```
$$\   $$\ $$\   $$\ $$\      $$\        $$$$$$\                      $$\       $$\
$$ |  $$ |$$$\  $$ |$$$\    $$$ |      $$  __$$\                     $$ |      \__|
$$ |  $$ |$$$$\ $$ |$$$$\  $$$$ |      $$ /  $$ |$$$$$$$\  $$\   $$\ $$$$$$$\  $$\  $$$$$$$\
$$ |  $$ |$$ $$\$$ |$$\$$\$$ $$ |      $$$$$$$$ |$$  __$$\ $$ |  $$ |$$  __$$\ $$ |$$  _____|
$$ |  $$ |$$ \$$$$ |$$ \$$$  $$ |      $$  __$$ |$$ |  $$ |$$ |  $$ |$$ |  $$ |$$ |\$$$$$$\
$$ |  $$ |$$ |\$$$ |$$ |\$  /$$ |      $$ |  $$ |$$ |  $$ |$$ |  $$ |$$ |  $$ |$$ | \____$$\
\$$$$$$  |$$ | \$$ |$$ | \_/ $$ |      $$ |  $$ |$$ |  $$ |\$$$$$$  |$$$$$$$  |$$ |$$$$$$$  |
 \______/ \__|  \__|\__|     \__|      \__|  \__|\__|  \__| \______/ \_______/ \__|\_______/
```

A multi-server MCNP-OpenFOAM/StarCCM+ coupling utility

User's Manual

Author: Khaled Talaat (University of New Mexico, Albuquerque)

Purpose of document: The user's manual provides a higher level description of Anubis and explains the compatibility, prerequisites, restrictions, and required input files to use Anubis. The code is written in MATLAB. No installation is necessary. Just make sure the Anubis folder and sub-folders are added to your MATLAB path and run Anubis. While Anubis is intended to be user-friendly, you must read this manual before using the code. It is not possible to learn how to use Anubis through trial and error. There are hundreds of input parameters (literally) to the code and a very large number of possible input combinations. The only way to use Anubis is to read this manual and study the examples. You are encouraged to also examine the Developer's Guide and the source code.

Scope
External coupling of steady state Monte Carlo neutron transport and computational fluid dynamics (CFD) for nuclear energy research applications.

Purpose
Anubis is a semi-modular, geometry-blind, and multi-server loose coupling utility that iteratively maps temperature and energy field effects between MCNP6 and OpenFOAM or STAR-CCM+ until convergence criteria are met. More specifically, Anubis transfers the steady state unnormalized prompt power profile from MCNP to CFD and uses the calculated temperature field from CFD to update the cross-section library, densities, and surface parameters in MCNP based on pre-defined user input.

Compatibility and Prerequisites
• Anubis is compatible with Windows 10 and Linux (tested on Ubuntu locally and remotely on CentOS). Mac is not supported.

• MATLAB R2021b or later release is required. Remote coupling in Anubis is not compatible with older versions of MATLAB that did not directly support SFTP.

• MCNP6, OpenFOAM, and STAR-CCM+ are not distributed with Anubis. You need MCNP6 and either of OpenFOAM or STAR-CCM+.

• MCNP6 is an export-controlled code. If you do not have MCNP, please see https://mcnp.lanl.gov/mcnp_how_to_get_to_mcnp.shtml.

• Anubis has been tested with MCNP6.1. Older versions of MCNP are not supported.

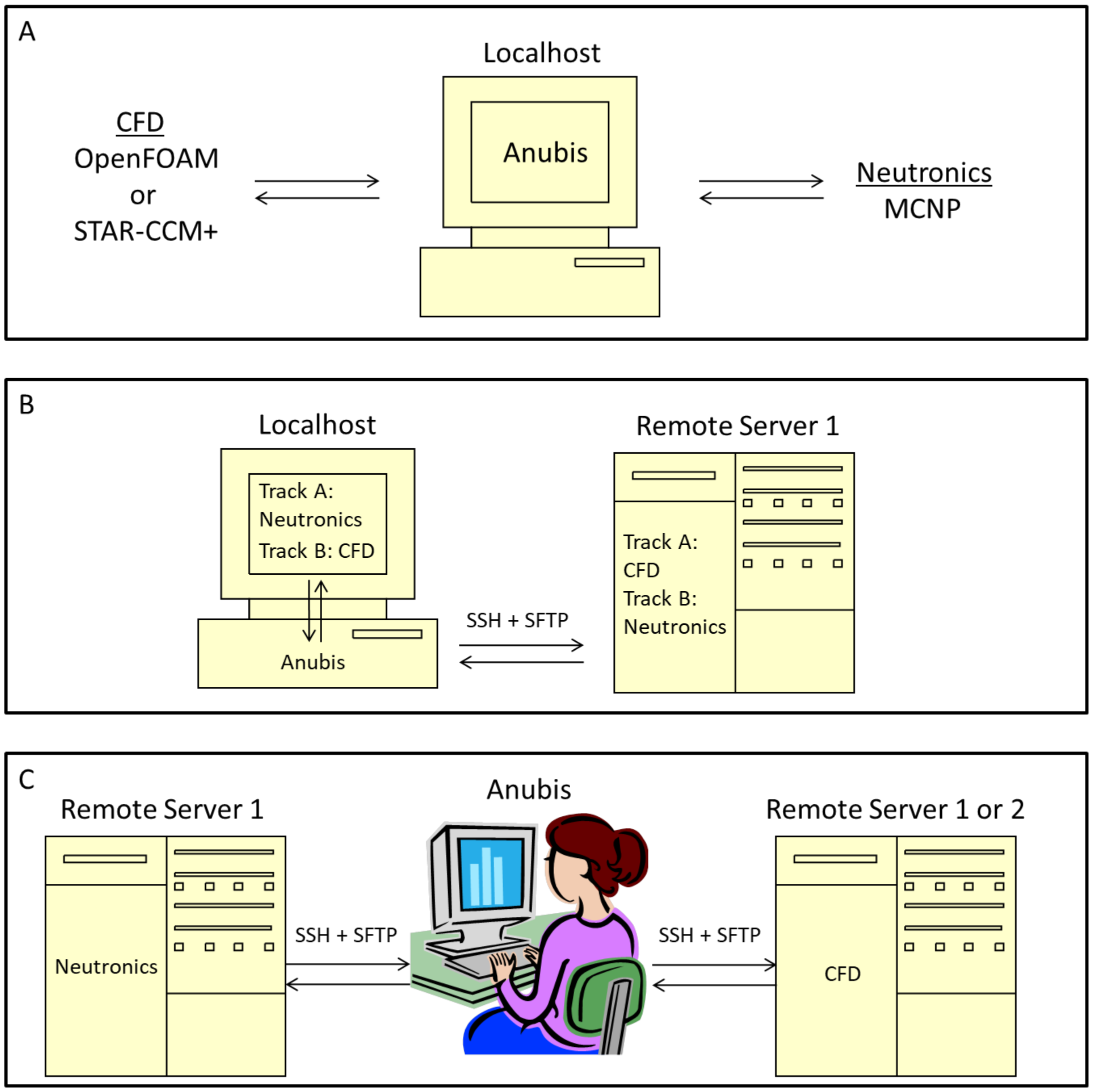
• Anubis is compatible with the chtMultiRegionFoam conjugate heat transfer solver in OpenFOAM 9 and OpenFOAM for Windows distributed by CFD Support (https://www.cfdsupport.com/openfoam-for-windows.html).

• SSHPASS is required for local runs of STAR-CCM+ on Linux.

• Anubis jobs can be local, entirely remote on one or more servers, or hybrid (i.e. it can run one program locally and one program remotely) as illustrated in the diagram below.

• To run jobs on remote servers, you must set up key based authentication by copying your RSA public key to authorized_keys in the remote host (see https://www.adminschoice.com/how-to-configure-ssh-without-password). Basically run ssh-keygen -t rsa and copy key from users/[yourusername]/.ssh folder to your .ssh/authorized_keys on your other machine and make sure that you are the only owner of all directories above .ssh starting from your own user directory and you must be the only one with write permission. If a folder is assigned group permission to write, remote connection will not work.

• The remote server must use a Portable Batch System (PBS) for job scheduling. MATLAB is not required on the remote server. You only need to run Anubis on the client.



High-level illustration of the different workflow options in Anubis: (a) entirely local, (b) hybrid, (c) entirely remote.

Statement on Cyber Security
As with any code with capability to transfer data from and to a remote system, cyber security can be a concern. While Anubis uses reliable and secure protocols such as SFTP and SSH, there is no guarantee that Anubis overall offers a secure environment. You are encouraged to consult with your IT team to evaluate the use of the code especially if you deal with sensitive information on the local or remote server(s). No information you provide in Anubis input is collected or transmitted to other parties besides the remote servers you specify in the input files. Passwords and other specified information are encrypted when transmitted to the remote servers that you specify. As some passwords for remote access and local access may be stored in input text files on your computer, it is recommended that you delete the files or redact the sensitive information after the Anubis runs are done and only use Anubis in work environments where you have exclusive access to the computer. Passwords are specified in input files as text instead of being passed as arguments to Anubis because different combinations of remote/local coupling options are possible (e.g. you can run MCNP locally and OpenFOAM/STAR-CCM+ remotely, or MCNP remotely and OpenFOAM/STAR-CCM+ locally, or both remotely on the same or different servers, or both locally). Nevertheless, you have the full right to modify the code as necessary for your needs.

Supported MCNP Surface Cards
Anubis recognizes the following surface mnemonics/cards in MCNP: "P","PX","PY","PZ","SO","S","SX","SY","SZ","C/X","C/Y","C/Z","CX","CY","CZ","K/X","K/Y","K/Z","KX","KY","KZ","SQ","GQ","TX","TY","TZ","X","Y","Z","P","BOX","RPP","SPH","RCC","RHP","HEX","REC","TRC","ELL","WED","ARB". Reflecting surfaces marked by * and white boundaries marked by + are also recognized. Any other surface types are not recognized and will cause errors.

Read Before Using (DO NOT SKIP)
• Anubis modifies xsdir index file you specify in case.json. You must back up your original xsdir index file before using the program. Anubis will attempt to back up your most recent xsdir index file under xsdirOriginalAnubis file and also under xsdirBackups directory in your MCNP data directory, but there is no guarantee that this operation will succeed (e.g. due to insufficient permissions).

• Doppler broadened cross-sections are generated using MAKXSF and Anubis assigns them extensions from .01c to .99c and places them in the xsdir file immediately below the "directory" keyword. You should manually restore your original xsdir file after using Anubis (not during an active run).

• Do NOT delete the xsdirOriginalAnubis file or the Coupler.signature file that is created in the MCNP data directory. The specs file generated using Anubis for MAKXSF uses the xsdirOriginalAnubis file which is created by copying original xsdir if the Coupler.signature file does not exist. This allows Anubis to use the original xsdir file in MAKXSF. Deleting any of them can mess up cross-sections that are generated using MAKXSF during an active run.

• For remote runs of MCNP, your MCNP cross-section data directory must be under your user account to be accessible to the transfer protocol. MAKXSF will run locally. Your local xsdir must be identical to the remote xsdir. Anubis will modify your local xsdir and upload modified xsdir to server in each iteration.

• The address of the output directory for an Anubis case must not contain any spaces. Make sure that you have write permission to the output directory that you specify. This is a common cause of errors.

• Mapping fields between iterations for remote runs of OpenFOAM requires that OpenFOAM should also be installed locally. If you do not have OpenFOAM locally, you should deactivate that option. No local installation of STAR-CCM+ is required for mapping fields in remote runs.

• Only continuous energy cross-sections are supported. MCNP uses the ZAID extension in the form .xxc, where xx is a double digit integer. As a result, Anubis can only define the same nuclide at 99 different temperatures with extensions from .01c to .99c. The implication of this MCNP limitation is that one nuclide can only be represented by 99 different temperature values within a simulation. This problem imposes limitations on the fundamental accuracy of the coupling for large systems with more than 99 regions.

• Temperatures below 250 K or above 2725 K will crash the program. You may modify code\Anubis_MCNP\getExtMCNP.m to change these limits if necessary for your run. Stay away from these limits in your initial guesses of temperature.

• Anubis is geometry-blind and does not provide tools for geometry registration. Coupling is based on input maps of CFD regions (collections of elements) to MCNP cells. You are expected to manually provide geometry mapping input or use an external utility for geometry registration.

• Anubis uses prompt energy deposition distribution. If you are studying a configuration where effective power distribution is different from prompt energy deposition distribution (e.g. in a liquid fueled reactor where delayed neutrons are emitted at a different location from fission), then the coupling framework here does not apply.

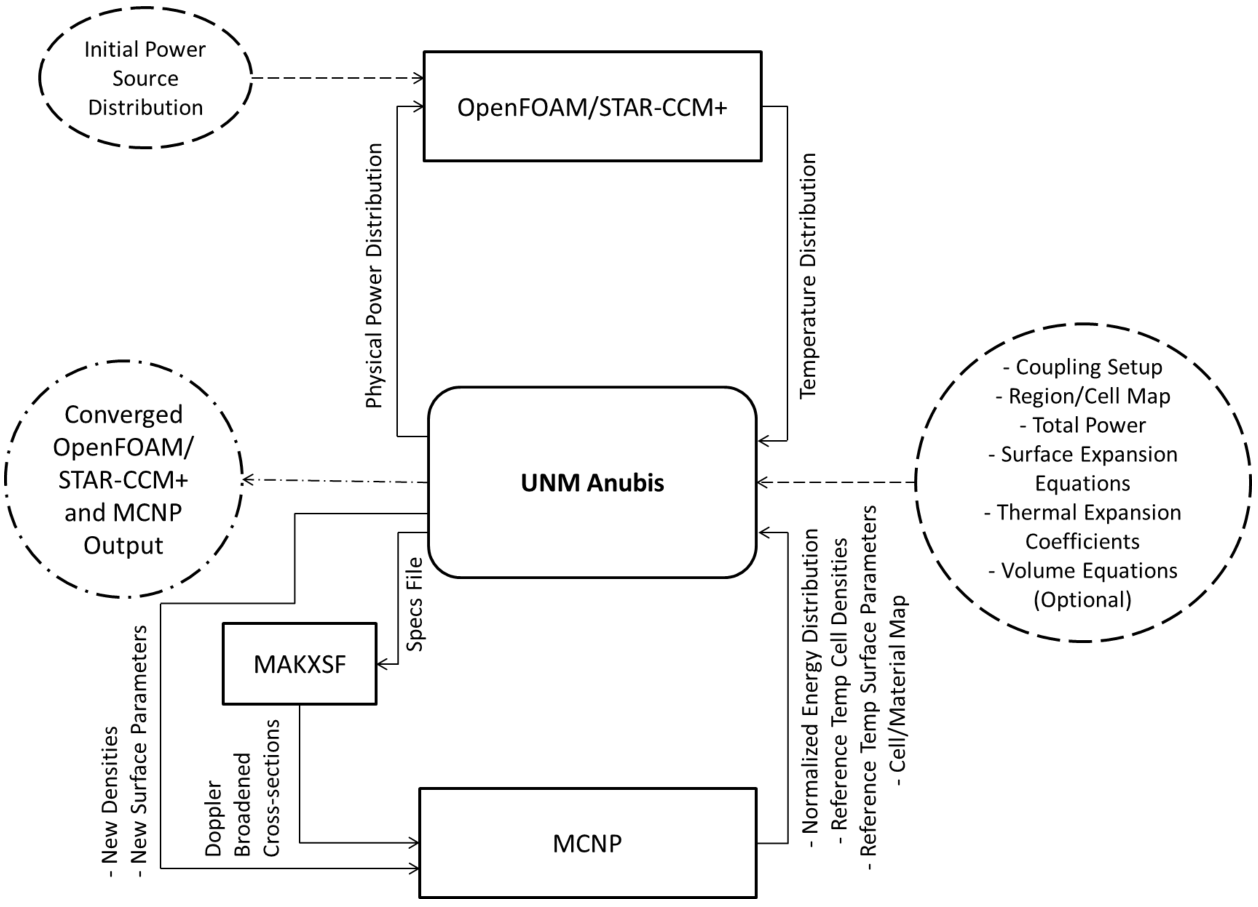
• Input files are read at the start of each iteration. You can change the input between iterations if necessary (e.g. to switch from a local run to a remote run or change a parameter or setting).

• Anubis input is case-sensitive.

• Last but not least, you understand that Anubis is a research code. It is not intended for industrial application. No warranty or liability is assumed. You are expected to read and understand the source code and adapt it to your needs if necessary – this is what good researchers do.

Data Flow

The coupling approach used in Anubis is illustrated in the following data flow diagram. The user is required to set up the initial conjugate heat transfer CFD case and the MCNP case as usual. The utility can start with either the neutronics case or the CFD case per pre-defined user input. If the user chooses to start with the neutronics case, then the initial temperature distribution has to be guessed during the setup of the initial MCNP case by the user (a uniform temperature assumption in initial case is okay). If the user chooses to start with the CFD case, the power distribution has to be guessed in initial CFD setup. The more accurate the initial guess is, the faster the convergence will be. Nevertheless, an accurate initial guess is difficult, and a viable option is to assume a uniform non-zero initial field which should conserve the total power specified in the case definition file assuming the user starts with the CFD case. The initial MCNP case should be defined at an arbitrary reference temperature with densities, surface parameters, and cross-sections defined at that temperature. The initial temperature distribution in MCNP should have no effect on convergence if the coupling starts with the CFD case and the vice versa is true.


Initial Power Source Distribution
OpenFOAM/STAR-CCM+
Physical Power Distribution
Temperature Distribution
Converged OpenFOAM/ STAR-CCM+ and MCNP Output
UNM Anubis
- Coupling Setup
- Region/Cell Map
- Total Power
- Surface Expansion Equations
- Thermal Expansion Coefficients
- Volume Equations (Optional)
Specs File
MAKXSF
- New Densities
- New Surface Parameters
Doppler Broadened Cross-sections
- Normalized Energy Distribution
- Reference Temp Cell Densities
- Reference Temp Surface Parameters
- Cell/Material Map
MCNP

Flow diagram of the Anubis coupling utility. Dashes denote user input, centerlines denote output to user, and boxes denote software applications.

The user is required to map the regions/cells, define the total power, define surface expansion equations, and define thermal expansion coefficient functions in the material library. User input is provided in JavaScript Object Notation (JSON) format, and the various options are discussed in detail in a later section. Anubis parses the user input files and starts the cycle based on the user's coupling set up. Assuming that the user starts with the CFD case, Anubis will execute the CFD code (OpenFOAM or STAR-CCM+) and extract the temperature field through direct parsing in case OpenFOAM is used and using a JAVA macro that Anubis dynamically generates for each problem in case STAR-CCM+ is used. Anubis then reads the initial MCNP case and builds objects that define the cell/material map, reference temperature cell densities, and reference temperature surface parameters. Anubis then calculates region average temperatures and maps them to the corresponding cells. It then uses the material/cell map to map the temperatures to the ZAIDs and generate a specs file that is used as input to MAKXSF and automatically executes MAKXSF. It then updates the ZAIDS in the MCNP input. Further, Anubis calculates the new cell densities based on Equation 1 (default option) unless otherwise specified. Surfaces are updated based on user-specified equations which can access parameters from the simulations. To ensure the conservation of the fissile mass, the user must make sure that the new volume that results from the user-specified surface equations satisfies Equation 2 in the fuel regions (if default option is used for density correction equation).

$$Density_{new}^{k} = \frac{Density_{old}^{k}}{(1 + (3\alpha^{k}\left(T^{k} - T_{ref}^{k}\right))} \tag{1}$$

$$Vol_{new}^{k} = Vol_{old}^{k}(1 + \left(3\alpha^{k}\left(T^{k} - T_{ref}^{k}\right)\right) \tag{2}$$

where k is the cell index, α_k is the thermal expansion coefficient of cell k evaluated at the average temperature of the corresponding CFD region. Anubis runs the updated MCNP case using the new cross-section libraries and parses the output to extract the normalized energy deposition distribution per source particle. As thermal hydraulics codes require physical power distribution, Anubis calculates an unnormalized power distribution for F6 and F7 tallies based on Equation 3. The shape of the power distribution is based solely on prompt neutrons. If the fuel is stationary, the total power distribution may be reasonably approximated by the prompt energy deposition shape scaled by the total power.

$$Power_{k} = \frac{M_{k} \times F7_{k} \times Power_{total}}{\sum M_{k} F7_{k}} \tag{3}$$

where M_k is the mass of cell k, $F7_k$ is the unnormalized F7 tally obtained from MCNP for cell k, and $Power_{total}$ is the total power specified by the user in the case.json file. In case the chtMultiRegionFoam solver in OpenFOAM is used, Anubis directly updates the heat sources by parsing and updating the relevant OpenFOAM files. In case STAR-CCM+ is used, Anubis dynamically generates a custom JAVA macro as shown in the example in Appendix C. The macro is used to update the power distribution for different regions and is executed when Anubis runs STAR-CCM+. The macro also exports tables that contain the temperature distribution in the system after the case is run. The region average temperatures are calculated, and the same cycle is repeated again until convergence criteria defined by the user is satisfied. Anubis currently allows the user to define convergence criteria based on either a maximum number of iterations or a convergence tolerance in the temperature field values compared to the previous n iteration(s).

Inputs
Anubis reads in 4 required user input files, 1 optional user input file, and a directory argument. Table 1 discusses the purpose of these files. Detailed examples of user input are shown in Appenix B.

Table 1: Description of user inputs read by Anubis

File/Input	Description
case.json	The file defines the coupling preferences and main set up. This is where the user should declare where the initial cases for neutronics and CFD are, type of coupling (local or remote) for each of CFD and neutronics, local paths for MCNP and OpenFOAM/STAR-CCM+ (always required for MCNP), configuration for remote connection if applicable, number of cores in processing, what cross-section evaluation to use, what materialsDB.json file to use, output directory, coupling flow (i.e. whether to start with CFD or neutronics), convergence criteria, total power in the system, type of MCNP tallies used, resume or new run, and various other options demonstrated in the example in Appendix B.
materialsDB.json	This is a material database file that

	allows Anubis to obtain thermal expansion coefficients of materials as function of temperature. The user may need to update the library and add new materials depending on the problem they are modeling. The active materialsDB.json file can be specified using a pointer in the case.json file. Units: µm/(m.K)
geometry.json	This file maps the MCNP cells to the OpenFOAM/STAR-CCM+ regions by ID. It also defines the initial/reference temperature at which the intial material properties of the cell are defined at, and what material ID in the materialsDB.json file the program should refer to for thermal expansion coefficients. The user may opt to leave some regions/cells uncoupled by simply not listing them. The user may match more than one OpenFOAM/STAR-CCM+ region to one MCNP cell using the “siblings” property in geometry.json. Power is divided equally amongst sibling regions. For sibling regions, only one sibling should have a record in the geometry.json file. The remaining siblings are listed by name under the siblings property of that region.
surfaceExpansion.json	This file defines how surface parameters should be adjusted with temperature using an in-house developed mark-up language. The new parameters can be specified as function of old or updated surface parameters, cell variables, and arbitrarily defined variables. The order of operations matters. The user declares two objects: variables and surfaces. Each object can have an arbitrary number of properties. Certain properties can update the surface object properties within Anubis which is later used to update the surface parameters in the MCNP file. The markup language is discussed in Tables 5 and 6.
volumes.json	For complex geometries involving lattices,

	the user may specify a file that describes how the cell volumes are related to the temperature-dependent surface parameters. This is an optional input file. If this option is used, mass densities should be used in MCNP input and not number densities. If the option is not used, Anubis will use volumes from MCNP output (you may use either mass or number densities in that case). These volumes are used in tally unnormalization to calculate the power distribution from the normalized prompt energy deposition dose profile.
directory argument	The user should specify the directory where the input .json files are located as an argument when running Anubis. Do not escape slashes. To run Anubis you simply call Anubis('directory').

Table 2: Objects in case.json input file.

Object	Level	Parent	Description
installDirs	1	-	-
MCNP	2	installDirs	*•Path to MCNP solver binaries directory.* *•Required for local and remote runs of MCNP.*
OpenFOAM	2^	installDirs	*•Path to OpenFOAM solver binaries directory.* *•Not required for local or remote runs if client (localhost) uses Ubuntu.* *•Required for remote runs on Windows if mapFieldsBetweenIterations is set to yes.* *•Required for local runs on Windows. This is typically under cygwin64\opt\OpenFOAM\OpenFOAM-*

			dev\platforms\ cygwin64mingw-w64DPInt32Opt\bin if you are running OpenFOAM for Windows from CFD Support.
StarCCM	2^	installDirs	*•Path to STAR-CCM+ binaries directory.* *•Required for local runs on Windows and Ubuntu.*
materialsDB	2	installDirs	*•Exact local path to thermal expansion coefficient library.*
setup	1	-	-
MCNP	2	setup	-
type	3	MCNP	Specify local or remote.
cores	3	MCNP	Number of cores per node.
username	3*	MCNP	*[*type: remote]*
hostname	3*	MCNP	*[*type: remote]*
password	3*	MCNP	*[*type: remote]*
module	3*	MCNP	*[*type: remote] Module to load on server.*
xsdatadirectory	3*	MCNP	*[*type: remote] Remote MCNP data directory (must be under your username and you must have write permission to it)*
remoteDir	3*	MCNP	*[*type: remote] Name of remote working directory for MCNP cases. You do not need to manually create this directory.*
queuesystem	3*	MCNP	*[*type: remote] Only option is PBS in this version of Anubis.*
PBStemplate	3*	MCNP	*[*type: remote] Exact local path to applicable PBS*

			template for MCNP.
nodes	3*	MCNP	*[*type: remote]*
walltime	3*	MCNP	*[*type: remote]*
jobtitle	3*	MCNP	*[*type: remote] May not contain spaces or special characters.*
cleanup	3*	MCNP	*[*type: remote] Specify yes to allow Anubis to delete data from remote server after each iteration or no to disable that feature.*
checktimequeue	3*	MCNP	*[*type: remote] Time in seconds between each time Anubis checks the status of the remote job when it is queued. Generally this should be > 200 seconds. Frequent connections can cause errors.*
checktimeactive	3*	MCNP	*[*type: remote] Time in seconds between each time Anubis checks the status of the remote job when it is running. Generally this should be > 200 seconds depending on how long you expect run to finish. Frequent connections can cause errors.*
OpenFOAM	2^	setup	-
type	3	OpenFOAM	Specify local or remote.
cores	3	OpenFOAM	Number of cores per node.
mapFieldsBetweenIterations	3	OpenFOAM	Specify yes to allow Anubis to call mapFields utility in

			OpenFOAM which accelerates convergence by passing temperature distribution from previous Anubis iteration's simulation as input.
username	3*	OpenFOAM	*[*type: remote]*
hostname	3*	OpenFOAM	*[*type: remote]*
password	3*	OpenFOAM	*[*type: remote]*
module	3*	OpenFOAM	*[*type: remote] This is the OpenFOAM module on the remote server not the solver.*
remoteDir	3*	OpenFOAM	*[*type: remote] Name of remote working directory for OpenFOAM cases. You do not need to manually create it.*
queuesystem	3*	OpenFOAM	*[*type: remote]*
PBStemplate	3*	OpenFOAM	*[*type: remote] Exact local path to applicable PBS template for OpenFOAM.*
nodes	3*	OpenFOAM	*[*type: remote]*
walltime	3*	OpenFOAM	*[*type: remote]*
jobtitle	3*	OpenFOAM	*[*type: remote]*
cleanup	3*	OpenFOAM	*[*type: remote]*
checktimequeue	3*	OpenFOAM	*[*type: remote]*
checktimeactive	3*	OpenFOAM	*[*type: remote]*
StarCCM	2^	setup	-
type	3	StarCCM	Specify local or remote.
cores	3	StarCCM	Number of cores per node.
mapFieldsBetweenIterations	3	StarCCM	If yes is specified, Anubis will start from the solution of the previous run/Anubis iteration. If no is specified, it will

			start using the temperature distribution of the initial case and the new power distribution.
sshpass	3*	StarCCM	*[*type: local] Your user password. Local runs on Ubuntu require use of sshpass utility. This is not required on Windows.*
licensePath	3	StarCCM	STAR-CCM+ license server path. Required for both local and remote runs.
username	3*	StarCCM	*[*type: remote]*
hostname	3*	StarCCM	*[*type: remote]*
password	3*	StarCCM	*[*type: remote]*
module	3*	StarCCM	*[*type: remote] This is the STAR-CCM+ module on the remote server.*
remoteDir	3*	StarCCM	*[*type: remote]*
queuesystem	3*	StarCCM	*[*type: remote]*
PBStemplate	3*	StarCCM	*[*type: remote] Exact local path to applicable PBS template for STAR-CCM+.*
nodes	3*	StarCCM	*[*type: remote]*
walltime	3*	StarCCM	*[*type: remote]*
jobtitle	3*	StarCCM	*[*type: remote]*
cleanup	3*	StarCCM	*[*type: remote]*
checktimequeue	3*	StarCCM	*[*type: remote]*
checktimeactive	3*	StarCCM	*[*type: remote]*
initialCases	1	-	-
MCNP	2	initialCases	Exact local path to initial MCNP input script. Required for both local and remote runs.
OpenFOAM	2^	initialCases	Local path to initial OpenFOAM

			case directory. Required for both local and remote runs. The case must be meshed and ready to run. For parallel cases, Anubis will call the decomposePar and recontructPar utilities in OpenFOAM.
StarCCM	2^	initialCases	Exact local path to initial STAR-CCM+ input file (.sim). Required for both local and remote runs. The case must be meshed and ready to run.
outputDir	1	-	Specify local path to output directory. This is required for both local and remote runs. Must not contain spaces and you must have write permission to the address.
couplingFlow	1	-	-
applicationCFD	2	couplingFlow	Specify OpenFOAM or StarCCM.
maximumStepsCCM	2*	couplingFlow	•Required for STAR-CCM+ runs only. •Specify maximum number of steps in each STAR-CCM+ run.
resumeExistingRun	2	couplingFlow	Specify yes to resume an existing Anubis run starting from the specified iteration. Specify no to start from initial case.
iteration	2	couplingFlow	Specify which iteration to start from if you are

			resuming a previous Anubis run. Make sure to check the saves directory and verify that a save file for the specified iteration exists (must be $\geq$ 1). If no is specified to resumeExistingRun, then 0 must be specified to iteration.
startWith	2	couplingFlow	Specify OpenFOAM or StarCCM or MCNP
minIterations	2	couplingFlow	Minimum number of Anubis iterations. Recommended value is 3.
maxIterations	2	couplingFlow	Maximum number of Anubis iterations. This depends on the size and complexity of your system but cases typically converge in about 4-5 iterations. Set to a large number to disable termination based on an iteration limit.
terminationCondition	2	couplingFlow	Currently only supported option you can specify is tempConvergence
convergenceCriterion	2	couplingFlow	Specify percentChange or absoluteChange. Comparison is done with respect to previous n iterations in all regions.
numberOfIterationsToCompareWith	2	couplingFlow	This is the number of iterations to be considered for

			convergence assessment. Specify 1 to compare with last iteration, 2 to compare with last 2 iterations, etc. This number must be smaller than minIterations.
convergenceTolerance	2	couplingFlow	Specify the value of percent change or absolute change that is tolerated for convergence.
geometricalOperations	1	-	-
thermalExpansionCorrection	2	geometricalOperations	Specify yes to allow Anubis to update surface parameters or no to disable updating surface parameters. Default should be yes.
densities	1	-	-
updateDensities	2	densities	Specify yes to allow Anubis to update densities or no to disable updating cell densities. Default should be yes.
densityCorrectionEquation	2	densities	•This is the density correction factor. •Specify default to use the factor $\frac{1}{(1+(3\alpha^k\left(T^k-T^k_{ref}\right))}$ where T is region average temperature, T_{ref} is cell's initial temperature, and α^k is thermal expansion coefficient evaluated at region's average temperature. •You may alternatively

			specify an equation. •Parameters you can use in equations are: - Temp (region average temperature). - maxTemp (max temperature in region). - MCNPInitialTemp (initial temperature of cell described in geometry.json or defaultInitialTemp). - alpha (thermal expansion coefficient evaluated at region's average temperature). - alphaMax (thermal expansion coefficient evaluated at region's maximum temperature). •Note that alpha is evaluated in units of 10^{-6} K in materialsDB.json. Every time you call alpha you should multiply it by 10^{-6}. •Make sure mass is conserved (especially fissile mass) after surface expansion and density correction.
defaultInitialTemp	2	densities	•This parameter is only used in the calculation if MCNPInitialTemp in geometry.json is not specified. •It defines the

			temperature at which the initial surface parameters and densities in the MCNP file are calculated at. This parameter is used in density correction calculation as T_{ref}. If MCNPInitialTemp is defined in geometry.json for a particular region, Anubis will use it instead.
crossSections	1	-	-
xsdirIndex	2	crossSections	Name of the active xsdir index file used by MCNP (e.g. xsdir or xsdir_mcnp6.1).
evaluation	2	crossSections	Specify 70s or 80s. If you choose 70s, then MCNP cross-sections ending with 70-74.c (ENDF/B-VII.1) will be used in Doppler broadening. If you choose 80s, then only cross-sections ending with 80-84.c (ENDF/B-VII.0) will be used.
notFound	2	crossSections	Applicable when cross-section at a particular temperature bound that is expected in interpolation for a particular ZAID is not found in xsdir.
accept80s	3	notFound	Specify yes or no. If you specify yes, Anubis will look for 80s (ENDF/B-VII.0) cross-section for

			the same ZAID at the same temperature limit as an alternative to use in specs file.
acceptNatural	3	notFound	Specify yes or no. If you specify yes, Anubis will look for natural element data at the same temperature limit as an alternative to use in specs file.
tallies	1	-	-
totalPowerWatts	2	tallies	This is the total power in the simulated system in watt. If you are simulating only one rod with periodic boundary conditions, this is the total power of that rod.
tallyType	2	tallies	Anubis can read energy deposition tallies (F6 or F7). Specify F6 or F7. Your specification must be consistent with what is used in initial MCNP input file.
volumes	2	tallies	•Specify as either MCNP or volumes.json. •If you choose MCNP, Anubis will use mass estimates from MCNP with latest surface parameters. •If you specify as volumes.json, which is useful if you have repeated structures, then you must define another input file

			(volumes.json) and specify the volume of each cell as a function of its surface parameters. Anubis will use updated values of surface parameters after thermal expansion to calculate the volumes.

(* denotes that object may or may not be necessary depending on your other choices at the same level, ^ denotes that at least one object at same level marked by the tag under same parent is required to be defined)

Important: slashes in directory addresses specified inside JSON input files must be escaped (replace any slash with \/) and there shall be no slashes at the end of a directory address.

Table 3: Objects in materialsDB.json file which is used to define thermal expansion coefficient library.

Object	Level	Parent	Description
mat[ID]	1	-	Arbitrary ID of element or compound. It is recommended that it starts with mat followed by atomic mass of each element separated by an underscore (e.g. mat92_7 indicates UN). The file can contain any number of materials each declared by a unique mat[ID].
chemicalFormula	2	mat[ID]	Define chemical formula for material for readability.
thermalExpCoeff	2	mat[ID]	Define thermal expansion coefficient as a function of temperature in µm/(m.K) (i.e. 10^{-6}/K). Temperature is indicated by the placeholder temp (e.g. -7.40741E-07*temp^2 + 0.00331037*temp + 6.59642).

Table 4: Objects in geometry.json input file.

Object	Level	Parent	Description
regions	1	-	Maps CFD regions to MCNP cells.
[RegionName]	2	regions	Defines the name of the CFD region. This should exactly match the region name defined in OpenFOAM or STAR-CCM+ (case-sensitive). Region names must not contain spaces. The file can contain any number of regions each declared by a unique [RegionName].
MCNPCellID	3	[RegionName]	ID of the MCNP cell to associate the CFD region with.
MCNPInitialTemp	3*	[RegionName]	•Optional. Required if initial MCNP case does not assume uniform temperature distribution. •Specifies the temperature in K that the initial density and surface parameters of the MCNP region are calculated at. If this is not defined, Anubis will use defaultInitialTemp defined in case.json.
materialsDBID	3	[RegionName]	The mat[ID] associated with the region in materialsDB.json which is used to define the thermal expansion coefficient for the effective material in the region as a function of temperature.
CCMRegionID	3*	[RegionName]	This is the Index of the region in STAR-CCM+ (required if STAR-CCM+ is used).
siblings	3*	[RegionName]	•Optional. •Use siblings to define sister regions in CFD that are coupled to the same MCNP cell if applicable.

			Anubis will divide the power associated with the MCNP cell equally between CFD siblings. •This feature is useful in cases involving a symmetrical configuration where the same cell ID in MCNP may be used to represent more than one element (non-contiguous) and is particularly useful in lattice geometries. As an example, "siblings": "rod2,rod3,rod4 " defined for rod1 indicates that rod1, rod2, rod3, and rod4 are represented by the same cell in MCNP (e.g. within a lattice structure). •If you are using OpenFOAM for CFD, you do not need to define a unique [RegionName] for each sibling. If you are using STAR-CCM+, you need to define [RegionName] for each sibling in the geometry.json file. Under each entry all siblings to the [RegionName] should be listed. This requirement is because Anubis needs to know the CCMRegionID which is under [RegionName].

Table 5: Objects in surfaceExpansion.json input file.

Object	Level	Parent	Description
variables	1	-	Used to define variables for use under surfaces object.
[VariableName]	2	variables	•An arbitrary number of variables can be defined. •Variables can be

			defined as function of surface parameters, other defined variables, or reserved variables in Anubis Markup (see Table 6). •Variables are not evaluated until they are called in statements under the surfaces object. •Order matters. Variables are evaluated from top to bottom. •Defined variables must not be named after surface variables or reserved variables.
surfaces	1	-	Used to define new surface parameters after surface expansion as a function of temperature to replace initial ones.
s_[surfaceID]_[paramnumber]	2	surfaces	•Surface parameters indicate surface ID and parameter number in MCNP input (e.g. s_11_1 points to the first surface parameter of surface 11 in updated MCNP

			input). •Right hand side (i.e. value) is an equation specified by the user. •Surface parameters can be defined as function of defined variables, reserved variables, and initial surface parameters in Anubis Markup (see Table 6).

Table 6: Reserved variables in Anubis Markup that can be used in equations defined in surfaceExpansion.json input file.

Reserved Variables (marked by #! and !# tags)	
Variable	Use
#!Temp(cellID)!#	Returns the average temperature of the CFD region corresponding to cellID in MCNP. For e.g.: #!Temp(1)!# returns the average temperature of the CFD region corresponding to cell 1 in MCNP.
#!maxTemp(cellID)!#	Returns the maximum temperature in the CFD region corresponding to cellID in MCNP.
#!alpha(cellID)!#	Returns the thermal expansion coefficient of cellID calculated at the corresponding CFD region’s average temperature.
#!refTemp(cellID)!#	Returns the reference temperature used for cellID in MCNP based on geometry.json

#!densityCorrection(cellID)!#	Returns the factor used for density correction from reference temperature density.
Surface Parameters The user may access reference temperature surface parameters or updated surface parameters of supported MCNP surface cards.	
Ref. Temperature Surface Parameters	Updated Surface Parameters
so_[surfaceID]_[paramnumber] For e.g. so_10_2 accesses the second parameter (indicated by paramnumber) of the surface with ID of 10 (indicated by surfaceID) in the original MCNP file specified by the user for the initial case. Read permission only is given (i.e. can be specified on right hand side of equation only). Example: "variables": { "originalvol": "2*pi*so_1_1^2*so_2_1", "deltavol": "#!alpha(1)!#*3*(#!Temp(1)!#-#!refTemp(1)!#)*originalvol*10^-6", "newvol": "originalvol + deltavol"} In this example, the surface parameters so_1_1 and so_2_1 are called in the definition of a variable (originalvol) in the variables object. The so_1_1 and so_2_1 represent the original first parameters in the initial MCNP input file for the surfaces	s_[surfaceID]_[paramnumber] For e.g. s_10_1 accesses the updated first parameter of the MCNP surface with ID of 10. Access to updated surface parameters provides write permission. Example: "s_2_1": "#!alpha(1)!#*(#!Temp(1)!#-#!refTemp(1)!#)*10^-6*s_2_1+s_2_1" In this example, the first parameter of surface 2 is updated as the product of the surface expansion coefficient evaluated at the average temperature in cell 1 and multiplied by thermal expansion coefficient unit of 10^{-6}/K and the difference in average temperature of last iteration compared to initial reference temperature and initial surface parameter all added to surface parameter from previous iteration. The updated surface parameter is effectively specified in same units as initial MCNP file (i.e. cm) because "#!alpha(1)!#*(#!Temp(1)!#-

1 and 2, respectively.	#!refTemp(1)!#)*10^-6 factor is unitless.

Important: do not escape division operators in equations specified in surfaceExpansion.json.

Table 7: Objects in the optional volumes.json input file.

Object	Level	Parent	Description
c[cellID]	1	-	Specifies an equation for cell volume of cellID as a function of updated surface parameters. The following example specifies equations for volume of cell 1 (c1) and cell 2 (c2) as function of associated surface parameters. "c1": "pi*s_100_1^2*(s_2_1-s_3_1)", "c2": "pi*s_100_1^2*(s_3_1-s_4_1)" Original initial surface parameters (i.e so_[surfaceID]_[paramnumber]) are also accessible if necessary. Equations may only be mathematical functions of updated or original surface parameters but volumes object cannot access other variables such as temperatures etc. Surfaces can be updated as a function of temperature using the surfaceExpansion.json input file.

At this stage all inputs to Anubis have been described. In the next section, complete examples are provided for clarity. These examples are not intended to represent problems of actual engineering or physics interest. They are only toy problems to illustrate how to use Anubis and for you to verify that Anubis does what is described. Let's do some Anubising.

Examples
Two examples are provided with Anubis for demonstration. The first one is a 3 region cylinder with a very low power core in the center and two reflectors at bottom and top. This example is very simple and it is recommended that you go over it by yourself. Different variants of the example are provided (check the case.json in each to develop some understanding of the Anubis input). The mesh is already generated and the case is ready to run. The physics setup of the initial cases for the 3 region cylinder example is not intended to be particularly meaningful. The case only serves as a quick test to ensure Anubis runs locally and remotely and is capable of reading the outputs and updating the inputs. Use it to check that Anubis runs on your system before looking at other examples. The more meaningful example is the second one which represents a 10-region uranium rod in water with reflecting boundaries in MCNP and symmetry boundary in CFD (STAR-CCM+). The mesh configuration is included but the mesh itself has not been generated. The first step is to generate the mesh and save the file. Use this as an opportunity to explore the case, and simultaneously look at MCNP initial input, geometry.json, surfaceExpansion.json, volumes.json, and materialsDB.json to develop solid understanding of how to set up Anubis input files. This is the example that is covered here in full. But first, here are some case.json examples for local and remote connections.

<u>Sample case.json input file with local MCNP and local OpenFOAM:</u>

```
{
"installDirs": {
            "MCNP": "\/media\/khaled\/PartitionE\/MCNP\/MCNP_CODE\/bin",
            "OpenFOAM": "",
            "materialsDB":
"\/media\/khaled\/PartitionE\/Anubis\/Anubis2021\/Examples\/Local\/FOAM_MCNP\/cylinde
rs\/casedefinition\/materialsDB.json"
            },
"setup": {
            "MCNP": {
                              "type":"local",
                              "cores": "4"
                        },
            "OpenFOAM": {
                              "type": "local",
                              "cores": "1",
                              "mapFieldsBetweenIterations": "yes"
                        }
            },
"initialCases": {
             "MCNP":
"\/media\/khaled\/PartitionE\/Anubis\/Anubis2021\/Examples\/Local\/FOAM_MCNP\/cylinde
rs\/initialcases\/mcnp\/cylinders.inp",
             "OpenFOAM":
"\/media\/khaled\/PartitionE\/Anubis\/Anubis2021\/Examples\/Local\/FOAM_MCNP\/cylinde
rs\/initialcases\/openfoam"
             },
"outputDir":
"\/media\/khaled\/PartitionE\/Anubis\/Anubis2021\/Examples\/Local\/FOAM_MCNP\/cylinde
rs",
"couplingFlow": {
             "applicationCFD": "OpenFOAM",
             "resumeExistingRun": "no",
             "iteration": "0",
             "startWith": "OpenFOAM",
             "minIterations": "3",
             "maxIterations": "5",
             "terminationCondition": "tempConvergence",
             "convergenceCriterion": "percentChange",
             "numberOfIterationsToCompareWith" : "2",
             "convergenceTolerance": "1"
             },
"geometricalOperations": {
        "thermalExpansionCorrection": "yes"
             },
"densities": {
             "updateDensities": "yes",
             "densityCorrectionEquation": "default",
             "defaultInitialTemp": "280"
             },
"crossSections": {
             "xsdirIndex": "xsdir_mcnp6.1",
             "evaluation": "70s",
             "notFound": {
                                "accept80s": "no",
                                "acceptNatural": "yes"
                         }
             },
"tallies": {
             "totalPowerWatts": "40000",
             "tallyType": "F7",
             "volumes":"MCNP"
      }
}
```

Notes/observations:

•All 9 level 1 inputs are present.

•Inputs are case-sensitive

•Numbers are defined as string inputs in case.json.

•Slashes in directory addresses are escaped.

•Notice that unnecessary objects could be left empty or could be removed (installDirs.OpenFOAM may be removed in the above example because it is not used in runs on Linux).

Sample case.json input file with local MCNP and local STAR-CCM+:

```
{
"installDirs": {
             "MCNP": "\/media\/khaled\/PartitionE\/MCNP\/MCNP_CODE\/bin",
             "StarCCM": "\/media\/khaled\/PartitionE\/StarCCM\/13.04.011\/STAR-
CCM+13.04.011\/star\/bin",
             "materialsDB":
"\/media\/khaled\/PartitionE\/Anubis\/Examples\/cylinders\/casedefinition\/materialsD
B.json"
             },
"setup": {
             "MCNP": {
                                 "type":"local",
                                 "cores": "4"
                           },
             "StarCCM": {
                                 "type": "local",
                                 "cores": "8",
                                 "sshpass": "REDACTED",
                                 "mapFieldsBetweenIterations": "yes",
                                 "licensePath": "REDACTED"
                           }
             },
"initialCases": {
             "MCNP":
"\/media\/khaled\/PartitionE\/Anubis\/Examples\/cylinders\/initialcases\/mcnp\/cylind
ers.inp",
             "StarCCM":
"\/media\/khaled\/PartitionE\/Anubis\/Examples\/cylinders\/initialcases\/StarCCM\/cyl
inders.sim"
             },
"outputDir": "\/media\/khaled\/PartitionE\/Anubis\/Examples\/cylinders",
"couplingFlow": {
       "applicationCFD": "StarCCM",
       "maximumStepsCCM": "200",
       "resumeExistingRun": "no",
       "iteration": "0",
        "startWith": "StarCCM",
             "minIterations": "3",
             "maxIterations": "5",
        "terminationCondition": "tempConvergence",
        "convergenceCriterion": "percentChange",
             "numberOfIterationsToCompareWith" : "2",
        "convergenceTolerance": "2"
             },
"geometricalOperations": {
        "thermalExpansionCorrection": "yes"
             },
"densities": {
```

```
            "updateDensities": "yes",
            "densityCorrectionEquation": "default",
            "defaultInitialTemp": "280"
            },
"crossSections": {
            "xsdirIndex": "xsdir",
            "evaluation": "70s",
            "notFound": {
                                "accept80s": "no",
                                "acceptNatural": "yes"
                          }
            },
"tallies": {
            "totalPowerWatts": "787.7543579",
            "tallyType": "F7",
            "volumes":"MCNP"
      }
}
```

Notes/observations:

•Local STAR-CCM+ runs require the definition of additional objects (sshpass and licensePath) under setup.StarCCM which do not have a required equivalent in OpenFOAM runs.

•In the couplingFlow object, an additional parameter is also required (maximumStepsCCM) when applicationCFD is set to StarCCM. This value should be large enough to allow the CFD run to converge on each Anubis iteration. This depends on your particular simulation.

•In this example mapFieldsBetweenIteration is enabled which allows Anubis to use field from previous CFD run as input to the new CFD run. This can accelerate convergence and reduce number of steps needed per CFD run.

Sample case.json input with remote MCNP and remote STAR-CCM+:

```
{
"installDirs": {
            "MCNP": "\/media\/khaled\/PartitionE\/MCNP\/MCNP_CODE\/bin",
            "materialsDB":
"\/media\/khaled\/PartitionE\/Anubis\/Examples\/UROD\/casedefinition\/materialsDB.jso
n"
            },
"setup": {
            "MCNP": {
                          "type":"remote",
                          "username": "REDACTED",
                          "hostname": "REDACTED",
                          "password": "REDACTED",
                          "module": "mcnp6",
                          "xsdatadirectory":"MCNPDATA",
                          "remoteDir": "Anubis1",
                          "queuesystem": "PBS",

  "PBStemplate":"H:\/Anubis\/templates\/PBS\/AnubisRunMCNP.pbs",
                          "nodes": "12",
                          "cores": "8",
                          "walltime": "48:00:00",
                          "jobtitle": "Anubis",
                          "cleanup": "yes",
                          "checktimequeue": "300",
                          "checktimeactive": "200"
                  },
            "StarCCM": {
                          "type": "remote",
                          "username": "REDACTED",
                          "hostname": "REDACTED",
                          "password": "REDACTED",
                          "module": "starccm/13.04.011",
                          "remoteDir": "Anubis2",
                          "queuesystem": "PBS",

  "PBStemplate":"H:\/Anubis\/templates\/PBS\/AnubisRunCCM.pbs",
                          "nodes": "12",
                          "cores": "8",
                          "walltime": "48:00:00",
                          "jobtitle": "Anubis",
                          "cleanup": "yes",
                          "checktimequeue": "300",
                          "checktimeactive": "200",
                          "mapFieldsBetweenIterations": "yes",
                          "licensePath": "REDACTED"
                  }
            },
"initialCases": {
            "MCNP":
"\/media\/khaled\/PartitionE\/Anubis\/Examples\/UROD\/initialcases\/mcnp\/urodchannel
.inp",
            "StarCCM":
"\/media\/khaled\/PartitionE\/Anubis\/Examples\/UROD\/initialcases\/StarCCM\/fuelchan
nel.sim"
            },
"outputDir": "\/media\/khaled\/PartitionE\/Anubis\/Examples\/UROD",
"couplingFlow": {
  "applicationCFD": "StarCCM",
  "maximumStepsCCM": "2500",
  "resumeExistingRun": "no",
  "iteration": "0",
   "startWith": "StarCCM",
            "minIterations": "3",
            "maxIterations": "5",
   "terminationCondition": "tempConvergence",
   "convergenceCriterion": "percentChange",
            "numberOfIterationsToCompareWith" : "2",
   "convergenceTolerance": "2"
```

```
             },
"geometricalOperations": {
        "thermalExpansionCorrection": "yes"
             },
"densities": {
             "updateDensities": "yes",
             "densityCorrectionEquation": "1/(1+(3*alpha*(Temp-MCNPInitialTemp)*10^-
      6))",
             "defaultInitialTemp": "293.6"
             },
"crossSections": {
             "xsdirIndex": "xsdir",
             "evaluation": "70s",
             "notFound": {
                                       "accept80s": "no",
                                       "acceptNatural": "yes"
                                 }
             },
"tallies": {
             "totalPowerWatts": "27489",
             "tallyType": "F7",

      "volumes":"\/media\/khaled\/PartitionE\/Anubis\/Examples\/UROD\/casedefinition
\/volumes.json"
      }
}
```

Notes/observations:

•The servers defined for MCNP and StarCCM can be the same or different ones. It is also possible to run one of the programs locally and one remotely.

•In remote runs we still need to define local installDir for MCNP because cross-sections are Doppler broadened locally using MAKXSF. This is also required for OpenFOAM on Windows if mapFieldsBetweenIterations is enabled (not required for STAR-CCM+).

•Initial cases are set up locally and local addresses are used.

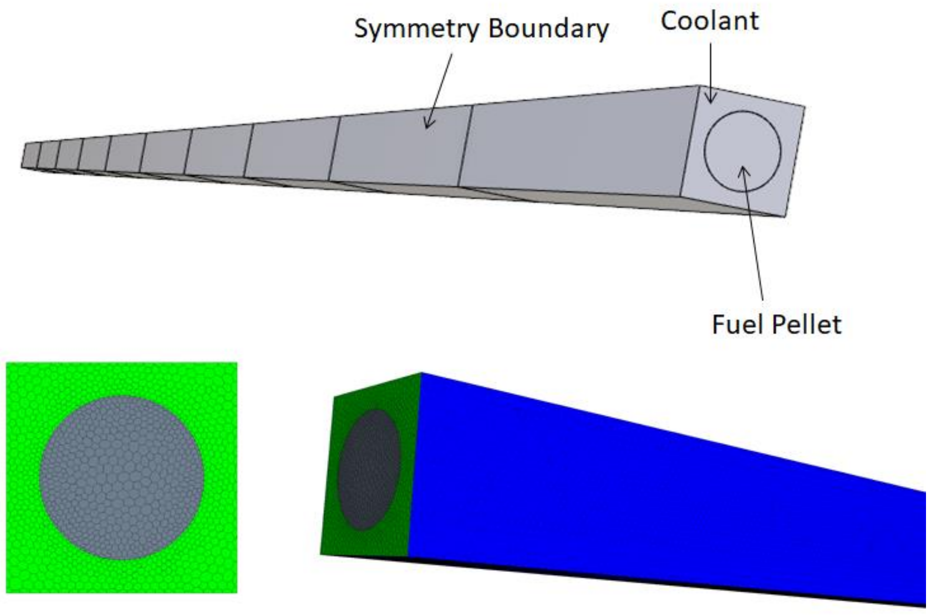
•The remote MCNP data directory must be defined and should be under your user account and accessible to Anubis. Anubis will upload the libraries generated by MAKXSF and the updated xsdir. Make sure that the active xsdir file on the remote server has the same name as the local one.

•Anubis and MATLAB are only required on the client (your local computer) and not the server. You only need to set up key-based authentication the first time you use Anubis.

•Notice that in the last example we specified an equation in densityCorrectionEquation using parameters described for the object in Table 2. The equation specified in this example is the same as the default equation (you may specify a different one).

The Uranium Rod (UROD) Example

A test case involving a 3.5% enriched uranium rod in water was used to verify the internal data transfer in the code. The geometry in the UROD example is illustrated in the figure below. The fuel rod and moderator regions are divided into 10 regions/cells each in both CFD and Monte Carlo. Each region is meshed using polyhedral cells in CFD (total of ~3 million). The total power of the rod is assumed to be 27.489 kW. The fuel rod is 1 m in length and 1 cm in diameter. A P/D ratio of 1.4 is used. Symmetry boundary conditions are used on the external side boundaries for the CFD case and insulation boundary conditions are used on top and bottom surfaces. A water flow velocity of 7 m/sec is assumed. In the MCNP case, the surfaces on the sides were specified as reflecting surfaces using the asterisk (*) marker. The example effectively represents an infinite lattice of uranium rods in water. To simplify, only the 10 fuel regions/cells were coupled in this test case and not the moderator (energy deposition is moderator is not significant). Uniform heat generation is assumed in the initial CFD case (while conserving total power).



Geometry used in the test case and hexahedral CFD mesh

The first step in running any Anubis job is setting up and initial CFD and MCNP simulations which should both be ready to run. The CFD geometry should be already meshed. The next step is to define the Anubis input files. In addition to case.json which was discussed earlier, at least 3 other input files are required and an additional one is optional. Do not let Table 2 scare you. The case.json file is pretty much a file you copy and paste from one of the examples provided and just adapt to your needs. The last example in case.json files is that of this UROD example. It suffices to note that the total power of the rod of 27489 W is specified under totalPowerWatts. Anubis uses this to unnormalize the prompt energy deposition distribution from MCNP according to Equation 3. Most of the time you spend setting up Anubis input for any realistic case will be on geometry.json and surfaceExpansion.json.

The geometry.json input file:

```
{
"regions": {
            "Fuel1": {
                                    "CCMRegionID": "2",
                                    "MCNPCellID": "10",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"
                              },
            "Fuel2": {
                                    "CCMRegionID": "3",
                                    "MCNPCellID": "9",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                              },
            "Fuel3": {
                                    "CCMRegionID": "7",
                                    "MCNPCellID": "8",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                              },
            "Fuel4": {
                                    "CCMRegionID": "9",
                                    "MCNPCellID": "7",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                              },
            "Fuel5": {
                                    "CCMRegionID": "0",
                                    "MCNPCellID": "6",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"
```

```
                            },
        "Fuel6": {
                                    "CCMRegionID": "4",
                                    "MCNPCellID": "5",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                            },
        "Fuel7": {
                                    "CCMRegionID": "8",
                                    "MCNPCellID": "4",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                            },
        "Fuel8": {
                                    "CCMRegionID": "1",
                                    "MCNPCellID": "3",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                            },
        "Fuel9": {
                                    "CCMRegionID": "5",
                                    "MCNPCellID": "2",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                            },
        "Fuel10": {
                                    "CCMRegionID": "6",
                                    "MCNPCellID": "1",
                                    "MCNPInitialTemp": "293.6",
                                    "materialsDBID": "mat92"

                            }
    }
}
```

Notes/observations:

•Region names must be identical to those defined in the CFD case.

•Regions represent groups of mesh elements in CFD. It is typical in conjugate heat transfer calculations to divide the geometry into multiple regions because different equations are solved in the solid and fluid. Anubis requires that you define the regions on the basis of the MCNP cells (i.e. your CFD regions are geometrically equivalent to your MCNP cells).

•CFD geometry does not have to precisely match the MCNP geometry since coupling of regions and cells is done by ID instead of spatial registration. This is one major advantage to the coupling approach employed here. You can employ approximations in the CFD model that you do not employ in neutronics and the vice versa. However, if they are substantially different, the problem would be unphysical and may never converge.

•Not all regions or cells in CFD and MCNP cases need to be coupled. In this example we only coupled fuel regions because that is where most of the power is generated. If you expect significant power generation in the moderator due to energy deposition, you may also couple moderator regions. The temperature of the moderator will still vary as a result of heat transfer regardless.

The surfaceExpansion.json *input file (explicit version):*

```
{
"variables": {
    "originalfuelregionvol": "pi*so_1_1^2*(so_2_1-so_3_1)",
    "deltavol1": "#!alpha(1)!#*3*(#!Temp(1)!#-
#!refTemp(1)!#)*originalfuelregionvol*10^-6",
    "deltavol2": "#!alpha(2)!#*3*(#!Temp(2)!#-
#!refTemp(2)!#)*originalfuelregionvol*10^-6",
    "deltavol3": "#!alpha(3)!#*3*(#!Temp(3)!#-
#!refTemp(3)!#)*originalfuelregionvol*10^-6",
    "deltavol4": "#!alpha(4)!#*3*(#!Temp(4)!#-
#!refTemp(4)!#)*originalfuelregionvol*10^-6",
    "deltavol5": "#!alpha(5)!#*3*(#!Temp(5)!#-
#!refTemp(5)!#)*originalfuelregionvol*10^-6",
    "deltavol6": "#!alpha(6)!#*3*(#!Temp(6)!#-
#!refTemp(6)!#)*originalfuelregionvol*10^-6",
    "deltavol7": "#!alpha(7)!#*3*(#!Temp(7)!#-
#!refTemp(7)!#)*originalfuelregionvol*10^-6",
    "deltavol8": "#!alpha(8)!#*3*(#!Temp(8)!#-
#!refTemp(8)!#)*originalfuelregionvol*10^-6",
    "deltavol9": "#!alpha(9)!#*3*(#!Temp(9)!#-
#!refTemp(9)!#)*originalfuelregionvol*10^-6",
    "deltavol10": "#!alpha(10)!#*3*(#!Temp(10)!#-
#!refTemp(10)!#)*originalfuelregionvol*10^-6",
    "newvol1": "originalfuelregionvol + deltavol1",
    "newvol2": "originalfuelregionvol + deltavol2",
    "newvol3": "originalfuelregionvol + deltavol3",
    "newvol4": "originalfuelregionvol + deltavol4",
    "newvol5": "originalfuelregionvol + deltavol5",
    "newvol6": "originalfuelregionvol + deltavol6",
    "newvol7": "originalfuelregionvol + deltavol7",
    "newvol8": "originalfuelregionvol + deltavol8",
    "newvol9": "originalfuelregionvol + deltavol9",
    "newvol10": "originalfuelregionvol + deltavol10",
    "fuelregionheight": "10"
},
"surfaces": {
      "s_11_1": "#!alpha(10)!#*(#!Temp(10)!#-#!refTemp(10)!#)*10^-
6*fuelregionheight+fuelregionheight + s_12_1",
      "s_10_1": "#!alpha(9)!#*(#!Temp(9)!#-#!refTemp(9)!#)*10^-
6*fuelregionheight+fuelregionheight + s_11_1",
      "s_9_1": "#!alpha(8)!#*(#!Temp(8)!#-#!refTemp(8)!#)*10^-
6*fuelregionheight+fuelregionheight + s_10_1",
      "s_8_1": "#!alpha(7)!#*(#!Temp(7)!#-#!refTemp(7)!#)*10^-
6*fuelregionheight+fuelregionheight + s_9_1",
      "s_7_1": "#!alpha(6)!#*(#!Temp(6)!#-#!refTemp(6)!#)*10^-
6*fuelregionheight+fuelregionheight + s_8_1",
      "s_6_1": "#!alpha(5)!#*(#!Temp(5)!#-#!refTemp(5)!#)*10^-
6*fuelregionheight+fuelregionheight + s_7_1",
      "s_5_1": "#!alpha(4)!#*(#!Temp(4)!#-#!refTemp(4)!#)*10^-
6*fuelregionheight+fuelregionheight + s_6_1",
      "s_4_1": "#!alpha(3)!#*(#!Temp(3)!#-#!refTemp(3)!#)*10^-
6*fuelregionheight+fuelregionheight + s_5_1",
      "s_3_1": "#!alpha(2)!#*(#!Temp(2)!#-#!refTemp(2)!#)*10^-
6*fuelregionheight+fuelregionheight + s_4_1",
      "s_2_1": "#!alpha(1)!#*(#!Temp(1)!#-#!refTemp(1)!#)*10^-
6*fuelregionheight+fuelregionheight + s_3_1",
      "s_100_1": "sqrt(newvol1/(pi*(s_2_1-s_3_1)))",
      "s_101_1": "sqrt(newvol2/(pi*(s_3_1-s_4_1)))",
      "s_102_1": "sqrt(newvol3/(pi*(s_4_1-s_5_1)))",
      "s_103_1": "sqrt(newvol4/(pi*(s_5_1-s_6_1)))",
      "s_104_1": "sqrt(newvol5/(pi*(s_6_1-s_7_1)))",
      "s_105_1": "sqrt(newvol6/(pi*(s_7_1-s_8_1)))",
      "s_106_1": "sqrt(newvol7/(pi*(s_8_1-s_9_1)))",
      "s_107_1": "sqrt(newvol8/(pi*(s_9_1-s_10_1)))",
      "s_108_1": "sqrt(newvol9/(pi*(s_10_1-s_11_1)))",
      "s_109_1": "sqrt(newvol10/(pi*(s_11_1-s_12_1)))",
}
}
```

The surfaceExpansion.json *input file (concise version):*

```
{
"variables": {
    "originalfuelregionvol": "pi*so_100_1^2*(so_2_1-so_3_1)",
    "newvol1": "originalfuelregionvol/#!densityCorrection(1)!#",
    "newvol2": "originalfuelregionvol/#!densityCorrection(2)!#",
    "newvol3": "originalfuelregionvol/#!densityCorrection(3)!#",
    "newvol4": "originalfuelregionvol/#!densityCorrection(4)!#",
    "newvol5": "originalfuelregionvol/#!densityCorrection(5)!#",
    "newvol6": "originalfuelregionvol/#!densityCorrection(6)!#",
    "newvol7": "originalfuelregionvol/#!densityCorrection(7)!#",
    "newvol8": "originalfuelregionvol/#!densityCorrection(8)!#",
    "newvol9": "originalfuelregionvol/#!densityCorrection(9)!#",
    "newvol10": "originalfuelregionvol/#!densityCorrection(10)!#",
    "fuelregionheight": "10"
},
"surfaces": {
      "s_11_1": "#!alpha(10)!#*(#!Temp(10)!#-#!refTemp(10)!#)*10^-
6*fuelregionheight+fuelregionheight + s_12_1",
      "s_10_1": "#!alpha(9)!#*(#!Temp(9)!#-#!refTemp(9)!#)*10^-
6*fuelregionheight+fuelregionheight + s_11_1",
      "s_9_1": "#!alpha(8)!#*(#!Temp(8)!#-#!refTemp(8)!#)*10^-
6*fuelregionheight+fuelregionheight + s_10_1",
      "s_8_1": "#!alpha(7)!#*(#!Temp(7)!#-#!refTemp(7)!#)*10^-
6*fuelregionheight+fuelregionheight + s_9_1",
      "s_7_1": "#!alpha(6)!#*(#!Temp(6)!#-#!refTemp(6)!#)*10^-
6*fuelregionheight+fuelregionheight + s_8_1",
      "s_6_1": "#!alpha(5)!#*(#!Temp(5)!#-#!refTemp(5)!#)*10^-
6*fuelregionheight+fuelregionheight + s_7_1",
      "s_5_1": "#!alpha(4)!#*(#!Temp(4)!#-#!refTemp(4)!#)*10^-
6*fuelregionheight+fuelregionheight + s_6_1",
      "s_4_1": "#!alpha(3)!#*(#!Temp(3)!#-#!refTemp(3)!#)*10^-
6*fuelregionheight+fuelregionheight + s_5_1",
      "s_3_1": "#!alpha(2)!#*(#!Temp(2)!#-#!refTemp(2)!#)*10^-
6*fuelregionheight+fuelregionheight + s_4_1",
      "s_2_1": "#!alpha(1)!#*(#!Temp(1)!#-#!refTemp(1)!#)*10^-
6*fuelregionheight+fuelregionheight + s_3_1",
      "s_100_1": "sqrt(newvol1/(pi*(s_2_1-s_3_1)))",
      "s_101_1": "sqrt(newvol2/(pi*(s_3_1-s_4_1)))",
      "s_102_1": "sqrt(newvol3/(pi*(s_4_1-s_5_1)))",
      "s_103_1": "sqrt(newvol4/(pi*(s_5_1-s_6_1)))",
      "s_104_1": "sqrt(newvol5/(pi*(s_6_1-s_7_1)))",
      "s_105_1": "sqrt(newvol6/(pi*(s_7_1-s_8_1)))",
      "s_106_1": "sqrt(newvol7/(pi*(s_8_1-s_9_1)))",
      "s_107_1": "sqrt(newvol8/(pi*(s_9_1-s_10_1)))",
      "s_108_1": "sqrt(newvol9/(pi*(s_10_1-s_11_1)))",
      "s_109_1": "sqrt(newvol10/(pi*(s_11_1-s_12_1)))"
}
}
```

Notes/observations:

•We first defined the original region's volume (here all fuel regions have equal volumes initially in this example).

•We then calculated the change in volume due to thermal expansion using the $\Delta V = \beta V \Delta T$ formula where $\beta \approx 3\alpha$.

•Note that alpha is specified in units of 10^{-6}/K.

•Notice that since each region can have a different temperature, we specified a different equation for each region with thermal expansion coefficient evaluated at average region temperature (MCNP Cell ID is indicated in the brackets inside the variable identifier). We also used the ΔT specific to each cell.

•We then calculated new volume for each region, updated the surfaces along the vertical direction of the fuel rod, and used the calculated new volume to correct the radius.

•Anubis allows you to access other parameters such as maximum temperature if necessary for your surface expansion model.

•Notice that division operators (slashes) are not escaped in the equations.

•The default density correction in Anubis uses the ΔV = βVΔT formula (see utilities\mapTempFields.m). The calculated density correction for any region can be accessed using the #!densityCorrection(cellID)!# reserved variable.

•In the concise version, we made use of the reserved variable #!densityCorrection(cellID)!# to make the equations more concise since the expansion model here is consistent with the default model.

The volumes.json *input file:*

```
{
	"c1": "pi*s_100_1^2*(s_2_1-s_3_1)",
	"c2": "pi*s_100_1^2*(s_3_1-s_4_1)",
	"c3": "pi*s_100_1^2*(s_4_1-s_5_1)",
	"c4": "pi*s_100_1^2*(s_5_1-s_6_1)",
	"c5": "pi*s_100_1^2*(s_6_1-s_7_1)",
	"c6": "pi*s_100_1^2*(s_7_1-s_8_1)",
	"c7": "pi*s_100_1^2*(s_8_1-s_9_1)",
	"c8": "pi*s_100_1^2*(s_9_1-s_10_1)",
	"c9": "pi*s_100_1^2*(s_10_1-s_11_1)",
	"c10": "pi*s_100_1^2*(s_11_1-s_12_1)"
}
```

Notes/observations:

•The equations indicate cell volumes as a function of surface parameters.

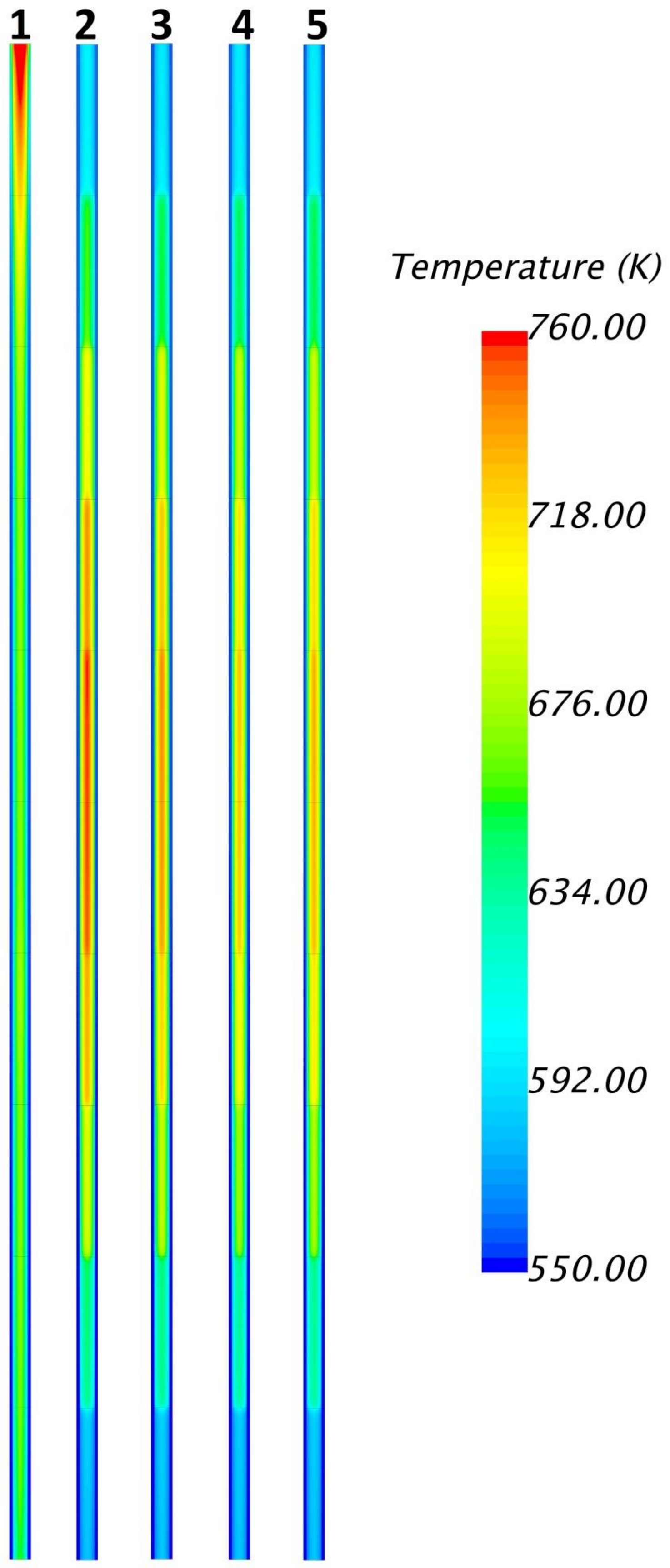
•Anubis will use the updated surface parameters from most recent iteration.

•The calculated volumes are used in tally unnormalization to obtain power profile from energy deposition dose profile in MCNP.

•If volumes.json is not specified and MCNP is used as the volumes option, Anubis will use masses calculated by MCNP itself in the unnormalization.

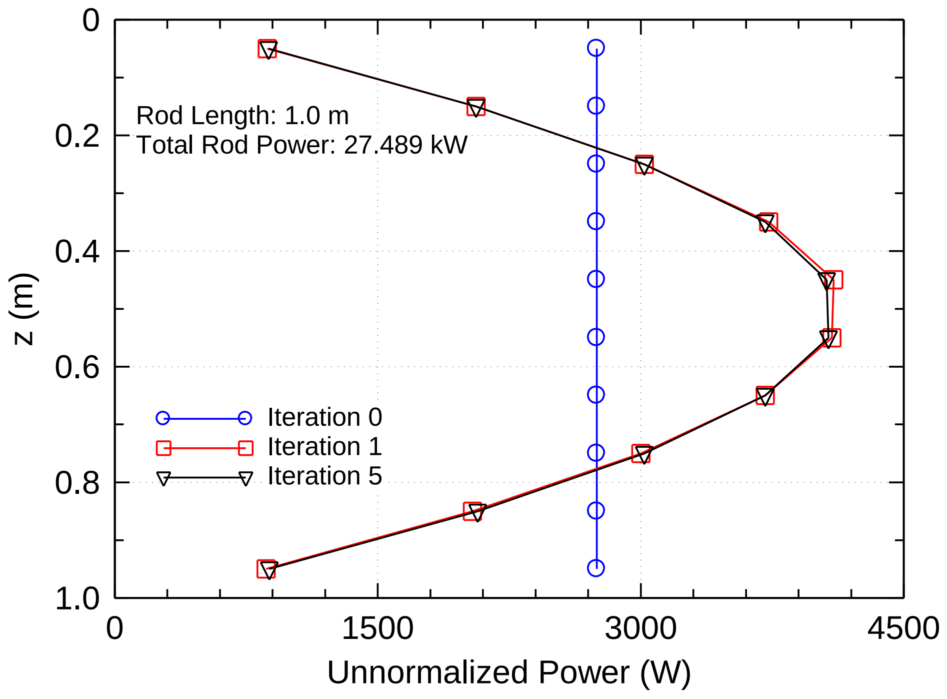
Simulation results:

The following figures show the results of the coupled simulation for this example. Each CFD run consisted for 2500 iterations with all fields from each CFD run transferred to that of the next Anubis iteration (for a total of 5 CFD runs). The temperature distribution after each Anubis iteration is compared in the following figure.



Temperature distribution in the fuel and coolant at different Anubis iterations.

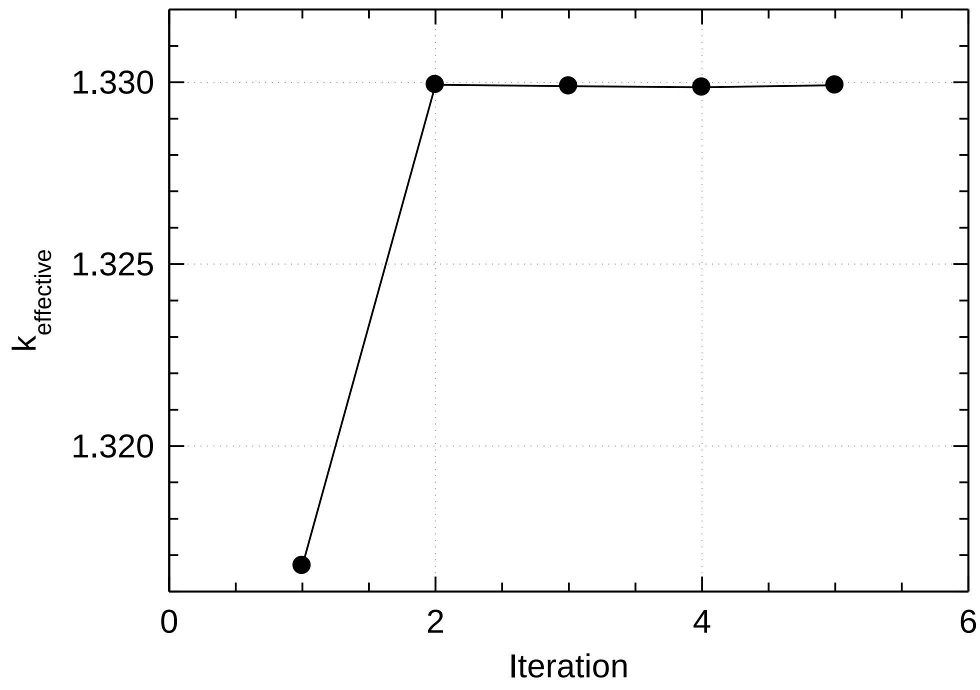
It is observed that temperature peaks at the top center of the fuel rod in the first Anubis run. This is consistent with expectations as power distribution is assumed to be uniform in the first iteration and as the coolant heats up as it flows upwards which decreases its ability to remove heat. In the second iteration which followed the neutronics solution, power peaked upwards of the center consistent (qualitatively) with expectations from the neutron energy deposition profile shown below and heat transfer. Convergence of temperature field is reached within 2% in the 5th iteration (which is very similar to 4th). This rapid convergence is because power distribution in this case is not strongly sensitive to heat transfer conditions. The k-effective of the system converged after only 2-3 iterations as shown in the next figure. The number of iterations necessary for convergence is expected to vary from system to system depending on power generation rates, system size, and boundary conditions.



Power distribution in the fuel rod at different iterations. Iteration 0 represents the initial power distribution used in the first CFD iteration.

In this example, the results qualitatively agreed with expectations and by inspecting the data it is verified that Anubis followed the user-specified equations for surface expansion, volumes, and density correction and correctly parsed and updated the input files with proper cell-region mapping.

Anubis does not introduce new physics or math. Anubis is effectively a robust data transfer platform that replaces tedious manual labor in coupling of MCNP with CFD codes which has long followed a similar scheme. Physical accuracy of the solution of this example, on the other hand, largely depends on the CFD models employed, cross-section data and accuracy of material properties used, statistical uncertainties of the Monte Carlo solution, and, in part, applicability of the coupling scheme described herein to the problem.



The k-effective of the reflected system at different iterations.

It should be noted that the loose steady state coupling scheme implemented in Anubis is suitable for long timescale problems where events in transient timescales of the neutronics and thermal hydraulics are not of interest. It is necessary to assess the applicability of the steady state coupling employed herein to the problem that you are modeling first. The motivation for the development of the Anubis code is for use in the prediction of flow accelerated corrosion in lead cooled reactor systems where temperature, velocity distribution, and shear stresses are required inputs. Corrosion is a very long timescale problem with a timescale in the order of thousands of hours. This is much longer than the timescales in neutronics and thermal hydraulics which allows for steady state coupling of neutronics and thermal hydraulics (but necessitates additional coupling with burn up and mass transfer). If you are studying reactor transients (e.g. due to rod insertion), it is likely that you will need transient coupling instead of steady state coupling of the neutronics and thermal hydraulics. Anubis also should not be used to study problems with liquid fuel where prompt power distribution is different from effective power distribution after delayed neutrons.