Annals of Nuclear Energy xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Annals of Nuclear Energy

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

Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications

Robert K. Salko [a,*], Rodney C. Schmidt [b], Maria N. Avramova [c]

[a] Oak Ridge National Laboratory, Oak Ridge, TN, United States
[b] Sandia National Laboratories, Albuquerque, NM 87185, United States
[c] The Pennsylvania State University, University Park, PA, United States

ARTICLE INFO

Article history:
Received 22 April 2014
Accepted 2 November 2014
Available online xxxx

Keywords:
Subchannel
COBRA-TF
Parallel
MPI
CTF

ABSTRACT

This paper describes major improvements to the computational infrastructure of the CTF subchannel code so that full-core, pincell-resolved (i.e., one computational subchannel per real bundle flow channel) simulations can now be performed in much shorter run-times, either in stand-alone mode or as part of coupled-code multi-physics calculations. These improvements support the goals of the Department Of Energy Consortium for Advanced Simulation of Light Water Reactors (CASL) Energy Innovation Hub to develop high fidelity multi-physics simulation tools for nuclear energy design and analysis.

A set of serial code optimizations—including fixing computational inefficiencies, optimizing the numerical approach, and making smarter data storage choices—are first described and shown to reduce both execution time and memory usage by about a factor of ten. Next, a "single program multiple data" parallelization strategy targeting distributed memory "multiple instruction multiple data" platforms utilizing domain decomposition is presented. In this approach, data communication between processors is accomplished by inserting standard Message-Passing Interface (MPI) calls at strategic points in the code. The domain decomposition approach implemented assigns one MPI process to each fuel assembly, with each domain being represented by its own CTF input file. The creation of CTF input files, both for serial and parallel runs, is also fully automated through use of a pressurized water reactor (PWR) pre-processor utility that uses a greatly simplified set of user input compared with the traditional CTF input.

To run CTF in parallel, two additional libraries are currently needed: MPI, for inter-processor message passing, and the Parallel Extensible Toolkit for Scientific Computation (PETSc), which is used to solve the global pressure matrix in parallel. Results presented include a set of testing and verification calculations and performance tests assessing parallel scaling characteristics up to a full-core, pincell-resolved model of a PWR core containing 193 17×17 assemblies under hot full-power conditions. This model, representative of Watts Bar Unit 1 and containing about 56,000 pins, was modeled with roughly 59,000 subchannels, leading to about 2.8 million thermal–hydraulic control volumes in total. Results demonstrate that CTF can now perform full-core analysis of a PWR (not previously possible owing to excessively long runtimes and memory requirements) on the order of 20 min. This new capability not only is useful to stand-alone CTF users, but also is being leveraged in support of coupled code multi-physics calculations being done in the CASL program.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

CTF is the shortened name given to the version of COBRA-TF being developed and improved by the Reactor Dynamics and Fuel Management Group (RDFMG) at the Pennsylvania State University (PSU) (Avramova, 2014; Salko and Avramova, 2014). CTF is a thermal–hydraulic simulation code designed for light water reactor (LWR) vessel analysis. It uses a two-fluid (hence the "-TF" designation), three-field modeling approach. The original COBRA-TF code was developed as a thermal–hydraulic rod-bundle analysis code in 1980 by Pacific Northwest National Laboratory under the sponsorship of the Nuclear Regulatory Commission (NRC) (Thurgood et al., 1980). It was subsequently implemented in the COBRA-TRAC code system (Thurgood et al., 1983) and further validated and refined as part of the FLECHT-SEASET 163-Rod

* Corresponding author at: Oak Ridge National Laboratory, Building 5700, Room I306-A, Oak Ridge, TN 37830, United States. Tel.: +1 865 576 5339.
E-mail addresses: salkork@ornl.gov (R.K. Salko), rcschmi@sandia.gov (R.C. Schmidt), mna109@psu.edu (M.N. Avramova).

http://dx.doi.org/10.1016/j.anucene.2014.11.005
0306-4549/© 2014 Elsevier Ltd. All rights reserved.

Please cite this article in press as: Salko, R.K., et al. Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.11.005

Blocked Bundle Test and analysis program (Paik et al., 1985). Over the past several decades, the COBRA series of codes has been used extensively throughout the nuclear industry, resulting in many variants of the code being created and validated (Avramova, 2007; Avramova and Cuervo, 2011; Avramova et al., 2011, 2009, 2007, 2003a,b; Cuervo et al., 2006; Frepoli et al., 2001; Ozdemir et al., 2011; Raja, 2011).

In the last decade, CTF has been extensively validated for both pressurized water reactor (PWR) and boiling water reactor (BWR) applications. Improvements have included translation from fixed- to free-format coding, use of features made available by modern Fortran standards, implementation of Krylov solver–based numerical techniques, enhanced input-error checking, the addition of a preprocessor to simplify input for PWR geometries, and the generation of code documentation (Avramova, 2014; Salko and Avramova, 2013, 2014).

A significant improvement is the PWR preprocessor, which allows users the option of a greatly simplified input for modeling PWR geometries. The preprocessor input consists of general assembly and core descriptions that are used by the preprocessor to generate the much more detailed input file required by CTF. The standard CTF input file can be burdensome for the user to create, particularly for large-scale models, because of the high level of detail it requires (e.g., rod-channel and channel-channel connections, spacer grid locations and loss coefficients, guide tube positioning, core geometry, control volume locations and sizes needed for visualization, effects of the baffle). The PWR preprocessor was also instrumental in the parallelization of CTF.

In addition to a simplified input, a new Visualization ToolKit (VTK) output option has been added to CTF that allows the visualization of many distributions with popular visualization tools such as VISIT (Lawrence Livermore National Laboratory, 2005) and ParaView (Kitware, 2011). The VTK output option also allows the creation of movies showing time-dependent distributions.

More recently, CTF has been collaboratively developed as part of the Consortium for Advanced Simulation of Light Water Reactors (CASL) program (see www.casl.gov). CASL is a Department of Energy (DOE) project, with joint participation from US national research laboratories, private industry, and research universities. Its mission is to provide leading-edge modeling and simulation capabilities to improve the performance of currently operating LWRs. Within CASL, CTF has become an important component of VERA, or "Virtual Environment for Reactor Applications." The goal of VERA development is to "*predict, with confidence, the performance of nuclear reactors through comprehensive, science-based modeling and simulation technology that is deployed and applied broadly throughout the nuclear energy industry to enhance safety, reliability, and economics.*" (www.casl.gov)

In support of CASL-VERA objectives, major improvements to the computational infrastructure of CTF have been completed. With these changes, full-core "pincell-resolved" simulations can now be performed in reasonable run-times, either in stand-alone mode or as part of coupled-code multi-physics calculations. The purpose of this paper is to describe these improvements and demonstrate current capabilities on an example problem.

Section 2 provides a high-level introduction and overview of current CTF models, numerics, and solution methodologies.
Section 3 has two parts. The first part covers serial code optimizations, and the second part describes the parallelization of CTF. This includes the domain decomposition approach implemented and the preprocessing utility created to simplify the task of input file creation.
Section 4 reviews important results, including a set of testing and verification calculations, performance tests assessing parallel scaling characteristics, and a demonstration calculation of a full-core, pincell-resolved model of Watts Bar Unit 1, which is an operating PWR being modeled with CASL tools.
Section 5 provides conclusions and discusses ongoing activities to leverage these new capabilities in support of coupled-code multi-physics calculations.

2. Synopsis of basic equations and numerical approach

2.1. Physics models and correlations

Thermal–hydraulics modeling in CTF is based on separating the conservation equations for mass, momentum, and energy into three distinct fields: vapor, continuous liquid, and entrained liquid droplets. This leads to three coupled sets of governing equations. In the equations that follow, all terms are defined per unit volume; the temporal and advection terms will appear on the left and the field source terms on the right; and the subscript k denotes the field.

Each continuity equation takes the following form:

$$\frac{\partial}{\partial t}(\alpha_k\rho_k) + \nabla\cdot(\alpha_k\rho_k\vec{V}_k) = L_k + M^T_{mass}, \tag{1}$$

where:
α_k = Volume fraction of phase k,
ρ_k = Density of phase k,
$\vec{V}_k$ = Velocity vector of phase k,
L_k = Mass transfer due to evaporation/condensation or entrainment/de-entrainment, and
M^T_{mass} = Mass transfer due to turbulent-mixing/void-drift.

Momentum equations are formulated for each field and each Cartesian coordinate direction (nine momentum equations). However, CTF has an option to use a simpler subchannel formulation that considers only two flow directions—axial flow and lateral flow, where the term lateral flow covers any orthogonal direction to the vertical axis. Lateral flow enters a subchannel volume through "gaps" between that volume and other adjacent subchannel volumes. Because fixed coordinates are not defined for the lateral direction in the subchannel approach, lateral flow has no direction once it leaves a gap. This is usually a good approximation for the axially dominated flow of a reactor fuel bundle, because the relatively minuscule lateral flows transfer little momentum across subchannel mesh cell elements. All calculations shown in this paper use the subchannel formulation.

The formulation of the axial momentum equation is shown in Eq. (2).

$$\frac{\partial}{\partial t}(\alpha_k\rho_k\vec{V}_k) + \nabla\cdot(\alpha_k\rho_k u_k\vec{V}_k) = \alpha_k\rho_k\vec{g} - \alpha_k\nabla P + \tau'''_w + \tau'''_i + M^L_{mom} + M^T_{mom}, \tag{2}$$

where u_k is the axial velocity of phase k. Source terms include the body force due to gravity ($\alpha_k\rho_k\vec{g}$, axial momentum equations only), pressure gradient ($\alpha_k\nabla P$), wall shear (τ'''_w), phase interface shear (τ'''_i), momentum source/sink due to phase change (M^L_{mom}) and momentum source/sink due to turbulent mixing and void drift (M^L_{mom}).

In CTF, the droplet and continuous liquid fields are assumed to be in thermodynamic equilibrium; thus there are only two energy equations. These take the following general form:

$$\frac{\partial}{\partial t}(\alpha_k\rho_k h_k) + \nabla\cdot(\alpha_k\rho_k u_k\vec{V}_k) = q'''_w + M^T_{en} + \alpha_k\frac{\partial P}{\partial t} \tag{3}$$

where h_k is the enthalpy of phase k, q'''_w represents energy transfer in/out due to turbulent exchange, M^T_{en} accounts for evaporation and condensation effects, and $\alpha_k\,\partial P/\partial t$ is the thermodynamic work term.

Please cite this article in press as: Salko, R.K., et al. Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.11.005

Eqs. (1)–(3) cannot be solved without appropriate closure models for the source terms. For example, models are needed for calculating wall shear, phase interface shear, wall heat transfer, phase interface heat transfer, lateral exchange due to turbulent mixing and void drift, mass transfer due to entrainment, and so forth. Details on these models used in CTF for these closure terms can be found in the *CTF Theory Manual* (Salko and Avramova, 2014).

In addition to the thermal–hydraulics of the coolant, CTF can optionally model conduction heat transfer in several types of solid structures that may surround the coolant channels. Eq. (4) represents the general form of the energy equation solved in each of these structures:

$$\frac{\partial}{\partial t}(\rho C_p T) + \nabla \cdot (-k\Delta T) = q''' \tag{4}$$

where ρ is the density of the conductor, C_p is the specific heat of the conductor, T is the temperature of the conductor, k is the thermal conductivity of the conductor, and q''' represents the local energy release rate from the conductor. In CTF, a user has the option of solving one (radial), two (radial-axial), or three (radial-axial-azimuthal) -dimensional versions of the heat conduction equations in fuel rods.

2.2. Numerics and solution methodology

CTF uses an Eulerian Control Volume (CV) – based approach to numerically approximate the thermal–hydraulic conservation Eqs. (1)–(3). Fig. 1 illustrates the irregular cross section of a subchannel centered control volume within a fuel bundle. In CTF each CV is characterized by its cross-sectional area, axial height, and the width and number of connections to adjacent CVs. These geometric characteristics are used within a staggered finite volume discretization scheme to define discrete approximations of the governing equations. In a staggered scheme, the velocities are obtained at mesh faces and the state variables (e.g., pressure, density, enthalpy, and phasic volume fractions) are obtained at the CV centers.

For large complex problems, manually specifying the geometric details of the CTF mesh in the input deck can be a tedious and error-prone effort. To simplify the input deck generation process, a specialized utility has been developed to build a CTF input deck from a reduced set of user input data for PWR-type reactor geometries ranging in size from a single assembly up to a full-sized core. Details on this utility can be found in the CTF *Preprocessor User Manual* (Salko and Avramova, 2013). In many ways, this utility is similar in function to commonly used meshing packages used by finite element and finite difference Computational Fluid Dynamics (CFD) codes, but in this case the descriptive data is embedded in a format consistent with the CTF-specific input file.

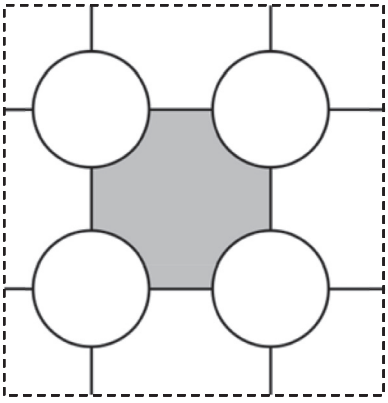

Fig. 1. Illustrative cross section of a subchannel centered control volume in CTF.

To solve the previously defined conservation equations, CTF uses an adaptation of the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) (Patankar, 1980). This approach involves a series of defined steps (see Fig. 2), iteratively repeated until a set of convergence criteria are satisfied. The most computationally expensive step requires the solution of an implicit "pressure correction" equation, which for large problems can be efficiently solved using an iterative Krylov method.

3. Optimization and parallelization

3.1. Serial code optimizations

CTF is a legacy Fortran code with a history that begins over 30 years ago. Before CTF was parallelized, fundamental aspects of the coding and memory usage were carefully scrutinized to identify and eliminate inefficiencies that might residually exist. This effort yielded a set of serial code optimizations that improved memory usage and data storage, removed computational inefficiencies, and enhanced the efficiency of the numerical approach. A common theme was that the inefficiencies identified were only apparent as the size of the problem being solved became large; the larger the problem was, the greater the relative speedup when the problem was fixed. Several of the most notable improvements are described here.

1. The CTF pressure correction equation is a large matrix of size N by N, with N being the total number of scalar cells in the mesh. CTF originally used a Gaussian elimination-based approach to

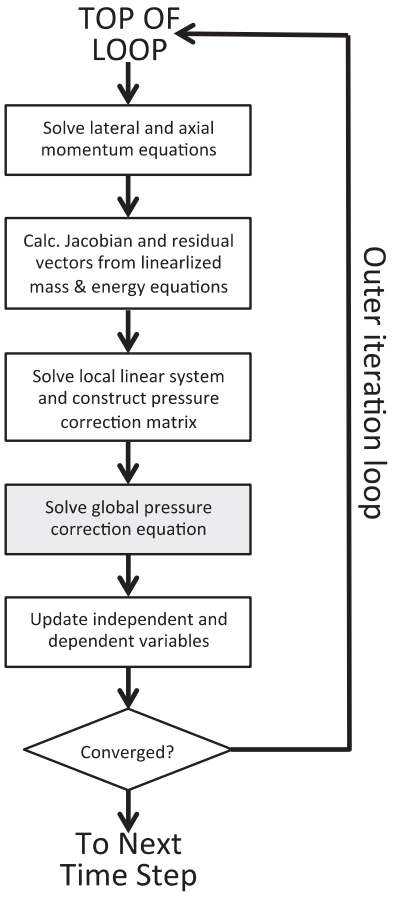

Fig. 2. CTF solves the coupled nonlinear equations using an adaptation of the SIMPLE approach.

Please cite this article in press as: Salko, R.K., et al. Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.11.005

directly solve the pressure correction equation. This solver required storage of the full N by N matrix even though most of the coefficients were zero. Although the original direct solver had been replaced by a more efficient Krylov solver, it was found that the pressure correction matrix was still being allocated as if direct elimination would be used for its solution. An enormous reduction in memory was realized for large, high-resolution models (e.g., pincell-resolved quarter-core) when only the non-zero components of this matrix were stored in a standard compressed sparse row format. Before this revision, memory consumption on a benchmark 7×7 assembly case (~460,000 CVs) was so high that CTF would not run on a cluster node with 96 GB of available memory. After the optimization, such a case consumes a total of 4 GB of memory.

2. Each time the fuel rod radial-direction heat transfer equation is solved, an array for mesh cell conductivities must be initialized. It was found that the length of this array, and its corresponding initialization loop, had been sized for all conductor mesh cells in the entire model instead of just those in the radial direction. For example, on a 7×7 assembly benchmark problem, an array with 14.5 million elements was constructed when an array of eight elements was only required. This array would be initialized more than 2.4 million times. Profiling showed that correcting this issue reduced the CPU time for the fuel rod conduction part of the solution from 4399 to 0.2 s.
3. Four global arrays are used in CTF for storing different interfacial heat transfer coefficients for each fluid cell in the mesh. Because of a bug related to array indices, these coefficients were being set to the entire array for each fluid cell, as opposed to being set to a single element in the array. Although this posed numerical problems only for the restart option (the arrays were only used outside the fluid-cell loop in order to write the restart deck), it introduced a significant inefficiency for large problems. For the 7×7 assembly benchmark problem mentioned above, correcting this problem reduced the execution time in the main driver routine from ~1436 to ~31 s.
4. CTF is a Fortran code with many multidimensional arrays. In Fortran, arrays are stored using column-major order. This makes it optimal to nest loops so that the first index of a multidimensional array is varied by the innermost loop. In practice, this means that data that is currently being used in the loop is more likely to be in the processor cache memory as opposed to the slower random access memory. It was discovered that a large, frequently used array, used for storing pressure correction equation coefficients, was nested in the "wrong" order for a Fortran program. Switching array indices resulted in a 24% reduction in runtime for the 7×7 assembly benchmark problem.
5. For calculating superheated vapor temperatures, CTF uses correlations presented by Wagner and Kruse (1998) in the Industrial Standard, IAPWS-IF97. These equations consist of the summation of a large number of terms and take the following form:

$$T = \sum_{i=1}^{imax} n(i) P_{norm}^{l(i)} (h_{norm} - 2.1)^{j(i)} \quad (5)$$

where i_{max} is a constant integer ranging from 23 to 38, depending on the equation. These correlations were implemented inside a loop, with lookup tables being referenced to obtain the integer values for the n, l, and j coefficients for each of the "i" terms being summed. The other terms, P_{norm} and h_{norm} represent the pressure and enthalpy of the fluid. Since raising a value to an integer power is more computationally costly than performing a similar multiplication (e.g., 2^3 takes longer than 2·2·2), a compiler will typically expand such operations into multiplication form when generating assembly code. However, because these coefficients were inside a lookup table, it was not possible for the compiler to do this optimization. This problem was remedied by fully expanding all of the superheated vapor temperature equations so that coefficients could be directly embedded within them. Since these correlations are used 12 times per mesh cell per fluid iteration, this change has resulted in approximately a 3.5 times speedup in code execution time for the benchmark 7×7 case.

1. An option to set an initial flow rate in the mesh was added. Originally, CTF initialized the mass flow rate in the momentum cell mesh to zero, even when the mass flow rate given at the inlet was non-zero. This artificially created steep gradients that can take significant time to remove during the numerical solution. Previously, an option to use a boundary condition ramp had been implemented. But tests showed that a better option was to simply change the momentum cell initialization from zero to the mass flow rate given at the inlet. This new option was added to CTF and execution times compared on a test problem. On average, this option decreased run times by ~30% compared with using the boundary condition ramp.

As a result of the aforementioned improvements and removal of source code inefficiencies, significant improvements in memory usage and simulation wall time were obtained. Table 1 presents the improvements observed for four models of increasing size. Because of excessive memory requirements and wall-time limits, original wall-time data was not available for the larger 4×4 and 7×7 assembly configurations.

3.2. CTF parallelization

We have chosen to parallelize CTF using a relatively standard Single Program Multiple Data (SPMD) strategy targeting distributed memory "multiple instruction multiple data" (MIMD) platforms and using domain decomposition. MIMD platforms are machines that have a number of processors that can operate independently on different data. The SPMD approach is a parallelization strategy whereby a single program is split into groups of tasks, each of which is performed by a separate processor that take its own input.

This choice was influenced by several important characteristics: (1) this approach required no change to the SIMPLE-based solution strategy, (2) coding changes/additions would not be strongly invasive—thus most of the coding in CTF has remained essentially unchanged, and (3) success could be tested/demonstrated by showing identical results with serial runs. In addition, it was expected that this parallelization strategy would lend itself well to running parallel CTF on cluster computers using hundreds of processors. Since a typical PWR has approximately 200 assemblies, and a single assembly steady state problem with serial CTF now takes only on the order of minutes to solve, a full-core pincell-resolved analysis running CTF in parallel should also be possible on the order of minutes.

3.2.1. Domain decomposition

The first step to performing a parallel solution using our approach is to divide the single model into multiple domains. It

Table 1
Improvements in CTF simulation wall-time for models of increasing size after implementing serial optimizations.

Assembly configuration	Number of mesh cells	Original wall time (min)	Post-improvement wall time (min)
1×1	9396	9.203	1.296
2×2	37,584	90.87	9.160
4×4	150,336	–	65.80
7×7	460,404	–	189.1

Please cite this article in press as: Salko, R.K., et al. Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.11.005

was necessary to develop a decomposition algorithm to do this model division in an automatic way. A natural way to decompose a core model is to break it up into a collection of single assembly models. This clearly defines the rods owned by each domain, requires a reasonable number of cores for a full-core problem (~200), and should lead to suitable wall times for a full-core solve (on the order of 10 min).

The most logical place to implement this decomposition algorithm was in the CTF Preprocessor Utility mentioned earlier. Using and modifying this utility software, rather than CTF itself, to perform the domain decomposition made sense for two reasons:

1. The preprocessor has knowledge of what rods belong in a given assembly—CTF has no concept of assemblies.
2. The preprocessor has an explicit knowledge of the location of rods, gaps, and channels and how they connect.

Fig. 3 illustrates how the channels and rods in a simple 3×3 rod, five-assembly test problem (small-cross geometry) is decomposed into five domains by the CTF preprocessor. Note that although each domain/processor is assigned the same number of rods, the domains cannot have the same number of channels because of geometric considerations. Gaps are allocated in a similar fashion.

Modifications to the CTF preprocessor to enable parallel domain decomposition are essentially transparent to a user. Its input files remain unchanged except for one optional flag. If the user activates this flag, the preprocessor will generate a set of parallel simulation files (one for each assembly) rather than a single standard CTF input file.

The structure of a parallel CTF input deck is nearly identical to that of a serial input deck, with the exception of three pieces of information that are written to three new card groups. These card groups give necessary information pertaining to the channels, gaps, and rods, respectively. Specifically, the card groups tell CTF:

- The index of the entity in the domain (local index).
- The index of the entity in the entire system (global index).
- The domain that owns the entity.
- The other domains that require data from the solved domain.

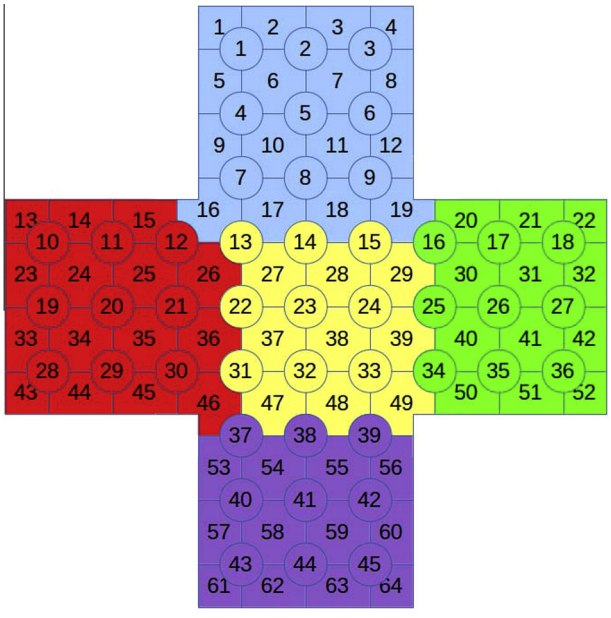

Fig. 3. Domain ownership of channels and rods for a five-assembly test problem.

The concept of "ghost" entities (i.e., channels, gaps, or rods) is important to the parallelization strategy used in CTF. To explain this, consider a channel as an example of the entity of interest. The solution response in this channel is dependent on the solution response in its connected channels (e.g., change the pressure in one channel and it will have an impact on connected channels). However, when we decompose the model, and if this channel is on a boundary with its once-neighboring channel, now removed, it will lack knowledge of its neighbor. To address this issue, we create a "ghost" channel as a substitute for this boundary channel's neighbor. The ghost channel is flagged as "owned" by another domain (i.e., processor) and acts like a container for data. It exists in the domain for the sole purpose of holding data that the boundary channel can pull from as needed. For this to work, however, it is necessary that the data in the ghost entity be updated at appropriate times throughout the simulation using MPI. This is the purpose of including the new card groups in the parallel simulation files—they provide CTF with the information needed to successfully pass data from entities owned by one domain to ghost entities in other domains.

Because CTF uses a SIMPLE-based solution algorithm, most of the solution steps can be performed locally and independently on each processor, with appropriately timed updates to the information in ghost entities. All of the information needed to perform these steps is included in the standard CTF input file (which now includes the three new card groups mentioned). However, one key step, the implicit solution of the pressure correction equations, must be solved on a global basis in a simultaneous fashion. Therefore, we must also know where to put each local equation's contribution into a global pressure correction matrix. But this information is not contained in the regular input file. An additional new master input file is generated for parallel runs that contains global information, including total number of channels, gaps, and rods in the model. This file is read by each processor in the simulation and provides data about the channels, gaps, and rods in the overall global system. Using this information, each process can assign a global index for every cell in its local domain, which then allows for proper placement of the pressure correction equations into the global pressure matrix.

3.2.2. Code modifications

To solve a CTF problem in parallel, data must be passed between processors during the simulation. The algorithmic locations of the required data transfers are indicated by stars in a flow chart of CTF's solution algorithm in Fig. 4, and are number 1–6. The information transferred is described next.

If the problem is running in parallel, each CTF process begins by reading its own unique domain input file (each of which will include the three new cards discussed above). Next, each process will do the usual setup: indexing the entities in the model, determining how things connect, and so on. All of this is done local to the domain using the existing variable and array names in CTF.

The communication that happens just before the system mesh setup (data transfer 1) has to do with the root process gathering system mesh information. Each domain sends the root process the number of entities in its domain, as well as their global identifications (IDs) and their space locations. When this is complete, and before starting the transient solution, every process reads the master deck and assigns each cell in its domain a global ID in addition to its local ID. This is the step shown in the second box. After this, the transient simulation starts.

The global time step loop is contained within the green block. The solution at each time step is found using a SIMPLE-based solution algorithm (outer iteration). An iteration starts with the setup and solution of the momentum equations. The orange vertical bar represents an axial node loop. We start at the first axial layer

Please cite this article in press as: Salko, R.K., et al. Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.11.005

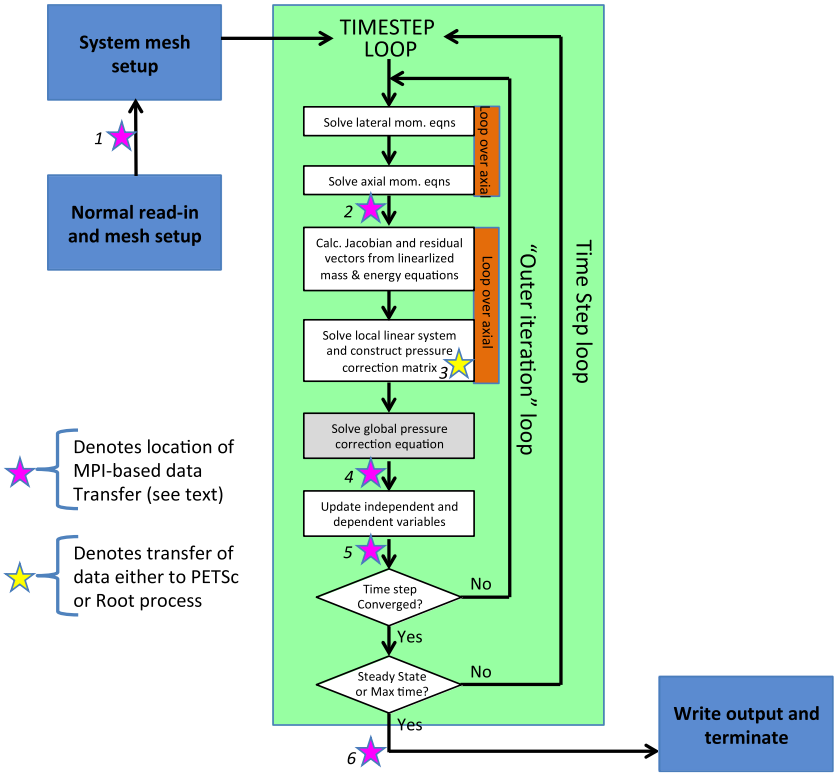

Fig. 4. Flowchart of the CTF solution algorithm and required data communication points.

in the model and March up to the top of the model. At each layer of scalar cells, we loop over all gaps (lateral flow paths between adjacent, connected channels). In each gap, we set up and solve the lateral momentum equations, of which there are three (one for liquid, one for vapor, and one for droplets). The three equations are solved simultaneously using Gaussian elimination for a single gap, yielding an updated lateral mass flow rate for each of the three fields. With this done, the axial momentum equations are then set up and solved for the current axial level. This is done for each channel in the model, rather than the gaps. This process is repeated for every axial level in the model.

After the momentum equations are finished, another parallel data transfer is required (data transfer 2). The data transfer is done using two generic routines—one routine transfers data for gaps and the other transfers data for channels. Each transfer routine loops over the owned entities that have ghosts in other domains and sends their data off to the buffer. After all sends are complete, the routines then loop over all ghosts in the domain and collect their data that is waiting in the buffer. This process completes the data transfer for all mass flow rates in the model. Subsequent transfers take place in the same manner.

When data transfer is finished, the setup of the continuity and energy equations commences. This is also inside a loop over the axial levels in the model and then the channels in the model; however, the equations are not solved right away. In keeping with the SIMPLE algorithm, the equations are used to build a system of pressure correction equations. For each scalar cell in the mesh, a Jacobian and residuals vector is constructed from the mass and energy governing equations. This linear system is reduced using forward elimination, and the resulting equation at the bottom of the reduced matrix is put into a pressure matrix. The pressure matrix is built piece-by-piece by taking one equation from each cell in the mesh—it is not complete and ready for solution until the loop over the mass/energy block is complete.

The yellow star in this block represents the building of the pressure matrix—it can currently be done in one of two ways. As previously mentioned, the pressure correction equations must be solved simultaneously, and that can be done either on one processor or on many. The end goal is to do the pressure matrix solution on all the processors in the MPI communicator, because this will lead to the fastest possible solution; however, there is value in also being able to do the pressure matrix solution on a single core. The reason for this is the nondeterministic process by which the pressure matrix will be solved when it is solved on many processors. In other words, the order of the operations in the pressure matrix solution will change in a parallel solution compared with its serial counterpart. This, because of unavoidable rounding errors in computer floating-point arithmetic, will lead to a different solution than if the pressure matrix were solved on a single processor.

Although this solution variation is small, it prevents our using a serial solution to validate our parallel algorithm. The validation of the algorithm is paramount, so for this reason, we also added an option to reduce the entire pressure matrix to the root processor so that the root processor alone could perform its solution. If this is done, the simulation path and result should be identical between a serial and parallel run. Therefore, depending on the user choice, the yellow star (data transfer 3) represents either

- Sending the pressure equation to the Parallel Extensible Toolkit for Scientific Computation (PETSc), which takes care of building the global, parallel pressure matrix, or
- Sending the pressure equation to the root process (if a non-root process) or receiving the pressure equations and building the global pressure matrix (if the root process).

When the mass/energy loop is complete, the pressure matrix is solved—it will be solved using either PETSc or the pre-existing Krylov solvers available in CTF. Upon completion of the solve

Please cite this article in press as: Salko, R.K., et al. Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.11.005

Table 2
Summary of CTF output files for serial and parallel runs.

Per domain output	Single system output
Simulation dump file	Mass balance file
Channels data file	Heat balance file
Gaps data file	Timestep information file
DNB-related data file	VTK file

operation, the solution is distributed to all the domains (data transfer 4), updating the pressure correction in all owned and ghost cells. After the pressure corrections are calculated, they can be back-substituted into the reduced Jacobian/residual arrays for the scalar cells to retrieve the updated void and enthalpy for the flow fields. The pressure corrections also lead to a new pressure field which, in turn, affects the velocity field. The mass flow rates are updated as well to reflect the new pressure field. After this step is complete, a communication is completed to get all updated variables from the owned entities to their ghosts.

In addition to the updating of the dependent variables, some important iteration controls are set based on the rate of change of dependent variables, as well as the maximum velocity in the model (Courant limit). These iteration controls are reduced on all processes (as part of data transfer 4), so all domains stay consistent with one another while marching through time.

The final step in the outer iteration loop is to check for convergence. This step requires that the convergence status—mass balance, mass storage, energy balance, and energy storage—be compared against specified criteria. Because these terms are global to the system, they are reduced to the root process (data transfer 5), which alone judges convergence and then distributes the verdict to the other domains. In this way, all processes exit the transient loop at the same point.

If convergence is judged to occur, a final communication (data transfer 6) reduces the required simulation data from the non-root processes to the root process. The root process stores this data in system arrays and writes it to the VTK file, which is common to the entire system.

3.2.3. Output files from parallel runs

CTF writes a collection of files during and after a simulation. For a serial run, all files represent results for the entire model. For a parallel run, on the other hand, it is advantageous to write some files on a per-domain basis and others on a per-model basis. A per-domain basis means that the file includes results only for the entities that exist in the domain from which the output file was written. A summary of the CTF files for both serial and parallel runs is given in Table 2. The first group of files is written on a per-domain basis, which accounts for the new naming convention. In the second group, one file is written for the model.

4. Parallel CTF calculations

This section reviews how parallel CTF was tested for correctness and performance on two sets of test problems. Selected results are also presented from a demonstration calculation of a full-core, subchannel-resolved model of Watts Bar Unit 1 (193 17 × 17 assemblies, ~56,000 pins, ~59,000 subchannels, and ~2.8 million TH control volumes) under hot full-power conditions (Kochunas et al., 2014).

4.1. Testing for correctness

Six test problems were constructed to verify that results remain identical between serial and parallel runs of the same model when the same pressure matrix solver is used. These are special-case tests, because CTF will normally use a different pressure matrix solver when running in parallel (which will lead to small differences in the solution). Successfully passing these tests demonstrates that the CTF solution algorithm was not inadvertently changed when the code was modified for running in parallel.

Fig. 5 illustrates the three assembly configurations considered. There are two models of each of the three assembly configurations—a small 3 × 3 rod bundle and a larger 17 × 17 rod bundle. The five-assembly model was included to better capture the types of geometry that exist in full-core models.

For each of these six cases, two models were made; a model for a serial run of CTF and a model for a parallel run of CTF. To ensure that both models start with identical boundary conditions, an option was added to allow a user to specify mass flux as the flow boundary condition, rather than a mass flow rate. This eliminated floating-point arithmetic rounding error as a source of difference between simulation results. Additionally, the pressure matrix was solved by reducing the matrix to root and solving using the internal Krylov solver.

To compare results between the two codes, the following quantities were written out to a text file and compared using a standard Unix "diff" utility.

1. Pressure matrix coefficient array.
2. Pressure matrix right-hand side vector.
3. Pressure correction solution vector.
4. Liquid, vapor, and droplet axial mass flow rates.
5. Liquid, vapor, and droplet lateral mass flow rates.
6. System pressure.
7. Liquid and vapor enthalpies.
8. Non-condensable gas pressure, density, and enthalpy.
9. Vapor generation rate.
10. Liquid, vapor, and droplet volume fractions.
11. Heat input to the liquid and vapor phases.
12. Heat generation in the fuel rods.
13. Temperatures throughout the solid conductors.

A diff was also used on the simulation heat balance file, the simulation mass balance file, and the simulation transient time step data file. Results verified that the files were identical for serial and parallel runs of the same model for all six of the test cases.

4.2. Functionality and performance testing

A second set of tests for parallel CTF were set up based on a full-core, pincell-resolved model of Watts Bar Unit 1 PWR. We started with the full-core model (193 assemblies) and then broke it down into smaller and smaller pieces, leading ultimately to a single 17 × 17 assembly. This allowed us to test the scaling behavior of Parallel-CTF for successively larger problems, specifically: 1, 2, 4, 9, 16, 25, 49, and 193 assemblies. The core count was increased equally with problem size (e.g., 2 cores used for 2 assembly model,

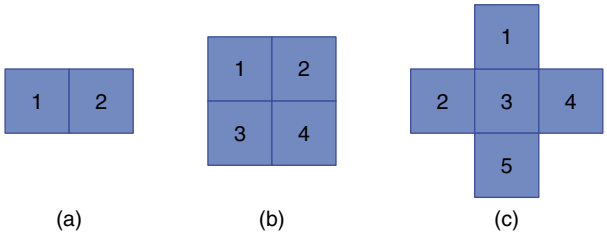

Fig. 5. Assembly configurations for models used in the test suite for validating the correctness of the parallelization of CTF.

4 cores used for 4 assembly model, etc.). These performance tests, therefore, constitute a "weak scaling" study.

Results were obtained on the Oak Ridge Leadership Computing Facility (OLCF) Titan Cluster (AMD Opteron(tm) processor 6140 2.6 GHz) at the Oak Ridge National Laboratory (ORNL) for these eight models of increasing size. These are presented in Table 3 and Fig. 6. Fig. 6 additionally presents cumulative wall time spent in different sections of the code.

In addition to the total simulation time, shown by the blue dashed line, total cumulative times spent in specific sections of the code are shown. In this figure, a line that is nearly horizontal demonstrates good scaling, as time spent in that section of the code is not increasing with problem size. Looking at the total simulation time, we see that it increases with problem size. To understand why this is the case, we timed individual pieces of the code and determined where the poor scaling was occurring.

The three parts of the code we timed were

(1) The setup and solve of the solid conductor equation, setup and solve of the axial and lateral momentum equations, the setup of the continuity and energy equations, and the back-substitution step required after the pressure matrix is solved (blue solid line).
(2) The solution of the global pressure matrix performed by PETSc (green solid line).
(3) The writing of the global VTK file (black solid line).

The dashed red line is a sum of the time spent in each of the four sections timed—since this is nearly equal to the total simulation time, it demonstrates that all relevant code sections have been captured in this study.

From Fig. 6, we see poor scaling of the blue and green solid lines for the first few jumps in problem size, but this is an expected side-effect of how we carry out the parallel solution. In moving from a single assembly to two assemblies, we are also increasing the problem size in each domain by adding layers of ghost channels and rods to one of the boundaries of each domain. As we increase the problem size further, we have to add ghost layers on multiple boundaries of each domain. We do not calculate the dependent variables in these ghost entities, but we still calculate other terms, which accounts for the jump in simulation time. After this initial cost, attributable to the increase in problem size in each domain, the scaling of these two code sections is very close to horizontal.

The line representing the VTK write, however, shows that this activity, as expected, does not scale well in the code because it is done in serial. Thus the time spent writing the VTK file increases linearly with problem size. It is evident from Fig. 6 that this back-end task is the primary reason for the imperfect total simulation time scaling. This poor scaling can be addressed by changing the code so that instead of using a single processor to write results to a single legacy VTK file, it writes out in parallel to a parallel data storage format.

Despite the high cost of writing out the VTK file in serial, for the current use-requirements of CTF in the CASL program, the results of this parallelization are considered excellent. As a reference, the time to solve the full-core Watts Bar, Unit 1 simulation problem on the ORNL Natasha cluster (AMD Opteron(TM) processor 62,72 2.1 GHz) was reduced from about 25 h, for a serial solve, to less than 19 min when it was done in parallel, an impressive 81 × speedup. Furthermore, as was discussed in Section 1, CTF is being used as a component in a multi-physics core simulator (Kochunas et al., 2014). In this context, CTF will be called upon many times during the iterative solution of a coupled problem, but it will be instructed to write the VTK file only once (at the end). In time, it is envisioned that a common output will be used for this core simulator and that data will be obtained from CTF using a data file format that will both support parallel output and be written in binary format. This should make the data output step a relatively small time requirement and lead to near optimum scaling for problems that consist of four or more assemblies.

Table 3
CTF wall-time for simulations of increasing size.

Number of assemblies	Serial wall-time (min)	Parallel wall-time (min)
1	6.76	6.76
2	15.20	6.97
4	29.37	7.53
9	78.07	8.93
16	152.58	9.33
25	257.99	9.89
49	510.54	11.29
193	1473.03	23.06

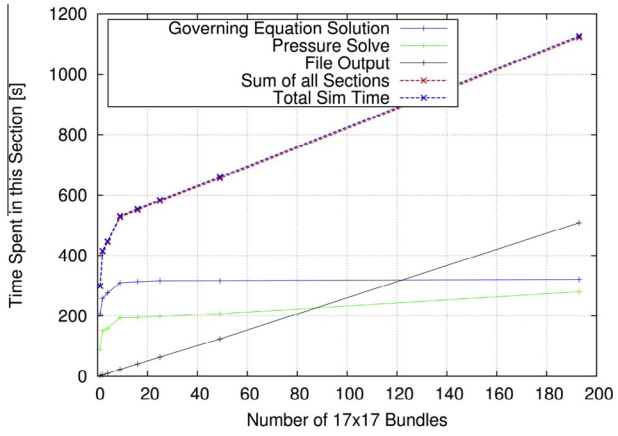

Fig. 6. Weak scaling and profile of parallel CTF obtained on the OLCF Titan Cluster. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5. Concluding remarks and future direction

The work reported in this paper has been largely motivated by the need in the CASL program for an efficient thermal–hydraulic, subchannel-resolved analysis component in the VERA core simulator (VERA-CS) for coupling with advanced neutronics and fuel-performance packages. Because VERA-CS is intended for full-core multi-physics simulations, it was important that CTF be as efficient and fast-running as possible.

This paper summarizes our work to significantly improve the performance of the CTF code. This consisted of two major parts: (1) improving source code efficiency to greatly reduce serial wall-times and memory usage and (2) parallelizing the code so that large problems could be efficiently solved using multiple cores.

The serial optimizations have resulted in approximately 10× improvements in simulation wall time for a model of a single 17 × 17 assembly under normal PWR operating conditions. Additionally, memory requirements for a quarter-core model have dropped from over 100 GB to under 4 GB, allowing for large-scale simulations to be performed on a desktop PC.

To parallelize CTF, we employed a SPMD strategy that targets distributed memory MIMD platforms. This required that we develop a domain decomposition algorithm, which we implemented into the CTF Preprocessor Utility. The algorithm is designed to make each assembly a domain in the model, meaning that conducting a parallel simulation requires having a processor

for each assembly in the model (e.g., a typical 177 assembly core would require 177 processors for the parallel simulation). Using a reduced set of model characteristics, the Preprocessor Utility will generate an individual input deck for each domain (assembly) in the core model, along with a master deck that specifies global core information needed by each domain. CTF reads all of these decks in when executed in parallel and produces output data that is tailored to the user's choices.

The correctness of the parallelization was verified by solving six different test problems both in serial and in parallel, and ensuring that results were identical to machine accuracy when using the same pressure solver. Parallel performance was evaluated by doing a weak scaling study for a set of problems ranging in size from a single 17×17 assembly up to a full core with 193 assemblies. Results showed that, with one exception, the parallelized code scaled very well. This one exception was the work required at the end of a simulation to write data to a legacy VTK file, which does not support parallel data output. Plans are to replace this inefficient data storage format with one that supports parallel output, which will enable more optimal scaling.

This work has resulted in the capability to do high-resolution, full-core simulations of PWR cores in what we consider reasonable wall-times ($\sim$20 min to reach steady-state convergence for normal operating conditions). Previously, it was not even possible to solve problems of this size and resolution with CTF. A user needing to do a full-core analysis was limited to coarse-core modeling, in which entire assemblies were modeled as subchannels and potentially important physical behavior was smeared and lost. This capability benefits stand-alone CTF users as well as users of the CASL project VERA-CS, which uses CTF as the thermal–hydraulic subchannel tool in performing coupled multi-physics simulations (Kochunas et al., 2014).

Acknowledgements

This research was supported by the Consortium for Advanced Simulation of Light Water Reactors (www.casl.gov), an Energy Innovation Hub (http://www.energy.gov/hubs) for Modeling and Simulation of Nuclear Reactors under US Department of Energy Contract No. DE-AC05-00OR22725.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC05-00OR22725.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the US Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

Avramova, M., 2007. Development of an Innovative Spacer Grid Model Utilizing Computational Fluid Dynamics within a Subchannel Analysis Tool (Ph.D. thesis), The Pennsylvania State University.

Avramova, M., 2014. COBRA-TF Input Manual. The Pennsylvania State University.

Avramova, M., Cuervo, D., 2011. Assessment of CTF boiling transition and critical heat flux modeling capabilities using the OECD/NRC BFBT and PSBT benchmarks database. In Proceedings of the 14th International Topical Meeting on Nuclear Reactor Thermal Hydraulics, Toronto, Ontario, Canada.

Avramova, M. et al., 2003a. PWR MSLB simulations using COBRA-TF advanced thermal–hydraulic code. In Transactions from 2003 ANS Winter Meeting, New Orleans.

Avramova, M. et al., 2003b. COBRA-TF PWR core wide and hot subchannel calculations. In Transactions from 2003 ANS Winter Meeting, New Orleans.

Avramova, M. et al., 2007. Analysis of steady state and transient void distribution predictions for Phase I of the OECD/NRC BFBT benchmark using CTF/NEM. In Proceedings of the 12th International Topical Meeting on Nuclear Reactor Thermal Hydraulics, Pittsburgh, Pennsylvania.

Avramova, M. et al., 2009. Uncertainty analysis of COBRA-TF void distribution predictions for the OECD/NRC BFBT benchmark. In Proceedings of the 2009 International Conference on Advances in Mathematics, Computational Methods and Reactor Physics, Saratoga Springs, New York.

Avramova, M. et al., 2011. Comparative analysis of CTF and TRACE thermal–hydraulic codes using OECD/NRC PSBT benchmark void distribution database. In Proceedings of the 14th International Topical Meeting on Nuclear Reactor Thermal Hydraulics, Toronto, Ontario, Canada.

Cuervo, D. et al., 2006. Evaluation and enhancement of COBRA-TF efficiency for LWR calculations. Ann. Nucl. Energy 33, 837–847.

Frepoli, C. et al., 2001. COBRA-TF simulation of BWR bundle dry out experiment. In Proceedings of the 9th International Conference on Nuclear Engineering, Nice, France.

Kitware, Inc. ParaView User's Guide, v 3.10. April 2011.

Kochunas, B. et al., 2014. Demonstration of neutronics coupled to thermal–hydraulics for a full-core problem using COBRA-TF/MPACT, CASL Technical Report: CASL-U-2014-0051-000, April 2014.

Lawrence Livermore National Laboratory. VisIt User's Manual, v. 1.5. October 2005.

Ozdemir, E. et al., 2011. Multidimensional boron transport modeling in subchannel approach. In Proceedings of the 14th International Topical Meeting on Nuclear Reactor Thermal Hydraulics, Toronto, Ontario, Canada.

Paik, C. et al., 1985. Analysis of FLECHT-SEASET 163-rod blocked bundle data using COBRA-TF. Technical Report NUREG/CR-3046, United States Nuclear Regulatory Commission.

Patankar, S., 1980. Numerical Heat Transfer and Fluid Flow. McGraw-Hill.

Raja, F., 2011. Implementation of point kinetics model and numerical improvements in CTF (Master's thesis), The Pennsylvania State University.

Salko, R., Avramova, M., 2013. CTF Preprocessor User's Manual.

Salko, R., Avramova, M., 2014. CTF Theory Manual. The Pennsylvania State University.

Thurgood, M. et al., 1980. COBRA-TF development. In 8th Water Reactor Safety Information Meeting, Gaithersburg, MD.

Thurgood, M. et al., 1983. COBRA/TRAC—a thermal–hydraulics code for transient analysis of nuclear reactor vessels and primary coolant systems. Technical Report NUREG/CR-3046, Pacific Northwest National Laboratory.

Wagner, W., Kruse, A., 1998. Properties of Water and Steam. Springer, Number IAPWS-IF97.

www.casl.gov (accessed 09.02.14).

Please cite this article in press as: Salko, R.K., et al. Optimization and parallelization of the thermal–hydraulic subchannel code CTF for high-fidelity multi-physics applications. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.11.005