ANL/NSE-20/4

Development of the PROTEUS and ANSYS Coupled System for Simulating Heat Pipe Cooled Micro Reactors

Nuclear Science and Engineering Division

About Argonne National Laboratory
Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via DOE's SciTech Connect (http://www.osti.gov/scitech/)

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandra, VA 22312
www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703) 605-6000
Fax: (703) 605-6900
Email: **orders@ntis.gov**

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728

ANL/NSE-20/4

Development of the PROTEUS and ANSYS Coupled System for Simulating Heat Pipe Cooled Micro Reactors

prepared by
Changho Lee and Yeon Sang Jung
Nuclear Science and Engineering Division, Argonne National Laboratory

February 15, 2020

ABSTRACT

The objective of this project is to develop the coupled system of PROTEUS/ANSYS for simulating a heat pipe cooled micro reactor. As an initial effort, while figuring out controlling ANSYS Mechanical with Workbench, the coupled system of PROTEUS/FLUENT in the CORBA environment were tried to demonstrate the coupled simulation for a transient problem of a heat pipe cooled micro reactor. Finally, the coupled system of PROTEUS/ANSYS was constructed using APDL commands in the CORBA environment, instead of ANSYS Workbench, since we found that ANSYS with the current version of Workbench Python scripting did not provide capabilities necessary for simulating a transient problem.

Python drivers were developed to complete the coupled system of PROTEUS/ANSYS, which coordinate the overall workflow including data exchange (power, temperature, mesh coordinates, and heat rate) required for the coupling and also control the individual calculation steps of the coupled system such as convergence check and boundary conditions.

The coupled systems were qualitatively verified using a 3D unit assembly problem composed of six fuel rods and seven heat pipes which was developed based on the specification of MegaPower. For the completeness of coupled calculations, ANLHTP was introduced and coupled together, which is the one-dimensional heat pipe performance analysis code developed and recently resurrected by ANL.

For verification, a steady-state problem was first solved by the coupled system of PROTEUS/FLUENT/ANLHTP, demonstrating a reasonable convergence behavior of heat pipe temperatures. A transient simulation with one heat pipe failure out of seven heat pipes was also performed by the coupled system, showing reasonable changes of total power and heat pipe temperatures with time accounting for temperature and power feedback effects. The same calculations were performed using PROTEUS/ANSYS/ANLHTP in which APDL commands were used for ANSYS and only thermal calculations were performed. In this coupled calculation, the similar trend of power and temperature changes with time but it was observed that the magnitude of the changes was larger than that obtained from the coupled system with FLUENT. Further investigation is being made to figure out what is causing the difference.

As future work, the coupled system will be verified for larger or actual-size 3D heat pipe cooled reactor benchmark problems and validated using experiments available such as the KRUSTY experiment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

In recent years, concerns in compact power generation are reemerging due to the growing demand of affordable and sustainable energy resources even in remote locations, military bases, etc. where electricity supply is limited. Since nuclear energy would be a good energy source for such purpose, efforts have been made to develop micro reactors including heat pipe cooled reactors and gas cooled reactors, which are small nuclear reactors generating power typically less than 10 MWe. Due to the current limitations of Monte Carlo approaches in performing transient and thermal expansion analyses, deterministic codes would be preferred to solve such reactor analyses. In this project, among various types of micro reactors, we focused on simulating a heat pipe cooled micro reactor using deterministic multiphysics tools.

On the other hand, Westinghouse Electric Company LLC (Westinghouse) is developing the heat pipe cooled micro reactor named eVinci [1] based on the experiences of a combination of nuclear fission space reactor technologies and over 50 years of commercial nuclear systems. Since eVinci is a new system which has not been designed and built previously, a reliable modeling and simulation tool for such a reactor is required to successfully design the reactor and prepare for its design licensing from the regulatory agency. In this project, we develop the multiphysics system of neutronics and thermal/mechanical analysis tools to support the eVinci design effort of Westinghouse.

Over a decade, Argonne National Laboratory (ANL) developed the high-fidelity capable three-dimensional (3D) deterministic neutron transport code PROTEUS [2,3,4] under the Department of Energy (DOE) Nuclear Energy Advanced Modeling and Simulation (NEAMS) program. PROTEUS is able to solve the neutron transport equation in steady-state and transient conditions, containing three solvers for flexible applications: the second-order discrete ordinates (SN) and method of characteristics (MOC) transport solvers based on unstructured finite element meshes as well as the nodal transport solver (NODAL) for hexagonal and Cartesian geometries for use in rapid design application. The SN and MOC solvers of PROTEUS can be used for simulating micro reactors which are mostly of irregular geometry. The SN solver is preferred to use for fast reactor systems where the heterogeneity effect is not severe, while the MOC solver is efficient to analyze thermal systems where neutron fluxes largely change over heterogeneous regions. A thermal expansion behavior, which is an important and challenging phenomenon of metal-fueled fast reactors, can be simulated by the unstructured mesh-based solvers of PROTEUS. As efforts of micro reactor applications, we simulated MegaPower [5,6] using PROTEUS, the detailed results of which can be found in the references [7,8].

ANSYS Mechanical [9], widely used in many areas, is the mechanical engineering software solution that uses finite element analysis for structural and thermal analysis. Most Ansys simulations are performed using the ANSYS Workbench system which offers the ability to record the actions you perform via the graphic user interface (GUI).

The main objective of the project is to develop the coupled system of PROTEUS/ANSYS and demonstrate its performance for simulating heat pipe cooled micro reactor in steady-state and

transient conditions. In the coupled simulation, thermal-hydraulic (T/H) and thermal expansion feedback will be accounted for in PROTEUS calculations. PROTEUS will be updated to include capabilities required for the coupling: write powers in a file, read temperatures from a file, and read and write a restart file.

During the development of the coupled system, a significant effort was spent to figure out how to control ANSYS within Workbench even for a steady-state problem. Controlling it for a transient problem was even more challenging. Because of the uncertainty in finding a way to control ANSYS Workbench, in parallel we tried the coupled simulation of PROTEUS/FLUENT [10] using the user defined functions (UDFs) in the CORBA (Common Object Request Broker Architecture) [11] environment. Finally, we were able to successfully make the coupled system of PROTEUS/FLUENT work for steady-state and transient problems, demonstrating the transient behavior of a small heat pipe cooled core with the failure of one out of seven heat pipes. In similar time period, we made the coupled system of PROTEUS/ANSYS Mechanical within Workbench successfully work for steady-state calculations as well.

After making significant efforts to figure out ways to control ANSYS Workbench for a transient calculation, we concluded based on ANSYS expert consultations [12] that it may be impossible to control the current version of ANSYS Workbench with Python scripting for a transient calculation because Python scripting may not provide capabilities of running a transient calculation time step-by-step playing with restart files and thus a lower level of control is required to handle ANSYS for a transient simulation.

In order to control PROTEUS/ANSYS for a transient calculation, we tried APDL commands directly in the CORBA environment as done in PROTEUS/FLUENT. ANSYS was more difficult to control for a transient calculation than FLUENT since ANSYS does not allow hold and resume a transient calculation, requiring quitting and restarting the transient calculation to read powers updated from PROTEUS with time. In order to use APDL, we used the ANSYS launcher instead of Workbench for the coupled simulation.

For complete multiphysics tests, ANLHTP [13] was introduced which is the one-dimensional heat pipe performance analysis code developed by ANL in 1980s and resurrected for the coupled simulation in this project. ANLHTP currently handles only sodium as coolant material and checks five operational limits such as a heat pipe including viscous, sonic, entrainment, boiling and capillary limits. The coupled calculations of the three codes are performed with the combination of two coupled calculations of PROTEUS/ANSYS and ANSYS/ANLHTP, having no direct communication between PROTEUS and ANLHTP.

The coupled systems were tested for a small 3D problem with 180 cm high (including 15 cm thick top and bottom reflectors) which is composed of six fuels and seven heat pipes based on the MegaPower specifications. The problem in a steady-state condition was tested first to make sure that solutions including eigenvalue, fission source, and temperature were converged with iterative calculations between PROTEUS/ANSYS/ANLHTP. The transient calculation was also conducted with one heat pipe failure which caused an immediate temperature increase at the failed heat pipe,

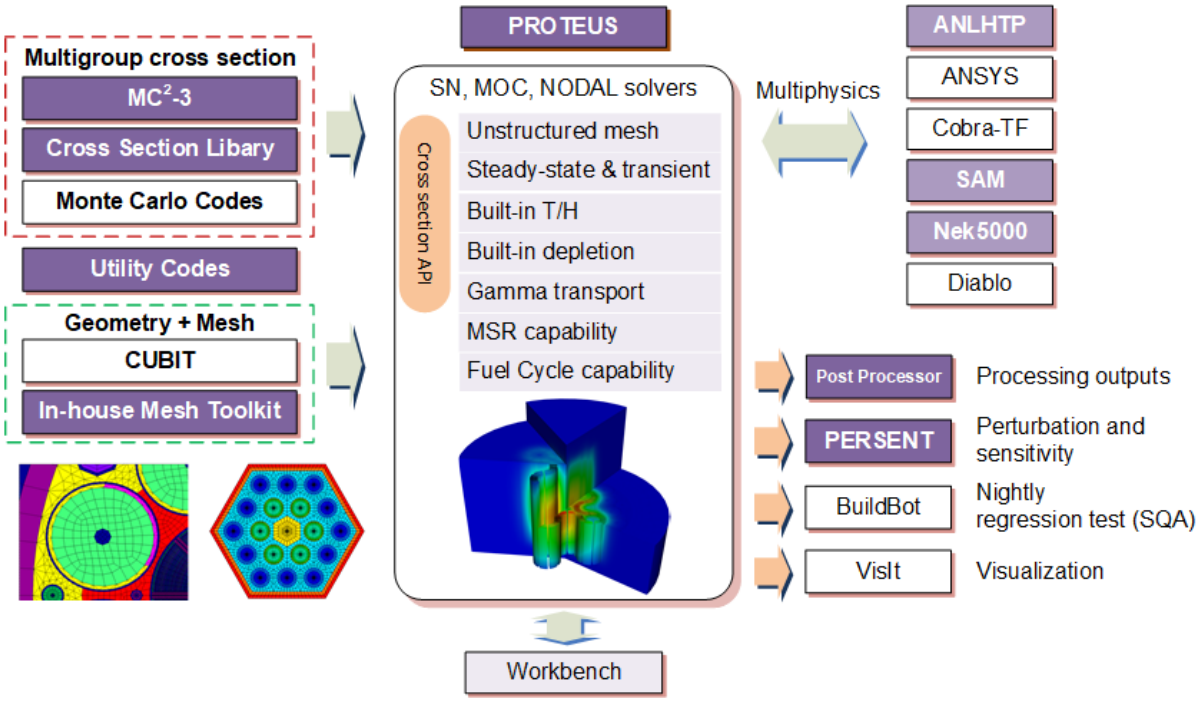
resulting in a power decrease due to Doppler feedback which eventually leads to the decrease of average temperatures at the other heat pipes.

Section 2 introduces the codes used in the coupled simulation of micro reactors, and Section 3 presents the approaches used for the coupled simulation of PROTEUS with ANSYS or FLUENT. Demonstration tests of the coupled systems are discussed in the section as well. Conclusions and future works are discussed in Section 4.

2. Computer Codes Used in Micro Reactor Simulation

2.1 PROTEUS

The PROTEUS code is a high-fidelity capable neutron transport code based on finite element discretization of the domain, which has been developed under the DOE-NE NEAMS program. The S_N and MOC transport solvers are available in PROTEUS to solve heterogeneous geometry problems with no or minimal geometrical approximations. The nodal transport solver based on homogenized assemblies and structured geometry was also implemented in PROTEUS to provide a conventional-fidelity level of solutions in a consistent framework for use in rapid design application. In the NODAL solver, two methodology options are available: P_N and Simplified P_N (SP_N). The P_N approach is the identical methodology used in DIF3D-VARIANT [14] although the release version only handles diffusion theory on Cartesian, hexagonal, and triangular-z grids. For the SP_N approach [4], a transverse integrated nodal methodology was built on the hexagonal grid model utilizing up to a SP_3 approximation.



(The purple-colored boxes denote the codes developed by Argonne)

Figure 2-1. Overview of the PROTEUS System

All the three solvers are able to solve steady-state and transient problems with the built-in thermal fluid calculation capability. The gamma transport calculation is possible to accurately solve for power distributions. In the NODAL solver [4], the molten salt reactor (MSR) capability was implemented along with relevant thermal fluid modules so that it is able to solve MSR problems, accounting for redistribution of delayed neutron precursor concentrations due to fuel flow velocities inside and outside the core.

The fuel cycle capability as well as depletion was recently implemented in the NODAL solver to meet needs for actual reactor core design and analysis. The NODAL solver is able to generate the CCCC interface files in order to run PERSENT [**Error! Reference source not found.**] for reactivity perturbation and sensitivity analysis.

Cross sections for PROTEUS can be generated in the ISOTXS format using MC^2-3 [16] and Monte Carlo codes (Serpent [17] and OpenMC [18]). With the Monte Carlo codes, the GenISOTXS code [19] produces ISOTXS files which can be combined to produce the ISOPAR format with state parameters in terms of temperature, burnup, control rod, void fraction, etc. In addition, multigroup cross sections can be generated on the fly from PROTEUS using the cross section API [20], accounting for the self-shielding effect of geometry and composition of the problem of interest.

Geometry and mesh are generated using CUBIT [21] standalone or a combination of CUBIT and the Argonne mesh toolkit [22]. For conventional hexagonal and Cartesian geometry problems, the Argonne mesh toolkit is able to easily generate meshes with text-type keyword inputs. Recently, the NEAMS Workbench [23] has been updated with PyPROTEUS [24] which help users create PROTEUS inputs, run the code, and post-process code outputs.

Multiphysics simulations with other physics tools such as Nek5000, Diablo, Cobra-TF, SAM, ANSYS, and ANLHTP. Currently, Nek5000 and Diablo work with PROTEUS-SN, Cobra-TF and ANSYS can be coupled with PROTEUS-MOC, and SAM (a MOOSE-based code) works with PROTEUS-NODAL. The coupling approach that we employed for the coupling of SAM and PROTEUS-NODAL will be extended for coupling with other MOOSE-based codes in the future.

Figure 2-1 shows the overview of the PROTEUS suite, including PROTEUS capabilities and the codes that were developed by ANL (purple-colored boxes) or used to support PROTEUS and multiphysics simulation.

2.2 ANSYS

2.2.1 ANSYS-Mechanical

ANSYS Mechanical is the mechanical engineering software solution that uses finite element analysis (FEA) for structural and thermal analysis. Covering a wide range of applications from geometry preparation to optimization, ANSYS Mechanical is able to model advanced materials, complex environmental loadings as well as industry-specific requirements.

ANSYS Workbench is the framework upon which the industry's suite of advanced engineering simulation technology is built. An innovative project schematic view ties together the entire simulation process, guiding the user through even complex multiphysics analyses with drag-and-drop simplicity. With bi-directional CAD connectivity, an automated project level

update mechanism, pervasive parameter management and integrated optimization tools, the ANSYS Workbench Platform delivers unprecedented productivity, enabling simulation driven product development.

ANSYS Workbench offers the ability to record the actions you perform via the GUI, referred to as journaling. Journals are recorded as Python-based scripts [25]. One can modify these scripts or create new ones, referred to as scripting. Together, these capabilities allow us to quickly and easily replay analyses you've already run via recorded journals, as well as to extend functionality, automate repetitive analyses, and run analyses in batch mode.

2.2.2 FLUENT

FLUENT is a well-known computer program for modeling fluid flow, heat transfer, and chemical reactions in complex geometries. The code provides complete mesh flexibility, including the ability to solve your flow problems using unstructured meshes that can be generated about complex geometries with relative ease. The code allows users to simulate: 1) 2D planar, 2D axisymmetric, 2D axisymmetric with swirl (rotationally symmetric), and 3D flows, 2) Flows on quadrilateral, triangular, hexahedral (brick), tetrahedral, wedge, pyramid, polyhedral, and mixed element meshes, 3) Steady-state or transient flows, 4) Incompressible or compressible flows, including all speed regimes (low subsonic, transonic, supersonic, and hypersonic flows), 5) Inviscid, laminar, and turbulent flows, 6) Newtonian or non-Newtonian flows, 7) Heat transfer, including forced, natural, and mixed convection, conjugate (solid/fluid) heat transfer, and radiation, and so on. FLUENT is ideally suited for incompressible and compressible fluid-flow simulations in complex geometries.

2.3 ANLHTP

For thermal analysis of a heat pipe cooled reactor core, a heat pipe code is required, which can evaluate the amount of the heat removed by heat pipes for given conditions. In this study, we selected ANLHTP [13], which is a one-dimensional heat pipe analysis code developed at ANL in the 1980s. The code was developed to simulate a sodium heat pipe based on theory, analyses, and experimental data presented by Chi and Dunn and Reay. For simplification, it was assumed that the evaporator and condenser are nearly isothermal (at uniform temperature) and there is negligible axial heat conduction along the pipe wall or wick. These assumptions allow the code to calculate the heat transfer rate without solving differential equations of fluid and solid structures.

The input data of ANLHTP are pipe geometry, working fluid, thermal boundary conditions, operation mode, etc. Based on the input data, the code calculates wick parameters (being able to handle covered groove, open groove, screen, and screen & artery wick types) and makes an initial guess for the heat transfer rate to solve the non-linear equation. Then, the flow rates in each part of the heat pipe and the required thermal resistances are evaluated. With these, the heat transfer rate is updated. Iterations are made until the convergence on the heat transfer is achieved. Then, the code compares the estimated value with five operational limits of a heat pipe including viscous, sonic, entrainment, boiling and capillary limits.

The simplified calculation procedure of the code is plotted in Figure 2-2 and the key numerical procedure of ANLHTP is denoted with a blue line. In this procedure, the code

calculates the thermal resistances at each part of the heat pipe as illustrated in Figure 2-3. Among them, resistances 5-7 are for the temperature changes along the vapor path. These resistances are small in most cases but cannot be neglected if the vapor pressure is low so that the Mach number in the vapor core is large or the compressibility effects are significant. The resistances in the vapor core are calculated using the temperature drops evaluated from the pressure drops. The pressure drops are calculated using the one-dimensional momentum conservation equation which requires constitutive equations for the friction factors depending on the flow condition. The code incorporates the friction factor models for both incompressible and compressible flows, either of which is applied considering the Mach number criterion, 0.2, for the compressibility. A reduced form of the one-dimensional equation is applied, enabling the code to estimate the pressure drops including the compressibility effect in the adiabatic region and the pressure recovery in the condenser. For the thermal resistances 1-3 and 9-11, the analytical thermal resistance model for a circular tube is applied. For the wick region, the effective thermal conductivity of the saturated wick is used along with the wick's radii. The thermal resistances at the boiling and condensing surfaces are calculated using the empirical model presented by Dunn and Reay.

For the capillary limit calculation, ANLHTP uses the pressure drop calculation results for both liquid and vapor. It compares the total pressure drop occurring at the beginning of the evaporator and the maximum capillary pressure difference for the limiting case when the meniscus is flat and a wet point exists. The total pressure drop should be smaller than the maximum capillary pressure difference to sustain the circulation. Otherwise, the heat pipe is assumed to reach its operational limit. At the same time, it checks if the liquid pressure exceeds the vapor pressure at any axial location. If the pressures of the liquid are all less than the corresponding vapor pressure, the flow would be stable. Otherwise, the code assumes that the operation limit is reached. In this manner, the capillary limit of the heat pipe can be evaluated. For other operational limits, empirical correlations for the limiting criteria were implemented.

ANLHTP has been validated against two existing experimental results with sodium heat pipes: one was the operation limit test result conducted at Los Alamos National Laboratory (LANL) [26] and the ANL's Heat Pipe Test Facility (HPTF) [27]. For both experiments, ANLHTP showed reasonable prediction.

In the LANL experiment, a 1 inch diameter and 1.1 m length sodium heat pipe was operated over a large range of evaporation inlet conditions, using the screen & artery wick. The heat was supplied to the apparatus by induction coils and rejected though a gas calorimeter. In the simulation, the input temperatures were ranging from 600 to 875 °C, and the sink temperatures were varied to search for performance limits. The ANLHTP results show very good agreement with the measurement. The code calculations indicated that the heat pipe performance met the sonic limitation below 630 °C, while in the range of 630-650 °C, the performance was limited by the capillary and pressure limitations. For the higher evaporator temperatures, the code indicated that the heat transport was limited by the entrainment limit.

In the HPTF experiment, a sodium heat pipe of 2 inch diameter and 0.7 m length was tested with various input powers to the evaporator. It consisted of an Incoloy 800 pipe, 2-1/2 wraps of 100 mesh screen lining inside wall, and two longitudinal arteries of 0.125 inch inner diameter. The steady-state heat pipe data were obtained under 19 power conditions, and the temperatures at the evaporator were measured. The heat transport rates and temperatures were in the range of 6.51-8.39 kW and 760-871°C, respectively. These conditions were well below the operational

limits of the heat pipe and in its operating states. In this context, the HPTF experiment was an operational performance test. The linearly increasing trend of the heat transport rate along with the evaporator temperature was well reproduced by ANLHTP.

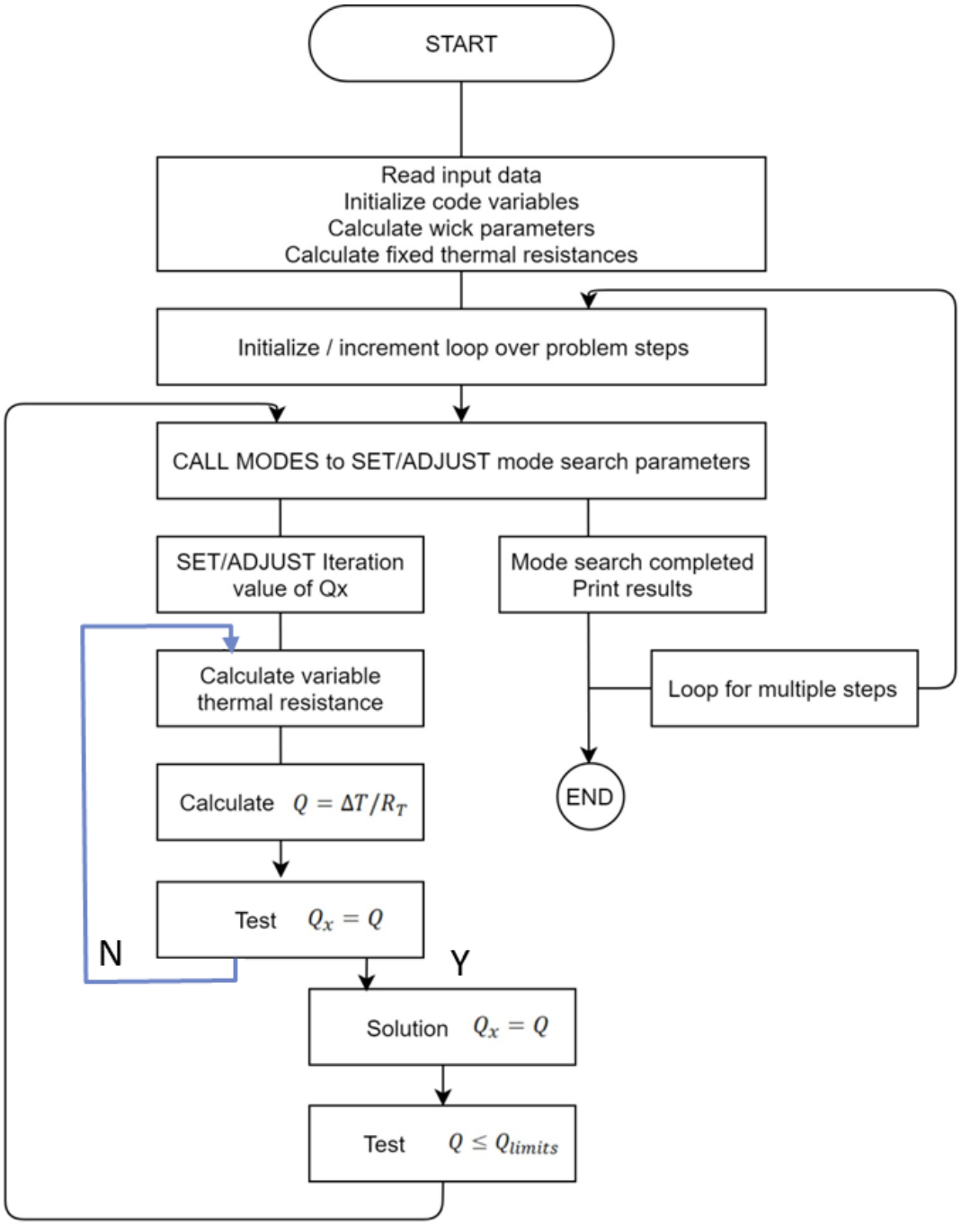


Figure 2-2. ANLHTP Calculation Flow

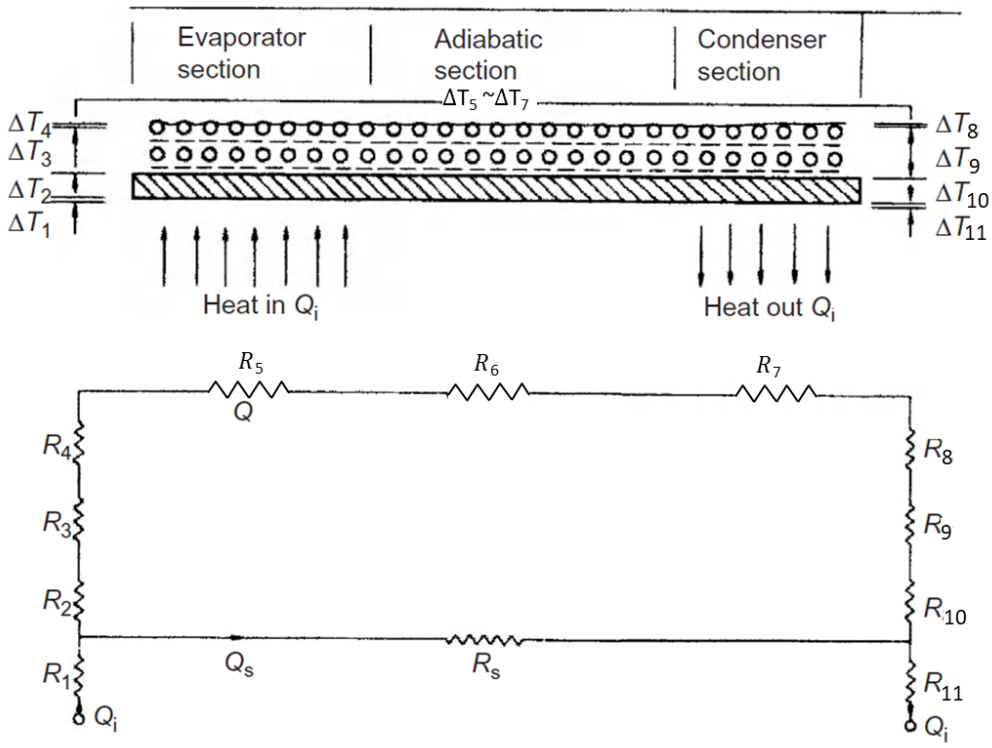


Figure 2-3. ANLHTP Thermal Resistance Network

The original ANLHTP was improved for coupling with a commercial thermal analysis code FLUENT. At first, its robustness was enhanced by removing the discontinuity or oscillation in physical models when a flow regime inside a heat pipe undergoes a laminar-turbulent or incompressible-compressible flow transition. This feature accelerated the convergence characteristic of the Picard iteration for the coupled simulation between ANLHTP and FLUENT. Secondly, solid properties for new material were added for various wick structure modeling. Finally, input and output processes were modified to handle the code using a MATLAB or Python script which was written to control the boundary conditions of ANLHTP. This improved ANLHTP and the scripts were used for the coupled simulation.

As ANLHTP was developed to predict heat pipe performance and temperature distributions during a steady-state operation, its application is limited to the steady-state conditions or possibly slow transient conditions. Moreover, it assumes a uniform heat flux distribution in calculating the pressure drop, which may cause errors if the heat flux is not uniform significantly and the pressure drop in the vapor core is considerable with compressibility. Therefore, quantitative analyses for the errors caused by these assumptions are required, and further improvements would be necessary in future.

2.4 Python Drivers

For coupling PROTEUS and ANSYS, we use Python scripts for controlling the codes. Since we cannot directly modify ANSYS, the communication between the codes are performed through reading and writing data files: i.e., providing powers from PROTEUS to ANSYS and temperatures and geometry changes from ANSYS to PROTEUS. Using Python scripts, holding

and resuming the execution of the codes is conducted by checking the update status of the transfer files. Python scripts are able to do data interpolation as well as to check convergence of the coupled calculation. In order to implement a control interface of PROTEUS in the framework of Python, the MPI interface for Python - mpi4py - was employed. This MPI interface allows users to directly communicate the Python driver and PROTEUS. The interface of the coupling driver to PROTEUS was built based on the MPI library with minimal modifications of PROTEUS, enabling to control the PROTEUS calculation and exchange the coupling variables with ANSYS as illustrated in Figure 2-4 which is an example of transferring temperature solutions from ANSYS to PROTEUS.

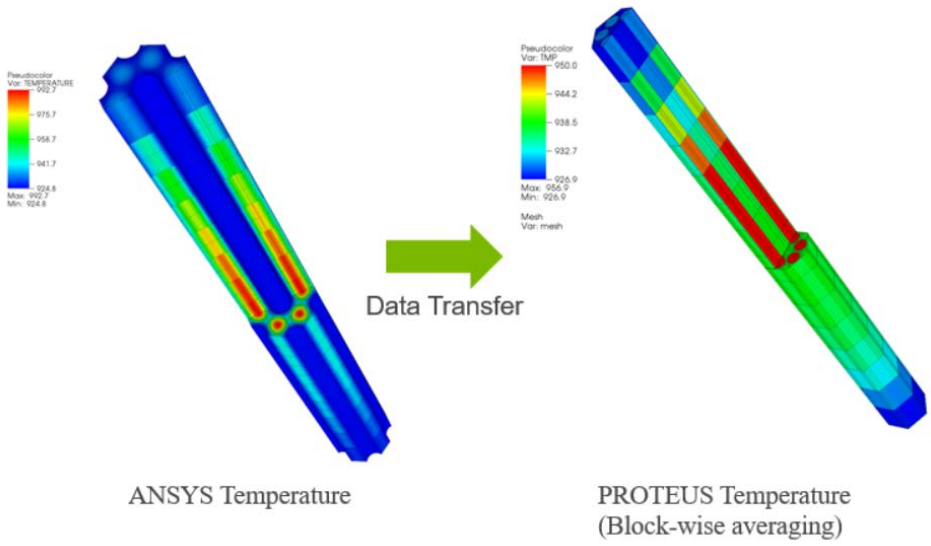


Figure 2-4. Demonstration of PROTEUS-MOC and ANSYS Coupling Simulation

3. Multiphysics Simulation of Heat Pipe Cooled Micro Reactors

The goal of the coupled system is the coupling of PROTEUS and ANSYS Mechanical with Workbench for simulating a heat pipe cooled micro reactor. However, since the coupling with ANSYS Workbench was successful for solving a steady-state problem but had difficulties finding a way for solving a transient problem, we tried the coupling of PROTEUS and FLUENT for steady-state and transient problems, while continuing looking at resolutions of the approach with ANSYS Workbench. Finally, since it appeared based on many ANSYS expert consultations that it is not possible to control ANSYS Workbench for a transient simulation using a Python scripting, we tried to control a transient simulation using APDL commands which was successful, allowing more flexibility of controlling ANSYS than Workbench Python scripting.

Section 3.1 presents three approaches: 1) the coupling of PROTEUS and FLUENT which are controlled by Python drivers and FLUENT user defined function (UDF) commands with CORBA, 2) the coupling of PROTEUS and ANSYS Workbench even though it is successful only for solving a steady-state problem, and 3) the coupling of PROTEUS and ANSYS with APDL commands for solving steady-state and transient problems. Section 3.2 shows the test results of the coupled systems from the approaches 1 and 3 for the transient problem with one heat pipe failure.

3.1 PROTEUS Standalone

For verification of PROTEUS solutions of heat pipe cooled micro reactors, multiple benchmark problems were developed based on the MegaPower reactor specification, including a fuel pin, a unit cell, 2D and 3D cores with different control material locations in the control drums. PROTEUS solutions for those benchmark problems were compared with the corresponding Serpent Monte Carlo solutions. Among the benchmark problems, the unit cell problem was extended to a 3D problem and used in the test calculations for the coupled system which is to be discussed in Section 3.3.

Cross sections for PROTEUS were generated from Serpent, whose cross section outputs were processed by GenISOTXS, or generated from MC^2-3 directly. Geometries and meshes were created using the combination of CUBIT and the Argonne mesh toolkit as briefly discussed in Section 2.1.

The comparison results indicated that PROTEUS eigenvalues were in good agreement with Serpent solutions within 200 pcm in most cases. The axial relative power distributions from the two codes also agreed well. Details of the PROTEUS standalone calculations can be found in the references [7,8].

3.2 Coupled System

3.2.1 PROTEUS/FLUENT/ANLHTP

For multiphysics simulation of micro reactors, three physics codes were coupled together as shown in Figure 3-1: PROTEUS for neutronics analysis, FLUENT for thermal analysis, and ANLHTP for heat pipe performance analysis. FLUENT is replaced by ANSYS for the coupled system of PROTEUS/ANSYS/ANLHTP. An efficient way of coupling different codes would be to develop a driver module and compile all codes together in order to control physics components by calling them directly and exchanging data required for individual physics calculations via data memories. Since, however, FLUENT is a commercial code which is difficult to compile with external codes, power and temperature data were transferred between PROTEUS and FLUENT via files. Additionally, Python-based external drivers were developed to coordinate the overall workflow including data transfer and to control individual calculation steps of the coupling of PROTEUS and FLUENT. Specifically for PROTEUS, the Python driver utilizes the Message Passing Interface (MPI) library to connect the PROTEUS solvers, which enables to control the neutronics calculation flow and transfer data through MPI communications. PROTEUS was updated to support the MPI-based Python driver, implementing the interface and data exchange routines for the Python driver.

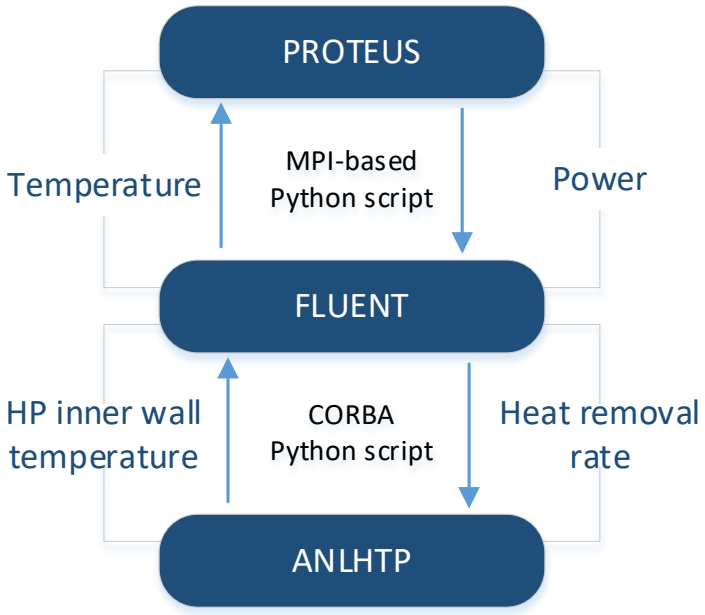


Figure 3-1. Overview of Data Exchange of PROTEUS/FLUENT/ANLHTP

In addition, we found that the ANSYS/FLUENT mesh exported in the Exodus format is difficult to be made compatible with PROTEUS because it is complicated to map region names (too many regions to manage) with compositions and identify boundary surfaces for setting up boundary conditions. In this study, therefore, meshes are constructed for each of PROTEUS and FLUENT, transferring data between the two codes based on their own meshes and requiring the

interpolation from one to the other meshes. FLUENT interpolates the power data transferred from PROTEUS using its built-in interpolation module, whereas the PROTEUS Python driver interpolates the temperature data given from FLUENT to provide the interpolated data to PROTEUS. The MPI-based Python makes it possible to transfer the interpolated data to PROTEUS via a memory instead of a file, once data are read and interpolated by the Python driver.

The coupling of PROTEUS and FLUENT is controlled by two separate Python drivers. The PROTEUS Python driver controls the overall system as well as PROTEUS, while the FLUENT Python driver controls the coupling of FLUENT and ANLHTP. The two Python drivers properly coordinate the simultaneous execution of the three codes to perform the workflow of coupled simulation, running or holding a code for a certain segment of calculations while the other codes generate the data required for the subsequent calculations. The CORBA protocol allows us to control FLUENT externally, executing or holding the code until the input data are updated by ANLHTP (temperatures at the wick-vapor interface of heat pipes) or PROTEUS (power). Figure 3-2 shows the coupling scheme of the three codes controlled by the two Python drivers.

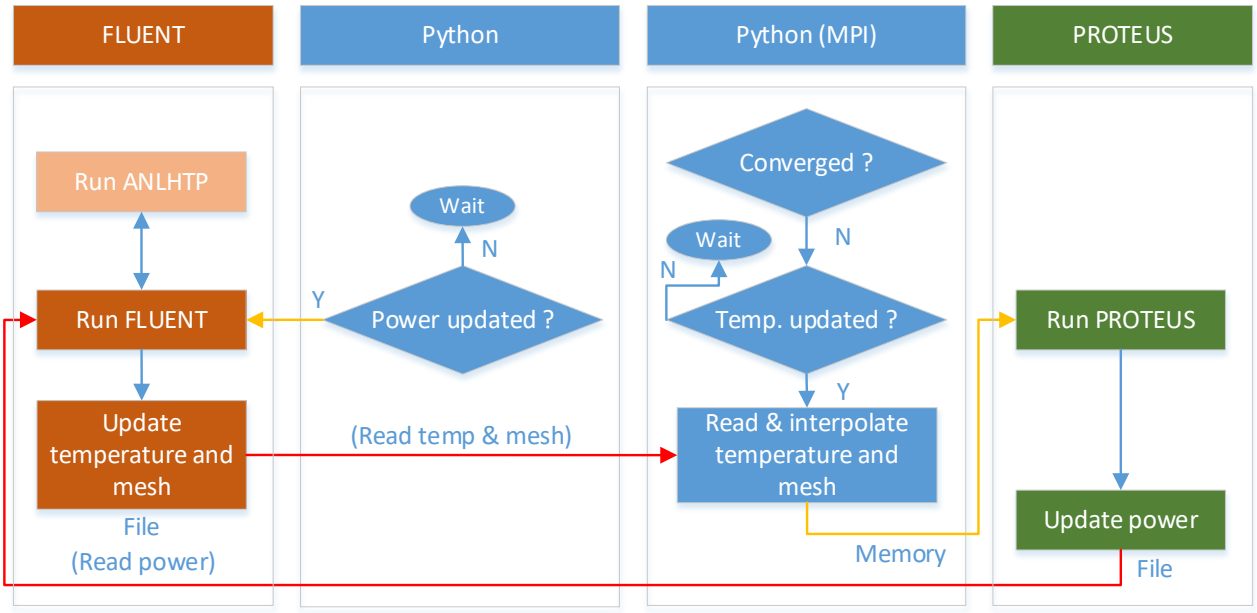


Figure 3-2. Data Exchange Flow of PROTEUS/FLUENT/ANLHTP

In the coupling of FLUENT/ANLHTP, FLUENT is used for the monolith and fuel thermal conduction analysis, while ANLHTP calculates heat pipe temperatures and operation limits. In the process of the coupling, FLUENT and ANLHTP exchange the wick-vapor interface temperature and heat removal rate from heat pipe evaporators to determine the temperature distribution of a core as shown in Figure 3-3. This implies that the FLUENT domain includes the wick and heat pipe wall. A 3D transient conduction equation is applied to these regions to consider the non-uniform axial power distribution and thermal inertia of the liquid and wick in the

evaporator. The effective thermal conductivity and heat capacity are calculated in FLUENT based on the wick geometry information.

Meanwhile, the regions analyzed by ANLHTP are the vapor core and the condenser, i.e. thermal resistances 4-11 shown in Figure 2-3. ANLHTP is the lumped parameter code unlike FLUENT and therefore, it is assumed that the wick-vapor interface temperature is uniform along the evaporator. Moreover, the thermal inertia of the vapor core is assumed to be negligible since the vapor mass in the core is relatively very small with low pressure inside the heat pipe. This approach can be defined as the quasi-steady-state model in accordance with the Zuo's transient analysis method, which showed little difference from a fully transient model.

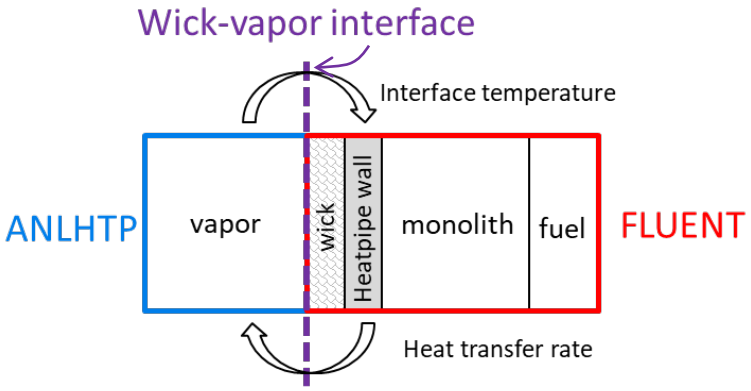


Figure 3-3. Boundary Conditions of the Coupled Simulation of FLUENT and ANLHTP

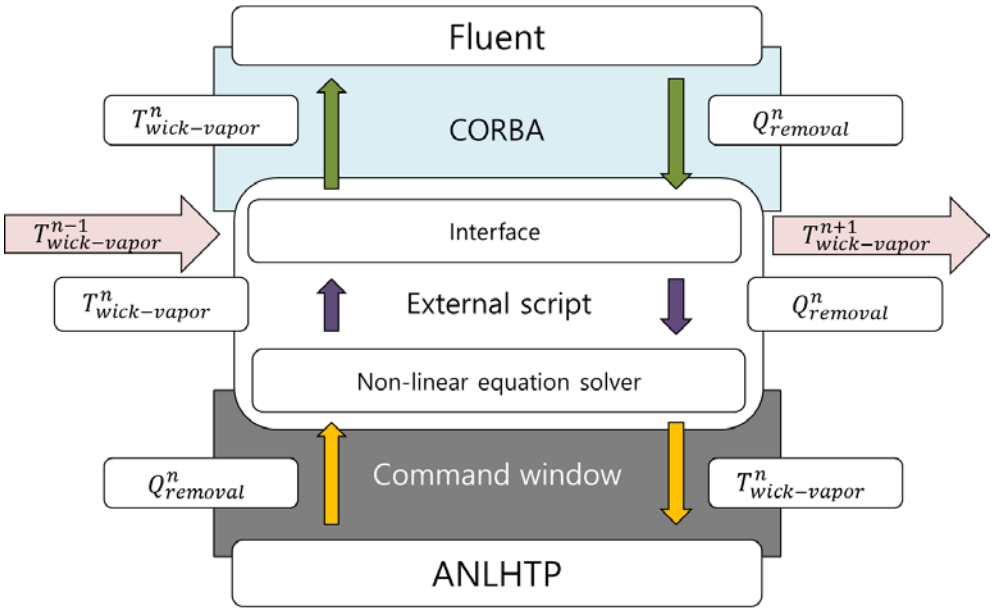


Figure 3-4. Coupled Simulation Procedure for a Single Time Step

The coupling requires data exchange between the two codes which was conducted using the function directed by "ANSYS FLUENT As a Server." This is a set of tools offering the functionality that allows local or remote client applications to access a full capability of the FLUENT solver. A client application can perform case setup, initialization, iteration, and result reporting using FLUENT as a Server interface. In the present work, a Python driver was written for the client application, controlling FLUENT using the CORBA protocol. This is a standard protocol defined by the Object Management Group (OMG) designed to facilitate the communication of systems that are deployed on diverse platforms and enables communication between codes written in different languages and running on different computers. Using the protocol, the Python driver controls the boundary conditions of the FLUENT calculation and extract the required data for the coupled calculation at every time step.

Figure 3-4 describes the coupled simulation procedure for a single time step. At the first step of the coupled calculation, initial values are imposed for the wick-vapor interface temperatures. The values are then transferred to FLUENT as boundary conditions by the Python driver. With the boundary conditions, FLUENT calculates the temperature distribution in the fuel, monolith, heat pipe wall and the wick region. The heat flux distribution is estimated based on the calculated temperature gradient on the wick-vapor interface. The calculated heat flux distribution is integrated along the wick-vapor interfaces of each heat pipe to evaluate the heat transfer rates which are transferred to ANLHTP via the Python driver for the interface temperature calculation.

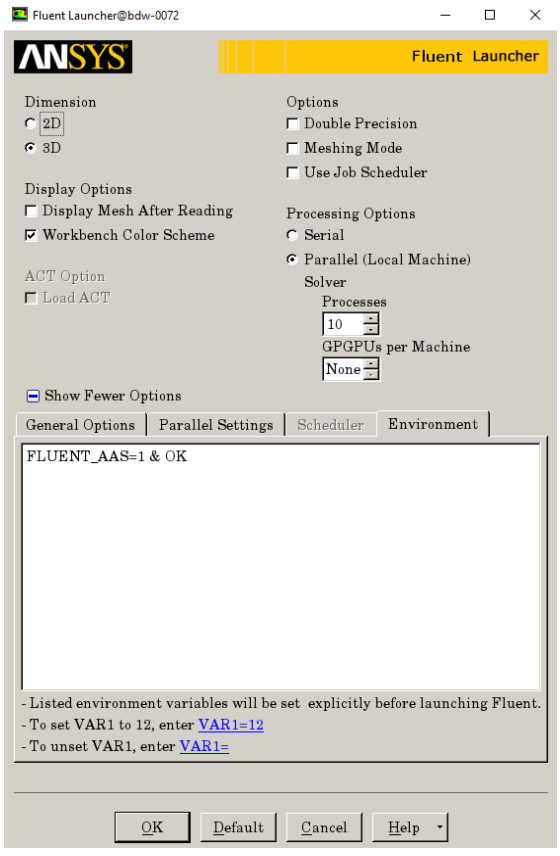


Figure 3-5. FLUENT Setting for CORBA

ANLHTP requires the wick-vapor temperature as an input and produces the heat transfer rate as an output. Thus, an iterative procedure was implemented to determine the interface temperature which satisfies the given heat removal rate. Once the solution converges, the determined wick-vapor interface temperature is transferred to FLUENT via the Python driver to repeat the conduction calculation with the updated boundary condition. This procedure is repeated at a single time step until the solution convergence is achieved. The coupling requires two iterative loops within a time step; one for ANLHTP to find the wall temperature satisfying the given wall heat transfer rate and the other to find the converged solution of the Picard iteration between the two codes. This implicit coupling method is advantageous in the transient simulation by allowing a large time step value. In the case of transient analysis, the procedure described above is repeated during the time marching.

In order to run the coupled codes, the environment of FLUENT first needs to be set up for CORBA, Python, and MPI interface for python. Next, FLUENT should be run in connection with CORBA, reading the case file that includes geometry, mesh, material, and boundary conditions of the problem of interest. Finally, the Python driver and PROTEUS should be run at the same time. The detailed steps to do are presented below:

1. Set up the MPI interface for python using Conda [28]. Conda is an open source package and environment management system that runs on different platforms. Python should be MPI-based version 2.7 or higher.

```
conda create –n “ansys_proteus_coupling” python=2.7
source activate ansys_proteus_coupling
pip install numpy
export MPICC=`which mpicc`
pip install mpi4py
source deactivate ansys_proteus_coupling
```

2. Load the CONDA environment before launching the application:

```
source activate ansys_proteus_coupling
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:{Anaconda library}
export PYTHONPATH={Python}
```

   Configure the path information of Python scripts. Set up either of the following options: add the Python path to PYTHONPATH

```
$ export PYTHONPATH=$(PYTHONPATH):/python/script/path
```

   or

```
PROTEYS_Python_Script_Path = ‘python/script/path’
if PROTEYS_Python_Script_Path is not None:
        sys.path
        sys.path.append(PROTEYS_Python_Script_Path)
```

   (Note that this is already included in the Python driver)

3. Compile the FLUENT IDL file for enabling control FLUENT through CORBA. This compile command creates new directories and files, which are used in the execution of python script and FLUENT .

```
export omniORB_PATH={omniORB}
cp {fluent}/addons/corba/lnamd64/CoFluentUnit.idl .
${omniORB_PATH}/bin/omniidl -p ${omniORB_PATH}/lib/python2.7/site-
   packages/omniidl_be -I ${omniORB_PATH}/share/idl/omniORB -bpython
   CoFluentUnit.idl
```

4. Run FLUENT after setting environment with FLUENT_AAS=1 & OK, as shown in Figure 3-5, to use CORBA and read the .cas file that contains geometry, mesh, and material property data.

```
AAS_CORBA      : directory necessary for CORBA
{case_name}.cas  : case file
wick_Cp.c        : heat capacity data for materials in the heat pipe
```

5. Inputs as well as the executable for ANLHTP are needed for the coupled calculation of ANSYS and ANLHTP

```
START      : contains the base input data for heat pipe
IN         : contains the inputs to check the limits for heat pipe
ANLHTP.x   : executable
```

6. Run the Python driver and PROTEUS simultaneously by using the following command in which a single processor should be used to run the Python drivers and multiple processors (e.g., 20 processors) can be used to run PROTEUS:

```
mpiexec -np 1 python Driver_Steady_PROTEUS_Fluent.py   : -n 20 ./mocex_mpc.x
mpiexec -np 1 python Driver_Kinetics_PROTEUS_Fluent.py : -n 20 ./mocex_kinetics_mpc.x
```

Driver_Steady_PROTEUS.py	Python script to control the steady coupled simulation
Driver_Kinetics_PROTEUS.py	Python script to control the transient coupled simulation
input.json	Inputs for the Python driver containing the control options and time steps

mocex_mpc.x	PROTEUS executable (steady)
mocex_kinetics mpc.x	PROTEUS executable (kinetics)
moex.inp	PROTEUS main input (req. for steady and kinetics)
kinetics.inp	PROTEUS kinetics input (req. only for kinetics)
{cross_section}.ISOTXS	Cross section
{cross_section}.ISOPAR	Cross section with state parameters
{fine_mesh}.ascii	Fine mesh
{coarse_mesh}.ascii	Coarse mesh (CMFD) for acceleration

{material}.assignment	Material assignment
DLAYXS	Delayed neutron data (kinetics)
kinetics.inp	PROTEUS kinetics input (kinetics)

The coupled calculation creates the following files to exchange power and temperature data between PROTEUS and FLUENT.

- MOCEX_ANSYS_Power.dat
- MOCEX_ANSYS_Temperature.dat

where MOCEX_ANSYS_Power.dat is generated from PROTEUS, containing powers (W/cm^3) with x, y, z coordinates, while MOCEX_ANSYS_Temperature.dat is created from FLUENT, including temperatures (˚F) with x, y, z coordinates. The meshes and coordinates for the two codes can be different from each other, therefore data generated from one code are converted to the other code by interpolation in the Python and FLUENT.

When running the coupled codes by conducting Step 6 in the previous page, one can see the iterations from the FLUENT screen, as shown in Figure 3-6, as well as the calculation progress from the terminal as shown in Figure 3-7 at the same time.

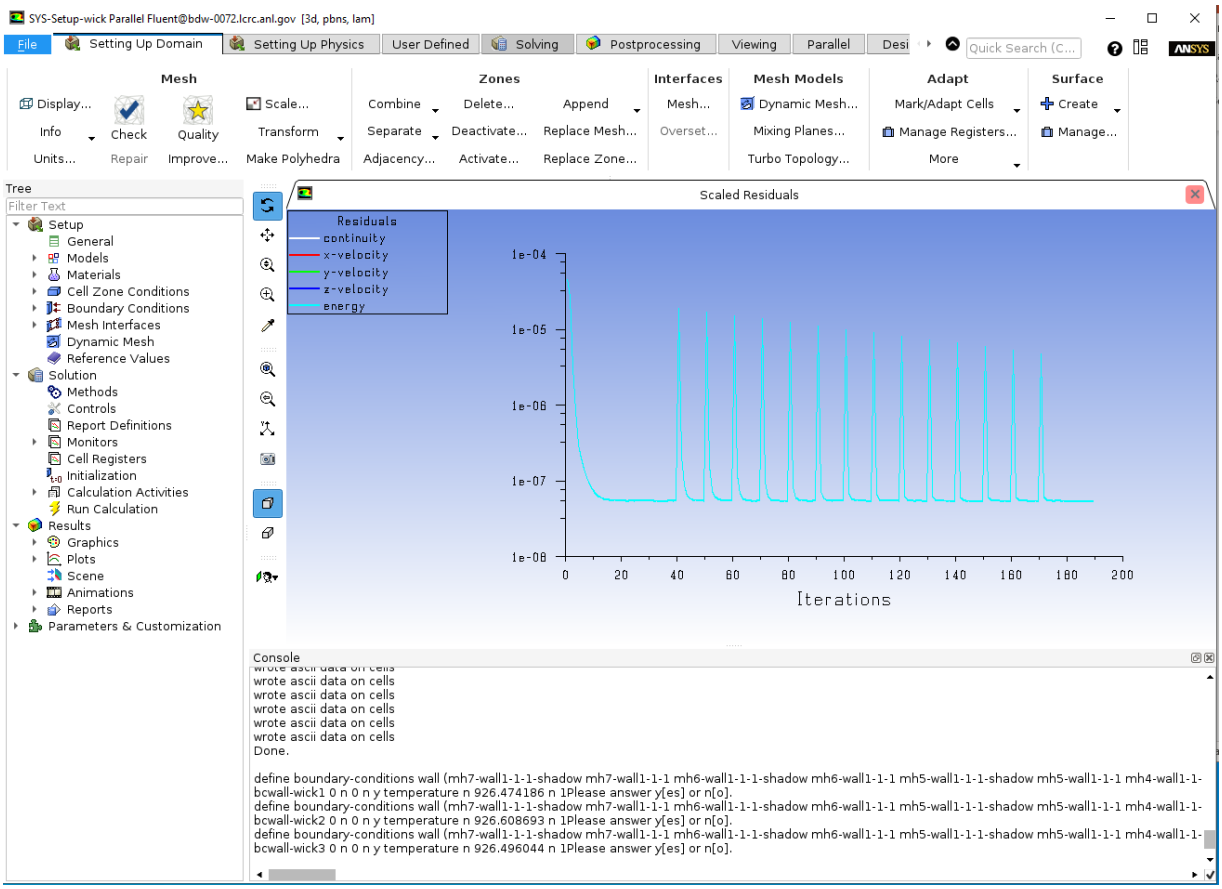


Figure 3-6. Screen Output from FLUENT

```
(mpi4py) [clee@bdw-0072 TRtest2]$ mpiexec -np 1 python Driver_Kinetics_PROTEUS.py : -n 12 ./mocex_kinetics_mpc.x
[ MOCEX ].....
[ MOCEX ]...Detected PYTHON Wrapper for Coupled Calulation.....
[ MOCEX ].....
[Input_Read]...Unknown key word.....USE_WGS_KRYLOV
[Input_Read]...USE_WGS_KRYLOV          YES
[ MOCEX ]...Successfully imported the MOCEX input (all errors reported above).....
[ MOCEX ]...Successfully imported the KINETICS input (all errors reported above).....
[ MOCEX ]...Successfully imported the ASSIGNMENT input (all errors reported above).....
[ MOCEX ]...Successfully imported the DLAYXS data into Delay_XS.....
[NTCmesh]...Importing the NTCmesh.....
[PNTmesh]...Repartitioning file to match the              1 decomposition.....
[ MOCEX ]...The total number of processors available is .....          12
[ MOCEX ]...The       1015 - spatial vertices - are chopped into      1 processor communication segments.....
[ MOCEX ]...The          9 -  energy groups   - are chopped into      1 processor communication segments.....
[ MOCEX ]...The         96 - angular moments  - are chopped into     12 processor communication segments.....
[ MOCEX ]...The         15 -   axial planes   - are chopped into      1 processor communication segments.....
[PNTmesh]...Successfully read the decomposition into        1.....
[PNTmesh]...        1015 vertices split into      1 max         1015 min         1015 RMS =        1015 AVG =       1015
[PNTmesh]...        1080 elements split into      1 max         1080 min         1080 RMS =        1080 AVG =       1080
[PNTmesh]...Successfully read the decomposition into        1.....
[PNTmesh]...          37 vertices split into      1 max           37 min           37 RMS =          37 AVG =         37
[PNTmesh]...          54 elements split into      1 max           54 min           54 RMS =          54 AVG =         54
[ MOCEX ]...Neutron transport spatial mesh data was imported and the alias names were applied.....
```

(Reduced)

```
= MOCEX =.....BEGINNING OF EIGENVALUE SOLVE.....
= MOCEX =...|             Fission Source Iteration                        |    Within-group Source Iteration   |.....
= MOCEX =...| Seconds|Itr| Eigenvalue | Error  |Fiss Err|FluxRErr| Dom  |  Error  Grp|  K  |Max  G |Min  G |.....
 WGS-KRYLOV |     3.8|001| 1.17780E+00| 1.8E-01| 2.9E-01| 1.5E+03| 0.000|1.5E+03    9|   41|  6    3|  3    1|.....
+MGCMFD-ACC+|     0.7|004| 1.22176E+00| 2.2E-01| 1.6E-02| 2.4E-03| 0.000|2.4E-03    0|  168| 41    0| 41    0|.....
 WGS-KRYLOV |     2.5|002| 1.22689E+00| 4.2E-03| 3.5E-02| 6.4E+02| 0.000|6.4E+02    9|   23|  3    3|  1    9|.....
+MGCMFD-ACC+|     1.0|006| 1.23296E+00| 9.2E-03| 2.7E-03| 4.4E-04| 0.000|4.4E-04    0|  236| 40    0| 40    0|.....
 WGS-KRYLOV |     2.5|003| 1.23939E+00| 5.2E-03| 2.2E-02| 3.8E+02| 0.340|3.8E+02    9|   23|  3    3|  1    9|.....
+MGCMFD-ACC+|     0.8|006| 1.24152E+00| 6.9E-03| 1.5E-03| 2.2E-04| 0.000|2.2E-04    0|  203| 30    0| 30    0|.....
 WGS-KRYLOV |     2.6|004| 1.24554E+00| 3.2E-03| 1.3E-02| 3.8E+01| 1.335|3.8E+01    9|   24|  3    3|  2    1|.....
+MGCMFD-ACC+|     0.7|006| 1.24611E+00| 3.7E-03| 8.5E-04| 1.2E-04| 0.000|1.2E-04    0|  169| 47    0| 47    0|.....
 WGS-KRYLOV |     2.6|005| 1.24833E+00| 1.8E-03| 7.9E-03| 2.7E+01| 0.032|2.7E+01    9|   24|  3    3|  2    1|.....
+MGCMFD-ACC+|     0.6|006| 1.24815E+00| 1.6E-03| 5.1E-04| 6.5E-05| 0.000|6.5E-05    0|  145| 26    0| 26    0|.....
[ MOCEX ]...!!!FAILURE!!! Reached specified iteration limit.....
[ MOCEX ]...!!!FAILURE!!! Could not converge within the specified criteria.....
[ MOCEX ]...Final eigenvalue.....      1.24815374
[ MOCEX ]...Final error on the eigenvalue.....      0.00164184
[ MOCEX ]...Final error on the fission source.....      0.00792742
[ MOCEX ]...Final error on the multi-group flux solution.....     27.11107495
[ MOCEX ]...Total number of fission source iterations.....              5
[ MOCEX ]...Total number of multi-group  Krylov subspace iterations.....              0
[ MOCEX ]...Total number of multi-group  preconditioner  iterations.....              0
[ MOCEX ].....
[ MOCEX ]...Waiting for Python Coupling Wrapper (PyTEUS).....
[ MOCEX ].....
[ MOCEX ]...Successfully exported the MOCEX solutions for coupling.....
[ MOCEX ].....
[ MOCEX ]...Waiting for Python Coupling Wrapper (PyTEUS).....
[ MOCEX ].....
[ PYTEUS ].Running ANSYS/ANLHTP.....
[ PYTEUS ].Successfully completed ANSYS/ANLHTP calculation.....
[ PYTEUS ].Converting  ANSYS temperature output to PROTEUS format.....
[ PYTEUS ].Successfully converted ANSYS temperature output.....
[ MOCEX ].....
[ MOCEX ]...Successfully Imported ANSYS Temperature data.....
[ MOCEX ].....
= MOCEX =.....BEGINNING OF EIGENVALUE SOLVE.....
= MOCEX =...|             Fission Source Iteration                        |    Within-group Source Iteration   |.....
= MOCEX =...| Seconds|Itr| Eigenvalue | Error  |Fiss Err|FluxRErr| Dom  |  Error  Grp|  K  |Max  G |Min  G |.....
 WGS-KRYLOV |     3.5|001| 1.25085E+00| 2.2E-03| 7.5E-03| 9.2E-02| 0.000|9.2E-02    9|   38|  5    3|  3    1|.....
+MGCMFD-ACC+|     0.6|006| 1.25021E+00| 1.6E-03| 3.2E-04| 4.3E-05| 0.000|4.3E-05    0|  144| 25    0| 25    0|.....
 WGS-KRYLOV |     2.3|002| 1.25066E+00| 3.6E-04| 1.8E-03| 4.9E-02| 0.000|4.9E-02    9|   20|  3    3|  2    1|.....
+MGCMFD-ACC+|     0.6|006| 1.25069E+00| 3.9E-04| 1.4E-04| 2.3E-05| 0.000|2.3E-05    0|  142| 24    0| 24    0|.....
 WGS-KRYLOV |     2.6|003| 1.25119E+00| 4.0E-04| 1.8E-03| 1.5E-02| 0.781|1.5E-02    9|   24|  3    3|  2    1|.....
+MGCMFD-ACC+|     0.6|006| 1.25107E+00| 3.0E-04| 1.0E-04| 1.4E-05| 0.000|1.4E-05    0|  145| 26    0| 26    0|.....
 WGS-KRYLOV |     2.6|004| 1.25132E+00| 2.0E-04| 9.5E-04| 2.8E-03| 0.376|2.8E-03    9|   24|  3    3|  2    1|.....
+MGCMFD-ACC+|     0.6|006| 1.25126E+00| 1.6E-04| 5.8E-05| 9.6E-06| 0.000|9.6E-06    0|  142| 24    0| 24    0|.....
 WGS-KRYLOV |     2.6|005| 1.25142E+00| 1.3E-04| 6.1E-04| 9.9E-03| 0.562|9.9E-03    9|   24|  3    3|  2    1|.....
+MGCMFD-ACC+|     0.6|006| 1.25138E+00| 9.5E-05| 3.7E-05| 6.0E-06| 0.000|6.0E-06    0|  145| 25    0| 25    0|.....
[ MOCEX ]...!!!FAILURE!!! Reached specified iteration limit.....
[ MOCEX ]...!!!FAILURE!!! Could not converge within the specified criteria.....
[ MOCEX ]...Final eigenvalue.....      1.25138069
```

Figure 3-7. Screen Outputs from the Coupled Calculation of PROTEUS/FLUENT/ANLHTP

3.2.2 PROTEUS/ANSYS-Mechanical/ANLHTP

3.2.2.1 Using ANSYS Workbench

ANSYS Workbench is a project management tool, the top-level interface linking all software tools, handling the passing of data between geometry, mesh, solver, and postprocessing tools. It makes it easy for users to perform design studies for design optimization.

Basically, a Python controlling of the PROTEUS/ANSYS coupled system is similar to that of PROTEUS/FLUENT in terms of controlling the codes, exchanging data through files and checking the update status of files as well as solution convergence. However, using ANSYS Workbench, the Python driver for ANSYS can be executed within it as shown in Figure 3-8.

As below, Python commands for Workbench scripting were used to reread the data file and rerun the case, which worked well to solve a steady-state problem.

```
System1 = GetSystem(Name = “SYS 1”)
Setup1 = System1.GetContainer(ComponentName = “Setup”)
Setup1.RereadDataFiles()
Update()
```

We have not found Python commands for controlling a transient simulation of ANSYS Mechanical within Workbench. Based on many ANSYS expert consultations, we concluded at the moment that Workbench Python scripting does not provide a good control of transient simulation yet.

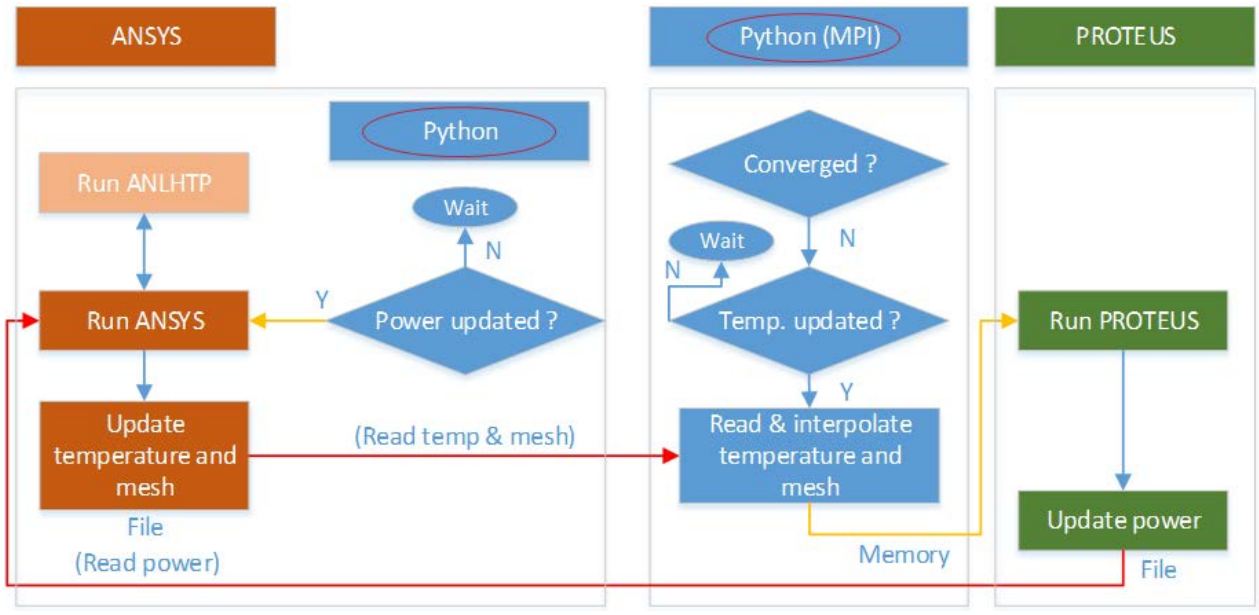


Figure 3-8. Data Exchange of PROTEUS/ANSYS Workbench/ANLHTP

1. Set up the environment with Conda and Python. As previously mentioned, Conda is an open source package and environment management system that runs on different platforms. Python should be MPI-based version 2.7 or higher.

```
conda create –n “ansys_proteus_coupling” python=2.7
source activate ansys_proteus_coupling
pip install numpy
export MPICC=`which mpicc`
pip install mpi4py
source deactivate ansys_proteus_coupling
```

   Load the CONDA environment before launching the application:

```
$ source activate ansys_proteus_coupling
```

   Configure the path information of Python scripts. Set up either of the following options: add the Python path to PYTHONPATH

```
$ export PYTHONPATH=$(PYTHONPATH):/python/script/path
```

   or

```
PROTEYS_Python_Script_Path = ‘python/script/path’
if PROTEYS_Python_Script_Path is not None:
        sys.path
        sys.path.append(PROTEYS_Python_Script_Path)
```

   (Note that this is already included in the Python driver)

2. Run PROTEUS and Python driver at the same time.

```
$ mpiexec -n  1  Python  Driver_PROTEUS.py : -n {# of processors} mocex_mpc.x
```

3. Run ANSYS at the same directory where PROTEUS is running, open the Workbench project file (.wboj), and run Driver_ANSYSWB.py from Workbench (file → scripting → run script file.

The two Python drivers (one for PROTEUS and the other for ANSYS) have PROTEUS and ANSYS communicate with each other, checking the update status of the power (MOCEX_ANSYS_Power.dat generated from PROTEUS) and temperature (coordinate_temperatures.dat generated from ANSYS) files and running the codes when the files are updated. The iterative calculation is terminated when all solutions including eigenvalue,

fission source, and temperature meet the convergence criteria which are set in an input file of the Python drivers, named "input.json" by default.

PROTEUS reads the x, y, z coordinates and temperature data of the file named coordinate_temperatures.dat generated from ANSYS, which are interpolated for the PROTEUS mesh coordinates using the interpolation function called by the Python driver and stored in MOCEX_ANSYS_Temperature.dat. ANSYS reads the x, y, z coordinates and power data of the file named MOCEX_ANSYS_Power.dat generated from PROTEUS, which are interpolated for the ANSYS mesh using the ANSYS internal interpolation function. Parts of the data file examples, MOCEX_ANSYS_Power.dat and MOCEX_ANSYS_Temperature.dat, are shown in Figure 3-9 and Figure 3-10, respectively.

X	Y	Z	Power in Watt
-1.2167617	1.3126858	4.5000000	6.382918E+01
-1.2167617	1.3126858	13.5000000	1.538284E+02
-1.2167617	1.3126858	22.5000000	2.335696E+02
-1.2167617	1.3126858	31.5000000	3.037799E+02
-1.2167617	1.3126858	40.5000000	3.627458E+02
-1.2167617	1.3126858	49.5000000	4.099685E+02
-1.2167617	1.3126858	58.5000000	4.453614E+02
...			
...			

Figure 3-9. Example of the Power File Generated from PROTEUS (MOCEX_ANSYS_Power.dat)

X	Y	Z	Temperature in C
-0.4948543E-03	-0.3786812E-03	0.7409567E-01	0.5969878E+03
-0.1660746E-02	-0.6996087E-03	0.7409551E-01	0.5950966E+03
-0.2586870E-03	0.1936172E-04	0.7409572E-01	0.5970931E+03
-0.4036574E-03	-0.1329166E-03	0.7409570E-01	0.5970882E+03
0.9620942E-04	-0.8184958E-04	0.7409574E-01	0.5970618E+03
-0.9883580E-03	0.1914008E-03	0.7409566E-01	0.5964950E+03
-0.5712671E-03	-0.6687563E-03	0.7409563E-01	0.5967685E+03
...			
...			

Figure 3-10. Example of the Temperature File Generated from ANSYS (coordinate_temperatures.dat)

When running the coupled codes by conducting the steps discussed in the previous page, one can see the calculation progress from the terminals, as shown in Figure 3-11. It is recommended to use two terminals: one for PROTEUS and the other for ANSYS because a Python driver for each code will write its own calculation progress to the terminal. If one uses a single terminal, then the progress information would be mixed up. Figure 3-12 illustrates how ANSYS Workbench works for the coupled simulation.

As aforementioned, it was concluded that the current version ANSYS Workbench with Python scripting does not provide flexibilities and capabilities of controlling a transient simulation for the coupled simulation with an external code. Therefore, we pursued an approach with ANSYS APDL for a transient simulation in coupling with PROTEUS. This is discussed in the following section.

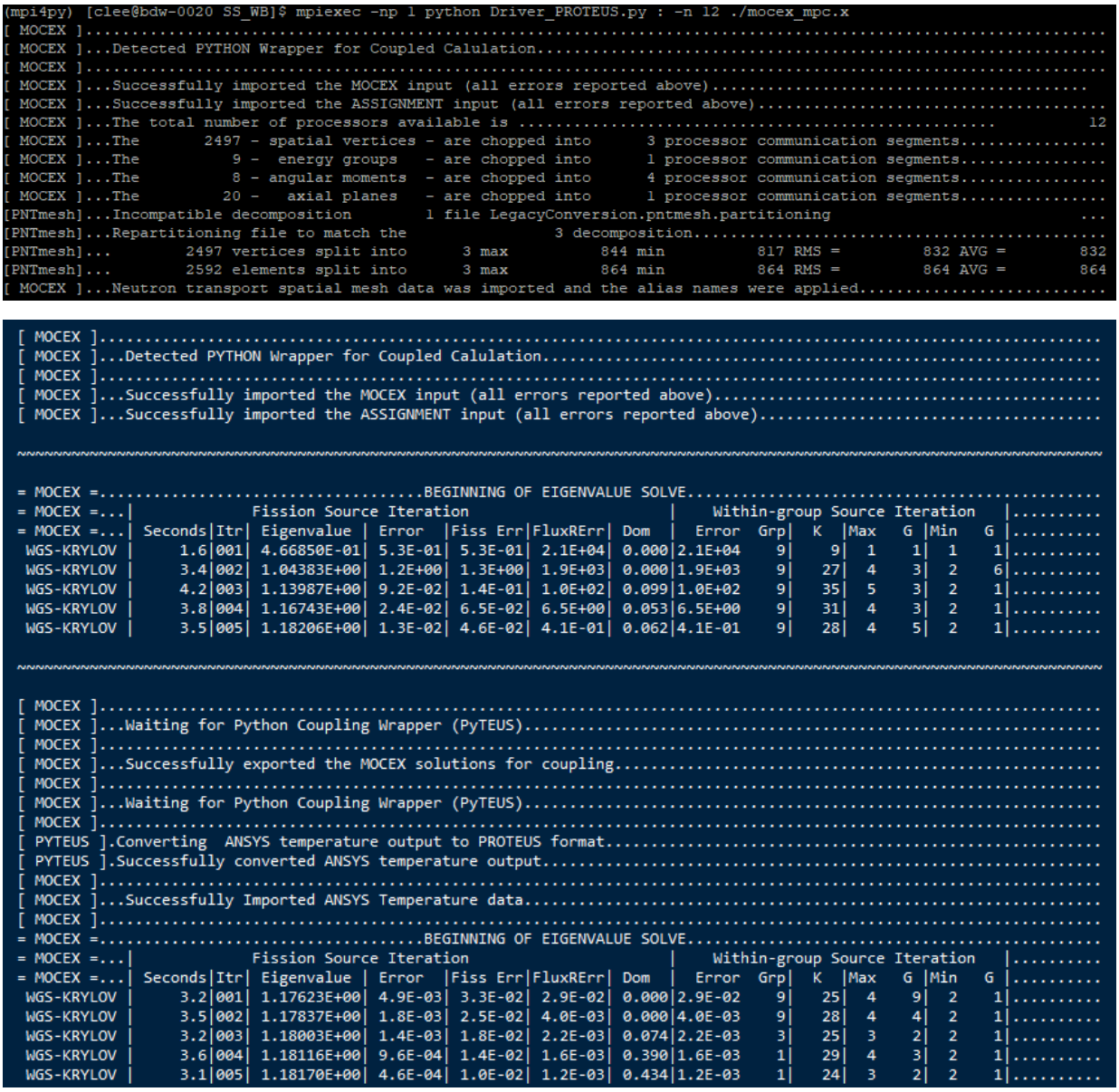
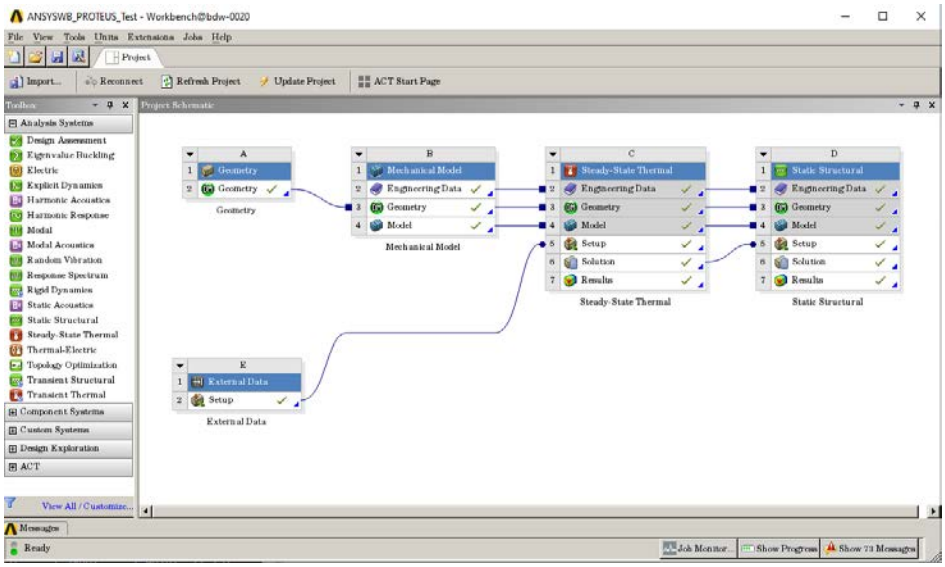


Figure 3-11. Screen Output of ANSYS Workbench for the Steady-state Problem




```
[ANSYSWB] Job started
[ANSYSWB]
[ANSYSWB] Wait for PROTEUS Launch and Intialization......
[ANSYSWB] Done
[ANSYSWB]================================================================================
[ANSYSWB] PROTEUS-ANSYSWB Iteration 001
[ANSYSWB] Wait for PROTEUS Transport Calculation......
[ANSYSWB] Done
[ANSYSWB]
[ANSYSWB] Perform S.S Thermal and Mechanical Analysis.....
[ANSYSWB] Done
[ANSYSWB]--------------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[ANSYSWB]--------------------------------------------------------------------------------
[ANSYSWB] PROTEUS-ANSYSWB Iteration 004
[ANSYSWB] Wait for PROTEUS Transport Calculation......
[ANSYSWB] Done
[ANSYSWB]
[ANSYSWB] Completion of PROTEUS Execution is detected
[ANSYSWB]================================================================================
[ANSYSWB] Job Completed
```

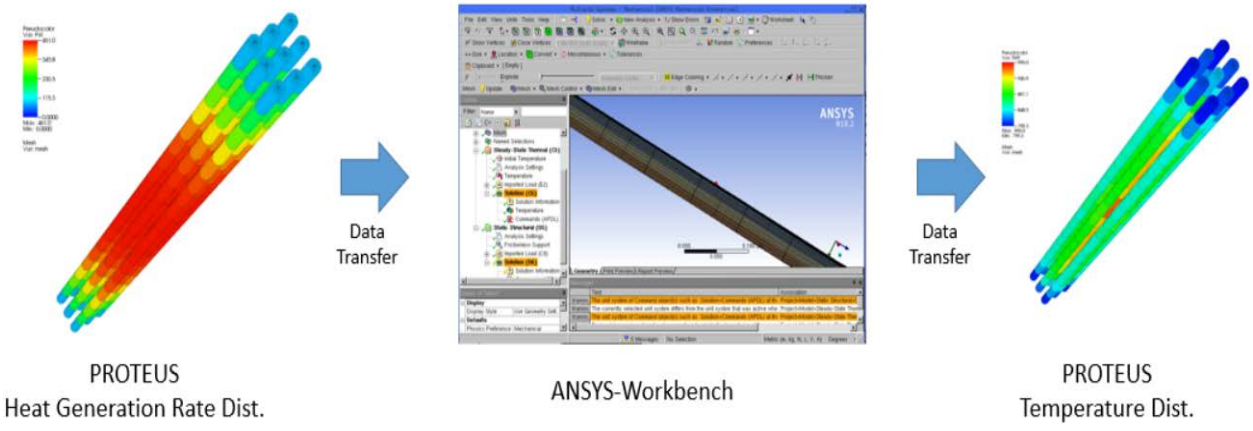


Figure 3-12. Screen Output of ANSYS Workbench for the Steady-state Problem

3.2.2.2 Using ANSYS APDL

A similar approach as used in the coupling with FLUENT via CORBA was adopted for the coupling with ANSYS with APDL. Basically, ANSYS-APDL replaced the heat conduction solver embedded in FLUENT, additionally performing structure analysis to account for geometry deformation. The coupling with PROTEUS was made directly through the APDL solver without employing the ANSYS-Workbench platform. Since the functionality of power profile interpolation available through Workbench was not found, the power profile generated from PROTEUS was interpolated in the Python driver and then assigned to individual solid elements in ANSYS using the APDL commands.

The followings present the steps in details for setting up the environment for and running the coupled calculation.

1. Set up the MPI interface for python using Conda which is the same process as used in the FLUENT coupling.

```
conda create –n “ansys_proteus_coupling” python=2.7
source activate ansys_proteus_coupling
pip install numpy
export MPICC=`which mpicc`
pip install mpi4py
source deactivate ansys_proteus_coupling
```

2. Load the CONDA environment before launching the application.

```
source activate ansys_proteus_coupling
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:{Anaconda library}
export PYTHONPATH={Python}
```

   Configure the path information of Python scripts. Set up either of the following options: add the Python path to PYTHONPATH

```
$ export PYTHONPATH=$(PYTHONPATH):/python/script/path
```

   or include below to the Python driver.

```
PROTEYS_Python_Script_Path = ‘python/script/path’
if PROTEYS_Python_Script_Path is not None:
        sys.path
        sys.path.append(PROTEYS_Python_Script_Path)
```

   (Note that this is already included in the Python driver)

3. Compile the APDL IDL file for enabling controlling FLUENT through CORBA. This compilation command creates new directories (AAS_CORBA and AAS_CORBA_POA) and

files (CoFluentUnit_idl and CoFluentUnit_idl.py) which are used in the execution of Python script and ADPL.

```
export omniORB_PATH={omniORB}
cp {fluent customize/user/ICoMapdlUnit.idl .
${omniORB_PATH}/bin/omniidl -p ${omniORB_PATH}/lib/python2.7/site-
   packages/omniidl_be -I ${omniORB_PATH}/share/idl/omniORB -bpython
   ICoMapdlUnit
```

4. Launch APDL after setting environment with the -AAS option as shown in Figure 3-13. The data file containing the problem information and corresponding ADPL commands are needed for the coupled calculations. Note that the APDL commands are executed via the ANSYS ADPL launcher without graphic user interface, which is located at the following directory of the ANSYS Mechanical package: {ansys directory}/ansys_inc/v192/ansys/bin/launcher.

5. The following inputs as well as the executable for ANLHTP are needed for the coupled calculation of ANSYS and ANLHTP

START	Contains the base input data for heat pipe
IN	Contains the inputs to check the limits for heat pipe
ANLHTP.x	Executable

6. Run the Python driver and PROTEUS simultaneously by using the following commands in which a single processor should be used to run the Python drivers and multiple processors (e.g., 20 processors) can be used to run PROTEUS.

```
mpiexec -np 1 python Driver_Steady_PROTEUS_APDL.py    : -n 20 ./mocex_mpc.x
mpiexec -np 1 python Driver_Kinetics_PROTEUS_APDL.py : -n 20 ./mocex_kinetics_mpc.x
```

Driver_Steady_PROTEUS_APDL.py	Python script to control the steady coupled simulation
Driver_Kinetics_PROTEUS_APLD.py	Python script to control the transient coupled simulation
input.json	Inputs for the Python driver containing the control options and time steps

mocex_mpc.x	PROTEUS executable (steady)
mocex_kinetics mpc.x	PROTEUS executable (kinetics)
moex.inp	PROTEUS main input (req. for steady and kinetics)
kinetics.inp	PROTEUS kinetics input (req. only for kinetics)
{cross_section}.ISOTXS	Cross section
{cross_section}.ISOPAR	Cross section with state parameters
{fine_mesh}.ascii	Fine mesh
{coarse_mesh}.ascii	Coarse mesh (CMFD) for acceleration

{material}.assignment	Material assignment
DLAYXS	Delayed neutron data (kinetics)
kinetics.inp	PROTEUS kinetics input (kinetics)

The simulation progress outputs of PROTEUS and ANSYS for steady-state and transient calculations are displayed at the terminal as shown in Figure 3-14 and Figure 3-15, respectively. The standard output files from the two codes are stored in the working directory as well.

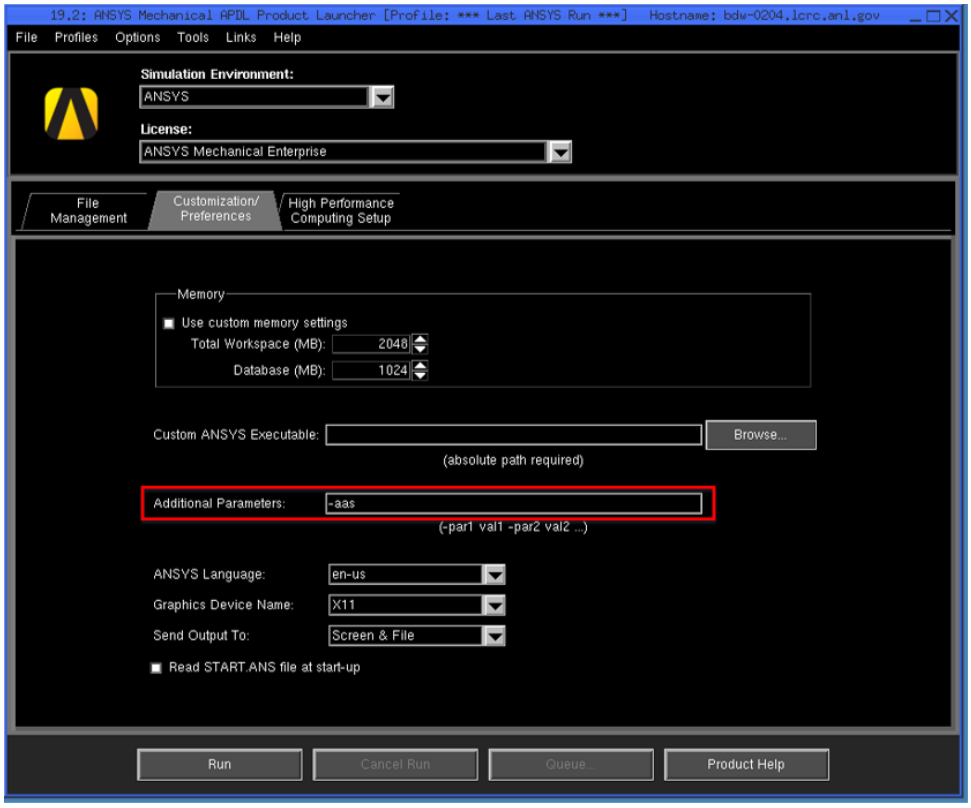


Figure 3-13. ANSYS-APDL Setting for CORBA

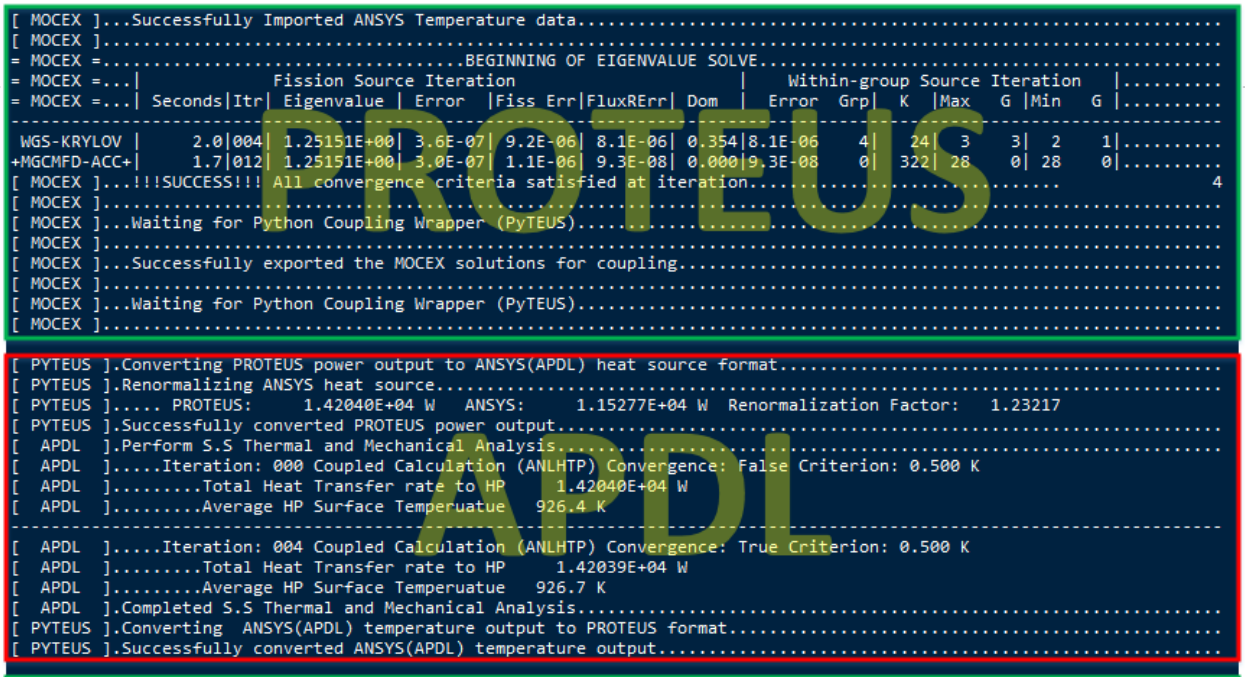


Figure 3-14. Screen Output of PROTEUS/ANSYS-APDL/ANLHTP for the Steady-state Problem

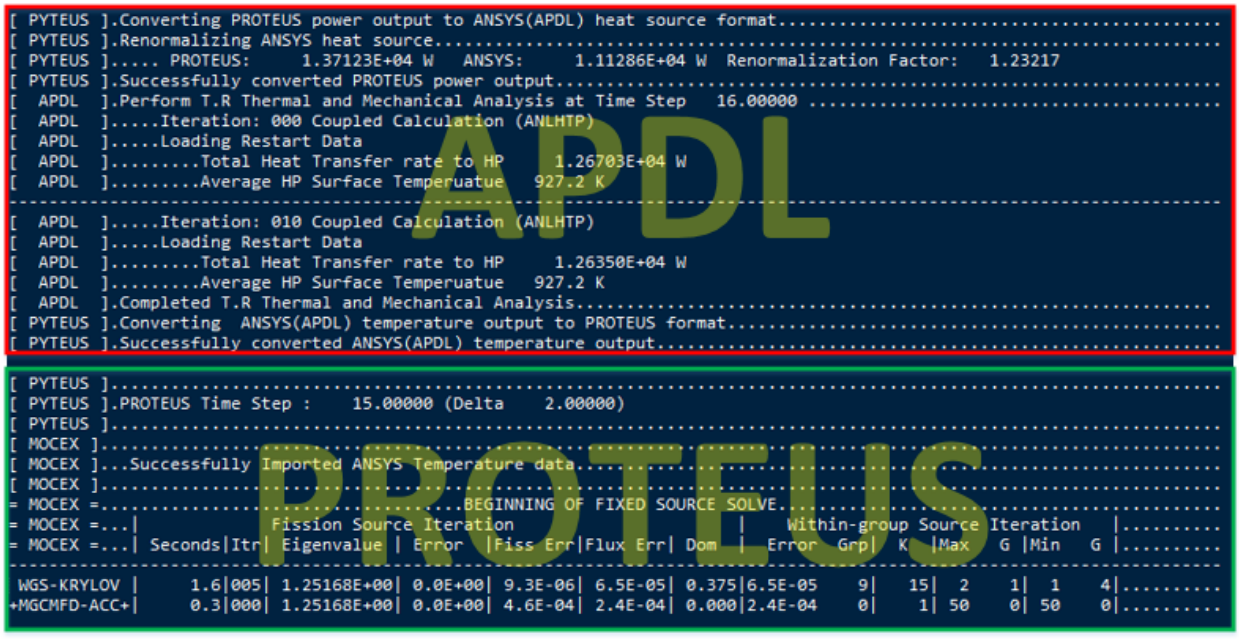


Figure 3-15. Screen Output of PROTEUS/ANSYS-APDL/ANLHTP for the Transient Problem

3.3 Coupled Calculation Results

3.3.1 Steady-state Problem

Single Heat Pipe Problem

As the first verification of the coupling, a single heat pipe conceptual problem was defined, which was surrounded by monolith and six fuel rods. In this single heat pipe problem, only one iteration step is necessary between FLUENT and ANLHTP as all the heat generated from the fuel rods is supposed to be removed by the heat pipe. Therefore, the objective of this conceptual problem is to check the coupling process and confirm the agreement in the heat pipe outer wall temperatures between the two codes.

The geometry and mesh scheme for this conceptual problem are shown in Figure 3-16. The detail dimensions were obtained from the core configuration of MegaPower. 88,260 elements were used with 30 layers in the axial direction. An axially cosine-shaped power distribution with total 4,680 W was applied for the core region, which was obtained from PROTEUS.

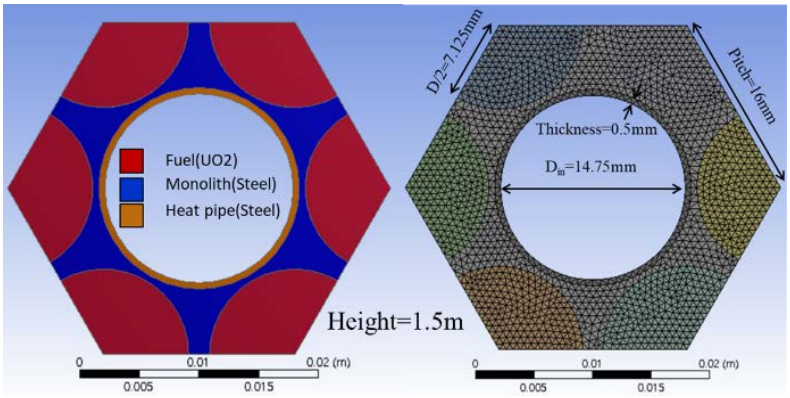


Figure 3-16. Geometry and Mesh Scheme of Single Heat Pipe Problem

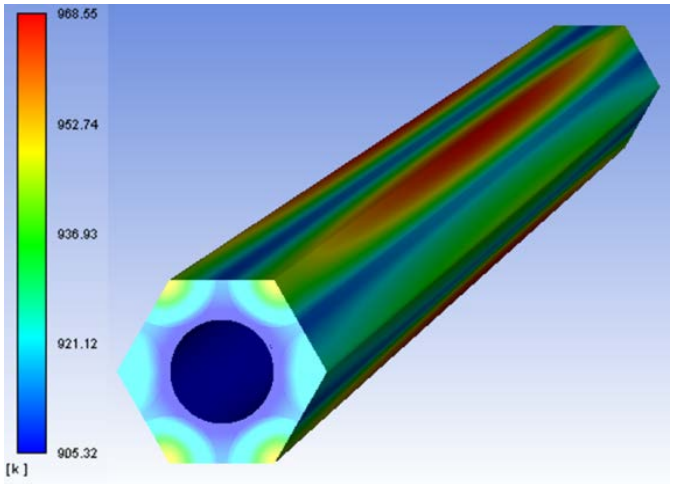


Figure 3-17. Temperature Distribution for Single Heat Pipe Problem

The calculated temperature distribution are shown in Figure 3-17. The result shows that the temperature distribution corresponding to the cosine shaped power distribution was properly calculated. The outer wall temperature calculated from ANLHTP was compared with the area averaged temperature along the heat pipe outer wall in FLUENT. The resulting ANLHTP and FLUENT temperatures were almost the same as each other, 906.55K and 906.56K, respectively, indicating that the energy conservation between the two codes was almost satisfied.

Multi Heat Pipe Problem

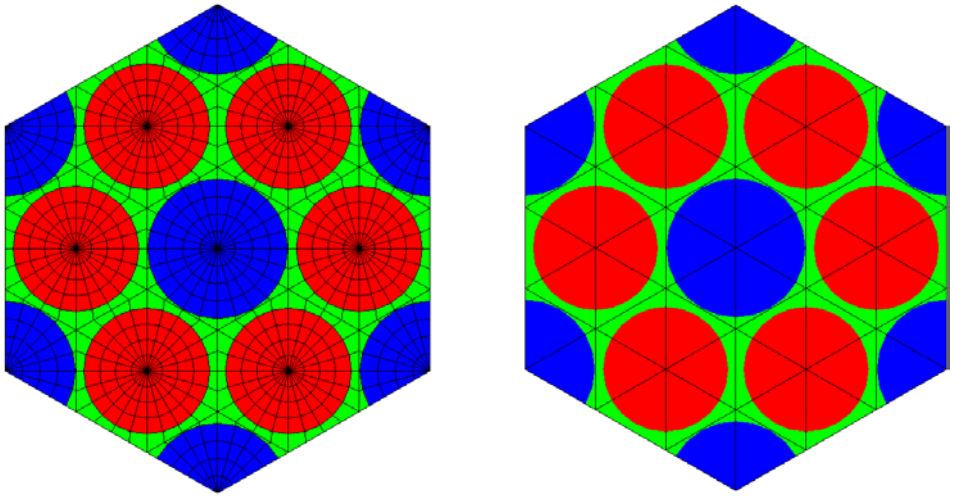
A heat pipe conceptual unit assembly problem was developed based on the core configuration of MegaPower was analyzed to verify the coupled system. In this conceptual problem, six fuel rods were surrounded by seven heat pipes in the monolith. The powers for six fuel rods were given as shown in Table 3-1, and the meshes and geometry used in the PROTEUS and FLUENT calculations are shown in Figure 3-18 and Figure 3-19, respectively. For PROTEUS, we normally provide a coarse mesh – often named a coarse mesh finite difference (CMFD) mesh – together to accelerate solution convergence.

The 3D problem was divided into 30 axial layers. The given total power was 14,400 W, and ~14% lower power with an axially cosine shaped profile was applied to the fuel rods 2 and 5 in order to generate asymmetric temperature distributions. Two different meshes (71,680 and 171,720 elements) were tested to confirm mesh convergence. The heat pipe inner wall temperatures obtained with the two meshes showed a difference of less than 0.1 K. Therefore, the coarser mesh was selected for the calculation.

The calculated temperature distribution and convergence of the heat pipe inner wall temperatures are shown in Figure 3-20 and Figure 3-21, respectively. The higher temperature values in the mid-plane were caused by the cosine shaped axial power distribution. The heat pipe temperatures were determined mostly by the powers generated from neighboring fuel rods. Based on given powers of the fuel rods, the inner wall temperatures of heat pipes 2 and 5 should be the highest, that of heat pipe 1 which is located at the center is to be the second highest, and those of heat pipes 3, 4, 6 and 7 should be the same lowest. Figure 3-21 shows apparently that the heat pipe temperatures converged to reasonable values, indicating that the coupled codes performed well for the steady-state calculation.

Table 3-1. Fuel Rod Powers for FLUENT/ANLHTP with the Steady-state Condition

Rod number	Power (Watt)
1	2,520
2	2,160
3	2,520
4	2,520
5	2,160
6	2,520



Fine Mesh Coarse (CMFD) Mesh

Figure 3-18. Geometry and Mesh Scheme of Multi-Heat Pipe Problem for PROTEUS

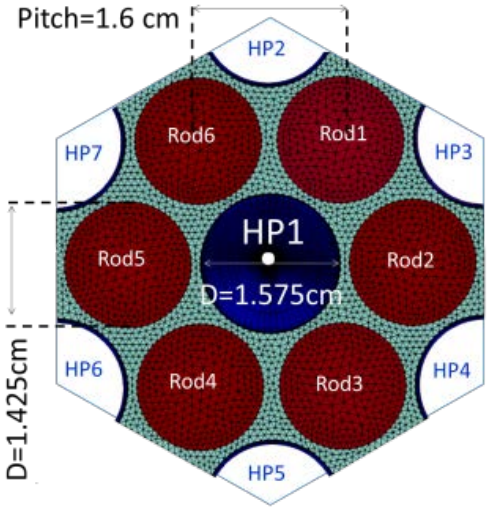
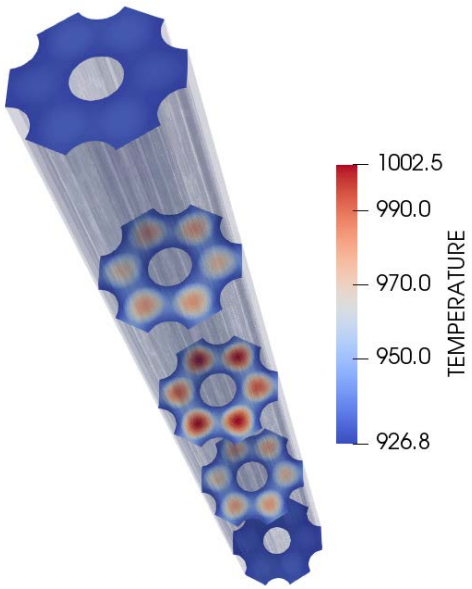


Figure 3-19. Geometry and Mesh Scheme of Multi-Heat Pipe Problem for FLUENT



(Scale ratio x : y : z = 1 : 1 : 0.25)

Figure 3-20. Temperature Distribution of Multi-Heat Pipe Problem

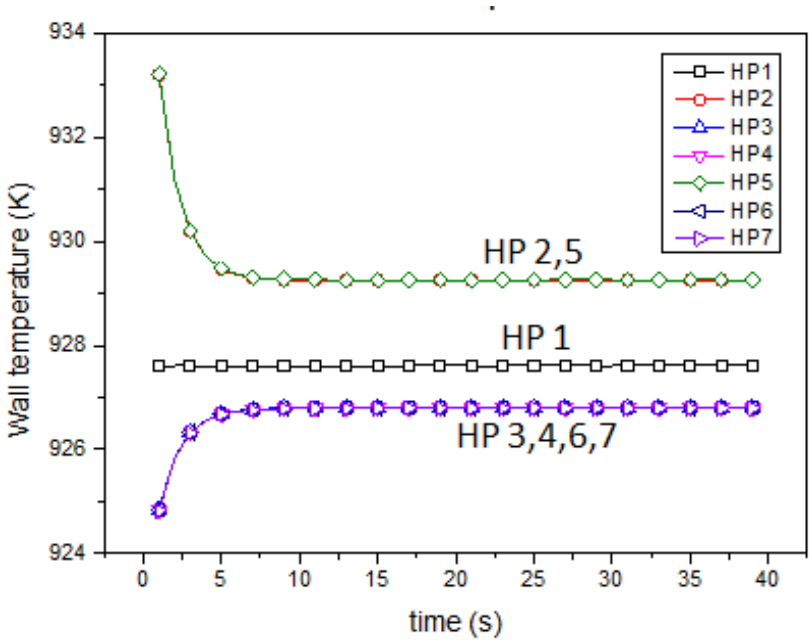


Figure 3-21. Heat Pipe Wall Temperature Convergence Result of Multi-Heat Pipe Problem

3.3.2 Transient Problem

3.3.2.1 Using PROTEUS/FLUENT/ANLHTP

The transient performance of the coupled codes was demonstrated using the 3D unit fuel assembly problem. As a postulated event of the micro reactor core, failure conditions of a heat pipe in the unit assembly were analyzed. The transient of the heat pipe failure was simulated in the following two steps: at first, it was analyzed using FLUENT/ANLHTP without accounting for neutronics feedback and then, the same condition was analyzed again using PROTEUS/FLUENT/ ANLHTP in order to consider the neutronics feedback.

Transient analysis without neutronics feedback

Initially, a steady-state calculation was conducted to find out a normal operational condition. Different from the previous case where powers are not symmetric, the symmetric power profile generated from PROTEUS was applied. The total power imposed at the steady-state condition was 13,968 W for this case. Then, transients with a single heat pipe failure were initiated from the steady-state temperature distribution. It was assumed that when a heat pipe failed, the boundary condition for the wick-vapor interface was turned to be the adiabatic wall immediately. Note that the power distribution used in the steady-state calculation remained the same during the entire transient time period: i.e., no power feedback was accounted for this test.

Two different cases of the single heat pipe failure problem were defined: heat pipe 1 (HP1) failure problem (Case-1) and HP2 failure problem (Case-2). The locations of the heat pipes are indicated in Figure 3-19. Figure 3-22 and Figure 3-23 present the resulting temperature distributions at the end of transient and the wick-vapor interface temperatures with time for Case-1, respectively. In Case-1, the failure of HP1 at the center of the assembly led to the increase of the overall temperature due to the absence of cooling source. As the monolith temperature increased, more heat was removed by the operating heat pipes, as shown in Figure 3-23. The temperatures of the intact heat pipes were evenly increased because of the symmetric locations of the intact heat pipes. The calculation was terminated at 500 sec when the temperatures reached another stable conditions.

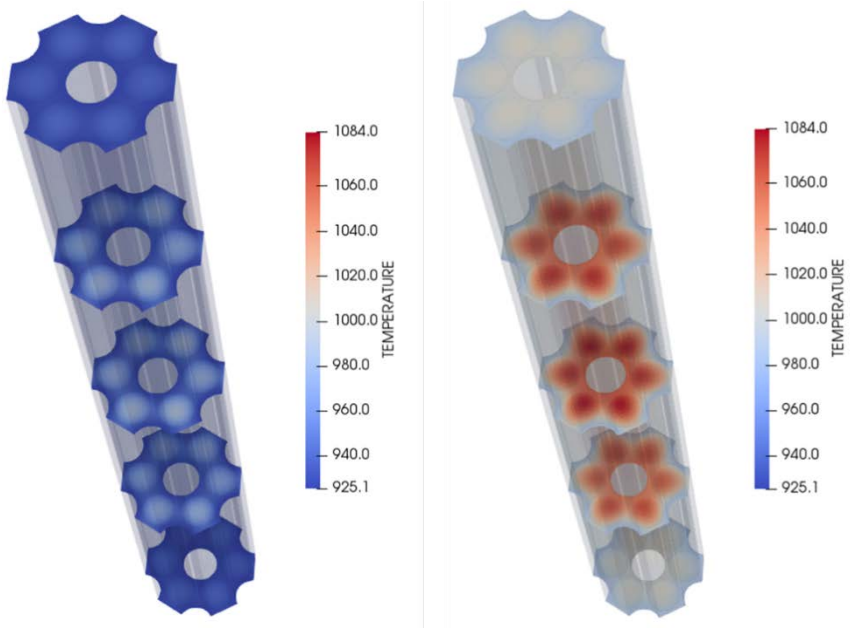
In Case-2, one of the surrounding heat pipes, HP2, was made fail. Figure 3-24 and Figure 3-25 show the results of temperature distributions and wick-vapor interface temperatures in the Case-2 simulation. As expected, a strongly asymmetric temperature distribution appeared and the maximum temperature was shifted toward the location of HP2. The maximum temperature was relatively lower than that in Case-1 because the heat removal rate of HP2 was one-third of HP1. As a result, the temperature increase of the other heat pipes in Case-2 showed less than that in Case-1 as well. HP3 and HP7 showed the largest temperature increase, as they were the closest to the failed heat pipe. HP1 has the same distance from HP2 but its surface area was three times larger than the others in the unit assembly. Thus, the temperature increase of HP1 was smaller than HP3 and HP7. The other heat pipes also showed temperature increases smaller than HP1 because they were at the farthest locations from the failed heat pipe in the assembly.

Transient analysis with neutronics feedback

In this transient simulation, the power feedback to FLUENT was accounted for using PROTEUS/FLUENT/ANLHTP. First of all, a steady-state calculation was performed using the three codes. The thermal feedback effect was accounted for in the power calculation of PROTEUS, and the power feedback was considered in the temperature calculation of FLUENT.

When the steady-state solution was converged, the transient calculation was started by initiating the HP2 failure immediately. As shown in the FLUENT/ANLHTP calculation above, the failure of HP2 led to the increase of the temperature at HP2 until ~170 sec which was propagated to the neighboring regions. However, due to the Doppler feedback, the power decreased as the fuel temperatures increased, leading to the reduction of the temperature increase rate and finally the decrease of HP temperatures starting from ~170 sec. Since the power kept decreasing, the HP temperatures decreased up to below the temperatures at the steady-state condition. As temperatures decreased below the steady-state condition, the decrease rate of power was reduced due to the negative Doppler reactivity and consequently the decrease rate of temperatures was slowed down as well. A negative reactivity feedback was initiated due to the temperature increase at the HP2 failure, whose magnitude was reduced due to the temperature decrease. As the negative reactivity was reduced to almost zero, the power and temperature of the benchmark problem were converged to another stable condition in terms of temperature and power.

Figure 3-26 shows the power and temperature changes with time after the transient started. The final temperature of HP2 became higher than the initial before the transient started and the temperatures of the other heat pipes were lower than those at the initial steady-state condition, as illustrated in Figure 3-27. As seen in the previous transient case without neutronics feedback, the temperature of HP3 and HP7 were the second highest, that of HP1 was next, and those of HP4 and HP6 were the same next, and that of HP5 was the lowest which was the farthest from the fail heat pipe. The transient results from the coupled simulation of PROTEUS/FLUENT/ANLHTP appear to be reasonable qualitatively, indicating that the coupled system was implemented correctly. Further verification tests will be conducted for different transient cases as well as larger or whole-core benchmark problems. Test results will be verified quantitatively as well.



(a) Before transient (b) After transient

Figure 3-22. Temperature Distributions of the HP1 Failure Problem Before and After Transient

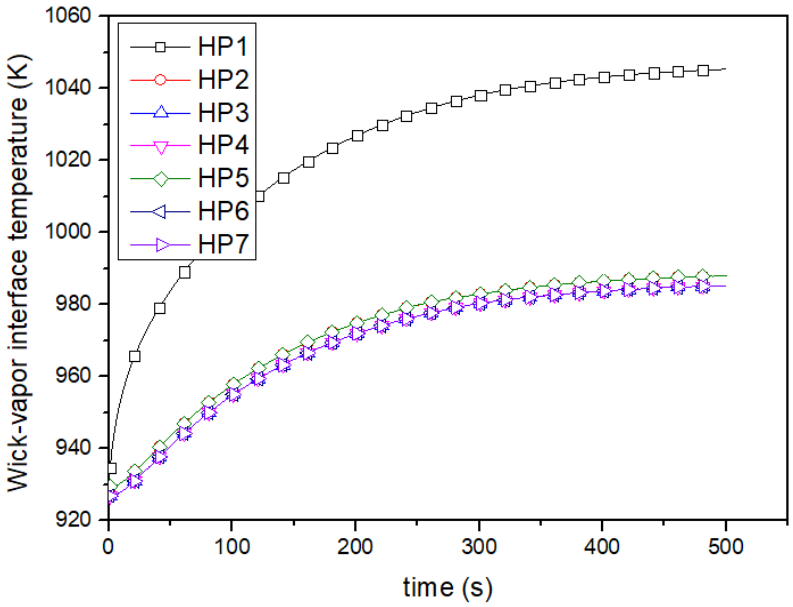
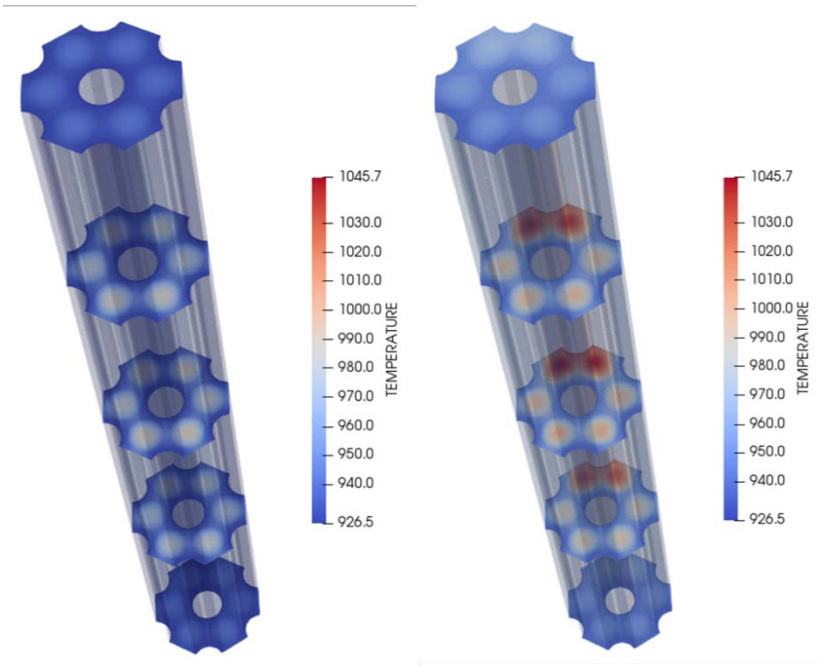


Figure 3-23. Wick-vapor Interface Temperature Transient for the HP1 Failure Problem



(a) Before transient (b) After transient

Figure 3-24. Temperature Distributions of the HP2 Failure Problem Before and After Transient

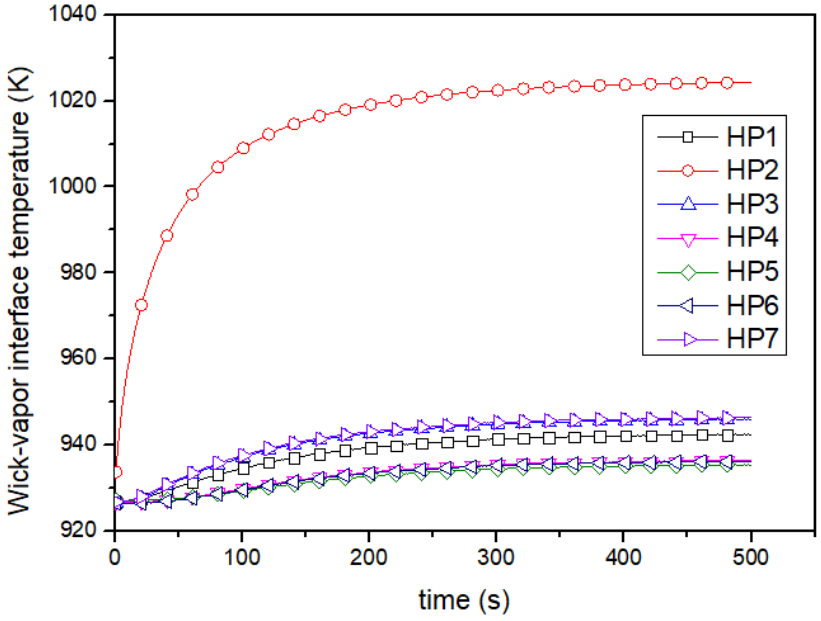


Figure 3-25. Wick-vapor Interface Temperature Transient for the HP2 Failure Problem

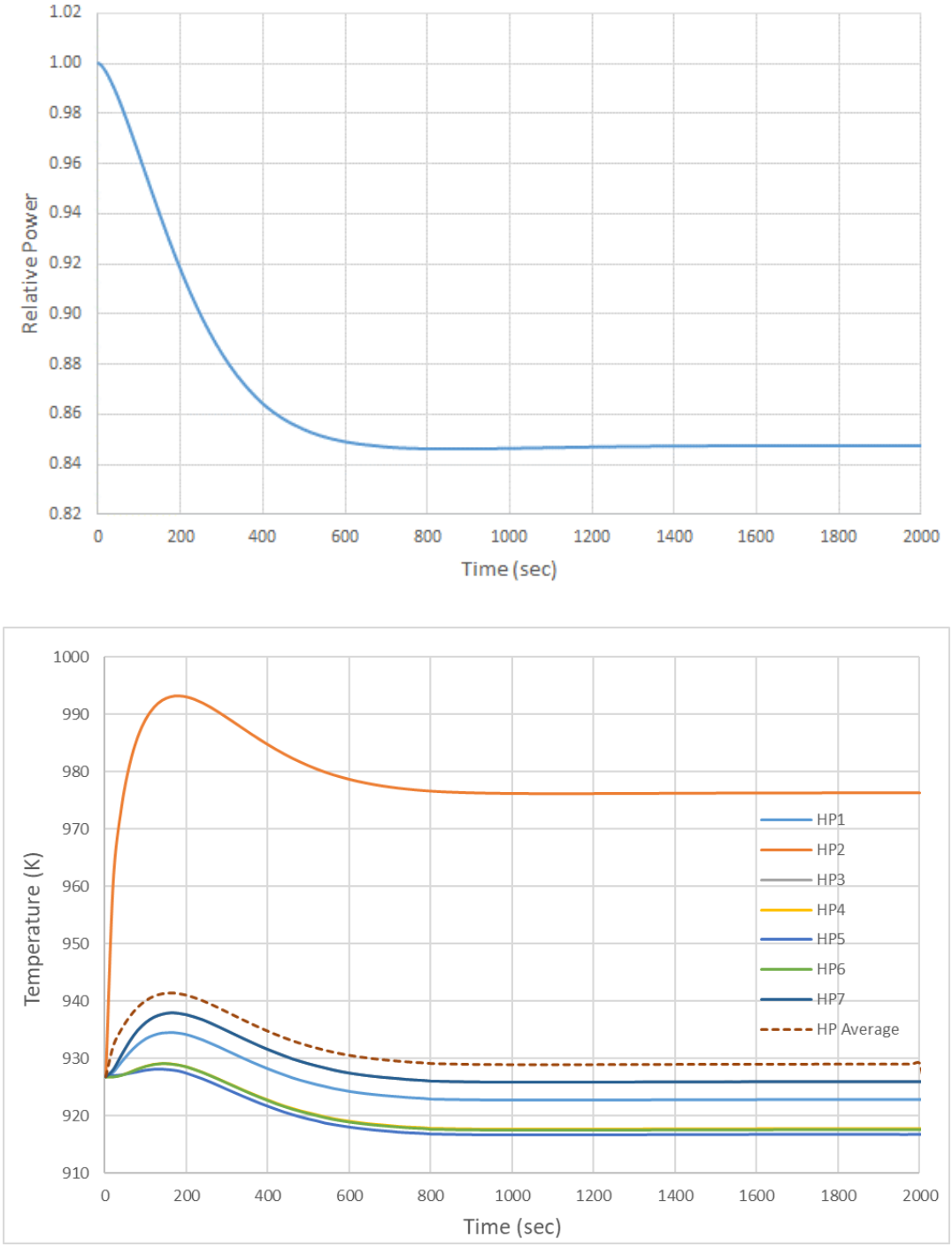


Figure 3-26. Relative Power (top) and Heat Pipe Temperature (bottom) Change with Time for One Heat Pipe Failure Transient Problem

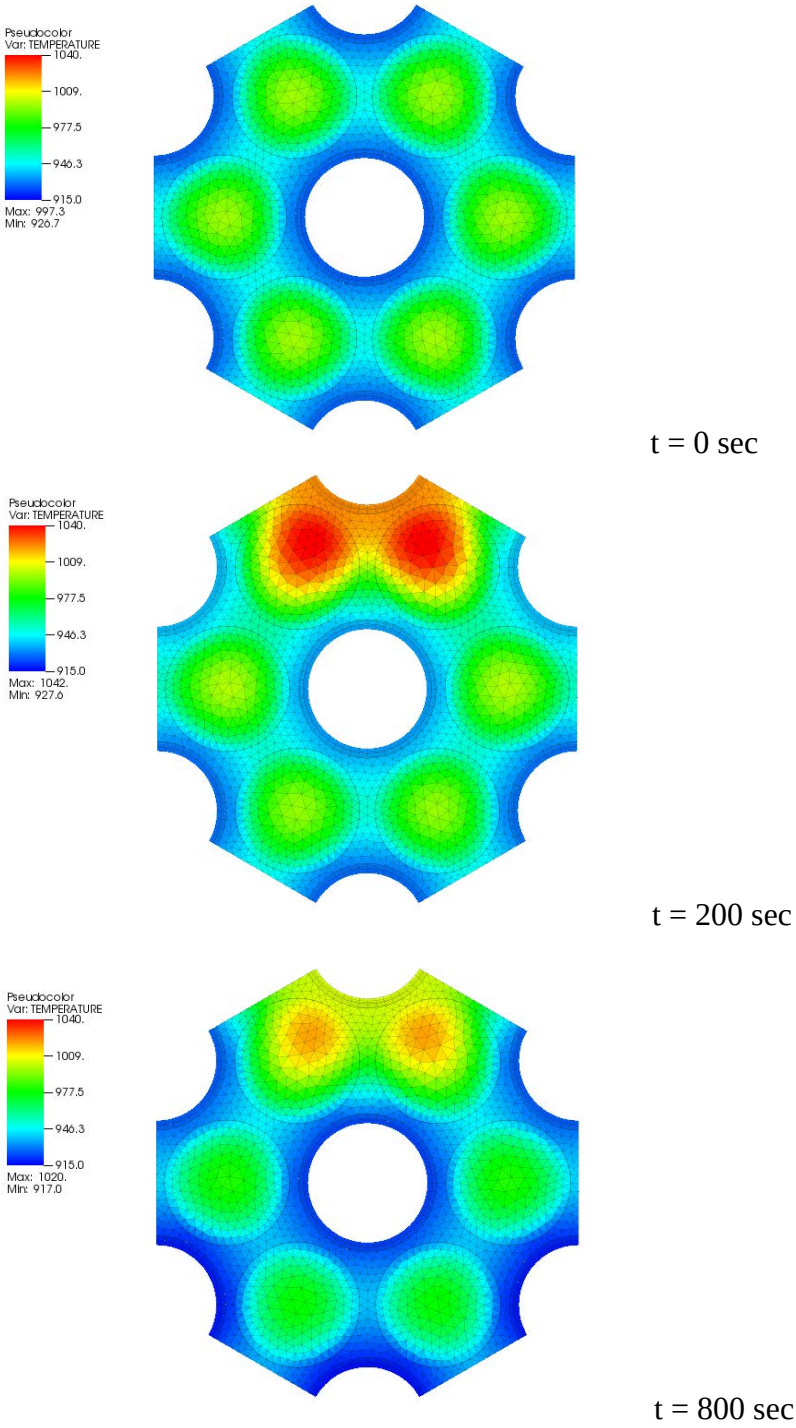


Figure 3-27. Temperature Distributions with Time for the Transient Case with One Heat Pipe Failure

3.3.2.2 Using PROTEUS/ANSYS/ANLHTP

The same transient calculation as done using the coupled system with FLUENT was performed using that with ANSYS-APDL. For verification, only thermal calculation was performed to compare results with those obtained from FLUENT. First, the steady-state calculation with the normal condition was performed, producing the similar heat pipe temperatures with those from FLUENT. Another steady-state calculation with the heat pipe failure condition in which the adiabatic boundary condition was applied to the surfaces of the failed heat pipe, resulting in reasonable temperature solutions as shown in Figure 3-28.

The null transient with the converged power from PROTEUS with the steady-state condition was tested to ensure that ANSYS solutions were converged to the same solution of the steady-state condition. After that, the transient calculation with the one heat pipe failure condition was performed. However, we observed that powers and temperatures decreased much faster with time than those changes observed in FLUENT. Currently, the problem is being investigated, whose results will be reported in the revision after it is resolved. As soon as reasonable agreement is made in transient solutions between the coupled systems, this section will be updated in the revision.

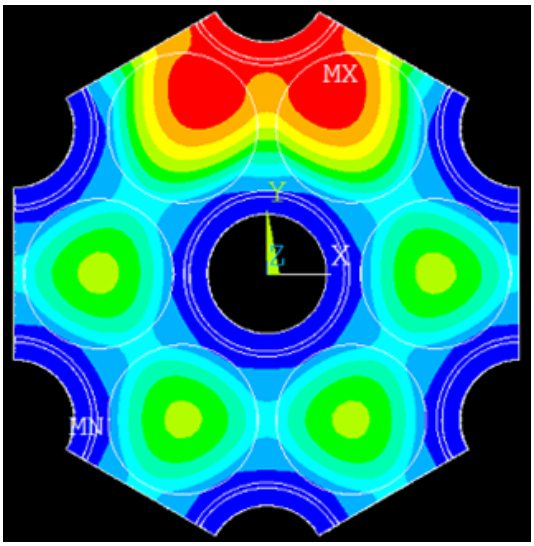


Figure 3-28. Temperature Distributions for the Steady-state Case with One Heat Pipe Failure

4. Conclusions and Future Work

The goal of this project is to develop the coupled system of PROTEUS/ANSYS for simulating a heat pipe cooled micro reactor. While figuring out how to control ANSYS Mechanical with Workbench for the coupled system, we first tried to build the coupled system of PROTEUS/FLUENT in the CORBA environment because FLUENT was easier than ANSYS Workbench to control for the coupled system simulating up to transient problems. In parallel, we found a way to control ANSYS Workbench with Python scripting for a steady-state problem. However, after making significant efforts to figure out ways to control ANSYS Workbench for a transient calculation, we concluded based on ANSYS expert consultations that it may be impossible to control the current version of ANSYS Workbench with Python scripting for a transient calculation because Python scripting may not provide capabilities of running a transient calculation time step-by-step playing with restart files and thus a lower level of control is required to handle ANSYS for a transient simulation.

The codes used in the coupled system are connected to each other using the Python drivers which coordinate the overall workflow including data exchange (power, temperature, mesh coordinates, and heat rate) required for the coupling and also control the individual calculation steps of the coupled codes such as convergence check and boundary conditions.

The coupled systems were qualitatively verified using a conceptual 3D unit assembly problem composed of six fuel rods and seven heat pipes with 180 cm high, which was developed based on the specification of MegaPower. For the completeness of coupled calculation, ANLHTP was introduced and coupled together which is the one-dimensional heat pipe performance analysis code developed and recently resurrected by ANL.

For verification, a steady-state problem was solved by the coupled system of FLUENT/ANLHTP with given asymmetric fuel powers, demonstrating the reasonable convergence of temperatures of seven heat pipes in the hexagonal lattice configuration. In addition, a transient simulation using PROTEUS/FLUENT/ANLHTP was also performed by making one out of seven heat pipes fail, showing reasonable changes of total power and heat pipe temperatures with time accounting for temperature feedback effects.

In the demonstration tests, even though it is a steady-state analysis code, ANLHTP was applied to a slow transient simulation, assuming that the vapor core has relatively negligible thermal inertia and a quick response to the change of the wick-vapor interface temperature. This assumption allows the code to analyze the vapor core using the steady-state model for a mild transient problem. To minimize errors that may arise from the assumption, part of the heat pipe components (i.e., heat pipe container, wick and liquid in the wick) that have relatively large thermal inertia was excluded from the ANLHTP domain and instead included in the FLUENT domain.

The same transient problem as tested with the coupled system with FLUENT was solved using PROTEUS/ANSYS/ANLHTP, in which APDL commands were used for ANSYS and only thermal calculation was performed, to qualitatively verify the coupled system, demonstrating the similar behavior in terms of core power and heat pipe temperature with that of the coupled system with FLUENT. However, we observed that the magnitude of the changes was noticeably larger than that obtained from the coupled system with FLUENT. Further investigation is underway to figure out what is causing the difference.

As future work, the following tasks will be conducted for the coupled system of PROTEUS/ ANSYS/ANLHTP:

- The coupled system should be updated to include the structural modeling and simulation of ANSYS using APDL commands with support of ANSYS Mechanical experts,
- The coupled system needs to be tested including a thermal expansion by ANSYS Mechanical. The demonstration of solving the transient problem did not include a structural (thermal expansion) calculation in order to compare results with those from the coupled system with FLUENT.
- Larger or actual-size 3D heat pipe cooled reactor problems will be tested for steady-state problems as well as heat pipe failure transient problems. Furthermore, the coupled system will be verified and validated using experiments available such as the KRUSTY experiment [29]. If possible, a demonstration model of eVinci or alternative will be simulated to support industry needs.
- An effort will be made to develop or update the user interface of the coupled system to make it easy to provide inputs for the coupled system. For this, a preliminary version of PyPROTEUS from NEAMS Workbench will be updated to support the input preparation of PROTEUS, and additional interfaces will be developed to support the coupled system of PROTEUS/ANSYS.

REFERENCES

1. A. Levinsky, J. J. V. Wyk, Y. Arafat, and M. C. Smith, "Westinghouse eVinci Reactor for off-Grid Markets," ANS Winter meeting, Orlando, Florida, November 11-15 (2018).
2. E. R. Shemon, M. A. Smith, and C. H. Lee, "PROTEUS-SN Methodology Manual," ANL/NE-14/5, Argonne National Laboratory, June (2014).
3. Y. S. Jung, C. H. Lee, and M. A. Smith, "PROTEUS-MOC User Manual," ANL/NE-18/10 Rev. 0, Argonne National Laboratory, September (2018).
4. Y. S. Jung, C. H. Lee, and M. A. Smith, "PROTEUS-NODAL User Manual," ANL/NE-18/4 Rev. 0, Argonne National Laboratory, September (2018).
5. P. R. Mcclure, D. I. Poston, V. R. Dasari, and R. S. Reid, "Design of Megawatt Power Level Heat Pipe Reactors," LA-UR-15-28840, Los Alamos National Laboratory (2015).
6. J. W. Sterbentz et al., "Reactor (5 MW) for Reliable Power at Remote Sites Assessment Report Using Phenomena Identification and Ranking Tables (PIRTs)," INL/EXT-16-40741, Rev.1, Idaho National Laboratory (2017).
7. C. H. Lee and Y. S. Jung, "Neutronics Simulation of Micro Nuclear Reactors Using the High-Fidelity Neutron Transport Code PROTEUS," ICAPP 2019, Juan-les-pins, France, May 12-15 (2019).
8. C. H. Lee and Y. S. Jung, "Micro Reactor Simulation Using the PROTEUS Suite in FY19," ANL/NSE-19/33, Argonne National Laboratory, September (2019).
9. ANSYS Web page, www.ansys.com.
10. FLUENT Web page, www.ansys.com/products/fluids/ansys-fluent.
11. CORBA Web page, www.corba.com.
12. Private communication with K. Hudspeth, www.simultechgroup.com (2020).
13. G. A. McLennan, "ANL/HTP: A Computer Code for the Simulation of Heat Pipe Operation," ANL-83-108, Argonne National Laboratory (1983).
14. M. A. Smith, E. E. Lewis, and E. R. Shemon, "DIF3D-VARIANT 11.0: A Decade of Updates," ANL/NE-14/1, Argonne National Laboratory, January (2014).
15. M. A. Smith, C. Adams, W. S. Yang, and E. E. Lewis, "VARI3D & PERSENT: Perturbation and Sensitivity Analysis," ANL/NE-13/8, Argonne National Laboratory, August (2013).
16. C. H. Lee and W. S. Yang, "MC^2-3: Multigroup Cross Section Generation Code for Fast Reactor Analysis," ANL/NE-11-41 Rev. 3, Argonne National Laboratory (2018).
17. J. Leppanen, "Serpent – a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code," VTT Technical Research Centre of Finland, June (2015).
18. P. K. Romano et al., "OpenMC: A State-of-the-Art Monte Carlo Code for Research and Development," *Ann. Nucl. Energy*, **82**, 90 (2015).
19. N. E. Stauff, C. H. Lee, P. K. Romano, and T. K. Kim, "Verification of Mixed Stochastic/Deterministic Approach for Fast and Thermal Reactor Analysis," ICAPP 2017, Fukui and Kyoto, Japan, April 24-28 (2017).
20. C. H. Lee and Y. S. Jung, "Generation of the Cross Section Library for PROTEUS," ANL/NE-18/2, Argonne National Laboratory, January (2018).
21. CUBIT Web page, www.cubit.sandia.gov.
22. M. A. Smith and E. R. Shemon, "User Manual for the PROTEUS Mesh Tools," ANL/NE-15/17 Rev. 2, Argonne National Laboratory, September (2016).

23. N. Stauff, et al., "Status of the NEAMS and ARC Neutronics Fast Reactor Tools Integration to the NEAMS Workbench," ANL/NEAMS-19/1, Argonne National Laboratory, September 30 (2019).
24. Private communication with N. Stauff, August (2019)
25. ANSYS, Inc., Workbench Scripting Guide, release 15.0, November (2013).
26. J. E. Kemme et al., "Performance Investigations of Liquid-Metal Heat Pipes for Space and Terrestrial Applications," *Proc. 3rd Int. Heat Pipe Conf.*, Palo Alto, CA, May (1978).
27. R. E. Holtz et al., "On the Experimental Operation of a Sodium Heat Pipe," ANL-85-61, Argonne National Laboratory (1985).
28. Anaconda www.anaconda.com.
29. D. I. Poston, T. Godfroy, P. R. Mcclure, and R. G. Sanchez, "KiloPower Project – KRUSTY Experiment Nuclear Design," LA-UR-15-25540, Los Alamos National Laboratory, July 20 (2015).

Nuclear Science and Engineering Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439-4842

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC