



Multiphysics Modeling/Simulation Capabilities Based on MOOSE

May 2021

Changing the World's Energy Future

Mark D DeHart, Sebastian Schunert, Vincent M Laboure



INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

Multiphysics Modeling/Simulation Capabilities Based on MOOSE

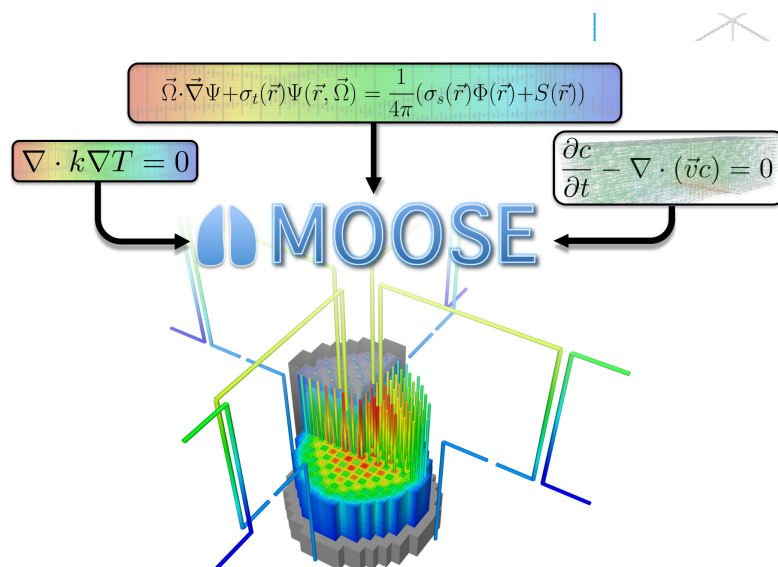
Mark D DeHart, Sebastian Schunert, Vincent M Laboure

May 2021

**Idaho National Laboratory
Idaho Falls, Idaho 83415**

<http://www.inl.gov>

**Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517**



May 26, 2021

Multiphysics Modeling/Simulation Capabilities Based on MOOSE

Presented at the NASA Nuclear Thermal Propulsion Modeling/Simulation Update

Presented by Mark DeHart, Sebastian Schunert and Vincent Labouré



Problem Statement

- In FY20, a portion of the INL Reactor Multiphysics Team (RMT) was chartered to introduce our multiphysics-based approach for reactor-based M&S.
 - Demonstrate the use of our tools
 - Develop representative models using these tools
 - 2D then 3D element
 - 2D then 3D core
 - Train and support NASA staff in use of these tools
 - Transfer models to NASA staff for their own analyses
- In FY21, the above work continued, but mostly in a support role.
- RMT was also tasked with using the same tools for simulation of SIRIUS experiments in the Transient Test Reactor (TREAT)
 - Validate our tools with data from the SIRIUS experiments completed to date
 - Use those tools to support design of PRIME-1
- For today, we will focus on the full core analysis efforts and demonstrate the application of these tools.



MOOSE and the Herd

- You are going to hear a lot of animal names thrown around. Let me explain...
- MOOSE (Multiphysics Object-Oriented Simulation Environment) is a framework from which other software can be developed.
 - For the most part, applications build using MOOSE are given animal names.
 - Most of those animals live(d) in what is now Idaho
 - The herd is also often referred to as the MOOSE ecosystem.
- Griffin is the reactor physics tool that most of this work has been built on.
- Bison is a fuels performance application used worldwide that will be used in future work
- Marmot is a mesoscale materials package predict the coevolution of microstructure and material properties of nuclear fuels and claddings due environmental effects
- THM (thermal hydraulics module) is a developmental single-phase compressible flow tool that will eventually be merged into MOOSE.
- MOOSE is a library of C++ classes that make coupled multiphysics relatively straightforward.
- All the other *animals* include MOOSE and some or all of its constituent modules and do all the real physics work.

What is MOOSE?

- Written in C++, is an objected-oriented framework that exists as a set of linkable libraries.
- MOOSE uses the finite element method and takes physics equations (specifies in an equation-like format) and internally develops the FEM equivalent
- To solve a specific physics equation, that equation is written in its *weak form* (a FEM requirement) and the equation (or set of equations) are written in the form of MOOSE *Kernels*. A kernel is a “piece” of physics.
- It's convenient to think of a kernel as a mathematical operator, such as a Laplacian or a convection term in a PDE.
- A kernel is typically composed of single lines of C++ code for the mathematical operator, the exact or approximate Jacobian, and one for each boundary condition.
- In MOOSE, kernels may be swapped or coupled together to achieve different application goals.
- Sets of related kernels are often packaged together as MOOSE *modules*.

MOOSE History and Purpose

- Development started in 2008
- Open-sourced in 2014
- Designed to solve computational engineering problems and reduce the expense and time required to develop new applications by:
 - being easily extended and maintained
 - working efficiently on a few and many processors
 - providing an object-oriented, pluggable system for creating all aspects of a simulation tool
 - Not written specifically for any specific type of physics
 - Applicable to any form of physics expressed as a set of PDEs



What is MOOSE?

“To solve a specific physics equation, that equation is written in its weak form and the equation(s) are written in the form of MOOSE Kernels”

Strong Form

$$\rho C_p \frac{\partial T}{\partial t} - \nabla \cdot k(T, B) \nabla T = f$$

Weak Form

$$\int_{\Omega} \rho C_p \frac{\partial T}{\partial t} \psi_i + \int_{\Omega} k \nabla T \cdot \nabla \psi_i - \int_{\partial\Omega} k \nabla T \cdot \mathbf{n} \psi_i - \int_{\Omega} f \psi_i = 0$$

Kernel Kernel BoundaryCondition Kernel

Actual Code

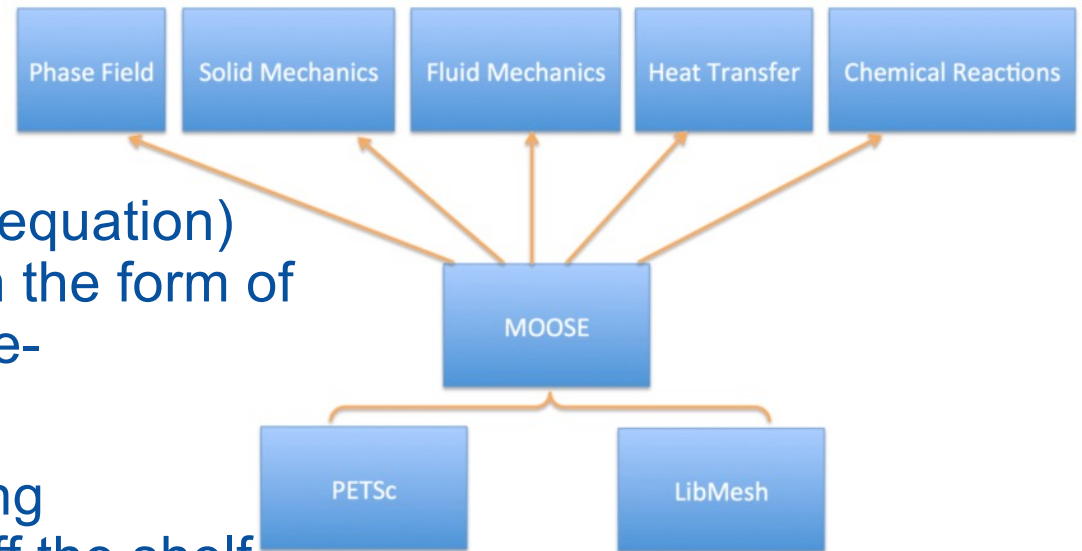
```
return _k[_qp]*_grad_u[_qp]*_grad_test[_i][_qp];
```

- where:
 - `_k` is the conductivity
 - `_grad_u` is the gradient of the temperature
 - `_grad_test` is the FEM test function

- A kernel is generated for each term in the equation
- Given properties (k , C_p , ρ), BC and a mesh, this C++ expression can essentially be linked to MOOSE and run (1, 2 or 3-D)
- MOOSE is only a framework including a solver – MOOSE can’t do anything by itself. Each new “animal” is compiled with and contains MOOSE

What is MOOSE (continued)

- Many kernels (e.g., conduction equation) are already built into MOOSE in the form of *modules* that don't need to be re-developed
- MOOSE is really good for solving uncommon physics where no off the shelf algorithms exist – a complete application can be developed literally in hours rather than weeks/months.
- MOOSE leverages scientific matrix solvers in PETSc and meshing toolset in libMesh
 - PETSc really helps for code parallelization as users typically only have to specify the desired number of procs/threads – the high-performance solution is inherited largely from PETSc. Almost all of the mathematical solution is done in PETSc and is very high-performance
 - libMesh provides mesh-related tools that we use for generating meshes and data translation





QA Processes for Development & Testing

- MOOSE-based software is all developed using the NQA-1 SQA standard
- It leverages 30 years of advancements in software design, numerical methods and physical models
- Designed to be easily extended and maintained, MOOSE and most native apps follows the continuous integration workflow
- MOOSE relies heavily on a robust automated testing tool called CIVET
- Development first merges into *next* branch, followed by additional automatic testing and updates to *devel* and *master* branches
- MOOSE is hosted on github; export-controlled applications are hosted in INL internal gitlab servers
- Tests are developed as part of the code and reside in the repository
- Most tests are executed 60 times with varying compilers, parallel execution settings, operating systems

MOOSE Documentation with MooseDocs

- Creating quality documentation can be a daunting task; it is not uncommon for large projects to have limited, difficult-to-navigate, out-of-date documentation.
- MOOSE documentation provides tools to automatically generate and stay up-to-date on NQA-1 documentation
- MOOSE documentation is part of the source code distribution and is version controlled
- Documentation is written in *MOOSE Markdown* syntax; it pulls in source code and test input syntax
- Cross-referencing links to examples and source code may be created.
- Dynamic source code listings allow source to be displayed without duplication and update automatically with code changes.
- Code merge requests are not approved until corresponding documentation changes (if necessary) are completed and reviewed.
- The markdown files may be converted into various media forms, including web pages, slide shows, and PDF
- Custom markdown syntax is easily created so it is possible to develop syntax to meet the needs of any project.

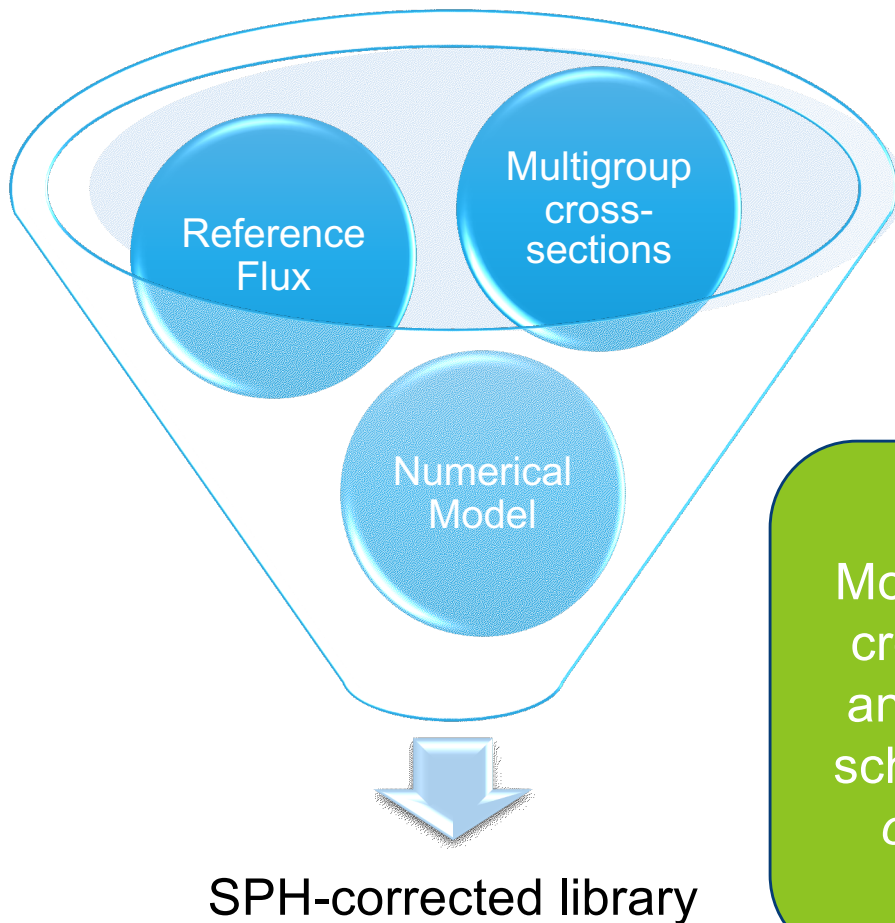
Summary of Steps for Creating Neutronics Model



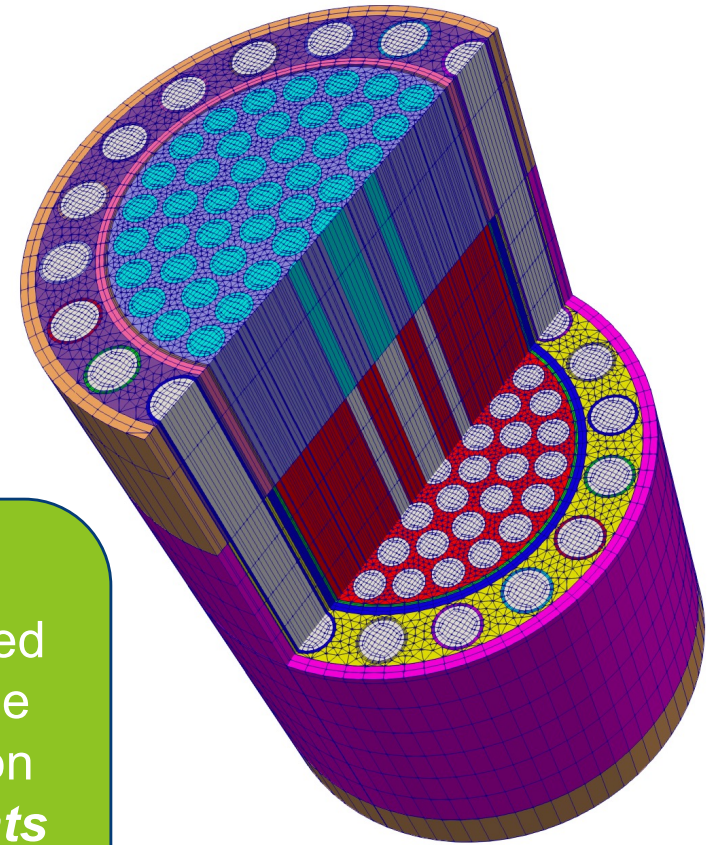
1. Geometry definition is obtained from drawings/concepts
2. A mesh is generated. We recommend CUBIT with NEMO (INL python interface for CUBIT) extension.
3. Serpent model is set up and executed for range of expected temperature and control drum positions.
4. Cross sections are converted using Griffin's *isoxml* tool.
5. Neutronics steady-state/transient models are set up.
6. Cross sections are transport-corrected using SPH treatment

Much of the above is done using automated scripts and a process to ensure that cross sections are mapped to the correct regions in the Griffin model

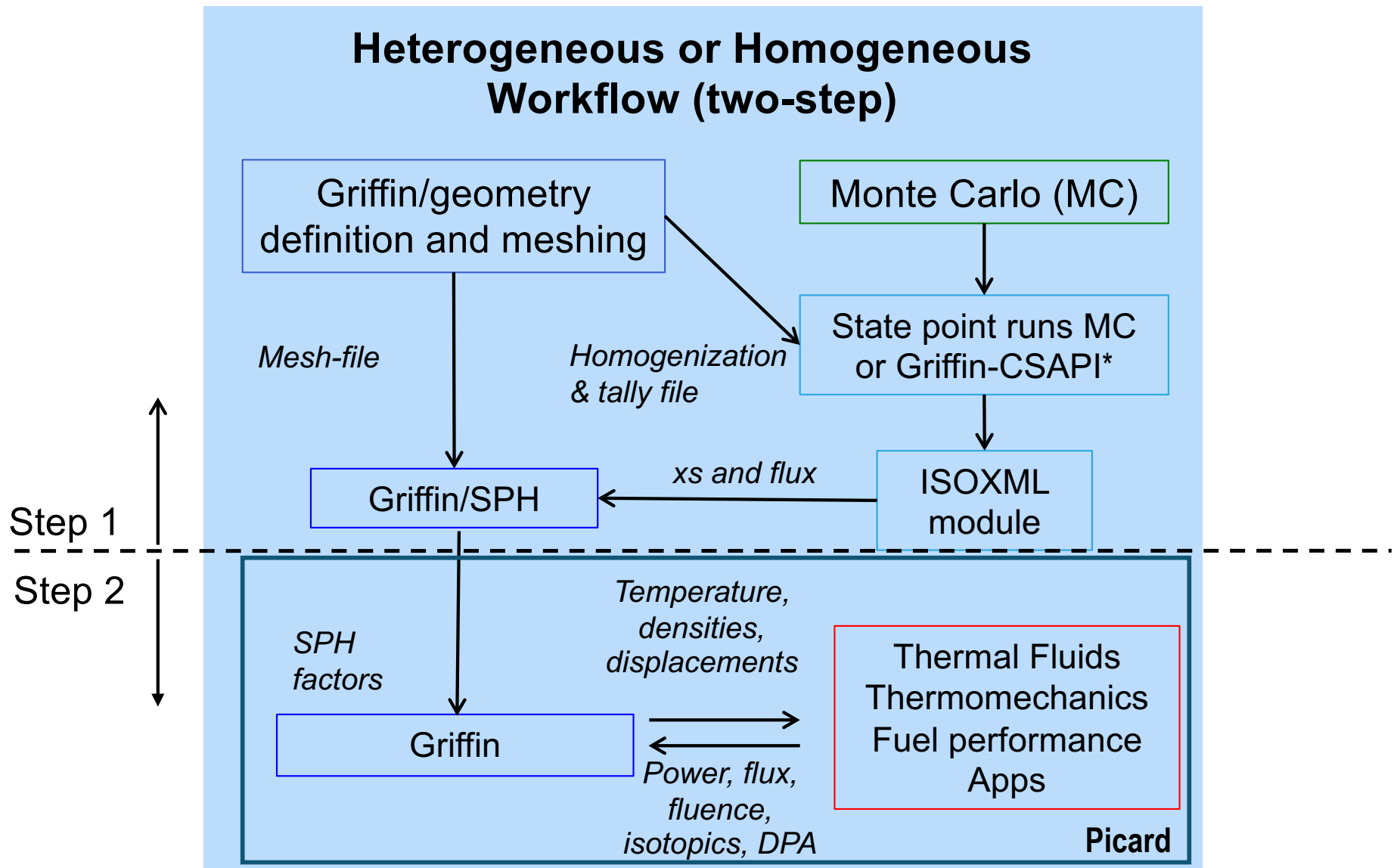
Physics-based Reduced Order Model: Super-Homogenization (SPH) Diffusion Approach



Monte-Carlo corrected cross sections to use an *accurate* diffusion scheme for *transients* on a *coarse mesh*

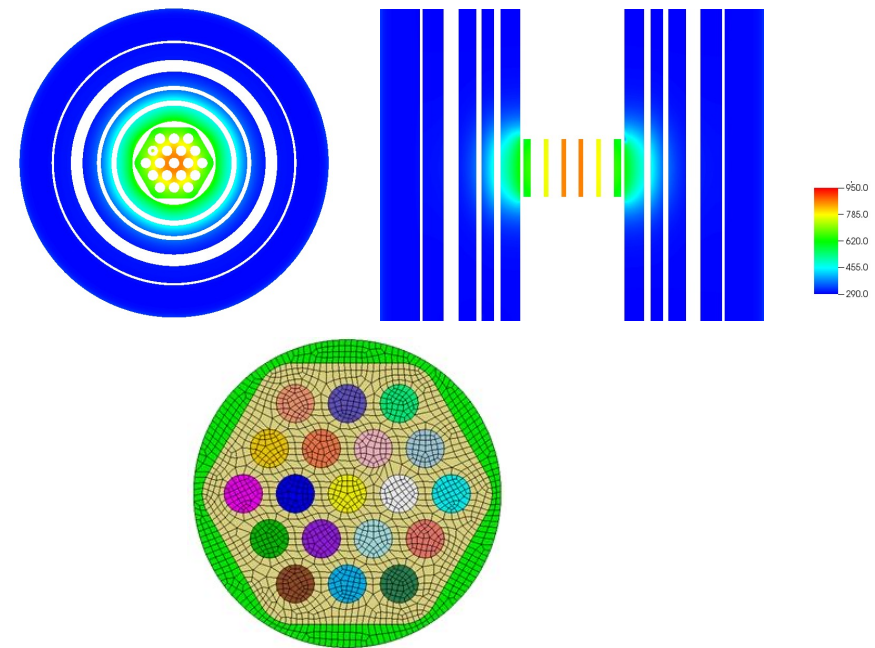
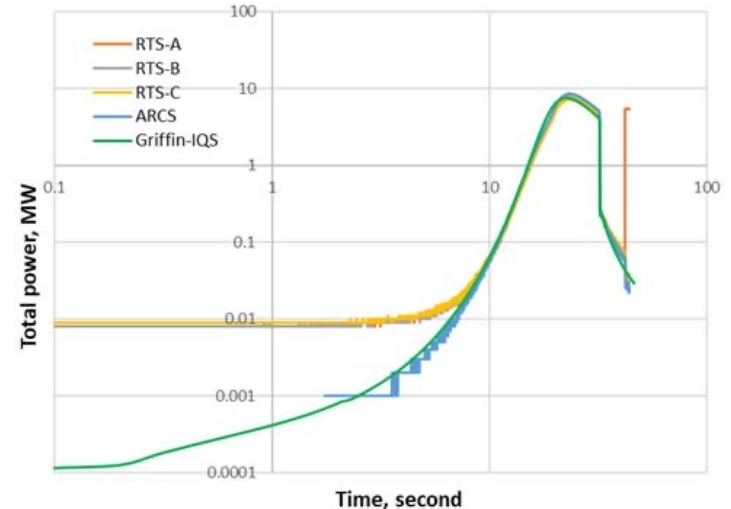


Two-Step Workflow in Griffin



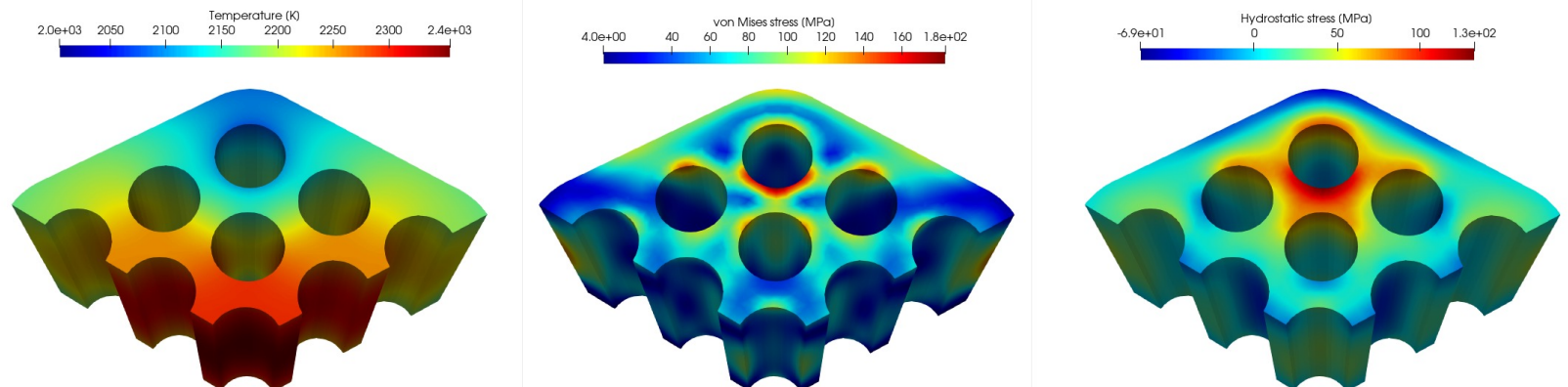
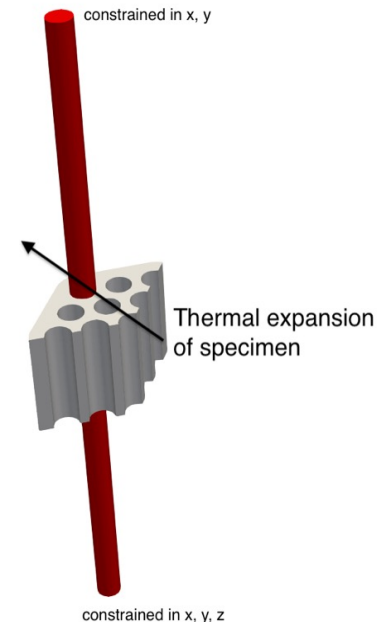
SIRIUS-CAL multi-physics simulation status

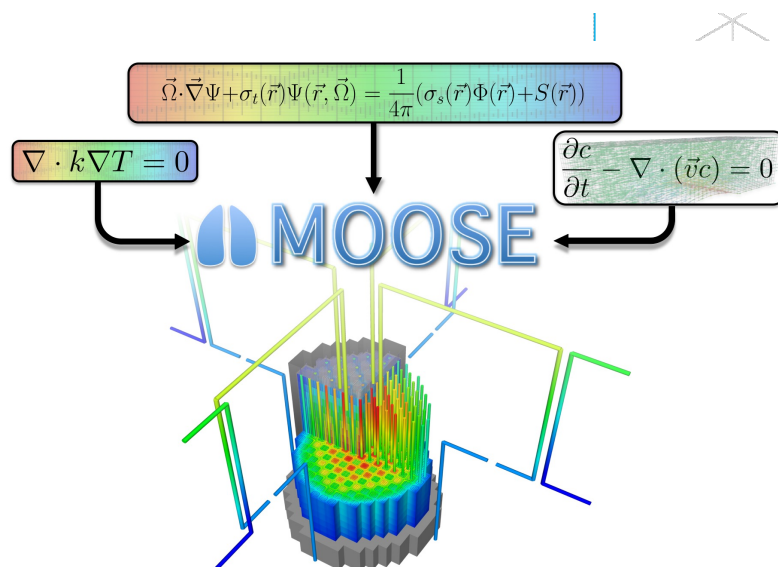
- Neutronics simulation of the TREAT transient with the SIRIUS-CAL experiment was performed to predict the specimen behavior
 - The predicted core power transient agrees well with measured data
- A coupled neutronics and heat conduction simulation was performed to predict the temperature field within the specimen
 - Good estimation of the peak temperature and timing for the fuel specimen
- Work is in progress to improve simulation of the transient temperature and stresses of the specimen.



SIRIUS-1 Temperature/Stress Distribution

- A preliminary mechanical model of the SIRIUS-1 fuel sample has been created.
- Model includes conduction, convection and radiative heat transfer using a flat power distribution from earlier work
- The simulation indicates that thermal expansion of the specimen brings it into contact with the rods
- The contact area experiences stresses of about 170 MPa (in tension) and temperatures of about 1900 K before the transient power peak is reached.
- The figure below shows current results. At $t \approx 27$ seconds, the temperature at the contact point is 2107 K and the von Mises and hydrostatic stresses are 183 and 133 MPa, respectively.





May 26, 2021

Multiphysics Modeling/Simulation Capabilities Based on MOOSE

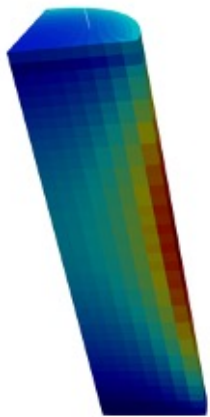
Presented at the NASA Nuclear Thermal Propulsion Modeling/Simulation Update

Presented by Mark DeHart, Sebastian Schunert and Vincent Labouré

SNP Full-Core Coupled Physics

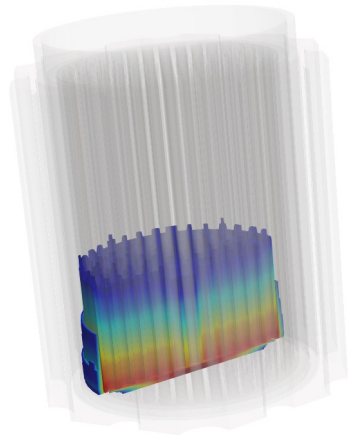
Concurrent solution in steady-state & transient

Neutronics

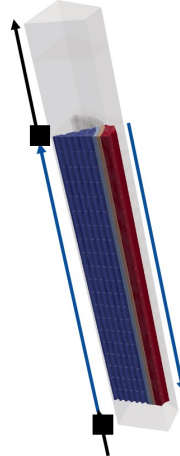


Power Density (unscaled)
4.6e+03
4000
3500
3000
2500
2000
1500
1000
4.6e+02

Heat Conduction



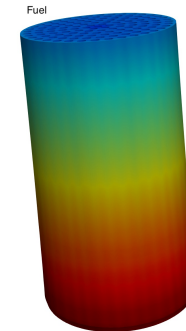
Thermal Hydraulics



Load data & run separately

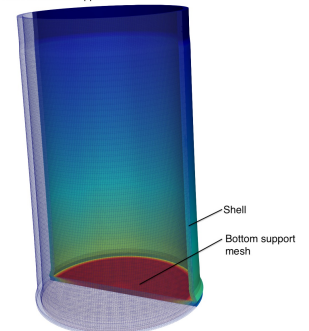
Mechanics

Scaled by 0.1 in axial direction
Displacement exaggerated by factor of 5



Displacement
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.001

Shell & Bottom support



How do we accomplish this with MOOSE-based tools?

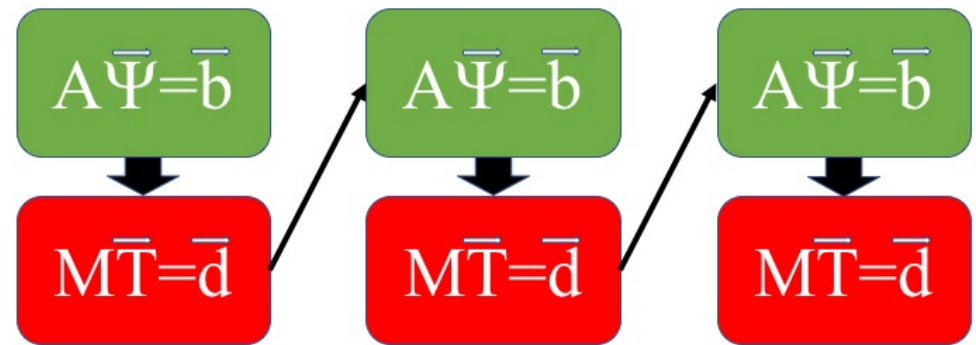
Coupling Approaches

Strongly coupled

$$\begin{pmatrix} A & \alpha \\ \beta & M \end{pmatrix} \begin{pmatrix} \vec{\Psi} \\ \vec{T} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{d} \end{pmatrix}$$

- Ideal to efficiently converge (off-diagonal terms easily included in Jacobian)
- Assumes comparable scales

Tightly coupled

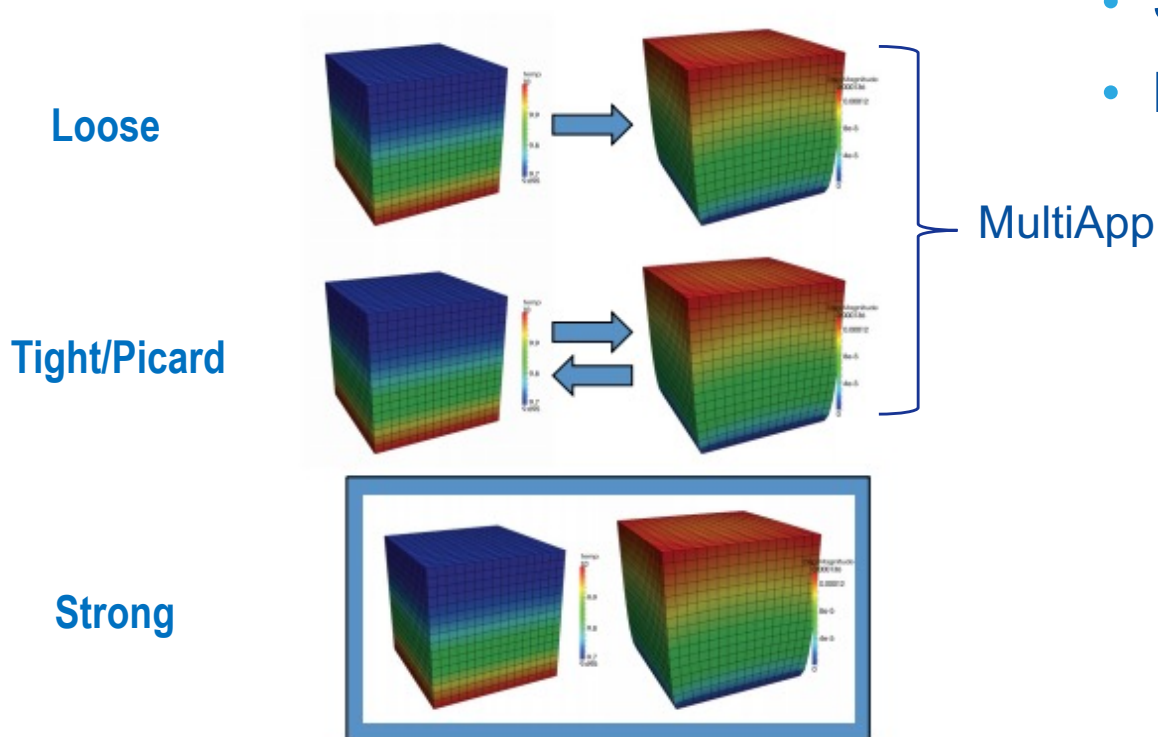


- Picard iterations may converge slowly
- Much greater flexibility when using intrinsically different tools (discretization, time scales, etc...)

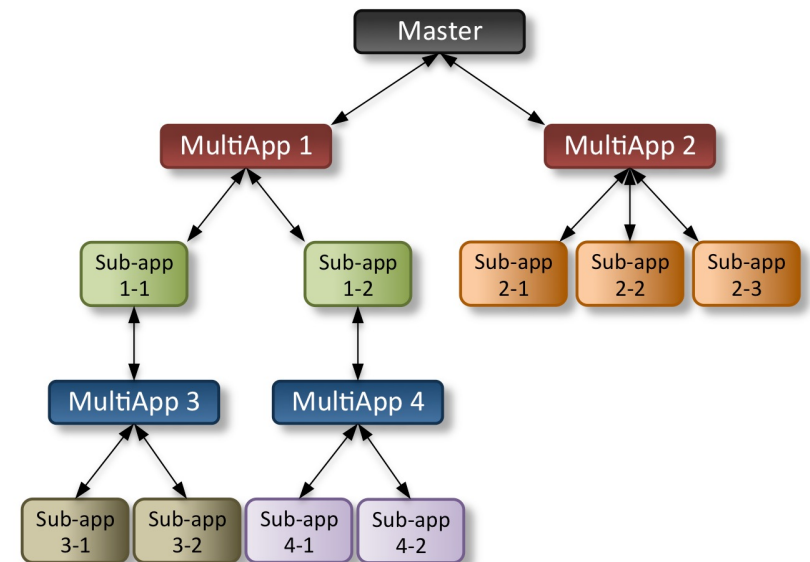
*Which one is more efficient?
It all depends...*

Flexibility by MultiApps

- MOOSE supports loose coupling (operator split) & Picard via MultiApps
- MOOSE supports strongly coupled solves

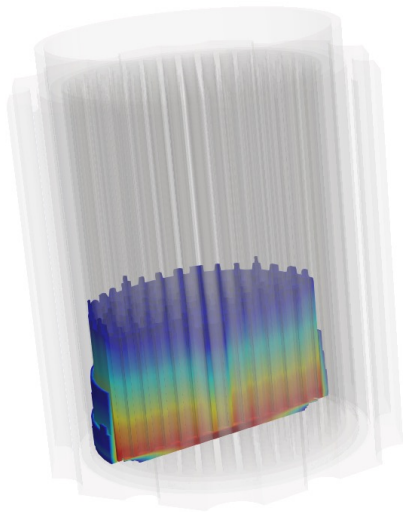


- Picard Iterations via MultiApp
- Master app owns a sub-app
- Recursive: sub-app can own its own sub-app tree
- Information transfer via flexible MOOSE transfers
- Different meshes & dimensionality
- Sub-cycling
- Mixing eigenvalue & transients



Split between Heat Conduction & Mechanics

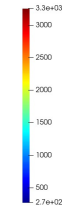
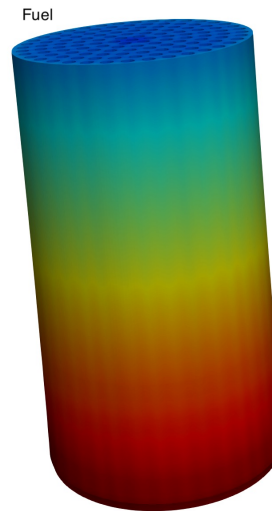
Step 1: Solve heat conduction



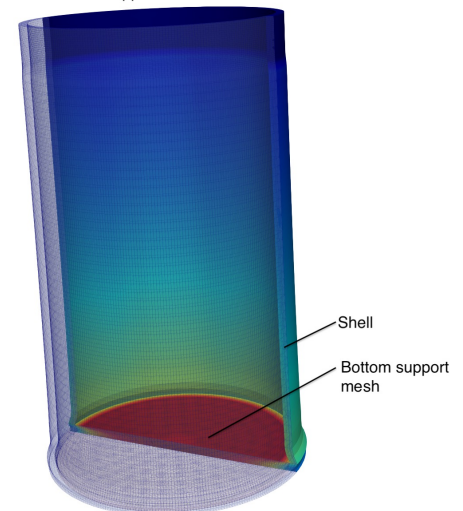
Load temperature
from output exodus file

Step 2: Solve mechanics

Scaled by 0.1 in axial direction
Displacement exaggerated by factor of 5



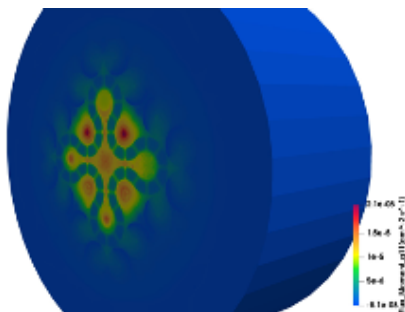
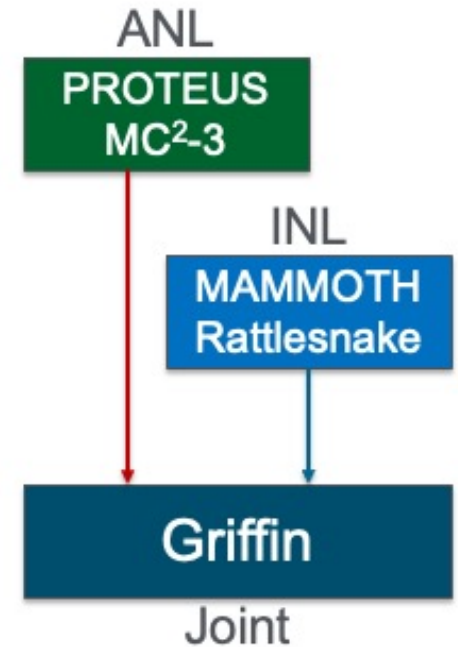
Shell & Bottom support



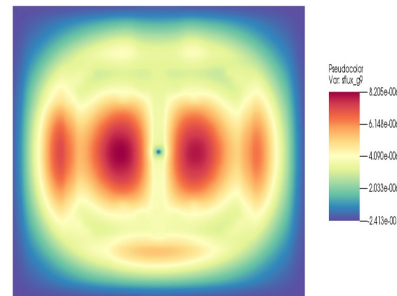
Both models are **pseudo-transient**. We only care about last time step solution! (extension to transients is straight-forward)

Griffin Radiation Transport code

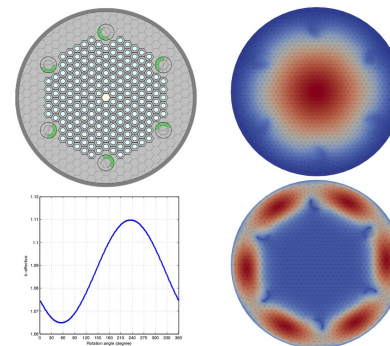
- Griffin is a joint effort of Argonne and Idaho National Labs
- Solves multigroup eigenvalue/transient neutron transport equations with **multi-physics** feedbacks
- Griffin is **different** from other transport solvers:
 - Built for multiphysics
 - Flexible geometry description
 - Built for transient analysis
- Cross section interface with Serpent & inline cross sections
- Discretization options:
 - First order, S_N transport (explicit geometry, high fidelity)
 - Diffusion with homogenization equivalence (workhorse)
 - Variational nodal method is under development



Advanced Test Reactor



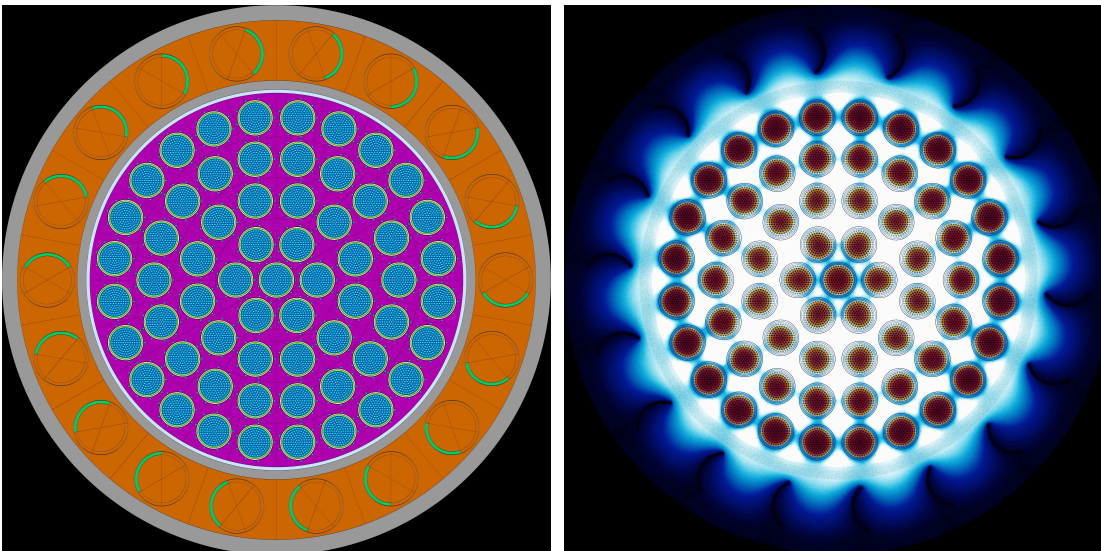
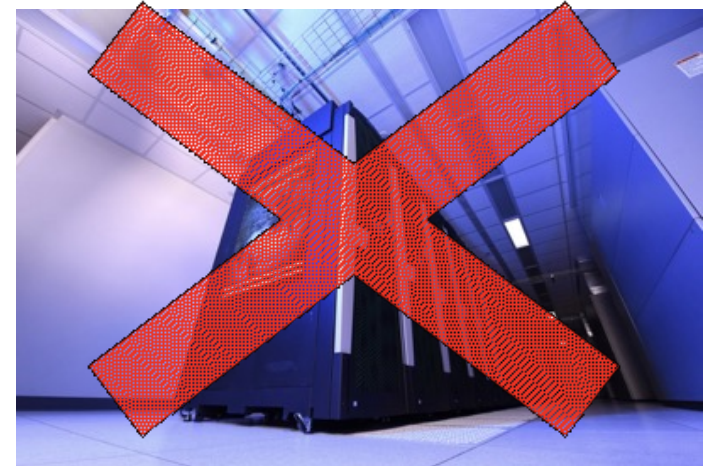
Thermal flux (10th group) distribution,
z-plane =125 cm, SIRIUS-CAL



Control Drum Modeling
and micro-reactors

How to approach large problems and/or limited resources?

- Fully-heterogeneous approach available in Griffin (high-fidelity transport schemes)
- If very large problems and limited computational resources → need a different approach



Available options in Griffin:

1. Homogenization equivalence:
Super Homogenization (SPH) or discontinuity factors
2. Multi-scheme
3. Both

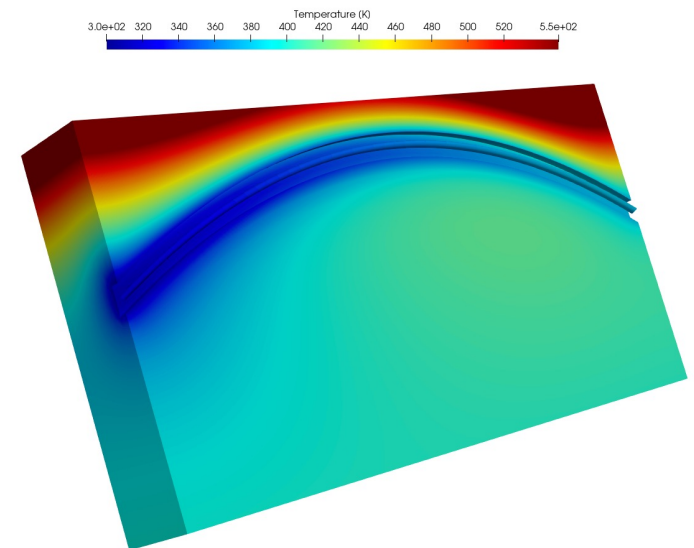
Thermal-hydraulics Module (THM)

THM is a MOOSE based thermal-hydraulics code

- **Features**

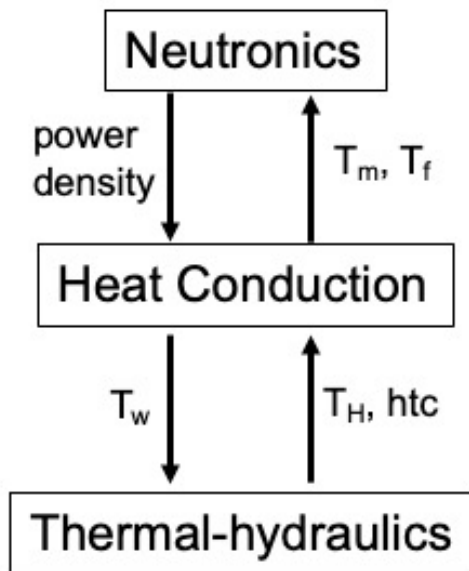
- A LEGO-like approach to building models
- **1-phase**, variable-area, inviscid, compressible flow
- **Conjugate heat transfer**
- Pumps, turbines, valves
- **Coupling** to external MOOSE-based codes

Example of conjugate heat transfer between 3D block and bent pipe

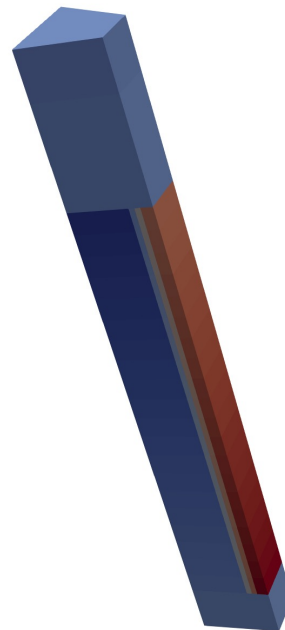


Fuel Element Modeling - Overview

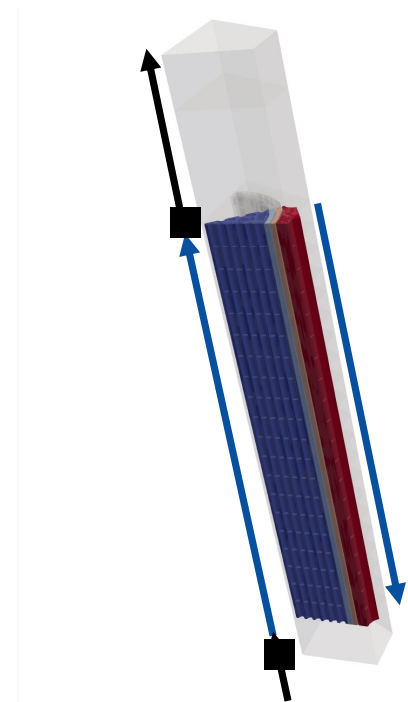
- Neutronics geometry: 90-degree, fuel + axial reflectors
- Heat conduction: active fuel region, 30-degree
- Thermal-hydraulics: representative fuel and moderator flow channels



Neutronics: primary app

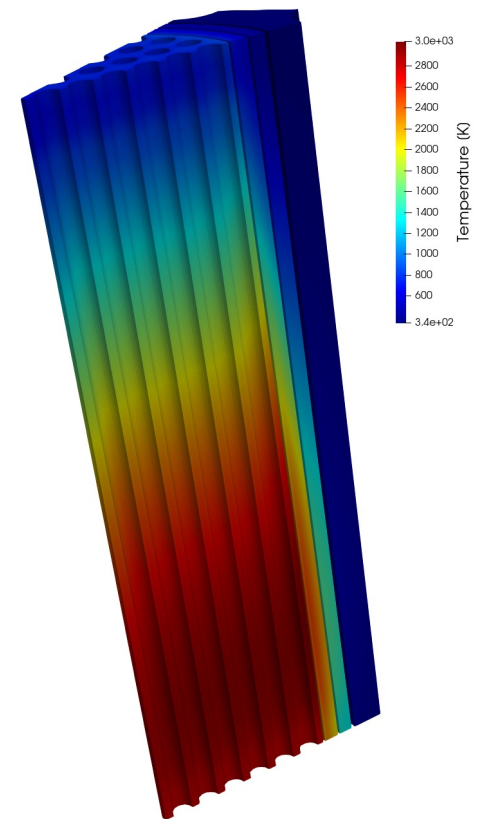
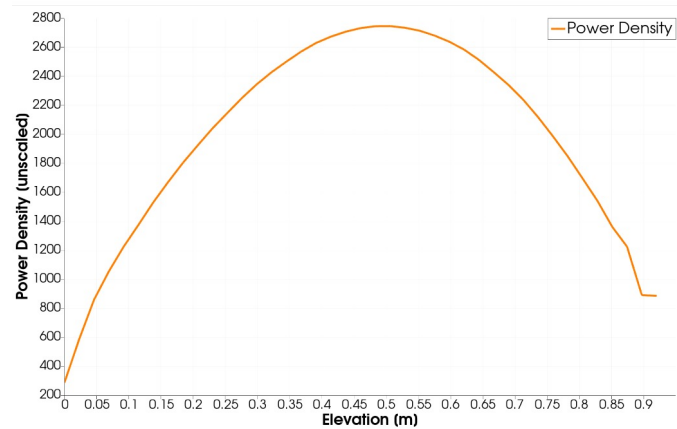
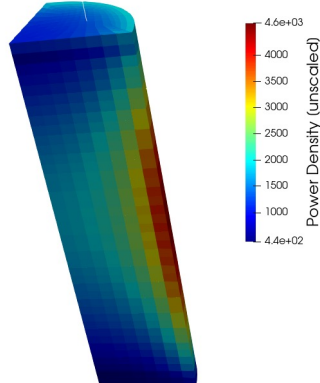


Heat conduction: sub-app 1
TH channels: sub-app 2



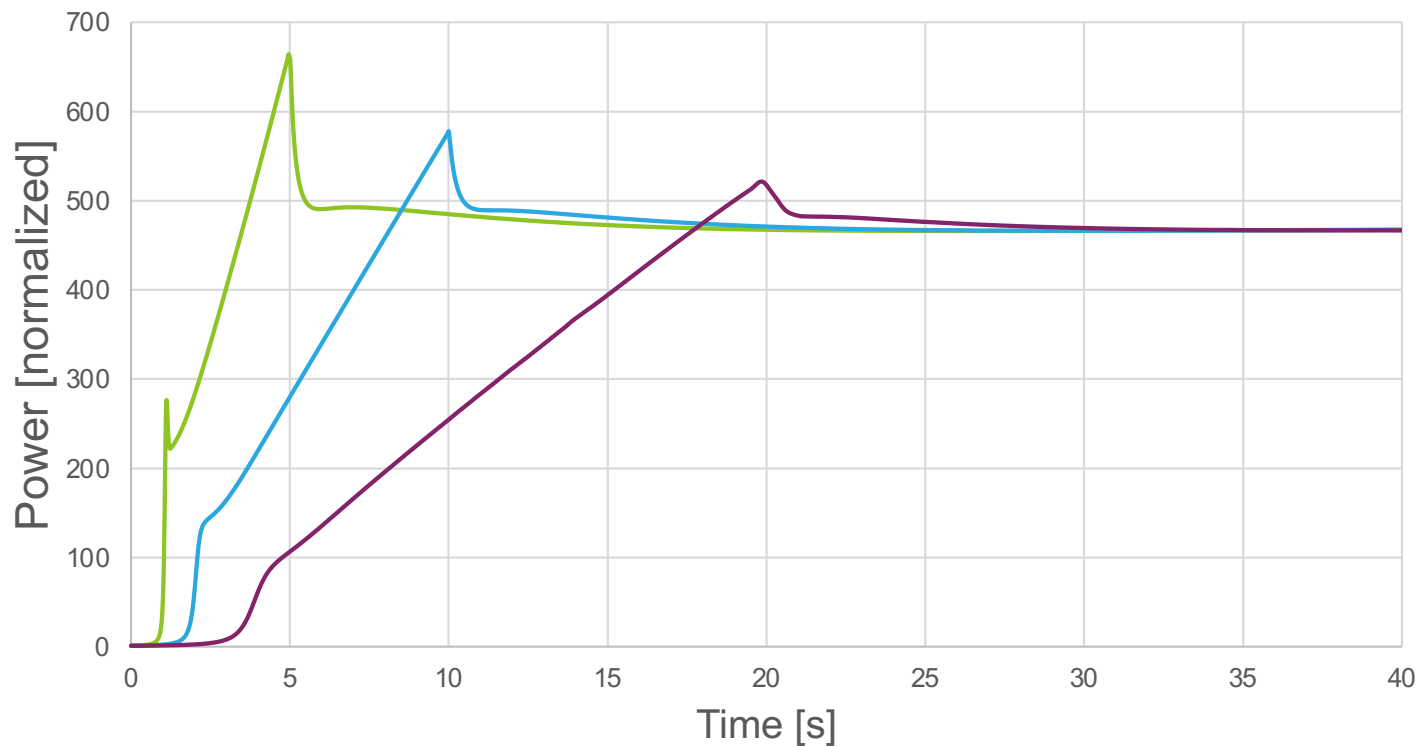
Coupled Steady-state – Results

Quantity	Value
Av. Fuel T [K]	2188
Av. Moderator T [K]	410
Max. T [K]	3033
Outlet T [K]	2656



Reactivity insertion ramps

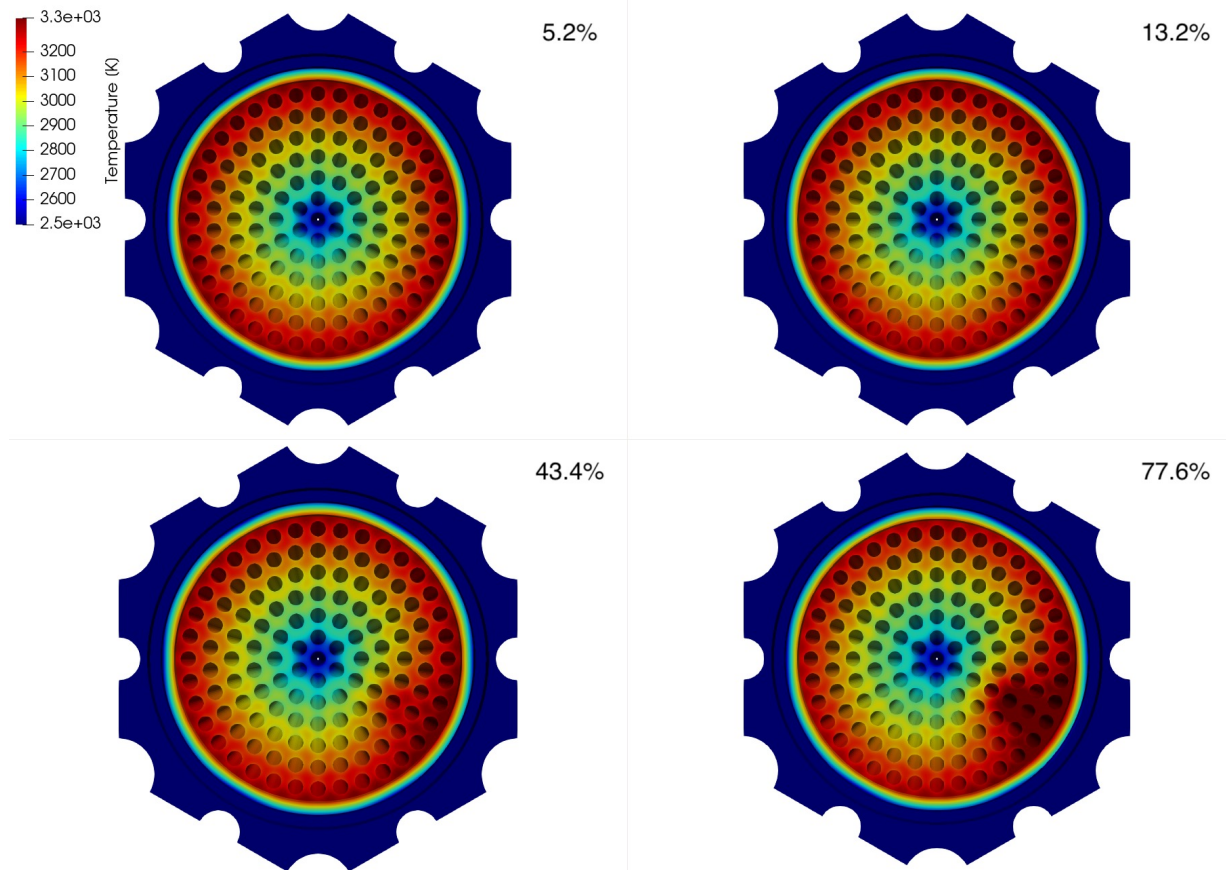
- Demonstration of “startup” transient using the fuel element model
- Linear reactivity insertion ramp (reactivity inserted via boron reduction)
- Linear reactivity insertion from 0 to 5\$ in 5/10/20 seconds
- Power overshoots are more pronounced for fast reactivity insertions



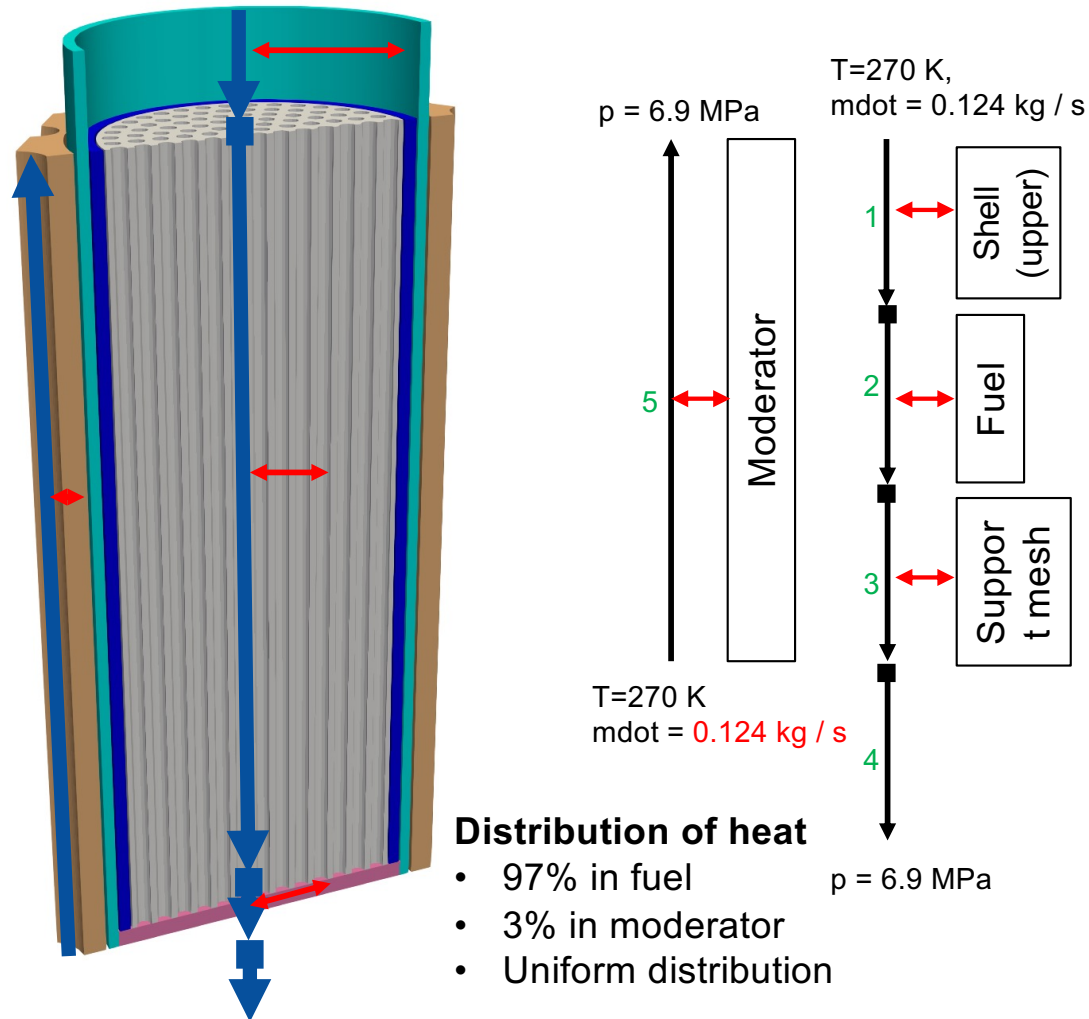
— 5 second ramp — 10 second ramp — 20 second ramp

Extended Fuel Element Model – Flow asymmetry

- Fuel element model with explicit fuel channels
- #THM channels = #physical channels
- Effect of flow rate asymmetry on temperature
- Reduction of flow rate by x% in channel 52
- With 77.6% reduction in channel 52 flow, we increase surrounding temperature by 300 K



Thermal-Mechanics Model



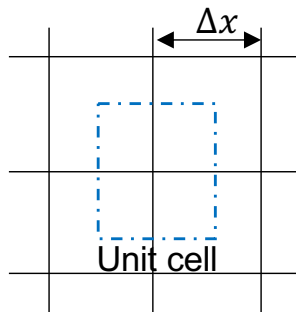
- Single, representative channel for fuel and moderator
- Channels are:
 1. channel_in
 2. channel_fuel
 3. ch_NbC
 4. channel_out
 5. channel_moderator
- Conjugate heat transfer:
 - channel_in & upper part of shell
 - channel_fuel & fuel meat
 - ch_NbC & bottom support mesh
 - channel_moderator & moderator

Modeling heat transfer in the bottom support mesh

- Fluid model: Model as a porous flow region:
 - Porosity: $\epsilon = \text{fluid volume} / \text{total volume} = 0.72$
 - Heat transfer area per volume = ϕ
 - Flow area = geometric area * ϵ
 - Wetted perimeter/heated perimeter = geometric area * ϕ
 - Hydraulic diameter: $D = \frac{4\epsilon}{\phi}$
 - Heat transfer coefficient set to 10,000 W / m² K (lack of correlation)
- Conjugate heat transfer: heat transfer with average solid temperature: $q''' = \phi h(\bar{T}_f - \bar{T}_s)$

How do you compute ϕ ?
(Example)

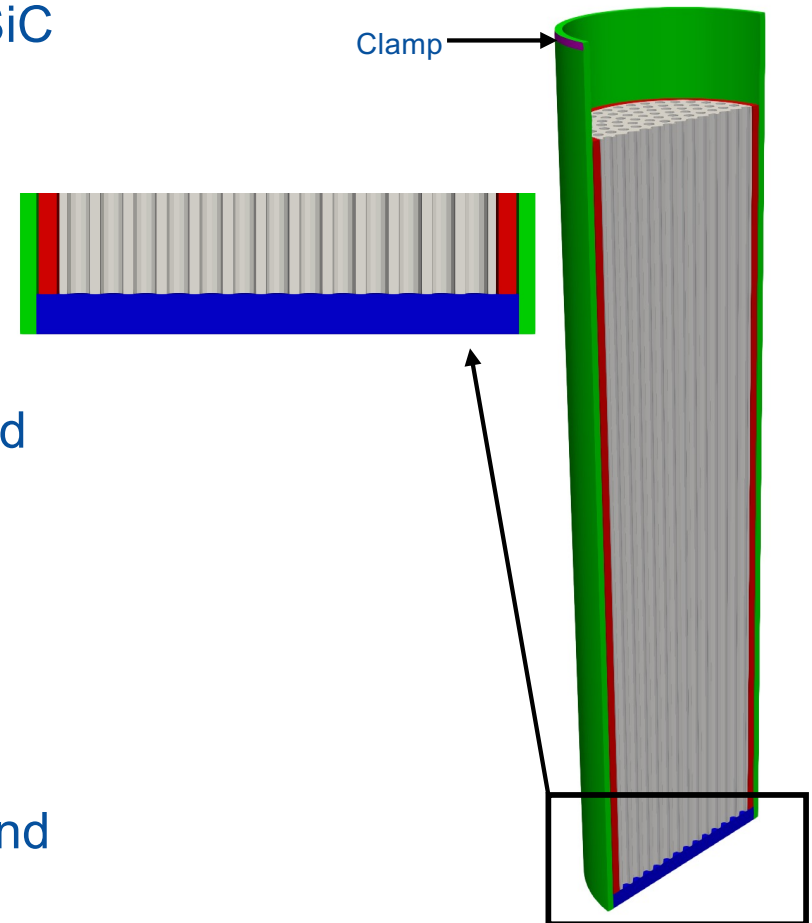
1. Assume simple cubic structure
2. Constant mesh width Δx
3. Circular wire with fixed diameter

$$\phi = \frac{3\pi d}{\Delta x^2}$$
$$\epsilon = \frac{3\pi}{4} \left(\frac{d}{\Delta x} \right)^2$$


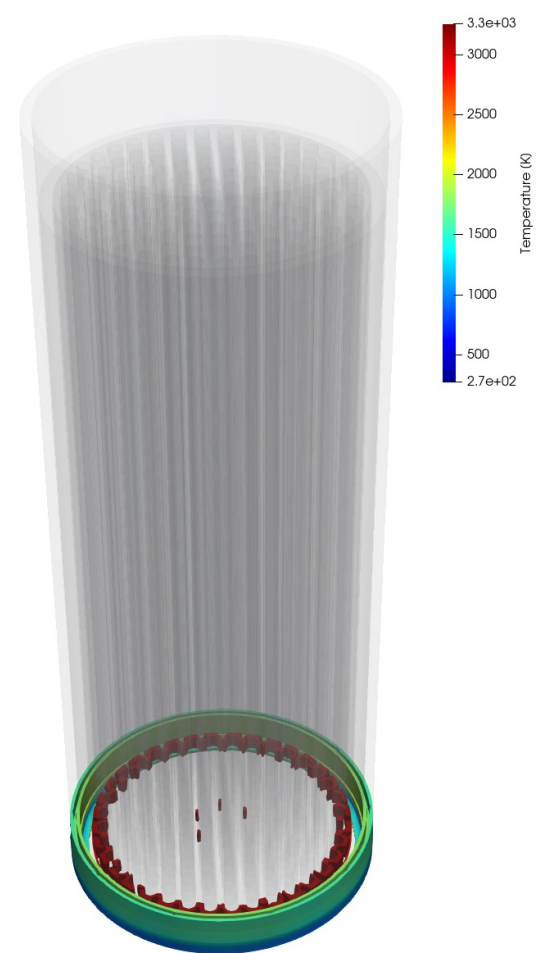
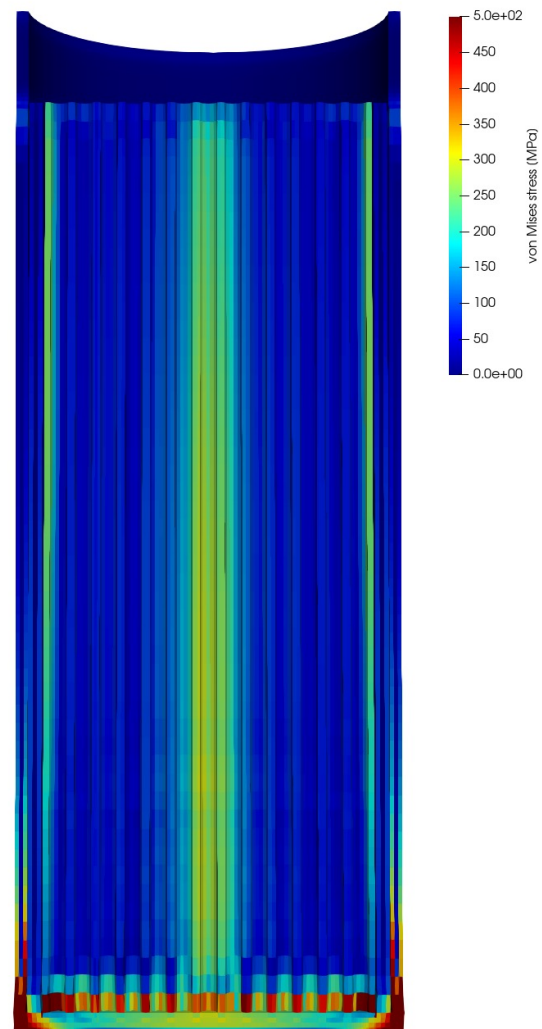
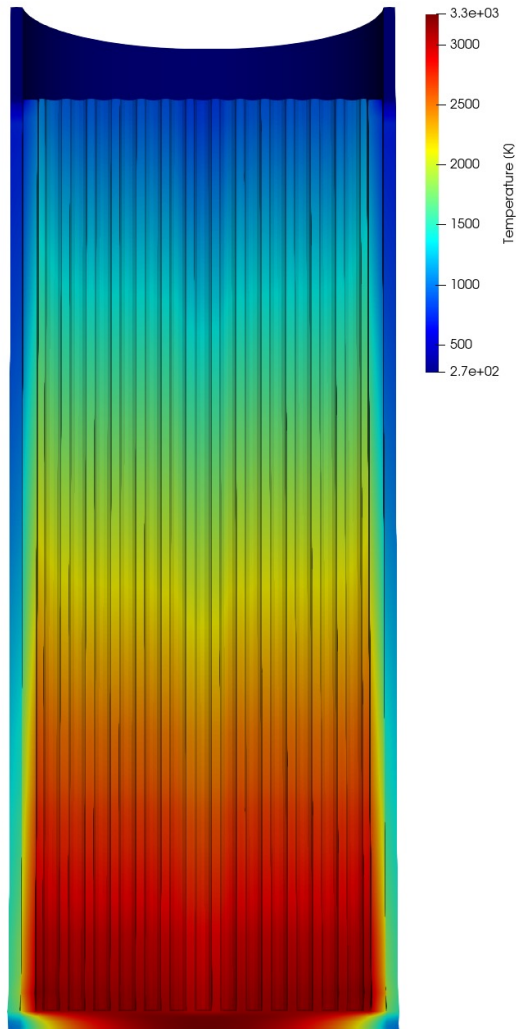
Note: current porosity is not consistent with this mesh morphology!

Mechanics model overview

- The moderator region has been removed because there is no mechanical contact with SiC
- Boundary conditions:
 - Defined a sideset called clamped
 - Enforce no displacement of entire sideset ($disp_x = disp_y = disp_z = 0$)
- Modeling the bottom support mesh
 - Mesh is modeled as a soft material with $E = 5 \times 10^4 \text{ MPa}$ and $\nu = 0.23$ (did not find good data to base this on)
 - Contacts to fuel, insulator, and shell are assumed to be perfectly bonded
- Approximations (we are working on a more representative model):
 - Better boundary conditions
 - Sliding contact between bottom support and structural materials
 - Include top support mesh
 - Mechanical contact between fuel, insulator, shell
 - Inclusion of aft structure?

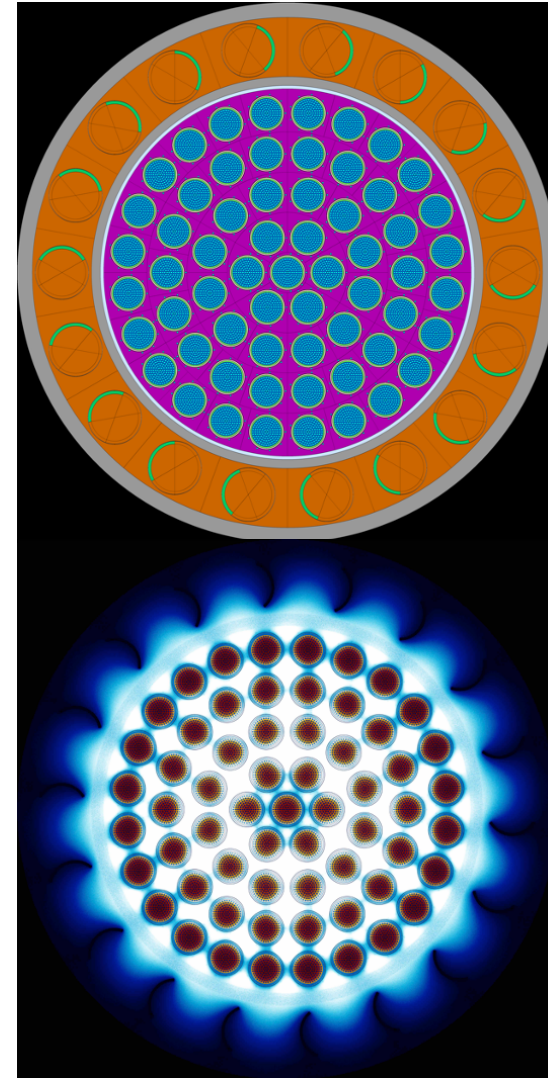


Results of the Mechanics Simulation (Elastic)



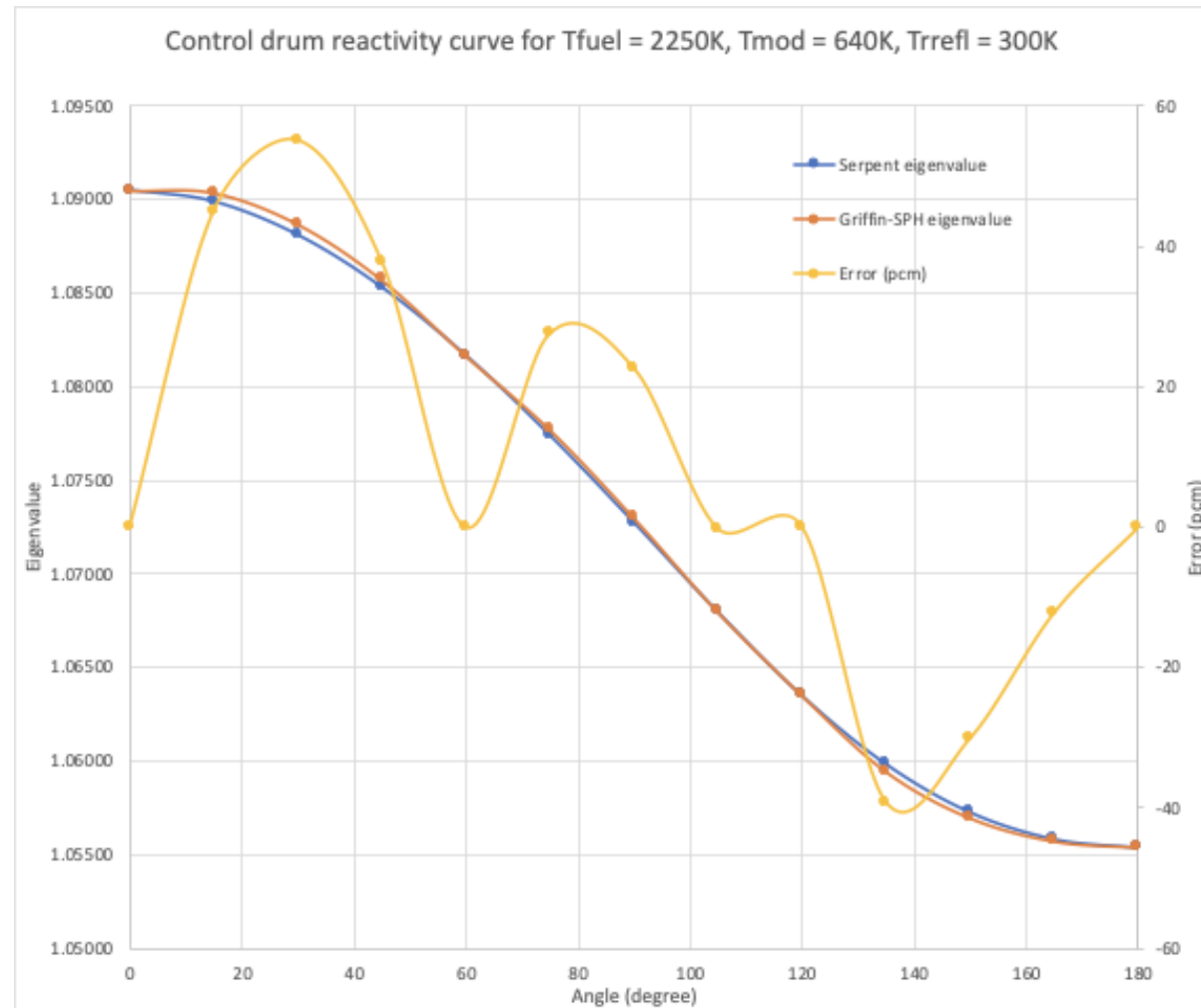
Full-Core Model Overview

- 61 Fuel Elements in 5 rings
- 18 Control Drums in Be reflector to adjust reactivity/power
- For this presentation, all the drums are simultaneously rotated with the same rotation angle
- Griffin allows independent rotation (e.g., for a simulated reactivity insertion accident)

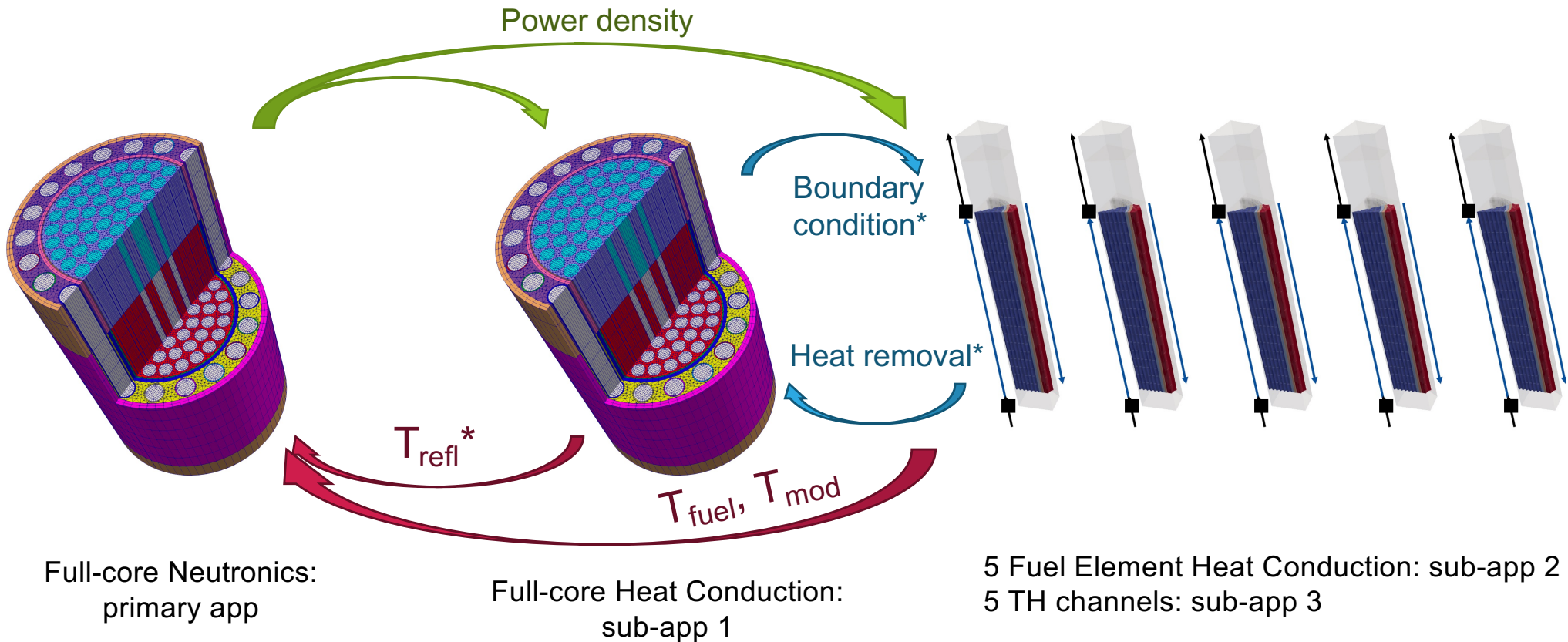


Control Drum Worth

- At state points (0, 60, 120, 180), eigenvalue and power profile exactly reproduced
- Between state points, cusping treatment from Griffin is utilized
- Bias between -40 and +60 pcm
- Single eigenvalue calculation in a few minutes



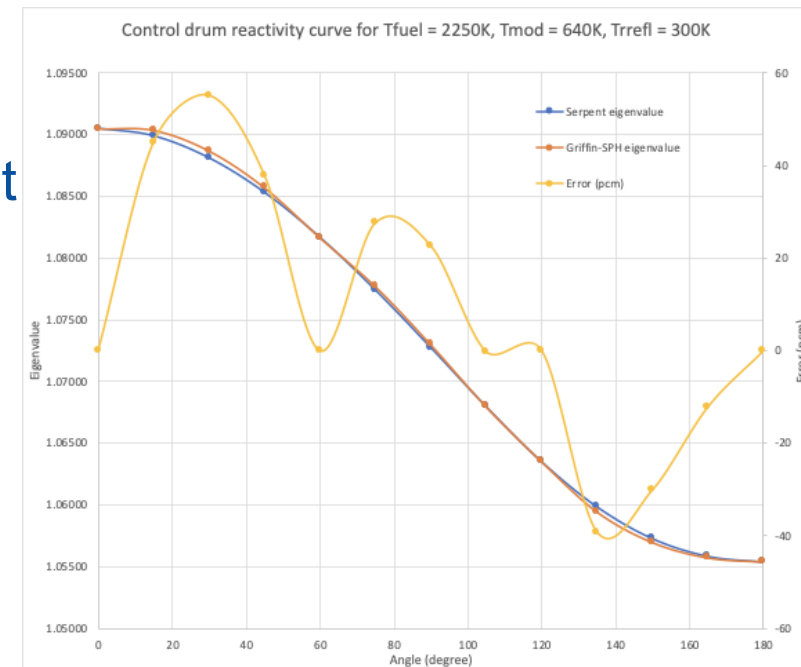
Example – Coupled NTP Full-Core Model



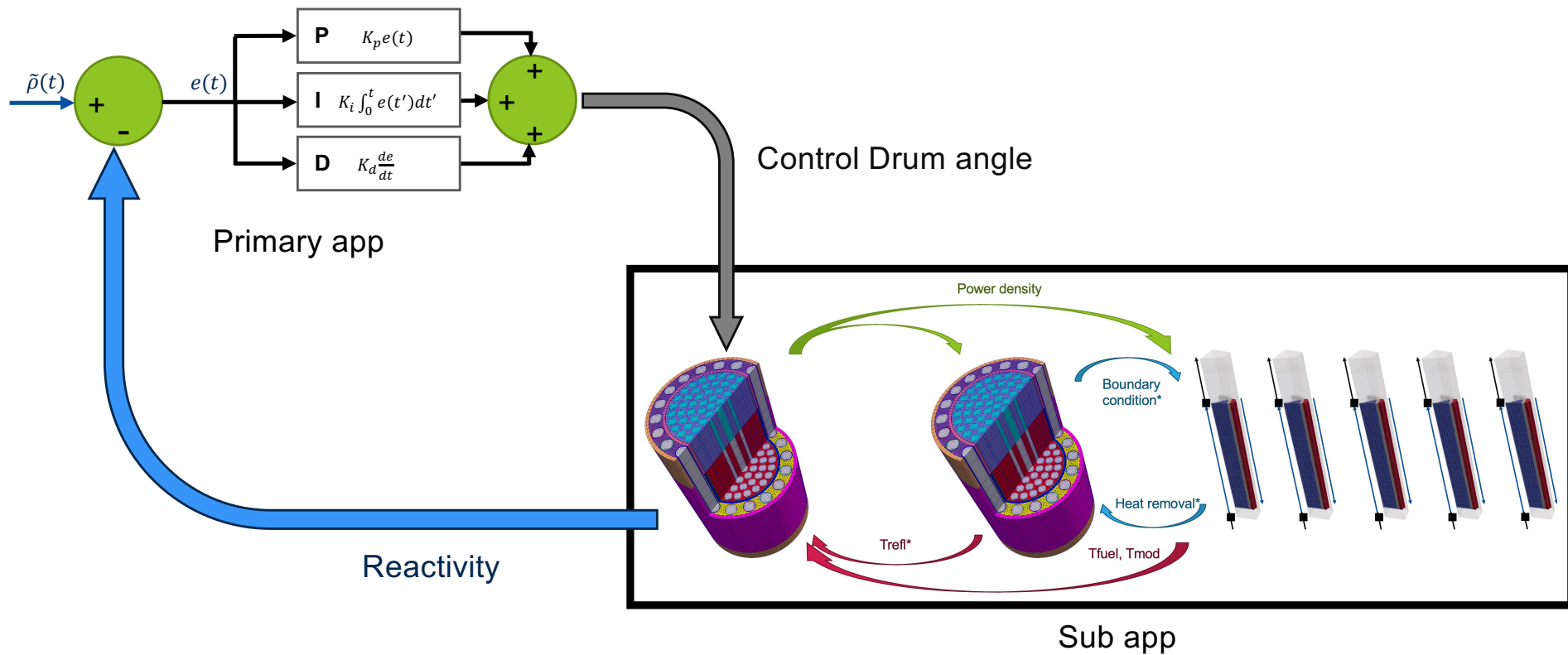
**not in current model*

Start-up Transient Assumptions

- *Goal: go from low power to full power in a few minutes*
- Currently, we assume 100% H₂ flow rate established at the beginning of the transient
- Assume initial temperature:
 - T_{fuel} = 500 K
 - T_{mod} = 270 K
 - T_{rrefl} = 300 K
- Initial power: 610 kW (10 kW/fuel element)
- Initial CD angle $\theta_i = 120^\circ$
- Final CD angle $\theta_f = 60^\circ$
- Drum rotation determined by PID controller



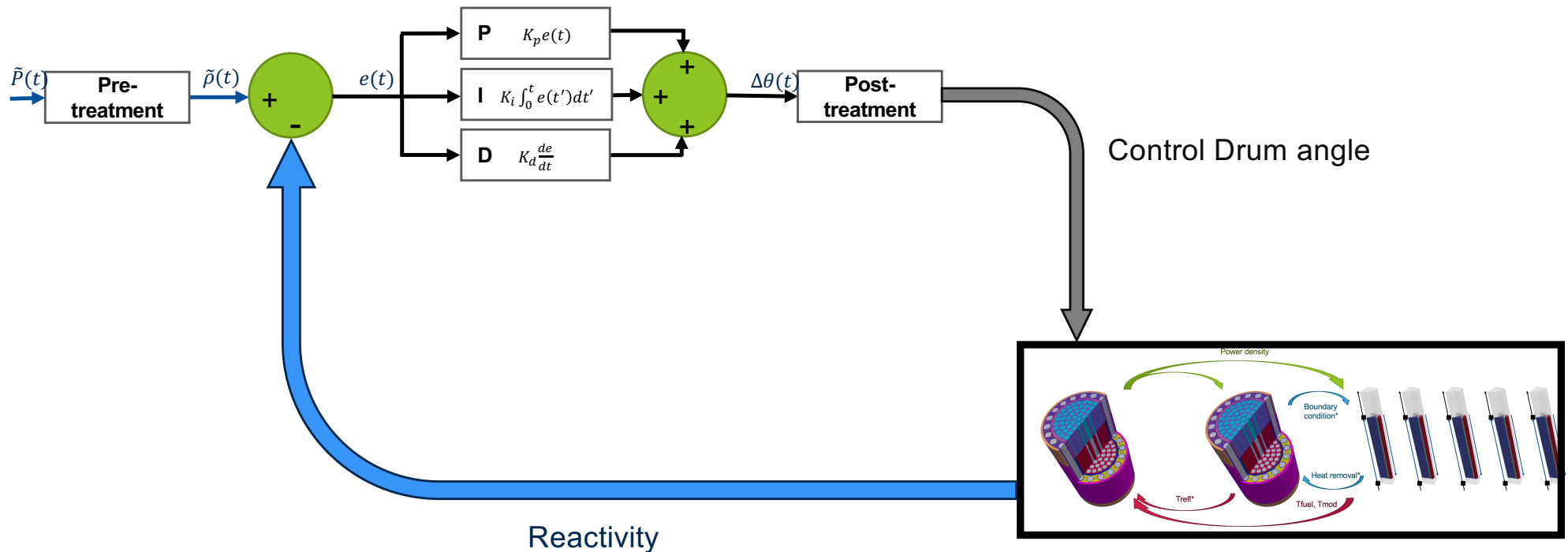
PID Control of Drums



- Requires less trial and error than manual control
- For now, ignores any limitation on drum rotation (speed, etc.)

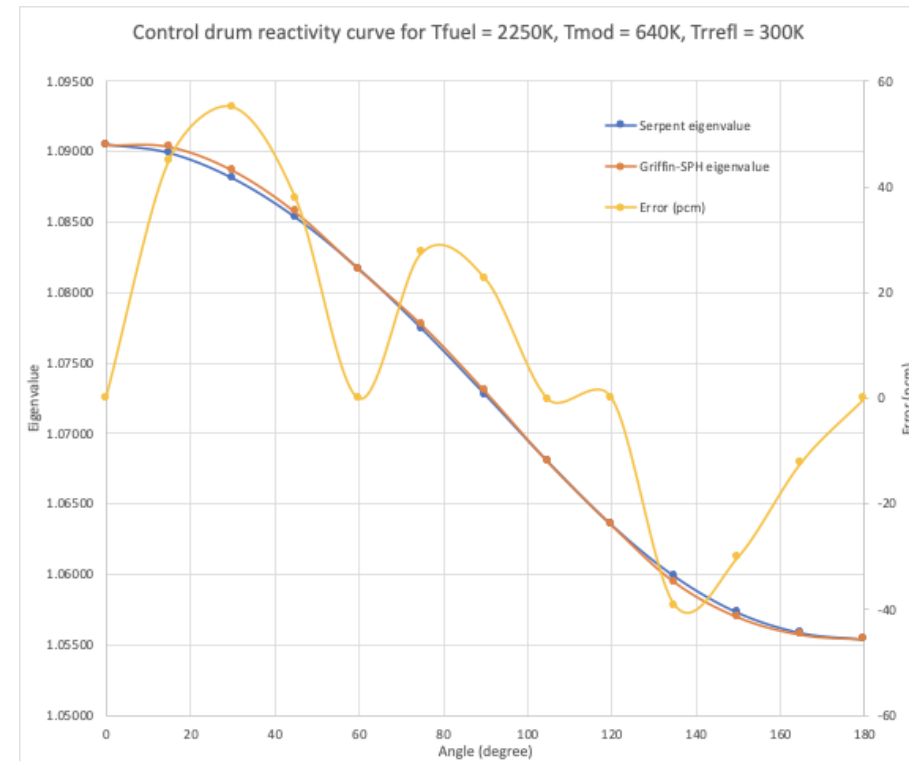
PID Control of Drums (2)

- If desired, could input a power setpoint
- Truncation to make sure the CD angle stay within $[\theta_f, \theta_i]$

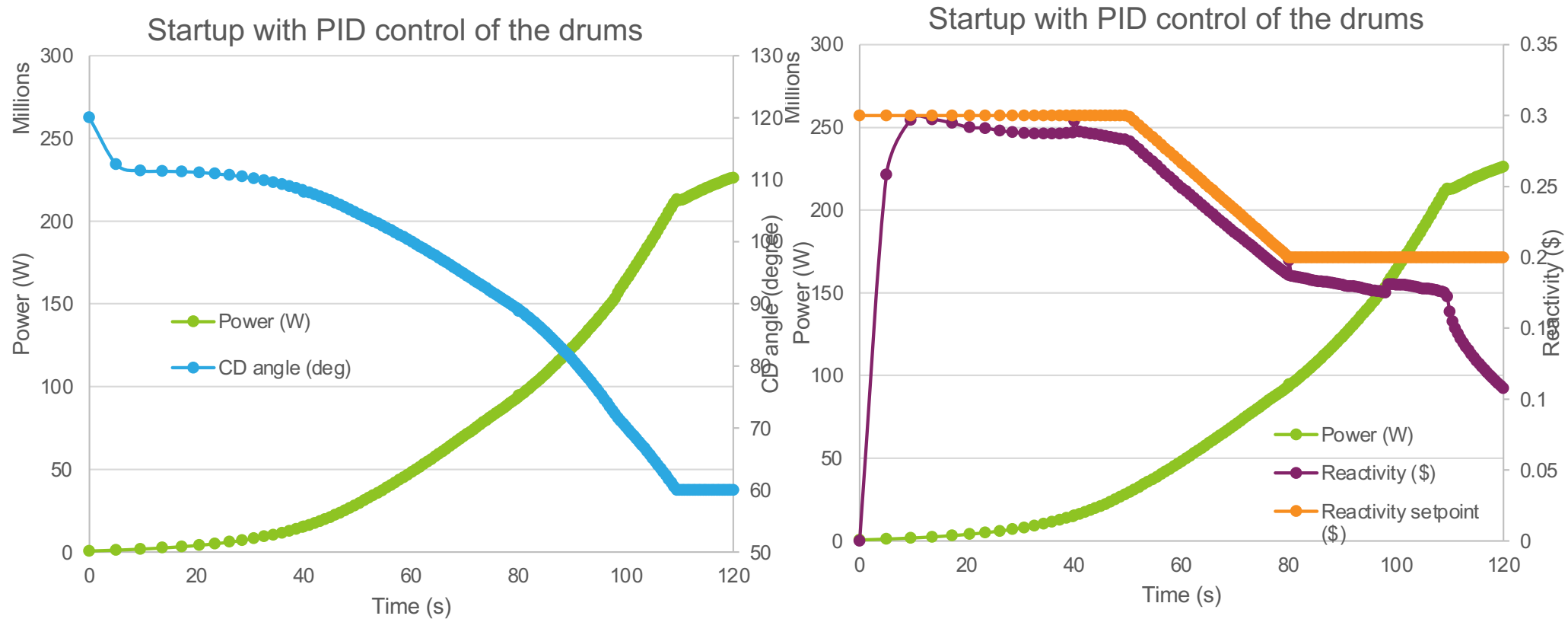


PID: Remarks

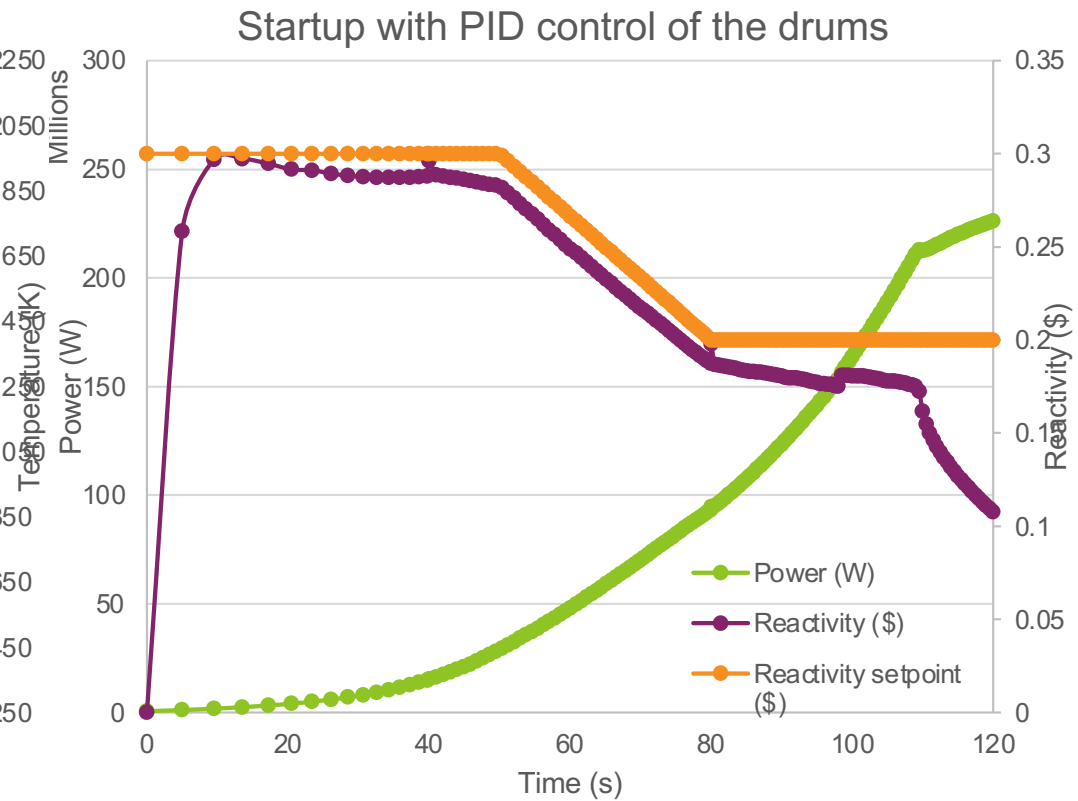
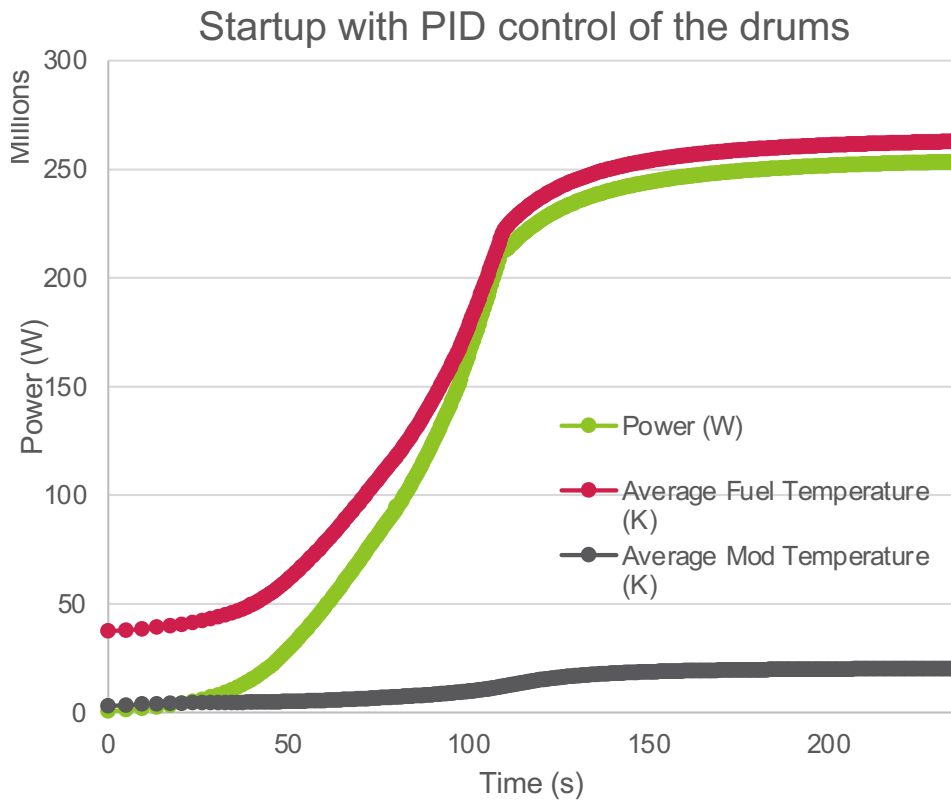
- If only proportional term is included in PID:
$$K_p = \frac{\Delta\theta}{\Delta\rho}$$
- For the CD angle between 60° to 120°, CD worth ~ 25-30 pcm/° or 24-30 °/\$
- $K_p = 25$ °/\$ should be reasonable
- Direct correlation between reactivity and CD angle: this is why we choose to rotate drums based on reactivity and not power (time delay between reactivity and power)



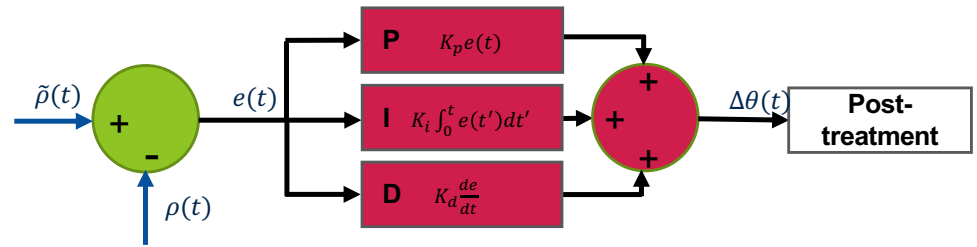
Start-up Transient with PID: $K_p = 25$, $K_i = K_d = 0$



Start-up Transient with PID: $K_p = 25$, $K_i = K_d = 0$



Changing K_i



- Increasing temperature creates negative feedback not captured by 'proportional-only' PID
- Integral term can capture the persistent lag behind the setpoint
- During the early stages the transient,

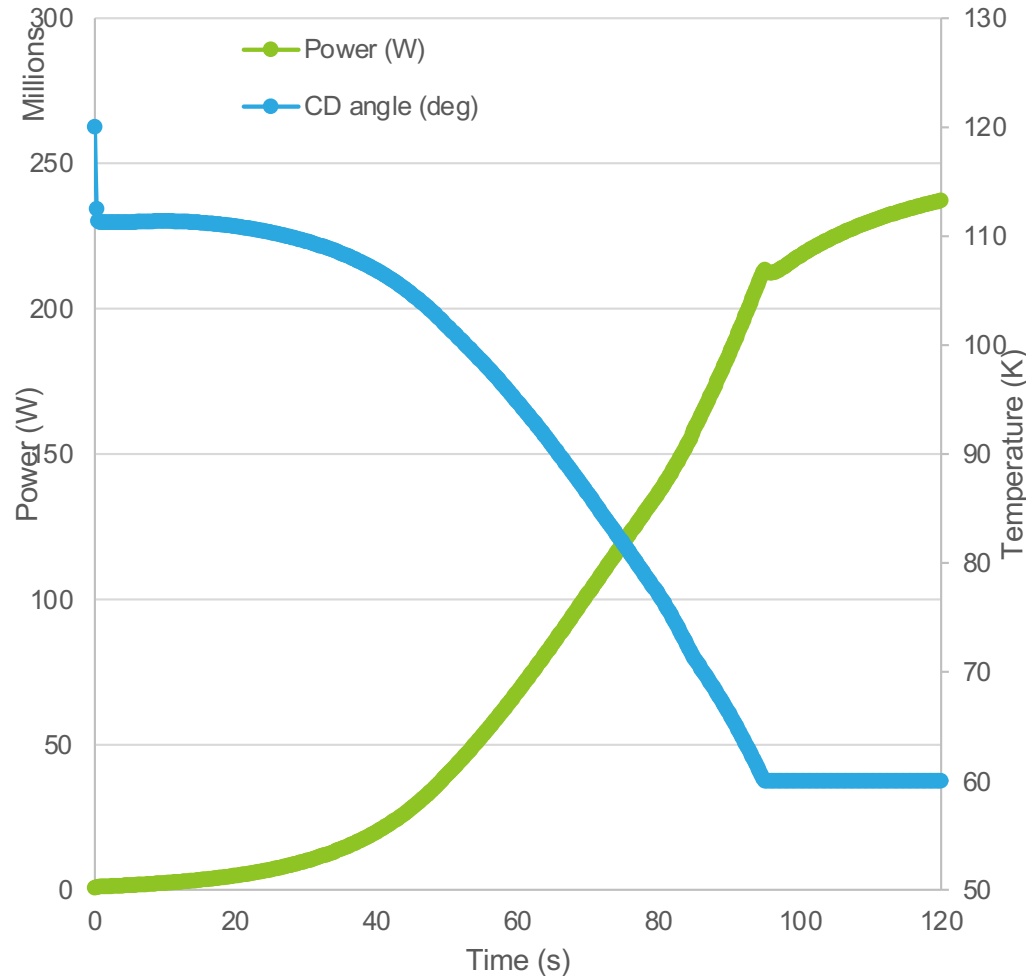
$$\frac{\int_0^t e(\tau) d\tau}{e(t)} \approx 100 - 200 \text{ s}$$

- Thus, we pick:

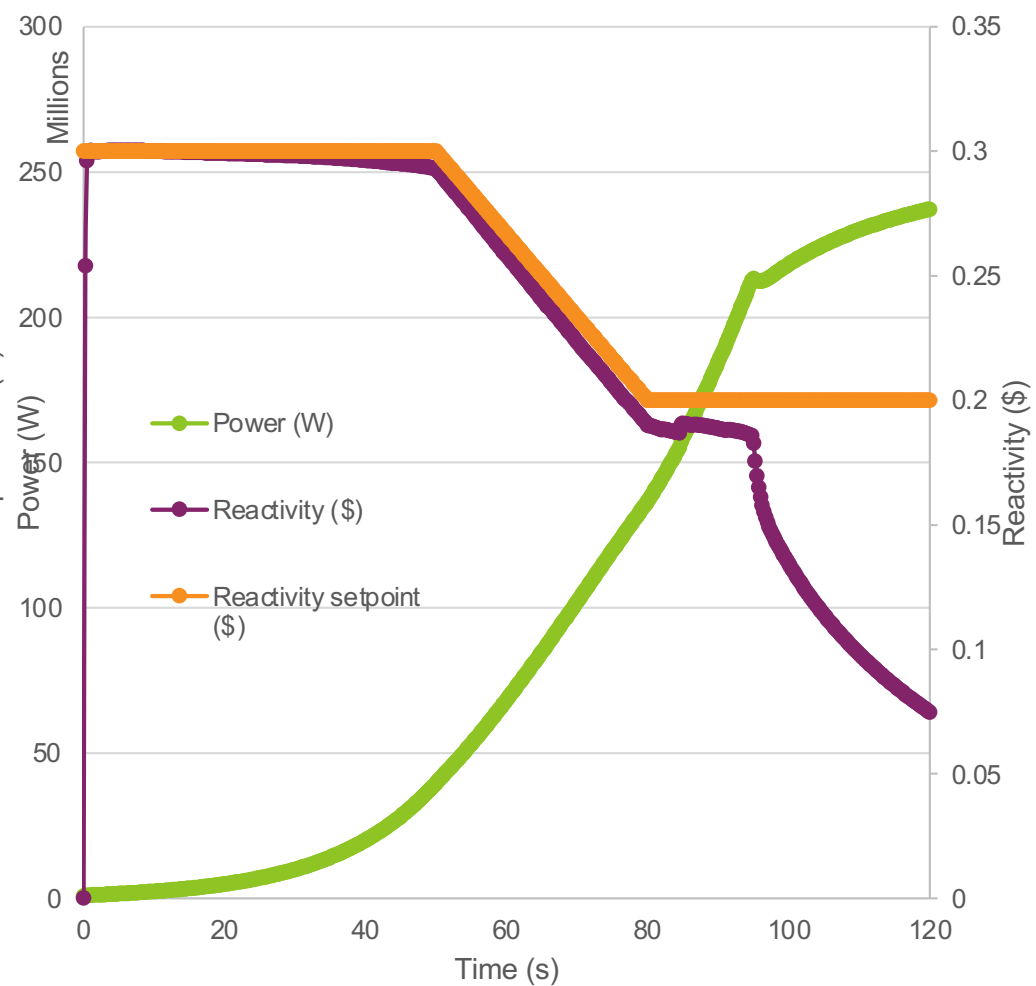
$$K_i = \frac{K_p}{100} = 0.25 \text{ } ^\circ/(\text{\$-s})$$

Start-up Transient with PID: $K_p = 25$, $K_i = 0.25$, $K_d = 0$

Startup with PID control of the drums

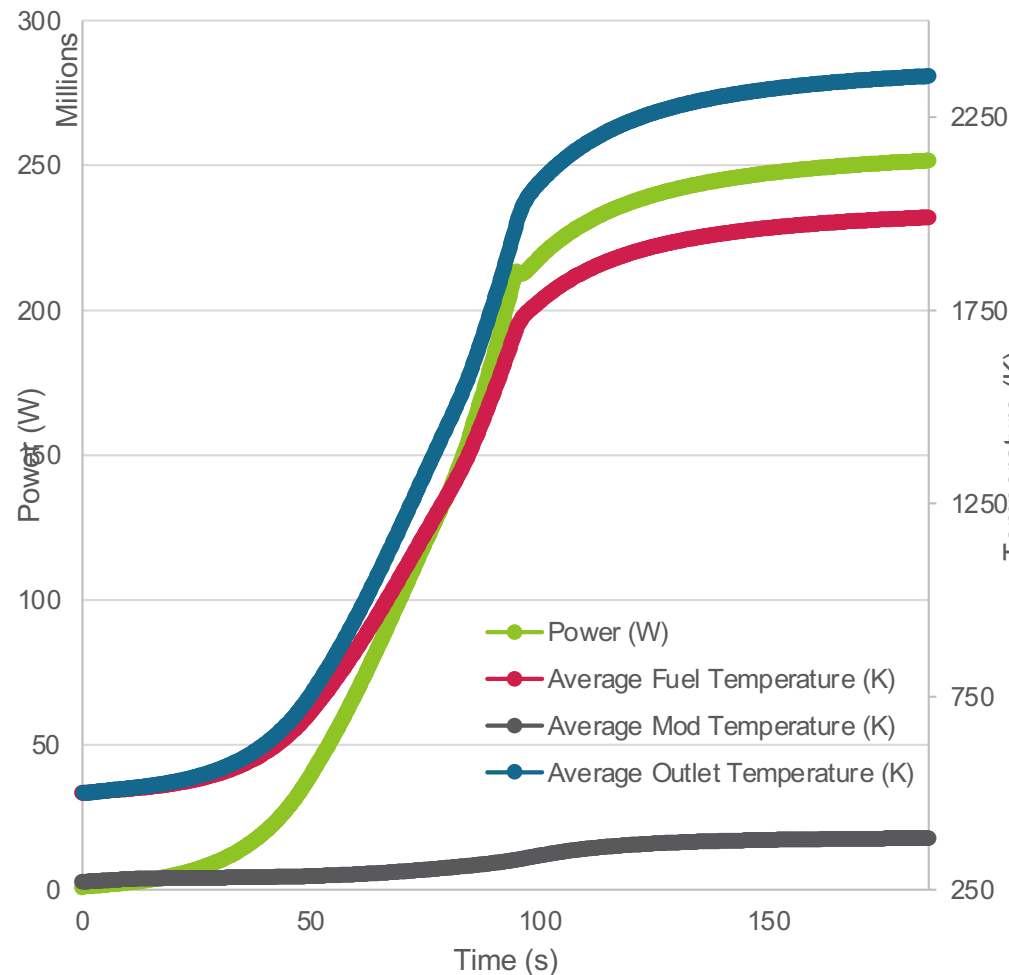


Startup with PID control of the drums

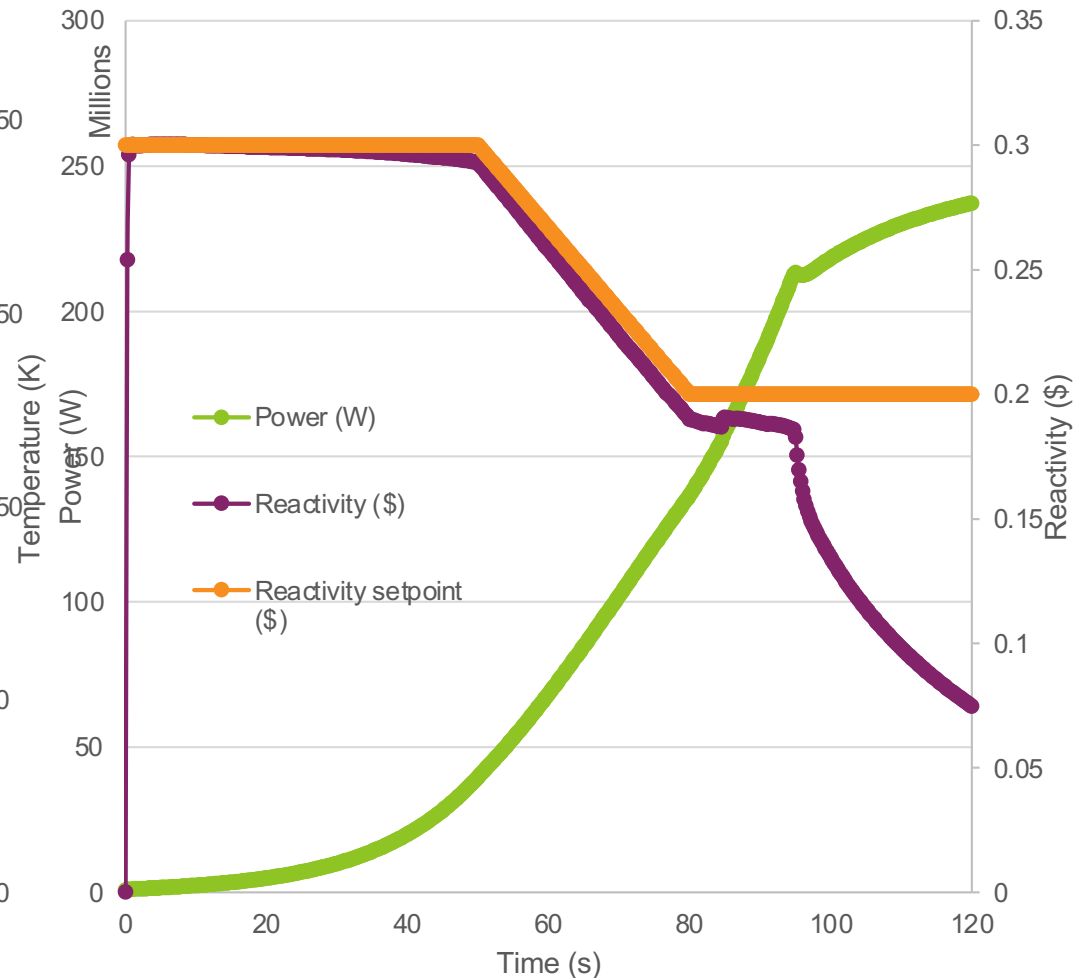


Start-up Transient with PID: $K_p = 25$, $K_i = 0.25$, $K_d = 0$

Startup with PID control of the drums



Startup with PID control of the drums



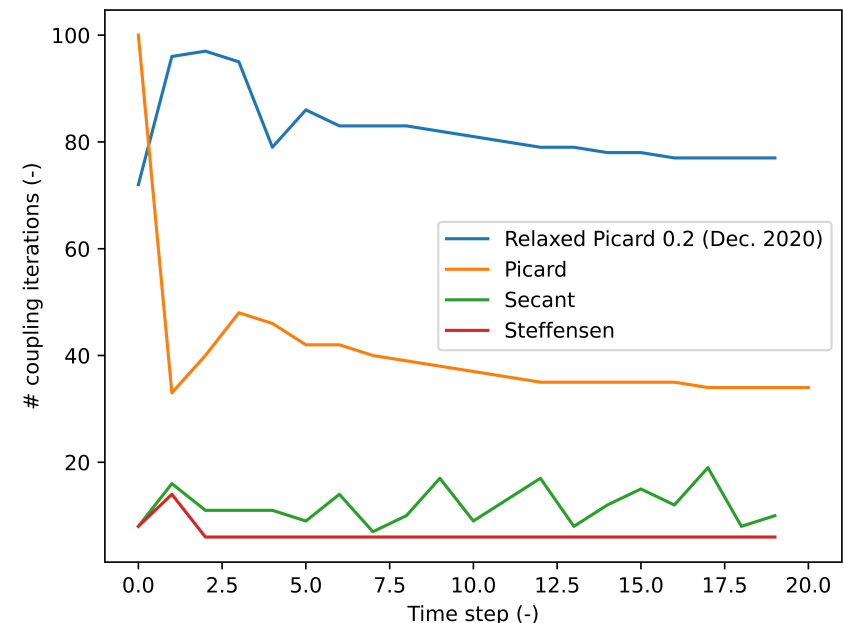
Improvement of Picard Iteration Convergence

- Strong coupling (single nonlinear problem) is often impractical (memory consumption, time/length scales, different meshes)
- Picard iterations convergence slowly:
 - Neutronics, heat conduction, thermal-hydraulics (~200k DoFs)
 - One evaluation is 1 minute, 600 times steps, 10 Picard
 - Runtime is **10 hours**

- Acceleration: Secant's, Steffensen's methods (Anderson's method TBD)
- Promising work done by Guillaume Giudicelli @ INL
- Previous work in the DOE CASL program by Alexander Toth

*"In general, we noted that Anderson provides a significant improvement in robustness."
- from A. Toth's dissertation*

Iterations for 2D/1D fluid flow coupling
(courtesy G. Giudicelli)





Concluding Comments

- MOOSE-based tools have been and are being developed to support multiphysics simulations of complex systems.
- Neutronics-thermal/hydraulics-structural analysis performed for a Fuel Element model
- Neutronics-thermal-hydraulics start-up transient performed on a full-core model with PID control of the drums
- Computational cost remains significant but promising work could help to reduce it.
- Concurrently INL is working to validate Griffin with available SIRIUS data and helping in design of PRIME-1.
- Future work
 - Adding a full T/H system model (nearing completion)
 - More realistic cross sections for startup (evaluated for startup temperatures)
 - Improved efficiency for startup simulations (fuel element first, then full core)



Computational Cost

- Currently, computational cost of each individual physics is reasonable (e.g., 30s for a single neutronics solve with 192 processors)
- However, Picard iterations can be slow to converge
- In particular, it seems that the fuel element heat conduction and T/H channels coupling terms are slow to converge
- Hence the calculation ends up taking a long time (48 hours with 192 procs)

