

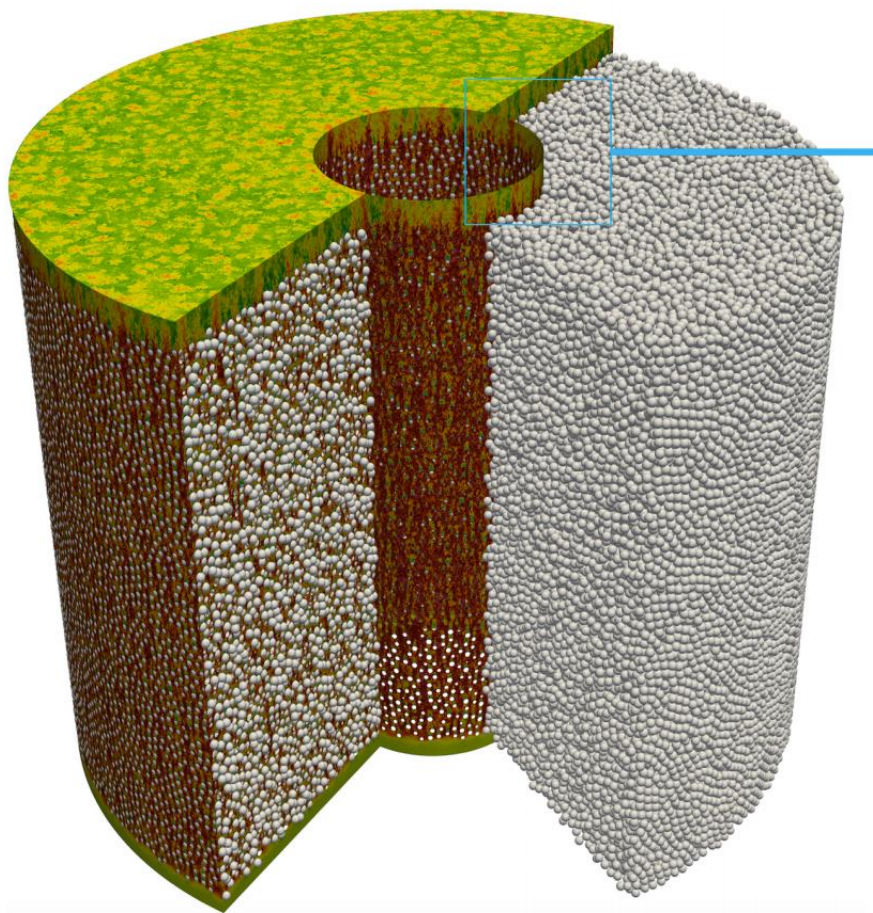


Postprocessing results from Cardinal using Pyvista

Matthew Falcone

1. Cardinal
 - What is Cardinal?
 - What data does Cardinal produce?
2. Pyvista
 - What is Pyvista
3. Implementation of readers for Cardinal in Pyvista
 - ExodusIIReader and Nek5000Reader
4. Simple example
 - Flow around a pebble Cardinal tutorial
5. Discussion

What is Cardinal?



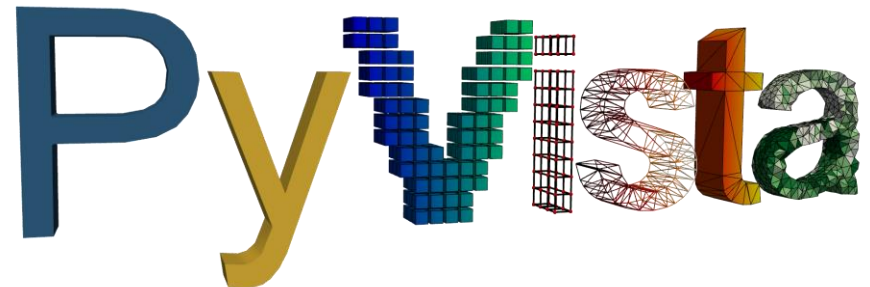
- Wraps nekRS and OpenMC into the MOOSE multiphysics framework.
- Can couple
 - Fluid dynamics
 - Neutron and photon transport
 - Conjugate heat transfer
 - Structural mechanics
 - Any MOOSE app
- Common existing use cases include:
 - Nuclear reactor cores (Fission).
 - Increasingly used for Fusion
 - Breeder blankets.

What data does Cardinal need/produce?

- Data required:
 - nekRS mesh
 - Bespoke *.re2 format
 - Often exported from Coreform Cubit via Exodus II and converted using exo2nek utility
 - Solid mesh for MOOSE:
 - *.e Exodus II file
 - OpenMC
 - CAD geometry
 - Assorted text files
- Data produced:
 - NekRS output data
 - *.nek5000 metadata file
 - Data file *0.f0001
 - Solid output data
 - *.e Exodus II file
 - *.e.N.p Parallel Exodus II (Nemesis) file
 - OpenMC
 - Has many of its own Python tools
 - Assorted text/CSV files.

What is Pyvista?

- A Pythonic wrapper to the Visualisation Toolkit (VTK)
- It inherits from the VTK python classes but makes them much more user friendly
 - VTK python classes are mirrors C++ API – very clunky
 - Pyvista classes are fully compatible with VTK parents so easily extendable
 - Much lower learning curve
- Most filters that you find in Paraview are in Pyvista
 - Both are based on VTK
 - For example:
 - 'Cell data to point data' in Paraview
 - `Obj.cell_data_to_point_data()` or `obj.ctp()`
- Easy access to underlying data.



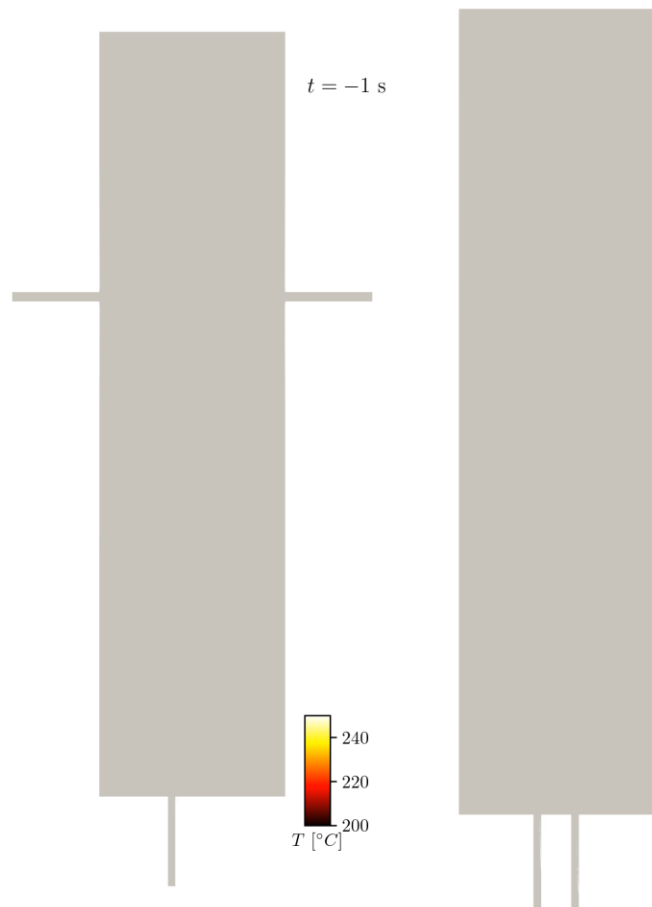
Implementation of readers for Cardinal in Pyvista

- Two VTK reader classes needed to be wrapped to enable visualisation of Cardinal data in Pyvista
- **ExodusIIReader:** wraps the `vtkExodusIIReader` class from VTK
 - Conveniently exposes blocks/sets and their respective arrays at high level.
 - By default, the element blocks arrays and nodal arrays are enabled.
- **Nek5000Reader:** wraps the `vtkNek5000Reader` from VTK.
 - Paraview options such as passing the spectral element IDs as cell data and merging element boundary points are available are exposed.
- Note on Nemesis files
 - On the todo list.
 - Wrap `vtkPExodusIIReader`: cannot get it to work
 - Wrap `vtkIOSSReader`: requires some donkey work to get into Pyvista
 - As a workaround each file could be read by the `ExodusIIReader` and merged.

Limitations and comparison with Paraview

- Pyvista has a similar role in a post-processing pipeline but using both is perfectly possible.
 - Pyvista: scripted repeated postprocessing for producing high-quality plots
 - Paraview: quick look at results using a GUI
- Compared to Paraview, Pyvista has some limitations performance-wise
 - VTK is not by default built with threading or MPI
 - Question: how often do you use Paraview with MPI.
- Pyvista can still run fast:
 - Build VTK from source with OpenMP enabled (I have scripts for this in the repo and a singularity container)
 - Can run it on the HPC for faster file access and use up to a node.
- Now let's look at some examples:

- Results from an LES Code_Saturne finite volume simulation of the liquid sodium Thermal Stratification Test Facility at UWM



1. What are trade-offs between using a posteriori and in-situ postprocessing/visualisation including some applications?
 - If you have experience of in-situ visualisation tools such Paraview/VTK Catalyst.
2. How do postprocessing/visualisation workflows change between unstructured and structured datasets?
3. Do you have suggestions about where a scripted high-level tool like Pyvista can be used in the work flows common to the group.