INL REPORT

INL/EXT-14-33977 (Revision 1)
Unlimited Release
Printed March 2018

RELAP-7 User's Guide

Prepared by
Idaho National Laboratory
Idaho Falls, Idaho 83415

INL/EXT-14-33977 (Revision 1)
Unlimited Release
Printed March 2018

RELAP-7 User's Guide

Hongbin Zhang, David Andrs, Joshua Hansel, Ling Zou, Ray Berry, Richard Martineau

Contents

Figures

Tables

RELAP-7 Overview

The RELAP-7 code is the next generation nuclear reactor system safety analysis code being developed at the Idaho National Laboratory (INL). The code is based on the INL's modern scientific software development framework, MOOSE (Multi-Physics Object Oriented Simulation Environment). The overall design goal of RELAP-7 is to take advantage of the previous thirty years of advancements in computer architecture, software design, numerical integration methods, and physical models. The end result will be a reactor systems analysis capability that retains and improves upon RELAP5's capability and extends the analysis capability for all reactor system simulation scenarios.

RELAP-7 will become the next generation tool in the RELAP reactor safety/systems analysis application series. The key to the success of RELAP-7 is the simultaneous advancement of physical models, numerical methods, and software design while maintaining a solid user perspective. Physical models include both PDEs (Partial Differential Equations) and ODEs (Ordinary Differential Equations) and experimental based closure models. RELAP-7 utilizes well-posed governing equations for compressible two-phase flow, which can be strictly verified in a modern verification and validation effort. Closure models used in TRACE provide a basis for the closure relations that are required in RELAP-7. RELAP-7 uses modern numerical methods, which allow implicit time integration, second-order schemes in both time and space, and strongly coupled multi-physics.

RELAP-7 is written with object oriented programming language C++. By using the MOOSE development environment, the RELAP-7 code is developed by following the same modern software design paradigms used for other MOOSE development efforts. The code is easy to read, develop, maintain, and couple with other codes. Most importantly, the modern software design allows the RELAP-7 code to evolve efficiently with time. MOOSE is an HPC development and runtime framework for solving computational engineering problems in a well planned, managed, and coordinated way. By leveraging millions of lines of open source software packages, such as PETSc (a nonlinear solver developed at Argonne National Laboratory) and LibMesh (a Finite Element Analysis package developed at University of Texas), MOOSE reduces the expense and time required to develop new applications. MOOSE provides numerical integration methods and mesh management for parallel computation. Therefore RELAP-7 code developers have been able to focus more upon the physics and user interface capability. There are currently over 20 different MOOSE based applications ranging from 3-D transient neutron transport, detailed 3-D transient fuel performance analysis, to long-term material aging. Multi-physics and multi-dimensional analysis capabilities, such as radiation transport and fuel performance, can be obtained by coupling RELAP-7 and other MOOSE-based applications through MOOSE and by leveraging with capabilities developed by other DOE programs. This allows restricting the focus of RELAP-7 to systems analysis type simulations and gives priority to retain and significantly extend RELAP5's capabilities.

This document provides a user's guide to help users to learn how to run the RELAP-7 code. A number of example problems and their associated input files are presented in this document to guide users to run the RELAP-7 code starting with simple pipe problems to problems with increasing complexity. Because the code is an ongoing development effort, this RELAP-7 User's Guide will evolve with periodic updates to keep it current with the state of the development, implementation, and model additions/revisions.

1 RELAP-7 Features

An overall description of the RELAP-7 architecture, governing theory, and computational approach is given here as an instructive, and executive overview of the RELAP-7 distinguishing features.

1.1 Software Framework

MOOSE is INL's development and runtime environment for the solution of multi-physics systems that involve multiple physical models or multiple simultaneous physical phenomena. The systems are generally represented (modeled) as a system of fully coupled nonlinear partial differential equation systems (an example of a multi-physics system is the thermal feedback effect upon neutronics cross-sections where the cross-sections are a function of the heat transfer). Inside MOOSE, the Jacobian-Free Newton Krylov (JFNK) method [1, 2] is implemented as a parallel nonlinear solver that naturally supports effective coupling between physics equation systems (or Kernels). The physics Kernels are designed to contribute to the nonlinear residual, which is then minimized inside of MOOSE. MOOSE provides a comprehensive set of finite element support capabilities (LibMesh [3], a Finite Element library developed at University of Texas) and provides for mesh adaptation and parallel execution. The framework heavily leverages software libraries from DOE SC and NNSA, such as the nonlinear solver capabilities in either the the Portable, Extensible Toolkit for Scientific Computation (PETSc [4]) project or the Trilinos project [5] (a collection of numerical methods libraries developed at Sandia National Laboratory).

1.2 Governing Theory

The primary basis of the RELAP-7 governing theory includes thermal fluids flow, reactor core heat transfer, and reactor kinetics models.

With respect to the thermal fluids flow dynamics models, RELAP-7 incorporates both single- and two-phase flow simulation capabilities encompassing all-speed and all-fluids. The single phase flow models include isothermal flow and anisothermal flow capabilities. The two-phase flow models include the well posed two fluid 7-equation model.

In addition to the fluids flow dynamics model, RELAP-7 necessarily simulates the heat transfer process with reactor kinetics as the heat source. The heat-conduction equation for cylindrical or slab geometries is solved to provide thermal history within solid structures such as fuel and clad. The volumetric power source in the heat conduction equation for the fuel comes from the point kinetics model with thermal hydraulic reactivity feedback considered [6]. The reactor structure is coupled with the thermal fluid through energy exchange (conjugate heat transfer) employing surface convective heat transfer [7] within the fluid. The fluid, heat conduction, conjugate heat transfer and point kinetics equations are solved in a fully coupled fashion in RELAP-7 in contrast to the operator-splitting or loose coupling approach used in the existing system safety analysis codes.

1.3 Computational Approach

Stated previously, the MOOSE framework provides the bulk of the ”heavy lifting” available to MOOSE-based applications with a multitude of mathematical and numerical libraries. For RELAP-7, LibMesh [3] provides the second-order accurate spatial discretization by employing linear basis, one-dimensional finite elements. The Message Passing Interface (MPI, from Argonne National Laboratory) provides for distributed parallel processing. Intel Threading Building Blocks (Intel TBB) allows parallel C++ programs to take full advantage of multicore architecture found in most large-scale machines of today. PETSc (from Argonne), Trilinos (from Sandia), and Hypre [8] (from Lawrence Livermore National Laboratory) provide the mathematical libraries and nonlinear solver capabilities for the Jacobian-free Newton-Krylov (JFNK) method. In MOOSE, a stiffly-stable, second-order backward difference (BDF2) formulation is used to provide second-order accurate time integration for strongly coupled physics in JFNK.

The JFNK method easily allows implicit nonlinear coupling of dependent physics under one general computational framework. Besides rapid (second-order) convergence of the iterative procedure, the JFNK method flexibly handles multiphysics problems when time scales of different physics are significantly varied during transients. The key feature of the JFNK method is combining Newton's method to solve implicit nonlinear systems with Krylov subspace iterative methods. The Krylov methods do not require an explicit form of the Jacobian, which eliminates the computationally expensive step of forming Jacobian matrices (which also may be quite difficult to determine analytically), required by Newton's method. The matrix-vector product can be approximated by the numerical differentiation of nonlinear residual functions. Therefore, JFNK readily integrates different physics into one solver framework.

2 Model Description

2.1 Fluids Flow Models

The RELAP-7 code has two flow models implemented. These include:
(1) a single phase fluid flow model,
(2) a nonhomogeneous, nonequilibrium seven-equation two phase flow model.

2.2 Fluid Properties

The fluid properties are specified in the *FluidProperties* block. For the **single phase flow model**, various types of equation of state can be used. These include:
(1). Linear equation of state for non-isothermal single phase flow. This is turned on by setting `type = LinearFluidProperties`.

(2). Stiffened gas equation of state which is turned on by setting
`type = StiffenedGasFluidProperties` for single phase flow

(3). Ideal gas equation of state which is turned on by setting `type=IdealGasFluidProperties`.
The equation of state for the nitrogen and helium gas has also been implemented into the code. It can be turned on by setting `type=N2FluidProperties` and `type=HeliumFluidProperties`, respectively.

(4). IAPWS-95 water and steam thermodynamic properties. It can be turned on by setting
`type=IAPWS95LiquidFluidProperties`.

For the **seven-equation two phase flow model**, the stiffened gas equation of state and the IAPWS-95 water and steam thermodynamic properties are applicable. They can be turned on by setting
`type = StiffenedGas7EqnFluidProperties`
or
`type = IAPWS957EqnFluidProperties`.

2.3 Solution Stabilization Schemes

It is well known that the continuous Galerkin finite element method is unstable when applied directly to hyperbolic systems of equations. Therefore the solution stabilization schemes are required for RELAP-7. The stabilization schemes can be turned on by adding the *Stabilizations* block in the input file. Currently available options of solution stabilization for RELAP-7 include:

(1). Streamline Upwind/Petrov Galerkin method (SUPG). The SUPG scheme works for the single phase flow only. It can be used by setting `type = SUPG`.

(2). Lapidus scheme works for both the single phase flow and the two phase flow cases. It can be used by setting `type = Lapidus`.

(3). The entropy viscosity method works for both the single phase flow and the two phase flow problems. This option can be used by setting `type = 'EntropyViscosity'`. Both the first order and second order entropy viscosity methods are available in the RELAP-7 code. The default method is the second order entropy viscosity method. In order to use the first order entropy viscosity method, some additional inputs are required in the *Stabilizations* block such as setting:
$use\_first\_order\_vf = true$,
$use\_first\_order\_liquid = true$,
$use\_first\_order\_vapor = true$.

(4). The pressure based stabilization scheme works for both the single phase and the two phase flow problems. This scheme can be used by setting `type = 'StabilizationPressure'`. Both the first order and the second order pressure based stabilization methods are available in the RELAP-7 code. They can be used by setting $order = FIRST$ or $order = SECOND$ respectively in the *Stabilizations* block of the input file.

2.4 Time Integration Schemes

All the time integration schemes in MOOSE are available in RELAP-7. However, there are two types of time integration schemes mostly used in the RELAP-7 code - Implicit Euler and BDF2. Implicit Euler is a first order accurate time integration scheme. This can be turned on by setting:`scheme = 'implicit-euler'` in the `Executioner` input block (explained later). This is the default option for RELAP-7. BDF2 is a second order accurate time integration scheme. This is the recommended option for RELAP-7.

2.5 Components

A real reactor system is very complex and contains hundreds of different physical components. It is impractical to resolve the real geometry of the entire system. Instead simplified thermal hydraulic models are used to represent (via "nodalization") the major physical components and describe the major physical processes (such as fluids flow and heat transfer). There are three main types of components developed in RELAP-7: (1) one-dimensional (1-D) components describing the geometry of the reactor system, (2) zero-dimensional (0-D) components for setting boundary conditions, and (3) 0-D components for connecting 1-D components.

2.5.1 Check Valves

The check valve (*CheckValve*) component simulates the dynamic behavior of check valves, with the form loss calculated by the abrupt area change model.

2.5.2 Compressible Valve

The compressible valve (*CompressibleValve*) component simulates the open and close behavior of valves for compressible fluid flow, including ckoking. It can be used to simulate the safety relief valves (SRV) of BWRs.

2.5.3 Core Channel

The core channel (*CoreChannel*) component is a composite component designed to simulate the coolant flow and heat conduction inside a fuel rod as well as the conjugate heat transfer between the coolant and the fuel rod. In this component, the fuel rod is divided into the same number of segments as that of the coolant flow pipe elements. Each fuel rod segment is further simulated as 1-D or 2-D heat conduction model perpendicular to the fluid flow model. Both plate type fuel rod and cylindrical fuel rod type can be simulated.

The solid fuel part is able to deal with typical LWR fuel rod with complex clad/gap/fuel pellet geometries. The flow model and conjugate heat transfer model are fully coupled in contrast to loosely coupled in the existing systems analysis codes such RELAP5 and TRACE, etc.

2.5.4 Down Comer

The down comer (*DownComer*) component simulates a large volume to mix different streams of water and steam and to track the water level. This component is applicable to BWRs only.

2.5.5 Elbow Pipe

The elbow pipe (*ElbowPipe*) component is a pipe component that includes a bend with a radius and an angle.

2.5.6 Free Boundary

The free boundary (*FreeBoundary*) component provides an open end boundary condition used for 1D components that must be connected to components at both ends.

2.5.7 Heat Generation

The heat genertion (*HeatGeneration*) component specifies the component and the material region that heat is generated in, and how that heat is distributed, from a power source originating in a separate component.

2.5.8 Heat Structure

The heat structure (*HeatStructure*) component models a solid material that conducts energy within the solid, and that convects energy into the adjacent pipe components. The heat structure is 2D only, and either a cylinder or plate. It can be a composite of several materials, and can be divided into a mesh.

2.5.9 Heat Transfer from an External Application

The *HeatTransferFromExternalAppTemperature* component connects the fluid flow in a pipe with the pipe surface temperatures calculated by an external application such as the BISON code.

2.5.10 Heat Transfer from Heat Flux

The *HeatTransferFromHeatFlux* component connects the fluid flow in a pipe with specified pipe wall heat flux such that the heat transfer between the fluids and the pipe wall can be calculated.

2.5.11 Heat Transfer from Heat Structure

The *HeatTransferFromHeatStructure* component connects the fluid flow in a pipe with the *HeatStructure* component such that the heat transfer between the fluids and heat structure can be calculated.

2.5.12 Heat Transfer from Specified Temperature

The *HeatTransferFromSpecifiedTemperature* component connects the fluid flow in a pipe with specified pipe wall temperatures such that the heat transfer between the fluids and pipe wall can be calculated.

2.5.13 Ideal Pump

The ideal pump (*IdealPump*) component is a junction component that provides a mass flow rate boundary condition between two pipe components.

2.5.14 InletDensityVelocity

The *InletDensityVelocity* component specifies density and velocity boundary conditions to the inlet of a pipe component.

2.5.15 InletMassFlowRateTemperature

The *InletMassFlowRateTemperature* component specifies mass flow rate and temperature boundary conditions to the inlet of a pipe component.

2.5.16 InletStagnationEnthalpyMomentum

The *InletStagnationEnthalpyMomentum* component provides prescribed momentum and prescribed specific total enthalpy boundary conditions to the inlet of a pipe component.

2.5.17 InletStagnationPressureTemperature

The *InletStagnationPressureTemperature* component provides prescribed stagnation pressure and prescribed stagnation temperature boundary conditions to the inlet of a pipe component.

2.5.18 Junction

The junction (*Junction*) model is a 0-D component representing a junction with no volume (inertia) effects considered, and with single/multiple inlets and single/multiple outlets, of which cross section areas can be different. This model conserves the mass and energy among all connecting components. This component thusfar only works for single phase fluids flow.

2.5.19 Outlet

The outlet (*Outlet*) component provides a backpressure boundary condition to the outlet of a pipe component. An example would be a pressure sink for a relief valve. There is an option for a reversible outlet with flow at the outlet conditions from the outlet component into the pipe component.

2.5.20 Pipe

The pipe (*Pipe*) component is the most basic component in RELAP-7. It is a 1-D component which simulates thermal fluids flow in a pipe. Both a constant cross section area and a variable cross section area options are available for the Pipe component. The wall friction and heat transfer coefficients are either calculated through closure models or provided by user input. All the thermal fluids dynamic models are available in the *Pipe* component which includes the single-phase non-isothermal flow model and the nonequilibrium 7-equation two-phase model.

2.5.21 Pipe With Heat Structure

The pipe with heat structure (*PipeWithHeatStructure*) component simulates fluids flow in a 1-D pipe coupled with 1-D or 2-D heat conduction through the pipe wall. The adiabatic, Dirichlet, or convective boundary conditions at the outer surface of the pipe wall are available. Either a plate type or cylindrical type of heat structure can be selected. Volumetric heat source within the fluids or solid materials can be added.

2.5.22 Point Kinetics

The point kinetics (*PointKinetics*) model is a lumped parameter neutron kinetics model to calculate the reactor power. User input reactivity or fully coupled feedback reactivity models are available.

2.5.23 Prescribed Reactor Power

The prescribed reactor power (*PrescribedReactorPower*) component is a virtual component that allows users to provide a constant value of reactor power, or a function name that returns the reactor power as a function of time.

2.5.24 Pump

A simple pump (*Pump*) model to provide a head and a reverse flow form loss coefficient (K) for either isothermal flow and non-isothermal flow. It can be driven by an user input head or through a driving component which provides shaft work.

2.5.25 Reactivity Feedback

The reactivity feedback (*ReactivityFeedback*) component calculates the reactivity feedback due to a change in moderator density (using a table of pairs of values) or using a moderator temperature coefficient of reactivity, and due to a change in the fuel temperature (using a table of pairs of values) or using a fuel temperature (Doppler) coefficient of reactivity. The resulting change in reactivity is then used in the point kinetics component.

2.5.26 Solid Wall

The solid wall (*SolidWall*) component is a boundary condition for a dead-ended pipe component. Therefore there is no flow circulating through the pipe, however the mass in the pipe can change due to coolant expansion or contraction.

2.5.27 RCIC Turbine

The turbine (*Turbine*) model in RELAP-7 is a simplified dynamical turbine model to simulate a reactor core isolation cooling (RCIC) turbine, which drives the RCIC pump through a common shaft.

2.5.28 Valve

The valve (*Valve*) component simulates the open and close behaviors of valves for incompressible flow with user given trigger and response time. The abrupt area change model is used to calculate the form loss.

2.5.29 Volume Junction

The volume junction (*VolumeJunction*) model is a 0-D component representing a joint/junction model with volume (inertia) effects considered. This model conserves the mass and energy among all connecting components. This component currently only works for single-phase fluids flow and two-phase flow capability is still being developed.

2.5.30 Wet Well

The wet well (*WetWell*) component simulates the dynamic response of a BWR suppression pool and its gas space.

3 Running RELAP-7

3.1 Complete Step 1 of MOOSE Environment Setup

The system environment setup for MOOSE can be found with the link:
`http://www.mooseframework.org/getting-started`

3.2 Off-site Access

If you are accessing the code from an off-site location (i.e. from a network that is ouside of the INL's internal network), you will need to open SSH and HTTP tunnel. The SSH tunnel itself will give you access to the code repository and the HTTP tunnel will allow you to access RELAP-7 web pages.

To setup your machine to use the SSH tunnel, modify your `~/.ssh/config` file so it contains:

```
Host hpcgitlab.inl.gov
User <your HPC username here>
ProxyCommand nc -x localhost:5555 %h %p
```

Note that you need to do this only once.

To open the SSH tunnel, you will need your RSA token that is associated with your HPC account.

In a terminal, run:

```
$ ssh -D 5555 hpclogin.inl.gov
```

When asked, enter your PIN and the number from your RSA token.

Now, you need to setup SOCKS proxy in your web browser, so you can access RELAP-7 web pages. The settings are:

```
hostname: localhost
port: 5555
```

This is browser and platform dependent, so we will not describe steps for every possible combination. Web search typically reveals necessary steps.

At this point you should be able to access `https://hpcgitlab.inl.gov/`. If you do not have access to the code yet, login into the system and send your user name to the technical point of contact. When the access is granted you will recieve an email from the system about it.

3.3 Setup Your SSH Key

You have to setup your SSH key on the hpcgitlab web page in order to access the code.

SSH key allows you to establish a secure connection between your computer and GitLab. Before generating an SSH key, check to see if your system already has one by running `cat ~/.ssh/id_rsa.pub`. If you see a long string starting with `ssh-rsa` or `ssh-dsa`, you can skip the `ssh-keygen` step below.

To generate a new SSH key, just open your terminal and use the code below. The `ssh-keygen` command prompts you for a location and filename to store the key pair and for a password. When prompted for the location and filename you can press `enter` to use the default. It is a best practice to use a password for an SSH key but it is not required and you can skip creating a password by pressing `enter`. Note that the password you choose here can not be altered or retrieved.

```
ssh-keygen -t rsa -C "$your_email"
```

Use the code below to show your public key.

```
cat ~/.ssh/id_rsa.pub
```

Go into the section `SSH Keys` in your profile and click `Add SSH Key`. Copy and paste the key to the `Key` section and name your key as well. Please copy the complete key starting with `ssh-` and ending with your username and host.

To test your SSH key.

```
$ ssh git@hpcgitlab.inl.gov
Welcome to GitLab, <your name here>!
Connection to hpcgitlab.inl.gov closed.
```

3.4 Checking Out the Code

```
$ cd ~/projects/
$ git clone git@hpcgitlab.inl.gov:idaholab/relap-7.git
$ cd relap-7
$ git submodule init
$ git submodule update
```

It is necessary to build libmesh before building any application.

```
$ cd ~/projects/relap-7/moose
$ ./scripts/update_and_rebuild_libmesh.sh
```

Once libmesh has been successfully compiled, you may now compile RELAP-7.

```
cd ~/projects/relap-7
make (add -jn to run on multiple "n" processors)
```

Once RELAP-7 has been compiled successfully, it is recommended to run the tests to make sure the version of the code you have is running correctly.

```
cd ~/projects/relap-7
./run_tests (add -jn to run "n" jobs at one time)
```

3.5 Executing RELAP-7

When first starting out with running RELAP-7, it is recommended to start from an example problem. Multiple example problems with input files are presented in this document. More examples can be found under the `/relap-7/test/tests` subdirectory. To demonstrate how to run RELAP-7, consider the `phy.PWR_core_channel.i` test problem.

```
cd ~/projects/relap-7/test/tests/components/core_channel/
# To run with one processor
~/projects/relap-7/relap-7-opt -i phy.PWR_core_channel.i
# To run using multiple treads:
~/projects/relap-7/relap-7-opt -i phy.PWR_core_channel.i --n-threads=4
```

3.6 Post Processing

RELAP-7 typically writes solution data to an ExodusII file. The solution data may also be written in other formats, one being a comma separated values (CSV) file, which allows the solution data to be saved in a table structured format. The other being a tecplot file in either binary or ASCII format.

Several options exist for viewing ExodusII output files. One good choice is to use open-source software, Paraview (`www.paraview.org`).

4 Input Files

RELAP-7 uses a block-structured input file. Each block is identified with square brackets. The opening brackets contain the type of the input block and the empty brackets mark the end of the block. Each block may contain subblocks.

```
[BlockName]
   <block line commands>
   [./subblock_name]
      <subblock line commands>
   [../]
[]
```

Each subblock must have an unique name when compared with all other subblocks in the current block.

Line commands are given as parameter and value pairs with an equal sign between them. They specify parameters to be used by the object being described. The parameter is a string, and the value may be a string, an integer, a real number, or a list of strings, integers, or real numbers. Lists are given in single quotes and are separated by whitespace.

Subblocks normally contain a type line command. This line command specifies the particular type of object being described.

RELAP-7 uses SI units. This stsndardizes the model input by eliminating the possibility of errors caused by using one set of units for one model and another set of units for a different model.

The following subsections have brief descriptions of each block. More detailed descriptions can be found in the examples section.

4.1 Global Parameters

The `GlobalParams` block specifies the global parameters used by the code such as the initial pressure (`initial_p`), velocity (`initial_vel`) and temperature (`initial_T`) of the system model, the stabilzation scheme type (`stabilization`), and the scaling factors (`scaling_factor_1phase`) for the primary variable, etc. The values of global parameters are available to any other block or subblock in the input file. If a command line is missing in a block or a subblock but defined in `GlobalParams`, the block or subblock will use the parameter defined in `GlobalParams`. However, if the block or subblock has a command line, that will be used regardless of what is in GlobalParams.

The following is an example of the `GlobalParams` block:

```
[GlobalParams]
  initial_p = 15.17e6
  initial_vel = 1.
  initial_T = 564.15

  scaling_factor_1phase = '1.e0 1.e-2 1.e-5'

  stabilization = evm1
[]
```

# symbol indicates comments in the input file and can be located anywhere in the input file.

`initial_p`	Initial pressure (Pa).
`initial_vel`	Initial velocity (m/s).
`initial_T`	Initial temperature (K).
`scaling_factor_1phase`	Scaling factors.
`stabilization`	Solution stabilization schemes.

4.2 Fluid Properties

The `FluidProperties` block specifies the equation of state to be used by the code. The following is an example of the `FluidProperties` block:

```
[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]
```

`type`	The type of equation of state to be used.

4.3 HeatStructureMaterials

The `HeatStructureMaterials` block specifies the properties of the solid materials for the code. The following is an example of the `HeatStructureMaterials` block:

```
[HeatStructureMaterials]
  [./fuel-mat]
    type = SolidMaterialProperties
    k = 2.5
    Cp = 300.
    rho = 1.032e4
  [../]
  [./gap-mat]
    type = SolidMaterialProperties
    k = 0.6
    Cp = 1.
    rho = 1.
  [../]
  [./clad-mat]
    type = SolidMaterialProperties
    k = 21.5
    Cp = 350.
    rho = 6.55e3
  [../]
[]
```

`k`	Thermal conductivity ($W/(m \cdot K)$).
`Cp`	Specific heat ($J/(Kg \cdot K)$).
`rho`	Density of solid materials (Kg/m^3).

4.4 Functions

The Functions block provides the functions to be used by the code during the simulations such as reactor power as a function of time or a boundary condition pressure as a function of time, etc. The following is an example of defining pressure distribution as a function of `x`.

```
[Functions]
```

```
  [./p_func]
    axis = x
    type = PiecewiseLinear
    x = '0        3'
    y = '1.05e5   1e5'
  [../]
[]
```

`axis`	For RELAP-7 1-D components, only the x coordinate is used in the calculations and hence `axis` is set to be 0 which allows a function to be defined as a function `x`. The axis used (0, 1, or 2 for x, y, or z) if this is to be a function of position. If `axis` is not given in the input file, the data pair of (`x,y`) will be taken as a time function.
`type`	Type of functions used to interpolate data between data points.
`x`	The coordinate used for the x-axis data. This is the pipe axial direction spatial coordinate relative to the starting point.
`y`	The initial pressure distribution function data point values.

4.5 Components

The `Components` block specifies the components to be used in the simulations. The list of available components is shown in Section 2.5.

```
[Components]
  [./reactor]
    type = PrescribedReactorPower
    function = 77337.69407
  [../]

  [./CCH1]
    type = CoreChannel
    fp = eos
    # geometry
    position = '0 0 0'
    orientation = '0 0 1'
    A = 8.78882e-5
    D_h = 0.01179
    length = 3.865
    n_elems = 20
    f = 0.01
    Hw = 5.33e4
    initial_Ts = 559.15
    dim_hs = 2
    fuel_type = cylinder
    name_of_hs = 'fuel gap clad'
    n_heatstruct = 3
    width_of_hs = '0.004096 0.0001 0.000552'
    elem_number_of_hs = '10 1 2'
    material_hs = 'fuel-mat gap-mat clad-mat'
    power = reactor
    power_fraction = '1.0 0.0 0.0'
  [../]

  [./inlet]
    type = InletDensityVelocity
    input = 'CCH1:pipe(in)'
    rho = 753.68
    vel = 4.43
  [../]
  [./outlet]
    type = Outlet
    input = 'CCH1:pipe(out)'
    p = '155.e5'
    legacy = true
  [../]
[]
```

`[./reactor]`	Subblock for the reactor component.
`function`	Prescribed reactor power either as a constant or as a function of time.
`[./CCH1]`	Subblock for the core channel component.
`Hw`	Wall heat transfer coefficient.
`initial_Ts`	Prescribed initial temperature of the solid materials.
`dim_hs`	The dimension of the mesh used for the heat conduction calculations in the heat structure. The options are `dim_hs=1` for 1D heat conduction calculations or `dim_hs=2` for 2D heat conduction calculations.
`fuel_type`	Geometry type of the fuel. The available options are `fuel_type = cylinder` or `fuel_type = plate`.

name_of_hs	Prescribed heat structure names.
n_heatstruct	Prescribed number of heat structures.
width_of_hs	Width of each heat structure.
elem_number_of_hs	Number of elements of each heat structure.
material_hs	Name of the materials (defined in the Materials block) used in the heat structure.
power_fraction	The fraction of reactor power that goes into each heat structure.

4.6 Preconditioner

The Preconditioning block specifies the preconditioner to be used by the preconditioned JFNK solver for the RELAP-7 code. The solution algorithm for RELAP-7 is the Jacobian-free Newton-Krylov (JFNK) method. However the Krylov methods need preconditioning to be efficient. Hence, the solvers available in RELAP-7 are preconditioned JFNK (PJFNK). Two options are available in RELAP-7 to build the preconditioning matrix, the single matrix preconditioner (SMP) and the finite difference preconditioner (FDP). The SMP option uses all the Jacobian terms derived analytically to build one preconditioning matrix. The FDP option uses numerical Jacobian by doing direct finite differences of the residual terms. The SMP option is the more efficient and is the recommended option, while the FDP option is normally slow and inefficient and is recommended to be used for small problems or for debugging purposes. The following is an example of the Preconditioning block:

```
[Preconditioning]
  # Uncomment one of the lines below to activate one of the blocks...
   active = 'SMP_PJFNK'
  #active = 'FDP_PJFNK'

  # The definitions of the above-named blocks follow.
  [./SMP_PJFNK]
    type = SMP
    full = true  # off diagonal blocks are used
    solve_type = 'PJFNK' # Preconditioned JFNK solver
  [../]


  [./FDP_PJFNK]
    type = FDP
    full = true
    solve_type = 'PJFNK'
  [../]
[]
```

`type`	Indicating the preconditioning matrix type is SMP for this case.
`full`	true means the entire preconditioning matrix will be assembled in the calculations.
`solve _type`	PJFNK: Preconditioned Jacobian-Free Newton Krylov JFNK: Jacobian-Free

4.7 Executioner

The `Executioner` block specifies the executioner that will be used in the simulations. There are two main types of executioner - Transient and Steafy.

```
[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 1.e-2
  dtmin = 1.e-5

  nl_rel_tol = 1e-8
  nl_abs_tol = 1e-6
  nl_max_its = 15

  l_tol = 1e-6
  l_max_its = 30

  start_time = 0.0
  end_time = 10.0
  num_steps = 300

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]
```

`type`	Transient executioner type will be used.

scheme	BDF2 time integration will be used. If this is absent, the default option of BDF2 will be used.
dt	Time step size to be used .
dtmin	Minimum time step size to be used. If the time step size is below this value, the case will stop executing.
nl_rel_tol	Relative tolerance for the nonlinear solve. The recommended value is 1.0E-8.
nl_abs_tol	Absolute tolerance for the nonlinear solve. The recommended value is 1.0E-6.
nl_max_its	Maximum number of nonlinear solve iterations. The recommended value is 15.
l_tol	Relative tolerance for the linear Krylov solve. The recommended value is 1.0E-6.
l_max_its	Maximum number of liner iterations. The recommended value is 30.
start_time	The start time of the simulation.
end_time	The end time of the simulation.
num_steps	The maximum number of time integration steps for the simulation. The simulation will stop either the maximum number of time steps is reached or the end_time is reach.
[./Quadrature]	The quadrature subblock. This is the recommended option.
type	Type of the quadrature rule. Advanced users can refer to the MOOSE manual for other options.
order	Order of the quadrature.

4.8 Outputs

The Outputs block controls the various screen and file output in the simulations.

```
[ Outputs ]
  print_perf_log = true
  [./ out_displaced ]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

print_perf_log	Enable printing of the performance log to the screen (Console)

`type`	Exodus output file type.
`sequence`	= true, otherwise defaults to false

5 Examples

5.1 Example 1: A Simple Pipe Flow Problem

5.1.1 Problem Description

Example 1 simulates water flowing through a pipe using IAPWS-95 fluid properties. The inlet boundary condition is set by $InletDensityVelocity$ with liquid density at $\rho = 753.632$ kg/m^3 and liquid velocity at $vel = 4.43$ m/s. The outlet boundary condition is set by $Outlet$ with the pressure set at $p = 15.51E6$ pa.

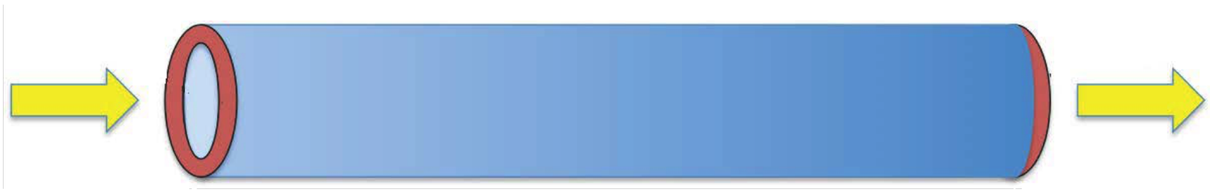

Figure 1. A simple pipe diagram

A 1D model can be viewed in Paraview to visualize the process better. Once Paraview is opened, on the left, select all of the variables and click ”Apply.” In the filters tab, under ”Data Analysis”, select ”Plot Over Line” and apply. When this is done, each parameter in the problem can be viewed plotted over the length of the pipe. The figure below shows pressure vs length.

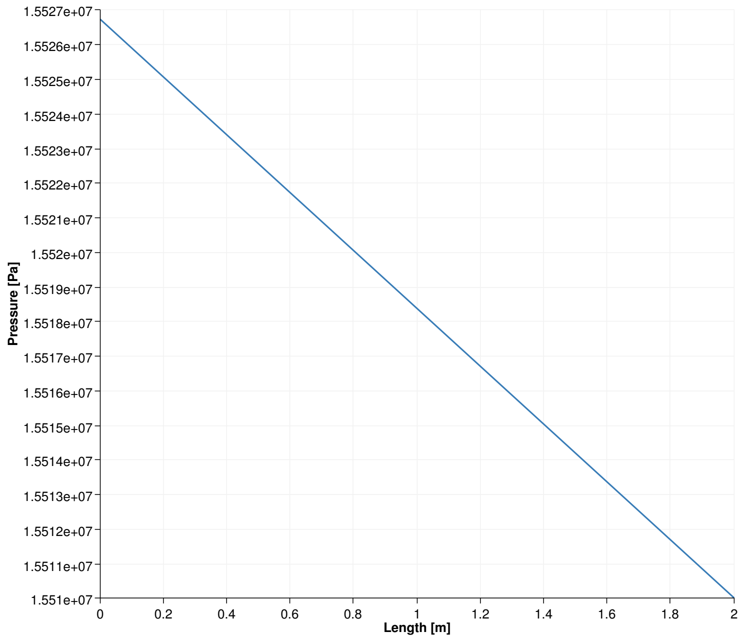

Figure 2. Pressure vs. length for the simple pipe flow problem

5.1.2 Input File

The following shows the input file to run this example problem:

```
[GlobalParams]
  gravity = '0 0 0'
  initial_p = 15.51e6
  initial_T = 559.15
  initial_vel = 0.0
  closures_type = TRACE
[]

[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[Components]
  [./pipe]
    type = Pipe
    position = '0 0 0'
    orientation = '1 0 0'
    length = 2.0
    n_elems = 5
    A = 8.78882e-5
    D_h = 0.01179
    fp = eos
  [../]

  [./inlet]
    type = InletDensityVelocity
    input = 'pipe(in)'
    rho = 753.632
    vel = 4.43
  [../]

  [./outlet]
    type = Outlet
    input = 'pipe(out)'
    p = 15.51e6
    legacy = true
  [../]
[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
  [../]
[]
[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 0.01
  dtmin = 1.0e-8

  nl_rel_tol = 1e-6
  nl_abs_tol = 1e-7
  nl_max_its = 30

  l_tol = 1e-5
  l_max_its = 10

  start_time = 0.0
  end_time = 10.0
  timestep_tolerance = 1e-10

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.2 Example 2: Use Functions to Set Initial Conditions for a Simple Pipe Flow Problem

5.2.1 Problem Description

This example illustrates how to use functions to set the initial conditions. This example allows the user to set the pipe wall friction coefficient. The Functions block is used to set the pipe wall friction coefficient. This example uses the PiecewiseLinear function which takes a set of x and y values and creates a linear function from it.

The pressure difference can be calculated using the following equation to verify the code calculated results:

$$\Delta P = \frac{f L \rho u^2}{2 D_h} \tag{1}$$

where f is the pipe wall friction coefficient, L is the pipe length, ρ is the density, u is the velocity, D_h is the hydraulic diameter, and P is pressure. Density and velocity values can be found using Paraview to perform the hand calculation.

5.2.2 Input File

```
[GlobalParams]
  gravity = '0 0 0'
  initial_p = 15.51e6
  initial_T = 559.15
  initial_vel = 0.0
  closures_type = TRACE
[]

[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[Functions]
  [./f_func]
    type = PiecewiseLinear
    x = '0     0.05  0.1'
    y = '0.01  0.02  0.02'
  [../]
[]

[Components]
  [./pipe]
    type = Pipe
    position = '0 0 0'
    orientation = '1 0 0'
    length = 2.0
    n_elems = 5
    A = 8.78882e-5
    D_h = 0.01179
    fp = eos
    f = f_func
  [../]

  [./inlet]
```

```
    type = InletDensityVelocity
    input = 'pipe(in)'
    rho = 753.632
    vel = 4.43
  [../]

  [./outlet]
    type = Outlet
    input = 'pipe(out)'
    p = 15.51e6
    legacy = true
  [../]
[]
[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
  [../]
[]

[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 0.01
  dtmin = 1.0e-8

  nl_rel_tol = 1e-6
  nl_abs_tol = 1e-7
  nl_max_its = 30

  l_tol = 1e-5
  l_max_its = 10

  start_time = 0.0
  end_time = 10.0
  timestep_tolerance = 1e-10
  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.3 Example 3: Adding Heat to a Simple Pipe Flow Problem

5.3.1 Problem Description

This example shows how to add heat to the fluid in a simple pipe flow problem. The heat can be added either from a specified pipe wall temperature or from a specified heat flux through the pipe wall. The input file for this example builds upon that used for example 2. In this example, an new input block is added to add the $HeatTransferFromSpecifiedTemperature$ component such that the pipe wall temperature can be specified. The wall temperature is set as: $T\_wall = 565$ K. Alternatively, if a constant pipe wall heat flux needs to be inputted, then the $HeatTransferFromHeatFlux$ component needs to be added and the heat flux is supplied to q_wall with the specified amount. The following input file only shows the example of how to add heat to a simple pipe flow problem with a specified pipe wall temperature.

Figure 3 shows the specified pipe wall temperature and the calculated water temperature versus pipe length for this example problem.

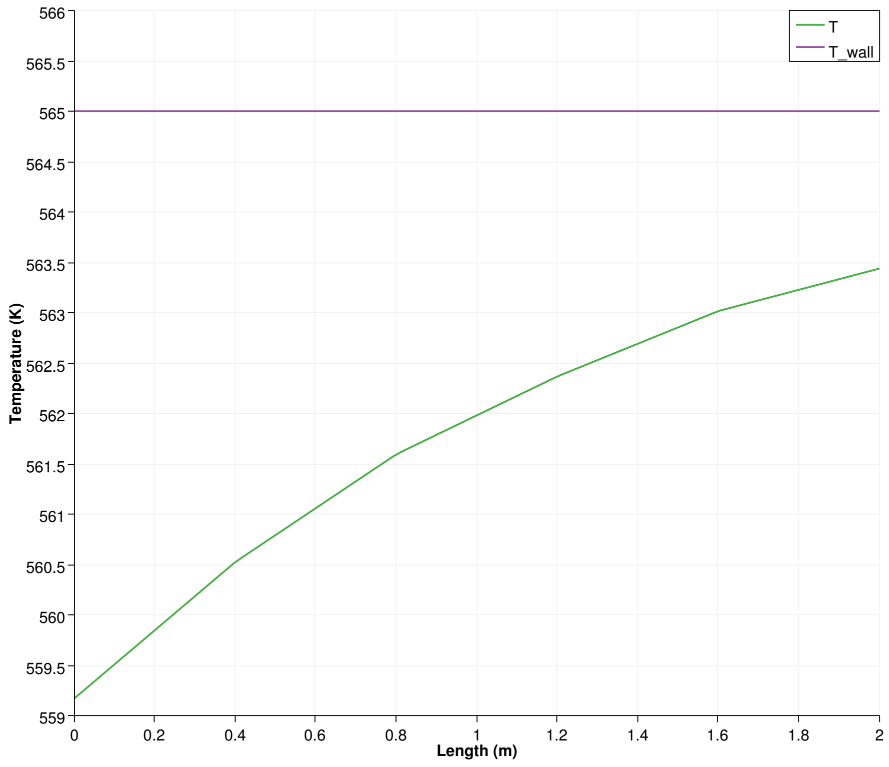

Figure 3. Wall temperature and fluid temperature vs. length for the simple pipe flow problem

5.3.2 Input File

```
[GlobalParams]
  gravity = '0 0 0'
  initial_p = 15.51e6
  initial_T = 559.15
  initial_vel = 0.0
  closures_type = TRACE
[]

[FluidProperties]
```

```
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[Functions]
  [./f_func]
    type = PiecewiseLinear
    x = '0     0.05  0.1'
    y = '0.01  0.02  0.02'
  [../]
[]

[Components]
  [./pipe]
    type = Pipe
    position = '0 0 0'
    orientation = '1 0 0'
    length = 2.0
    n_elems = 5
    A = 8.78882e-5
    D_h = 0.01179
    fp = eos
    f = f_func
  [../]

  [./ht_pipe1]
    type = HeatTransferFromSpecifiedTemperature
    pipe = pipe
    T_wall = 565
  [../]

  [./inlet]
    type = InletDensityVelocity
    input = 'pipe(in)'
    rho = 753.632
    vel = 4.43
  [../]
  [./outlet]
    type = Outlet
    input = 'pipe(out)'
    p = 15.51e6
    legacy = true
  [../]
[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
  [../]
[]

[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 0.01
  dtmin = 1.0e-8

  nl_rel_tol = 1e-6
  nl_abs_tol = 1e-7
  nl_max_its = 30

  l_tol = 1e-5
  l_max_its = 10

  start_time = 0.0
  end_time = 10.0
  timestep_tolerance = 1e-10

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
```

```
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.4 Example 4: A Simple Pipe Flow Problem with the 7-Equation Model

5.4.1 Problem Description

This example shows a two phase flow with the 7-equation model through one pipe. The stagnation pressure and temperature boundary conditions for both the liquid and vapor composition are provided at the pipe inlet by using the *InletStagnationPressureTemperature* component. The static pressure boundary condition is used at the pipe outlet by using the *Outlet* component. The initial conditions are set by the *Functions* block. The initial pressure, velocity, temperature and void fraction distribution are provided in this example. The initial conditions are set close to steady state solutions and the simulation quickly converges to steady state solutions.

5.4.2 Input File

```
[GlobalParams]
  gravity = '0 0 0'
  initial_T_liquid = 558.98002280575
  initial_T_vapor  = 558.98002280575
  initial_p_liquid = 7.0e6
  initial_p_vapor  = 7.0e6
  initial_vel_liquid = 0
  initial_vel_vapor = 0
  initial_alpha_vapor = 0.95

  scaling_factor_2phase = '1
                             1e1 1e1 1e-3
                             1e1 1e1 1e-3'

  phase_interaction = true
  pressure_relaxation = true
  velocity_relaxation = true
  interface_transfer = true
  wall_mass_transfer = true
  specific_interfacial_area_max_value = 1000
  specific_interfacial_area_min_value = 1e-15
[]

[Functions]

  # initial pressure distribution in the pipe
  [./p_func]
    axis = x
    type                       = PiecewiseLinear
    x                          = '0        3.66'  # this is the pipe axial directio
    y                          = '7.0008e6 7e6'
  [../]

  # initial velocity distribution in the pipe
  [./v_func]
    axis = x
    type                       = PiecewiseLinear
    x                          = '0       3.66'
    y                          = '2.44685 2.4471'
  [../]

  # initial temperature distribution in the pipe
  [./temp_func]
    axis = x
    type                       = PiecewiseLinear
    x                          = '0        3.66'
    y                          = '558.978  558.974'
  [../]

\enitial void fraction distribution in the pipe
  [./void_func]
    axis = x
    type                       = PiecewiseLinear
    x                          = '0      3.66'
    y                          = '0.95   0.95003'
  [../]
```

```
[]

[FluidProperties]
  [./eos]
    type = IAPWS957EqnFluidProperties
  [../]
[]

[Components]
  [./pipe]
    type = Pipe
    fp = eos
    position = '0 0 0'
    orientation = '1 0 0'
    A = 1.907720E-04
    D_h = 1.698566E-02
    length = 3.66
    f = 1.698566E-02
    f_interface = 0
    n_elems = 50

    # initial condition close to steady state solutions
    initial_p_liquid = p_func
    initial_T_liquid = temp_func
    initial_vel_liquid = v_func
    initial_p_vapor = p_func
    initial_T_vapor = temp_func
    initial_vel_vapor = v_func
    initial_alpha_vapor = void_func
  [../]

  [./inlet]
    type = InletStagnationPressureTemperature
    input = 'pipe(in)'
    p0_liquid = 7.001e6
    p0_vapor = 7.001e6
    T0_liquid = 558.98002280575
    T0_vapor = 558.98002280575
    alpha_vapor = 0.95
  [../]
  [./outlet]
    type = Outlet
    input = 'pipe(out)'
    p_vapor = 7.0e6
    p_liquid = 7.0e6
    legacy = true
  [../]
[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
  [../]
[]

[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 1e-1
  dtmin = 1.e-5

  nl_rel_tol = 1e-10
  nl_abs_tol = 1e-8
  nl_max_its = 30

  l_tol = 1e-3
  l_max_its = 30

  start_time = 0.0
  end_time = 10.0

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]


[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.5 Example 5: A Core Channel Problem

5.5.1 Problem Description

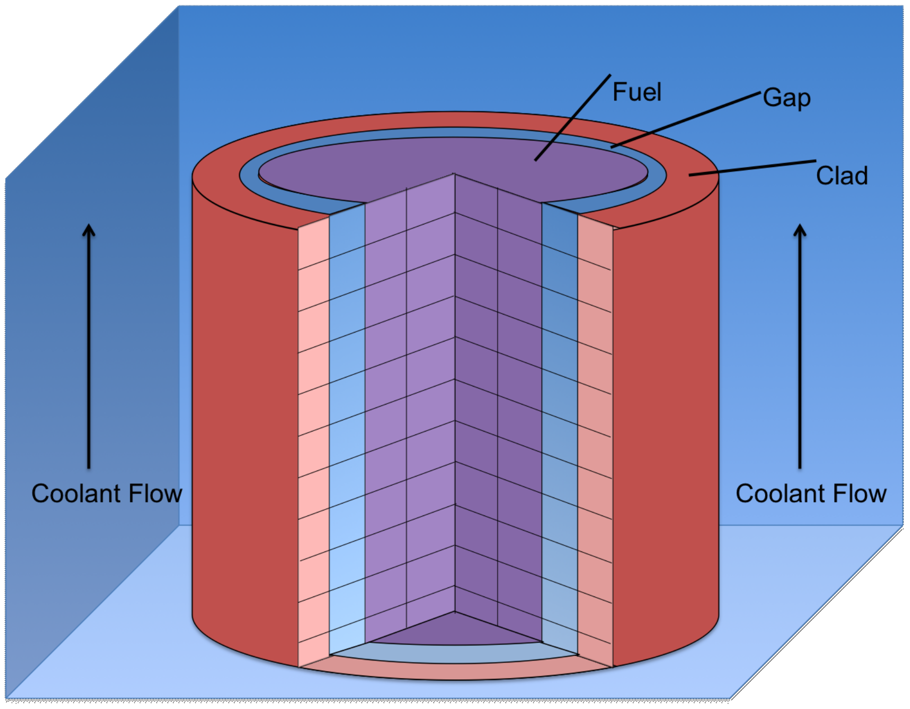

Figure 4. Diagram of a core channel

This example simulates fluids flow and heat transfer in a core channel. The core channel is schematically shown in Fig. 4. This problem simulates fluid flow in one subchannel with a single fuel rod as the heat source. A prescribed power of 77.3 KW is supplied to the fuel rod. The boundary conditions are applied to the ends of the core channel with *InletDensityVelocity* used to set the inlet boundary conditions and *Outlet* used to set the outlet boundary condition.

This example includes the *HeatStructureMaterials* block where the parameters of the fuel, gap, and clad are defined. Each of them are made up of different materials which are defined by their thermal conductivity, density, and specific heat.

The core channel requires the hydraulic diameter, and cross sectional area to be defined. These can easily be calculated using a script in RELAP-7. To do this, the user must go to scripts folder and run `./core-channel-cylinder.py` followed by two values for the radius of the cylinder and the pitch. For example, given the radius is .004748 m and the pitch is .0126 m. Using these values in the script will give an area of 8.7937e-5 m^2, a hydraulic diameter of .01179 m.

5.5.2 Input File

```
[GlobalParams]
  initial_p = 155.e5
  initial_vel = 0.
  initial_T = 559.15

[]

[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[HeatStructureMaterials]
  [./fuel-mat]
    type = SolidMaterialProperties
    k = 2.5
    Cp = 300.
    rho = 1.032e4
  [../]
  [./gap-mat]
    type = SolidMaterialProperties
    k = 0.6
    Cp = 1.
    rho = 1.
  [../]
  [./clad-mat]
    type = SolidMaterialProperties
    k = 21.5
    Cp = 350.
    rho = 6.55e3
  [../]
[]

[Components]
  [./reactor]
    type = PrescribedReactorPower
    function = 77337.69407
  [../]

  [./CCH1]
    type = CoreChannel
    fp = eos
    position = '0 0 0'
    orientation = '0 0 1'
    A = 8.79375e-5  #PWR, A = pitch^2 - PI * D_fuel * D_fuel / 4, pitch = 12.6 mm,
    D_h = 0.01179
    length = 3.865
    n_elems = 40

    f = 0.01
    Hw = 5.33e4
    initial_Ts = 559.15

    # fuel-gap-clad
    dim_hs = 2
    fuel_type = cylinder
    name_of_hs = 'fuel gap clad'
    n_heatstruct = 3  #fuel-gap-clad
    width_of_hs = '0.004096 0.0001 0.000552'
    elem_number_of_hs = '10 1 2'
    material_hs = 'fuel-mat gap-mat clad-mat'
    power = reactor
    power_fraction = '1.0 0.0 0.0'
  [../]
  [./inlet]
    type = InletDensityVelocity
    input = 'CCH1:pipe(in)'
    rho = 753.68
    vel = 4.43
  [../]
  [./outlet]
    type = Outlet
    input = 'CCH1:pipe(out)'
    p = '155.e5'
    legacy = true
  [../]
[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
  [../]
[]

[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 1.e-1
  dtmin = 1.e-5

  nl_rel_tol = 1e-8
  nl_abs_tol = 1e-6
  nl_max_its = 10

  l_tol = 1e-3
  l_max_its = 30

  start_time = 0.0
  end_time = 50.0
  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.6 Example 6: A Two Pipes Flow Problem

5.6.1 Problem Description

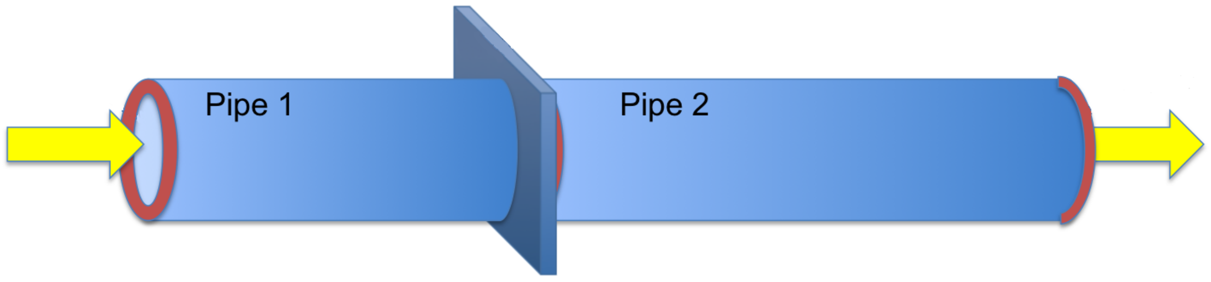

Figure 5. Diagram of a two pipes flow problem

This example demonstrates flow through two pipes with different flow areas connected by a junction. The *Junction* model is used in this example. The inlet boundary conditions are set by using the *InletDensityVelocity* component with density set at $\rho = 995Kg/m^3$ and fluid velocity set at $vel = 1m/s$. The outlet boundary condition is set by using the *Outlet* component with the outlet pressure set at $p = 0.95E5Pa$. The initial velocity, pressure, and temperature are $1m/s$, $1.0E5P$a, and $300K$ respectively. There is no heat exchange through the pipe walls for this case.

5.6.2 Input File

```
[GlobalParams]
  initial_p = 1.e5
  initial_vel = 1.
  initial_T = 300.
[]

[FluidProperties]
  [./eos]
```

```
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[Components]
  [./pipe1]
    type = Pipe
    fp = eos
    # geometry
    position = '0 0 0'
    orientation = '1 0 0'
    A = 2.
    f = 1.0
    length = 1
    n_elems = 20
    supg = true
  [../]

  [./pipe2]
    type = Pipe
    fp = eos
    # geometry
    position = '1 0  0'
    orientation = '1 0 0'
    A = 1.
    f = 1.0
    length = 1
    n_elems = 20
    supg = true
  [../]

  [./junction]
    type = Junction
    inputs = 'pipe1(out)'
    outputs = 'pipe2(in)'
    K = '1 1'
    A_ref = 1.0
  [../]
  [./inlet_1]
    type = InletDensityVelocity
    input = 'pipe1(in)'
    rho = 995
    vel = 1
  [../]

  [./outlet_2]
    type = Outlet
    input = 'pipe2(out)'
    p = 9.5e4
    legacy = true
  [../]
[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
    petsc_options_iname = '-pc_type -pc_factor_shift_type -pc_factor_shift_amount'
    petsc_options_value = ' lu       NONZERO              1e-10'
  [../]
[]

[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 1.e-1
  dtmin =1.0e-5

  nl_rel_tol = 1e-6
  nl_abs_tol = 1e-6
  nl_max_its = 10

  l_tol = 1e-3
  l_max_its = 30

  start_time = 0.0
  end_time = 10.0

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.7 Example 7: A Junction Case with Multiple Pipes In and Out

5.7.1 Problem Description

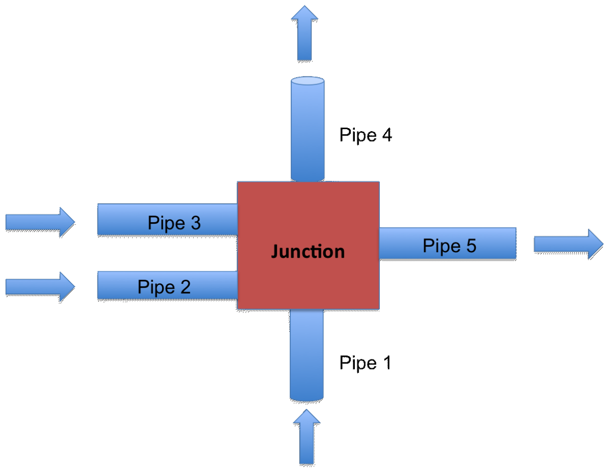

Figure 6. Diagram of a volume junction case with three pipes flowing in and two pipes flowing out

This example shows a junction case which connects two pipes with water flowing into the junction and three pipes with water flowing out of the junction. The *Junction* model is used in this example. The pipe flow areas are different and the boundary conditions also vary. The *InletDensityVelocity* component is used to set the boundary conditions at the inlets of the those two pipes with water flows into the junction. The *Outlet* component is used to set the boundary condition at the outlets of those three pipes with water flows out of the junction. In this example, the *HeatTransferFromSpecifiedTemperature* component is used to add heat to the water in each of the pipes.

5.7.2 Input File

```
[GlobalParams]
  initial_p = 1.e5
  initial_vel = 1.
  initial_T = 305.
```

```
[]

[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[Components]
  [./pipe1]
    type = Pipe
    fp = eos
    position = '0 0 0'
    orientation = '1 0 0'
    A = 2.
    f = 1.0
    length = 1
    n_elems = 50
  [../]

  [./ht_pipe1]
    type = HeatTransferFromSpecifiedTemperature
    pipe = pipe1
    T_wall = 310
    Hw = 1e4
  [../]
  [./pipe2]
    type = Pipe
    fp = eos
    position = '0 0.3 0'
    orientation = '1 0 0'
    A = 1.
    f = 1.0
    length = 1
    n_elems = 50
  [../]

  [./ht_pipe2]
    type = HeatTransferFromSpecifiedTemperature
    pipe = pipe2
    T_wall = 310
    Hw = 1e4
  [../]

  [./pipe3]
    type = Pipe
    fp = eos
    position = '1.2 0 0'
    orientation = '1 0 0'
    A = 2.0
    f = 1.0
    length = 1
    n_elems = 50
  [../]

  [./ht_pipe3]
    type = HeatTransferFromSpecifiedTemperature
    pipe = pipe3
    T_wall = 310
    Hw = 1e4
  [../]

  [./pipe4]
    type = Pipe
    fp = eos
    position = '1.2 0.5 0'
    orientation = '1 0 0'
    A = 1.
    f = 1.0
    length = 1
    n_elems = 50
  [../]

  [./ht_pipe4]
    type = HeatTransferFromSpecifiedTemperature
    pipe = pipe4
    T_wall = 310
    Hw = 1e4
  [../]

  [./pipe5]
    type = Pipe
    fp = eos
    position = '1.2 1.0 0'
    orientation = '1 0 0'
    A = 1.5
    f = 1.0
    length = 1
    n_elems = 50
  [../]

  [./ht_pipe5]
    type = HeatTransferFromSpecifiedTemperature
    pipe = pipe5
    T_wall = 310
    Hw = 1e4
  [../]
  [./junction]
    type = Junction
    inputs = 'pipe1(out) pipe2(out)'
    outputs = 'pipe3(in) pipe4(in) pipe5(in)'
    K = '5 7 11 13 17'
    A_ref = 3.0
  [../]

  [./inlet_1]
    type = InletDensityVelocity
    input = 'pipe1(in)'
    rho = 990
    vel = 1
  [../]

  [./inlet_2]
    type = InletDensityVelocity
```

```
    input = 'pipe2(in)'
    rho = 995
    vel = 1.5
  [../]

  [./outlet_3]
    type = Outlet
    input = 'pipe3(out)'
    p = 9.5e4
    legacy = true
  [../]

  [./outlet_4]
    type = Outlet
    input = 'pipe4(out)'
    p = 9.5e4
    legacy = true
  [../]
  [./outlet_5]
    type = Outlet
    input = 'pipe5(out)'
    p = 9.5e4
    legacy = true
  [../]
[]


[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
    petsc_options_iname = '-pc_type -pc_factor_shift_type -pc_factor_shift_amount'
    petsc_options_value = ' lu       NONZERO              1e-10'
  [../]
[]
```

```
[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 1.e-1
  dtmin = 1.e-4

  nl_rel_tol = 1e-8
  nl_abs_tol = 1e-6
  nl_max_its = 10

  l_tol = 1e-3
  l_max_its = 100

  start_time = 0.0
  end_time = 5

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.8 Example 8: A Simple Pipe Loop with a Pump

5.8.1 Problem Description

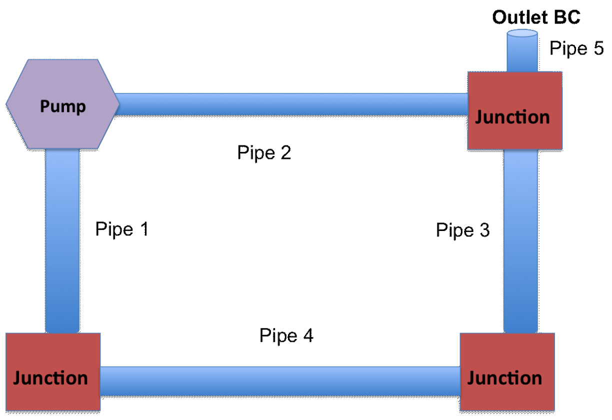

Figure 7. Diagram of a simple loop of pipes connected by junctions and a pump

This example shows a single phase flow loop made up of 5 pipes, 3 junctions and a pump. The *Junction* model is used for the three junctions. The pump acts as a junction, connecting pipes 1 and 2, and pipe 5 is used as an outlet to help control the pressure. The initial pressure of the system is 1.0e5 Pa and it is operating at a temperature of 300 K. The pipes all have the same length with the exception of pipe 5 which is smaller in length than the rest.

5.8.2 Input File

```
[GlobalParams]
  initial_T = 300
  initial_p = 1e5
  initial_vel = 0
[]
```

```
[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[Components]
  [./pipe1]
    type = Pipe
    fp = eos
    position = '0 0 0'
    orientation = '1 0 0'
    A = 0.785398163e-4    #1.0 cm (0.01 m) in diameter, A = 1/4 * PI * d^2
    D_h = 0.01
    f = 0.01
    length = 1
    n_elems = 20
  [../]

  [./pipe2]
    type = Pipe
    fp = eos
    position = '1 0 0'
    orientation = '0 1 0'
    A = 3.14159e-4    #2.0 cm (0.02 m) in diameter, A = 1/4 * PI * d^2
    D_h = 0.01
    f = 0.01
    length = 1
    n_elems = 20
  [../]

  [./pipe3]
    type = Pipe
    fp = eos
    position = '1 1 0'
    orientation = '-1 0 0'
    A = 0.785398163e-4    #1.0 cm (0.01 m) in diameter, A = 1/4 * PI * d^2
    D_h = 0.01
    f = 0.01
    length = 1
    n_elems = 20
  [../]
  [./pipe4]
    type = Pipe
    fp = eos
    position = '0 1 0'
    orientation = '0 -1 0'
    A = 3.14159e-4    #2.0 cm (0.02 m) in diameter, A = 1/4 * PI * d^2
    D_h = 0.01
    f = 0.01
    length = 1
    n_elems = 20
  [../]

  [./pipe5]
    type = Pipe
    fp = eos
    position = '1 1 0'
    orientation = '0 1 0'
    A = 0.785398163e-4    #1.0 cm (0.01 m) in diameter, A = 1/4 * PI * d^2
    D_h = 0.01
    f = 0.01
    length = 0.5
    n_elems = 20
  [../]

  [./pump]
    type = Pump
    fp = eos
    inputs = 'pipe1(out)'
    outputs = 'pipe2(in)'
    head = 1.0
    K_reverse = '10. 10.'
    A_ref = 0.785398163e-3
    initial_p = 1.e5
  [../]
```

```
  [./junction2]
    type = Junction
    inputs = 'pipe2(out)'
    outputs = 'pipe3(in) pipe5(in)'
    K = '3. 3. 3.'
    A_ref = 0.785398163e-3
  [../]

  [./junction3]
    type = Junction
    inputs = 'pipe3(out)'
    outputs = 'pipe4(in)'
    K = '3. 3.'
    A_ref = 0.785398163e-3
  [../]
  [./junction4]
    type = Junction
    inputs = 'pipe4(out)'
    outputs = 'pipe1(in)'
    K = '3. 3.'
    A_ref = 0.785398163e-3
  [../]

  [./outlet]
    type = Outlet
    input = 'pipe5(out)'
    p = '1.e5'
    legacy = true
  [../]
[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
    petsc_options_iname = '-pc_type -pc_factor_shift_type -pc_factor_shift_amount'
    petsc_options_value = ' lu       NONZERO               1e-10'
  [../]
[]
[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 1.e-1
  dtmin = 1e-7

  nl_rel_tol = 1e-9
  nl_abs_tol = 1e-7
  nl_max_its = 10

  l_tol = 1e-3
  l_max_its = 100

  start_time = 0.0
  end_time = 10

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.9 Example 9: A Heat Exchanger Problem

5.9.1 Problem Description

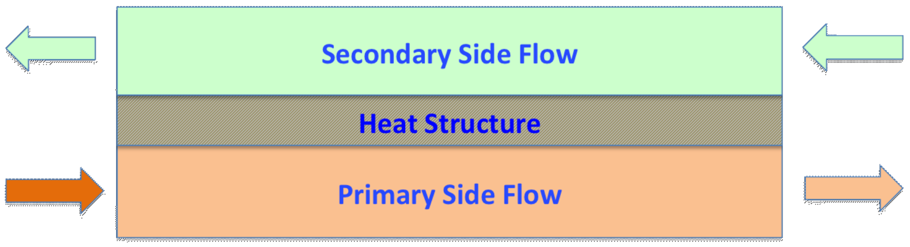

Figure 8. Diagram of a heat exchanger problem

This examples illustrates how to build a model to simulate heat transfer in an heat exchanger. The heat exchanger model consists of coolant flow in a pipe on the primary side, a heat structure, and coolant flow in a pipe on the secondary side. There are two components of *HeatTransferFromHeatStructure* used to simulate transfer transfer from the primary side pipe to heat structure and from heat structure to the secondary side pipe. The boundary conditions at the pipes' ends are set by *InletStagnationPressureTemperature* at the inlets and *Outlet* at the outlets. The inlet at the primary side lets in the hotter liquid flow, while the secondary side inlet lets in liquid at a cooler temperature. The secondary side liquid is used to cool down the primary side liquid so its outlet temperature is somewhat cooler while the secondary side outlet temperature is higher. This example problem is for subcooled water only. It is operating with inlet stagnation pressures of 1.05e5 Pa and outlet static pressures of 1.0e5 Pa. The inlet temperature for the primary side liquid is 400 K and 300 K for the secondary side liquid. The *HeatStructureMaterials* block is used to define the properties of the heat structure material.

5.9.2 Input File

```
[GlobalParams]
  initial_p = 1.0e5
  initial_vel = 1.
  initial_T = 300.0
  supg = true
  scaling_factor_1phase = '1.e-1 1.e-5 1.e-8'
  scaling_factor_temperature = 1e-2
[]

[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[HeatStructureMaterials]
  [./fuel-mat]
    type = SolidMaterialProperties
    k = 3.65
    Cp = 288.734
    rho = 1.0412e2
  [../]
  [./gap-mat]
    type = SolidMaterialProperties
    k = 1.084498
    Cp = 1.0
    rho = 1.0
  [../]
  [./clad-mat]
    type = SolidMaterialProperties
    k = 16.48672
    Cp = 321.384
    rho = 6.6e1
  [../]
  [./clad3-mat]
    type = SolidMaterialProperties
    k = 16.48672
    Cp = 6.6e3
    rho = 6.6e1
  [../]
  [./wall-mat]
    type = SolidMaterialProperties
    k = 100.0
    rho = 100.0
    Cp = 100.0
  [../]
[]


[Components]
  [./Primary_pipe]
    type = Pipe
    position = ’0 0.02 0’
    orientation = ’1 0 0’
    length = 1
    n_elems = 10
    fp = eos
    A = 0.785398163e-4
    D_h = 0.01
    f = 0.01
    heat_transfer_geom = PIPE
  [../]

  [./Wall]
    type = HeatStructure
    dim = 2
    position = ’0  0.005  0’
    orientation = ’1 0 0’
    initial_T = 300.0
    length = 1
    n_elems = 10
    names = ’solid_wall’
    widths = ’0.01’
    n_part_elems = ’2’
    materials = ’wall-mat’
    hs_type = PLATE
    depth = 1
  [../]
```

```
[./Secondary_pipe]
  type = Pipe
  position = '1 0  0'
  orientation = '-1 0 0'
  length = 1
  n_elems = 10
  fp = eos
  A = 0.785398163e-4
  D_h = 0.01
  f = 0.01
  heat_transfer_geom = PIPE
[../]

[./Hx_conn_pri]
  type = HeatTransferFromHeatStructure
  pipe = Primary_pipe
  hs = Wall
  hs_side = bottom
  Hw = 1.e4
[../]

[./Hx_conn_sec]
  type = HeatTransferFromHeatStructure
  pipe = Secondary_pipe
  hs = Wall
  hs_side = top
  Hw = 1.e4
[../]

[./inlet_primary_side]
  type = InletStagnationPressureTemperature
  input = 'Primary_pipe(in)'
  p0 = 1.05e5
  T0 = 400.0
[../]

[./outlet_primary_side]
```

```
    type = Outlet
    input = 'Primary_pipe(out)'
    p = 1.0e5
    legacy = true
  [../]

  [./inlet_secondary_side]
    type = InletStagnationPressureTemperature
    input = 'Secondary_pipe(in)'
    p0 = 1.05e5
    T0 = 300.0
  [../]

  [./outlet_secondary_side]
    type = Outlet
    input = 'Secondary_pipe(out)'
    p = 1.0e5
    legacy = true
  [../]

[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
    petsc_options_iname = '-pc_type -pc_factor_shift_type -pc_factor_shift_amount'
    petsc_options_value = ' lu       NONZERO               1e-10'
  [../]
[]

[Executioner]
  type = Transient
  scheme = 'bdf2'
  dt = 1.e-1
  dtmin = 1.e-4
  nl_rel_tol = 1e-8
  nl_abs_tol = 1e-6
  nl_max_its = 10

  l_tol = 1e-3
  l_max_its = 100

  start_time = 0.0
  end_time = 10

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out_displaced]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.10 Example 10: A Loop With Core Channel and Heat Exchanger

5.10.1 Problem Description

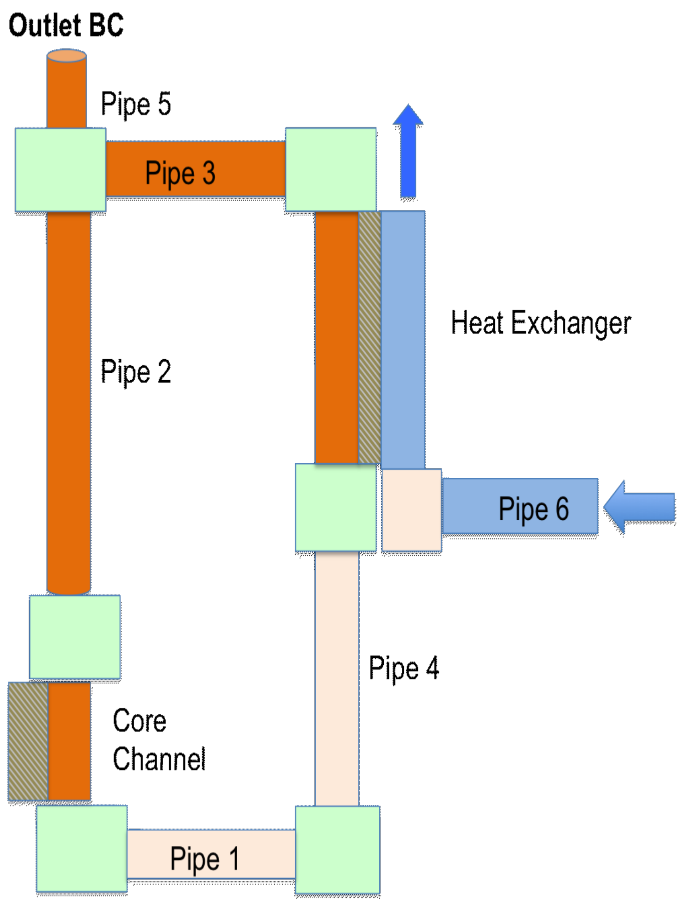

Figure 9. Diagram of a loop with core channel and heat exchanger

This example is a simple loop which consists of 6 pipes, a core channel, and a heat exchanger. Each component differs in length, but has the same area and diameter. The core channel is used to heat up the liquid, while the heat exchanger is used to cool it down. Pipe 5, with a pressure of 2.0e5 Pa imposed at pipe outlet, is used to control the system pressure.

The initial pressure, velocity, and temperature are 2.0e5 Pa, 1.0 m/s, and 300.15 K. The coolant is operating with an inlet temperature of 300.15 K and mass flow rate of 223.77 Kg/s, and the outlet pressure of 2.0e5 Pa on the secondary side of the heat exchanger.

5.10.2 Input File

```
[GlobalParams]
  initial_p = 2.0e5
  initial_vel = 1.0
  initial_T = 300.15
  scaling_factor_1phase = '1.e0 1.e-2 1.e-5'
  scaling_factor_temperature = 1e-2

  stabilization = evm1

[]

[Stabilizations]
  [./evm1]
    type = EntropyViscosity
    use_first_order = true
  [../]
[]

[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[HeatStructureMaterials]
  [./fuel-mat]
    type = SolidMaterialProperties
    k = 29.3
    Cp = 191.67
    rho = 1.4583e4
  [../]
  [./clad-mat]
    type = SolidMaterialProperties
    k = 26.3
    Cp = 638
    rho = 7.646e3
  [../]

  [./wall-mat]
    type = SolidMaterialProperties
    k = 26.3
    rho = 7.646e3
    Cp = 638
  [../]
[]

[Components]
  [./reactor]
    type = PrescribedReactorPower
    function = 5.1296e7
  [../]

  [./pipe1]
    type = Pipe
    fp = eos
    position = ’0 1 0’
    orientation = ’0 -1 0’

    A = 0.44934
    D_h = 2.972e-3
    length = 1
    n_elems = 20
    f = 0.001
  [../]

  [./CH1]
    type = CoreChannel
    fp = eos
    position = ’0 0 0’
    orientation = '0 0 1'

    A = 0.44934
    D_h = 2.972e-3
    length = 0.8
    n_elems = 20

    f = 0.022
    Hw = 1.6129e5 #liquid metal
    P_hf = 497.778852000000

    name_of_hs = 'fuel clad'
    initial_Ts = 300.15
    n_heatstruct = 2
    fuel_type = cylinder
    width_of_hs = '0.00348  0.00052'
    elem_number_of_hs = '4 1'
    material_hs = 'fuel-mat clad-mat'
    power = reactor
    power_fraction = '1.0 0.0'
  [../]
  [./pipe2]
    type = Pipe
    fp = eos
    position = '0 0 0.8'
    orientation = '0 0 1'

    A = 0.44934
    D_h = 2.972e-3
    length = 5.18
    n_elems = 50
    f = 0.001
  [../]

  [./pipe3]
    type = Pipe
    fp = eos
    position = '0 0 5.98'
    orientation = '0 1 0'

    A = 0.44934
    D_h = 2.972e-3
    length = 1
    n_elems = 20
    f = 0.001
  [../]

  [./IHX:primary_pipe]
    type = Pipe
    position = '0 1.0 5.98'
    orientation = '0 0 -1'
    length = 3.71
    n_elems = 50
    fp = eos
    A = 0.44934
    D_h = 0.0186
    f = 0.022
  [../]

  [./IHX:wall]
    type = HeatStructure
    dim = 2
    position = '0 1.1 5.98'
    orientation = '0 0 -1'
    initial_T = 300.15
    length = 3.71
    n_elems = 10
    names = 'solid_wall'
    widths = '0.0044'
    depth = 1
    n_part_elems = '2'
    materials = 'wall-mat'
    hs_type = PLATE
  [../]

  [./IHX:secondary_pipe]
```

```
  type = Pipe
  position = '0 1.2044 5.98'
  orientation = '0 0 -1'
  length = 3.71
  n_elems = 50
  fp = eos
  A = 0.44934
  D_h = 0.014
  f = 0.022
[../]

[./IHX:hx_conn_pri]
  type = HeatTransferFromHeatStructure
  pipe = IHX:primary_pipe
  hs = IHX:wall
  hs_side = bottom
  Hw = 1.6129e5 # the same as the core
  P_hf = 327.568860000000
[../]

[./IHX:hx_conn_sec]
  type = HeatTransferFromHeatStructure
  pipe = IHX:secondary_pipe
  hs = IHX:wall
  hs_side = top
  Hw = 1.6129e5 #the same as the core
  P_hf = 327.568860000000
[../]

[./pipe4]
  type = Pipe
  fp = eos
  position = '0 1.0 2.27'
  orientation = '0 0 -1'

  A = 0.44934
  D_h = 2.972e-3
  length = 2.27
  n_elems = 50
  f = 0.001
[../]

[./pipe5]
  type = Pipe
  fp = eos
  position = ’0 0 5.98’
  orientation = ’0 0 1’

  A =  0.44934
  D_h = 2.972e-3
  length = 0.02
  n_elems = 10
  f = 10
[../]


[./Junction1]
  type = Junction
  inputs = ’pipe1(out)’
  outputs = ’CH1:pipe(in)’
  K = ’0.5 0.5’
  A_ref = 0.44934
[../]

[./Junction2]
  type = Junction
  inputs = ’CH1:pipe(out)’
  outputs = ’pipe2(in)’
  K = ’0.5 0.5’
  A_ref =  0.44934
[../]
[./Junction3]
  type = Junction
  inputs = ’pipe2(out)’
  outputs = ’pipe3(in) pipe5(in)’
  K = ’0.0 0.0 0.0’
  A_ref =   0.44934
[../]

[./Junction4]
  type = Junction
  inputs = 'pipe3(out)'
  outputs = 'IHX:primary_pipe(in)'
  K = '0.1 0.1'
  A_ref = 0.44934
[../]

[./Junction5]
  type = Junction
  inputs = 'IHX:primary_pipe(out)'
  outputs = 'pipe4(in)'
  K = '0.0 0.0'
  A_ref = 0.44934
[../]

[./Junction6]
  type = Junction
  inputs = 'pipe4(out)'
  outputs = 'pipe1(in)'
  K = '0.0 0.0'
  A_ref = 0.44934
[../]

[./pipe6]
  type = Pipe
  fp = eos
  position = '0 1.5 2.27'
  orientation = '0 -1 0'

  A = 0.44934
  D_h = 2.972e-3
  length = 0.3
  n_elems = 5
  f = 0.001
  [../]

  [./Junction7]
    type = Junction
    inputs = 'pipe6(out)'
    outputs = 'IHX:secondary_pipe(in)'
    K = '0.0 0.0'
    A_ref = 0.44934
  [../]
  [./inlet]
    type = InletMassFlowRateTemperature
    input = 'pipe6(in)'
    m_dot = 223.77
    T = 300.15
  [../]
  [./outlet]
    type = Outlet
    input = 'IHX:secondary_pipe(out)'
    p = 2.0e5
    legacy = true
  [../]

  [./prezr]
    type = Outlet
    input = 'pipe5(out)'
    p = 2e5
    reversible = true
    legacy = true
  [../]
[]

[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    petsc_options_iname = '-pc_type -pc_factor_shift_type -pc_factor_shift_amount'
    petsc_options_value = ' lu       NONZERO               1e-10'
  [../]
[]


[Executioner]
  type = Transient
  scheme = 'bdf2'

  dt = 1e-1
  dtmin = 1e-5

  nl_rel_tol = 1e-6
  nl_abs_tol = 1e-6
  nl_max_its = 30

  l_tol = 1e-3
  l_max_its = 30

  start_time = 0.0
  end_time = 50.0

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]

[Outputs]
  [./out]
    type = Exodus
    use_displaced = true
    sequence = false
  [../]
[]
```

5.11 Example 11: A Model Pressurized Water Reactor Problem

5.11.1 Problem Description

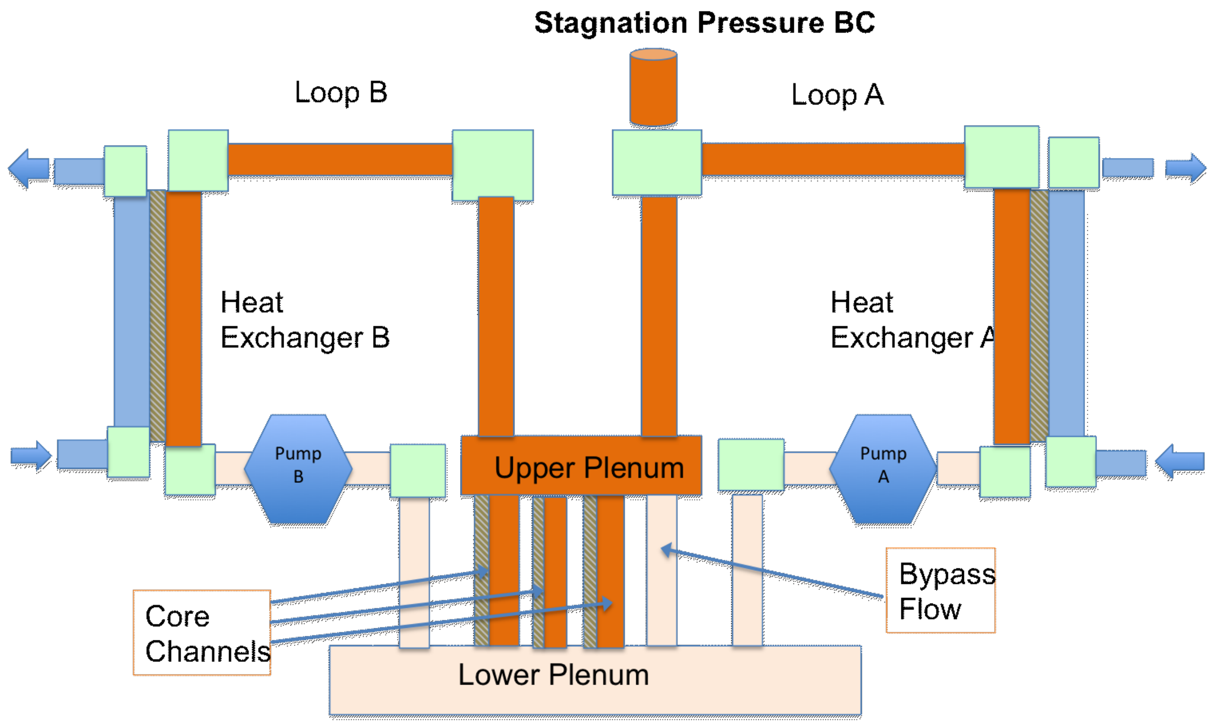

Figure 10. Diagram of a model pressurized water reactor problem

This example shows a simplified PWR plant model. It is made up of three sections - Loop A, Loop B and a reator vessel model. The reactor vessel region contains three parallel core channels, a bypass flow channel, and an upper and lower plenum. The three core channels represent all the cooling channels and fuel rods in the high power region, average power region and low power region of the reactor core respectively. The upper and lower plenum are modeled with junctions. The two loops have a Hot Leg, a Heat Exchanger and its secondary side pipes, the Cold Leg and a primary Pump. The details of the Heat Exchanger modeling can be found in Example 9. The heat exchanger secondary side pipes are modeled with subcooled water. Loop A contains a *InletStagnationPressureTemperature* component that works as a pressurizer to help regulate the system pressure.

The boundary conditions are set with the mass flow rate at 9419.64 kg/s and the temperature at 564.15 K at the inlet of the secondary pipe of the Heat Exchanger for each of of the two loops. The secondary pipe outlet pressure is set at 15.17 MPa. The pressurizer operates at a temperature of 564.15 K and a pressure of 15.17 MPa.

5.11.2 Input File

```
[GlobalParams]
  initial_p = 15.17e6
  initial_vel = 1.
  initial_T = 564.15

  scaling_factor_1phase = '1.e0 1.e-2 1.e-5'
  scaling_factor_temperature = 1e-2

  stabilization = evm1
[]

[Stabilizations]
  [./evm1]
    type = EntropyViscosity
    use_first_order = true
  [../]
[]

[FluidProperties]
  [./eos]
    type = IAPWS95LiquidFluidProperties
  [../]
[]

[HeatStructureMaterials]
  [./fuel-mat]
    type = SolidMaterialProperties
    k = 3.65
    Cp = 288.734
    rho = 1.0412e2
  [../]
  [./gap-mat]
    type = SolidMaterialProperties
    k = 1.084498
    Cp = 1.0
    rho = 1.0
  [../]
  [./clad-mat]
    type = SolidMaterialProperties
    k = 16.48672
    Cp = 321.384
    rho = 6.6e1
  [../]
  [./clad3-mat]
    type = SolidMaterialProperties
    k = 16.48672
    Cp = 6.6e3
    rho = 6.6e1
  [../]

  [./wall-mat]
    type = SolidMaterialProperties
    k = 100.0
    rho = 100.0
    Cp = 100.0
  [../]
[]


[Components]
  [./reactor]
    type = PrescribedReactorPower
    function = 2.77199979e9
  [../]

  #Core region components ########################################################
```

```
[./CH1]
  type = CoreChannel
  fp = eos
  position = ’0 -1.2 0’
  orientation = ’0 0 1’
  A = 1.161864
  D_h = 0.01332254
  length = 3.6576
  n_elems = 8

  f = 0.01
  Hw = 5.33e4
  n_rods = 9360
  initial_Ts = 564.15

  n_heatstruct = 3
  name_of_hs = ’FUEL GAP CLAD’
  fuel_type = cylinder
  width_of_hs = ’0.0046955  0.0000955  0.000673’
  elem_number_of_hs = ’9 3 3’
  material_hs = ’fuel-mat gap-mat clad-mat’
  power = reactor
  power_fraction = ’3.33672612e-1 0 0’
[../]

[./CH2]
  type = CoreChannel
  fp = eos
  position = ’0 0 0’
  orientation = ’0 0 1’
  A = 1.549152542
  D_h = 0.01332254
  length = 3.6576
  n_elems = 8

  f = 0.01
  Hw = 5.33e4
  n_rods = 12480
  initial_Ts = 564.15

  n_heatstruct = 3
  name_of_hs = 'FUEL GAP CLAD'
  fuel_type = cylinder
  width_of_hs = '0.0046955  0.0000955  0.000673'
  elem_number_of_hs = '9 3 3'
  material_hs = 'fuel-mat gap-mat clad-mat'
  power = reactor
  power_fraction = '3.69921461e-1 0 0'
[../]

[./CH3]
  type = CoreChannel
  fp = eos
  position = '0 1.2 0'
  orientation = '0 0 1'
  A = 1.858983051
  D_h = 0.01332254
  length = 3.6576
  n_elems = 8

  f = 0.01
  Hw = 5.33e4
  n_rods = 14976
  initial_Ts = 564.15

  n_heatstruct = 3
  name_of_hs = 'FUEL GAP CLAD'
  fuel_type = cylinder
  width_of_hs = '0.0046955  0.0000955  0.000673'
  elem_number_of_hs = '9 3 3'
  material_hs = 'fuel-mat gap-mat clad3-mat'
  power = reactor
  power_fraction = '2.96405926e-1 0 0'
[../]

[./bypass_pipe]
```

```
  type = Pipe
  fp = eos
  position = '0 1.5 0'
  orientation = '0 0 1'
  A = 1.589571014
  D_h = 1.42264
  length = 3.6576
  n_elems = 5

  f = 0.001
[../]

[./LowerPlenum]
  type = Junction
  inputs = 'DownComer-A(out) DownComer-B(out)'
  outputs = 'CH1:pipe(in) CH2:pipe(in) CH3:pipe(in) bypass_pipe(in)'
  K = '0.2 0.2 0.2 0.2 0.4 40.0'
  A_ref = 3.618573408
  scaling_factors = '1e-3 1'
[../]

[./UpperPlenum]
  type = Junction
  inputs = 'CH1:pipe(out) CH2:pipe(out) CH3:pipe(out) bypass_pipe(out)'
  outputs = 'pipe1-HL-A(in) pipe1-HL-B(in)'
  K = '0.5 0.5 0.5 80.0 0.5 0.5'
  A_ref = 7.562307456
  scaling_factors = '1e-3 1'
[../]
######################################################################################

#Loop A components ###################################################################
[./DownComer-A]
  type = Pipe
  fp = eos
  position = '0 2.0 4.0'
  orientation = '0 0 -1'
  A = 3.6185734
  D_h = 1.74724302
  length = 4
  n_elems = 3

  f = 0.001
[../]

[./pipe1-HL-A]
  type = Pipe
  fp = eos
  position = '0 0.5 4.0'
  orientation = '0 0 1'
  A = 7.562307456
  D_h = 3.103003207
  length = 4.
  n_elems = 3

  f = 0.001
[../]

[./pipe2-HL-A]
  type = Pipe
  fp = eos
  position = '0 0.5 8.0'
  orientation = '0 1 0'
  A = 2.624474
  D_h = 1.828
  length = 3.5
  n_elems = 3

  f = 0.001
[../]

[./pipe1-CL-A]
  type = Pipe
  fp = eos
  position = '0 3.0 4.0'
  orientation = '0 -1 0'
  A = 2.624474
  D_h = 1.828
  length = 1.
  n_elems = 3

  f = 0.001
[../]

[./pipe2-CL-A]
  type = Pipe
  fp = eos
  position = '0 4 4.0'
  orientation = '0 -1 0'
  A = 2.624474
  D_h = 1.828
  length = 0.8
  n_elems = 3

  f = 0.001
[../]

[./pipe1-SC-A]
  type = Pipe
  fp = eos
  position = '0 5.2 4.0'
  orientation = '0 -1 0'
  A = 2.624474
  D_h = 1.828
  length = 1.
  n_elems = 3

  f = 0.001
[../]

[./pipe2-SC-A]
  type = Pipe
  fp = eos
  position = '0 4.2 8.0'
```

```
    orientation = '0 1 0'
    A = 2.624474
    D_h = 1.828
    length = 1.
    n_elems = 3

    f = 0.001
  [../]

  [./Junction1-A]
    type = Junction
    inputs = 'pipe1-HL-A(out)'
    outputs = 'pipe2-HL-A(in) pipe-to-Pressurizer(in)'
    K = '0.5 0.7 80.'
    A_ref = 7.562307456
    scaling_factors = '1e-3 1'
[../]

  [./Junction2-A]
    type = Junction
    inputs = 'pipe1-CL-A(out)'
    outputs = 'DownComer-A(in)'
    K = '0.5 0.7'
    A_ref = 3.6185734
    scaling_factors = '1e-3 1'
  [../]

  [./Junction3-A]
    type = Junction
    inputs = 'pipe2-HL-A(out)'
    outputs = 'HX-A:primary_pipe(in)'
    K = '0.5 0.7'
    A_ref = 2.624474
    scaling_factors = '1e-3 1'
  [../]

  [./Pump-A]
    type = Pump
    fp = eos
    inputs = 'pipe2-CL-A(out)'
    outputs = 'pipe1-CL-A(in)'
    A_ref = 2.624474
    K_reverse = '0 0'
    head = 8.66
  [../]


  [./HX-A:primary_pipe]
    type = Pipe
    position = '0 4 8'
    orientation = '0 0 -1'
    length = 4
    n_elems = 10
    fp = eos
    A = 5.
    D_h = 0.01
    f = 0.01
  [../]

  [./HX-A:wall]
    type = HeatStructure
    dim = 2
    position = '0 4.1 8'
    orientation = '0 0 -1'
    initial_T = 564.15
    length = 4
    n_elems = 10
    names = 'solid_wall'
    widths = '0.001'
    n_part_elems = '2'
    materials = 'wall-mat'
    hs_type = PLATE
    depth = 1
  [../]

  [./HX-A:secondary_pipe]
```

```
  type = Pipe
  position = '0 4.201 4'
  orientation = '0 0 1'
  length = 4
  n_elems = 10
  fp = eos
  A = 5.
  D_h = 0.01
  f = 0.01
[../]

[./HX-A:hx_conn_pri]
  type = HeatTransferFromHeatStructure
  pipe = HX-A:primary_pipe
  hs = HX-A:wall
  hs_side = bottom
  Hw = 1.e4
  P_hf = 2695.1
[../]

[./HX-A:hx_conn_sec]
  type = HeatTransferFromHeatStructure
  pipe = HX-A:secondary_pipe
  hs = HX-A:wall
  hs_side = top
  Hw = 1.e4
  P_hf = 2695.1
[../]

[./Junction4-A]
  type = Junction
  inputs = 'pipe1-SC-A(out)'
  outputs = 'HX-A:secondary_pipe(in)'
  K = '0.5 0.7'
  A_ref = 2.624474e2
  scaling_factors = '1e-3 1'
[../]
```

```
[./Junction5-A]
  type = Junction
  inputs = 'HX-A:secondary_pipe(out)'
  outputs = 'pipe2-SC-A(in)'
  K = '0.5 0.7'
  A_ref = 2.624474e2
  scaling_factors = '1e-3 1'
[../]

[./Junction6-A]
  type = Junction
  inputs = 'HX-A:primary_pipe(out)'
  outputs = 'pipe2-CL-A(in)'
  K = '0.5 0.7'
  A_ref = 2.624474e2
  scaling_factors = '1e-3 1'
[../]

[./MassFlowRateIn-SC-A]
  type = InletMassFlowRateTemperature
  input = 'pipe1-SC-A(in)'
  m_dot = 9419.640610595952
  T = 564.15
[../]
[./PressureOutlet-SC-A]
  type = Outlet
  input = 'pipe2-SC-A(out)'
  p = '151.7e5'
[../]
#############################################################################################

#Loop B components ###########################################################################
[./DownComer-B]
  type = Pipe
  fp = eos
  position = '0 -2.0 4.0'
  orientation = '0 0 -1'
  A = 3.6185734
  D_h = 1.74724302
  length = 4
  n_elems = 3

  f = 0.001
[../]

[./pipe1-HL-B]
  type = Pipe
  fp = eos
  position = '0 -0.5 4.0'
  orientation = '0 0 1'
  A = 7.562307456
  D_h = 3.103003207
  length = 4.
  n_elems = 3

  f = 0.001
[../]

[./pipe2-HL-B]
  type = Pipe
  fp = eos
  position = '0 -0.5 8.0'
  orientation = '0 -1 0'
  A = 2.624474
  D_h = 1.828
  length = 3.5
  n_elems = 3

  f = 0.001
[../]

[./pipe1-CL-B]
  type = Pipe
  fp = eos
  position = '0 -3.0 4.0'
  orientation = '0 1 0'
  A = 2.624474
  D_h = 1.828
  length = 1.
  n_elems = 3

  f = 0.001
[../]

[./pipe2-CL-B]
  type = Pipe
  fp = eos
  position = '0 -4.0 4.0'
  orientation = '0 1 0'
  A = 2.624474
  D_h = 1.828
  length = 0.8
  n_elems = 3

  f = 0.001
[../]

[./pipe1-SC-B]
  type = Pipe
  fp = eos
  position = '0 -5.2 4.0'
  orientation = '0 1 0'
  A = 2.624474
  D_h = 1.828
  length = 1.
  n_elems = 3

  f = 0.001
[../]

[./pipe2-SC-B]
  type = Pipe
  fp = eos
  position = '0 -4.2 8.0'
  orientation = '0 -1 0'
  A = 2.624474
  D_h = 1.828
  length = 1.
  n_elems = 3

  f = 0.001
[../]

[./Junction1-B]
  type = Junction
  inputs = 'pipe1-HL-B(out)'
  outputs = 'pipe2-HL-B(in)'
  K = '0.5 0.7'
  A_ref = 7.562307456
  scaling_factors = '1e-3 1'
[../]

[./Junction2-B]
  type = Junction
  inputs = 'pipe1-CL-B(out)'
  outputs = 'DownComer-B(in)'
  K = '0.5 0.7'
  A_ref = 3.6185734
  scaling_factors = '1e-3 1'
[../]

[./Junction3-B]
  type = Junction
  inputs = 'pipe2-HL-B(out)'
  outputs = 'HX-B:primary_pipe(in)'
  K = '0.5 0.7'
  A_ref = 2.624474
  scaling_factors = '1e-3 1'
[../]

[./Pump-B]
  type = Pump
```

```
    fp = eos
    inputs = 'pipe2-CL-B(out)'
    outputs = 'pipe1-CL-B(in)'
    A_ref = 2.624474
    K_reverse = '0 0'
    head = 8.66
  [../]

  [./HX-B:primary_pipe]
    type = Pipe
    position = '0 -4 8'
    orientation = '0 0 -1'
    length = 4
    n_elems = 10
    fp = eos
    A = 5.
    D_h = 0.01
    f = 0.01
  [../]

  [./HX-B:wall]
    type = HeatStructure
    dim = 2
    position = '0 -4.101 8'
    orientation = '0 0 -1'
    axial_offset = 0
    initial_T = 564.15
    length = 4
    n_elems = 10
    names = 'solid_wall'
    widths = '0.001'
    n_part_elems = '2'
    materials = 'wall-mat'
    hs_type = PLATE
    depth = 1
  [../]

  [./HX-B:secondary_pipe]
```

```
    type = Pipe
    position = '0 -4.201 4'
    orientation = '0 0 1'
    length = 4
    n_elems = 10
    fp = eos
    A = 5.
    D_h = 0.01
    f = 0.01
  [../]

  [./HX-B:hx_conn_pri]
    type = HeatTransferFromHeatStructure
    pipe = HX-B:primary_pipe
    hs = HX-B:wall
    hs_side = top
    Hw = 1.e4
    P_hf = 2695.1
  [../]

  [./HX-B:hx_conn_sec]
    type = HeatTransferFromHeatStructure
    pipe = HX-B:secondary_pipe
    hs = HX-B:wall
    hs_side = bottom
    Hw = 1.e4
    P_hf = 2695.1
  [../]

  [./Junction4-B]
    type = Junction
    inputs = 'pipe1-SC-B(out)'
    outputs = 'HX-B:secondary_pipe(in)'
    K = '0.5 0.7'
    A_ref = 2.624474e2
    scaling_factors = '1e-3 1'
  [../]
```

```
[./Junction5-B]
  type = Junction
  inputs = 'HX-B:secondary_pipe(out)'
  outputs = 'pipe2-SC-B(in)'
  K = '0.5 0.7'
  A_ref = 2.624474e2
  scaling_factors = '1e-3 1'
[../]

[./Junction6-B]
  type = Junction
  inputs = 'HX-B:primary_pipe(out)'
  outputs = 'pipe2-CL-B(in)'
  K = '0.5 0.7'
  A_ref = 2.624474e2
  scaling_factors = '1e-3 1'
[../]

[./MassFlowRateIn-SC-B]
  type = InletMassFlowRateTemperature
  input = 'pipe1-SC-B(in)'
  m_dot = 9419.640610595952
  T = 564.15
[../]
[./PressureOutlet-SC-B]
  type = Outlet
  input = 'pipe2-SC-B(out)'
  p = 15.17e6
[../]
######################################################################################


# Pressurizer ########################################################################
[./pipe-to-Pressurizer]
  type = Pipe
  fp = eos
  position = '0 0.5 8.0'
  orientation = '0 0 1'
```

```
    A = 2.624474
    D_h = 1.828
    length = 0.5
    n_elems = 3

    f = 10.
  [../]

  [./Pressurizer]
    type = InletStagnationPressureTemperature
    input = 'pipe-to-Pressurizer(out)'
    p0 = 15.17e6
    T0 = 564.15
    reversible = true
  [../]
  ################################################################################
[]



[Preconditioning]
  [./SMP_PJFNK]
    type = SMP
    full = true
    solve_type = 'PJFNK'
    line_search = basic
    petsc_options_iname = '-pc_type -pc_factor_shift_type -pc_factor_shift_amount'
    petsc_options_value = ' lu       NONZERO               1e-10'
  [../]
[]

[Executioner]
  type = Transient
  scheme = 'bdf2'

  nl_rel_tol = 1e-6
  nl_abs_tol = 1e-6
  nl_max_its = 10
  l_tol = 1e-3
  l_max_its = 30

  start_time = 0.0
  end_time = 200
  [./TimeStepper]
    type = SolutionTimeAdaptiveDT
    dt = 0.01
  [../]
  dtmin = 1e-6

  [./Quadrature]
    type = TRAP
    order = FIRST
  [../]
[]


[Outputs]
  [./exodus]
    type = Exodus
    file_base = TMI_2loop
    use_displaced = true
    sequence = false
    execute_on = 'initial final'
  [../]
  [./console]
    type = Console
    execute_scalars_on = none
  [../]
[]
```

References

[1] D. A. Knoll and D. E. Keyes, "Jacobian-free Newton-Krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, vol. 193, pp. 357–397, Jan. 2004. `http://dx.doi.org/10.1016/j.jcp.2003.08.010`.

[2] P. N. Brown and A. C. Hindmarsh, "Matrix-free methods for stiff systems of ODEs," *SIAM J. Numer. Anal.*, vol. 23, pp. 610–638, June 1986. `http://www.jstor.org/stable/2157527`.

[3] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "`libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations," *Engineering with Computers*, vol. 22, no. 3–4, pp. 237–254, 2006. `http://dx.doi.org/10.1007/s00366-006-0049-3`.

[4] S. Balay, W. D. Gropp, L. Curfman-McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing* (E. Arge, A. M. Bruaset, and H. P. Langtangen, eds.), pp. 163–202, Birkhäuser Press, 1997.

[5] M. Heroux *et al.*, "An overview of Trilinos," Tech. Rep. SAND2003-2927, Sandia National Laboratories, 2003.

[6] H. Zhang, L. Zou, D. Andrs, H. Zhao, and R. C. Martineau, "Point Kinetics Calculations with Fully Coupled Thermal Fluids Reactivity Feedback," in *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2013)*, (Sun Valley, Idaho, USA), May 5–9, 2013.

[7] L. Zou, J. Peterson, H. Zhao, H. Zhang, D. Andrs, and R. C. Martineau, "Solving Multi-Mesh Flow and Conjugate Heat Transfer Problems with RELAP-7," in *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2013)*, (Sun Valley, Idaho, USA), May 5–9, 2013.

[8] R. D. Falgout and U. M. Yang, "HYPRE: A Library of High Performance Preconditioners," in *International Conference on Computational Science*, pp. 632–641, 2002.

v1.28

Idaho National Laboratory