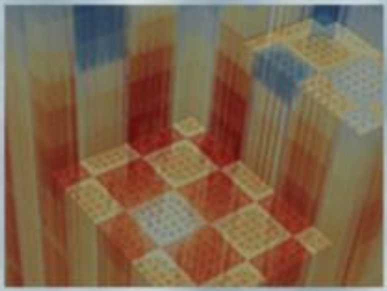
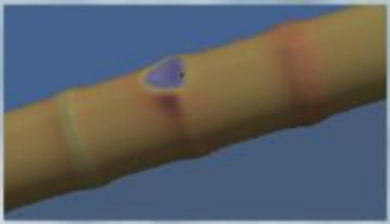
CASL
A DOE Energy Innovation Hub

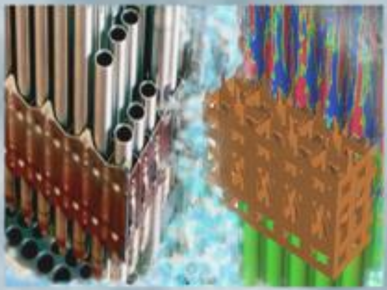
SAND2017-12545R
CASL-X-2017-1421-000





STAR-CCM+ (CFD) Calculations and Validation

L3:VVI.H2L.P15.02



Lindsay Gilkey, Sandia National Laboratories

September 5, 2017

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0		All	Initial Release

Document pages that are:

Export Controlled ____

IP/Proprietary/NDA Controlled____

Sensitive Controlled____

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Requested Distribution:

To:

Copy:

EXECUTIVE SUMMARY

This milestone presents a demonstration of the High-to-Low (Hi2Lo) process in the VVI focus area. Validation and additional calculations with the commercial computational fluid dynamics code, STAR-CCM+, were performed using a 5x5 fuel assembly with non-mixing geometry and spacer grids. This geometry was based on the benchmark experiment provided by Westinghouse. Results from the simulations were compared to existing experimental data and to the subchannel thermal-hydraulics code COBRA-TF (CTF). An uncertainty quantification (UQ) process was developed for the STAR-CCM+ model and results of the STAR UQ were communicated to CTF. Results from STAR-CCM+ simulations were used as experimental design points in CTF to calibrate the mixing parameter β and compared to results obtained using experimental data points. This demonstrated that CTF's β parameter can be calibrated to match existing experimental data more closely. The Hi2Lo process for the STAR-CCM+/CTF code coupling was documented in this milestone and closely linked L3:VVI.H2LP15.01 milestone report.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AMA	Advanced Modeling Applications
CASL	Consortium for Advanced Simulation of Light Water Reactors
CHF	Critical Heat Flux
CILC	CRUD-induced localized corrosion
CIPS	CRUD-induced power shift
CFD	computational fluid dynamics
CP	Challenge Problem
CRUD	corrosion-related unidentified deposits or Chalk River unidentified deposits
CTF	COBRA-TF subchannel thermal-hydraulics code
DOE	US Department of Energy
FA	Focus Area
Hi2Lo	High-to-Low
INL	Idaho National Laboratory
KOM	Standard k-ω turbulence model
LANL	Los Alamos National Laboratory
NCSU	North Carolina State University
NIST	National Institute of Standards and Technology
NMV	No-Mixing Vane
ORNL	Oak Ridge National Laboratory
PHI	Physics Integration
RANS	Reynolds-Averaged-Navier-Stokes
RKE	Realizable k-ε turbulence model
SKE	Standard k-ε turbulence model
SNL	Sandia National Laboratories
STAR	STAR-CCM+ CFD code
UQ	uncertainty quantification
V&V	verification and validation
VMA	Validation and Modeling Applications
VUQ	Validation and Uncertainty Quantification
VVI	Verification and Validation Implimentation
VVUQ	Verification, Validation and Uncertainty Quantification
WEC	Westinghouse Electric Company

NOMENCLATURE

A_{total}	Total Area
$AFLUX$	Average Linear Heat Rate Per Rod
β	CTF Beta Coefficient
c_p	Specific Heat
CS	Control Surface
CV	Control Volume
$\Delta\dot{E}$	Change in Power
$\dot{E}_{in}$	Inlet Power
$\dot{E}_{out}$	Outlet Power
e	Internal Energy
h	Specific Enthalpy
HF	Heat Flux
i, j, k	index notation, usually subchannel number
k	Thermal Conductivity
$\Delta\dot{K}E$	Change in Rate of Kinetic Energy
L	Bundle Length
$\dot{m}$	Mass Flow Rate
μ	Dynamic Viscosity
N_{rods}	Number of rods
$\hat{n}$	Normal Vector
P	Pressure
$\Delta\dot{P}E$	Change in Rate of Potential Energy
PF	Normalized Power Factor
$Power_{total}$	Total Power
$\dot{Q}_{in}$	Rod Power
ρ	Density
r	Rod Radius
T	Temperature
$\bar{T}$	Average Temperature
T_0	Reference Temperature
T_{exp}	Experiment Temperature
T_{model}	Simulation Temperature
$\Delta\dot{U}$	Change in Rate of Internal Energy
u, v, w	Velocity in x, y, z
$\overrightarrow{u_{rel}}$	Relative Velocity
V	Volume
$\Delta\dot{W}$	Change in Work Rate
w	Cross-Section Width

1. MILESTONE DESCRIPTION

High-fidelity computational fluid dynamics (CFD) codes require significant resources and long run times, which can make these calculations very computationally expensive. At national laboratories, these simulations are typically run on super computing platforms, using thousands of core hours to complete the computations. This upfront computational cost can make detailed CFD analysis difficult or out of reach for many analysts in industry. Lower-fidelity thermal-hydraulics codes have a much lower computational cost than traditional CFD as well as significantly shorter run times. High-fidelity codes that have been assessed with experimental data are generally more predictive than lower-fidelity codes. These high-fidelity codes can be used to calibrate and validate these lower-fidelity codes. A demonstration of this using the High-to-Low process is described below in the context of the two codes, STAR-CCM+ (STAR) and COBRA-TF (CTF) and is the focus of this milestone report.

1.1 Description of the Hi2Lo Process and Methology

High-to-low (Hi2Lo) is the process of using a validated higher fidelity code to generate synthetic data that improves and informs a lower-fidelity code. Collecting experimental data at a resolution or parameter space needed to validate models is often impractical due to time, cost, or the limitations of the equipment/geometry. Synthetic data generated with a high-fidelity code can supply information to computationalists that would otherwise be unavailable from experiments alone. The synthetic data tends to have error and uncertainties quantified and can be used in combination with available experimental data to calibrate lower-fidelity fidelity codes. In the context of this milestone, STAR-CCM+ is the high-fidelity code, relative to COBRA-TF, which is the lower-fidelity code.

This milestone presents a demonstration of the Hi2Lo process in the VVI focus area. This process is reproducible and can be used for different code couplings based on the steps outlined in Section 1.1.1. The work done with STAR as the high-fidelity code is the focus of this milestone report. Natalie Gordon's work on the companion L3:VVI.H2L.P15.01 milestone, which details work done with CTF for Hi2Lo, [1] is briefly described whenever relevant to provide context to the L3:VVI.H2L.P15.02 work. The statistical and VVUQ components of the Hi2Lo framework were performed using Dakota version 6.6. Experimental data from Westinghouse Electric Company (WEC) was used for code validation and to set code parameter spaces [2].

This report contains the following technical content. The experimental data and the geometry is described in Section 2. Section 3 summarizes the STAR and the CTF model configurations, model assumptions, and boundary conditions used during simulation. Section 4 describes early steps during the milestone that were performed to ensure that the codes were behaving as anticipated and that STAR and CTF were suitable for coupling during the Hi2Lo process. The quantitative validation of the STAR and CTF models using the experimental data is presented in Section 5. Section 5 also includes a comparison of the results from both codes before any calibration was performed on the CTF model. Section 6 describes uncertainty quantification that was performed with STAR and the tools that were built to implement the UQ are described in detail. Sections 7 and 8 summarize parallel activities that were performed with CTF, such as calibration studies and the experimental design process. The report concludes with Sections 9 and 10, which describe the final results from CTF and summarizes the insights and notable knowledge gained during the milestone progress.

1.1.1 Milestone Tasks and Implementation

The work to complete the L3:VVI.H2L.P15.01 (CTF) and L3:VVI.H2L.P15.02 (STAR) milestones was split into several steps outlined in Table 1. This milestone report addresses calculations and material relevant to STAR, such as the model, validation, and uncertainty quantification in STAR.

Table 1: Outline of Milestone Steps.

STAR		CTF[1]	
Report Section	**Step/Description**	**Report Section**	**Step/Description**
		2.2	0.1: Determine Experimental Domain and Separate Data into Validation and Calibration Sets.
		3.1.4	0.2: IAPWS IF97 tables in CTF to generate fluid property polynomials for STAR
4.1	0.1: Conservation Equation Check	4.1	0.3: Conservation Equation Check
4.2	0.2: Evaluate Cross Flow Magnitude	4.2	0.4: Evaluate Cross Flow Magnitude
5	1: Quantitative Validation (21 data points)	5	1: Quantitative Validation (10 data points)
6	2: Run Uncertainty Quantification on 1 Experimental Test	7	2: Construct Surrogate
		7	3: Bayesian Calibration with Surrogate (1 data point)
		7	4: Quantitative Validation (10 data points)
8	3: Evaluate Experimental Design Points.	8	5: Experimental Design with surrogate (1 data points + points from STAR)
		8	6: Calibrate with Experimental Design points and 11 data points
		9	7: Quantitative Validation (10 data points)
	4: Automate Process		8: Automate Process

[1] Work done for CTF is contained in [1].

1.2 Working Group and Ackknowledgements

This report summarizes work performed by Lindsay Gilkey and closely related milestone work by Natalie Gordon [1] of Sandia National Laboratories, as part of the VVI Focus Area of CASL. Vince Mousseau (SNL), Brian Williams (LANL), Ralph Smith (NCSU), Adam Hetzler (SNL), and Chris Jones (SNL), provided invaluable technical advice and support for the milestone work. Yixing Sung and Emre Tatli of Westinghouse provided experimental data and expertise that made this analysis possible. The Dakota Team, which included Brian Adams (SNL), Laura Swiler (SNL), Adam Stephens (SNL), and Kathryn Maupin (SNL), provided support while performing calculations using Dakota version 6.6. All STAR simulations were performed using the HPC resources on Falcon at INL.

A technical review of this report was performed by Adam Hetzler and Vince Mousseau.

2. EXPERIMENTAL DATA

Westinghouse provided single-phase non-mixing vane (NMV) test data from experiments performed on a 5x5 set of electrically heated rods [2]. Section 2 gives an overview of the experimental geometry and experimental data that was provided.

2.1 Experimental Configuration

The experimental geometry and the test condition ranges are described in the following sections.

2.1.1 Experiment Geometry

The test bundle consisted of a 5x5 rod bundle of electrically heated rods. The bundle incorporated five grid spacers without mixing vanes along the heated length and one additional grid spacer upstream to precondition the incoming flow. The grid spacers include geometric features such as springs but do not include mixing vanes. Figure 2-1 shows the axial length of the experimental geometry, as well as the locations of the grid spacers.

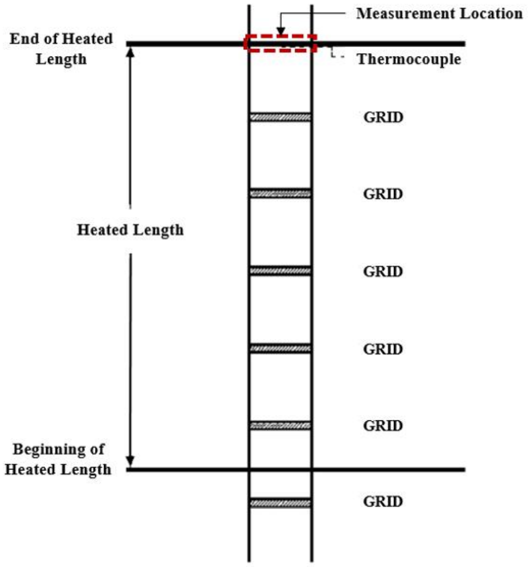


Figure 2-1: Axial view of the experimental geometry. Figure adapted from [3].

Six of the twenty-five rods are “hot,” which means they have a higher power than the remaining nineteen “cold” rods. The exit cross section, with the hot rods indicated, is shown in Figure 2-2. Thermocouples were placed at the center of each of the thirty-six subchannels at the measurement location indicated in Figure 2-1 to collect time-averaged temperature data. The heated section is 3 meters in length and the cross-section width is approximately 7 cm.

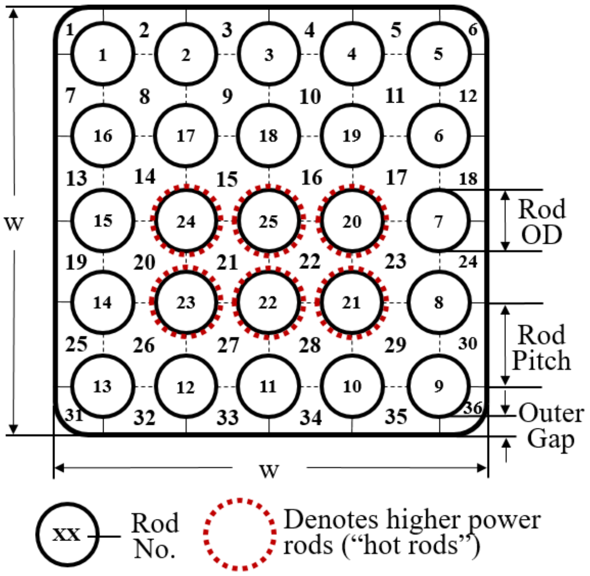


Figure 2-2: 5x5 exit cross section with rod, subchannel numbering, and hot rods indicated. Figure adapted from [3].

2.1.2 Experiment Test Conditions

The ranges of the experimental test conditions are given in Table 2. In total, there are twenty-three NMV tests. Case 1, which will be referred to throughout the report, corresponds to one of the experimental test conditions.

Table 2: Experimental Test Condition Ranges for the NMV Data.

	Test Section Exit Pressure (bars)	Test Section Inlet Temperature (°C)	Mass Velocity (kg/m^{2}s)	Test Section Power (MW)
Min Value	101.333	213.031	2431.932	0.713
Max Value	164.765	312.441	3730.37	2.441

2.2 Experimental Data

The outlet temperature measurements of the thirty-six subchannels was reported for each of the twenty-three tests. WEC reported that there was a ±6 ^{o}F (3.333 ^{o}C) repeatability error on the experimental data in addition to some uncertain amount of experimental error.

There was some concern expressed early on by WEC that the thermocouple array may have shifted during testing or that the thermocouples had become uncalibrated. This concern was raised as the experimental exit temperatures were asymmetric (Figure 2-3), which was not the anticipated shape of the temperature data.

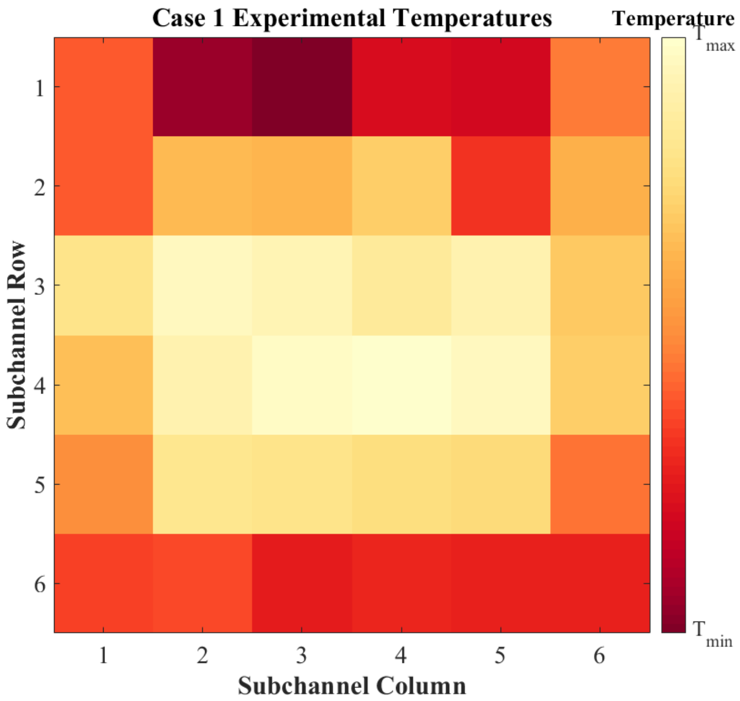


Figure 2-3: Case 1 temperatures represented as a contour plot. $T_{max} - T_{min}$~15 °C.

If the thermocouple array or the heated rods had shifted in the channels during testing, this would be the most pronounced in the outer subchannels as shown in Figure 2-4. Additionally, the experiments were not performed continuously, with other test types (critical heat flux) being performed on the bundle between testing phases for the non-mixing data. The critical heat flux experiments could result in damaged or incorrectly calibrated thermocouples during the NMV tests, therefore there is lower confidence in the accuracy of data collected during later tests, which were performed after the critical heat flux experiments.

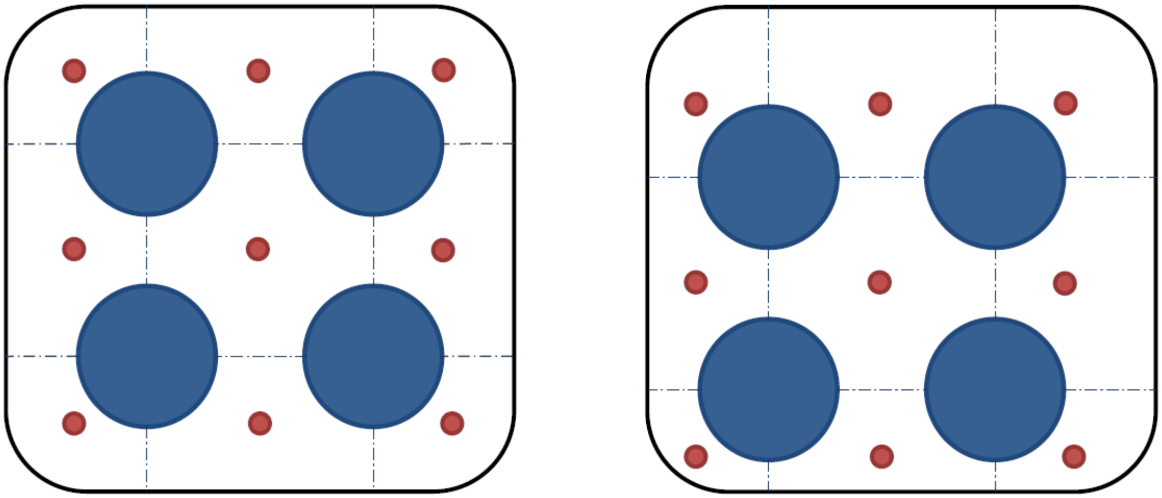


Figure 2-4: Representation of potential thermocouple and rod shift. The red dots indicate thermocouple locations. The shift has been exaggerated for illustrative purposes.

When evaluating the quality of the experimental data, an energy balance was performed for Case 1. This was done by using the total mass flow rate and the average quantities at the outlet. To calculate average quantities, it was assumed at the amount of mass and energy in each channel was proportional to the area of the channel. For example, the average outlet temperature was calculated from the exit temperatures by weighting the temperatures by the individual subchannel area.

$$\overline{T} = \frac{1}{A_{total}} \sum_{i=1}^{36} T_i A_i$$

At the inlet, total mass flow rate and average inlet temperature were given as part of the experimental configuration data. Temperature dependent quantities were found in the NIST fluid property tables [4]. Pressure at the inlet was estimated using the experimental pressure drop and the given exit pressure. The following equation was used to estimate the energy at the inlet (a detailed explanation of the terms of the energy equation is in Section 4.1.2):

$$\dot{E}_{in} = \dot{m}\left(h(\overline{T}) - \frac{P_{in}}{\rho(\overline{T})}\right) + Q_{in}$$

At the outlet, total mass flow rate (constant value), exit pressure, and the temperature per subchannel are known quantities. The area averaged temperature was used to evaluate temperature dependent quantities.

$$\dot{E}_{out} = \dot{m}\left(h(\overline{T}) - \frac{P_{out}}{\rho(\overline{T})}\right)$$

Conservation of energy in a closed system requires $\dot{E}_{in} - \dot{E}_{out} = 0$. The experiment shows a 1.4% decrease of energy from the inlet to the outlet. Several simplifying approximations were made during the calculation, such as assuming the average channel temperature can be approximated with channel center temperatures and the mass/internal energy distribution at the outlet is proportional to the subchannel area. Increasing or decreasing the pressure drop had minimal impact on the energy balance as the change in internal energy in the form of heat is the most significant term. This demonstrated that while there are experimental concerns, it can be assumed that the experiment is not losing a significant amount energy as a result of a major experimental or systematic error. Severely damaged thermocouples would result in a large imbalance in the energy equation as the measured temperatures (and associated internal energy) would likely be very far from their actual values.

Two test points were removed from the experimental data, which is discussed in [1]. These points were eliminated as they were at a much lower pressure than the remaining test suite, which had complicating effects at temperatures nearing saturation.

STAR used all remaining twenty-one data points for validation calculations. For CTF calculations, the data was split into a validation and a calibration set. This was done to check for improvement in the model predictions against temperature measurements that were not included during calibration. An effort was made to distribute the individual tests between the CTF validation and calibration sets such that both sets included the full range of input experiment test conditions, as listed in Table 2 [1]. The experimental data was split into validation and calibration data sets as seen in Table 3.

Table 3: Experimental Data Divided into Validation and Calibration Data Sets.

Test #	STAR[1]	CTF	
	Validation	Validation	Calibration
9	X		X
10	X	X	
11	X	X	
12	X		X

Test #	STAR[1]	CTF	
	Validation	Validation	Calibration
13	X		X
14	X	X	
15	X	X	
16	X		X
17	X		X
18	X	X	
19	X	X	
20	X		X
21	X		X
22	X	X	
23	X	X	
24	X		X
25	X		X
114	X	X	
115	X		X
116	X	X	
117	X		X

[1] STAR uses all experimental data during validation.

3. MODEL DESCRIPTION AND CONFIGURATION

The configuration and boundary conditions of the STAR and CTF models where matched as closely as possible as the Hi2Lo process between codes requires identical parameters and nearly equivalent model configurations.

3.1 STAR-CCM+ Model Configuration

The STAR mesh was obtained from WEC. It contains approximately sixty million cells and incorporates the full heated length of the 5x5 assembly, including all grid features, as shown by Figure 3-1. Figure 3-2 shows a cross-section view of the mesh at the outlet, normal to the flow direction. The mesh uses a base cell size of 0.6 mm and includes a prism layer to improve modeling in the boundary layer.

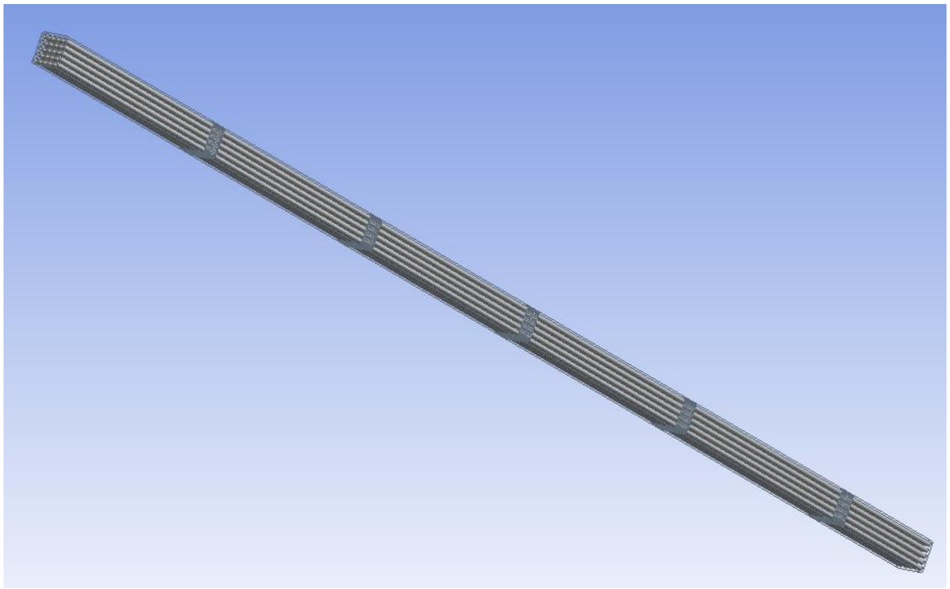


Figure 3-1: STAR-CCM+ simulation full geometry of the 5x5 rod bundle with grid spacers.

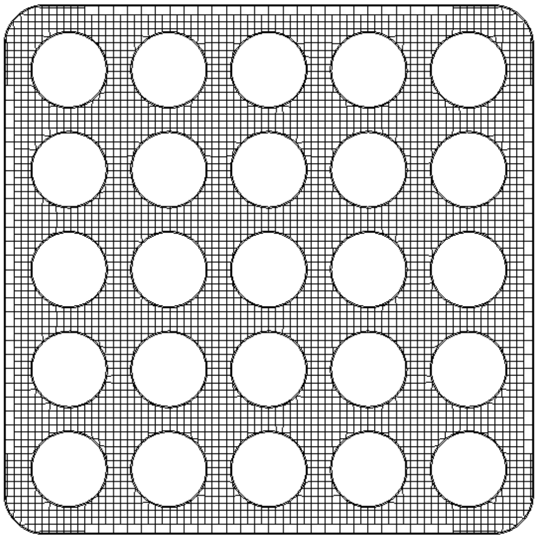


Figure 3-2: Cross sectional view of the mesh at the outlet.

Grid convergence index studies for fuel bundles are difficult to perform due to the size of the meshes required and the very different physical length scales that occur in the simulation. To capture the smallest features in the simulation, this would require a very small first cell size (and y+ value), which would imply potentially hundreds of millions of cells in the simulation. In contrast, the largest flow features can be captured with a much coarser mesh. For the current NMV mesh configuration, the L_2norm was used to quantify the closeness of the simulation data to the experiment and was defined as:

$$L_2norm = \frac{\sqrt{\sum_{i=1}^{36}\left(T_{model,i} - T_{exp,i}\right)^2}}{\sqrt{\sum_{i=1}^{36}\left(T_{exp,i}\right)^2}} \times 100$$

where i corresponds to the subchannel number. The current mesh yields results that are less than 2% difference of the experimental temperatures and is assumed to be satisfactorily mesh converged for the current Hi2Lo application (which can potentially require dozens or hundreds of simulations). The mesh will be revisited if application needs or computational limits change. Aaron Krueger [5] is working on a solution verification method to utilize imperfect meshes better than current methods.

The STAR model is single-phase and density in the fluid region is determined using a polynomial function of temperature at isobaric conditions (detailed in Section 3.1.4). The model includes only the fluid region and has the gravity physics model enabled.

3.1.1 STAR-CCM+ Model Boundary Conditions

The boundary conditions used in the STAR-CCM+ model are listed below.

- Inlet: Mass flow inlet. Total mass flow rate and temperature of the incoming fluid is specified as inlet parameters. STAR uses this information (in additional to fluid properties) to estimate a velocity field at the inlet. Total mass flow rate can be calculated using mass velocity (experimental test data) and cross sectional area.

$$\frac{kg}{s} = \frac{kg}{m^2 s} \times m^2$$

- Outlet: Pressure outlet. The negative pressure gradient accelerates the flow in the channel towards the outlet. The outlet pressure was set equal to the test section pressure (from the experimental data).
- Channel Walls and Grid Spacers: Adiabatic wall boundaries. No heat is transferred across these surfaces.
- Rods: Power source. Each of the rods has a total power specified. Heat flux of the rod is automatically calculated using the rod surface area and is assumed to be constant along the rod length. It is calculated from the quantity $AFLUX$, as defined below:

$$AFLUX = \frac{Power_{total}}{LN_{rods}}$$

$$Power_i = Power_{total} \times \frac{PF_i}{N_{rod}}$$

$$Power_i = AFLUX \times rLPF_i$$

where i corresponds to the rod number, N_{rod} is the total number of rods (25), r is the radius of the rods, L is the rod length (3 m), and PF is the normalized power factor as given by the test documentation [3].

3.1.2 STAR-CCM+ Turbulence Model

The simulation uses the steady-state 3D Reynolds-Averaged-Navier-Stokes (RANS) standard k-ω model. More detail on the standard k-ω model is given in Section 6.1.1.1. Standard k-ω was chosen during FY2016 after performing a sensitivity study with the mixing vane grid (MVG) mesh/data and different turbulence models. The results of the sensitivity analysis are in Figure 3-3. The turbulence models included are:

- Realizable k-ε Two-Layer with All-y^+ Wall Treatment (RKE_{2layer})
- Realizable k-ε with High-y^+ Wall Treatment (RKE)
- Standard Linear k-ε with High-y^+ Wall Treatment (SKE_{linear})
- Standard Quadratic k-ε with High-y^+ Wall Treatment ($SKE_{quadratic}$)
- Standard Cubic k-ε with High-y^+ Wall Treatment (SKE_{cubic})
- Standard k-ω (KOM)

These models were suggested by Bob Brewster of WEC and Sal Rodriguez of Sandia National Laboratories, who are both CFD experts and have experience modeling fuel bundles. These turbulence models were down selected out of all available STAR implemented turbulence models as being those they thought were most appropriate for the fuel bundle simulation. Standard k-ω was chosen for the MVG mesh and data as it showed the best agreement with the data and also was a relatively stable model when compared with RKE_{2layer}, which was originally selected for modeling. Realizable k-ε is commonly known among CFD analysts as the most popular turbulence model as it is able to simulate a wide range of geometries and flow conditions without running into major numerical errors or instabilities. Figure 3-3 clearly shows that there is uncertainty in the simulation results due to the turbulence model selection. Standard k-ω and Realizable k-ε were used in the STAR uncertainty quantification (UQ) in Section 6.

The sensitivity study was not repeated for the NMV data due to time limitations, however the NMV mesh geometry was identical to the MVG mesh at the outlet location (located far from the grid spacers) and the flows have similar Reynolds numbers. Therefore, the information learned with the MVG data/mesh could reasonably be applied to the NMV data/mesh. The surface average y+ value was between 115 and 150 on all surfaces for the current mesh. Ideally, the y+ value would be lower for standard k-ω, but this would require an approximate first cell size 0.04 mm, which would result in hundreds of millions of cells in the mesh. Because of this limitation, a larger y+ and an all y+ wall treatment was used for the turbulence model.

It should be noted that the most suitable turbulence model for any flow is highly dependent on the mesh configuration [6]. The most suitable turbulence model for a given simulation can vary as the mesh is refined or coarsened.

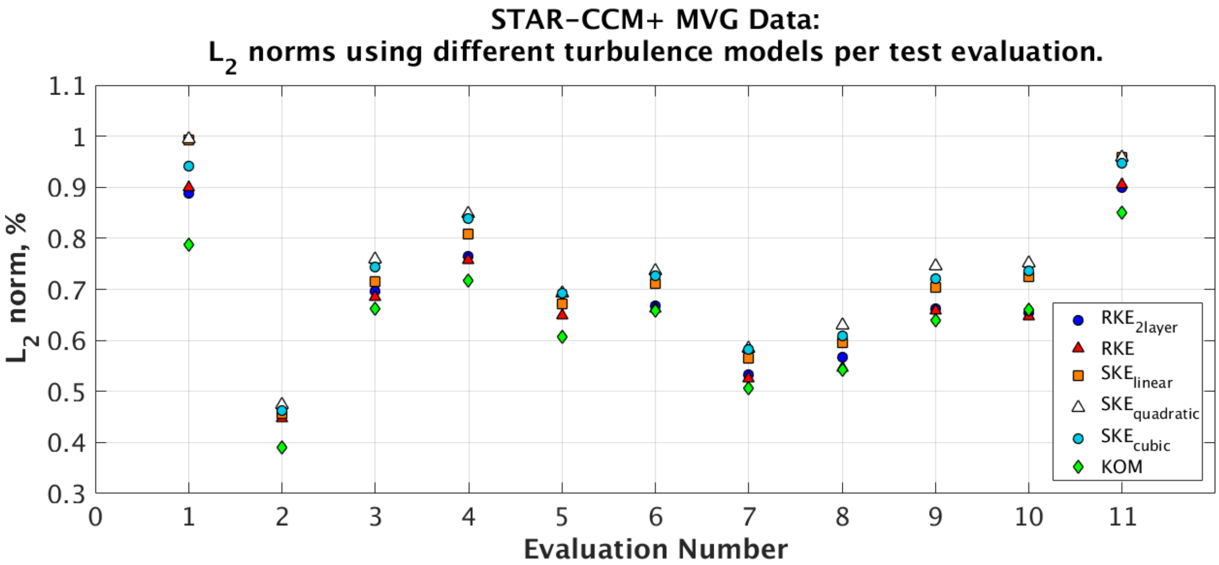


Figure 3-3: STAR-CCM+ sensitivity study with the MVG data from previous milestone [7]. KOM and RKE$_{2layer}$ were used during the STAR UQ study for Case 1 in Section 6.

3.1.3 STAR-CCM+ Model Response

The STAR simulation is configured to return the following quantities:

- Thirty-six temperature measurements at the outlet, measuring the temperature at the center of each channel. These temperature measurements approximate the experimental data and are collected with point probes.
- Thirty-six temperature measurements at the outlet, measuring the surface averaged temperature of each channel. The data is collected with a surfaced averaged report in STAR. This quantity is equal to $\frac{1}{A}\int T(x,y)dA$.

Simulation convergence was judged by monitoring residuals of momentum, turbulent kinetic energy, and mass continuity for asymptotic behavior and relative residual values below 1×10^{-3}.

3.1.4 STAR-CCM+ Fluid Properties

The fluid properties in STAR were implemented in a way to match CTF fluid properties as closely as possible. STAR is generally an incompressible code when modeling liquids. Efforts are being made to expand the compressible models for liquids (such as the IAPWS-IF97 tables for water), however these implementations can create simulation stability issues. STAR has built-in capabilities to use polynomial functions of temperature at constant pressure to set temperature-dependent fluid properties such as density, specific heat, thermal conductivity, and viscosity. CTF, which is a compressible code, can use built in IAPWS-IF97 tables to generate data for the experimental temperature and pressure ranges in Table 2. After post-processing, it is possible to make a single function for each fluid property as a function of temperature.

The procedure to make the following polynomial functions of temperature for the STAR fluid properties is reported in [1]. This section lists the polynomial fitted functions from CTF that are used with STAR. It was found during this step that fourth-order polynomials were the most appropriate to use for the fluid properties given the large temperature range of the experimental data. In the relevant temperature range, the interpolant behaves properly between data points.

Using the following polynomials in STAR ensures that STAR matches the fluid properties from CTF at a given temperature.

Table 4: Summary of Polynomials Used to Define Fluid Properties in STAR [1].

Fluid Properties	Polynomial Functions of Temperature (T)
Density	ρ = -5.408416E-07T^4+1.132377E-03T^3-8.923019E-01T^2+3.121676E+02T-3.990909E+04
Specific Heat	c_p = 4.277308E-05T^4-9.077996E-02T^3+7.223995E+01T^2-2.553508E+04T+3.386247E+06
Thermal Conductivity	**k** = -2.345491E-10T^4+4.936334E-07T^3-3.936989E-04T^2+1.398904E-01T-1.791301E+01
Dynamic Viscosity	μ = -3.398248E-14T^4+6.255825E-11T^3-4.080035E-08T^2+1.031840E-05T-5.335271E-04

3.1.5 STAR-CCM+ Simulation Time and Cost

The STAR simulations were run on Falcon at the Idaho National Laboratory. Each simulation used approximately 1000 cores for one hour. Simulation time is dependent on the stopping point (number of iterations) used. The simulation was terminated when the number of steps reached 3000. At 3000 steps, the simulation had reached steady state, which was judged by residuals of continuity, momentum, and energy and monitors for the average velocity and maximum (most sensitive monitor) outlet temperature. There were twenty-one validation tests, which resulted in the STAR simulations having a computational expense of 21,000 core hours. The simulation initial conditions such as initial velocity and temperature were derived from the boundary conditions specified in Section 3.1.1.

During future work with the STAR model, it may be possible to reduce the computational time by using an existing simulation's solution as a new simulation's initial state. This would be especially valuable if a more extensive UQ process was done with STAR.

3.2 CTF Model Configuration

The CTF model configuration is briefly described in this section. A more complete description can be found in [1].

CTF is a lower-fidelity subchannel that uses a very coarse mesh. Whereas the STAR simulations require approximately 1000 core hours, the CTF simulations take approximately five minutes on a single processor (0.08 core hours). The CTF model contains thirty-six subchannels and has an axial resolution of 2.54 cm (1 inch) over the heated length of the bundle. The five grids are incorporated into the CTF model with loss coefficients that are applied at the node locations that correspond to the experimental geometry's grid center locations. Loss coefficients vary based on the subchannel location and geometry (side, corners, center subchannel locations) in the bundle. The boundary conditions of the STAR simulation were chosen so that they would match the CTF boundary conditions as closely as possible. The boundary conditions for CTF are:

- Inlet: A total mass flow rate and average fluid temperature is specified.
- Outlet: Exit pressure is specified.
- Channel Walls and Grid Spacers: No heat is transferred across these surfaces. The grid spacers are incorporated in the CTF model by using loss coefficients.
- Rods: $AFLUX$ (defined in Section 3.1.1) and the radial power distribution (specified in the CTF input deck) are used to set the power per rod in the model. The model assumes that the heat flux is constant along the length of the rod.

The temperature results (and other reported quantities) are reported by CTF as subchannel averaged values with an axial cell length of 2.54 cm.

Some key assumption that are made with the CTF analysis are that the results are at steady-state and that no additional cross-flow effect modeling is needed. CTF is also a two-phase, compressible code, however since the flow parameters put the STAR and CTF simulations in the single-phase regime, it can be assumed that CTF behaves similarly to the single-phase, polynomial density STAR model. The NMV CTF model input deck uses symmetric input parameters, which results in symmetric flow.

CTF uses β (also notated as Beta) to manipulate the flow of mass, momentum, and energy from "high" energy channels to "low" channels. β was selected as the calibration parameter in this milestone. A low β value indicates that there is little communication between channels (and therefore little cross flow and mixing) whereas a high β value indicates that the channels are more closely coupled.

3.3 Model Comparison

Table 5 shows the high-level code differences. This list is not all-inclusive and is a summary of the main code-level differences to consider when implementing the Hi2Lo process between STAR and CTF. These differences were highlighted in Sections 3.1 and 3.2.

Table 5: High-Level Code Differences.

Characteristics	STAR-CCM+	CTF
Compressibility	Incompressible, with polynomial density function.	Compressible
Phase	Single-Phase[1]	Two-Phase
Computational Time	1000 Core Hours	0.08 Core Hours
Temperature Measurements	Per Cell	Averaged Per Node

[1] STAR-CCM+ has two-phase capabilities, however it was assumed that the flow (using information from experimental data ranges) was single-phase in this application.

4. INITIAL STEPS

After CTF was used to generate appropriate equations for the STAR fluid properties, the two codes were compared to verify that they were behaving as anticipated and suitably similar for the Hi2Lo process. Case 1 was used for all comparisons.

4.1 Conservation Equations

An initial step with the NMV data was to compare the two models in detail and to verify that both codes conserve mass, energy, and momentum. Verifying agreement with the conservation equations allowed a few checks to be done that would otherwise be difficult, such as ensuring that the boundary conditions and code physics were implemented properly by the users and that key assumptions were accounted for and known. This check made it necessary to perform the calculations by hand as laid out in this section using quantities such as fluid velocity, specific enthalpy, and temperature to evaluate values for mass flow rate, energy through the simulation boundaries, and the momentum loss.

This step also made it possible to verify that the models were substantially similar and were calculating the same values for mass, momentum, and energy. If the codes were returning vastly different values for these quantities while still obeying conversation laws, it would indicate that the codes are not substantially similar.

4.1.1 Conservation of Mass

The equation for the conservation of mass is expressed as:

$$0 = \frac{d}{dt}\left(\int \rho dV\right)_{CV} + \left(\int \rho(\overrightarrow{u_{rel}} \bullet d\vec{A}\right)_{CS}$$

where ρ is density, u is velocity, V is volume, A is area. CV and CS represent the control volume and control surfaces, respectively. At steady state, $\frac{d\bullet}{dt} = 0$. Additionally $\overrightarrow{u_{rel}} = \vec{u}$ as the boundaries are stationary. $\vec{u}$ has only a z+ component and therefore is parallel to $\vec{A}$, which simplifies the above to the following:

$$\left(\int \rho u dA\right)_{inlet} = \left(\int \rho u dA\right)_{outlet}$$

The conservation equation is approximated with a hand equation, which makes it convenient to approximate the integral with a summation and use the channel averaged quantities for density, velocity and area at the inlet and the outlet.

$$\left(\sum_{i=1}^{36} \rho_i\, u_i A_i\right)_{outlet} = \left(\sum_{i=1}^{36} \rho_i u_i A_i\right)_{inlet}$$

The results from STAR-CCM+ and CTF for inlet and outlet mass flow rate is shown in Table 6 of Section 4.1.4.

4.1.2 Conservation of Energy

In a closed system, conservation of energy is defined as:

$$\Delta\dot{E} = \Delta\dot{U} + \Delta\dot{KE} + \Delta\dot{PE} = \Delta\dot{Q} - \Delta\dot{W}$$

It is assumed that $\Delta\dot{KE}$ and $\Delta\dot{PE}$ are orders of magnitude smaller than heat transfer and internal energy, $\Delta\dot{Q}$ and $\Delta\dot{U}$, and that no work is being done on the system. This simplifies the conservation equation to the form, where e is the internal energy of the flow and $\dot{m}$ is mass flow rate:

$$\Delta\dot{U} = \Delta\dot{Q} \rightarrow \Delta\dot{m}e = \Delta\dot{Q}$$

Specific enthalpy is defined as:

$$h = e + \frac{P}{\rho}$$

where h is specific enthalpy, P is pressure, and ρ is density.

Specific heat at constant pressure, c_p, is a quantity defined as:

$$c_p \equiv \left(\frac{\partial h}{\partial T}\right)_p$$

where h is specific enthalpy and T is temperature.

The equation for c_p is given to STAR as a polynomial function temperature. This equation was found previously and is shown in Section 3.1.4.

$$c_p(T) = A + BT + CT^2 + DT^3 + ET^4$$

Specific enthalpy can be written in terms of the coefficients of c_p and a reference temperature.

$$c_p \equiv \left(\frac{\partial h}{\partial T}\right)_p$$

$$\int dh = \int_{T_0}^{T} c_p(T)dT$$

$$h = \int_{T_0}^{T} (A + BT + CT^2 + DT^3 + ET^4)dT$$

$$h = \left(AT + \frac{1}{2}BT^2 + \frac{1}{3}CT^3 + \frac{1}{4}DT^4 + \frac{1}{5}ET^5\right) - \left(AT_0 + \frac{1}{2}BT_0^2 + \frac{1}{3}CT_0^3 + \frac{1}{4}DT_0^4 + \frac{1}{5}ET_0^5\right)$$

It is possible to set the reference temperature in the above equation to match the internal energy of CTF at the inlet. This is desirable as it allows the energy of the codes to be easier to compare at the outlet locations. For example, this temperature for Case 1 is:

$$T_0 = 385.7\ K$$

For the CTF hand calculation of internal energy, a simplified version of the above equation can be used as CTF is able to report specific enthalpy values. STAR cannot output this quantity.

$$e = h - \frac{P}{\rho}$$

The conservation of energy equation calculated using surface averaged values from the thirty-six subchannels is shown below:

$$\left(\sum_{i=1}^{36} \rho_i u_i A_i \left(h_i - \frac{P_i}{\rho_i}\right)\right)_{inlet} + \dot{Q}_{in} = \left(\sum_{i=1}^{36} \rho_i u_i A_i \left(h_i - \frac{P_i}{\rho_i}\right)\right)_{outlet}$$

The results from STAR and CTF for inlet and outlet power (energy) rate is shown in Table 6 of Section 4.1.4.

4.1.3 Conservation of Momentum

One-dimensional compressible Navier-Stokes equation in conservative form has the following form. At steady state, $\frac{d\bullet}{dt} = 0$.

$$\frac{\cancel{\partial \rho u}}{\cancel{\partial t}} + \frac{\partial \rho u^2}{\partial x} = -\frac{\partial P}{\partial x} + \mu \frac{\partial^2 u}{\partial x^2} + \rho g_x$$

CTF uses a different equation, where the effects of the viscous term are modeled with F and H. F is a model for the pressure drop due to viscous forces on the pins and H is the model for the pressure drop due to the grid spacers.

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} = -\frac{\partial P}{\partial x} - Fu^2 - Hu^2$$

It is difficult to directly compare the two equations since they have such different forms, and the values from CTF for F and H cannot be isolated easily. Conservation of momentum was judged by measuring the pressure drop from inlet to outlet for each code. As long as the pressure drop is substantially similar, momentum was assumed to be conserved. The results from STAR and CTF for pressure drop is shown in Table 6 of Section 4.1.4.

4.1.4 Conservation Equations Summary

Both codes conserved mass, energy, and momentum which indicated that STAR and CTF were behaving as anticipated and suitable for use of a Hi2Lo process. It also indicated that several key assumptions were being accounted for during calculations. For example, initially gravity was turned off in the physics model in one code (STAR). Comparing the pressure drop brought attention to this discrepancy and allowed corrections to be made in the STAR model before the validation step was carried out. Another example of an assumption that was accounted for while completing the calculations was the $\frac{P}{\rho}$ term of the specific enthalpy equation. It was initially discarded, which lead to the assumption $h \sim e$. After going through the calculations, the $\Delta \frac{\dot{m}P}{\rho}$ term was shown to be small compared to $\Delta \dot{m}e$, but not negligible.

Table 6: Conservation Equation Summary for STAR and CTF, Using Case 1.

Quantity	Inlet	Outlet	$outlet - inlet$	% Difference
		STAR		
Mass Flow Rate (kg/s)	8.3929	8.3924	-0.0004867	-0.00580
Power (MW)	11.766	11.756	-0.010463	-0.08894
Pressure (MPa)	-	-	-0.069377	N/A
		CTF		
Mass Flow Rate (kg/s)	8.3956	8.3941	-0.00145	-0.01729
Power (MW)	11.766	11.750	-0.01646	-0.13993
Pressure (MPa)	-	-	-0.069377	N/A

4.2 Cross Flow Magnitude

It was also important to compare the cross flow magnitude in both simulations. Cross flow affects symmetry in the simulations and it may be numerical or discretization artifact. For a Hi2Lo process to be implemented between STAR-CCM+ and CTF, the magnitude of the crossflow in both codes must be small. The amount of crossflow in the CTF model is small as directed cross flow is not enabled in the input deck and β also has a small nominal value. If the magnitude of the cross-flow in STAR is not small, this creates a difficult situation for Hi2Lo as it would make informing CTF from synthetic data from STAR a difficult exercise.

To calculate an estimate for cross flow between channels in STAR, 10 planes (normal to the x+ or y+ directions) were drawn through the middle of the rods and through the full length of the bundle. By using this configuration, only the flow through the gaps at five planes normal to x+ and five planes normal to y+ were considered.

Table 7 and Table 8 summarize the plane placements and surface averaged velocities and densities. Mass flow rate is calculated as:

$$\bar{\dot{m}} = \rho\vec{u} \bullet (A\hat{n})$$

which gives us an estimate of cross flow between the channels.

Table 7: Cross Flow in Planes Normal to Y+ in STAR.

X (in)	Area (m2) $\hat{n} = \hat{\imath}$	u (m/s)	v (m/s)	w (m/s)	Density (kg/m3)	$\dot{m}\ \hat{\imath}$ (kg/s)
-1	0.060	0.001	9.68E-04	2.794	847.139	0.062
-0.5	0.060	0.001	4.04E-04	2.872	845.050	0.056
0	0.060	-0.001	8.41E-04	2.861	844.883	-0.070
0.5	0.060	-0.002	8.76E-04	2.833	845.168	-0.097
1	0.060	-0.002	9.44E-04	2.782	847.203	-0.081

Table 8: Cross Flow in Planes Normal to X+ in STAR.

Y (in)	Area (m2) $\hat{n} = \hat{\jmath}$	u (m/s)	v (m/s)	w (m/s)	Density (kg/m3)	$\dot{m}\ \hat{\jmath}$ (kg/s)
1	0.060	-4.12E-04	-0.001	2.786	847.374	-0.075
0.5	0.060	-1.09E-03	-0.002	2.841	845.811	-0.092
0	0.059	-1.08E-03	-0.001	2.882	844.492	-0.045
-0.5	0.060	-7.87E-04	0.002	2.860	844.695	0.091
-1	0.060	-3.55E-04	0.001	2.789	847.168	0.073

The average velocities and mass flow rate of the gaps in the x+ and y+ direction is small compared to velocity and mass flow in the z+ direction (approximately 4 m/s and 8.4 kg/s). This indicates that there is little cross flow between channels in the STAR simulation and that the STAR simulation can reasonably be used to calibrate CTF. The cross flows additionally have approximately the same magnitudes (CTF cross flow ~ ±0.002 m/s).

5. QUANTITATIVE VALIDATION

After the models were checked for correctness for conservation of mass, energy, and momentum with a single test, quantitative validation was performed for the sets of tests given in Section 2.2. During the quantitative validation step, the STAR and CTF results were compared quantitatively with the experiment by using the exit temperatures as the quantity of interest and the $L_2 norm$ as the evaluation metric. For the following calculations, the $L_2 norm$ is defined as:

$$L_2 norm = \frac{\sqrt{\sum_{i=1}^{36} \left(T_{model,i} - T_{exp,i}\right)^2}}{\sqrt{\sum_{i=1}^{36} \left(T_{exp,i}\right)^2}}$$

The $L_2 norm$ is made nondimensional by the term in the denominator, which allows it to be expressed as the relative error. The $L_2 norm$ is multiplied by 100 to express that quantity as a percent error.

5.1 STAR-CCM+

The results from STAR show close agreement with the experiment, as illustrated in Figure 5-1. Figure 5-1 shows the STAR temperatures plotted against the experimental temperatures per subchannel for all validation tests, with lines drawn to bound 0% error, 1% error, and 2% error relative to the experimental data. The mean absolute error of the STAR validation temperatures, as calculated below, is equal to:

$$\frac{1}{36\times21} \sum_{i,j=1}^{i=36,j=21} \frac{\left|T_{model,ij} - T_{exp,ij}\right|}{T_{exp,ij}} = 0.009 \text{ or } 0.9\%$$

The groups of the experimental data points, such as near 315 °C correspond to experiments that have similar input parameters/boundary conditions in the simulations. The data points associated with the lower temperatures (less than 290 °C) have larger errors, but these data points have a more uniform scatter around the 0% error line whereas the higher simulation temperatures appear to be biased high compared to the experiment. An example of a cluster that appears to be biased high appears around 300 °C.

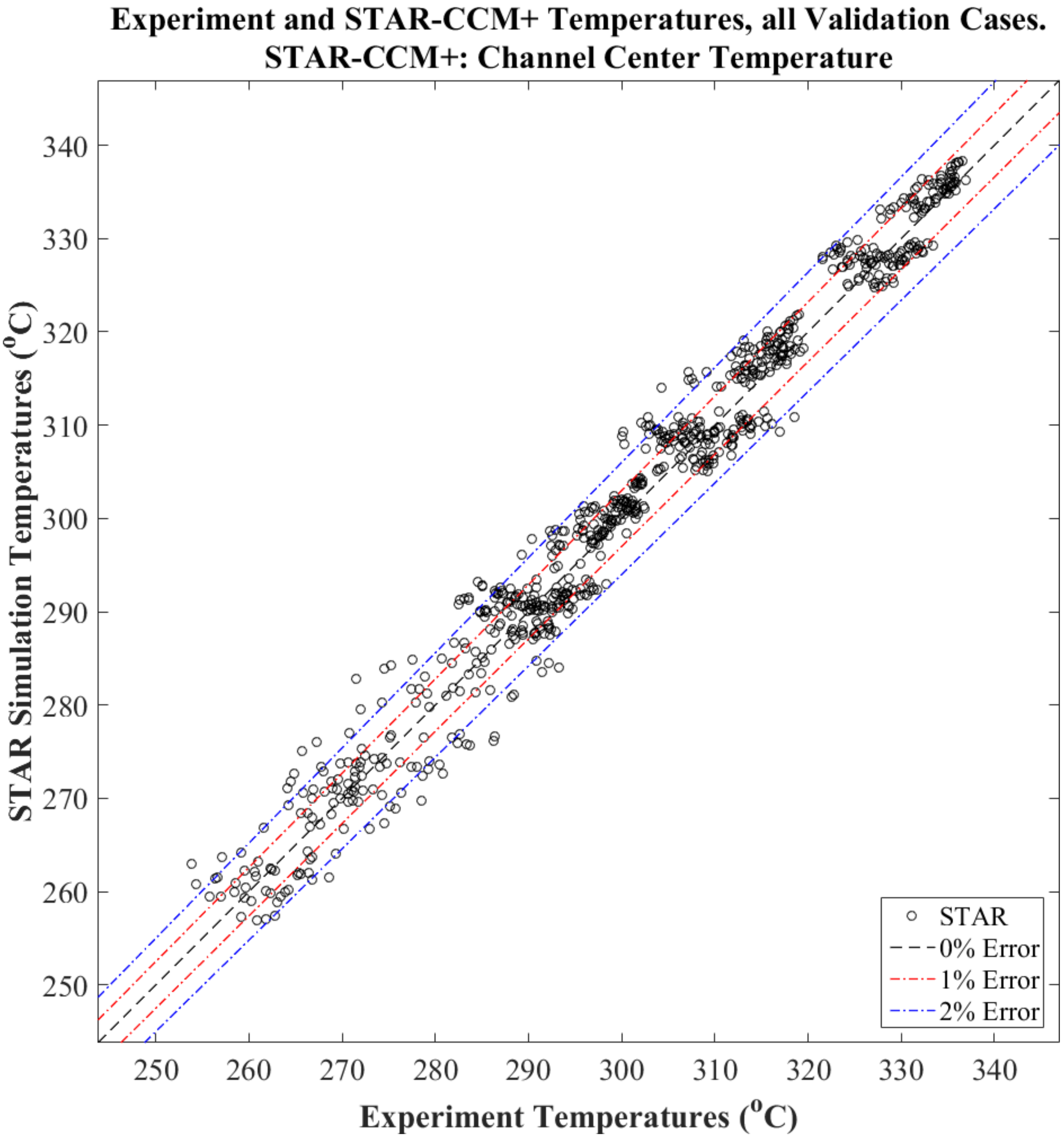


Figure 5-1: Experimental and STAR-CCM+ center temperatures, validation tests.

During quantitative validation of the twenty-one non-mixing tests for STAR, $L_2 norms$ were calculated using the STAR channel center temperature and the STAR channel average temperatures at the outlet. The definition of these two quantities was described previously in Section 3.1.3. A summary table of the $L_2 norms$ is given in Table 9. All $L_2 norms$ from are below 3% (0.03). Tests #116 and #117 have higher $L_2 norms$ than the remaining set of tests for both the channel center (~0.019) and the channel average temperatures (~0.021), however it should be noted that these tests (and tests 114 and 115) were performed at a much later time, which accounts for the large numbering gap between tests 9 to 25 and tests 114 to 117. Experimental concerns related to the testing gap were given in Section 2.2.

Table 9: STAR Quantitative Validation L_2norm Results.

Test #	Channel Center L_2norm	Channel Average L_2norm
9	0.008373	0.009931
10	0.00804	0.010152
11	0.010674	0.013746
12	0.012746	0.015005
13	0.010471	0.013137
14	0.010707	0.013393
15	0.009951	0.012612
16	0.01125	0.013826
17	0.010215	0.012244
18	0.010746	0.013702
19	0.007961	0.010176
20	0.008058	0.010276
21	0.012199	0.014328
22	0.010922	0.013588
23	0.010721	0.013301
24	0.011229	0.014614
25	0.012794	0.014549
114	0.012821	0.01596
115	0.015007	0.017326
116	0.018023	0.020426
117	0.019152	0.021721

The channel center temperatures measurements have a lower L_2norm value than the channel average temperatures, which was anticipated as the channel center temperature locations more closely approximate the experimental data locations. The channel center temperatures are lower than the channel averaged temperatures of the same subchannels as the data collection point is furthest from the rods (Figure 5-2). The channel averaged values are also being quantified as they are more analogous to results from CTF (which are channel averaged), which is relevant when using STAR to generate data for CTF.

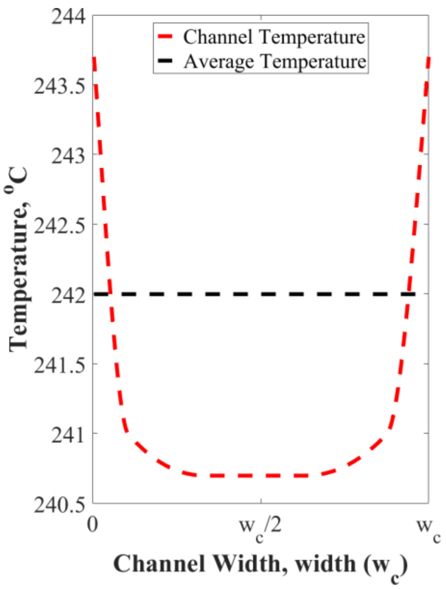


Figure 5-2: Example temperature profile in a single subchannel. The coldest location is at the center of the channel.

The STAR simulations closely match the experimental data, as defined by the $L_2 norms$, however the validation experimental data is relatively sparse for CFD. Boundary conditions are given, as is exit pressure and exit temperatures, but there is no additional information spatially or at the length scales is needed to do a full validation of the CFD simulations. The bulk quantities of mass, momentum, and energy as well (and associated fluid properties, the most notable one is temperature) are very close to experimental values which allows for a reasonable degree of confidence that the STAR simulation is performing the correct bulk physics calculations and that the flow is fully developed at the temperature collection location.

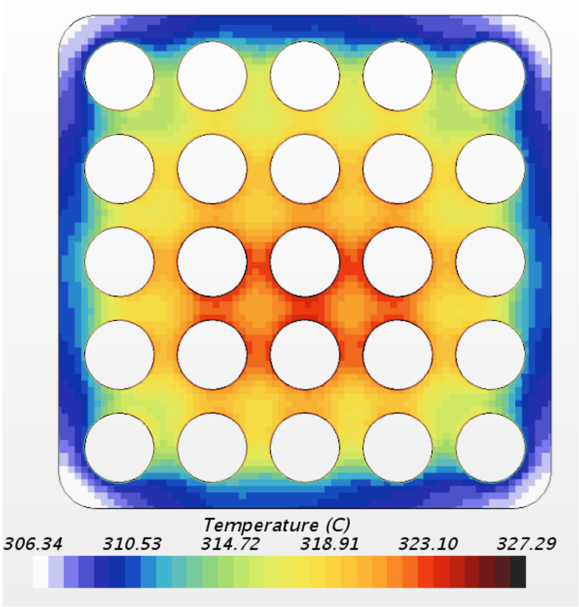


Figure 5-3: Scalar representation of the STAR-CCM+ outlet temperature for Case 1.

The results of the STAR simulations are symmetric (Figure 5-3). This is likely due to the symmetric and idealized simulation geometry. Without any of the actual geometric irregularities that would occur in the experimental apparatus, only a small amount of crossflow is induced by buoyancy and pressure differences between the channels in the STAR simulation. However, solution verification should be done on the CFD model to ensure that flow asymmetries (as seen in the experiment) are not being removed from the STAR simulations as a result of mesh coarseness.

5.1.1 Workflow of STAR-CCM+ Validation in Dakota 6.6

The STAR steps were mostly facilitated with scripts to make the entire process require little human intervention. The Dakota scripts for STAR and the java macros used in STAR are described in more detail below.

Dakota 6.6 was used for the STAR validation. The Dakota method used was a list parameter study. In a list parameter study, the full list of input parameters and number of parameters to be used per test is supplied to Dakota. Dakota uses this information to step through each set of test parameters. Java macros are used to interact with the STAR simulation in batch mode and collect outlet temperatures for the simulation results.

The workflow is described below. Scripts can be found in Appendix A.

1. Dakota Input File (Appendix A.1):
    a. Specifies list of evaluation points, variable names, the evaluation driver to use, and the number of responses to expect at the evaluation completion.
2. Dakota Driver (Appendix A.2):
    a. Copies files into working directory needed to run the STAR simulation, such as scripts written to collect responses.
    b. Uses Dakota tool Dprepro to search user created template files for keywords notated with "{{keyword_here}}" and replace with values specified in the Dakota input file. The keyword names match variable names given in the Dakota input file.
    c. STAR is launched in batch mode and the following operations are performed with java macros (Appendix A.3):
        i. Star_set.java: The simulation stopping criteria (number of steps) is specified.
        ii. Star_physics.java (Appendix A.4): The simulation boundary conditions such as mass flow rate, temperature, and total power are specified.
        iii. Star_fluid_props.java: The polynomials for density, specific heat, thermal conductivity, and viscosity as function of temperature are input into the STAR fluid properties of the physics model.
        iv. Star_initialize.java: The simulation history is cleared from the fields and the simulation is initialized.
        v. Star_runsim.java: The simulation is run until it reaches the stopping criteria.
        vi. Star_post.java: Scalar and monitor plots of temperature are saved as *.png files. Data is exported from monitors of the subchannels as .csv files for later post processing.
        vii. Plane_macro_stl.java (Appendix A.5): A separate script generates reports for channel averaged temperatures at the outlet. This is recorded in a file called results.txt.

viii. Star_postsave.java: If desired, the finished simulation can be saved for archival purposes.

d. A python script does additional post-processing on the csv files such as:

i. Converting temperatures in files from Kelvin to Celsius.

ii. Calculating iteration averaged temperatures for the final 500 iterations of each monitor. This step can be neglected if there is little noise in the temperature signal from the simulation. In the NMV, there is little noise, however this step was left in as it may prove valuable in future work where there is a significant signal to noise ratio.

iii. Recording the temperatures to an archival file.

e. The driver concludes by renaming the results.txt file that contains the channel averaged temperatures to results.out.

3. Dakota evaluation concludes.

a. Dakota reads the results.out file and records these values as the indicated responses.

b. Dakota begins the next iteration of evaluation points.

5.2 CTF

During quantitative validation of the twenty-one non-mixing tests, $L_2 norms$ were calculated using the CTF temperatures at the outlet for ten tests using a nominal value β=0. A summary table of the $L_2 norms$ is shown in Table 10. A more detailed description can be found in [1].

Table 10: CTF Quantitative Validation $L_2 norm$ Results.

Test #	CTF $L_2 norm$
10	0.01235
11	0.01351
14	0.01369
15	0.01279
18	0.01378
19	0.01093
22	0.01396
23	0.01405
114	0.01660
116	0.02223

5.3 Comparison of Results

Case 1 is being used for comparative purposes in Figure 5-4. The figure shows the averaged channel temperatures and the channel center temperatures for STAR-CCM+ comparted to the CTF temperatures. From the figure, for Case 1, a few conclusions can be drawn. The center temperatures calculated by STAR-CCM+ are smaller than all averaged temperatures from STAR and almost all (33 out of 36 temperatures) CTF test measurements. This is expected as the temperatures from the channel center are taken from cold points in the flow furthest from the hot rods, and this is reflected in Figure 5-3. In the peripheral channels (channels 1-7, 12-13, 18-19, 24-25, 30-36), STAR predicts a higher channel average temperature than CTF. For the inner channels, the STAR channel averaged temperatures and the CTF temperatures are in closer agreement.

Figure 5-5 shows that the STAR center temperatures have a lower $L_2 norm$ than CTF for all validation cases. The mean STAR center temperature calculated $L_2 norm$ using all twenty-one tests is 0.0115 (1.15%), whereas the mean CTF $L_2 norm$ using ten tests is 0.0144 (1.44%). The STAR channel averaged temperature $L_2 norm$ is 0.0140 (1.40%), which indicates that for the Hi2Lo process, information should be sent from STAR to CTF using the channel averaged values. These more closely approximate the averaged for the CTF temperatures (which are also temperature averaged).

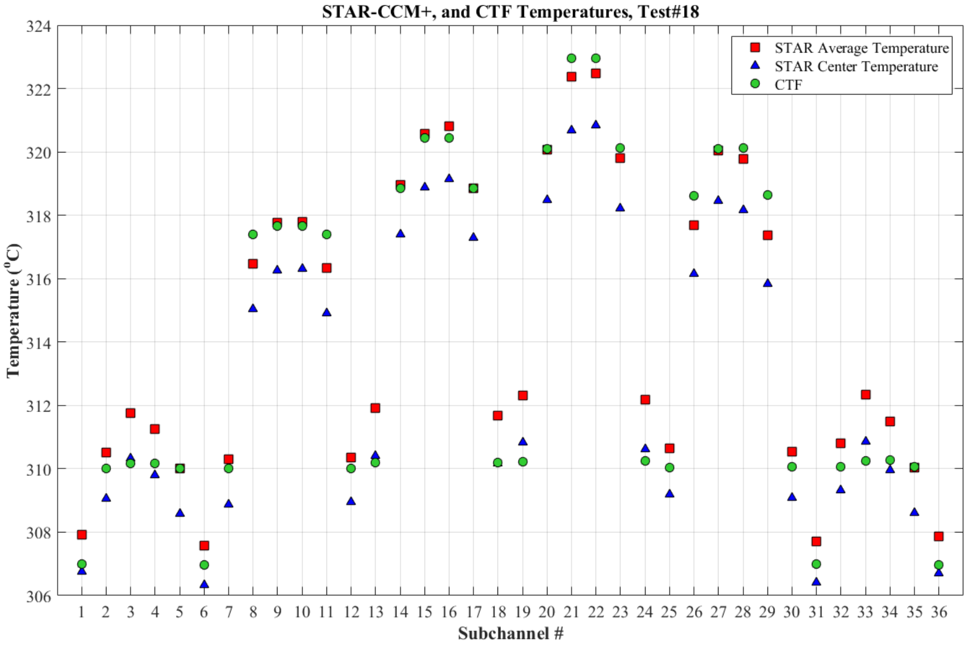


Figure 5-4: STAR-CCM+ and CTF subchannel temperature comparison for Case 1.

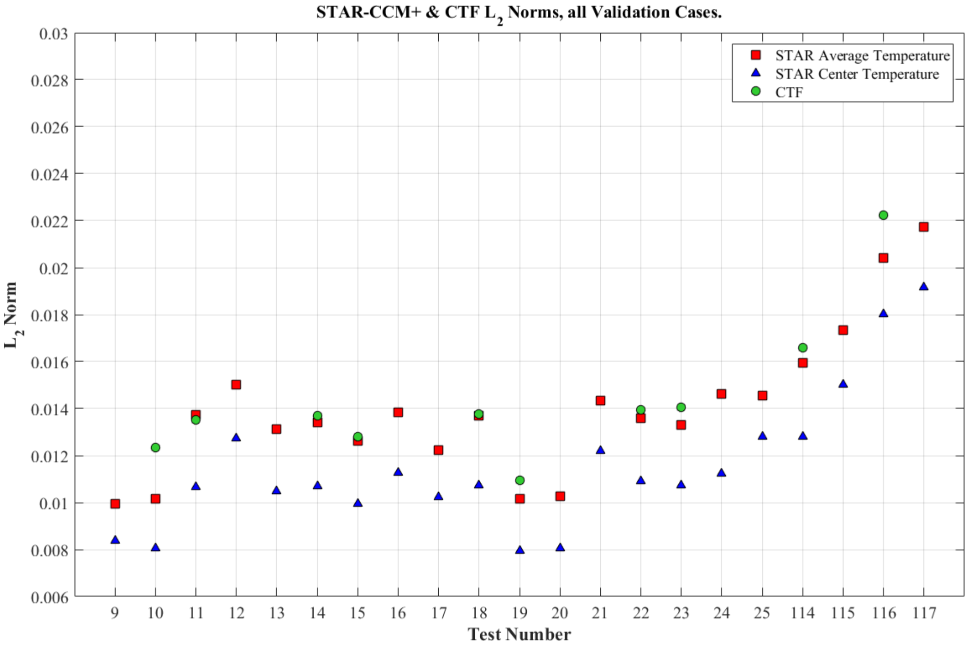


Figure 5-5: $L_2 norms$ for STAR-CCM+ and CTF, for validation.

6. UNCERTAINTY QUANTIFICATION

Limited uncertainty quantification (UQ) was performed in STAR, using the experimental test conditions as inputs. The UQ performed with STAR was limited to Case 1 due to time constraints and the computational expense of the STAR simulations. The tools for the Case 1 STAR UQ were built such that they can be used to perform a more complete UQ study with STAR. The tools can create an infinite number of unique profiles for the chosen parameters (mass flow rate and heat flux). These tools and their implementation to perform the UQ on Case 1 are described in the following sections.

6.1 Parameters for UQ

The parameters chosen for UQ in STAR are the turbulence model and the distributions of the boundary parameters mass flow rate and heat flux from the rods.

Two turbulence models are used for the UQ in addition to two mass flow rate profiles and two total power distribution profiles. This results in six total UQ simulations. This number was chosen due to time constraints as the STAR simulations are computationally expensive (1000 core hours per simulation) and a larger number of distributions per parameter has the potential to exponentially increase the total computational cost.

The individual parameters selected for the uncertainty quantification and their framework are discussed in the following sections. The scripts used to make the profiles for mass flow rate and heat flux use a random number generator and an initial input (the nominal boundary parameter value) to make the profiles. This allows this step to be automated for any number of desired profiles and initial inputs to the Matlab scripts without user intervention to make each unique profile. The Matlab scripts and java macros to incorporate the scripts into STAR are described below when relevant. The scripts (using a general case as input parameters) can be found in Appendix B.

6.1.1 Turbulence Model

The turbulence model was chosen as the first uncertainty quantification parameter. While changing the turbulence model, only the turbulence model type was changed. This parameter study did not involve uncertainty quantification based on changing the turbulence model coefficients, which were left as the default values. The turbulence models are Standard k-ω and Realizable 2-Layer k-ε. Standard k-ω was used during the previous validation steps and the Realizable k-ε model selected was suggested by Sal Rodriguez (SNL) and Bob Brewster (WEC), both of whom are experienced CFD analysts. The exit temperature is sensitive to the turbulence model selection, as shown in Figure 6-1.

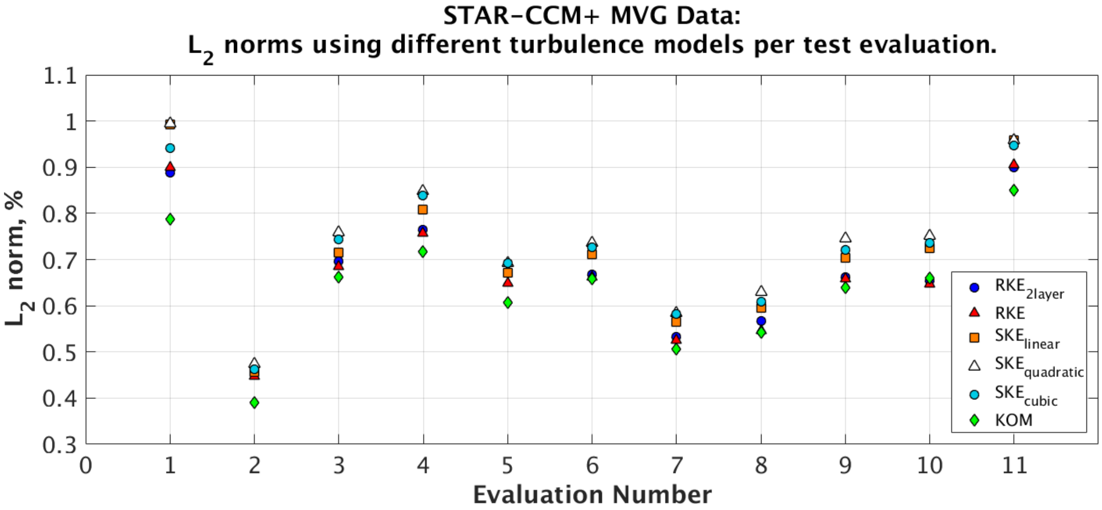


Figure 6-1: Standard k-ω (KOM) and Realizable 2-Layer k-ε (RKE_{2layer}) were used during the STAR UQ study for Case 1. This figure is reproduced from Section 3.1.2.

6.1.1.1 Standard k-ω

Standard (Wilcox) k-ω was chosen during previous work done for the 5x5 MVG data. STAR-CCM+ uses the 2006 correction, which is a major improvement the original 1988 model [6]. Standard k-ω yielded stable results with the lowest residuals out of the various turbulence models tested during previous work for a fixed mesh. It is important to note that the optimal turbulence model used for a given simulation is highly mesh dependent.

Standard k-ω assumes that turbulence is isotropic, i.e. $\overline{u'^2} = \overline{v'^2} = \overline{w'^2}$, however it includes a cross-diffusion and blending term to improve predictions near the walls [6].

6.1.1.2 Realizable k-ε

Realizable 2-layer k-ε was the second turbulence model to be used. Realizable k-ε is generally considered among CFD analysists to be the most popular turbulence model. It is based off of Standard k-ε with the following major differences [6]:

- It has a mathematical mechanism that eliminates negative normal stresses (realizability).
- It adds a production term for turbulent energy dissipation.

Realizable k-ε has the disadvantage that it is not as stable as Standard k-ε. Additionally, Realizable k-ε is generally not as versatile as k-ω. It has the advantage that it is easier to program than k-ω.

6.1.2 Mass Flow Rate

The total mass flow rate was kept as a fixed nominal value during the STAR UQ. The profile at the inlet was changed, keeping the total mass flow rate constant. In STAR, a simple way to change the shape of the inlet profile for a given mass flow rate is to change the temperature distribution of the inlet boundary. This can be illustrated by considering the mass flow rate equation.

$$\dot{m} = \rho(T)uA$$

Density is solely a function of temperature in STAR (which uses polynomial function of temperature). Changing the temperature affects the density and velocity profiles, while keeping the total mass flow rate fixed.

Temperature in the x and y direction were written as two vectors that were multiplied together to populate a matrix that spans both x and y coordinates to create a smooth table of temperature values in STAR.

$$f_x(x) = a_x + b_x x + c_x x^2$$

$$f_y(y) = a_y + b_y y + c_y y^2$$

$$\overrightarrow{T_x} = \langle f_x(x_0), f_x(x_1), \cdots, f_x(x_n)\rangle\hat{\imath}$$

$$\overrightarrow{T_y} = \langle f_y(y_0), f_y(y_1), \cdots, f_y(y_n)\rangle\hat{\jmath}$$

$$T = \overrightarrow{T_x}'\overrightarrow{T_y}$$

The total internal energy of the fluid at the inlet needs to remain the same as constant inlet temperature case, which is ensured by scaling the values within the T matrix so the average temperature value is equal to the nominal temperature. The net gain or loss of kinetic energy at the inlet is negligible as the total inlet kinetic energy of the flow is several orders of magnitude smaller than the net internal energy.

$$n = T_{nom}/\,\overline{T}$$

$$T_{new} = Tn$$

It is also necessary to remove x and y positions that would be occupied by the rods. This was done by implementing the Matlab `inpolygon` function and removing T(x,y) that appear in positions in the matrix that would be occupied by the rods.

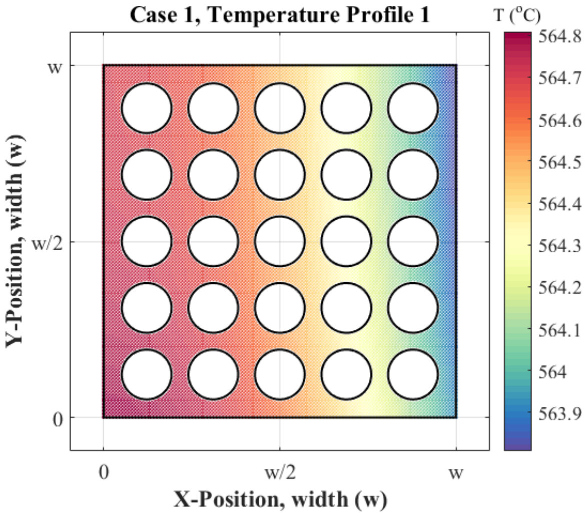


Figure 6-2: Inlet temperature profile example created for Case 1 UQ.

After these points are made, they are written to a table that lists temperature, x, y, and z coordinates. This table can be read by STAR and used as the inlet temperature profile.

6.1.3 Heat Flux Profile

To specify power shapes, the rods will be changed from power sources in STAR to utilizing a heat flux profile. Power is specified on a per-rod basis. HF is the heat flux of the corresponding rod.

$$\overline{HF} = \frac{Power_{rod}}{2\pi r L}$$

$$HF = func(z) = A\cos(\omega z + \pi) + B$$

The following equations demonstrate the power of the rod is solely determined by variable B in for the heat flux equation given above for a rod of constant radius. It is assumed the cosine function completes one period from 0 to 2π over z = 0 to 3 m and ω=2π/3.

$$P = \int_0^3 (Acos(\omega z + \pi) + B) dz \times 2\pi r$$

$$P = \int_0^3 \left(Acos\left(\frac{2\pi}{3} z + \pi\right) + B \right) dz \times 2\pi r$$

$$P = \frac{3}{2\pi} Asin\left(\frac{2\pi}{3} z + \pi\right) + Bz \Big]_0^3 \times 2\pi r$$

$$P = \left(\frac{3}{2\pi} Asin(3\pi) + 3B - \frac{3}{2\pi} Asin(\pi) \right) \times 2\pi r$$

$$P = 3B \times 2\pi r$$

$$P = 6\pi r B$$

Closer examination shows that B is the value of the rod average heat flux as the rod is 3 meters in length.

To correctly implement the heat flux equation in STAR it is also necessary to include an area scaling factor. It can be applied directly to the heat flux equation or solely to B. The area scaling factor, C, is included as a result of a small difference in geometry between the STAR meshed rod and a perfect cylinder.

$$HF(z) = Acos\left(\frac{2\pi}{3} z + \pi\right) + C\overline{HF}$$

The STAR rod cross section is a polygon with a slightly smaller perimeter (and therefore surface area) than a corresponding circle. A simplified example is shown below in Figure 6-3. The number of straight sides used in STAR-CCM+ when generating the volume mesh is dependent on the cell base size. The difference in perimeter size for the current simulation accounts for a 0.6% error in total power if no area correction factor is used.

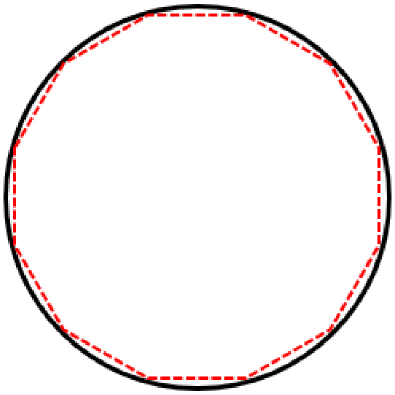


Figure 6-3: Simplified example showing a polygon (decagon) inscribed within a circle.

Figure 6-4 shows the heat flux profile for a single rod defined by the cosine equation. The total normalized power is the cumulative trapezoidal sum of the cosine function multiplied by the total surface area of the rod. This form assumes that the cells are evenly spaced in the axial direction, z+. The integrated power is 0.6% higher than the total power as the integrated power cosine function takes the area scaling factor into account.

After these functions are made, they can be written in STAR as field functions and applied at each of the rods to set the heat flux profile. This can be easily done with scripts.

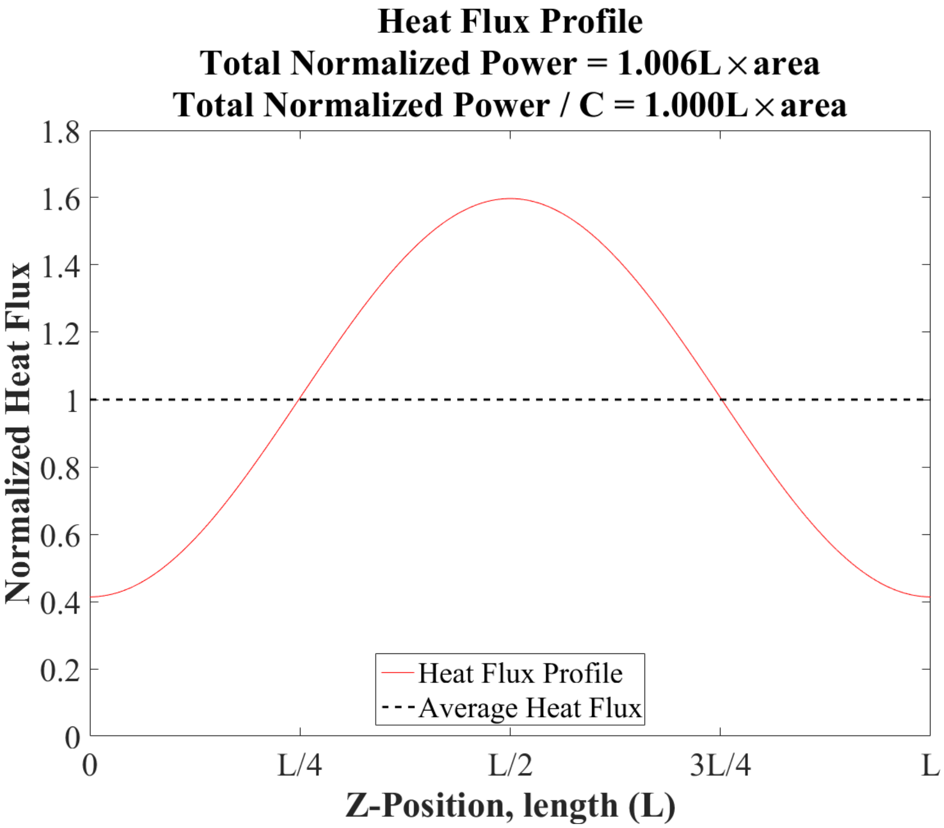


Figure 6-4: Heat flux profile. Total normalized power is the power before the area correction factor is applied. Total normalized power / C shows that the normalized power after the area correction factor is applied is equal to 1.000L x area.

6.2 Case 1, UQ Study

Case 1 was selected for a UQ study to inform CTF. As described previously, two turbulence models, mass flow rate profiles, and heat flux profiles were run for a total of six simulations. This number was selected due to time limitations. The temperatures per subchannel are shown in Figure 6-5 for the six UQ simulations. The effects of perturbing the sets of parameters on the outlet temperatures are discussed in the following sections.

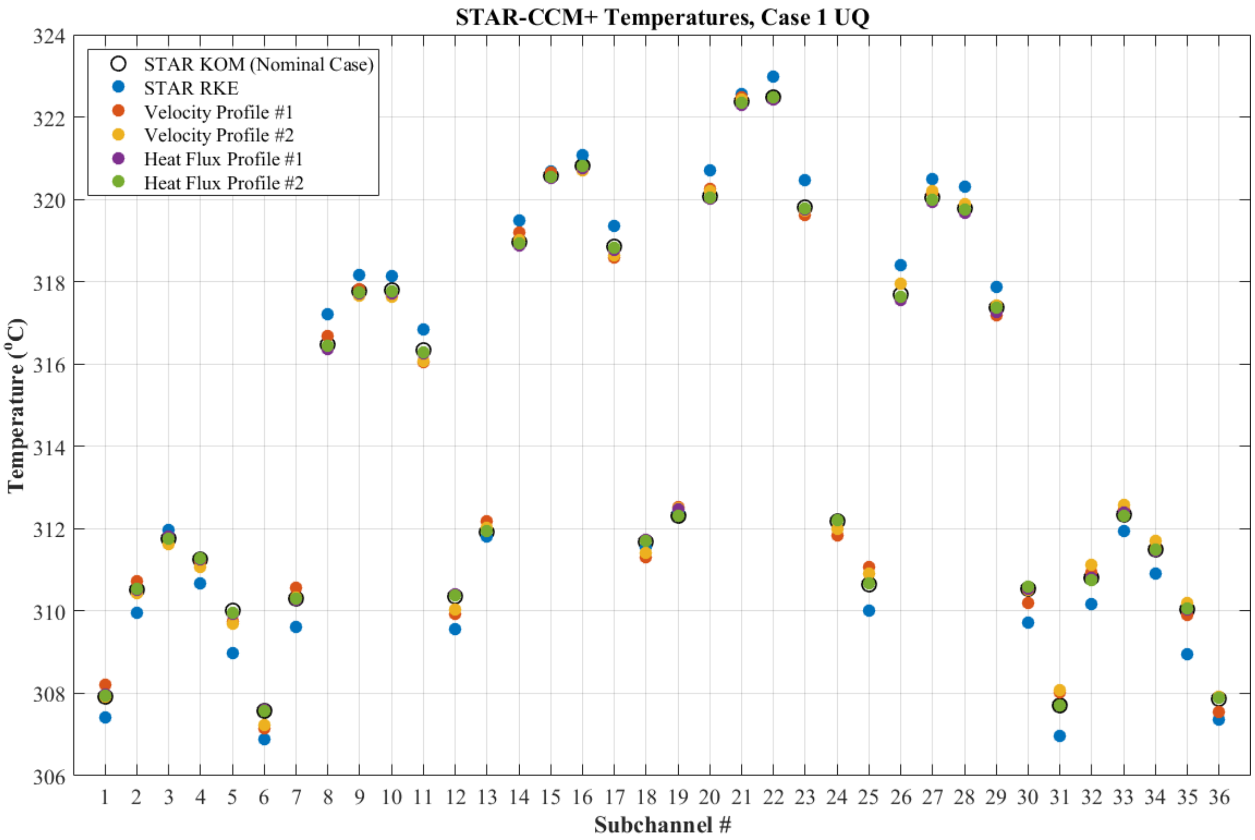


Figure 6-5: Results from the STAR UQ. Changing the turbulence model had the largest effect on the temperature results.

6.2.1 Turbulence Model

The largest temperature change per subchannel for the six parameter perturbations occurred when the turbulence model was changed from k-ω to k-ε. This was anticipated as a similar effect was seen previously with the mixing data simulations. Figure 3-3 illustrates the sensitivity of the temperature results for the MVG data to the turbulence model selection. The changes in the $L_2 norms$ when comparing k-ω or realizable k-ε directly correspond to changes in the subchannel temperatures, keeping all other parameters constant.

6.2.2 Mass Flow Rate

Changing the velocity/temperature profile at the inlet can have a potentially large effect on the outlet temperature distribution due to the lack of mixing vanes or other aspects of the geometry that would encourage crossflow or mixing. The figure below is a scalar plot of the STAR inlet temperature for temperature profile #1. Comparing subchannels 1 and 6 in the scalar plot (Figure 6-6) to the temperature plot (Figure 6-5) reveals that subchannel 1 is hot compared to the nominal (which uses a uniform inlet temperature) and subchannel 6 is cold compared to the nominal. The maximum change in temperature around the average temperature allowed for the UQ was 0.5°C. If a larger range for allowable temperatures was used at the inlet, the larger ΔT would have been seen at the outlet as there is limited buoyancy or pressure induced crossflow during the simulations. In cases where more mixing was induced by geometry (such as the MVG simulations), this effect would be less pronounced due to increased mixing.

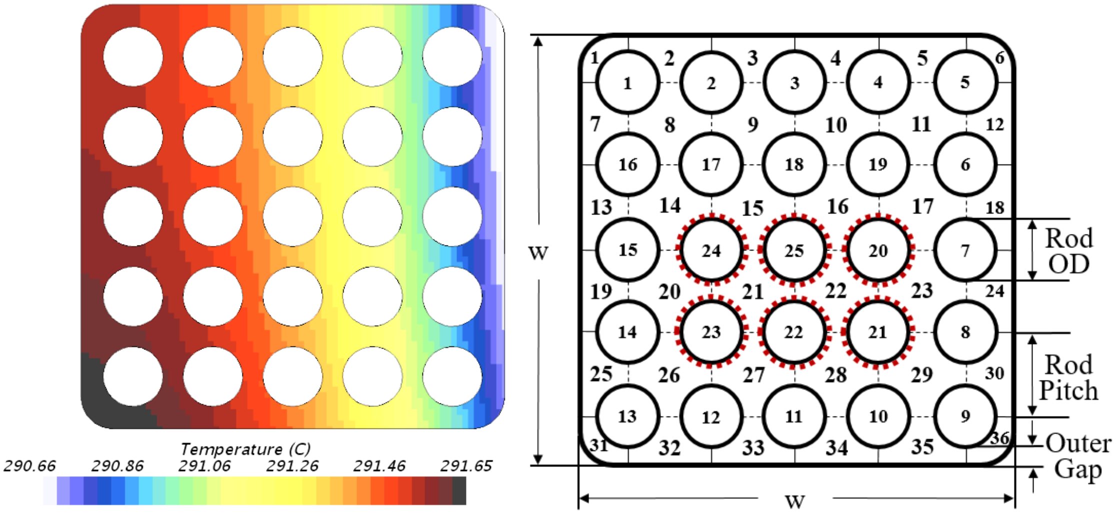


Figure 6-6: Case 1 STAR-CCM+ scalar plot of temperature at the outlet. Figure 2-2 is reproduced on the right for illustrative purposes.

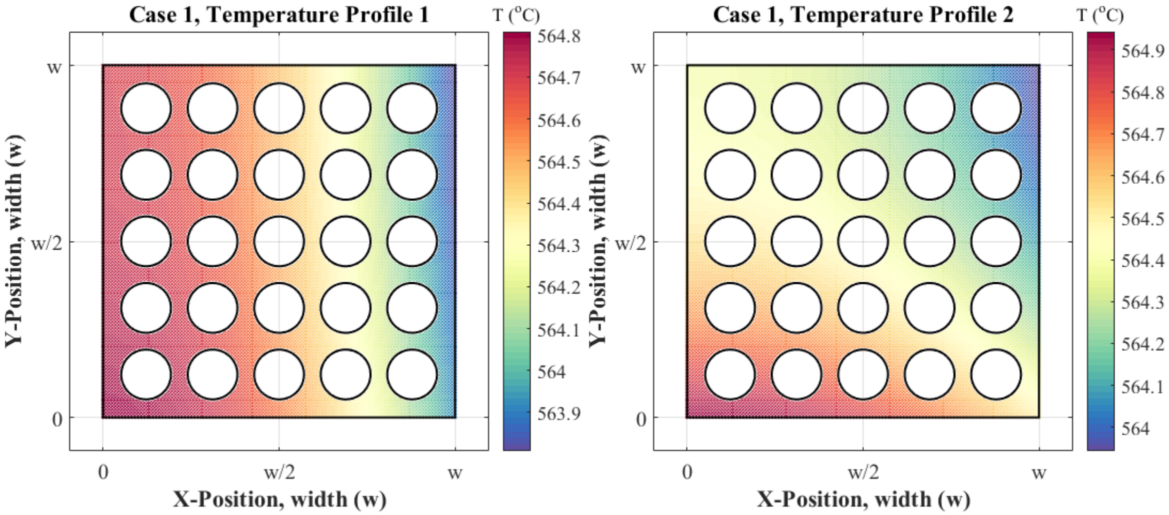


Figure 6-7: Case 1 UQ temperature profiles 1 and 2. $T_{max} - T_{min}$ ~ 1 ℃.

6.2.3 Heat Flux Profile

The heat flux profile shape has little effect on the outlet temperatures. The mean absolute change in temperature for the two heat flux profiles was 0.05 °C and 0.02 °C. This is small compared to the mean absolute change in temperature for changing the turbulence model (0.51 °C) and or by changing the mass flow rate profile (0.21 and 0.16 °C). This small effect is a result of the heat flux profile only varying in the axial direction. The total power of the individual rods is the same as the nominal case. The rods have different maximum amplitudes depending on the nominal total power of the rod, but all generated heat flux profiles have the maximum heat flux at location L/2 in the axial direction.

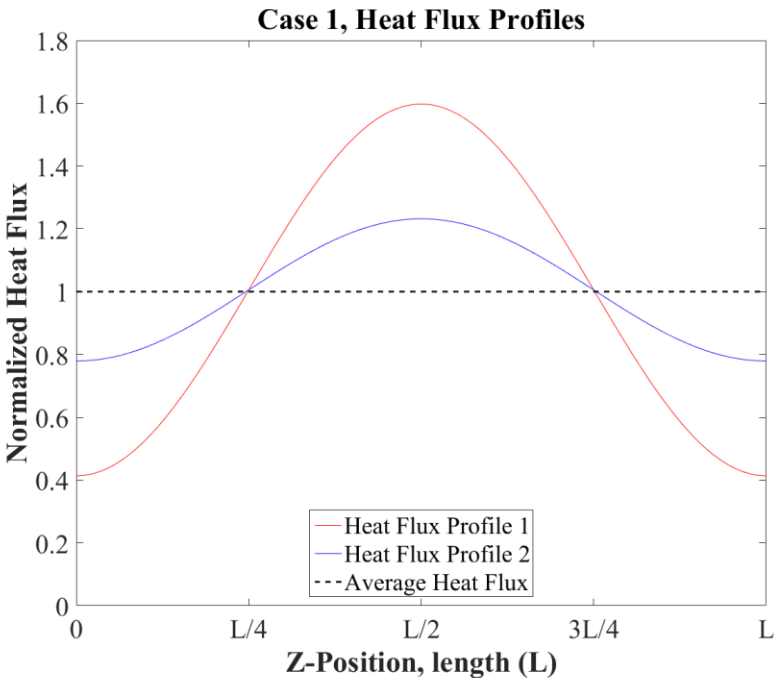


Figure 6-8: Case 1 UQ heat flux profiles 1 and 2 in terms of length L in the z direction. Both heat flux profiles average to be equivalent to the average heat flux, which is indicated by the dotted line.

6.3 Inclusion to CTF

The STAR UQ was used to approximate experimental noise for use with the CTF calculations. The STAR UQ results were reshaped into a 6x36 matrix and a covariance matrix (size 36x36) was calculated using this information. The covariance matrix was multiplied by 100 to approximate the magnitude of the experimental noise. This process can be found in more detail in [1]. A more complete STAR UQ study can be performed in the future that uses a greater number of parameter perturbations.

7. BRIEF SUMMARY OF PARALLEL ACTIVITIES PERFORMED WITH CTF

While quantitative validation and UQ was being performed with STAR-CCM+, several parallel activities occurred in CTF. These activities are briefly summarized in this section. The complete description of the CTF work can be found in [1].

After initial validation, a surrogate was constructed with Dakota using an LHS design. The surrogate was tested, trained, and built using the training design. After this, the surrogate was evaluated at the test points. Following the construction of the surrogate, Bayesian calibration was performed using a single experimental data point, which yielded a β value of 0.003197 as optimal (nominal was β=0). A second validation was run with the calibrate values in CTF which showed an improvement in the results. The mean $L_2 norm$ decreased from 0.0144 to 0.0136, which showed a statistical improvement in the CTF predictions for the validation test points.

8. EXPERIMENTAL DESIGN

The experimental design stage of the Hi2Lo process was mainly a CTF exercise with Dakota. It is contained in full in [1] and only briefly described here where relevant to STAR.

8.1 Workflow

The experimental design in Dakota uses a list of candidate evaluation points, the LHS design used in Bayesian calibration of CTF, and a calibration data file. The experimental design process was performed twice. The first experimental design used experimental conditions/results (only one test was used in the original calibration) as design points/ high-fidelity results. The second experimental design used STAR simulation conditions/results corresponding to the experimental evaluation points. The results of the two experimental design studies were compared once the entire experimental design procedure was complete.

This information was passed to Dakota using a manual process. The capability to perform the process inline with Dakota is possible with Dakota 6.6, however the two codes (CTF and STAR) were not able to be run on the same computing resources (account permissions for licenses differed between users). Once the high-fidelity simulation completes, results are passed back to Dakota. Additional configuration variables and their corresponding temperature results are supplied to Dakota as experimental design points.

8.2 Points Evaluated in STAR and Results

For the second experimental design process (defined above), CTF/Dakota used results from STAR that had been previously calculated during the validation step of the STAR process. The list of candidates was the twenty-one STAR validation runs minus Test #9. The experimental design took eighteen iterations to complete and the results are contained in [1].

When one experimental data point is used for calibration, β optimal is 0.004228. The experimental design step in CTF showed that when the STAR temperatures are introduced, the optimal value changes from 0.004228 to approximately 0.002881. The smaller β value in the STAR experimental design is likely a result of the simulation geometry. The simulation uses an idealized geometry, whereas the experimental geometry is unknown and may contain imperfections that induced additional turbulence and cross-flow that would not be seen with the idealized (symmetric) geometry, which would manifest in the CTF calibration as a larger β value. This is discussed more in Section 5.1.

9. SUMMARY OF FINAL CTF RESULTS

The third calibration of CTF using β=0.004228 after the experiment design step was completed and the $L_2 norms$ were compared to the results from the initial and second validations. The table and figure below are reproduced from [1].

Table 11: Summary of CTF $L_2 norm$ Values during Hi2Lo Process.

Test Number	Initial Validation (Beta = 0)	2nd Validation (Beta = 0.003197)	3rd Validation (Beta = 0.004228)
10	0.01235	0.01144	0.01128
11	0.01351	0.01342	0.01360
14	0.01369	0.01292	0.01299
15	0.01279	0.01247	0.01265
18	0.01378	0.01318	0.01322
18	0.01093	0.01079	0.01087
22	0.01396	0.01309	0.01308
23	0.01405	0.01295	0.01293
114	0.01660	0.01596	0.01612
116	0.02223	0.01997	0.01988
Overall L2	**0.0144**	**0.0136**	**0.0136**

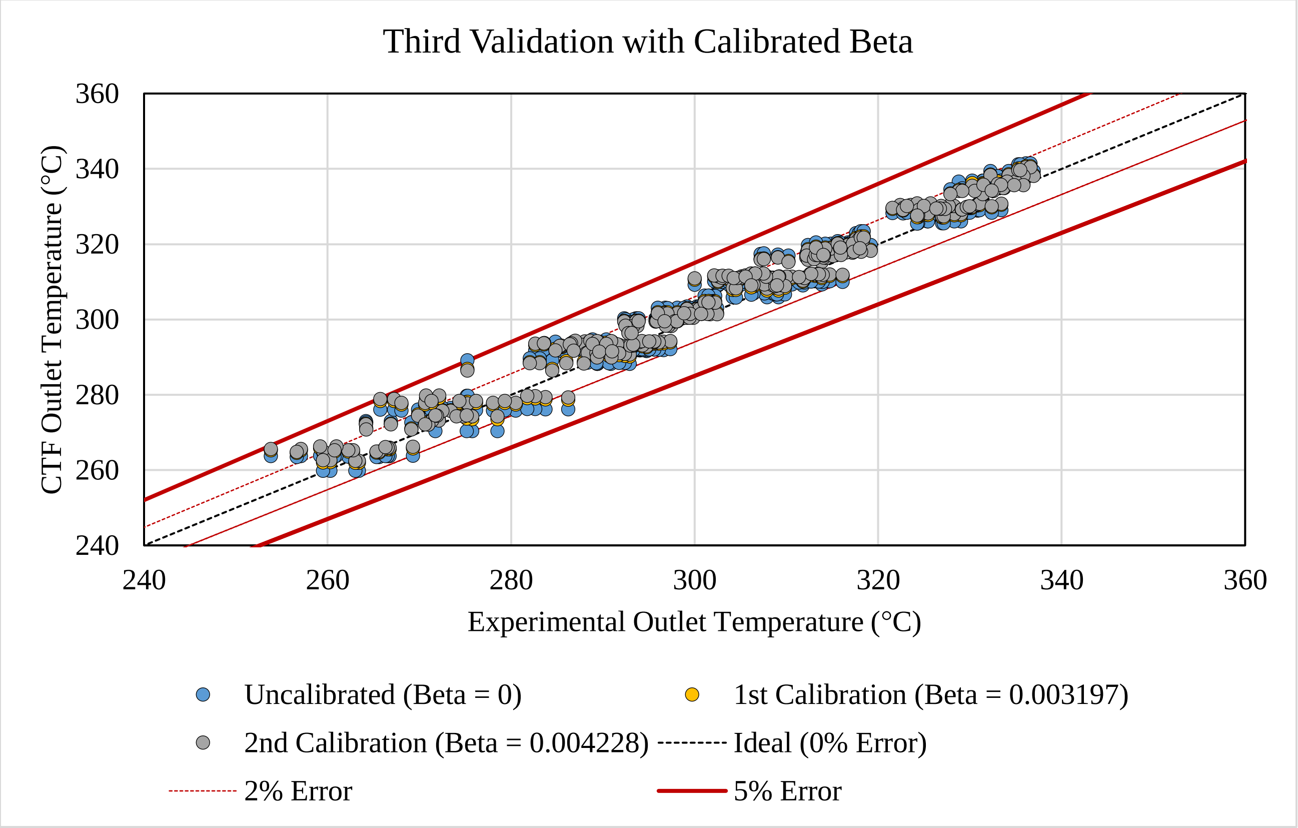


Figure 9-1: CTF final results after third validation.

Statistically, improvement was shown by the CTF $L_2 norms$ decreasing between validations 1 and 2. However it is difficult to determine statistically improvement was made by the experimental design step. The experimental design step $L_2 norm$ value matches that of the calibration $L_2 norm$, which is discussed in some detail in [1].

The optimal β calculated using the STAR temperature results (β=0.002881) for the experimental design data points differed from the results using solely the experimental data (β=0.004228) during the experimental design process. Some theories were given that might explain the discrepancy in the behaviors of the STAR and the experimental data during the previous sections, however these discrepancies can be difficult to quantify in terms of their effects on exit temperatures. The differences in behavior (asymmetric results and more cross flow in the experiment) are likely the cause of the different optimal β values during experimental design. However, it needs to be emphasized that the difference in subchannel temperatures between β=0 and β=0.004228 are small. Figure 9-1 shows β=0 and β=0.004228 on the same plot, and it is clear that changes in subchannel temperatures for the values are small. Statistical improvement ($L_2 norm$) was made during the CTF calibration/experimental design, however it is difficult to judge visually which β value matches the experimental data better due to the experimental data's asymmetry.

Further study of the high-fidelity simulations would help determine if the difference in optimal β are caused by discrepancies in the experimental geometry compared to the idealized (CFD) geometry, systematic error in the experiment (i.e. thermocouple damage or calibration), or incorrect physics or assumptions in the CFD code (such as a coarse fluid mesh). This could include solution verification or study of the effect of mesh refinement on the results of the simulation. More knowledge in this area would also be extremely useful for Hi2Lo-like applications where the high-fidelity code is operating in parameter spaces that lack experimental data. This study clearly showed the importance of validating high-fidelity codes when experimental data is available. Without experimental data, the optimal β value would have been set as β=0.002881 which would have shown the most agreement with STAR.

10.CONCLUSIONS

The work performed for this milestone demonstrated the steps needed to implement a Hi2Lo process between STAR-CCM+ and CTF by manipulating the β coefficient in CTF.

The work done to demonstrate the Hi2Lo process is complete and the framework is reproducible and useable for a problem better suited for such analysis. Major improvements can be made in lower-fidelity codes with the steps and tools laid out in this milestone. The results of the Hi2Lo coupling between STAR and CTF were not as anticipated due to uncertainties in the experimental data and the STAR simulations, however the process for both codes and their respective couplings with Dakota (and each other) were well documented. The UQ in STAR can be greatly expanded on with dozens of additional computation points, and the tools have been made to facilitate this.

The milestone work showed that the STAR simulations are performing well and returning values close to experimental values during this study. Bulk values of both are roughly equivalent, which indicates that no critical assumptions are being neglected for the simulations in this flow regime. Based off of the $L_2 norms$, the STAR simulation results are closer to the experimental data than CTF.

The work performed showed that CTF's β can be calibrated to match the experimental data more closely. It is clear that manipulations of β can match the experimental results more closely statistically, as judged by $L_2 norms$, however it is not obvious that adjusting β is capturing some unresolved physics. The work also showed that β can be manipulated to match STAR simulation results. The results of both codes are symmetric, whereas the experiment is asymmetric, which resulted in the calibration of β returning different results for STAR and the experimental data.

Future work is needed to address the symmetry of the code results and asymmetry in the experiment. More work can be done on the STAR simulation to attempt to better match the experimental data (and asymmetry) by changing simulation physics, mesh, or geometry.

Experimental data for fuel bundle flows can be sparse, therefore it is valuable to take advantage of any existing experimental data (such as the NMV set) when performing code validation and calibrations. Information gained about the differences in experimental and CFD data can be used to improve the high-fidelity simulations where no experimental data exists by quantifying (and identifying) experimental errors, differences in the physics between simulation and experiment (ideal geometry and other assumptions), and effects of the mesh on results.

LIST OF REFERENCES

[1] N. Gordon, "L3:VVI.H2L.P15.01," CASL, 2017.

[2] "PFT-13-8, Description of Non-Mixing Vane Grid CHF Test for CASL DNB Challenge Problem," Westinghouse Electric Company LLC, February 2013. (Proprietary).

[3] P. F. Joffre, "PFT-16-3, Rev 1," Westinghouse Electric Company LLC, February 2016. (Proprietary).

[4] E. Lemmon, M. McLinden and D. Friend, in *NIST Chemistry WebBook, NIST Standard Reference Database Number 69*, P. Linstrom and W. Mallard, Eds., Gaithersburg, MD, 20899: National Institute of Standards and Technology.

[5] A. Krueger, "L3:VVI.VUQ.P15.02," CASL, 2017.

[6] S. Rodriguez, Personal communication, 2016.

[7] L. Gilkey, "L3:VMA.VUQ.P13.08 Milestone Report," CASL, 2016.

APPENDIX A: VALIDATION SCRIPTS

For the following scripts, the input parameters (inlet temperature, total heat, bundle width) have been changed to representative values for illustrative purposes. The parameters do not reflect actual experimental values.

A.1 Dakota Input File

```
## Validation input to Dakota


environment
      tabular_data
      tabular_data_file   =   'NMV_KOM_Validation.dat'
      output_precision     =   1e-16

method
      output verbose
      list_parameter_study

      list_of_points    =

      #     Pressure    Tin      m_dot       AFLUX

      9     1000        500      30.0        5.0
      10    2000        600      25.0        6.0


model
    single

variables
   continuous_design    =  5
   descriptors        =

      'testNumber'   'Pressure'  'T_in'   'G_in'      'AFLUX'


interface
    fork
    asynchronous evaluation_concurrency = 12
        analysis_driver =  'driver_Validation.sh'
        parameters_file =  'params.in'
        results_file       'results.out'
   failure_capture recover
      NaN NaN NaN NaN
      NaN NaN NaN NaN
      NaN NaN NaN NaN
      NaN NaN NaN NaN
   work_directory
   directory_tag named 'STAR_case_dir'
   file_save            directory_save

responses
    objective_functions     =   36
      descriptors
         'T1'  'T2'  'T3'  'T4'  'T5'  'T6'
         'T7'  'T8'  'T9'  'T10' 'T11' 'T12'
         'T13' 'T14' 'T15' 'T16' 'T17' 'T18'
         'T19' 'T20' 'T21' 'T22' 'T23' 'T24'
         'T25' 'T26' 'T27' 'T28' 'T29' 'T40'
         'T31' 'T32' 'T33' 'T34' 'T35' 'T36'

      no_gradients
        no_hessians
```

A.2 Dakota Driver File

```
#!/bin/bash

# This was made for dakota 6.6.


#####################################################################
##
##  Dakota driver.sh for STAR-CCM
##  Originally written 2/1/2017 by Lindsay Gilkey to work with version
##  6.6 of Dakota. Modified to work with NMV data.
##
##
###################################################################

# $1 and $2 are special variables in bash that contain the 1st and
# 2nd command line arguments to the script, which are the names of
# the Dakota parameters and results files, respectively.

params=$1
results=$2

# Begin the time log file.
printf 'START TIME FILE' > time.txt

# Copy all files from the "copy" folder into the current working
# directory
cp -r ../copy/* ./

#####################################################################
##
##  Pre-processing Phase -- Generate/configure an input file for
##  your simulation by substituting in parameter values from the
##  Dakota parameters file.
##
##  This includes all the dprepro / template steps in dakota.
##
###############################################################################

# dprepro for all java scripts needed to run STARCCM+.

# Set physics parameters and boundary conditions
dprepro --left-delimiter={{ --right-delimiter=}} $params
../templates/star_physics.java.template ./starscripts/star_physics.java

# dprepro for saving the sim file with the correct new name.
 dprepro --left-delimiter={{ --right-delimiter=}} $params
../templates/star_postsave.java.template ./starscripts/star_postsave.java

# dprepro for python postprocessing script
dprepro --left-delimiter={{ --right-delimiter=}} $params
../templates/python_postProcessor.py.template ./python_postProcessor.py
```

```
#####################################################################
##
## Execution Phase -- Run STAR-CCM+
##
###################################################################

# Make sure all shell scripts have the correct permissions (can be
# executed).
chmod 755 *.sh

# Locate the STAR simulation.
locateSim=$(printf '../finishedSimulations/NMV_KOM.8_gravity.sim')

# Start and time STARCCM+
{ time starccm+ -batch star_meshAndRun.java -batchsystem pbs -power -np 1008 -
rsh ssh $locateSim > star_output.log ; } 2>> time.txt


# Once the simulations are finished, sleep to allow the
# files to be written and closed out before proceeding.

sleep 1

#####################################################################
##
##  Post-processing Phase
##
###################################################################

# Copy files and remove first lines from appropriate csv before it
# is read into python. The csv reader in python needs commas and
# cannot read headers.

# Navigate to the /starscripts/outputs/ folder.
cd ./starscripts/outputs

# The two files that I have in there are:
# 1. The instantaneous temperature readings.
# 2.  The monitor table that contains iteration & temperature
#     readings for all subchannels.

# For file #1:
# Copy the "raw" file that has a header.
cp data_star_TempPressure_table_raw.csv
data_star_TempPressure_table_noheader.csv
# Remove the first line of the copied file.
sed -i '1d' data_star_TempPressure_table_noheader.csv

# For file #2:
# Copy the "raw" file that has a header.
cp TEMP_Monitor_Plot.csv TEMP_Monitor_Plot_noheader.csv
# Remove the first line of the copied file.
sed -i '1d' TEMP_Monitor_Plot_noheader.csv

# Run Python Postprocessor that will post process and perform
# calculations.

# Navigate up to your case file folder.
cd ../../
```

```
# Run python and record results.
python   python_postProcessor.py >results_center.out

# Print that dakota finished in your time.txt file.
printf '\n\nFIN' >> time.txt

# Rename results.txt to results.out to sent the results directly to # dakota.
mv results.txt results.out

sleep 2

# Remove any backup files created by STAR.
rm ../finishedSimulations/*.sim~
```

A.3 Main STAR Macro

```
// STAR-CCM+ macro: beep.java
// Written by STAR-CCM+ 11.02.009


package macro;

import java.util.*;

import star.common.*;

public class star_meshAndRun extends StarMacro {

   public void execute() {
      execute0();
   }

   private void execute0() {

      Simulation simulation_0 =
         getActiveSimulation();

   ////////////////////////////////////////////////////////////////
   // Run all java macros
   ////////////////////////////////////////////////////////////////

new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/star_set.java"))).play();

new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/star_physics.java"))).play();

new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/star_fluid_props.java"))).play();

new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/star_initialize.java"))).play();

new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/star_runsim.java"))).play();

new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/star_post.java"))).play();
new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/plane_macro_stl.java"))).play();

new StarScript(getActiveSimulation(),
new java.io.File(resolvePath("./starscripts/star_postsave.java"))).play();

   }
}
```

A.4 Example Dprepro Template File (Star_physics.java.template)

```
// STAR-CCM+ macro: ref.java
// Written by STAR-CCM+ 11.02.009


package macro;

import java.util.*;

import star.common.*;
import star.base.neo.*;
import star.flow.*;
import star.energy.*;

public class star_physics extends StarMacro {

   public void execute() {
      execute0();
   }

   private void execute0() {

      Simulation sim =
         getActiveSimulation();

   // Get Region
      Region FV =
         sim.getRegionManager().getRegion("FV");

   ////////////////////////////////////////////////////////////////////
   // Set initial conditions
   ////////////////////////////////////////////////////////////////////

      PhysicsContinuum physicsContinuum_0 =
      ((PhysicsContinuum) sim.getContinuumManager().getContinuum("Physics 1"));

   // Set up all my units:

      Units units_psi =
         ((Units) sim.getUnitsManager().getObject("psi"));

      Units units_F =
         ((Units) sim.getUnitsManager().getObject("F"));

      Units units_lbs =
         ((Units) sim.getUnitsManager().getObject("lb/s"));

   // Set initial velocity
   // Assumption that the density of the water = 1 kg/m^3
      VelocityProfile velocityProfile_0 =
         physicsContinuum_0.getInitialConditions().get(VelocityProfile.class);

   // Dakota param inVelocity. Assume that initial velocity is
   // normal to the inlet, it will only have one component in the Z
   // direction
   // From the mass flow rate, initial velocity will be: V = mdot/A/rho

      velocityProfile_0.getMethod(ConstantVectorProfileMethod.class).
         getQuantity().
         setComponents(0.0, 0.0,{{G_in*0.45359237/4.2628/0.00064516/1000}});

   // Set initial temperature.

      StaticTemperatureProfile staticTemperatureProfile_0 =
      physicsContinuum_0.getInitialConditions().
         get(StaticTemperatureProfile.class);

   // Dakota param inTemp. Assume that initial temperature is the
   // same as the inlet temperature

      staticTemperatureProfile_0.getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setValue({{T_in=0.0}});


      staticTemperatureProfile_0.getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setUnits(units_F);

   //////////////////////////////////////////////////////////////////
   // Set boundary conditions
   //////////////////////////////////////////////////////////////////

   int i=0;

   // These values have been changed from experimental values.

   double[] P =
   {0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,
   1.1,1.1,1.1,1.1,1.1,1.1};

   // Rod Boundary Condition

   while (i<25) {

      Boundary boundary_rod =
         FV.getBoundaryManager().getBoundary("Rod " + (i+1));

      boundary_rod.getConditions().get(WallThermalOption.class).
         setSelected(WallThermalOption.Type.HEAT_SOURCE);

      HeatSourceProfile heatSourceProfile_rod =
         boundary_rod.getValues().get(HeatSourceProfile.class);

      double PR = {{ AFLUX * 1000 / 0.3048 * 3 }}*P[i];

      heatSourceProfile_rod.getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setValue(PR);

      i++;
   }

   // Set Inlet BC

      Boundary boundary_inlet =
         FV.getBoundaryManager().getBoundary("Inlet");

      boundary_inlet.setBoundaryType(MassFlowBoundary.class);

      TotalTemperatureProfile totalTemperatureProfile_inlet =
         boundary_inlet.getValues().get(TotalTemperatureProfile.class);

   // Dakota param inTemp in degrees F

      totalTemperatureProfile_inlet.
         getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setUnits(units_F);

      totalTemperatureProfile_inlet.
         getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setValue({{T_in=0}});


   // Dakota param Gin in lbm/s
      boundary_inlet.setBoundaryType(MassFlowBoundary.class);

      MassFlowRateProfile massFlowRateProfile_inlet =
         boundary_inlet.getValues().get(MassFlowRateProfile.class);

      massFlowRateProfile_inlet.getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setValue({{G_in=0}});

      massFlowRateProfile_inlet.getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setUnits(units_lbs);


   // Set Outlet BC
      Boundary boundary_outlet =
         FV.getBoundaryManager().getBoundary("Outlet");

   // Set temperature == dakota param T_in in degrees F. This only is applied if
   // there is backflow
      StaticTemperatureProfile staticTemperatureProfile_outlet =
         boundary_outlet.getValues().get(StaticTemperatureProfile.class);

      staticTemperatureProfile_outlet.
         getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setValue({{T_in=0}});

      staticTemperatureProfile_outlet.
         getMethod(ConstantScalarProfileMethod.class).
         getQuantity().setUnits(units_F);

   }
}
```

A.5 plane_macro.stl.java

```
// STAR-CCM+ macro plane_macro

package macro;

import java.util.*;
import star.common.*;
import star.base.neo.*;
import star.vis.*;
import star.base.report.*;
import star.energy.*;
import java.io.*;
import java.nio.*;
import star.flow.*;

public class plane_macro_stl extends StarMacro {

   BufferedWriter bwout = null;

   public void execute() {

   try {

      //set simulation
      Simulation sim = getActiveSimulation();

      //Units

      // Units of the constrained plane
      Units units_0 =
         sim.getUnitsManager()
         .getPreferredUnits(new IntVector(

      new int[] {0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}));

      Units units_temp = // Temperature Units
         ((Units) sim.getUnitsManager().getObject("K"));

      Units units_length = // length Units
         ((Units) sim.getUnitsManager().getObject("m"));

      Units units_pressure= // Pressure Units
         ((Units) sim.getUnitsManager().getObject("Pa"));

      // Field Functions:

      // Temperature
      PrimitiveFieldFunction primitiveFieldFunction_temp =
         ((PrimitiveFieldFunction) sim.getFieldFunctionManager().
         getFunction("Temperature"));

      // Pressure
      PrimitiveFieldFunction primitiveFieldFunction_pressure =
         ((PrimitiveFieldFunction) sim.getFieldFunctionManager().
         getFunction("Pressure"));
```

```
// Velocity
PrimitiveFieldFunction primitiveFieldFunction_velocity =
   ((PrimitiveFieldFunction) sim.getFieldFunctionManager().
   getFunction("Velocity"));

// Velocity k
VectorComponentFieldFunction vel_k =
   ((VectorComponentFieldFunction) primitiveFieldFunction_velocity.
   getComponentFunction(2));

// Velocity Magnitude
VectorMagnitudeFieldFunction vel_mag =
   ((VectorMagnitudeFieldFunction) primitiveFieldFunction_velocity
   .getMagnitudeFunction());

// Mass Flow Rate
PrimitiveFieldFunction primitiveFieldFunction_mdot =
   ((PrimitiveFieldFunction) sim.getFieldFunctionManager().
   getFunction("MassFlux"));

// Specific Heat
PrimitiveFieldFunction primitiveFieldFunction_cp =
   ((PrimitiveFieldFunction) sim.getFieldFunctionManager().
   getFunction("SpecificHeat"));

// rho
PrimitiveFieldFunction primitiveFieldFunction_rho =
   ((PrimitiveFieldFunction) sim.getFieldFunctionManager().
   getFunction("Density"));

//Fluid Region
Region region_00 = sim.getRegionManager().getRegion("FV");
   Object[] region= {region_00};

//create a multi-dimensional array holding the subchannel properties
//read in the values from the 'sub_vals' array in the python driver

// The array has been removed from this script as it contains proprietary
// Westinghouse information about the bundle geometry. There will be 36
// lines, one for each of the subchannels.

double[][] sub_vals= new double[][]{
   {1.0, ..., ..., ..., ..., ..., ..., ..., ..., ..., ...,
   };

double[] z_positions_1= {3.0};
//number of subchannels & axial positions

int n_subchannels= sub_vals.length;
int n_z= z_positions_1.length;

// Create Vector in the Z direction for normal vector
// flow direction **only works in +z-direction**

double[] norm_vec= {0.0, 0.0, 1.0};

//set the index convention for parameters within the sub_vals array:

int c1_x= 3;      int c1_y= c1_x+1;
int c2_x= c1_x+2; int c2_y= c2_x+1;
int c3_x= c2_x+2; int c3_y= c3_x+1;
int c4_x= c3_x+2; int c4_y= c4_x+1;

//declare the variables for the constrained plane corners

double x1, x2, x3, x4, y1, y2, y3, y4, z1, z2, z3, z4;
double x_avg, y_avg, z_avg;
double[] plane_avg= {0.0, 0.0, 0.0};
double[] sub_corners;

//strings for the axial planes

String plane_name, ax_con_name;
String plane_name1, plane_name2, plane_name3;
String ax_con_name1, ax_con_name2, ax_con_name3,
      ax_con_name4, ax_con_name5;


// I am using this as a "header" line

sim.println("Z_Position(m) SC cp T rho A V");

sim.println("NOTE: FOR Z=3.0m & Z=0m, IT IS SLIGHTLY OFFSET IN STAR.
PRESSURE AT 3.0 METERS SHOULD = 0 PASCALS");

bwout = new BufferedWriter(new FileWriter(resolvePath("../results.txt")));

// Iterate through all the axial positions

for (int n=0; n < n_z; n++) {
   for (int i=0; i < n_subchannels; i++) {

      //set the constrained corner locations, based on:
      //{x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4}
      //the values come from the array sub_vals

      x1= sub_vals[i][c1_x];
      y1= sub_vals[i][c1_y];
      x2= sub_vals[i][c2_x];
      y2= sub_vals[i][c2_y];
      x3= sub_vals[i][c4_x];
      y3= sub_vals[i][c4_y];
      x4= sub_vals[i][c3_x];
      y4= sub_vals[i][c3_y];
      z1= z2= z3= z4= z_positions_1[n];

      sub_corners= new double[]{x1, y1, z1, x2, y2, z2, x3, y3, z3, x4,
      y4, z4};

      //calculate average locations for the plane 'origin'
      x_avg= (x1+x2+x3+x4)/4.0;
      y_avg= (y1+y2+y3+y4)/4.0;
      z_avg= (z1+z2+z3+z4)/4.0;
      plane_avg= new double[]{x_avg, y_avg, z_avg};

      //create axial constrained plane cPS_ax for each subchannel in the
      //geometry
      ArbitrarySection cPS_ax =
         (ArbitrarySection) sim.getPartManager().
```

```
            createArbitraryImplicitPart(new NeoObjectVector(new Object[]
            {region_00}), new NeoObjectVector(new Object[] {}),
            resolvePath("/home/gilklind/star/NMV/Dakota_Files/SS_Simulations/
            KOM/extract_information/stl_files/file"+(i+1)+"_"+(n+1)+".stl"),
            units_0, true);

         // Names of everything
         ax_con_name1= new String(String.format("%.3f",z_positions_1[n]));

         // Make a report based off of it :

         AreaAverageReport SC_AverageReport =
            sim.getReportManager().createReport(AreaAverageReport.class);

         SC_AverageReport.getParts().setObjects(cPS_ax);

         // Cp
         SC_AverageReport.setScalar(primitiveFieldFunction_cp);
         Double cp_1 = SC_AverageReport.getReportMonitorValue();

         // Temperature
         SC_AverageReport.setScalar(primitiveFieldFunction_temp);
         SC_AverageReport.setUnits(units_temp);
         Double temp_1 = SC_AverageReport.getReportMonitorValue();

         // Density
         SC_AverageReport.setScalar(primitiveFieldFunction_rho);
         Double rho_1 = SC_AverageReport.getReportMonitorValue();

         // Velocity K
         SC_AverageReport.setScalar(vel_k);
         Double Vz_1 = SC_AverageReport.getReportMonitorValue();

         // Area
         FrontalAreaReport SC_AreaReport =
            sim.getReportManager().createReport(FrontalAreaReport.class);
         SC_AreaReport.getParts().setObjects(cPS_ax);
         Double A_1 = SC_AreaReport.getReportMonitorValue();


         // Print this to star_output.log and then delete the report.
         sim.println(ax_con_name1+ " " + (i+1) + " " + cp_1 + " " + temp_1 +
         " " + rho_1 + " " + A_1 + " " + Vz_1  );

         bwout.write( temp_1+ " T" + (i+1) + "\n");

         sim.getReportManager().removeObjects(SC_AverageReport);

         sim.getPartManager().removeObjects(cPS_ax);
      }

   }

   bwout.close();

// Pressure Drop Report

 PressureDropReport pressureDropReport =
   sim.getReportManager().
   createReport(PressureDropReport.class);
```

```
    Boundary boundary_inlet =
      region_00.getBoundaryManager().getBoundary("Inlet");

   Boundary boundary_outlet =
      region_00.getBoundaryManager().getBoundary("Outlet");

    pressureDropReport.getParts().setObjects(boundary_inlet);

    pressureDropReport.getLowPressureParts().setObjects(boundary_outlet);

    pressureDropReport.printReport();

   } catch (IOException iOException) {
   }
}
}
```

APPENDIX B: UQ SCRIPTS

For the following scripts, the input parameters (inlet temperature, total heat, bundle width) have been changed to representative values for illustrative purposes. The parameters do not reflect actual experimental values.

B.1 Mass Flow Rate Matlab Script

```
close all
clear all
clc

%%%%%%%%%% Matlab Profiles for UQ Demo %%%%%%%%%%
% Lindsay Gilkey - 5/4/17

%% Mass Flow rate:
%% This is used if you wish to perturb parameters around various input parameters
for test_index=1
    % Inlet temp in F
    T_all=[500];
    % Convert to K
    T=(T_all(test_index)-32)*5/9 + 273.15;
    % AFLUX
    AFLUX=[4.0];
    % Power per Rod
    PR=AFLUX(test_index) * 1000 / 0.3048 * 3 * 1.0;

for index = 1:2
    close all

% In STAR-CCM+, for a mass flow inlet, total mass flow rate over the inlet
% is specified, and STAR uses inlet temp specs & density to set velocities.
% By setting a temperature profile in x,y at the inlet, this is the easiest
% way to vary the mass flow rate profile.

% mdot = A*V*rho, where rho=function(T^4).

% Nominal Temperature:
T = T ;   % K

% Make x and y coordinates
x1 = [-0.1:0.6/1000:0.1] ; % m
y1 = x1 ;

x = [1:length(x1)] ;     % index
y = x ;                  % index

% I will use two polynomials, one in x and one in y.

% Make the x and y coefficients

% Order of Polynomial
ord = 2 ;

cx=[] ;
cy=[] ;

% Generate the random coefficients
for i = 1:(ord+1)
    cx = [cx, rand-0.5] ;
    cy = [cy, rand-0.5] ;
end

% Evaluate the polynomial functions
TX = polyval(cx,x) ;
TY = polyval(cy,y) ;

% Give the values from ~range +/- .5 K:
Tmin = min([TX,TY]) ;
Tmax = max([TX,TY]) ;
Tdiff = Tmax - Tmin ;
n = 1 / Tdiff ;

% Shift the Temperature values up by random value so they are in range of
% the nominal val.
TX = (TX*n + T) ;
TY = (TY*n + T) ;

% Account for effects of X and Y polynomials by summing their components.
% This will make temperature a "smooth" matrix.

A=TX'*TY./T;

% Make the table:
% this reshape prints out A, going column by column and then creates a meshgrid.

A2 = reshape(A,length(x)^2,1) ;

[x2,y2] = meshgrid(x1,y1) ;

x2 = reshape(x2,length(x)^2,1) ;
y2 = flip(reshape(y2,length(x)^2,1)) ;

z2 = ones(size(x2))*0 ;

% Remove points that appear in the rods.
theta=-pi:.01:pi;
% Changed from actual
r=0.3/2;

posx=[1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5];
posy=[1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5];

% Changed from actual
po=[-1,-.6,0,.6,1]*0.0254;

for i=1:25
    ix=posx(i);
    iy=posy(i);
    xv=r*sin(theta)+po(ix);
    yv=r*cos(theta)+po(iy);

    [in,on] = inpolygon(x2,y2,xv,yv);

    x2=x2(~in);
    y2=y2(~in);
    A2=A2(~in);
    z2=z2(~in);

end
```

```
% Changed from actual
xlimit=0.7/2;
xv=[-xlimit,xlimit,xlimit,-xlimit,-xlimit];
yv=[xlimit,xlimit,-xlimit,-xlimit,xlimit];


[in,on] = inpolygon(x2,y2,xv,yv);

    x2=x2(in);
    y2=y2(in);
    A2=A2(in);
    z2=z2(in);

% Rescale the matrix to ensure that the average temperature is unchanged.
n = mean(A2)/T ;

A2 = A2/n ;


% Make representative plots
scatter(x2,y2,1,A2)

% set the plotting box to a square.
pbaspect([1 1 1])

colorbar

% Plot the rods as black outlines.
plot(xv,yv,'- black')

title(['Heat Flux ',num2str(test_index),'_ ',num2str(index)])
saveas(gcf,['Temp_xy_',num2str(test_index),'_',num2str(index),'.png'])

tempxyz = [A2,x2,y2,z2]' ;

% Print out the table file:
f = fopen(['Temp_xy_',num2str(test_index),'_',num2str(index),'.txt'],'w') ;
fprintf(f, '"Temperature (K)","X (m)","Y (m)","Z (m)"')
fprintf(f, '\r%f,%f,%f,%f',tempxyz)
fclose(f)

end
```

B.2 Mass Flow Rate STAR Java Macro

```
// STAR-CCM+ macro: ref.java
// Written by STAR-CCM+ 11.02.009
package macro;

import java.util.*;

import star.common.*;
import star.base.neo.*;
import star.energy.*;

public class star_tempprofile extends StarMacro {

  public void execute() {
    execute0();
  }
  private void execute0() {

   int vel_prof=0;

   if (vel_prof == 0 ) {

   Simulation simulation_0 =
      getActiveSimulation();
   }

   else {

   Simulation simulation_0 =
      getActiveSimulation();

    Region region_0 =
      simulation_0.getRegionManager().getRegion("FV");

    Boundary boundary_0 =
      region_0.getBoundaryManager().getBoundary("Inlet");

    boundary_0.setBoundaryType(MassFlowBoundary.class);

    TotalTemperatureProfile totalTemperatureProfile_0 =
      boundary_0.getValues().get(TotalTemperatureProfile.class);

    totalTemperatureProfile_0.setMethod(XyzTabularScalarProfileMethod.class);

    XyzInternalTable xyzInternalTable_0 =
      simulation_0.getTableManager().createTable(XyzInternalTable.class);

    FileTable fileTable_0 =
      (FileTable) simulation_0.getTableManager().
      createFromFile(resolvePath("../../Temp_profiles/Temp_xy_1_0.txt"));

    totalTemperatureProfile_0.getMethod(XyzTabularScalarProfileMethod.class).
      setTable(fileTable_0);

    totalTemperatureProfile_0.getMethod(XyzTabularScalarProfileMethod.class).
      setData("Temperature");

    }
  }
}
```

B.3 Heat Flux Profile Matlab Script

```
close all
clear all
clc

%%%%%%%%%% Matlab Profiles for UQ Demo %%%%%%%%%%
% Lindsay - 5/4/17

%% Mass Flow rate:

for test_index=1
    % Inlet temp in F
    T_all=[500];
    % Convert to K
    T=(T_all(test_index)-32)*5/9 + 273.15;
    % AFLUX
    % Changed from actual
    AFLUX=[4.0];
    % Power per Rod
    PR=AFLUX(test_index) * 1000 / 0.3048 * 3 * 1.0;
%% Heat Flux


close all
format long

% The variable read by STAR is total Power. Easiest way to set this is to
% integrate a heat flux function over z, and then set power as a function
% of z.

% Changed from actual
P=PR*0.9;


for index = 1:2
    hold all

% Length
L = 3 ; % m
z = [0:0.6/1000:L];
w = 2.0*pi/L ;
rd = z*w ;

% Radius, changed from actual
r = 0.3/2*0.0254 ; % m

% Area Factor
C = 1.0055 ;

% Rod Heat Flux
area = 2*pi*r*L ;
HF = P/area*C;% ;
B=HF/C;

A = rand*HF;
theta = pi;

% Make a sine function
HF1 = A.*cos(z*w*1.0+theta)+B ;

% Numerically integrate heat flux function
a=cumtrapz(z,HF1*2*pi*r*C);
a=a(end);

vector = [A;C;theta;HF];

close all
plot(z,HF1)

% a=cumtrapz(z,HF1*2*pi*r);
% a=a(end);
```

```
title({['Heat Flux Profile ',num2str(test_index),'_ ',num2str(index),': For Rod
1'], ...
    ['Total Power = ', num2str(P), ' W'],...
    ['Integrated Power = ',num2str(a), 'W']})

grid on
xlabel('Z Position, m')
ylabel('Heat Flux, W/m^2')

saveas(gcf,['HeatFluxProfile_',num2str(test_index),'_',num2str(index),'.png'])

% Make the java macro files, this step is omitted but was done with fprintf.

hf=fopen(['star_heatflux_',num2str(test_index),'_',num2str(index),'.java'],'w')

fprintf()

fclose(hf)

end

end
```

B.4 Heat Flux Java Macro

All coefficients have been changed from their actual values.

```
// STAR-CCM+ macro: ref.java

// Written by STAR-CCM+ 11.02.009
package macro;

import java.util.*;

import star.common.*;
import star.energy.*;

public class star_heatflux_1_1 extends StarMacro {

  public void execute() {
    execute0();
  }

  private void execute0() {

    Simulation simulation_0 =
      getActiveSimulation();

    Region region_0 =
      simulation_0.getRegionManager().getRegion("FV");


   int i=0;
   // Changed from actual
   double[] P =
   {0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,
   1.1,1.1,1.1,1.1,1.1,1.1};
   while (i<25) {

      Boundary boundary_0 =
         region_0.getBoundaryManager().getBoundary("Rod " + (i+1));

      boundary_0.getConditions().get(WallThermalOption.class)
         .setSelected(WallThermalOption.Type.HEAT_FLUX);

      HeatFluxProfile heatFluxProfile_0 =
         boundary_0.getValues().get(HeatFluxProfile.class);

      heatFluxProfile_0.setMethod(FunctionScalarProfileMethod.class);


      // Create the user field function for the heat profile

      UserFieldFunction heatFlux_function =
         simulation_0.getFieldFunctionManager().createFieldFunction();

      heatFlux_function.getTypeOption()
         .setSelected(FieldFunctionTypeOption.Type.SCALAR);

      heatFlux_function.setFunctionName("HeatFlux_"+(i+0));

      // Changed from actual
      double PR = 50000*P[i];

      double C = 1.005500000000000;
      double A = 27000;
      double w = 2.094395102393195;
      double theta = 3.141592653589793;;

      // Changed from actual
      double area =0.1;
      double B = PR/area;

      heatFlux_function.setDefinition("(" + A + "*cos(" + w +
      "*$${Position}[2]+" + theta + ")+" + B + ")*"+C);

      heatFluxProfile_0.getMethod(FunctionScalarProfileMethod.class)
         .setFieldFunction(heatFlux_function);

      i++;


   }
  }
}
```