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

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

Developer's Guide

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

Purpose of the document: This document provides brief descriptions of the functions in the Anubis code. It is intended for developers who wish to integrate new capabilities or improve existing functionality and to users who wish to develop a lower-level understanding of how the code operates. It is recommended that you read this document in conjunction with the source code. Anubis uses an object-oriented structure to minimize number of parameters. It is also semi-modular allowing for additional thermal hydraulics and computational fluid dynamics (CFD) packages to be integrated and removed without affecting existing functionality. The Monte Carlo neutronics part, however, is not modular. The core of the code is developed with MCNP in mind.

Anubis Functions

Main folder

Anubis.m: Calls CFD iterator functions based on application specified in case.json, passes directory argument to iterator, and type of run (new or resume).

code/utilities

readCaseDef.m: Reads case.json input file into a structure array.

readGeometryDef.m: Reads geometry.json input file into a structure array.

readMaterialsDB.m: Reads materialsDB.json input file into a structure array.

readSurfaceExpansionDef.m: Reads surfaceExpansion.json input file into a structure array.

checkForTempConvergence.m: Compares the region average temperatures to those in the last n iterations (as specified in case.json) and per user choice it uses either percentage change or absolute change to make a decision whether the field is converged based on a user-specified threshold.

getThermalExpCoeff.m: Evaluates the thermal expansion coefficient at an input temperature for a material with an equation specified in materialsDB.json.

getPossibleSurfaceVarNames.m: Returns a list of valid surface parameter names that could be called in surfaceExpansion.json based on parsed surface cards defined in MCNP input.

getParamsFromSurfaceName.m: Extracts surface ID and parameter number from a given surface parameter name.

resolveSurfaceExpVars.m: Evaluates expressions specified under the variables object and then evaluates a statement for a surface object.

updateSurfacesObject.m: Calls resolveSurfaceExpVars.m function for each surface equation specified and updates surfaces object during each iteration to enable sequential evaluation of surface

expansion equations described in surfaceExpansion.json. The output is a surfaces object with updated surface parameters.

mapTempFields.m: Reads cells object, geometry.json structured array, materialsDB, maps CFD regions and MCNP cells, and calculates new cell densities based on equation specified in case.json or default equation.

updateCellsObject.m: Updates cells object based on output from mapTempFields.m.

getPowerFromCells.m: Reads the absolute power for a particular CFD region from cells object. Power is written to cells object using the addTallyDataToCells.m function under code/Anubis_MCNP.

getCellValue.m: Gets parameter value for a particular cell from mapTempField object.

code/remote

initiateRemoteDir.m: Creates the remote directory specified in case.json initially.

preparePBSFile.m: Creates a PBS file based on the specified PBS template and case parameters.

uploadToHost.m: Uploads the last iteration directory with updated inputs from local output directory to the remote working folder.

uploadXSToHostMCNP.m: Uploads the updated xsdir index and libraries generated by MAKXSF locally to remote MCNP data directory.

submitPBSToHost.m: Transfers PBS script to remote host and submits the job.

checkJobStatus.m: Checks the status of a remote PBS job that Anubis submitted until job no longer exists (i.e. complete). The time between each check depends on the status (whether the job is queued or running) and is based on user input in case.json.

downloadFromHost.m: Downloads the folder for the last iteration to local output directory.

cleanUpHost.m: Deletes the folder for last iteration on the remote working directory and all files inside it.

code/Anubis_MCNP

cloneCaseMCNP.m: Prepares input for a new Anubis iteration by locally cloning initial MCNP case which is then modified by other functions.

getLineDelimiterMCNP.m: Identifies the line delimiter used in MCNP input (i.e. whether it is \r\n or just \n).

getCellsMCNP.m: Parses MCNP input to extract parameters from cell cards and insert them into cells object.

getParamFromCard.m: Used in the parsing of cell cards to extract parameters.

getSurfacesMCNP.m: Parses surface cards in MCNP and extracts parameters of interest into a structure.

isNewCardMCNP.m: Used in the parsing of MCNP input to check whether a new line represents a new card or a continuation of a previous card.

issurfacemnemonic.m: Checks if a given string parsed from surface cards is a recognized surface mnemonic.

getExtMCNP.m: Generates ZAID extension for a Doppler broadened cross-section based on temperature.

getInterpolationExtensions.m: Returns the lower and upper ZAID extensions to be specified in specs file based on temperature.

getIsotopesInpMCNP.m: Parses material cards in MCNP and extracts parameters of interest into a structure.

checkXSMCNP.m: Checks if cross-section specified or used in interpolation is present in xsdir file. If not found, it will look for replacements based on user's pre-defined choices on what constitutes an acceptable replacement.

createSpecsFile.m: Generates the specs input file for MAKXSF for a set of nuclides at updated temperatures.

runMakxsf.m: Runs MCNP's MAKXSF utility locally and backs up previous xsdir.

updateXSdirMCNP.m: Updates xsdir index with references to libraries generated by MAKXSF for Doppler broadened cross-sections.

executeMCNP.m: Executes MCNP locally on either Windows or Linux.

readTallies.m: Parses MCNP output and reads F6 and F7 MCNP tallies into an object.

getCellMassMapMCNP.m: Used in readTallies.m to extract cell mass data from MCNP tally tables.

getCellTallyMapMCNP.m: Used in readTallies.m to extract cell data from MCNP tally tables.

addTallyDataToCells.m: Calculates unnormalized power for each cell in MCNP input and adds the power to the cells object. It also processes the volumes.json file if that option is used for volume specification.

updateIsotopes.m: Updates the isotopes object with new temperatures, cross-section extensions, and other parameters.

updateCellCardsMCNP.m: Updates cell cards in MCNP with new parameters.

updateSurfaceCardsMCNP.m: Updates the surface parameters in the MCNP cell cards following thermal expansion.

updateMaterialCardsMCNP.m: Updates the ZAIDs in materials cards following Doppler broadening.

code/Anubis_FOAM

FOAM_MCNP_Iterator.m: Controls the coupling of MCNP and OpenFOAM and saves after each iteration.

cloneCaseFOAM.m: Prepares input for a new Anubis iteration by locally cloning initial OpenFOAM case which is then modified by other functions to update power distribution.

runMapFieldsFOAM.m: Runs the mapFields utility which transfers fields from one simulation to another. It is used to map fields from one Anubis iteration to another involving OpenFOAM runs.

updateHeatSourcesFOAM.m: Updates the power distribution in new OpenFOAM run through modifying fvOptions files for all coupled regions. It also handles power distribution to sibling regions (refer to user's manual for more details).

readControlDictFOAM.m: Reads the controlDict file under the system folder in the OpenFOAM case into a structure.

executeFOAM.m: Executes the OpenFOAM solver specified in controlDict for local runs. It also runs decomposePar and reconstructPar for parallel runs.

getFolderNames.m: Returns list of folders in a specified directory while excluding default OpenFOAM setup folders.

getTimeStepsFOAM.m: Used in reading OpenFOAM output. It identifies last step that ran.

getStartLineInternalFieldFoam.m: Used in parsing OpenFOAM fields. It identifies the line at which field arrays are specified beyond a set point in the file (to allow for parsing of multiple arrays within one file).

removeCommentLinesFOAM.m: Removes comment lines from OpenFOAM files.

getTempFieldAverageFoam.m: Calculates the average temperature value in an OpenFOAM output file for a particular region (can be applied to other parameters).

getTempFieldMaxFoam.m: Calculates the maximum temperature value in an OpenFOAM output file for a particular region (can be applied to other parameters).

readResultFOAM.m: Gets parameters from processed OpenFOAM output (here only temperature is needed).

writeRegionTempsFOAM.m: It writes a text file with region average temperatures.

code/Anubis_CCM

CCM_MCNP_Iterator.m: Controls the coupling of MCNP and STAR-CCM+ and saves after each iteration.

cloneCaseCCM.m: Prepares input for a new Anubis iteration by locally cloning either initial STAR-CCM+ case if mapFieldsBetweenIterations is disabled or the latest case if mapFieldsBetweenIterations is enabled.

updatePowerCCM.m: Generates the JAVA macro to update power distribution in heat generating regions, control the run, and extract and export the temperature field.

executeCCM.m: Executes STAR-CCM+ case with Anubis-generated JAVA macro in local runs on Windows and Linux.

readTempCCM.m: Reads the temperature field exported from STAR-CCM+.

getTempFieldAverageCCM.m: Calculates average temperature in a particular STAR-CCM+ region.

getTempFieldMaxCCM.m: Calculates maximum temperature in a particular STAR-CCM+ region.

writeRegionTempsCCM.m: Writes average temperature values to a file.