Summer 2016 Report – INL

Purpose: Incorporate thermal feedback into the time dependent transport code TDKENO to improve simulations of TREAT transient experiments.

**Current status of TDKENO**

Feedback is incorporated into TDKENO with a simple empirical model from historical data for TREAT specific problems. This model calculates the reactivity from the yield and provides a correction factor to the reactivity. This provides reasonable answers but seems to be way off in calculating the power vs. time.

Quenching coefficients and several problem specific feedback methods were included in the past. A simple adiabatic model was put in place for a benchmark problem in (Bentley). This is the closest to a generalized feedback model as was every attempted.

**Methods of Incorporating Thermal Feedback**

For TREAT experiments and most transients, the thermal effect that needs to be resolved is the Doppler broadening of cross sections. As the reactor increases in temperature the cross sections are broadened. These changes in cross sections propagate through the entire problem. Without feedback or with incorrect feedback the cross sections will not be updated and the results will differ from experiment. Transients occur quickly and therefore can be looked at with models as simple as an adiabatic heat-up model. From conversations with others working on TREAT the clipped transients work very well with this model. The next improvement, we believe for shaped transients, is to look at a steady state heat equation to capture the thermal conductivity effects. The ultimate goal for reactors besides treat is to couple the neutronics of TDKENO with a CFD (computational fluid dynamics) or TH (thermal-hydraulics) specific code. Conventional reactors are moderated with water that may be in several phases. Such simple approaches generally do not work with systems such as these. As the goal is first to understand TREAT we looked at incorporating the adiabatic and a heat conduction model.

Before getting into details of such we describe the basic coupling strategies.

1. **Internal coupling** – Two or more code bases may be built together such that one may call the others routines/functions/data. In the case of TDKENO we would want to build it under the MOOSE framework. That would enable data to be based around instead of being written out and read by another program. The difficult in internally coupling TDKENO is the vastly different structures and less formal software development practices within TDKENO. One such difficulty is TDKENO being written in Fortran 90. Talking with Cody at the MOOSE team, Fortran 90 is slightly more difficult because of it inconsistent name mangling makes it difficult to make compiler agnostic functions. He said it is definitely possibly though, and they have built legacy programs within MOOSE before. However, he said each time it was a unique problem and challenging. Something that in itself may take the whole summer. Looking to get some kind of results by the end of the summer I did not opt for the internal coupling method.
2. External Coupling – External coupling generally refers to code coupling methods build exclusively and rely on passing around input/output files to achieve coupling of data. These methods are not as “clean” or robust but may be quicker to implement. This has an additional benefit in that KENO does not have to be built under the MOOSE framework, keeping TDKENO open to the possibility of using a different solver for the flux shape. The reliance on KENO as the flux shape solver will be addressed in the *future work* section.

**External Coupling Implementation**

The basic strategy of coupling MOOSE and TDKENO externally is to write out the necessary information with several scripts to parse data and call either program. This turned out to be the most difficult portion to do in a robust manner.

The fundamental issue with coupling TDKENO/MOOSE is the differing geometric approaches to modeling system. TDKENO models the system with KENO’s Monte Carlo Geometry defined by quadratic functions known as CSG (combinatorial solid geometry, a common method in M.C). This allows for relatively simple definitions of the geometry at may describe the problem exactly. Whereas MOOSE is a finite element analysis tool that requires meshing the geometry. This is typically less detailed and more difficult to create than with Monte Carlo. Looking through the literature and KENO manuals there was no clear method for mapping information between CSG and mesh based geometries. Additionally, KENO has no generic interfaces for multi-physics coupling. Several strategies were investigated to create a generic enough mapping of information between the two codes.

The first was to tally the fission density on top of the Monte Carlo geometry. This is already possible in KENO. The user defines a Cartesian grid, specifying the height, width, depth and number of points in-between. The fission density is then found in each component of this Cartesian geometry. The discretization would have to be done such that the grid defined would match up to fuel elements. If done this would provide the fission density in each fuel element. If the CSG geometry defined fuel elements did not align with the grid then fissions may be tallied over larger or smaller regions than they should. The result would some fuel elements appearing to have artificially larger or lower temperatures. While not ideal in principle it is possible to define a grid that lines up well with the existing geometry.

KENO writes out a mesh in a proprietary file type “.3dmap” that may be read by several SCALE applications. There are tools within SCALE to convert these file types to more generic ones such as VTK and SILO. The VTK format, supported by VisIt, seemed the most appropriate as it appeared MOOSE could read those formats. Upon further inspection was not entirely true. MOOSE is only able to read unstructured XML based VTK files. This is a newer more robust format that is used by VisIt and supported by LibMesh. So the simple convertor already available in SCALE was not useful as it was only able to convert from .3dMap to rectilinear VTK format. In principle it would be possible to modify MOOSE to read these formats but would require significant changes according to the MOOSE team.

The next thought was to convert from VTK rectilinear to XML unstructured VTK. It was possible to convert to XML but it remained a structured mesh. With no way to have this KENO information into MOOSE the VTK XML was converted to .XYZ format. This was read into moose and captured the nodal points correctly but all of the associated information about the fission density was lost.

One more approach was attempted. That was to utilize the Exodus II API to construct a mesh directly from the KENO data. Exodus is the preferred format in MOOSE. The API allows for creation of exodus Meshes fairly simply. The difficulty came in the format KENO stored everything made the mapping between nodal points confusing and difficult to come up with a scheme that would work in general.

3dMap 🡪 VTK (rectilinear) 🡪 VTK XML 🡪 MOOSE?

3dMap 🡪 VTK (rectilinear) 🡪 VTK XML 🡪 XYZ 🡪 MOOSE?

3dMap 🡪 VTK (rectilinear) 🡪 Exodus II🡪

3dMap 🡪 Exodus II 🡪 MOOSE

Instead of converting the mesh into a format the MOOSE can already read it started to make sense to construct a mesh in MOOSE and map values onto that mesh that approximately represents the geometry in KENO. This seemed simpler as one can query mesh points in MOOSE and add values to it on the fly. Instead of utilizing the .3dmap based mesh problem was discretized based on each fuel element of TREAT. Note, this method is not robust and generally would only work for simple heatup problems like TREAT. The fission density per unit was already given in KENO without additional tallies. Our model of TREAT is organized into units of fuel elements positioned based on its array location. See the figure below. A unique unit may be constructed so the fission density is not averaged over similar units that are repeated. Since the units were arranged into an array it is relatively simple to get create the geometry in 2D and capture the radial temperature dependence. Furthermore additional discretization may not have been useful due to the method our model was created and how the cross sections are created. Each fuel element unit has several materials in it that are used to create multi group cross sections. These materials may be re-used and therefor the cross section values re-used. Even in different parts of the core at possibly different temperatures. Updating the cross sections to new temperatures means assigning materials a new temperature and having that material correspond to a unique unit.

FIG. TREAT MODEL 2-D to show array nature of unit placement.

A pseudo-mesh was created based on the location of each fuel element in 2D with the corresponding fission density, temperature. A series of python scripts parse the KENO input to gather the fission density, create an (x,y) location for the center of each fuel element, and calculate the power density. The power density at first is calculated simply by multiplying by the average release per energy for U-235. In reality this energy/fission value is more complicated to calculate but for an initial test the simpler option was chosen.

Next, we move on to working on the MOOSE side. The goal is to solve the heat conduction problem with the heat source as the calculated power density. A heat conduction module is already available within MOOSE. This was modified slightly to read in the text file produced form the KENO data and assign a different thermal conductivity value and power density at each quadrature point. The mesh generated in MOOSE is created *a priori* to have approximately the same dimensions in 2D as the TREAT problem. The KENO data was really providing values on nodal points but the values needed to be assigned on quadrature points, which are slightly offset from the nodal points. This was achieved by simply determining the absolutely closest point from the KENO data to the corresponding quadrature point.

FIG. Nodal vs. quadrature points.

This all appears to work in terms of mapping the data from KENO to a MOOSE mesh. Historical data is available on the thermal conductivity values for TREAT fuel elements. These values are dependent on the temperature in fuel. So this value needs to change as the simulation progresses. This can be done within some of the python scripts or as an auxiliary calculation on the MOOSE side. At present the maximum value is used to give a rough estimate.

Despite all of the details not fully resolved it came to my attention that for the TREAT problems of interest it might suffice to just do an adiabatic heat-up model. This calculation does not have to be done with MOOSE since the relationship is so simple. It can all be programmed within TDKENO. In Bentley’s thesis an adiabatic heat-up model was programed for a specific benchmark problem. It seemed plausible to see if this could be extended to work in more general problems and specifically for TREAT. The hope is this model could vary the temperature in-between flux shapes as well. The method outline above, with MOOSE solving the heat conduction would only be updating the temperatures on the time scale of the flux shapes. Relying on getting the thermal feedback with the updated cross sections broadened to the new temperature.

**Cross Section Processing**

No matter the method of coupling to capture the affect of Doppler broadening the cross sections need to be modified. There are several approaches to this..

1. Updating the cross sections by creating a new set each time with the CENTRM utility. Basically, once MOOSE has calculated a new temperature for each fuel element this information is mapped back to the KENO input file. Each units materials has a new temperature. This input file is then processed with CENTRM to create cross sections unit dependent based on the new temperature. The cross sections is then saved and place in the TDKENO working directory. \*\* need to reference CENTRM some more
2. Interpolating from existing cross sections. Instead of calculating the cross sections for each update of the temperature several cross sections for the same geometry may be made available before hand and interpolated within TDKENO to get as close to the new temperature as possible. This requires knowing the range of temperatures the problem will be at. For TREAT and many problems this is known. The approach still is not as robust as the first and not as accurate as it relies on interpolation not broadening to the exact new temperature.

All of the pieces are close to completion for a rudimentary coupling of TDKENO of MOOSE in order to capture the thermal affects. Given the time constraints it was not possibly to complete a robust method to couple TDKENO with MOOSE. There are several different options that may be pursued in the future to develop a robust code to simulate transients with accurate thermal feedback.

**Future Work**

1. Build TDKENO within MOOSE. Establish 3D mapping of CSG geometry to MOOSE meshes
   1. Some advantages. Would give ability to steadily improve thermal calculation on the MOOSE side. Clean implementation, as only data would be passed around.
   2. Disadvantages. Application would be all export controlled if build under MOOSE. Would have to build KENO and maybe CENTRM on the MOOSE platform, some “political” considerations here. These are already difficult to do on the system SCALE established.. Lots of programming challenges here. Mapping still not entirely clear how to do robustly.
   3. Could opt to build the KENO/CENTRM executable separately and only call them when needed. The “TDKENO” portion, basically the point kinetics solver/flux modifier would be built under MOOSE.
2. Modify TDKENO to not utilize KENO for the flux shape calculation. Instead base it on a code such as Serpent or OpenMC with established multi-physics interfaces.
   1. Some disadvantages. These codes are not as mature, do not all possess the same features as KENO. For instance in performing the adjoint calculation. Require lots of rewriting of KENO possibly new formulation of the governing equations.
   2. Advantages. Established multi-physics interfaces. Lots of reference points and code already made for thermal feedback within MOOSE/each respective code. Modern codes may be easier to manage in the future.
3. To capture the temperature dependence the fission/power density may be tallied as a distribution based on Zernike polynomials. Keep the coupling somewhat external. The coupling would be simplified since only the Zernike
   1. Disadvantages. Further reliance on KENO. Modifying KENO source code can be challenging. A newer methodology and it is not widely implemented.
   2. Advantages. Construct almost exactly temperature distribution in great detail without discretizing the geometry from the CSG format. Only have to pass around coefficients and read the temperatures from MOOSE. Has been done in Serpent (in progress at INL), OpenMC but for Zernike polynomials that are orthogonal only circular domains.
      1. It is possible to have Zerneki polynomials orthogonal on polygons (for radial dependence) with greater than 3 sides. It has been show in some optics papers from 2015. The example given provided a general methodology for P>3 sided polygons and a specific case of an 8 sided polygon with equal length sides. For the TREAT problem the 8-sided case would need to be modified for sides that are not all the same but with two sides. It seems in principle possible. This would be a great solution for the TREAT problem. Could obviously be extended to cylindrical fuel elements with some ease. It doesn’t make this methodology super robust as it relies on having fuel elements with a specific geometry. For instance pebble bed reactors would be incompatible with this implementation.
4. Solving the heat conduction problem via Monte Carlo techniques. There has been some work over the last 50 years on solving the heat conduction equation with a Monte Carlo approach. Since the diffusion approximation in transport theory is mathematically equivalent to the heat conduction problem there are many analogous methods. There is a fair amount of work published on this approach but nothing too mature. Citing various computational difficulties and technical challenges.
   1. Disadvantages. Computational time may be larger than deterministic. Require significant programming from the ground up. May be tied to a specific geometry introduced by the neutronics solver. No ties with MOOSE.
   2. Advantages. Extremely robust! Can solve problems with complex geometries with no discretization beyond the CSG description. More novel research than existing coupling strategies?