U.S. DEPARTMENT OF ENERGY | Nuclear Energy

CASL-U-2019-1889-000

Consortium for Advanced Simulation of LWRs

CTFFuel User's Manual

A. Toptan[1], R. Salko[2], J. Hu[2], A. Wysocki[2], and M. Avramova[1]

[1]North Carolina State University
[2]Oak Ridge National Laboratory

8/1/2019

Revision Log

Revision	Date	Affected Pages	Revision Description
0	8/1/2019	All	Initial Release

Document pages that are:

Export Controlled:	None
IP/Proprietary/NDA Controlled:	None
Sensitive Controlled:	None
Unlimited:	All

The North Carolina State University

Department of Nuclear Engineering

Reactor Dynamics and Fuel Modeling Group

NC STATE UNIVERSITY

CTFFuel User's Manual

March 4, 2023

Executive Summary

The fuel heat transfer capability within CTF has been extended and enabled to be used as an independent fuel solver code, CTFFuel to simulate steady-state and transient thermo-mechanical responses of **lwr!** (**lwr!**) fuel rods. CTF is a modernized version of the **cobratf!** (**cobratf!**) subchannel thermal-hydraulic code that is jointly developed by **ncsu!** (**ncsu!**) and Oak Ridge National Laboratory for the **doe!** (**doe!**)-funded **casl!** (**casl!**). The code was originally developed by **pnnl!** (**pnnl!**) in 1980 and had been used and modified by several institutions over the last few decades. The CTFFuel interfaces the CTF's fuel rod modeling capabilities with predefined wall boundary conditions to eliminate fluid domain calculations. By separating fluid domain computations from fuel thermo-mechanics, through coupling at the cladding-coolant interface, this code acts as an interface to CTF's fuel modeling and can be executed with an independent input as a stand-alone capability. The code's use outside of CTF provides several benefits, including ease of code-to-code benchmark and calibration with fuel performance codes, ease of sensitivity/uncertainty analysis of the fuel thermo-mechanical capabilities by eliminating any uncertainties that might propagate from CTF's thermal-hydraulics calculations, and ease of coupling to neutronics and surface chemistry codes in multi-physics calculations. The purpose of this manual is to cover description how to run CTFFuel and to process its output data.

Contents

List of Figures

Acronyms

CASL Consortium for Advanced Simulation of Light Water Reactors

COBRA-TF Coolant-Boiling in Rod Arrays—Two Fluids

DOE Department of Energy

LWR light water reactor

NCSU North Carolina State University

PNNL Pacific Northwest National Laboratory

1. Introduction

The fuel heat transfer capability within CTF has been extended and enabled to be used as an independent fuel solver code, CTFFuel to simulate steady-state and transient thermo-mechanical responses of **lwr!** fuel rods. CTF is a modernized version of the **cobratf!** subchannel thermal-hydraulic code that is jointly developed by **ncsu!** and Oak Ridge National Laboratory for the **doe!**-funded **casl!**. The code was originally developed by **pnnl!** in 1980 and had been used and modified by several institutions over the last few decades. The CTFFuel interfaces the CTF's fuel rod modeling capabilities with predefined wall boundary conditions to eliminate fluid domain calculations. By separating fluid domain computations from fuel thermo-mechanics, through coupling at the cladding-coolant interface, this code acts as an interface to CTF's fuel modeling and can be executed with an independent input as a stand-alone capability. The code's use outside of CTF provides several benefits, including ease of code-to-code benchmark and calibration with fuel performance codes, ease of sensitivity/uncertainty analysis of the fuel thermo-mechanical capabilities by eliminating any uncertainties that might propagate from CTF's thermal-hydraulics calculations, and ease of coupling to neutronics and surface chemistry codes in multi-physics calculations. The purpose of this manual is to cover description how to run CTFFuel and to process its output data. The **lwr!** material properties are documented in CTF's Property Manual [**ctfProperty**].

2. CTFFuel Input Specifications

This chapter provides information how to use the `CTF`'s standalone fuel solver, `CTFFuel`. `CTFFuel` uses the same fuel models as CTF, merley acting as a separate, standalone interface to the CTF fuel solvers. The code was setup to generate a solution for a single fuel rod with no fluid-side solution. This document describes the structure of the input file and provides sample input decks.

2.1 Input Parameters

`CTFFuel` is executed with an independent input as a stand-alone capability. To illustrate some of geometry parameters, a nodalization scheme of fuel rod is provided in Figure **??**. The input is constructed based on several blocks which are:

- `[CONTROL]` to define system and boundary conditions (see Table **??**).
- `[GEOM]` to define rod dimensions and initial gap condition (see Table **??**).
- Time-dependent state parameters (see Table **??**):
  - `[HTCV]` to define vapor heat transfer coefficient as a boundary condition,
  - `[HTCL]` to define liquid heat transfer coefficient as a boundary condition,
  - `[TV]` to define vapor temperature as a boundary condition,
  - `[TL]` to define liquid temperature as a boundary condition,
  - `[PRESSURE]` to define the mesh-dependent pressure vector in the channel,
  - `[Tw]` to define a fixed wall temperature,
  - `[RODQ]` to define axial power distribution,
  - `[RADP]` to define radial power distribution,
  - `[EXPOSURE]` to define mesh-dependent fuel burnup,
  - `[GAD]` to define mesh-dependent gadolinia content.

Three classes of problems can be modeled including:

[1] Steady state solution

[2] Transient solution

[3] Depletion solution

The input structure changes slightly when modeling a depletion versus steady state or transient. The "depletion" card in the [CONTROL] block specifies which format is to be read. If modeling a depletion, the irradiation times must be specified in the [CONTROL] block. This list provides the depletion time (in hours) at each state point in the depletion. A depletion is essentially a string of steady state solves with each state using the last state solution as the starting point for the next state. When specifying boundary conditions for a depletion, the first value in the list provides the state number. If the boundary condition is required, the first state value must be provided. After that, it is permissible to skip boundary conditions for subsequent statepoints and the code will automatically copy the previous state value for the current state.

If modeling a transient, the end time of the transient must be specified in the [CONTROL] block as well as the timestep size and a list of transient times when edits should be made. The first value of a boundary condition list is the time in the transient (in seconds). If the boundary condition is required, at least one point in the transient must be provided. If only one point in time is provided, the code will use that value for the entire transient. If more than one points are provided, the code will linearly interpolate between specified boundary condition times.

If modeling a steady state problem, the input is the same structure as the transient. Boundary conditions should be entered at a time of zero, the end time of the transient should be zero, and edits should be printed at the zero time location.

Boundary conditions can be specified in two ways: the rod surface temperature can be provided or the heat transfer coefficient and fluid temperature can be provided. If using the second option, the liquid heat transfer coefficient and liquid temperature are required and the vapor heat transfer coefficient and vapor temperature are optional and will be set to the liquid values if not provided.

2.2 Sample Input Decks

2.2.1 Setting Thermal Boundary Condition

The user can specify either axially varying temperature profile by providing `[HTCL]`, `[HTCV]`, `[TL]`, and `[Tv]` blocks (`Set_BC_1`) or predefined rod surface temperature temperature by providing `[Tw]` block (`Set_BC_2`).

`Set_BC_1`: **Predefined Axially Varying Temperature Profile** `Set_BC_1` requires the user to specify `HTCL`, `HTCV`, `TL`, and `Tv` blocks in order to set axial temperature profile provided in `CTFFuel`. Sample input deck for uniform in-pellet power distribution is given as:

```
[CONTROL]
pressure 155.0 ! bar
power 28.296 ! kW/m

[GEOM]
```

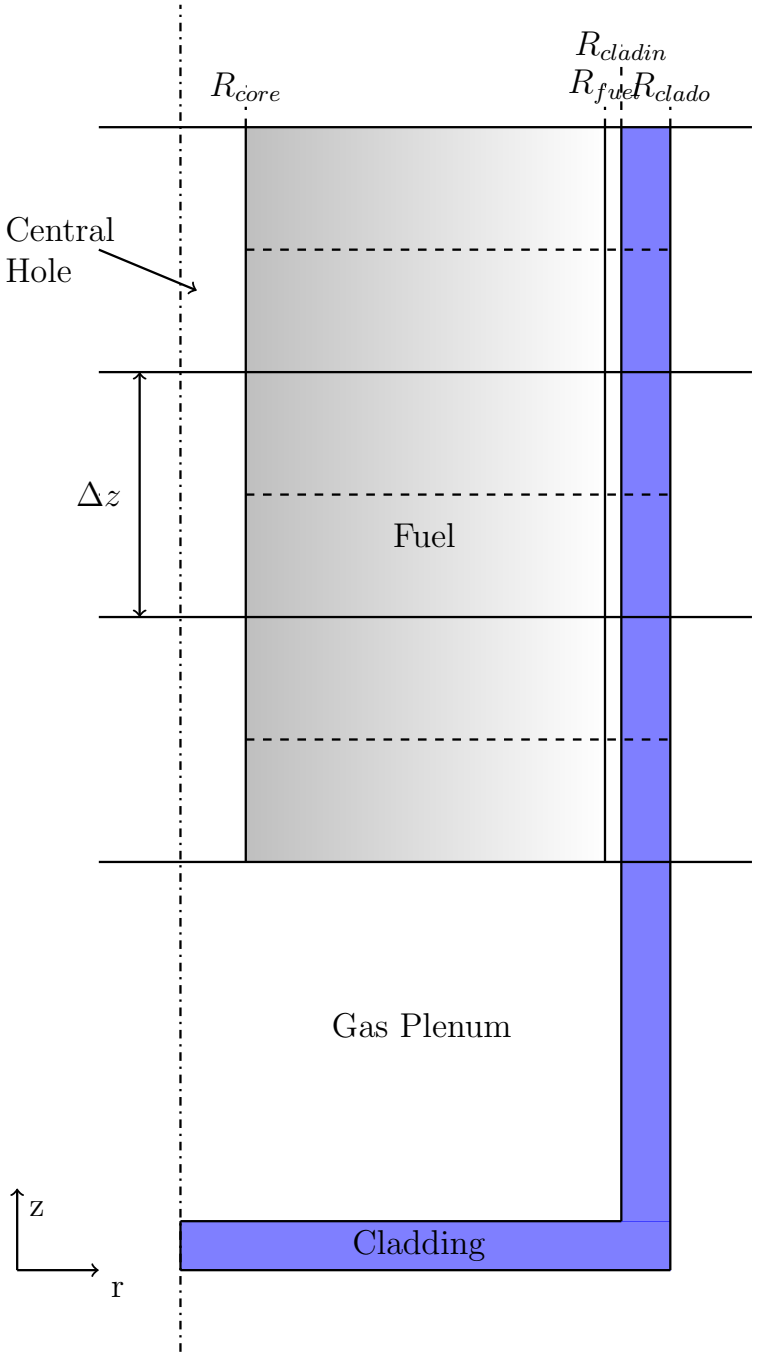

Figure 2.1: Nodalization scheme of fuel rod

Table 1: `[CONTROL]` block with list of input parameters.

Parameter	Description	Unit	Required (Default)
`pressure`	System pressure	bar	✓
`power`	Reference rod power	$^{kW}/_{m}$	✓
`fraction_directheat`	Fraction of local heat rate generated by the heater rods, released directly into the coolant	-	(0.0)
`num_iterations`	Number of iterations for convergence of conduction solution	-	(25)
`tolerance`	Temperature convergence for convergence of conduction solution	K	(0.05)
`dt`	The time-step size for the transient. Required only if transient.	s	(1.0e-3)
`tend`	The end of the transient (0.0 if modeling steady state only). Required only if transient.	s	
`tinit`	The initial rod temperature	°C	
`edits`	A list of the points in time in which edits will be made. Default behavior is always to print an edit at 0.0 s (after the intial steady state solution). If modeling a transient, an edit will always be made at the end of the transient.		
`depletion`	Set to 0 if modeling a transient or steady-state simulation and 1 if modeling a depletion.	-	✓
`irradiation_times`	A list of the time at each state. The first value should be the number of state points (integer value) in the list. The remaining values should be float values specifying the time, in hours, of each state point. This is only required when modeling a depletion.	hr	
`vtk_output`	Three options are availables: 0 - no vtk output files will be generated. 1 - only one vtk output file will be generated. The results are updated in the file every edit point defined in `edits`. 2 - multiple vtk output files will be generated for every point defined in `edits`. The files will be stored in an new automatically created directory.	-	(0)

Table 2: `[GEOM]` block with list of input parameters.

Parameter	Description	Unit	Required (Default)
`pellet_radius`	Fuel pellet radius	cm	✓
`inner_clad_radius`	Cladding inner radius	cm	✓
`outer_clad_radius`	Cladding outer radius	cm	✓
`central_hole_radius`	Central hole radius	cm	(0.0)
`oxide_thickness`	Thickness of the oxide layer	cm	(0.0)
`num_rings`	Number of rings in the fuel pellet	-	(10)
`th_density`	Fuel theoretical density	-	(0.95)
`fuel_rough`	Fuel surface roughness	cm	(0.0)
`cladi_rough`	Clad inner surface roughness	cm	(0.0)
`vplen`	Plenum volume	cm^3	
`pgas`	Initial fill gas pressure	bar	
`constant_kf`	User-defined fuel thermal conductivity	$W/m\text{-}K$	
`constant_kc`	User-defined cladding thermal conductivity	$W/m\text{-}K$	
`constant_gap_htc`	User-defined gap conductance	$W/m^2\text{-}K$	
`gapmod`	Gap conductance model selection	-	✓
`kfmodel`	Fuel thermal conductivity model selection	-	(matpro)
`dx`	Scalar axial mesh in rod	cm	✓
`gap_htc`	Axially defined gap conductance	$W/m^2\text{-}K$	
`ax_htc`	Axial location for `gap_htc` array	m	
`fillgas_He`	Molar fraction of fill gas, helium	-	
`fillgas_Xe`	Molar fraction of fill gas, xenon	-	
`fillgas_Ar`	Molar fraction of fill gas, argon	-	
`fillgas_Kr`	Molar fraction of fill gas, krypton	-	
`fillgas_H`	Molar fraction of fill gas, hydrogen	-	
`fillgas_N`	Molar fraction of fill gas, nitrogen	-	

Table 3: Time-dependent state blocks with list of input parameters.

Block	Description		Unit
[HTCL]	Block to define heat transfer coefficient of liquid		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	HTCL(i=1,NDX)	Liquid HTC at a given axial mesh	W/m^2-K
[HTCV]	Block to define heat transfer coefficient of vapor		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	HTCV(i=1,NDX)	Vapor HTC at a given axial mesh	W/m^2-K
[TL]	Block to define bulk liquid temperature		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	TL(i=1,NDX)	Bulk liquid tempeature at a given axial mesh	°C
[TV]	Block to define bulk vapor temperature		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	TV(i=1,NDX)	Bulk vapor temperature at a given axial mesh	°C
[PRESSURE]	Block to define bulk pressure		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	PRESSURE(i=1,NDX)	Pressure at a given axial mesh	bar
[TW]	Block to define the rod surface temperature		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	TW(i=1,NDX)	Rod surface temperature at a given axial mesh	°C
[RODQ]	Block to define axial power distribution		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	RODQ(i=1,NDX)	Normalized axial power fraction at a given axial mesh	-
[RADP]	Block to define radial power distribution		
	size	Number of radial rings (NDR) to specify radial power shape	-
	rad(i=1,NDR)	Normalized radius at a given radial ring	-
	power(i=1,NDR)	Normalized radial power fraction at a given radial ring	-
[EXPOSURE]	Block to define the fuel burnup		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	EXPOSURE(i=1,NDX)	Fuel burnup at a given axial mesh	MWd/kgU
[GAD]	Block to define gadolinia content		
	key	Shall be "trans" when modeling steady state or transient or "state" when modeling a depletion	–
	time	Time (float) or statepoint (integer)	s or —
	GAD(i=1,NDX)	Gadolinia content at a given axial mesh	-

```
pellet_radius 0.4096 ! cm
inner_clad_radius 0.418 ! cm
outer_clad_radius 0.475 ! cm
num_rings 3
th_density 0.95 ! -
gapmod constant
dx 5 3.8660E-02 8.2110E-02 8.2110E-02 8.2110E-02 3.86600E-02
constant_gap_htc 1000.0 ! W/m**2/K

[HTCL]
0.0  2.0210E+04 2.0230E+04 2.0251E+04 2.0274E+04 2.0299E+04

[HTCV]
0.0 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

[TL]
0.0 2.5947E+02 2.6050E+02 2.6161E+02 2.6281E+02 2.6413E+02

[Tv]
0.0 3.1896E+02 3.1896E+02 3.1895E+02 3.1894E+02 3.1893E+02

[RODQ]
0.0 1.00 1.00 1.00 1.00 1.00
```

`Set_BC_2`: **Predefined Rod Surface Temperature** To set the constant rod surface temperature as a boundary condition in `CTFFuel`, the user has to specify the rod surface temperature `[Tw]` block.

```
[CONTROL]
pressure 155.0 ! bar
power 28.296 ! kW/m

[GEOM]
pellet_radius 0.4096 ! cm
inner_clad_radius 0.418 ! cm
outer_clad_radius 0.475 ! cm
num_rings 3
th_density 0.95 ! -
gapmod constant
dx 5 3.8660E-02 8.2110E-02 8.2110E-02 8.2110E-02 3.86600E-02
constant_gap_htc 1000.0 ! W/m**2/K

[Tw]
0.0 2.5947E+02 2.6050E+02 2.6161E+02 2.6281E+02 2.6413E+02

[RODQ]
0.0 1.00 1.00 1.00 1.00 1.00
```

Figure **??** shows CTF-predicted fuel temperature profile with predefined thermal boundary conditions that are specified by using `Set_BC_1` and `Set_BC_2`. Since the thermal boundary conditions are identical, the contour plot is exactly the same for both sample decks provided above.

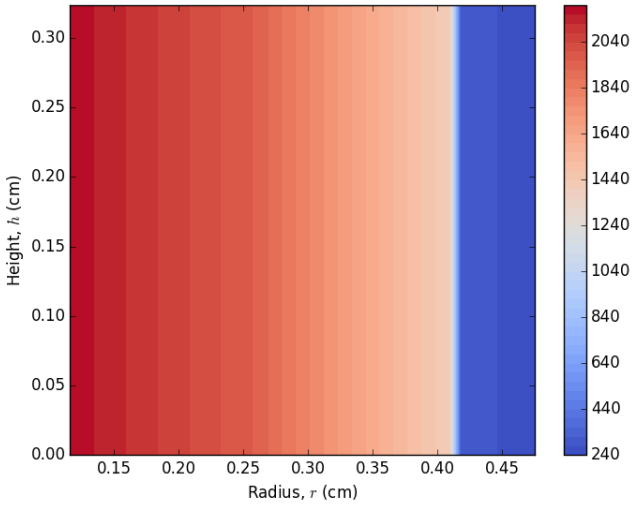

Figure 2.2: CTF-predicted temperature contour plot with `Set_BC_1` and `Set_BC_2`.

2.2.2 Axial Power distribution

The user can either specify an uniform or non-uniform power distribution axially. CTF `CTFFuel` requires the user to specify the axial power distribution in `[RODQ]` block. The uniform power shape can be specified by defining 1.0 in `[RODQ]` block. Sample input deck for non-uniform axial power distribution is given as:

```
[CONTROL]
pressure 155.132   ! bar
power    18.29943  ! kW/m
tinit    300.0  ! K

[GEOM]
central_hole_radius     0.0 ! cm
pellet_radius     0.5430 ! cm
inner_clad_radius 0.6370 ! cm
outer_clad_radius 0.6402 ! cm
num_rings         11
th_density        0.955   ! -
kfmodel         constant ! constant fuel thermal conductivity
constant_kf     5.0 ! fuel thermal conductivity
constant_kc     5.0 ! clad thermal conductivity
gapmod            constant
constant_gap_htc 1000.0 ! W/m**2/K
dx         10     0.06435      0.06435     0.06435    0.06435    0.06435    0.06435    0.06435
0.06435    0.06435    0.06435

[RODQ]
! Set at time 0.0, leave constant rest of transient
 0.0 1.40   1.60   1.80   2.00   2.20    2.00   1.80   1.60   1.40   1.20

[Tw]
 0.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0
```

Figure **??** shows CTF-predicted centerline temperatures with non-uniform axial power distribution. The axial power shape is arbitrarily selected for demonstration.

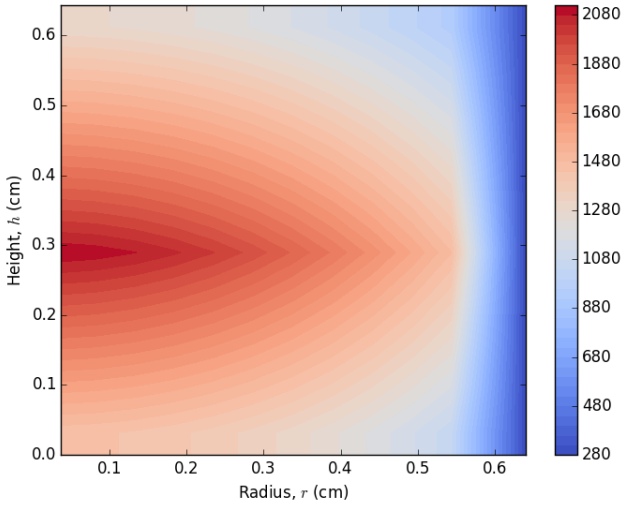

Figure 2.3: CTF-predicted temperature contour plot using predefined non-uniform axial power distribution

2.2.3 Radial Power distribution

The user can either specify an uniform or non-uniform power distribution radially. `CTFFuel` assumes uniform radial power distribution unless it is stated. The non-uniform in-pellet power profile is set in `[RADP]` block with specification of `size`, normalized radius `radp` and radial power factor `powr`. Sample input deck for non-uniform in-pellet power distribution is given as:

```
[CONTROL]
pressure 155.132   ! bar
power    18.29943  ! kW/m
tinit    300.0  ! K

[GEOM]
central_hole_radius     0.0 ! cm
pellet_radius     0.5430 ! cm
inner_clad_radius 0.6370 ! cm
outer_clad_radius 0.6402 ! cm
num_rings         10
th_density        0.955   ! -
gapmod            constant
constant_gap_htc 1000.0 ! W/m**2/K
dx         10     0.06435      0.06435     0.06435    0.06435    0.06435    0.06435
0.06435    0.06435    0.06435

[RODQ]
! Set at time 0.0, leave constant rest of transient
 0.0 1.00   1.00   1.00   1.00   1.00    1.00   1.00   1.00   1.00   1.00

[Tw]
 0.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0

[RADP]
 size 7
 rad  0.00000   0.20000   0.25000    0.55000    0.85000    0.90000    1.00000
 power 1.80000   1.60000   1.20000    1.00000    0.80000    0.40000    0.20000
```

Figure **??** shows CTF-predicted fuel temperature profile with arbitrarily chosen non-uniform radial power distribution.

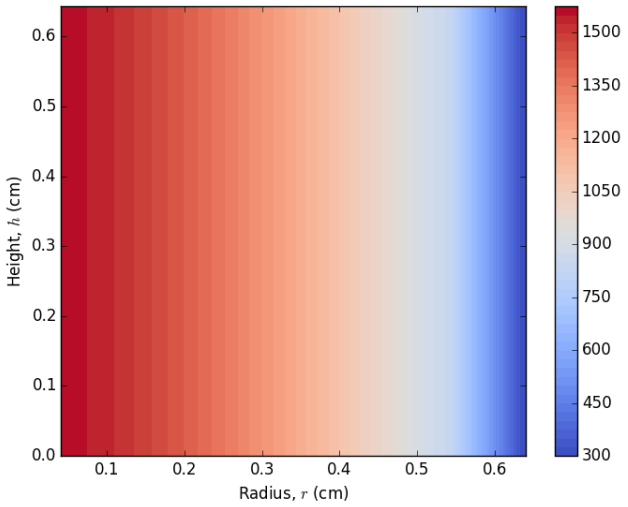

Figure 2.4: CTF-predicted temperature contour plot using predefined non-uniform radial power distribution

2.2.4 Fuel Thermal Conductivity

Knowledge of the fuel thermal conductivity is required to predict fuel performance during irradiation, in particular determination of the temperature distribution and of the fission gas release. Details of the implemented burnup dependent fuel thermal conductivities can be found in References [**atoptan˙kfuel**, **atoptan˙caslms2010**]. Options for the existing models are tabulated in Table **??**.

Table 4: Options for specification of fuel thermal conductivity

kfuel	**Model**
constant	User-defined fuel thermal conductivity
matpro	MATPRO-9 model (UO_2 fuel)
modnfi-uo2	Modified NFI model (UO_2 fuel)
halden-uo2	Halden model (UO_2 fuel)
duriez-mox	model (MOX fuel)
halden-mox	Halden model (MOX fuel)
amaya-mox	model (MOX fuel)
u3si2	White model [**white15**]

Importance of using burnup-dependent fuel thermal conductivities is to capture thermal degradation in conductivity. The thermal conductivity is correlated as a function of global parameters; irradiation temperature, microstructure (porosity, voids, Pu distribution, etc.) and burnup. The fuel burnup is the parameter to encounter irradiation effects. Some particular effects exist at low burnup: the fast buildup of irradiation damage, and at high burnup: the formation of **hbs!** (**hbs!**). Presence of additives (such as Pu, Gd, Cr) reduce the thermal conductivity of fresh fuel. Gadolinia rods offer the nuclear designer flexibility in choosing the optimal combination of initial reactivity worth and poison depletion rate [**nucmaterials˙vol2**].

Constant Fuel Thermal Conductivity The user has to specify `kfmodel` in `[GEOM]` block as `constant`. In addition to that, value of the fuel thermal conductivity is required to be defined using `constant_kf` in `[GEOM]` block. Sample input deck for constant fuel thermal conductivity:

```
[CONTROL]
pressure 155.0 ! bar
power 28.296 ! kW/m

[GEOM]
pellet_radius 0.4096 ! cm
inner_clad_radius 0.418 ! cm
outer_clad_radius 0.475 ! cm
num_rings 3
th_density 0.95 ! -
kfmodel    constant
constant_kf    10.0
gapmod constant
dx 5 3.8660E-02 8.2110E-02 8.2110E-02 8.2110E-02 3.86600E-02
constant_gap_htc 1000.0 ! W/m**2/K

[Tw]
0.0 2.5947E+02 2.6050E+02 2.6161E+02 2.6281E+02 2.6413E+02

[RODQ]
0.0 1.00 1.00 1.00 1.00 1.00
```

Figure **??** shows CTF-predicted fuel temperature profile with predefined fuel thermal conductivity in the input deck.

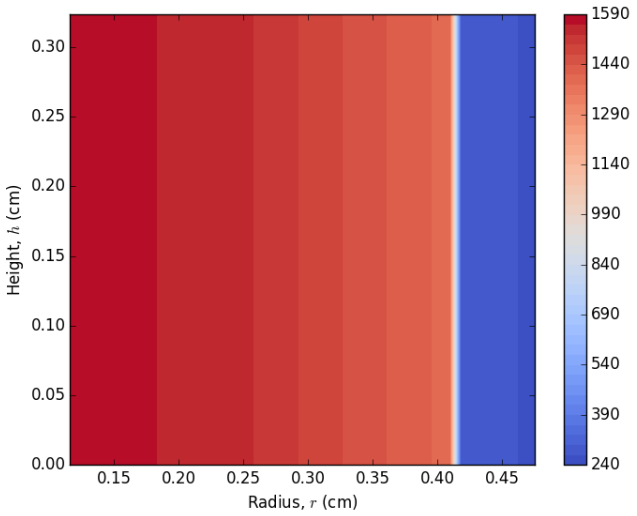

Figure 2.5: CTF-predicted temperature contour plot with predefined fuel thermal conductivity.

Default Fuel Thermal Conductivity Model `CTFFuel` assumes MATPRO model as its default fuel thermal conductivity model. Sample input deck for selection of default (temperature-dependent) MATPRO model:

```
[CONTROL]
pressure 155.0 ! bar
power 28.296 ! kW/m

[GEOM]
pellet_radius 0.4096 ! cm
inner_clad_radius 0.418 ! cm
outer_clad_radius 0.475 ! cm
num_rings 3
th_density 0.95 ! -
gapmod constant
dx 5 3.8660E-02 8.2110E-02 8.2110E-02 8.2110E-02 3.86600E-02
constant_gap_htc 1000.0 ! W/m**2/K

[Tw]
0.0 2.5947E+02 2.6050E+02 2.6161E+02 2.6281E+02 2.6413E+02

[RODQ]
0.0 1.00 1.00 1.00 1.00 1.00
```

Figure **??** shows CTF-predicted fuel temperature profile using the default fuel thermal conductivity model.

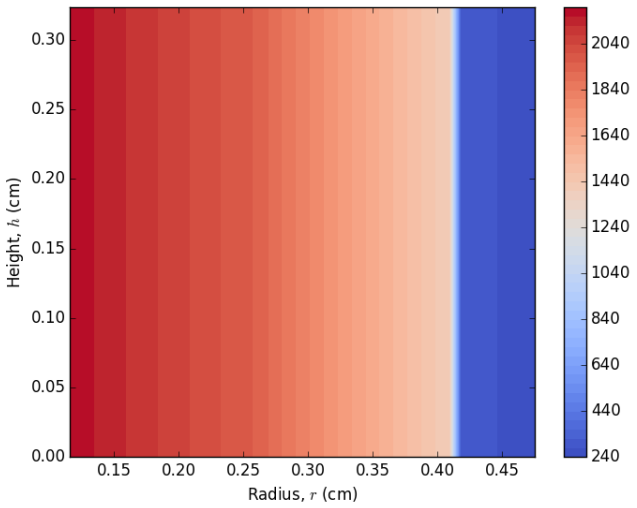

Figure 2.6: CTF-predicted temperature contour plot with predefined fuel thermal conductivity.

Burnup Dependent Fuel Thermal Conductivity Model The user has to specify the selected fuel thermal conductivity model using parameter `kfmodel`. In addition to that, function of those models (such as burnup `burnup` and Gd content `gdcontent` in `[STATE]` block. A list of the burnup-dependent fuel thermal conductivity models is listed in Table **??**. Sample input deck for selection of burnup-dependent fuel thermal conductivity model:

```
[CONTROL]
pressure 155.132   ! bar
power    18.29943  ! kW/m

[GEOM]
central_hole_radius     0.0 ! cm
pellet_radius      0.5430 ! cm
inner_clad_radius  0.6370 ! cm
outer_clad_radius  0.6402 ! cm
num_rings          3
th_density         0.955   ! -
gapmod             constant
dx       10      0.06435       0.06435      0.06435     0.06435     0.06435     0.06435     0.06435
0.06435     0.06435     0.06435
constant_gap_htc   1000.0  ! W/m**2/K
kfmodel      modnfi-uo2 ! burnup depdent fuel thermal cond

[RODQ]
! Set at time 0.0, leave constant rest of transient
 0.0 1.00    1.00    1.00    1.00    1.00     1.00    1.00    1.00    1.00    1.00

[Tw]
 0.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0

[EXPOSURE]
! Axial profile for the burnup values
 0.0    30.0    30.0    30.0    30.0    30.0    30.0    30.0    30.0    30.0    30.0
```

```
[GAD]
! Axial profile for doping of gadolinia content
 0.0    0.05    0.05    0.05    0.05    0.05    0.05    0.05    0.05    0.05    0.05
```

Figure ?? shows CTF-predicted centerline temperatures with different burnup levels and gadolinia contents. '`modnfi`' is selected as the fuel thermal conductivity model[1] of for demonstration. The thermal degradation of conductivity can be observed in Figure ??b due to irradiation effects (i.e., globally represented by burnup) and Figure ??c due to presence of Gd additives. Degraded thermal conductivity results in higher fuel temperature, as shown in Figure ??. For better understanding of the thermal degradation effects, line plots are shown in Figure ?? for three cases presented in Figure ?? since there is no axial variation in temperature.

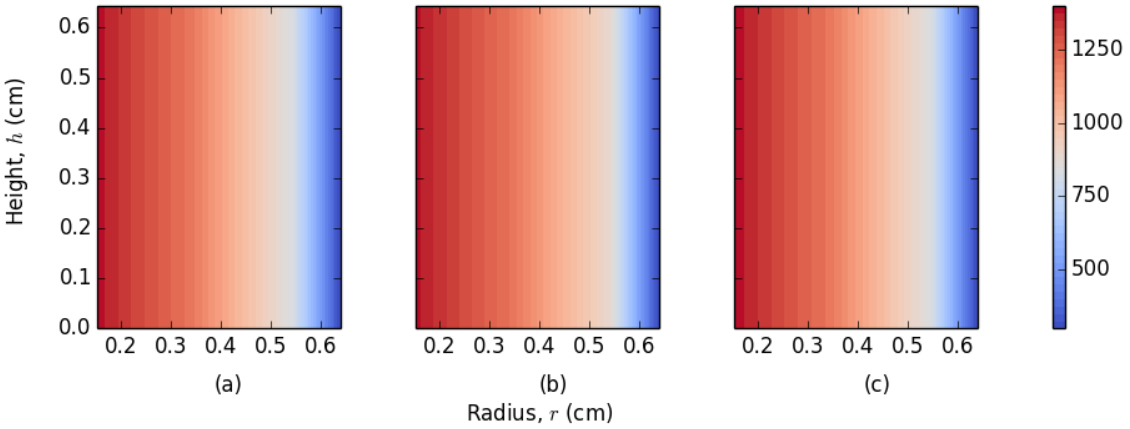

Figure 2.7: CTF-predicted temperature contour plots using '`modnfi`' model with different burnup levels and gadolinia contents: (a) zero burnup and Gd content (Bu = 0.0 MWd/kgU, Gd = 0.00), (b) non-zero burnup with zero Gd content (Bu = 30.0 MWd/kgU, Gd = 0.00), and (c) zero-burnup with non-zero Gd content (Bu = 0.0 MWd/kgU, Gd = 0.05).

[1]It is important to note that the thermal conductivity models have their applicability ranges, the results outside of these bounds are not be reliable.

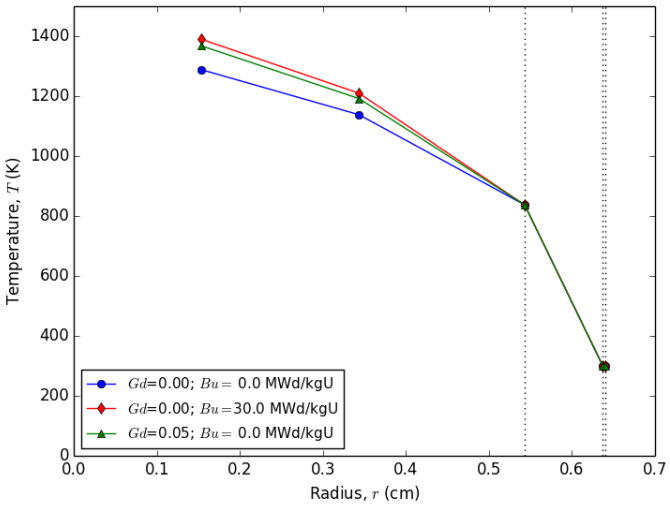

Figure 2.8: CTF-predicted temperature line plots using '`modnfi`' model with different burnup levels and gadolinia contents.

2.2.5 Gap Conductivity

The user can select any of three options to specify the gap conductance in fuel rod: constant uniform gap conductance, axially varied gap conductance and enabling CTF's dynamic gap conductance model for CTF to calculate the gap conductance.

Constant Gap Conductivity The user has to specify `gapmod` in `[GEOM]` block as `constant`. In addition to that, value of the gap conductance is required to be defined using `constant_gap_htc`. Sample input deck for assigning a constant gap conductance:

```
[CONTROL]
pressure 155.0 ! bar
power 28.296 ! kW/m

[GEOM]
pellet_radius 0.4096 ! cm
inner_clad_radius 0.418 ! cm
outer_clad_radius 0.475 ! cm
num_rings 3
th_density 0.95 ! -
dx 5 3.8660E-02 8.2110E-02 8.2110E-02 8.2110E-02 3.86600E-02
gapmod constant
constant_gap_htc 1000.0 ! W/m**2/K

[Tw]
0.0 2.5947E+02 2.6050E+02 2.6161E+02 2.6281E+02 2.6413E+02

[RODQ]
0.0 1.00 1.00 1.00 1.00 1.00
```

Figure **??** shows CTF-predicted fuel temperature profile with predefined gap conductance in the input deck.

Axially Non-Uniform Gap Conductivity The user has to specify `gapmod` in `[GEOM]` block as `axial_forcing`. In addition to that, values of the gap conductance at each axial level are required to be defined using `gap_htc` and `ax_htc`. Sample input deck when axially assigning gap conductance value:

```
[CONTROL]1.00
pressure 155.132   ! bar
power    18.29943  ! kW/m

[GEOM]
central_hole_radius     0.0 ! cm
pellet_radius     0.5430 ! cm
inner_clad_radius 0.6370 ! cm
outer_clad_radius 0.6402 ! cm
num_rings         11
th_density        0.955   ! -
kfmodel         constant ! constant fuel thermal conductivity
constant_kf     5.0 ! fuel thermal conductivity
constant_kc     5.0 ! clad thermal conductivity
gapmod            axial_forcing
dx        10     0.06435      0.06435     0.06435    0.06435    0.06435    0.06435
0.06435    0.06435    0.06435
gap_htc   10    1000.0    2000.0    2500.0    3000.0    4000.0   5000.0    3000.0    2000.0
1500.0   1000.0
```

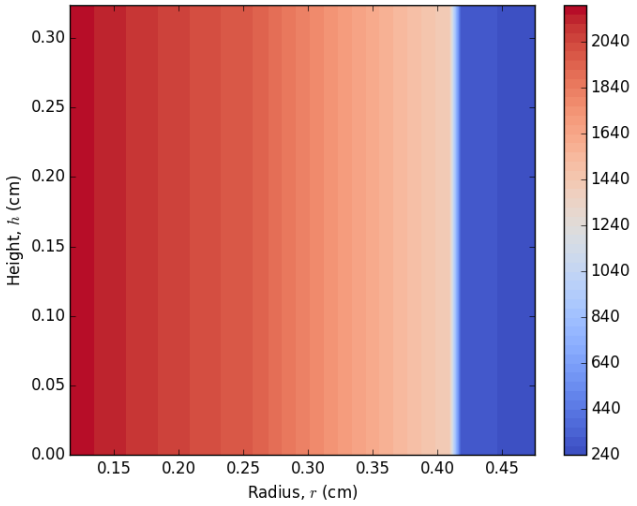

Figure 2.9: CTF-predicted temperature contour plot with predefined gap conductivity.

```
ax_htc     10      0.06435       0.06435      0.06435     0.06435     0.06435     0.06435     0.06435
0.06435     0.06435     0.06435

[RODQ]
! Set at time 0.0, leave constant rest of transient
 0.0 1.00    1.00    1.00    1.00    1.00     1.00    1.00    1.00    1.00    1.00

[Tw]
 0.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0 300.0
```

Figure **??** shows CTF-predicted fuel temperature profile with axially predefined gap conductance in the input deck.

Enabling CTF's Dynamic Gap Conductivity Model The user has to specify `gapmod` in `[GEOM]` block as `dynamic`. In addition to that, some specific parameters are required to be defined to be able to use CTF's dynamic gap conductance model such as: fuel/clad surface roughness (`fuel_rough`, `clad_rough`), plenum volume (`vplen`), fill gas pressure (`pgas`), gas composition[2] (`fillgas_He`, etc.) and so on. Sample input deck when enabling CTF's dynamic gap conductance model:

```
[CONTROL]
pressure 155.132   ! bar
power    18.29943  ! kW/m

[GEOM]
central_hole_radius     0.0 ! cm
pellet_radius     0.5430 ! cm
inner_clad_radius 0.6370 ! cm
outer_clad_radius 0.6402 ! cm
```

[2]Sum of fill gas fractions should be 1.0. If the `fillgas_` is not defined, the code sets it to zero.

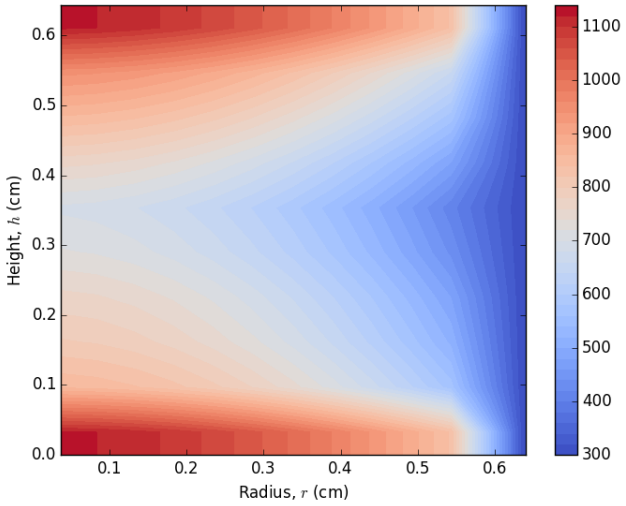

Figure 2.10: CTF-predicted temperature contour plot with axially predefined gap conductivity.

```
gapmod     dynamic
fuel_rough         1.0e-3 ! cm
cladi_rough        1.0e-3 ! cm
th_density         0.955   ! -
vplen    1.0e-5 ! m3
pgas     10.0 ! 1.00 bar
fillgas_He  1.0
dx        10  7.3152 7.3152 7.3152 7.3152 7.3152 7.3152 7.3152 7.3152 7.3152 7.3152

[RODQ]
! Set at time 0.0, leave constant rest of transient
 0.000 1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00

[Tw]
 0.0000    300.0    300.0    300.0    300.0    300.0    300.0    300.0    300.0    300.0    300.0
```

CTF's gap conductance model is reviewed in References [**atoptan˙caslms2010**, **atoptan˙reviewctfdyn**] with provided some preliminary results. However, this milestone report covers more benchmark studies with improved CTF's fuel performance modeling. Figure **??** shows CTF-predicted centerline temperatures with enabled CTF's dynamic gap conductance model.

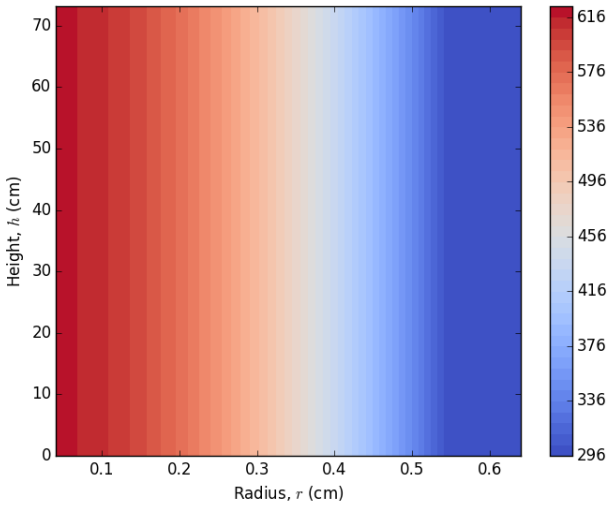

Figure 2.11: CTF-predicted temperature contour plot with enabled CTF's dynamic gap conductance model.

2.2.6 Cladding Thermal Conductivity

The user either can define constant cladding thermal conductivity or use CTF's default cladding thermal conductivity model.

Constant Cladding Thermal Conductivity The user has to specify `kcmodel` in `[GEOM]` block as `constant`. In addition to that, value of the cladding thermal conductivity is required to be defined using `constant_kc`. Sample input deck for assigning constant fuel thermal conductivity:

```
[CONTROL]
pressure 155.0 ! bar
power 28.296 ! kW/m

[GEOM]
pellet_radius 0.4096 ! cm
inner_clad_radius 0.418 ! cm
outer_clad_radius 0.475 ! cm
num_rings 3
th_density 0.95 ! -
kcmodel    constant
constant_kc    10.0
gapmod constant
dx 5 3.8660E-02 8.2110E-02 8.2110E-02 8.2110E-02 3.86600E-02
constant_gap_htc 1000.0 ! W/m**2/K

[Tw]
0.0 2.5947E+02 2.6050E+02 2.6161E+02 2.6281E+02 2.6413E+02

[RODQ]
0.0 1.00 1.00 1.00 1.00 1.00
```

Figure **??** shows CTF-predicted centerline temperatures with predefined cladding thermal conductivity.

Default Cladding Thermal Conductivity Model The user can use CTF's default cladding thermal conductivity model. Unless it stated, the cladding thermal conductivity model is applicable for Zr-2 and Zr-4 from MATPRO. Sample input deck for selection of cladding fuel thermal conductivity:

```
[CONTROL]
pressure 155.0 ! bar
power 28.296 ! kW/m

[GEOM]
pellet_radius 0.4096 ! cm
inner_clad_radius 0.418 ! cm
outer_clad_radius 0.475 ! cm
num_rings 3
th_density 0.95 ! -
gapmod constant
dx 5 3.8660E-02 8.2110E-02 8.2110E-02 8.2110E-02 3.86600E-02
constant_gap_htc 1000.0 ! W/m**2/K

[Tw]
0.0 2.5947E+02 2.6050E+02 2.6161E+02 2.6281E+02 2.6413E+02

[RODQ]
```

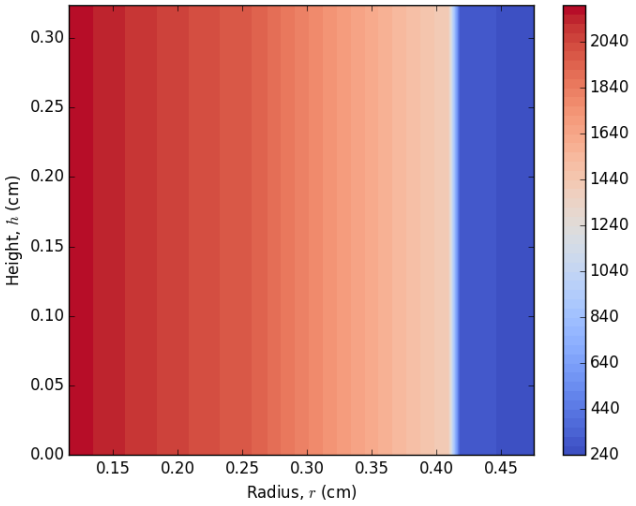

Figure 2.12: CTF-predicted temperature contour plot with predefined cladding thermal conductivity.

```
0.0  1.00  1.00  1.00  1.00  1.00
```

Figure **??** shows CTF-predicted centerline temperatures with default cladding thermal conductivity model.

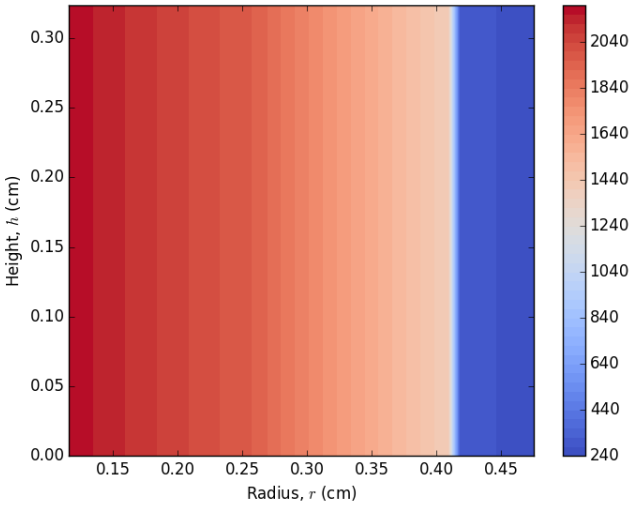

Figure 2.13: CTF-predicted temperature contour plot with predefined cladding thermal conductivity.

3. CTFFuel Output Specifications

The simulation results from `CTFFuel` are in `HDF5`[1] data format. Therefore, post-processing of data is slightly different than typical `CTF` outputs. `CTFFuel` outputs two sets of data; `[CORE]` data block with axial/radial scalar mesh information and `[STATE]` data blocks with simulation results (for example, pin temperature `'pin_temp'`) at defined state(s). `[STATE_0000]` is referred to the steady-state solution (e.g., simulation at time= 0.0). An example of HDF5 data output from `CTFFuel` is shown in Figure **??**. The `STATE` numbers increase based on the predefined time-steps in the input deck.

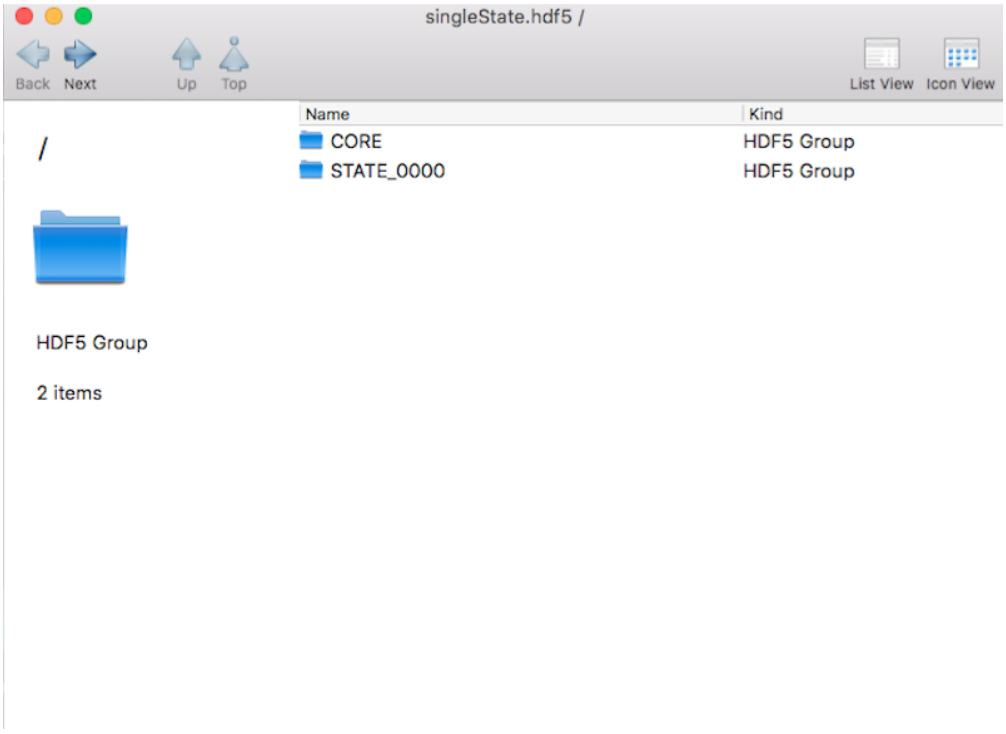

Figure 3.1: HDF5 data format

[1] **hdf!** (**hdf!**) is a file format designed to store and organize large amounts of data.

Using HDF5 with Python: For mac-os, Python module `h5py` for HDF5 can be downloaded as:

```
sudo pip install h5py
```

After installing,

```
>>> import h5py
>>> h5py.run_tests()
```

See http://docs.h5py.org/en/2.3/build.html for more information about the installation.

How to read groups and subgroups: Currently, outputs from `CTFFuel` are HDF5. If we want to compare simulation results with analytical calculations, it would be handful to read `HDF5` data from `Python` and plot the results. To do so, a sample script is provided to read `HDF5` data from `Python` script:

```
with h5py.File('test_output.h5','r') as hdf:
    base_items= list(hdf.items())
    print (base_items)
    group1= hdf.get('STATE_0000')
    g1_items= list(group1.items())
    print g1_items
    group1_sub= group1.get('/STATE_0001/pin_temp')
    data= np.array(group1.get('pin_temp'))
    print(data.shape)
    print data
```

A. Supplementary Python Scripts

Python script (`contourPlot.py`) to create contour plots from CTFFuel output in h5 format. The ease use of script is by importing `contourPlot` in your Python script and call the function that you want to use. The script is given as:

```
# Author: Aysenur Toptan
# Date: 07/27/2017
# Description:
#    Python script to generate contour plots by using H5 data
# out of CTFFuel resuts.

import h5py
import numpy as np
import matplotlib as mpl
from matplotlib import cm
mpl.use('Agg')
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import AxesGrid

## contour plot of fuelSolve predictions
# ftag        = fileName
# stateName   = stateName to read parameter out of H5
# variable    = parameter of interest (for ex. 'pin_temp')
# numberLevels= number of levels for contour plotting
def h5plot(ftag,stateName,variable,numberLevels):
    with h5py.File(ftag+'.h5','r') as hdf:
        base_items= list(hdf.items())
        # core parameters
        core= hdf.get('CORE')
        core_items= list(core.items())
        pin_radial= np.array(core.get('pin_radial'))
        pin_axial= np.array(core.get('pin_axial'))
        # state parameters
        state1= hdf.get(stateName)
        s1_items= list(state1.items())
        data= np.array(state1.get(variable))

        fig=plt.figure()
        plt.contourf(pin_radial, pin_axial, zip(*data), numberLevels, cmap=cm.coolwarm)
        plt.colorbar()
        plt.xlabel('Radius, $r$ (cm)');
        plt.ylabel('Height, $h$ (cm)');
        plt.grid(False);
        plt.savefig('./'+ftag+'.png');
        plt.close(fig)
```

For example, plotting contour plots of temperature distribution at 'STATE_0000':

```
import sys
# append the path, where contourPlot.py is located
sys.path.append('../')
import contourPlot

contourPlot.h5plot('output','STATE_0000','pin_temp',50)
```