Annals of Nuclear Energy 109 (2017) 712–719

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

Benchmark and demonstration of the CHD code for transient analysis of fast reactor systems

C. Hellesen [a,*], S. Qvist [a,b]

[a] Uppsala University, Uppsala, Sweden
[b] Department of Nuclear Engineering, University of California Berkeley, USA

ARTICLE INFO

Article history:
Received 20 January 2017
Received in revised form 5 May 2017
Accepted 10 May 2017

Keywords:
Fast reactor
Thermal-hydraulics
Point-kinetics
Transient
Passive safety
ULOF

ABSTRACT

In this paper the dynamic thermal hydraulic fast reactor simulation code CHD is presented. The code is built around a scriptable object-oriented framework in the programming language Python to be able to flexibly describe different reactor geometries including thermal-hydraulics models of an arbitrary number of coolant channels as well as pumps, heat-exchangers and pools etc. In addition, custom objects such as the Autonomous Reactivity Control (ARC) system for enhanced passive safety are modeled in detail.

In this paper we compare the performance of the CHD code with other similar fast reactor dynamics codes using a benchmark study of the European Sodium cooled Fast Reactor (ESFR). The results agree well, both qualitatively and quantitatively with the code benchmark. In addition, we demonstrate the code's ability to simulate the long-term asymptotic behavior of a neutronically shut down reactor in an unprotected loss of flow scenario using a model of the Advanced Burner Reactor (ABR).

1. Introduction

A dynamic fast reactor simulation code, CHD, has been developed at Uppsala University. The main purpose of the code is for scientific as well as educational work. It was identified that a flexible and modular code was needed that was capable of building reactor geometries with custom designed elements as well as being able to easily modify the reactor geometry and behavior of its parts in a scripting language. The CHD code is fully object oriented and is written entirely in the scripting language Python and contains no interfaces to other code environments. Numerical calculations were done with the standard packages numpy and scipy.

All elements in the CHD code are designed as Python classes, and elements with custom functionality can be designed by sub classing already existing classes in the code. One example is a custom behavior of the reactor's control system, such as control rods and coolant pumps, to study the reactor's response to a load following situation. Other applications where the code is used are a dynamic modeling of the Autonomous Reactivity Control (ARC) shutdown system (Qvist, 2016a). Such types of reactivity systems have very specific reactivity responses, which can make them difficult to model in many existing fast reactor dynamics codes. The dynamics of the ARC system is rather complex and needs to be simulated a priori in a CFD code and later parameterized for use in a transient code. CHD is designed to easily incorporate this.

Further examples of applications are automated transient safety analyses within the reactor optimization code ADOPT (Qvist and Greenspan, 2014). By means of trial and error, the ADOPT code finds an optimal core design when given a set of optimization criteria. By including a transient safety analysis in the optimization, the core's robustness with respect to serious accident scenarios, such as the Unprotected Loss Of Flow (ULOF), can also be included within the optimization criteria. The object oriented structure of the CHD code with no interfaces to other code environments makes the interfacing with ADOPT straight forward since it is also written entirely in the Python language. The results of the transient safety optimization will be presented in an accompanying paper.

In this paper, the structure of the CHD code is presented along with a comparative benchmark with other established codes. The benchmark simulates an Unprotected Loss Of Flow (ULOF) accident scenario in the European Sodium cooled Fast Reactor (ESFR) and is presented in (Lázaro et al., 2014). Both a full ULOF as well as a partial ULOF are simulated. In the former case the coolant reaches boiling after about 20–35 s, and in the latter case, the reactor stabilizes at a lower power that is ultimately set by the heat rejection capability of the water flow in the turbine.

This benchmark was chosen because it comprises several transient fast reactor codes, and it also includes all three circuits of a

* Corresponding author at: Ångströmslaboratoriet, Lägerhyddsvägen 1, 752 37 Uppsala, Sweden.
E-mail address: carl.hellesen@physics.uu.se (C. Hellesen).

http://dx.doi.org/10.1016/j.anucene.2017.05.031
0306-4549/

typical Sodium Fast Reactor (SFR), i.e., primary loop, secondary loop and steam cycle. In the full ULOF, the reactor's response within the first half minute is studied before the coolant reaches boiling. Here it is mainly the core and the primary coolant that are important as the progression of the accident scenario is too rapid for any significant heat transfer between the primary and the secondary coolant loops to take place. In the partial ULOF, the reactor's behavior over several minutes is studied and the responses of the secondary and tertiary coolant loops become important as well.

The benchmark in (Lázaro et al., 2014) did not include modeling of the coolant velocity dependent pressure drop in the core as well as the long-term asymptotic behavior of a SFR that does not reach coolant boiling. Therefore, in addition to the ESFR benchmark discussed above, we also study the long-term behavior (several days) during a station blackout (SBO) scenario in the Advanced Burner Reactor (ABR) (Kim, 2009). We chose a metal fuel design with a conversion ratio of 0.75 since it has very favorable properties with respect to passive safety.

The main functions of the CHD code are conceptually similar to those of other multi-channel point-kinetics based transient analysis codes such as SAS4A/SASSYS-1 [ANL/NE-12/4, 2011], THACOS (Hu et al., 2013), SSC-L (Agrawal, 1978) and MAT5-DYN (Darmet and Massara, 2012). The major additions being the ability to explicitly model certain customized components, such as the operation of the ARC systems with great flexibility. The CHD code is also designed to be easily coupled to the reactor optimization code ADOPT (Qvist et al., 2016a).

The scope of this paper is to describe the structure of the CHD code as well as to compare its results in standard applications with those of other established fast reactor dynamics codes. Results including features specific to the CHD code, such as the modeling of ARC feedback systems, are presented in accompanying papers, e.g. (Qvist et al., 2016b). We focus here on the SFR design. However, the CHD code can also be configured for a Lead cooled Fast reactor (LFR), see e.g. (Hellesen et al., 2014) where a reactivity transient in a LFR was studied using an early version of the code. This paper is organized as follow. In Section 2 the code structure is described, in Section 3 the ESFR benchmark as well as the ABR simulation are described and in Section 4 the results are presented.

2. Code structure

The CHD code is built around an object oriented Python framework using the libraries numpy and scipy as well as matplotlib for plotting data and the IAPWS-IF97 (IAPWS, 2012) formulation for the thermodynamic data of water and steam. There are three main types of objects that define the reactor: flow objects, which model the coolant flow, solid structures and the core. In addition, there are objects controlling e.g. material definitions and reactivity effects. See Fig. 1 for a typical structure of a CHD simulation.

Flow objects are connected serially in loops. In a loop, the output temperature and mass flow of each object define the input boundary conditions of the proceeding object. The simplest flow objects are the **pools**, which are scalar, and their temperature is solved by

$$\frac{dT}{dt} = dm\frac{T_{in} - T}{V\rho}, \tag{1}$$

where T_{in} and dm are the outlet temperature and mass flow of the preceding flow object, V is the pool volume, and ρ is the density of the coolant. The **channel** objects are one dimensional, and the evolution of the axial temperature profile of the coolant channel is obtained from the diffusion-advection equation:

$$\rho C_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) - \rho C_p v_z \frac{\partial T}{\partial z} + \sum_i s_i \frac{T_i - T}{A_{flow}} + q, \tag{2}$$

where k, ρ and C_p, are the heat conductivity, density and specific heat capacity of the coolant, v_z is the coolant flow velocity along the channel. Heat transfer to and from adjacent structures, such as fuel rods, wrapper tubes and neutron shields, are modeled in the summation over i, where s_i is the heat conductance per axial unit length between the coolant and structure i, T_i is the surface temperature of the structure and A_{flow} is the cross sectional flow area of the coolant. The axial heat conductance is given by

$$\frac{1}{s_i} = \frac{1}{a \cdot h} + \sum_j \frac{1}{s_{ij}} \text{ and } h = \frac{Nu \cdot k}{d_h}, \tag{3}$$

where a is the surface area of the structure per axial unit length and h is the convective heat transfer coefficient. In addition, s_{ij} is the axial heat conductance of layers separating the structure and the coolant, such as the cladding and the air gap between the fuel and the cladding. Nu is the Nusselt number and d_h is the hydraulic diameter of the channel. At $z = 0$, Dirichlet boundary conditions based on the outlet temperature of the preceding flow object are used.

Further, q is the internal volumetric heat generation in the coolant. With solid fuel rods, internal heat generation in the coolant can e.g. be a result of gamma ray absorption. For fission reactions, the energy released by gamma radiation only amounts to a few percent, but for decay heat about 50% of the energy is released as gamma radiation. In a SFR the gamma absorption in the coolant is typically small because of the coolant's low density as well as the low coolant to fuel ratio. The gamma radiation will instead mainly be absorbed by the surrounding fuel rods. However, in a lead cooled reactor the coolant will absorb a larger fraction of the gamma radiation due to the high density of lead as well as a higher coolant to fuel ratio compared to a SFR. Heat exchangers are modeled as two counter flowing channels, each governed by Eq. (2) and connected to a common structure object that provides a convective heat transfer between the channels. The two channels are solved in a tightly coupled manner to avoid numerical instabilities.

The PDE in Eq. (2) is solved with a linear backwards finite difference scheme. The advection part uses upwind operators to avoid numerical instabilities. The code is therefore not compatible with backwards flow in the channels. The mass flow in the channel can either be set explicitly as a function of time, or it can be calculated from the pressure drop along the channel using friction factors (Chenu et al., 2011). This is useful when studying loss of flow scenarios that involve a transition to natural circulation flow (Wade and Fujita, 1989). The flow rate is then calculated from the pressure head provided by the pump, the thermal driving head from the temperature difference between the cold and the hot pools and the head loss over the fuel bundles. In that case the time-dependent pump pressure head must be given explicitly. By default, correlations for a triangular grid of fuel pins with a wire wrap are used to calculate Nu (Mikityuk, 2009) as well as the friction factor (Chenu et al., 2011). However, any set of correlations can be used by sub classing the channel object. The channel is connected with one temperature feedback.

The **rod** object is a structure that is typically connected to a channel and calculates the heat transfer within the fuel rod as well as the heat transfer through the gap between the pellet and the cladding and through the cladding. The heat conduction within the pellet is solved in a cylindrical geometry in either 1 or 2 dimensions (radially and axially), and the gap and cladding are treated as thin thermally conductive layers. Neumann boundary conditions