'core'
  []
[]
```

- Material System
  - Use a Griffin material to define the mesh block, isotope names, isotope number densities

Executioner & Output Systems

- Executioner System
  - Eigenvalue (nonlinear power iteration method)
  - PJFNKMO – Matrix only PJFNK
  - Nonlinear tolerance (for Newton iteration)
  - Maximum number of iteration

```
# ================================
# EXECUTION PARAMETERS
# ================================
[Executioner]
  type                = Eigenvalue
  solve_type          = PJFNKMO
  l_max_its           = 200
  nl_max_its          = 200
  nl_abs_tol          = 1e-6
[]
```

```
# ================================================================
# OUTPUTS
# ================================================================
[Outputs]
  file_base                   = msre_ss_out
  csv                         = true
  exodus                      = true
  perf_graph                  = true
  print_linear_converged_reason = false
  print_linear_residuals      = false
  execute_on                  = 'INITIAL FINAL TIMESTEP_END'
[]
```

- Output System
  - **File_base** is the root name for output files
  - Outputs requested in Exodus II format and CSV
  - **Perf_graph** is the table with performance metrics
  - User chooses the execution of outputs (initial, timestep_begin, timestep_end, final)

MultiApps & Transfers Systems

- MultiApps System
  - Type of the app (FullSolveMultiApp, TransientMultiApp,)
  - Input file of subapp
  - Execution time (initial, final, timestep_end,)
  - Maximum number of processors
  - Takes the final solution from the previous coupling iteration and re-uses it as the initial guess

```
# ================================================
# MULTIAPPS AND TRANSFERS
# ================================================
[MultiApps]
  [flow_dnp]
    type                          = FullSolveMultiApp
    input_files                   = 'msre_ph_ss.i'
    execute_on                    = 'timestep_end'
    max_procs_per_app             = 48
    keep_solution_during_restore = true
  []
[]
```

```
[Transfers]
  [power_density]
    type                  = MultiAppGeneralFieldShapeEvaluationTransfer
    to_multi_app          = flow_dnp
    source_variable       = 'power_density'
    variable              = 'power_density'
    execute_on            = 'timestep_end'
  []
  [c1]
    type                  = MultiAppGeneralFieldShapeEvaluationTransfer
    from_multi_app        = flow_dnp
    source_variable       = 'c1'
    variable              = 'c1'
    execute_on            = 'timestep_end'
  []
[]
```

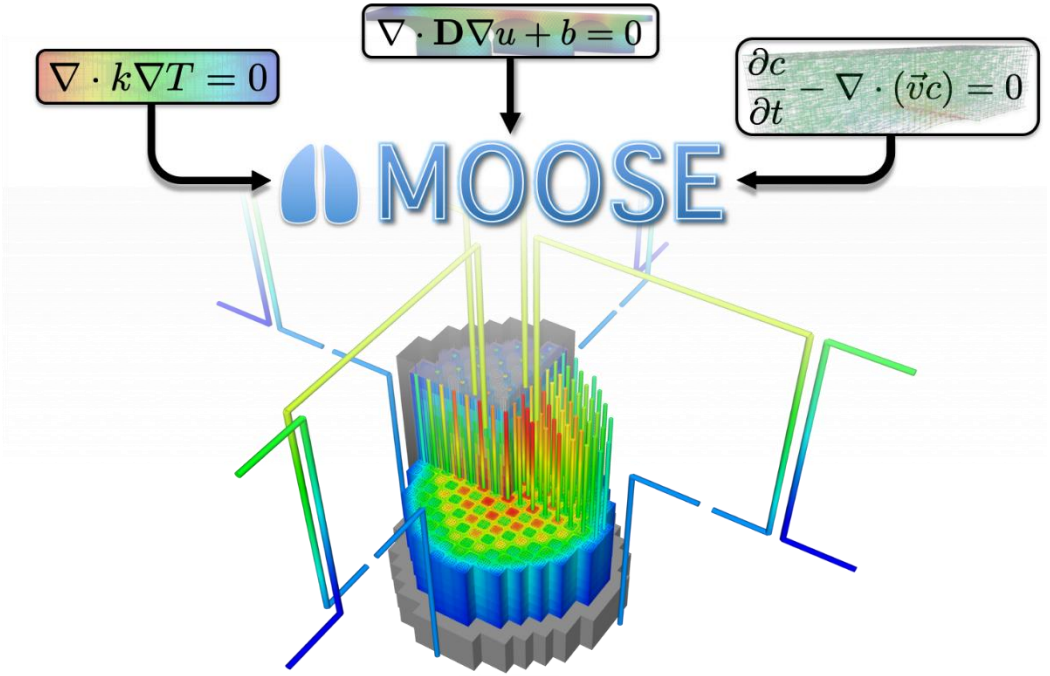
- Transfers System
  - Type of the transfer
  - Direction of the transfer (from_multi_app, to_multi_app)
  - Transferred and received variables
  - Execution time (initial, final, timestep_end,)

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.

MOOSE BASICS

- Multiphysics Object-Oriented Simulation Environment (MOOSE)
- Various spatial discretization methods
  - Finite Element: Continuous and Discontinuous Galerkin
  - Finite Volume: Cell centered variables, interpolation to faces with limiters, orthogonality & skewness correction
- 1D, 2D and 3D Cartesian, 2D RZ, 1D R-Spherical
- Unstructured Mesh
  - Many shapes (polygons)
  - Higher order geometry (curvilinear)
  - Reads and writes multiple formats
- Parallel
- Built-in Postprocessing



MOOSE BASICS

- Our main purpose is to solve PDEs (like the neutron transport equation)

$$\underbrace{\frac{1}{v}\frac{\partial \psi}{\partial t}}_{\text{Time derivative}} + \underbrace{\Omega \cdot \nabla \psi}_{\text{Streaming}} + \underbrace{\Sigma_t \psi}_{\text{Collision}} = \underbrace{\int_0^\infty \int_{4\pi} \Sigma_s(\Omega' \cdot \Omega, E' \rightarrow E)\psi(\Omega', E')d\Omega' dE' +}_{\text{Scattering}}$$

$$\underbrace{\frac{\chi_p}{k_{\text{eff}} 4\pi} \int_0^\infty \nu \Sigma_f(E')\phi(E')dE'}_{\text{Fission}} + \underbrace{\frac{\chi_d}{4\pi} \sum_{i=1}^{I} \lambda_i C_i}_{\text{Delayed Neutron}}$$

- For this purpose, we require
  - A finite element mesh where to solve the PDEs
  - Coefficients for the PDEs (i.e., material properties, neutron cross sections)
  - Instructions on what PDEs to solve (Physics/MOOSE kernels)
  - Instructions on what output to generate
- MOOSE provides various systems to handle these requirements

Transient & Restart

Mustafa K. Jaradat , Ph.D.

4/21/2024

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

Idaho National Laboratory

Griffin: Steady State vs. Transient

Steady State

```
# ================================================================
# TRANSPORT SYSTEM
# ================================================================
[TransportSystems]
  particle                        = neutron
  equation_type                   = eigenvalue
  G                               = 16
  ReflectingBoundary              = 'left'
  VacuumBoundary                  = 'bottom right top loop_b
  [transport]
    scheme                        = CFEM-Diffusion
    family                        = LAGRANGE
    order                         = FIRST
    n_delay_groups                = 6
    assemble_scattering_jacobian = true
    assemble_fission_jacobian     = true
    external_dnp_variable         = 'dnp'
    fission_source_aux            = true
  []
[]
```

Transient

```
# ================================================================
# TRANSPORT SYSTEM
# ================================================================
[TransportSystems]
  particle                        = neutron
  equation_type                   = transient
  G                               = 16
  ReflectingBoundary              = 'left'
  VacuumBoundary                  = 'bottom right top loop_
  [transport]
    scheme                        = CFEM-Diffusion
    family                        = LAGRANGE
    order                         = FIRST
    n_delay_groups                = 6
    assemble_scattering_jacobian = true
    assemble_fission_jacobian     = true
    external_dnp_variable         = 'dnp'
    fission_source_aux            = true
  []
[]
```

- Remove initial conditions of the restarted variables.

Griffin: Steady State vs. Transient

Steady State

```
[Executioner]
  type                = Eigenvalue
  solve_type          = PJFNKMO
  petsc_options_iname = '-pc_type -pc_hypre_type -ksp_gmres_restart'
  petsc_options_value = 'hypre boomeramg 50'
  l_max_its           = 200
  nl_abs_tol          = 1e-6
  fixed_point_min_its = 2
  fixed_point_max_its = 50
  fixed_point_rel_tol = 1e-6
  fixed_point_abs_tol = 1e-6
[]
```

Transient

```
[Executioner]
  type                = Transient
  solve_type          = PJFNK
  petsc_options_iname = '-pc_type -pc_hypre_type -ksp_gmres_restart'
  petsc_options_value = 'hypre boomeramg 50'

  start_time = 0.0
  [TimeStepper]
    type                = IterationAdaptiveDT
    dt                  = 1.0
    timestep_limiting_postprocessor = dt_max_pp
    optimal_iterations  = 20
    iteration_window    = 2
    growth_factor       = 2
    cutback_factor      = 0.5
  []
  end_time            = 2000.0
  auto_advance        = true

  l_max_its           = 200
  nl_abs_tol          = 1e-5
  fixed_point_min_its = 1
  fixed_point_max_its = 50
  fixed_point_rel_tol = 1e-5
  fixed_point_abs_tol = 1e-5
[]
```

Restart Griffin Steady State Solution

- Remove initial conditions of the restarted variables.
- Restart steady state transport solution from binary file using user object.

```
# ================================================================================================
# USER OBJECTS FOR RESTART
# ================================================================================================
[UserObjects]
  [transport_solution]
    type             = TransportSolutionVectorFile
    transport_system = transport
    writing          = false
    execute_on       = 'INITIAL'
    scale_with_keff  = false
    folder           = '../../ex2_ss/ex23_fp_multiphysics/'
  []
  [auxvar_solution]
    type             = SolutionVectorFile
    var              = '  vel_x   vel_y  T_salt  T_solid     c1     c2     c3     c4     c5     c6     ad_C12
                         ad_U235 ad_U238  ad_Be9   ad_Li7  ad_F9  ad_Zr90  ad_Zr91  ad_Zr92  ad_Zr94   ad_Zr96'
    loading_var      = '  vel_x   vel_y  T_salt  T_solid     c1     c2     c3     c4     c5     c6     ad_C12
                         ad_U235 ad_U238  ad_Be9   ad_Li7  ad_F9  ad_Zr90  ad_Zr91  ad_Zr92  ad_Zr94   ad_Zr96'
    writing          = false
    execute_on       = 'INITIAL'
    folder           = '../../ex2_ss/ex23_fp_multiphysics/'
  []
[]
```

```
[AuxVariables]
  [T_salt]
    family           = MONOMIAL
    order            = CONSTANT
    block            = ${all_blocks}
  []
  [T_solid]
    family           = MONOMIAL
    order            = CONSTANT
    block            = ${solid_blocks}
  []
  [c1]
    order            = CONSTANT
    family           = MONOMIAL
    block            = ${salt_blocks}
  []
  [c2]
    order            = CONSTANT
    family           = MONOMIAL
    block            = ${salt_blocks}
  []
  [c3]
    order            = CONSTANT
    family           = MONOMIAL
    block            = ${salt_blocks}
  []
  [c4]
    order            = CONSTANT
    family           = MONOMIAL
    block            = ${salt_blocks}
  []
```

Restart Pronghorn Steady State Solution

- Remove initial conditions of the restarted variables.
- Restart steady state solution from exodus output file.

```
# ================================================================================
# GEOMETRY AND MESH
# ================================================================================
[Mesh]
  coord_type                = 'RZ'
  [fmg]
    type                    = FileMeshGenerator
    file                    = '../../ex2_ss/ex23_fp_multiphysics/msre_ss_out_flow_dnp0.e'
    use_for_exodus_restart = true
  []
[]
```

```
# ================================================================
# FV VARIABLES
# ================================================================
[Variables]
  [superficial_vel_x]
    type                      = PINSFVSuperficialVelocityVariable
    block                     = ${fluid_blocks}
    initial_from_file_var     = superficial_vel_x
    initial_from_file_timestep = LATEST
  []
  [superficial_vel_y]
    type                      = PINSFVSuperficialVelocityVariable
    block                     = ${fluid_blocks}
    initial_from_file_var     = superficial_vel_y
    initial_from_file_timestep = LATEST
  []
  [pressure]
    type                      = INSFVPressureVariable
    block                     = ${fluid_blocks}
    initial_from_file_var     = pressure
    initial_from_file_timestep = LATEST
  []
  [T_fluid]
    type                      = INSFVEnergyVariable
    block                     = ${fluid_blocks}
    initial_from_file_var     = T_fluid
    initial_from_file_timestep = LATEST
  []
```

```
[AuxVariables]
  [power_density]
    type                      = MooseVariableFVReal
    initial_from_file_var     = power_density
    initial_from_file_timestep = LATEST
  []
  [power_density_fuel]
    type                      = MooseVariableFVReal
    initial_from_file_var     = power_density_fuel
    initial_from_file_timestep = LATEST
  []
  [power_density_graph]
    type                      = MooseVariableFVReal
    initial_from_file_var     = power_density_graph
    initial_from_file_timestep = LATEST
  []
  [fission_source]
    type                      = MooseVariableFVReal
    initial_from_file_var     = fission_source
    initial_from_file_timestep = LATEST
  []
```

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.

Species Tracking

Mauricio E. Tano, Ph.D.

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

Idaho National Laboratory

Adding two-phase flow modeling to thermal-hydraulics model
Pronghorn's two-phase mixture model

The mixture continuity equation:

$$\frac{\partial \rho_m}{\partial t} + \nabla \cdot (\rho_m \bar{v}_m / \gamma) = 0$$

The continuity equation for phase 2:

$$\frac{\partial \alpha_2 \rho_2}{\partial t} + \nabla \cdot (\alpha_2 \rho_2 \bar{v}_m / \gamma) = \Gamma_2 - \nabla \cdot \left(\frac{\alpha_2 \rho_1 \rho_2}{\rho_m} \bar{v}_{slip,d} \right)$$

The mixture momentum equation:

$$\frac{\partial}{\partial t} \rho_m \bar{v}_m + \nabla \cdot (\rho_m \bar{v}_m \bar{v}_m / \gamma) = -\nabla \gamma p_m + \nabla \cdot (\alpha_1 \mu_1 + \alpha_2 \mu_2) \nabla \bar{v}_m - \nabla \cdot \left(\frac{\alpha_2}{1 - \alpha_2} \frac{\rho_1 \rho_2}{\rho_m} \bar{v}_{slip,d} \bar{v}_{slip,d} / \gamma \right) + \gamma \rho_m g_m + \gamma M_m$$

The slip velocity is modeled as

$$\bar{v}_{slip,d} = \frac{\tau_d}{f_{drag}} \frac{\rho_d - \rho_m}{\rho_d} \bar{a}$$

where:

- τ_d is the particle relaxation time, f_{drag} is the linear drag coefficient function, ρ_d is the density of the dispersed phase, $\bar{a}$ is the acceleration vector.

The particle relaxation time is modeled as follows Bilicki and Kestin (1990):

$$\tau_d = \frac{\rho_d d_d^2}{18 \mu_m}$$

where:

- d_d is the particle diameter, μ_m is the mixture dynamic viscosity.

The acceleration vector is the particle acceleration vector:

$$\bar{a} = \bar{g} + \frac{\bar{f}}{\rho_m} - \bar{u}_m \cdot \nabla \bar{u}_m - \frac{\partial \bar{u}_m}{\partial t}$$

where:

- $\bar{f}$ is the volumetric force

Adding two phase flow to thermal-hydraulics

Adding Void Fraction Solution Variable

```
[void_f]
  type = INSFVScalarFieldVariable
  initial_from_file_var = void_f
  block = ${fluid_blocks}
[]
```

Adding Void Fraction Kernels

```
[void_f_time]
  type = FVFunctorTimeKernel
  variable = void_f
  block = ${fluid_blocks}
[]
[void_f_advection]
  type = INSFVScalarFieldAdvection
  variable = void_f
  u_slip = 'superficial_vel_slip_x'
  v_slip = 'superficial_vel_slip_y'
  block = ${fluid_blocks}
[]
[vod_f_diffusion]
  type = INSFVMixingLengthScalarDiffusion
  schmidt_number = '${fparse Sc_t/10.0}'
  variable = void_f
  block = ${fluid_blocks}
[]
[void_f_src]
  type = FVCoupledForce
  variable = void_f
  v = fission_source
  coef = ${gas_fraction}
[]
[void_f_sink_pump]
  type = NSFVEnergyAmbientConvection
  variable = void_f
  T_ambient = 0.0
  alpha = 100.0
  block = 'pump'
[]
```

Time Derivative

Advection via Slip Velocity

Turbulent Diffusion

Void Source due to Nuclear Fission

Void Sink at Pump due to Gas Removal

Defining mixture properties materials

```
[mixing_material]
  type = NSFVMixtureFunctorMaterial
  phase_1_names = 'rho mu'
  phase_2_names = '${rho_d} ${mu_d}'
  prop_names = 'rho_mixture mu_mixture'
  phase_1_fraction = 'phase_1'
  output_properties = 'rho_mixture mu_mixture'
  block = ${fluid_blocks}
[]
```

Defines mixture properties for density and viscosity as a function of the liquid fraction

Compute the arithmetic mean of material properties using a phase fraction.

This material is mainly used in multiphase modeling. Given a phase fraction functor "phase_1_fraction" (λ_1) and two vectors of (functor) properties "phase_1_names" and "phase_2_names", named generically $p_{i,1}$ and $p_{i,2}$, respectively, the material computes the phase-weighted average of the property as follows:

$$p_i = \lambda_1 p_{i,1} + (1 - \lambda_1) p_{i,2} \, ,$$

From:
https://mooseframework.inl.gov/source/functormaterials/NSFVMixtureFunctorMaterial.html

Defining slip velocities (drift flux formulation)

```
[populate_u_slip]
  type = WCNSFV2PSlipVelocityFunctorMaterial
  slip_velocity_name = 'superficial_vel_slip_x_no_porous'
  momentum_component = 'x'
  u = superficial_vel_x
  v = superficial_vel_y
  rho = 'rho_mixture'
  mu = 'mu_mixture'
  rho_d = ${rho_d}
  particle_diameter = ${dp}
  block = ${fluid_blocks}
  linear_coef_name = 100.0
[]
[populate_v_slip]
  type = WCNSFV2PSlipVelocityFunctorMaterial
  slip_velocity_name = 'superficial_vel_slip_y_no_porous'
  momentum_component = 'y'
  u = superficial_vel_x
  v = superficial_vel_y
  rho = 'rho_mixture'
  mu = 'mu_mixture'
  rho_d = ${rho_d}
  particle_diameter = ${dp}
  block = ${fluid_blocks}
  linear_coef_name = 100.0
[]
```

Defines slip velocity in the ‘x’ direction

Defines slip velocity in the ‘y’ direction

Defining interface area concentration

```
[interface_area_concentration]
  type = ADParsedFunctorMaterial
  expression = 'void_f * 6 / ${dp}'
  functor_names = 'void_f'
  functor_symbols = 'void_f'
  property_name = 'interface_area_concentration'
[]
```

Defines interface area concentration

1) $\chi_g = \frac{A_{lg}}{V}$

2) $A_g = n_b 4\pi R^2$

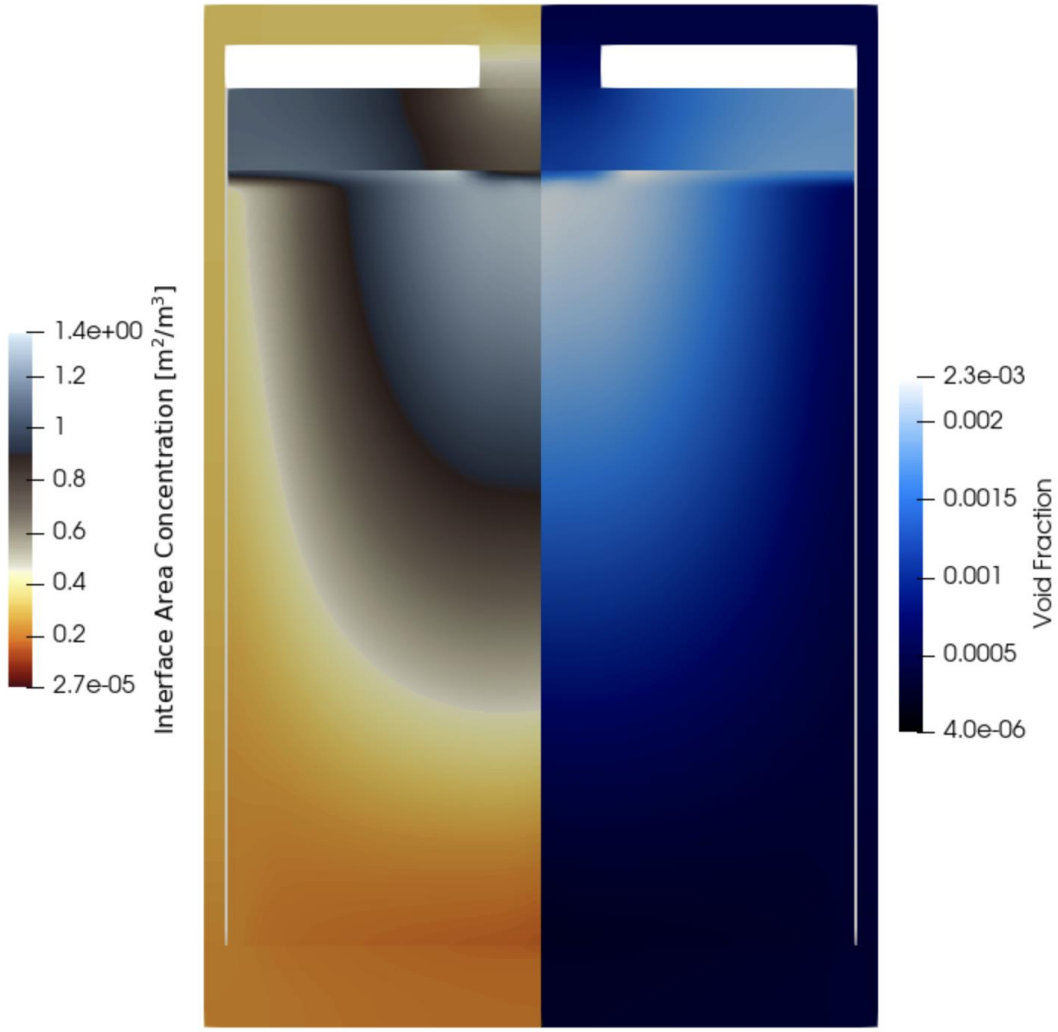
3) $\alpha_g = \frac{V_g}{V} = \frac{1}{V}\, n_b \frac{4}{3}\pi R^3 \rightarrow V = \frac{1}{\alpha_g}\, n_b\, \frac{4}{3}\pi R^3$

2 and 3 into 1) $\chi_g = \alpha_g \frac{n_b 4\pi R^2}{n_b \frac{4}{3}\pi R^3} = \alpha_g \frac{3}{R} = \alpha_g \frac{6}{D_p}$

```
# Average bubble diameter
dp = 0.01
```

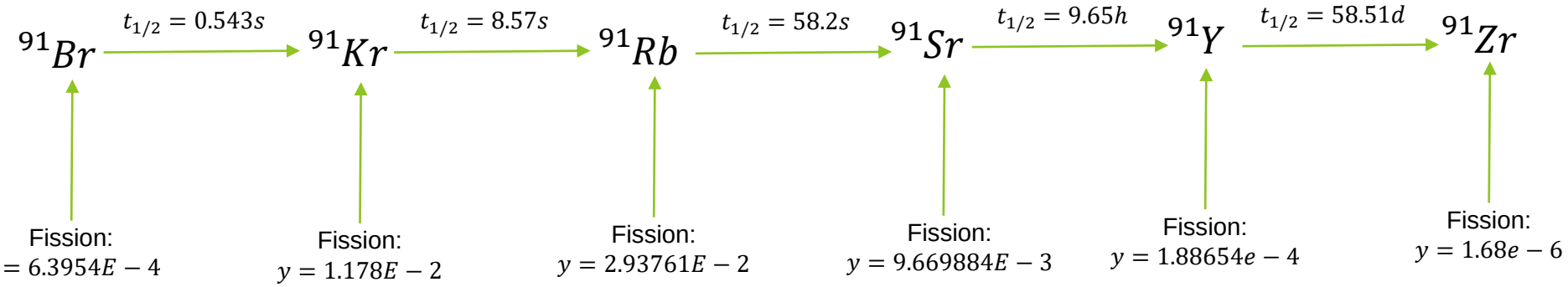
Steady-state void distribution

- Void increases as the fuel circulate through the reactor core
- As flow mixes in the upper plenum two different behaviors are observed:
  - Flow jets to the riser, where void is removed at the pump
  - Flows is more occluded towards the external radius of the upper plenum, resulting on a higher void concentration
- Interface area concentration and hence, liquid-gas exchange processes are larger at the top of the core and the partially occluded flow in the outer radius of the upper plenum



Species Tracking
Example tracking the ^{91}Br depletion chain

- The depletion chain for reads as follows:



- In practice, these reactions happen while species are transported in the fluid. We solve the following equation for isotope concentration c_i^x for isotope i in state x:

$$\frac{\partial c_i^x}{\partial t} + \nabla \cdot \left(\frac{\boldsymbol{u}}{\gamma} c_i^x\right) - \nabla \cdot \left[\left(D_{c_i} + \frac{\nu_t}{Sc_t}\right)\nabla c_i^x\right] = y_i f - \lambda c_i^x + h^{xy}(c_i^y - c_i^x)$$

Time derivative | Convection in Porous Media | Molecular and Turbulent Diffusion | Fission Yield Source | Natural Decay Sink | Exchange of phases between states y and x

Defining the solution variables

```
[Variables]
  [Br91_f]
    type = MooseVariableFVReal
    block = ${fluid_blocks}
  []
  [Br91_g]
    type = MooseVariableFVReal
    block = ${fluid_blocks}
  []
  [Kr91_f]
    type = MooseVariableFVReal
    block = ${fluid_blocks}
  []
  [Kr91_g]
    type = MooseVariableFVReal
    block = ${fluid_blocks}
  []
  [Rb91_f]
    type = MooseVariableFVReal
    block = ${fluid_blocks}
  []
  [Rb91_g]
    type = MooseVariableFVReal
    block = ${fluid_blocks}
  []
  [Sr91_f]
    type = MooseVariableFVReal
    block = ${fluid_blocks}
  []
```

For each of the tracked isotopes, solution variables are defined in the liquid (dissolved) and gaseous states

Defining the solution kernels
Example ^{91}Kr

```
[Kr91_time]
  type = FVFunctorTimeKernel
  variable = 'Kr91_f'
  block = ${fluid_blocks}
[]
[Kr91_advection]
  type = PINSFVMassAdvection
  variable = 'Kr91_f'
  rho = 'Kr91_porous'
  block = ${fluid_blocks}
[]
[Kr91_diffusion]
  type = FVDiffusion
  variable = 'Kr91_f'
  coeff = ${D_mu}
  block = ${fluid_blocks}
[]
[Kr91_turb_diffusion]
  type = INSFVMixingLengthScalarDiffusion
  variable = 'Kr91_f'
  schmidt_number = ${Sc_t}
  block = ${fluid_blocks}
[]
[Kr91_src]
  type = FVCoupledForce
  variable = 'Kr91_f'
  v = fission_source
  coef = ${yield_Kr91}
  block = ${fluid_blocks}
[]
[Kr91_decay]
  type = FVReaction
  variable = 'Kr91_f'
  rate = ${lambda_Kr91}
  block = ${fluid_blocks}
[]
[Kr91_grow]
  type = FVCoupledForce
  variable = 'Kr91_f'
  v = 'Br91_f'
  coef = ${lambda_Br91}
  block = ${fluid_blocks}
[]
[Kr91_fg]
  type = NSFVMixturePhaseInterface
  variable = 'Kr91_f'
  phase_coupled = 'Kr91_g'
  alpha = 'hlg_Kr91'
[]
```

Time derivative

Porous media advection

Molecular diffusion

Turbulent diffusion

Fission source

Natural decay

Buildup due to ^{91}Br

Fluid-to-gas exchange

```
[Kr91_gf]
  type = NSFVMixturePhaseInterface
  variable = 'Kr91_g'
  phase_coupled = 'Kr91_f'
  alpha = 'hlg_Kr91'
[]
[Kr91_g_advection]
  type = PINSFVMassAdvection
  variable = 'Kr91_g'
  rho = 'Kr91_g'
  block = ${fluid_blocks}
[]
[Kr91_g_diffusion]
  type = FVDiffusion
  variable = 'Kr91_g'
  coeff = ${D_mu_g}
  block = ${fluid_blocks}
[]
[Kr91_g_decay]
  type = FVReaction
  variable = 'Kr91_g'
  rate = ${lambda_Kr91}
  block = ${fluid_blocks}
[]
[Kr91_grow_g]
  type = FVCoupledForce
  variable = 'Kr91_g'
  v = 'Br91_g'
  coef = ${lambda_Br91}
  block = ${fluid_blocks}
[]
```

Gas-to-fluid exchange

Advection is porous media

Molecular gas diffusion

Natural decay

Growth due to decay of ^{91}Br in the gas phase

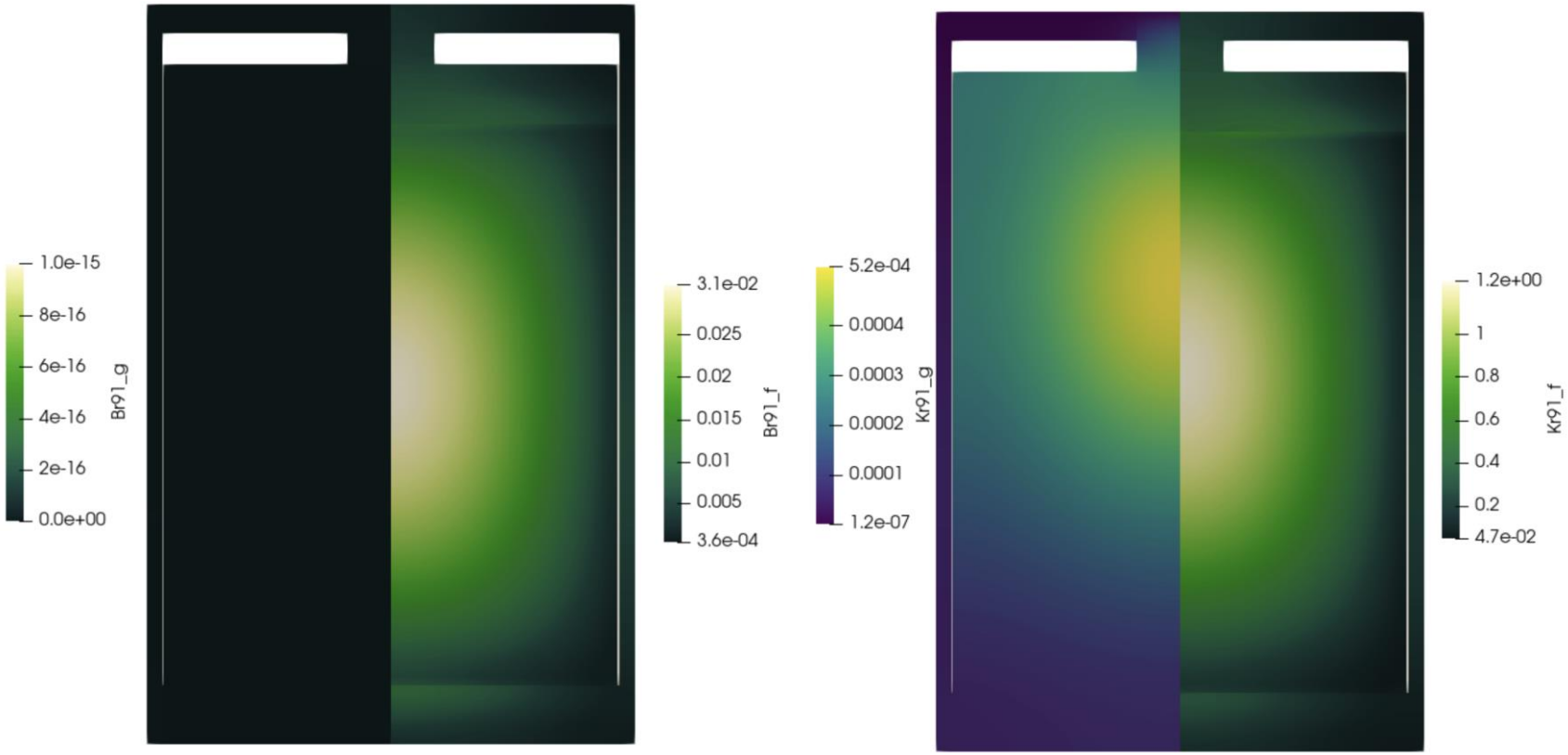
Liquid-gas exchange coefficients

```
[hlg_Kr91]
  type = ADParsedFunctorMaterial
  expression = 'interface_area_concentration * ${Kr91_alpha_fg}'
  functor_names = 'interface_area_concentration'
  functor_symbols = 'interface_area_concentration'
  property_name = 'hlg_Kr91'
[]
```

For every species, the net volumetric mass exchange coefficient is defined as the surface-specific mass exchange coefficient multiplied by the interface area concentration

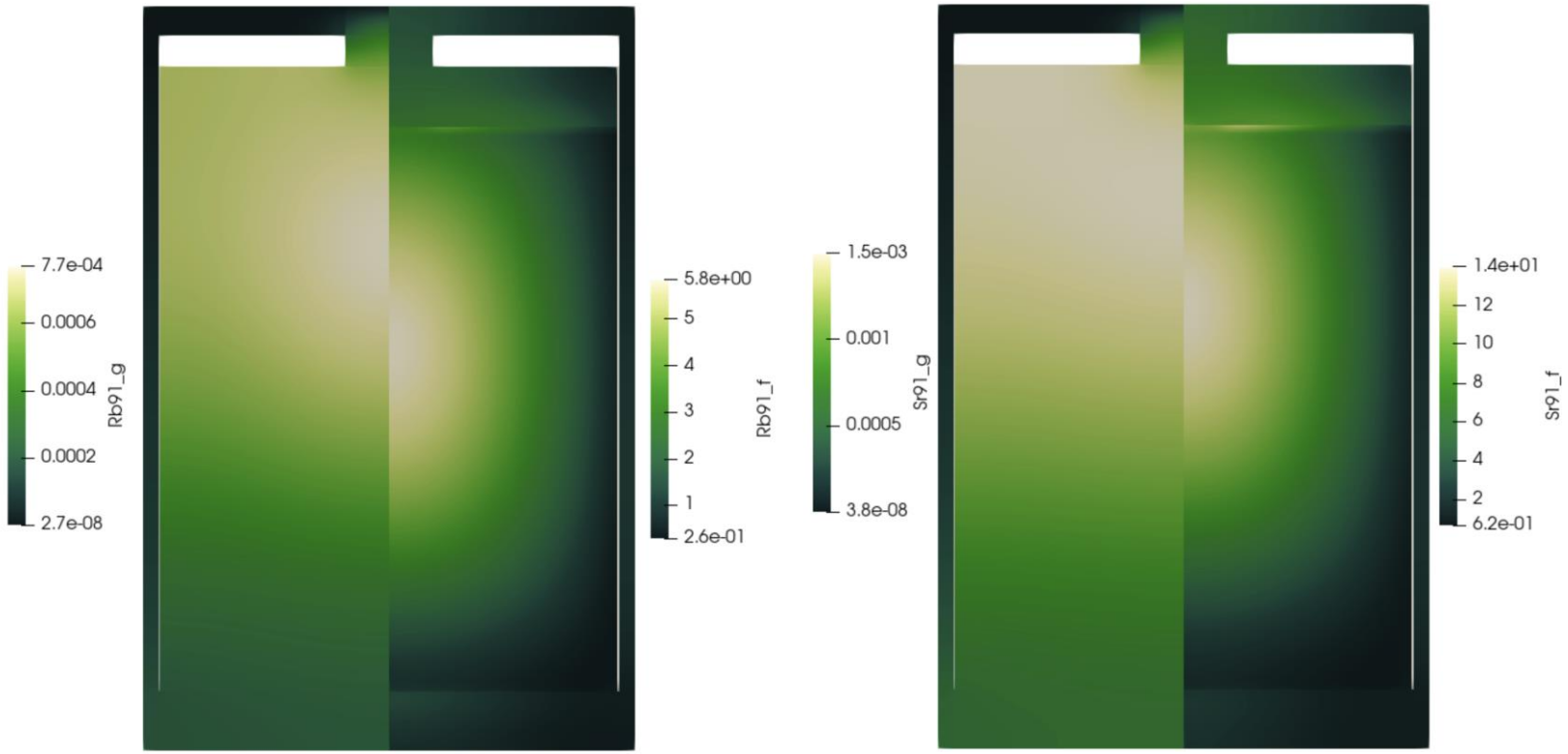
Simulation Results 1/3

- ^{91}Br is produced by fission and stays in solution in the fuel salt
- No gaseous ^{91}Br is present
- ^{91}Kr is produced by fission and by natural decay of ^{91}Br in the fuel salt
- Some ^{91}Kr converts into the gas phase, where is transported toward the extraction point at the top of the reactor core



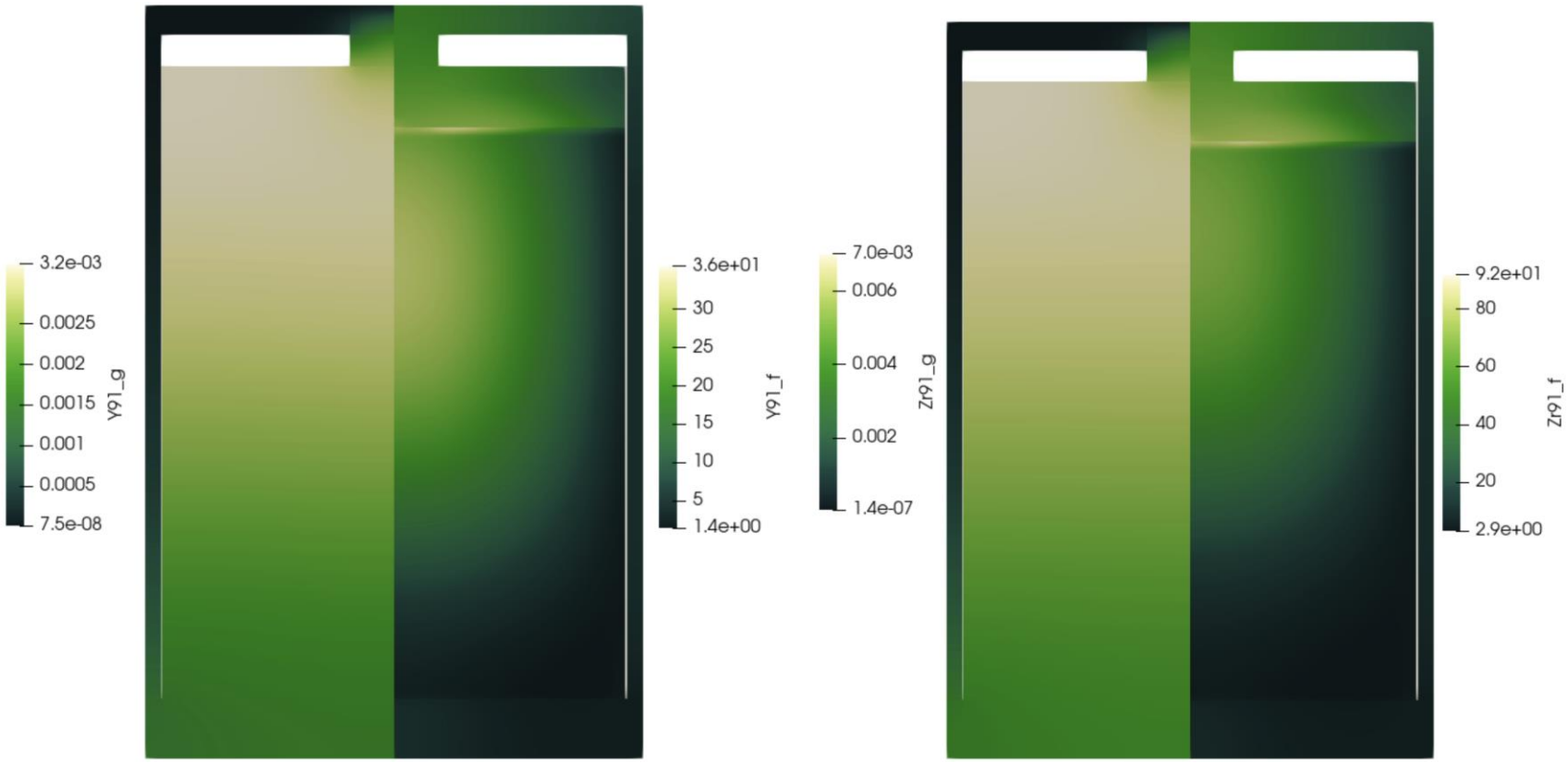
Simulation Results 2/3

- ^{91}Rb is produced by fission and by the decay of ^{91}Kr in the fuel salt
- Also, ^{91}Rb is produced by the decay of ^{91}Kr in the gas phase
- A similar behavior is observed for ^{91}Sr



Simulation Results 3/3

- ^{91}Y is produced by fission and by the decay of ^{91}Sr in the fuel salt
- There is no direct exchange between the liquid and the gas phase for ^{91}Y
- Also, ^{91}Y is produced by the decay of ^{91}Sr in the gas phase
- A similar behavior is observed for ^{91}Zr



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.