Validation of Conjugate Heat-Transfer Capability for Water-Cooled High-Speed Flows

Bruno R. Fletcher

Embry-Riddle Aeronautical University - Daytona Beach

Follow this and additional works at: https://commons.erau.edu/db-theses

Part of the Aerospace Engineering Commons

Scholarly Commons Citation
https://commons.erau.edu/db-theses/62

This thesis is brought to you for free and open access by Embry-Riddle Aeronautical University – Daytona Beach at ERAU Scholarly Commons. It has been accepted for inclusion in the Theses - Daytona Beach collection by an authorized administrator of ERAU Scholarly Commons. For more information, please contact commons@erau.edu.
Validation of Conjugate Heat-Transfer Capability for Water-Cooled High-Speed Flows

By

Bruno R. Fletcher, B.S.

Thesis
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.
VALIDATION OF CONJUGATE HEAT-TRANSFER CAPABILITY
FOR WATER- COOLED HIGH-SPEED FLOWS

by

Bruno R. Fletcher

This thesis was prepared under the direction of the candidate's thesis committee co-chairman, Dr. William Engblom, Department of Mechanical Engineering, and approved by the members of the candidate's thesis committee. It was submitted to the Aerospace Engineering Department and was accepted in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Engineering.

THESIS COMMITTEE:

Dr. William Engblom
Co-Chairman

Dr. Eric Perrell
Co-Chairman

Dr. Ray Mankbadi
Member

Dr. Yi Zhao
Graduate Program Coordinator

Dr. Habib Eslami
Department Chairman

Dr. Chris Grant
Dean

Dr. Richard Heist
Provost

9/24/07 Date
Acknowledgments

I would like thank Dr. William Engblom, full-time professor at Embry-Riddle Aeronautical University, for giving me the opportunity to continue the development of FOGO throughout my graduate studies. Thank you for all your insight, support, and infinite willingness to help.

I also sincerely thank Dr. Perrell, full-time professor at Embry-Riddle Aeronautical University, for coming up with the numerical scheme present in the FOGO conduction model, and assigning it initially as a class project. You have truly been responsible for the kick-off of this research. Thank you for all your expertise and advice.

I would also like to acknowledge the participation of Brian Ruby and Matt Houston in the initial development of FOGO; they were my partners in Dr. Perrell's class assignment. You guys were extremely helpful. Thanks!

Thank you Dr. Nicholas Georgiadis, Dr. Chris Nelson, Dr. Greg Power, and all members of the NPARC alliance for all your contributions to this research.

I would like to thank the committee members, Dr. William Engblom, Dr. Perrell, and Dr. Mankbadi for dedicating their time reading and evaluating this work.

I would like to acknowledge that this work was funded under a grant from the Air Force at Arnold Engineering Development Center (AEDC) and in support of a project led by NASA Glenn Research Center.
For my Wife, my Dad, my Mom, my Sister, and the Zobaran family.

“Great discoveries and improvements invariably involve the cooperation of many minds. I may be given credit for having blazed the trail, but when I look at the subsequent developments I feel the credit is due to others rather than to myself.”

Alexander Graham Bell
Abstract

A FORTRAN conduction/convection heat transfer module named FOGO (meaning fire in Portuguese) has been developed to perform the conjugate heat transfer analyses with existing flow solvers. In this work, FOGO is coupled with TBD and WIND-US flow solvers. WIND-US is a well validated flow solver developed by NASA, AEDC, and Boeing. TBD is a new flow solver developed at ERAU. TBD is verified for laminar stagnation point heat transfer prediction and turbulent flowfield prediction. The FOGO module is verified against an exact solution for transient one-dimensional heat flow in a semi-infinite solid.

The coupled TBD-FOGO and WIND-FOGO conjugate heat-transfer solvers are validated against existing data for three high speed water-cooled wind tunnel experiments: i) Air Force high-enthalpy nozzle test which recorded the increase in water coolant temperature, ii) Jet Propulsion Lab converging-diverging nozzle test for which nozzle and run-up were well-instrumented with thermocouples, iii) NASA Glenn Research Center experiment on a series of water cooled panels subject to heating from an impinging rocket engine plume. Results for all three applications are directly compared to recently published attempts by other researchers.

Also included in this work is a set of instructions detailing how to couple FOGO to other existing flow solvers to perform similar conjugate heat transfer analyses.
Table of Contents

Acknowledgments ................................................................. i
Abstract ................................................................................... iii
List of Figures ........................................................................ vii
List of Tables ........................................................................... ix
Nomenclature ............................................................................ x
1 Introduction ................................................................. 1
   1.1 Background ...................................................................... 1
   1.2 Objectives ...................................................................... 1
2 Methodology ........................................................................... 2
   2.1 Coupling Approach .......................................................... 2
   2.2 FOGO Theory and Numerical Scheme ............................... 3
      2.2.1 Conduction Model ..................................................... 3
      2.2.2 Coolant Channel Model ............................................ 7
      2.2.3 Conduction Module Verification ................................. 12
   2.3 Flow Solver Description .................................................. 14
      2.3.1 TBD Flow Solver Description ..................................... 14
      2.3.2 TBD Verification and Validation ............................... 15
      2.3.3 Wind-US Flow Solver .............................................. 17
3 Results .................................................................................. 18
   3.1 Case #1 (AEDC Water-Cooled Nozzle) ....................... 18
   3.2 Text Case #2 (JPL Water-Cooled Nozzle) ....................... 24
   3.3 Case #3 (NASA GRC Water-Cooled Thermal Panel within Rocket Plume) ..................................................................................................................... 26
   3.4 Conclusions ................................................................. 32
4 FOGO Module Description ................................................... 34
   4.1 Object-Oriented Approach ............................................. 34
   4.2 Module Dependency and Hierarchy ................................. 35
   4.3 Compiling FOGO .......................................................... 37
   4.4 Required FOGO Inputs .................................................. 38
   4.5 Attaching FOGO to Existing Flow Solvers ....................... 38
      4.5.1 Importing the FOGO Module .................................. 40
      4.5.2 Initializing FOGO .................................................... 40
      4.5.3 Estimating the Heat Flux ......................................... 41
List of Appendices

Appendix A: FOGO Modules .................................................................45
Appendix B: “main.f90” Sample File .......................................................47
Appendix C: FOGO.inp File for Case 1, Run 1 ...........................................49
Appendix D: FOGO.inp File for Case 2 .....................................................51
Appendix E: FOGO.inp File for Case 3 .....................................................53
List of Figures

Figure 1: Tight Coupling of Solvers.................................................................2
Figure 3: Control Volume Sketch (cell depth ζ-direction not shown for simplicity)........6
Figure 4: Horizontal chrome wire with 0.10 cm diameter heated in water at 1 atm [ref 4]..9
Figure 5: Transient Heat Verification, Semi-Infinite Solid with Constant Wall Temp........12
Figure 6: Transient Heat Verification, Semi-Infinite Solid with Constant Heat Flux........13
Figure 7: Temp. Contour in Shock Layer and Heat Flux Variation Along Surface...........16
Figure 8: Schematic of Burrow-Kurkov Supersonic Combustion Experiment.................16
Figure 9: Stagnation Temperature Contours for Burrow-Kurkov Case........................16
Figure 10: Combustor Exit Profiles for Burrow-Kurkov Case................................17
Figure 11: AEDC Nozzle Geometry......................................................................18
Figure 12: Predicted Water Coolant Temperature Rise during Run #2.........................20
Figure 13: Coolant Heat Flux Distribution for Run #2...........................................21
Figure 14: Predicted Air-side Nozzle Surface Temperature Distributions for Case#1 (TBD-FOGO)........................................................................................................23
Figure 15: Nozzle Temperature Distribution for JPL Experiment...............................24
Figure 16: Case #2 Run-up and Nozzle Grid (every other point removed; Blue: flow; Black: solid)................................................................................................................25
Figure 17: Nozzle Inner-wall Surface Temperatures from TBD-FOGO vs. Experiment....25
Figure 18: Nozzle Inner-wall Surface Temperatures from WIND-FOGO vs. Experiment..26
Figure 19: Test Section Cutaway View.....................................................................27
Figure 20: Numerical Representation of Calorimeter Geometry.................................27
Figure 21: Calorimeter Panel – cross section..........................................................27
Figure 23: Heat Flux Profile along Calorimeter Surface for Case #3 Simulation with TBD-FOGO (multi-specie; no chemical reactions)................................................29
Figure 24: Water and Air-side Calorimeter Surface Temperatures for Case #3 Simulation with TBD-FOGO........................................................................................................30
Table 7: TBD-FOGO Predicted vs. Experimental Water Coolant (multi-specie; chemically-reacting flow)..........................................................................................31
Figure 25: Static Temperature and Mach Contours with Wind-FOGO (multi-specie; no chemical reactions).........................................................................................31
Figure 26: Heat Flux Profile along Calorimeter Surface with Wind-FOGO (multi-specie; no chemical reactions).........................................................................................32
Table 8: WIND-FOGO Predicted vs. Experimental Water Coolant (multi-specie; no...
List of Tables

Table 1: AEDC Nozzle Run Conditions..............................................................18
Table 2: Comparison of Coolant Bulk Temperature Rise from Experiment with TBD- FOGO with only Nucleate Boiling.................................................................19
Table 3: Comparison of Coolant Bulk Temperature Rise from Experiment with TBD- FOGO with Nucleate and Film Boiling.................................................................20
Table 3a: Comparison of Coolant Bulk Temperature Rise from Experiment with WIND- FOGO and Nucleate Boiling.............................................................................22
Table 3b: Comparison of Coolant Bulk Temperature Rise from Experiment with WIND- FOGO and Nucleate/Film Boiling......................................................................22
Table 4: Run Conditions for JPL Experiment (coolant flow details unknown)........24
Table 5: Nozzle Inflow Stagnation Conditions for Simulation of Case #3 (Run No. 95).....28
Table 6: TBD-FOGO Predicted vs. Experimental Water Cooler (multi-specie; no chemical reactions).........................................................................................30
**Nomenclature**

**English Symbols**

- $c$: specific heat (incompressible)
- $C_{sf}$: surface-fluid constant
- $D$: hydraulic diameter
- $e$: specific energy
- $f$: friction factor
- $F,G,H$: heat flux component along $x,y,z$ directions
- $F',G',H'$: heat flux component along $(\xi, \eta, \zeta)$ directions
- $\vec{F}$: fluid flow flux vector
- $g$: gravity
- $h$: specific enthalpy
- $h_c$: heat transfer coefficient
- $k$: thermal conductivity
- $K$: turbulence kinetic energy
- $m$: mass flow rate
- $\vec{N}$: airflow source term vector
- $Nu$: Nusselt Number
- $n$: unit vector
- $p$: pressure
- $Pr$: Prandtl Number
- $q$: heat flux
- $Re$: Reynolds Number
- $S$: flow solver's cell face area
- $T$: temperature
- $\nu$: volume
Greek Symbols

\( \varepsilon \) rate of the dissipation of \( K \)

\( \mu \) dynamic viscosity

\( \sigma \) surface tension

\( \rho \) density

\( \tau \) shear stress

\( \xi, \eta, \zeta \) Curvilinear body-fitted axis, pointing in the \( i,j,k \) directions, respectively

Subscripts

\( b \) bulk

\( i \) inlet (also specie number)

\( e \) exit

\( f \) film boiling

\( fg \) vaporization

\( n \) nucleate boiling (also contravariant)

\( o \) stagnation (also total)

\( sat \) saturation

\( w \) wall

\( \infty \) freestream

\( \text{init} \) initial

\( l,v \) saturated liquid, saturated vapor

\( vf \) saturated vapor at film temperature
1 Introduction

1.1 Background

Aerodynamic heating plays an important role in high speed flows where high temperatures are often encountered. In this kind of environment, an understanding of how the flow field affects the temperature distribution of an immersed solid is crucial to its design due the possibility of thermal fatigue/failure. Engineering an aerospace vehicle capable of withstanding such thermal loads can be expensive due to high costs associated with wind tunnel experiments and flight testing. To decrease these costs and to better understand the phenomena, scientists and engineers can use computer based multi-physics packages to simulate the heat transfer between the flow field and the immersed solid.

A conjugate heat transfer analysis is defined as a numerical simulation in which the heat transfer is modeled between solid and fluid interfaces. For high-speed applications, this capability more specifically includes both a flow solver which can accurately simulate high enthalpy compressible, turbulent, and possibly reacting flows, plus a solid body heat conduction solver which includes proper treatment of coolant flows and nucleate boiling effects.

1.2 Objectives

The main objective is to develop a capability to accurately simulate conjugate heat transfer for water-cooled, high-speed applications, within a RANS (Reynolds-averaged Navier-Stokes) framework. A conjugate heat transfer module named FOGO (meaning fire in Portuguese) has been developed to enable such simulations. Features include coolant flow simulation, multi-block heat conduction capability, and uses CGNS (CFD General Notation System ) for input and output. Another goal is to develop the heat conduction module, and its related coolant treatment, as a distinct module with minimal “hooks” to the flow solver, so as to easily attach to any existing structured CFD code for loosely coupled conjugate heat transfer computations. For example, the heat conduction module (FOGO) uses the CGNS (CFD General Notation System) file system – an AIAA recommended practice to improve portability. A secondary goal is to investigate means to improve the robustness and efficiency of the coupled computations.
2 Methodology

2.1 Coupling Approach

FOGO is written in FORTRAN 90/95 and is designed to attach to any existing flow solver to perform a coupled conjugate heat transfer analysis. The term “loosely coupled” refers to the situations that in which the boundary conditions at the fluid/solid interface wall are updated every N-iterations. For “tightly coupled” simulations the boundary condition information is passed every iteration (see Figure 1).

![Diagram of Tight Coupling of Solvers](image)

**Figure 1:** Tight Coupling of Solvers
2.2 FOGO Theory and Numerical Scheme

2.2.1 Conduction Model

FOGO uses an explicit finite volume scheme with multi-block capability for the heat conduction module. The derivatives in this scheme are represented by second order accurate in space and first order accurate in time discretizations. The underlying heat conduction algorithm implemented in FOGO was originally developed by Perrell [ref 1].

The conduction model is based on the general three-dimensional heat-conduction equation in Cartesian coordinates with no internal heat generation:

\[
\frac{\partial}{\partial x} \left( k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( k \frac{\partial T}{\partial z} \right) = \rho c \frac{\partial T}{\partial t}
\]  

(1)

This form admits numerical solution with non-constant material properties \(k, \rho,\) and \(c\).

FOGO solves the heat conduction equation using a general body-fitted coordinate system. The general body-fitted coordinate system \((\xi, \eta, \zeta)\) is used instead of the Cartesian coordinates \((x, y, z)\) to avoid having to derive special differencing equations for boundaries on non-rectangular geometries. The body we will analyze is discretized into cells (or control volumes). Figure 1 shows the discretization of a given solid using the body-fitted coordinate system.

![Body-fitted Coordinate System and the Resulting Computational Domain](image)

Figure 2: Body-fitted Coordinate System and the Resulting Computational Domain
One of the main advantages of using the body-fitted coordinate system is that each cell element located at a \((\xi, \eta, \zeta)\) position can be easily mapped to a corresponding \((i,j,k)\) element in a computational array/block, as shown on the cell center "*" on Figure 1. This facilitates the “book-keeping” process of all cell element properties during the simulation and is simpler to implement when compared with other discretization methods such as unstructured grids. The grid spacing in the computational domain is also conveniently chosen to have a unit spacing \((\Delta\xi=1, \Delta\eta =1, \Delta\zeta= 1)\) which simplifies discretization of any spatial derivative.

In order to use body-fitted coordinates, the general heat conduction equation (Equation 1) must be modified to account for a coordinate transformation. Since the coordinates of the nodes of the body can be expressed in both body-fitted and Cartesian coordinates, we can say that:

\[
\begin{align*}
\xi &= \xi(x, y, z) \\
\eta &= \eta(x, y, z) \\
\zeta &= \zeta(x, y, z)
\end{align*}
\] (2)

Then using the chain rule from Calculus we can estimate the derivatives in \((x,y,z)\) space in terms of the \((\xi,\eta,\zeta)\) space:

\[
\begin{align*}
\frac{\partial}{\partial x} &= \frac{\partial}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial}{\partial \eta} \frac{\partial \eta}{\partial x} + \frac{\partial}{\partial \zeta} \frac{\partial \zeta}{\partial x} \\
\frac{\partial}{\partial y} &= \frac{\partial}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial}{\partial \eta} \frac{\partial \eta}{\partial y} + \frac{\partial}{\partial \zeta} \frac{\partial \zeta}{\partial y} \\
\frac{\partial}{\partial z} &= \frac{\partial}{\partial \xi} \frac{\partial \xi}{\partial z} + \frac{\partial}{\partial \eta} \frac{\partial \eta}{\partial z} + \frac{\partial}{\partial \zeta} \frac{\partial \zeta}{\partial z}
\end{align*}
\] (3)

Here the partial derivatives terms of the \((\xi,\eta,\zeta)\) space variables with respect to the \((x,y,z)\) are known as the metric derivatives.
Using short hand notation for the metric derivatives, the partial derivatives in Equation 3 can be rewritten as follows:

\[
\frac{\partial ()}{\partial x} = \frac{\partial ()}{\partial \xi} \xi_x + \frac{\partial ()}{\partial \eta} \eta_x + \frac{\partial ()}{\partial \xi} \xi_x
\]

\[
\frac{\partial ()}{\partial y} = \frac{\partial ()}{\partial \xi} \xi_y + \frac{\partial ()}{\partial \eta} \eta_y + \frac{\partial ()}{\partial \xi} \xi_y
\]

\[
\frac{\partial ()}{\partial z} = \frac{\partial ()}{\partial \xi} \xi_z + \frac{\partial ()}{\partial \eta} \eta_z + \frac{\partial ()}{\partial \xi} \xi_z
\]

Then the F, G, and H heat fluxes in the x,y,z direction are computed by:

\[
F = k \frac{\partial T}{\partial x} = \left( \xi_x \frac{\partial T}{\partial \xi} + \eta_x \frac{\partial T}{\partial \eta} + \xi_x \frac{\partial T}{\partial \xi} \right)
\]

\[
G = k \frac{\partial T}{\partial y} = \left( \xi_y \frac{\partial T}{\partial \xi} + \eta_y \frac{\partial T}{\partial \eta} + \xi_y \frac{\partial T}{\partial \xi} \right)
\]

\[
H = k \frac{\partial T}{\partial z} = \left( \xi_z \frac{\partial T}{\partial \xi} + \eta_z \frac{\partial T}{\partial \eta} + \xi_z \frac{\partial T}{\partial \xi} \right)
\]

The components of the cell face heat fluxes in the \( \xi, \eta, \zeta \) directions, also known as contravariant heat fluxes, are obtained by using the metric unit derivatives \( \hat{\xi}, \hat{\eta}, \hat{\zeta} \) :

\[
F' = \hat{\xi}_x F + \hat{\xi}_y G + \hat{\xi}_z H
\]

\[
G' = \hat{\eta}_x F + \hat{\eta}_y G + \hat{\eta}_z H
\]

\[
H' = \hat{\zeta}_x F + \hat{\zeta}_y G + \hat{\zeta}_z H
\]

The general heat conduction equation in body-fitted coordinates can be expressed as:

\[
\frac{\partial F'}{\partial \xi} + \frac{\partial G'}{\partial \eta} + \frac{\partial H'}{\partial \zeta} = \rho c \frac{\partial T}{\partial t}
\]
We want to solve Equation 7 using a finite volume approach. Applying the divergence theorem we first integrate the left hand side of Equation 7:

\[ \int_{cs} F' \hat{n} dS = \int_{cv} \nabla \cdot F' dV \]

\[ = \int_{cv} \left( \frac{\partial F'}{\partial \xi} + \frac{\partial G'}{\partial \eta} + \frac{\partial H'}{\partial \zeta} \right) dV \]  \hspace{1cm} (8)

Integrating the right hand side of Equation 7 over the control volume yields \( \rho c \dot{T} \) as we assume these variables are taken as cell centered averages. The left hand side of Equation 8 is just the vector summation of the heat fluxes in the \( \xi, \eta, \zeta \) directions times the corresponding face area of the control volume.

Figure 3: Control Volume Sketch (cell depth \( \zeta \)-direction not shown for simplicity)

By looking at the control volume sketched in Figure 3, we can obtain the vector summation of the heat
fluxes crossing the control surface and entering/leaving the control volume:

\[
\int_{CS} \vec{F} \cdot \hat{n} \, dS = F'_{i+1,j,k} A_{i+1,j,k} - F'_{i,j,k} A_{i,j,k} + G'_{i+1,j,k} A_{i+1,j,k} - G'_{i,j,k} A_{i,j,k} + H'_{i,j,k+1} A_{i,j,k+1} - H'_{i,j,k} A_{i,j,k}
\] (9)

Which yields the integral form of Equation 7:

\[
\int_{CV} \rho c \bar{T} \, dV = F'_{i+1,j,k} A_{i+1,j,k} - F'_{i,j,k} A_{i,j,k} + G'_{i+1,j,k} A_{i+1,j,k} - G'_{i,j,k} A_{i,j,k} + H'_{i,j,k+1} A_{i,j,k+1} - H'_{i,j,k} A_{i,j,k}
\] (10)

Discretizing the time derivative and treating the terms within the differential volume as constants, we solve for the temperature at the next time step \((n+1)\) to obtain the explicit scheme for the conduction module:

\[
T_{i,j,k}^{n+1} = T_{i,j,k}^n + \frac{\Delta t}{\rho c V_{i,j,k}} \left[ F'_{i+1,j,k} A_{i+1,j,k} - F'_{i,j,k} A_{i,j,k} + G'_{i+1,j,k} A_{i+1,j,k} - G'_{i,j,k} A_{i,j,k} + H'_{i,j,k+1} A_{i,j,k+1} - H'_{i,j,k} A_{i,j,k} \right]
\] (11)

Using the above equation we can estimate the cell centered temperature variation of a given volume element once all cell face heat fluxes have been computed.

### 2.2.2 Coolant Channel Model

FOGO treats the coolant flow as a 1-D channel with a bulk temperature and velocity. The coolant flow is discretized as a single layer of cells with potentially varying thickness and exchange heat with the solid body via forced convection and boiling.

Newton's Law of Cooling states that:

\[
q_c = h_c (T_w - T_b)
\] (12)

Where "\(q_c\)" is the convective heat flux, "\(T_b\)" is the bulk temperature of a given coolant cell, and "\(T_w\)" is the wall temperature of the coolant channel. The convection coefficient "\(h_c\)" depends on the Nusselt number. FOGO utilizes the Bhatti and Shah correlation [ref 2] that predicts the local Nusselt number for rough walled channels:

\[
Nu_B = \frac{(f/8) Re_d Pr}{1 + \sqrt{f/8(4.5 Re_d^0.2 Pr^{0.5} - 8.48)}}
\] (13)
Here “Re₀” is the local Reynolds number based on the hydraulic diameter of channel, “Reₑ” is the roughness Reynolds number which is a function of the roughness of the channel wall and its hydraulic diameter; “f” is the friction factor. Equation 13 applies for the ranges:

\[ 10^4 \leq Re_D, \ 0.5 \leq Pr \leq 10, \ 0.002 \leq \frac{\varepsilon}{D} \leq 0.05 \]

The corresponding friction factor equation is computed using Haaland's equation:

\[
f = \frac{1}{1.8 \log_{10} \left[ \frac{6.9}{Re_D} + \left( \frac{\varepsilon/D}{3.7} \right)^{11} \right]^2} \]  

(14)

For ranges where Equation 13 is no longer applicable, FOGO utilizes the Petukhov/Gnielinski correlation [ref 3] for the Nusselt number, which is valid for turbulent flows in smooth pipes in the range of \( 2300 \leq Re_D \leq 5 \times 10^6 \).

\[
Nu_D = \frac{(f/8)(Re_D - 1000) Pr \left( \frac{\mu_b}{\mu_w} \right)^n}{1 + 12.7 \sqrt{f/8 (Pr^{2/3} - 1)}} 
\]

(15)

The \( \mu_b \) and \( \mu_w \) variables represent the coolant viscosity at bulk and wall temperatures. For liquids, over the interval \( 0.025 \leq (\mu_b/\mu_w) \leq 12.5 \), the variable \( n \) is determined by:

\[
\begin{align*}
n & = 0.11 \ for \ T_w > T_b \\
n & = 0.25 \ for \ T_w < T_b 
\end{align*}
\]

(16)

The friction factor for a smooth pipe is given by:

\[
f = \frac{1}{\left[ 1.82 \log_{10} Re_D - 1.64 \right]^2} \]

(17)
Once the Nusselt number is known the convection heat transfer coefficient can be calculated:

\[
h_c = \frac{k \text{Nu}_d}{D}
\]  
(18)

When the wall temperature of the channel is larger than the saturation temperature \( T_{\text{sat}} \) of the coolant, boiling occurs. Boiling is directly dependent on this excess temperature, \( \Delta T = T_w - T_{\text{sat}} \).

Figure 4: Horizontal chrome wire with 0.10 cm diameter heated in water at 1 atm [ref 4]

FOGO takes into account the nucleate boiling and film boiling regions. The boiling free convection regime is not modeled due to its negligible effect on heat transfer and small excess temperature variation. FOGO models flow boiling by superimposing pool boiling (stagnant fluid) and forced convection heat transfer components.

During the nucleate boiling regime bubbles appear on the surface of the channel. As we increase the temperature of the wall boiling becomes very intense such that neighboring bubbles merge and vapor bubble columns are formed. These bubbles are very efficient in removing heat from the hot surface since it has a high heat transfer rate for a small excess temperature variation, as illustrated in Figure 4. The
nucleate boiling heat flux is estimated using an empirical relationship developed by Rohsenow for flat plates [ref 4]:

\[
q_n = \frac{\mu}{h_{fg}} \left( \frac{c_p(T_w - T_{sat})}{C_f Pr s} \right)^3 \sqrt{\frac{g(\rho_l - \rho_v)}{\sigma}}
\]  

(19)

Here the coolant properties \( \mu, c_p, h_{fg}, \) and \( Pr \) are evaluated at the saturation temperature. The coefficient \( s \) has the value of 1.0 for water and 1.7 for most other fluids. The variable \( \sigma \) represents the surface tension of the coolant while the experimental parameter \( C_s f \) is a function of the channel's wall material and the type of fluid used.

The nucleate boiling equation is valid for the region where \( T_w > T_{sat} \) and the below the peak heat flux. Once peak heat flux is surpassed we have entered the film boiling region. The peak heat flux is estimated using Zuber's empirical relationship [ref. 5]:

\[
q_{max} = \frac{\pi}{24} h_{fg} \rho_v \left[ \frac{\sigma g(\rho_l - \rho_v)}{\rho_v^2} \right]^{\frac{1}{2}} \left( 1 + \frac{\rho_v}{\rho_l} \right)^{\frac{1}{2}}
\]  

(20)

During film boiling the bubbles form so rapidly that many regions of the solid are covered preventing new fluid to move in and remove the heat load. Since the coolant is unable to remove the heat effectively we see a dramatic increase in wall temperature and decrease in heat flux, and in many cases leading to thermal failure of the solid (burnout failure). Equation (21), which was developed by Bromely [ref. 6 ] for water flow over horizontal pipes, is used to estimate the film-boiling heat transfer flux for horizontal tubes neglecting radiation effects:

\[
q_f = 0.62 (T_w - T_{sat}) \left[ \frac{k_{vf} \rho_{vf}(\rho_l - \rho_{vf}) g (h_{fg} + 0.4c_{p,vf} \Delta T)}{d \mu_{vf} \Delta T} \right]^{\frac{1}{4}}
\]  

(21)

The enthalpy of vaporization “\( h_{fg} \)” is evaluated at saturation temperature while the “\( vf \)” subscript means that the properties are evaluated at the vapor film temperature \( (T_f = 0.5(T_w + T_{sat})) \).
The total heat rate carried by the coolant is calculated by superimposing the convective and boiling terms:

\[ \dot{Q} = (q_c + q_n + q_f) A \]  

(22)

The energy rate balance equation for a control volume is used to estimate the bulk temperature rise of the coolant due to the heat transfer between the solid/coolant zones:

\[ \frac{dE_{cv}}{dt} = \dot{Q} - W + \sum_i m_i \left( h_i + \frac{V_i^2}{2} + gz_i \right) - \sum_e m_e \left( h_e + \frac{V_e^2}{2} + gz_e \right) \]  

(23)

FOGO uses coolant cells as control volumes with the coolant flowing in the positive i-direction and entering and leaving this control volume through one inlet and one exit. The coolant heat exchange occurs through the j-direction faces. Kinetic and potential energy variations are neglected and no external work is applied; the energy rate balance Equation 23 is simplified to:

\[ \frac{dE_{cv}}{dt} = \dot{Q} + m \left( h_i - h_e \right) \]  

(24)

Performing a first order discretization of Equation 24 we can estimate the change of the bulk temperature of a given coolant cell:

\[ T_i^{n+1} = T_i^n + \frac{\Delta t}{c_i \rho_i \nu_i} \left( (q_c + q_n + q_f) A_i + m c_i \left( T_{i-1/2}^n - T_{i+1/2}^n \right) \right) \]  

(25)

The above temperatures at the inlet and exit are upwinded (i.e., taken upstream from the face) to provide stability. Adding artificial damping and/or leapfrog scheme treatment were found to be much less effective at eliminating spurious temperatures.

It is important to note that FOGO interpolates the NIST thermodynamic tables to obtain all coolant
properties (c, ρ, μ, k, Pr) described in this section.

### 2.2.3 Conduction Module Verification

The analytical treatment of transient 1-D heat flow into a semi-infinite solid was used to verify FOGO's conduction model. A semi-infinite solid with initial temperature, \( T_{\text{init}} \), suddenly experiencing a rise in wall temperature, \( T_w \), will experience a transient temperature distribution given by [ref. 5]:

\[
T(x, t) = T_w + \text{erf} \left( \frac{x}{2\sqrt{\alpha t}} \right) (T_{\text{init}} - T_w) \quad \text{(26)}
\]

A tungsten cube with a 1 m\(^3\) volume was used for this verification. A fine uniform mesh of the cube was created with \(1 \times 10^6\) volume elements to minimize the effects of spatial discretization. The temperature at one of the faces of the cube was maintained constant throughout the simulation while all other faces of the cube were insulated to simulate a semi-infinite solid. Initial temperature of the cube was set to 0°C and the isothermal wall temperature to 1000°C at \( x = 0 \). The results for the simulations are presented in Fig. 5 which shows that FOGO produces excellent agreement with the analytic solution (largest temperature variation is 0.0101°C).

![Figure 5: Transient Heat Verification, Semi-Infinite Solid with Constant Wall Temp.](image)
A second verification test for the conduction the module was performed by applying a constant heat flux $q_w$ at one of the faces of the semi-infinite solid. The analytical solution for this problem is [ref. 5]:

$$T(x,t) = T_{ini} + \frac{2q_w}{k} \sqrt{\frac{\alpha t}{\pi}} \exp \left( -\frac{x^2}{4\alpha t} \right) - \frac{q_w x}{k} \left[ 1 - \text{erf} \left( \frac{x}{2\sqrt{\alpha t}} \right) \right]$$

(27)

The same grid and material properties from the previous simulation were used. The initial temperature of the cube was again set to 0°C and the constant heat flux to 80 kW/m$^2$ at $x = 0$. The results for the simulations are presented in Fig. 6 which shows that FOGO has close agreement with the analytic solution of this problem (largest temperature deviation of 0.1658 deg C).

![Figure 6: Transient Heat Verification, Semi-Infinite Solid with Constant Heat Flux](image)

Similar verification tests were performed on all faces of the computational domain (imin, imax, jmin, jmax, kmin, kmax) to verify that the heat gradients were being estimated correctly and to also assert that the solution was symmetric.
2.3 Flow Solver Description

2.3.1 TBD Flow Solver Description

This section includes a brief overview of the new flow solver utilized in the validation. TBD is a finite-volume, Reynolds-Averaged Navier-Stokes (RANS) flow solver which can model thermally-perfect, chemically-reacting flows. The TBD flow solver is written in Fortran 90/95 and uses MPI libraries to conduct parallel computations. The solver requires abutting, multi-block, structured grids, and uses a cell-centered discretization. The conservative state vector (Q) lists transport equations for each i\textsuperscript{th} specie density, three momentum components, total energy, plus additional equations for transport of turbulent kinetic energy and dissipation rate of turbulent kinetic energy per Chien [Ref. 7].

\[
\frac{d}{dt} \int S \bar{Q} dV = \int S \bar{F}_{\text{inv}} \cdot \hat{n} dS + \int S \bar{F}_{\text{visc}} \cdot \hat{n} dS + \int V \bar{N} dV
\]

(28)

The inviscid, viscous, and source terms for the RANS equations are listed as Eqn. 29. The inviscid terms are treated using the Roe flux-vector splitting treatment for multi-specie flows following Grossman and Cinnella [Ref. 8]. High-order spatial fluxes are obtained using MUSCL extrapolations and use of min-mod limiter to enforce the TVD property, per Hirsch [Ref. 9]. Since the Roe scheme does not satisfy the second law of thermodynamics, a user-adjusted entropy fix is implemented. Viscous terms are treated with central-differencing in computational space, and both in-plane and cross derivative terms are retained. The source terms include potential contributions from chemical reactions and turbulence production and dissipation, and are evaluated using the cell center state vector values.
The transport equations are solved using an implicit, lower-upper factored, point-based, symmetric Gauss-Seidel method with Newton subiterations [Ref. 10]. The flux Jacobians are estimated using first order numerical derivatives (i.e., flux change due to small increment in each relevant conservative variable in the state vector). This increment is user-specified. This approach avoids the need to develop analytical Jacobians, and has rarely been identified to hinder convergence, but is comparatively expensive. Cross-derivative viscous Jacobians are neglected.

\[ F_{\text{sub}} \cdot \hat{n} = \begin{bmatrix} \rho V_x \\ \rho V_x u + \hat{n}_x p \\ \rho V_x v + \hat{n}_y p \\ \rho V_x w + \hat{n}_z p \\ \rho V_x h_0 \\ \rho V_x K \\ \rho V_x e \end{bmatrix} \quad \tilde{F}_{\text{visc}} \cdot \hat{n} = \begin{bmatrix} -\rho V_i \cdot \hat{n} \\ \begin{array}{c} \tau_{xx} \cdot \hat{n}_x + \tau_{xy} \cdot \hat{n}_y + \tau_{xz} \cdot \hat{n}_z - 2/3 \rho K \\ \tau_{yx} \cdot \hat{n}_x + \tau_{yy} \cdot \hat{n}_y + \tau_{yz} \cdot \hat{n}_z - 2/3 \rho K \\ \tau_{zx} \cdot \hat{n}_x + \tau_{zy} \cdot \hat{n}_y + \tau_{zz} \cdot \hat{n}_z - 2/3 \rho K \end{array} \\ - (\nabla q + \bar{\tau} \cdot \hat{n}) \\ (\mu + \mu_t/\sigma) \nabla K \cdot \hat{n} \\ (\mu + \mu_t/\sigma) \nabla e \cdot \hat{n} \end{bmatrix} \quad \tilde{N} = \begin{bmatrix} \dot{\rho} \\ 0 \\ 0 \\ 0 \\ P_K - D_K \\ P_e - D_e \end{bmatrix} \] (29)

2.3.2 TBD Verification and Validation

The flow solver is prepared for the current study by verification and validation of the following related capabilities: i) laminar heat flux prediction for supersonic flow and ii) high-speed turbulent chemically-reacting flow.

The first test case involves Mach 5 freestream flow over a hemisphere cylinder used to validate INCA [Ref. 11]. The laminar stagnation point heat flux is verified by comparison to Fay and Riddell’s correlation [Ref. 12]. This correlation for predicting stagnation point heating is based on detailed analytical solutions to the compressible boundary layer similarity equations. A temperature contour plot from TBD is provided in Fig. 3. For the current case the freestream conditions are $P_\infty = 6400$ Pa, $T_\infty = 64K$ ($T_0 = 384K$), and $T_w = 300K$. The Fay & Riddell analytical estimate is 61.7 kW/m$^2$, compared to TBD result of 64.0 kW/m$^2$ (see Fig. 4), for a 3.5% overprediction. The heat flux variation along the entire surface of the hemisphere is shown in Fig 7. A “Newtonian” ($\cos^2 \theta$) dependence is assumed for the Fay & Riddell derived result since the TBD result has a similar dependence.
The second case is validation against a well-known supersonic H2-air combustion experiment from Burrows and Kurkov [Ref. 13]. A schematic for this experiment is provided in Fig. 8. The inflow consists of hot (T=2286 K) vitiated air at Mach 2.44 in the main stream, with cold (T=400 K), pure hydrogen injected at sonic speed. The grid used for the Burrows-Kurkov test case consisted of a 121x 145 points and the domain is 35.6 cm long. The kinematic and thermal boundary layer profiles of the main stream are imposed at the inflow boundary using digitized \( u \) and \( T \) profiles from the experiment. An equilibrium turbulent profile is specified. Ebrahimi [Ref. 14] demonstrated that this inflow boundary definition was crucial towards obtaining the correct combustor exit conditions. The H2-air chemical kinetics are from Evan and Schexnayder [Ref. 15] and include 3rd body efficiencies.

The TBD results for the stagnation temperature field in Fig. 9 with K-\( \varepsilon \) indicate ignition occurs around 20 cm downstream from where fuel is injected, which is reasonably close to the experiment results of ~23 cm downstream. Combustor exit profiles (@ \( x = 35.6 \) cm) for four key combustion relation parameters are illustrated in Fig. 10 for the simulation and experiment. The peak H2O mole fraction levels compare
very well. The stagnation temperature and local Mach profiles also show excellent agreement. These results compare well with those provided for Wind-US validation [Ref. 16].

Figure 10: Combustor Exit Profiles for Burrow-Kurkov Case

2.3.3 Wind-US Flow Solver

The Wind-US flow solver is also coupled to FOGO for the subsequent water-cooled high-speed flow validation exercises. Wind-US is a general purpose flow solver provided by the NPARC Alliance, a partnership between NASA Glenn, Air Force AEDC, and Boeing [Ref. 17]. The Wind-US code has been extensively validated for a wide range of flow problems [Ref. 18]. For the computations presented herein, inviscid fluxes are computed using a true 2nd order Roe scheme which accounts for grid stretching, and the solution is advanced using local time stepping (i.e., based on global CFL constraint) and a spatially-split line-based factorization scheme. Several turbulence models are available.
3 Results

3.1 Case #1 (AEDC Water-Cooled Nozzle)

The water-cooled high-enthalpy, supersonic wind tunnel nozzle analyzed by Shope [Ref. 19] has been simulated using TBD-FOGO and Wind-FOGO. The nozzle configuration is shown in Fig. 11. The nozzle material is a copper-zirconium alloy and is 0.16 cm thick. The nozzle throat diameter is 2.29 cm (0.9 in.) and is designed for parallel exit flow at Mach number 1.8. The effect of the coflowing coolant is also considered. The run conditions implemented in the simulation are provided in Table 1, and provide direct comparison to experiment. Water inlet pressure was 68 atm for each run. A single specie, calorically and thermally perfect gas model was assumed for these computations to avoid costly reacting flow computations. A constant $\gamma = 1.24$ was assumed for these high enthalpy flows, as assumed in Ref. 19.

![Figure 11: AEDC Nozzle Geometry](image)

<table>
<thead>
<tr>
<th>Run</th>
<th>$P_{O,air}$ (atm)</th>
<th>$T_{O,air}$ (K)</th>
<th>$m_{water}$ (kg/s)</th>
<th>$T_{water, inlet}$ (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>126.5</td>
<td>5000</td>
<td>5.234</td>
<td>309</td>
</tr>
<tr>
<td>2</td>
<td>137.0</td>
<td>5240</td>
<td>5.234</td>
<td>307</td>
</tr>
<tr>
<td>3</td>
<td>104.4</td>
<td>4600</td>
<td>3.216</td>
<td>289</td>
</tr>
<tr>
<td>4</td>
<td>94.3</td>
<td>5100</td>
<td>3.204</td>
<td>289</td>
</tr>
</tbody>
</table>

Table 1: AEDC Nozzle Run Conditions

The TBD-FOGO simulations for Case #1 were conducted using a tightly-coupled approach in which each solver computed only one cycle before passing needed data to the other solver. Residuals for both the flow solver and solid conduction model typically reached three orders for these computations; however, the chosen convergence metric was to monitor the asymptotic rise in coolant temperatures until the value did not change over several hours of computation. The Chien K-ε model is specified for these runs, along with Sarkar compressibility and variable $C_p$ corrections. A single-zone grid of 73 X 81, with packing along the nozzle wall to 1e-5 inches, is utilized for the flow solver. The $y+$ value for 1st cell off...
surface typically ranges from 0.1 to 0.3. The solid body and coolant regions are discretized with 73 X 21 and 73 X 2 grids.

Predicted coolant bulk temperature rise for TBD-FOGO simulations are listed in Tables 2 and 3. The first set of results (Table 2) was obtained with the coolant removing heat by convection and nucleate (pool) boiling only. The second set of results (Table 3) was obtained with the coolant removing heat by convection and either nucleate or film boiling depending on temperature potential \((T_v - T_{sat})\). In each run with the nucleate (pool) boiling model, the predicted temperature rise is within 10% of the experimentally measured values. The error level increases to as much as 17% when the film boiling is active. These are excellent results considering the complex physics involved. In fact, we ran one case with 10% increase to the choice of turbulent Prandtl number (from default and typical value of 0.9) and found roughly a 10% increase in coolant temperature rise. Fig. 12 illustrates the water bulk temperature rise for Run #2. Note that the water coolant passes alongside nozzle in two distinct sections, so that a second water coolant stream enters just upstream of throat. Fig. 13 shows that the convective heat transfer is predicted to be the dominant mechanism for the water coolant. The pool-type nucleate boiling is significant in the vicinity of the nozzle throat since the nozzle/surface temperatures greatly exceed the saturation temperature.

<table>
<thead>
<tr>
<th>RUN #</th>
<th>Experimental Coolant Temp Rise</th>
<th>TBD-FOGO Temp Rise</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.9 K</td>
<td>13.3 K</td>
<td>-4.41%</td>
</tr>
<tr>
<td>2</td>
<td>15.0 K</td>
<td>14.8 K</td>
<td>-1.48%</td>
</tr>
<tr>
<td>3</td>
<td>18.9 K</td>
<td>18.1 K</td>
<td>-4.18%</td>
</tr>
<tr>
<td>4</td>
<td>20.0 K</td>
<td>18.0 K</td>
<td>-9.90%</td>
</tr>
</tbody>
</table>

Table 2: Comparison of Coolant Bulk Temperature Rise from Experiment with TBD-FOGO with only Nucleate Boiling
<table>
<thead>
<tr>
<th>RUN #</th>
<th>Experimental Coolant Temp Rise</th>
<th>TBD-FOGO Temp Rise</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.9 K</td>
<td>12.8 K</td>
<td>-8.2%</td>
</tr>
<tr>
<td>2</td>
<td>15.0 K</td>
<td>14.1 K</td>
<td>-6.0%</td>
</tr>
<tr>
<td>3</td>
<td>18.9 K</td>
<td>16.6 K</td>
<td>-12.1%</td>
</tr>
<tr>
<td>4</td>
<td>20.0 K</td>
<td>16.6 K</td>
<td>-17.1%</td>
</tr>
</tbody>
</table>

Table 3: Comparison of Coolant Bulk Temperature Rise from Experiment with TBD-FOGO with Nucleate and Film Boiling

![Graph showing nozzle geometry](image)

![Graph showing water coolant temperature rise](image)

Figure 12: Predicted Water Coolant Temperature Rise during Run #2
The WIND-FOGO simulations for Case #1 were conducted using a loosely-coupled approach in which each solver computed a sufficient number of cycles to converge several orders of magnitude before passing data to the other solver. This approach is extremely efficient, taking about 1 cpu hour of computation to complete each run.

Three sets of runs were completed with the pool boiling active (i.e., no film boiling) with different turbulence models: i) Chien K-£ with Sarkar and variable Cµ, ii) Chien K-£ without additional corrections, and iii) Menter’s SST. The first and third sets of results are quite similar and the K-£ results are listed in Table 3a. However, the K-£ without the corrections produces nearly 20% larger coolant temperature rises for each run. Subsequent runs with only one of the two corrections active indicate that most of the effect is due to the variable Cµ. Apparently, the adverse pressure gradients caused by the slope discontinuities along the nozzle are significant enough to make this correction important. Table 3b includes the results for another set of WIND-FOGO with K-£ and corrections, plus the film boiling is active. The error level shifts from 0-20% range to ± 10% range due to the expected reduction in boiling heat transfer effectiveness due to film boiling. Again, this is excellent agreement considering the complexity of the fluid and thermal physics.
### Table 3a: Comparison of Coolant Bulk Temperature Rise from Experiment with WIND-FOGO and Nucleate Boiling

<table>
<thead>
<tr>
<th>RUN #</th>
<th>Experimental Coolant Temp Rise (K)</th>
<th>TBD-FOGO Temp Rise (K)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.9 K</td>
<td>15.7 K</td>
<td>12.8%</td>
</tr>
<tr>
<td>2</td>
<td>15.0 K</td>
<td>17.8 K</td>
<td>18.9%</td>
</tr>
<tr>
<td>3</td>
<td>18.9 K</td>
<td>19.9 K</td>
<td>5.2%</td>
</tr>
<tr>
<td>4</td>
<td>20.0 K</td>
<td>20.4 K</td>
<td>1.9%</td>
</tr>
</tbody>
</table>

### Table 3b: Comparison of Coolant Bulk Temperature Rise from Experiment with WIND-FOGO and Nucleate/Film Boiling

<table>
<thead>
<tr>
<th>RUN #</th>
<th>Experimental Coolant Temp Rise (K)</th>
<th>TBD-FOGO Temp Rise (K)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.9 K</td>
<td>14.1 K</td>
<td>1.4%</td>
</tr>
<tr>
<td>2</td>
<td>15.0 K</td>
<td>16.3 K</td>
<td>8.7%</td>
</tr>
<tr>
<td>3</td>
<td>18.9 K</td>
<td>17.2 K</td>
<td>-9.0%</td>
</tr>
<tr>
<td>4</td>
<td>20.0 K</td>
<td>17.7 K</td>
<td>-11.5%</td>
</tr>
</tbody>
</table>

A primary effect of inclusion of a film boiling region is a reduction of the air-side nozzle surface temperatures. The heat rejection effectiveness of the coolant drops past the critical temperature potential. So, this potential must increase to reject the same heat load as compared to when pool boiling only is assumed. This effect is illustrated in Fig. 14 for the TBD-FOGO results. The surface temperature increases more than 100 K for each run when film boiling is permitted. The WIND-FOGO results show a similar sensitivity. The dashed line is the melting point of the zirconium-copper nozzle of 1350 K. Also, it should be noted that Runs #2 and #4 resulted in nozzle failure [Ref. 19]. Since Run #2 is predicted to have the highest surface temperature, and this temperature is roughly 83% of the melting point temperature, it would seem plausible that this nozzle might experience structural failure. However, the predictions do not identify Run #4 as any more likely to fail than Run #3.
Figure 14: Predicted Air-side Nozzle Surface Temperature Distributions for Case #1 (TBD-FOGO)
3.2 Text Case #2 (JPL Water-Cooled Nozzle)

The second validation case for TBD-FOGO and WIND-FOGO is the low-enthalpy water-cooled nozzle tested at JPL [Ref. 21]. The nozzle geometry is shown in Fig. 15 along with temperature distribution that was developed based on measurements from 22 thermocouples on and within the nozzle structure. The temperature distribution along the external nozzle wall was digitized and used as a boundary condition for the FOGO model. The nozzle throat diameter is 1.8 in, and is designed for Mach 2.2 exit flow. The nozzle material is unknown but needed material properties were deduced in Ref. 20. The run conditions for this case are provided in Table 4. A single specie, calorically and thermally perfect gas model was again assumed for these computations to avoid costly reacting flow computations. A constant $\gamma = 1.35$ was assumed in Ref. 20. This value was verified based on assumption of air at equilibrium at the stagnation conditions. A web tool from University of Colorado [Ref. 22] is used to obtain a specie distribution, which is used as input to TBD to calculate an inflow mixture $\gamma = 1.35$.

TBD-FOGO and WIND-FOGO are applied to Case #2 using the same procedures and convergence metric described earlier for Case #1. A single-zone grid of 121 X 81, with packing along the nozzle wall to 1e-5 inches, is utilized for the flow solver. The $y+$ value for 1st cell off surface typically ranges from 1 - 2. The solid body is discretized with a single zone 121 X 21 grid (see Fig. 16). Note that there is significant run-up to this nozzle that must be modeled to compare directly with experiment. There is no coolant model required since the nozzle-to-coolant wall temperature is a given boundary condition.

![Figure 15: Nozzle Temperature Distribution for JPL Experiment](image)

<table>
<thead>
<tr>
<th>$P_{O,\text{air}}$ (psia)</th>
<th>75.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{O,\text{air}}$ (K)</td>
<td>543</td>
</tr>
</tbody>
</table>

Table 4: Run Conditions for JPL Experiment (coolant flow details unknown)
TBD-FOGO is applied to Case #2 with two variants of the Chien K-ε model, including with and without the Sarkar compressibility and variable C_u corrections. Figure 17 provides comparison of predicted air-side wall temperature with the experimental measurements. As with Case #1, these corrections have a significant effect on the Case #2 results. Unlike Case #1, omission of the corrections produces a result which agrees more closely with experiment. Since the experimentalists determined that the experiment was reproducible within a ±2% in Ref. 21, we placed uncertainty bars of 2% at each data point. The numerical result for the uncorrected K-ε fall within all but the first two and one other data point.

![Figure 16: Case #2 Run-up and Nozzle Grid (everyother point removed; Blue: flow; Black: solid)](image)

Figure 17: Nozzle Inner-wall Surface Temperatures from TBD-FOGO vs. Experiment

WIND-FOGO is applied to Case #2 with the same two variants of the Chien K-ε model plus the Spalart Allmaras (SA) and SST turbulence models. Figure 14 provides comparison of predicted air-side wall temperature with the experimental measurements. The effect of the K-ε corrections is similar to those with TBD-FOGO. The best result is again obtained with the standard K-ε. The SA model also
produces a good match to the data. Clearly, it is difficult to know apriori which turbulence model will provide the best agreement with experiment, and so a significant uncertainty must be assumed if this approach is used in a predictive capacity.

Figure 18: Nozzle Inner-wall Surface Temperatures from WIND-FOGO vs. Experiment

### 3.3 Case #3 (NASA GRC Water-Cooled Thermal Panel within Rocket Plume)

The third validation case for TBD-FOGO and WIND-FOGO is to simulate a run condition from the high-enthalpy, water-cooled thermal panel tests conducted at NASA Glenn Research Center [Ref. 23]. An H$_2$-O$_2$ rocket engine (see Fig. 19) exhausts hot combustion products consisting of primarily water vapor along the rocket nozzle, nozzle ramp extension, and finally along thermal test panels. Two sidewalls are present causing the flow to expand in a roughly two-dimensional manner along the panels. There are three thin copper test panels, each mounted to a section of the calorimeter block (see Fig. 20). A separate water coolant flow passes thru each of the calorimeter sections via several cooling channels. The water temperature rise in each section is measured as well as panel surface temperatures using thermocouples. A fourth “sacrifice” section was attached to the final calorimeter section and is not shown.
The numerical representation of the calorimeter is most easily explained using Fig. 21. An eight zone grid is required. There are zones for each copper test panel plus one for the sacrifice panel that is not water cooled. There are four additional zones for the regions just upstream of each panel, since the panels do not extend the entire length of each calorimeter section. These intermediate regions are not water cooled. The three water-cooled panels are in contact with 8 rectangular coolant channels which remove heat by convection and boiling.

We chose Run No. 96 from these tests since this test was successfully repeated and involved the most extreme heat transfer levels. Based on the available combustion chamber stagnation pressure, mass flow rate, and oxidizer/fuel ratio, the NASA Glenn Chemical Equilibrium Code was utilized to estimate the resulting stagnation temperature and exhaust products assuming chemical equilibrium. Video of these
panel tests show afterburning occurs over the panels, and it was important to have the capability to investigate afterburning effects on the results. Consequently, a multi-specie H2-air model was adopted. The 7-specie H2-air chemical kinetics are the same as those used to validate TBD against the Burrows-Kurkov experiment mentioned in section III.D. The numerical nozzle inflow conditions are summarized in Table 5. Although there is no nitrogen in the nozzle exhaust, this specie will exist in the plume due to mixing with ambient air. A two-dimensional, 6-zone grid was constructed with 1.0E-4 inch and 1.0E-5 inch first cell thickness along nozzle and calorimeter walls, respectively. The y+ value for 1st cell off calorimeter wall surface typically ranges from 0.1 to 1.0. The grid includes a farfield which extends approximately 40 and 80 nozzle exit diameters radially and downstream. The Wilcox K-ω turbulence model was selected for TBD-FOGO simulation due to severe stability issues using K-ε, and Menter’s SST turbulence model was selected for Wind-FOGO simulation.

<table>
<thead>
<tr>
<th>$P_o$ (atm)</th>
<th>$T_o$ (K)</th>
<th>$Y_{O2}$</th>
<th>$Y_H$</th>
<th>$Y_{H2}$</th>
<th>$Y_{H2O}$</th>
<th>$Y_{OH}$</th>
<th>$Y_O$</th>
<th>$Y_N$</th>
<th>$\dot{m}_{water}$ (kg/s)</th>
<th>$T_{water}$ inlet (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.2</td>
<td>3557</td>
<td>0.061</td>
<td>0.003</td>
<td>0.0197</td>
<td>0.7836</td>
<td>0.1148</td>
<td>0.0178</td>
<td>0.0</td>
<td>-0.662</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 5: Nozzle Inflow Stagnation Conditions for Simulation of Case #3 (Run No. 95)

Fig. 22 illustrates the static temperature and Mach contours within the nozzle and vicinity of calorimeter. High temperatures develop along the upstream portion of the calorimeter, and within shear layers at downstream edge of nozzle and calorimeter due to intense viscous effects which decelerate the flow and cause temperature recovery. The temperatures near the calorimeter wall subside as the flow progresses downstream. Evidence of shock cell structure is also evident. Fig. 23 shows the heat flux variation along the calorimeter surface, and as expected from the contour plot, the peak surface heat flux occurs on the upstream portion. Minor kinks in the profile occur at regions between the cooled panels (roughly at 8 cm intervals) due to sudden change in thermal boundary condition.
Table 6 contains the TBD-FOGO predicted coolant temperature rise for each panel section versus experimental data. FOGO is calculating boiling heat transfer using both pool and film mechanisms active, as discussed earlier for Case #2. The error level is within 12% for all three panels, which should be considered excellent agreement considering the complexity of the fluid physics.
<table>
<thead>
<tr>
<th></th>
<th>Panel 1</th>
<th>Panel 2</th>
<th>Panel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD-FOGO $\Delta T$ Rise (K)</td>
<td>26.1</td>
<td>12.9</td>
<td>6.96</td>
</tr>
<tr>
<td>Experiment $\Delta T$ Rise (K)</td>
<td>23.4</td>
<td>12.4</td>
<td>6.66</td>
</tr>
<tr>
<td>% Error</td>
<td>11.4%</td>
<td>4.28%</td>
<td>4.50%</td>
</tr>
</tbody>
</table>

Table 6: TBD-FOGO Predicted vs. Experimental Water Coolant (multi-specie; no chemical reactions)

The predicted panel surface temperatures, however, fail to reach close to the measured values. Fig. 24 shows that the predicted surface temperature profile is 200 K-500 K below the thermocouple measurements. This discrepancy is related to the coolant model. If the coolant is less efficient in removing heat, the temperature potential would increase, and this comparison would improve. This potential is only a maximum of 100 K based on Fig. 24. Consequently, the adopted empirical boiling relations are apparently inappropriate for Case #3.

![Figure 24: Water and Air-side Calorimeter Surface Temperatures for Case #3 Simulation with TBD-FOGO](image)

Figure 24: Water and Air-side Calorimeter Surface Temperatures for Case #3 Simulation with TBD-FOGO

The effect of afterburning is found to be quite significant for the temperature rise prediction for panel 1 and to lesser extent, also panel 3 (see Table 7). The effect on the water and air-side temperature profiles is more subtle, and is not shown. Based on video of the experiment, significant afterburning is evident downstream of the calorimeter.
Table 7: TBD-FOGO Predicted vs. Experimental Water Coolant (multi-specie; chemically-reacting flow)

<table>
<thead>
<tr>
<th></th>
<th>Panel 1</th>
<th>Panel 2</th>
<th>Panel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD-FOGO ΔT Rise (K)</td>
<td>37.0</td>
<td>13.0</td>
<td>9.83</td>
</tr>
<tr>
<td>Experiment ΔT Rise (K)</td>
<td>23.4</td>
<td>12.4</td>
<td>6.66</td>
</tr>
<tr>
<td>% Error</td>
<td>57.9%</td>
<td>5.09%</td>
<td>47.6%</td>
</tr>
</tbody>
</table>

Predictions obtained using Wind-FOGO are very similar to those for TBD-FOGO based on comparisons of Figs. 25 and 26 with Figs. 22 and 23, and Table 8 with Table 6.

Figure 25: Static Temperature and Mach Contours with Wind-FOGO (multi-specie; no chemical reactions)
Figure 26: Heat Flux Profile along Calorimeter Surface with Wind-FOGO (multi-specie; no chemical reactions)

<table>
<thead>
<tr>
<th></th>
<th>Panel 1</th>
<th>Panel 2</th>
<th>Panel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD-FOGO ΔT Rise (K)</td>
<td>24.6</td>
<td>9.65</td>
<td>4.08</td>
</tr>
<tr>
<td>Experiment ΔT Rise (K)</td>
<td>23.4</td>
<td>12.4</td>
<td>6.66</td>
</tr>
<tr>
<td>% Error</td>
<td>4.82%</td>
<td>-22.0%</td>
<td>-38.7%</td>
</tr>
</tbody>
</table>

Table 8: WIND-FOGO Predicted vs. Experimental Water Coolant (multi-specie; no chemical reactions)

### 3.4 Conclusions

A new conjugate heat transfer solver, FOGO, has been introduced, verified, and successfully validated for simulation of water-cooled high-speed flows in conjunction with two RANS solvers. FOGO is written in Fortran 90/95, uses CGNS-format for I/O, uses multi-block structured grids, and is easily coupled to flow solvers. A new compressible RANS flow solver, TBD, has also been introduced and successfully validated for simulation of water-cooled high-speed flows in conjunction with FOGO.

Validation exercises of FOGO with TBD and Wind-US flow solvers demonstrate excellent accuracy with respect to predicting the local heat load for three independent cases including water-cooled nozzles and thermal panels. Specifically, prediction of the heat load via the coolant temperature rise (as in Cases #1 and #3) or local heat flux profile (as in Case #2) is shown to typically fall within 20% of experimental data. We conclude that state-of-the-art CFD can potentially be used as a general predictive tool for
thermal load in water-cooled high-speed flows provided a significant safety factor is placed on predictions.

The predictive accuracy of the conjugate heat transfer capability described herein is also limited by the turbulence model. It is unlikely that any single turbulence model could produce the best result in all cases, and so, a spread of heat flux and wall temperature predictions based on several well-established turbulence models is necessary in practice. This spread in the predictions for local heat load (based on Cases #1 and #2) can reach 30%.

It has been shown that superimposing heat transfer effects of coolant flow convection and boiling using simple empirical models does not provide sufficient accuracy to permit reliable prediction of surface wall temperatures and thermal failure. A more realistic treatment of the boiling and convective heat transfer mechanisms of the water coolant flow is highly desirable.
4 FOGO Module Description

4.1 Object-Oriented Approach

FOGO was developed using modern features introduced in FORTRAN 90/95. The module relies heavily on basic object-oriented programming (OOP) techniques using FORTRAN's user defined data types (analogous to C structures). FORTRAN 90/95 is not fully an OOP language (such as Python [ref. 24]) but it does offer basic features such as encapsulation, data abstraction, and class definitions.

The building block of the FOGO module are the definitions of the solid and fluid volume elements. These objects (the volume elements) are defined using user defined data types. In older FORTRAN versions (F77 and earlier), a variable could only hold a single value, such as an integer or a floating point number. A variable defined as an object (or user defined type), is theoretically capable of storing infinite variables to describe a given object (example: size, shape, color). For example, the definition of the object that represents a solid cell in FOGO (see class_My_Volume.f90 file) resembles:

```fortran
  type My_Volume !! This is the object representing a solid volume element cell
    !! Object variables:
    real :: T_Center ! Temperature at the center of the volume element
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    !! Avg. Temperature @ the cell faces
    real :: Face_1_T, Face_2_T, Face_3_T
    real :: Face_4_T, Face_5_T, Face_6_T
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    !! **Cell Face Areas:
    real :: Face_1_Area !! surface area @ face 1-2-3-4, Xi face
    
    end type My_Volume
```

FOGO uses 3-dimensional arrays filled with volume element objects to represent a computational block of the body that we are modeling. For example, let's say we want to define an array (block) filled with 5x5x5 solid volume elements:
FOGO is a multi-block structured code, which means that many blocks (arrays) can be used to describe the geometry of the solid. This feature is extremely useful for meshing objects with complicated geometries using structured grids. Multi-block is implemented by using a container object which groups all arrays/blocks filled with volume elements into one large global variable (see My_Volume_ArrayContainer.f90 and global_array.f90 file). For example, the following syntax would be used to change the cell centered temperature of cell i=1, j=1, k=1 located in block number 5 to 100 Kelvin (where CVContainer is the array container object):

\[
\text{CV\_Container}(5)\%\text{Volume\_Elements}(1,1,1)\%\text{T\_Center} = 100.0
\]

While the current object-oriented programming approach does have certain advantages such as code maintainability (adding and removing control volume object definitions are "felt" everywhere in the code; less prone to array indexing errors, especially for cell face variables) and code reusability (inheritance and encapsulation), there is a performance price to pay: all abutting cell face values are stored twice in memory. This happens because in FORTRAN all variables of an array must have the same type. This problem is not encountered in traditional solvers where the cell face variables of the control volumes (which are traditionally represented by real numbers, and not objects/structures) are stored in distinct arrays. Although an increase in solver performance is always desired, FOGO proved to have acceptable performance for the simulations performed in this study (less than $10^4$ cells) using a single processor.

### 4.2 Module Dependency and Hierarchy

The FOGO source code is divided into a series of FORTRAN modules. A FORTRAN 90/95 module can be thought as a library in which the programmer defines variables and procedures with global scope...
that can be called by other programming units.

There are a total of 18 modules in FOGO (see Appendix A). A Python script was developed to scan the existing FOGO source code to search for the FORTRAN-90 “use” statement and stored the results in a text file. Graphviz, an open-source program used to study network-based relationships [Ref. 25], was then used to read in text file to study FOGO’s module hierarchy. The resulting output is presented in Figure 26.

The arrows in Figure 26 represent how the modules are being imported amongst each other. For example, an arrow leaving module “A” and arriving at module “B” would mean that module “B” depends on variables and subroutines defined in module “A”. Using this convention and looking at Figure 26 we can see that the hierarchy increases from the top to the bottom of the figure. We can also note that the modules class_My_Volume and Coolant.CV are base modules (where solid and coolant volume objects are defined), while the FOGO module (responsible for calling the heat transfer solver) is located on the top of the hierarchal chain. Any changes performed to the base modules (ie, control volume object definitions) are felt throughout the code; this feature assists with code extendibility and maintainability.

The information presented Figure 26 and Appendix A (module description) should be used to assist the user with future modifications to FOGO’s source code.
4.3 Compiling FOGO

There is a utility Python script named run_FOGO in the source directory which is responsible compiling and running FOGO under Linux. The user should perform changes to the run_FOGO source code (under the "#<User interface>") section if he or she wants to:

1. Change the FORTRAN compiler (default is the "ifort" Intel compiler).
2. Attach FOGO to existing flow solvers; the user should specify the flow-solver source code files to be compiled with FOGO.
3. Change FOGO source files that will be compiled (used to extend the FOGO library).
4. Change the name of the resulting executable file which is generated through the source compilation (default executable file is named FOGO).

This utility can be thought of as a "fancy" makefile script. Much care has been given to comment the source code of the run_FOGO script so that it could be easily modified and extended by the user. The run_FOGO utility script should be called from the prompt using the following commands from the Linux prompt:

1. To compile (no array bound checking):
   username@computer:~$ ./run_FOGO --compile
2. To compile using array bound checking (checks for array index violation):
   username@computer:~$ ./run_FOGO --compile_cb
3. To compile and run interactively:
   username@computer:~$ ./run_FOGO --compile_and_run
4. To compile with array bound checking and run interactively:
   username@computer:~$ ./run_FOGO --compile_cb_and_run
5. To run FOGO interactively as a standalone heat transfer solver with boundary and initial conditions specified in FOGO.inp file:
   username@computer:~$ ./run_FOGO --run

It should be noted that most CFD research codes are comprised of many different FORTRAN source files and normally have their own makefile utility script to compile the source code (such as the HYP flow solver). To compile FOGO with these types of CFD codes a "FOGO.a" library file must be generated and the user should make appropriate changes to the flow solver's makefile to account for this.
new library file. To generate the “FOGO.a” library file, the user should set CFD_SOURCE_FILES = "main.f90" in the run_FOGO script and compile FOGO using the “run_FOGO --compile” command from the prompt. The “main.f90” file is a dummy source file which is included in FOGO’s source directory (see Appendix B).

4.4 Required FOGO Inputs

In order to run FOGO successfully the user should have a structured grid of the solid (stored in CGNS file format), and specify boundary and initial conditions plus miscellaneous solver variables within the FOGO.inp input text file. This input file should be placed in the same directory where the FOGO executable is located. During runtime initialization FOGO loads all the variables stored in FOGO.inp; therefore a change in the input file does not require the recompilation of the source code.

The FOGO.inp file uses a FORTRAN namelist structure to define all the input variables. The namelist is divided into two sections (also known as FORTRAN group-names): “INPUT” and “IC”. Alternatively, a third section is included in FOGO.inp to define miscellaneous variables for the coolant module or WIND-FOGO interface. The “INPUT” section of the file contains miscellaneous variables required by the FOGO solver, such as the name of the grid file of the solid the body and its material properties.

The “IC” section contains all boundary and initial conditions for the simulation; it must be noted that only constant boundary conditions (B.C.) and initial conditions can be specified in the FOGO.inp file (ie, you can only define constant wall temperature and heat flux B.C.s in FOGO.inp). The user can specify a more complicated boundary and initial conditions (ie, distributions) by hard-coding them into the flow solver's source code using the FOGO module. This technique is documented in section 5.5.

Sample FOGO.inp files for the three validation cases are presented in Appendices C, D, and E.

4.5 Attaching FOGO to Existing Flow Solvers

In the source directory the main.f90 file contains a dummy “flow solver” program which demonstrates how the user should attach and call the FOGO from their code to perform the conjugate heat transfer analysis. This file is included in Appendix B. General instructions on how to communicate with FOGO and how to specify custom boundary condition distributions (which cannot be specified from the
FOGO.inp input file) are described in the following paragraphs.

The first step in attaching FOGO to the existing flow solver (if any, since FOGO can also run as a standalone application) is for the programmer to determine in which section(s) of the CFD code that would be best to exchange boundary condition information along the wall that separates the solid and fluid zones. The programmer must identify where the heat fluxes for the current time step can be readily estimated, and also determine where to receive the new wall temperature boundary condition estimated by FOGO for the upcoming time step.

In general the flow solver should perform the following steps to exchange information with FOGO successfully (refer to Figure 1):

1. Import the FOGO module
2. Initialize all FOGO variables and arrays
3. Estimate the heat flux at the wall between the solid and fluid zones
4. Send the heat flux distribution to FOGO (BC for the conduction model is updated)
5. Run FOGO for N iterations so that a new wall temperature distribution for the wall can be estimated
6. Retrieve a new wall temperature distribution from FOGO (BC for the flow solver is updated)

The following sections will provide the programmer with more details how to couple the flow solver and FOGO.
4.5.1 Importing the FOGO Module

The programmer should import both “FOGO” and “Conjugate_Interface” modules using the “use” FORTRAN-90 directive. The “use” statement should be placed in the first couple of lines of the flow solver's code where FOGO is to be called from (ie, beginning of subroutine or program body); this statement should appear before any variable declaration.

```fortran
use FOGO, ONLY: FOGO_RUN, FOGO_initial_setup, &
FOGO_Deallocate, CV_Container

use FOGO, ONLY: Grid_array, My_Zone_Array, Plot_Cell_Center_as_Node, &
Save_Ghost_Cells_to_Output, N_zones

use Conjugate_Interface
```

The first two “use” statements imports all required variables and subroutines pertinent to FOGO's heat transfer calculations. The “Conjugate_Interface” module contains interface subroutines which are responsible for exchanging wall temperature and heat flux information between both solvers.

4.5.2 Initializing FOGO

The programmer should first declare some crucial variables that are used to communicate with FOGO:

```fortran
real, dimension(123,1,1) :: TW_array, Heat_Flux_Array
integer, parameter :: CGNS_Zone = 1
character(32), parameter :: CGNS_Face = "JMIN"
```

Here the wall temperature and heat flux arrays are declared; these arrays will hold face centered values to send and receive information between solvers. Both arrays in this case have 123 x 1 x 1 elements (i x j x k). The current FOGO interface subroutines requires these arrays to be 3-dimensional and have same index ordering as the solid mesh stored in the CGNS geometry file.

The “CGNS_Zone” variable will be used to tell FOGO at which zone/block number we will apply the heat fluxes and receive the temperatures. The zone/block numbering of the solid is determined by Gridgen (or similar preprocessor) when the user is creating the mesh for the solid body.
The "CGNS_Face" variable will instruct FOGO which face of the solid's mesh that flow solver will apply the heat flux and receive wall temperature. The available options for this variable are "IMIN", "IMAX", "JMIN", "JMAX", "KMIN", and "KMAX". All of these variables are case sensitive and should capitalized. In our example the flow solver will be applying heat fluxes and reading wall temperatures for the solid block number 1 (CGNS_Zone = 1) at the "JMIN" (where j = 1) face of the solid (CGNS_Face = "JMIN").

The user can now initialize FOGO by calling the "FOGO_initial_setup" subroutine which will make the heat transfer module to load the variables and boundary conditions stored in the FOGO.inp input file. This subroutine will also instruct FOGO to calculate all the required geometric parameter related to the mesh of the solid body (like cell face areas, volumes, metric derivatives) and also estimate a stable time step for the explicit conduction scheme.

```
call FOGO_initial_setup()
```

### 4.5.3 Estimating the Heat Flux

Once the required variables have been initialized the programmer should estimate the heat flux at the wall for the first CFD iteration and "populate" the heat flux array with the estimated values. The CFD solver should use the wall temperature and a stencil of fluid cells perpendicular to the surface at a given i-location (assuming that the flow is in the i-direction) to estimate the heat flux. In this example we will just pass a dummy heat flux value to our heatflux array:

### 4.5.4 Send Heat Flux to FOGO

Once the heat flux has been calculated we can pass it to FOGO and apply the boundary condition to the desired CGNS zone/block and face:

```
call pass_heat_flux_to_FOGO(CGNS_Zone,CGNS_Face,Heat_Flux_Array)
```

Note that this will overwrite any existing heat flux boundary condition previously defined in the FOGO.inp input file. This subroutine could have also been used to specify a unique boundary condition distribution when running FOGO as a standalone application.
4.5.5  Running FOGO for N iterations

FOGO can be executed for N-iterations by simply using the FOGO_RUN subroutine. In this example we will subiterate FOGO for 10 iterations:

```fortran
  do N = 1,10
    call FOGO_RUN()
  end do
```

4.5.6  Retrieve New Wall Temperature Distribution

Once FOGO has subiterated the user should retrieve the new wall temperature distribution predicted by FOGO:

```fortran
  call get_wall_temp_from_FOGO(CGNS_Zone,CGNS_Face,TW_array)
```

The flow solver's wall temperature boundary conditions should be updated with the "newer" wall temperatures, which in our case is stored in TW_array. The steps described in sections 5.5.4 through 5.5.6 are iterated until a convergence criteria has been established.

4.5.7  Deallocating FOGO Dynamic Arrays

When the simulation has reached the end the user should call the FOGO_Deallocate subroutine to erase all FOGO dynamic arrays from memory. If this is not done, these variables will remain in memory until the computer is rebooted.

```fortran
  call FOGO_Deallocate()
```

4.6  FOGO Restrictions/Limitations

The current multi-block implementation in FOGO assumes that all the abutting faces of blocks to have point-matched connectivity and also that the blocks must have same i-j-k cell index ordering. Therefore the user must pay special attention while meshing a multi-block solid so that abutting blocks can "communicate" properly the heat loads during the simulation. Figure 27 and 28 illustrates how the user should mesh the solid so that both of these requirements are met.
The "*" in the above figures represents the first cell of each block (i=1, j=1, k=1). The meshing strategy presented in Figure 27 would yield in an ill posed FOGO grid since the abutting cell faces of the blocks don't match-up and the blocks use a different cell ordering. The solution for this problem is presented in Figure 28. Here "Block 2" was broken into two separate blocks to impose a one-to-one point-matched connectivity between the cell faces of the blocks. The same cell index ordering was also applied on all three blocks. This would force the I,J,K-MIN face of a block to exchange information with the I,J,K-MAX face of the abutting block, which is how FOGO currently handles multi-block grids.

FOGO currently uses a cell-centered thermal conductivity values for conduction heat transfer calculations. A more correct approach would be to evaluate the thermal conductivity of the material at cell faces. This would "regulate" the correct amount of energy flowing into the cell in case the conduction coefficient varies from cell to cell (ie, abutting cells with different materials or conductivity is a function of temperature). Although the user can still change the value of the thermal conductivity for each cell during runtime, FOGO still lacks a subroutine that would extrapolate this material property to the cell faces. Therefore, with the current scheme, specifying different thermal conductivity values would induce errors to the computation. This limitation to FOGO did not introduce any errors to the three
validation cases presented in this study since a constant conductivity was assumed.

Currently there is no mesh file support for the coolant cell geometry definition. The coolant cells must be defined programmatically in FOGO (such as: cell face areas, bulk temperature, flow velocity, etc.). The coolant cells are represented using an object-oriented approach similar to the solid cell object. The reader should refer to the "Coolant_CVf90" file for coolant cell object definition and convective heat transfer subroutines. The convective subroutines assumes that the coolant is flowing along the i-direction, while the coolant block is assumed to be one cell thick, by one cell deep, with N elements along the coolant flow direction (N × 1 × 1 elements).

4.7 Code Conclusion and Future Improvements

While the current object-oriented approach in FOGO has yielded a fast and versatile way to extend and manage the code, there has been a performance price to pay. The current programming approach uses almost twice the amount of memory as traditional solvers since cell face variables are duplicated amongst abutting cells. Although the performance has been acceptable for all the validation cases presented in this study, there was a dummy test case performed with FOGO on a cube with 1 million cells which resulted in unacceptable performance (using an AMD-Turion 1.8 Ghz HP laptop with 1 Gb of RAM).

FOGO's performance could be increased by moving away from the object oriented programming approach to the traditional functional programming, but this would mean that most of the code would have to be rewritten. Rather than rewriting most the code, an MPI (Message Passing Interface) capability could be added to FOGO which would permit distributing the computational load among separate processors in the cluster. This alternative would be viable since FOGO was primarily designed to be coupled CFD solvers that run on clusters using MPI. Furthermore, FOGO has already shown acceptable performance for grids of moderate size (N ≤ 10^5, per processor).

There are still multi-block connectivity issues that were related in the previous section (5.6) that forces the user to pay attention while creating the mesh. This could be fixed by creating a more elaborate multi-block connectivity subroutine that would check for generalized multi-block arrangements/interfaces. Other improvements such as the evaluation of thermal conductivity at cell faces (see section 5.6) and the inclusion for a implicit solver are desired for future releases.
## Appendix A: FOGO Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class My Volume (see class My Volume.f90)</td>
<td>Definition of the solid control volume object (named My Volume) and where all major calculations related to solid heat transfer explicit scheme is implemented.</td>
</tr>
<tr>
<td>Coolant CV (see Coolant CV.f90)</td>
<td>Definition of coolant control volume object and where all major calculations related to coolant heat transfer explicit scheme is implemented.</td>
</tr>
<tr>
<td>MultiZoneCGNS (see sorting_connections.f90)</td>
<td>Handles multi-block connectivity for the grid of the solid.</td>
</tr>
<tr>
<td>My Volume Container (see My Volume Array Container.f90)</td>
<td>Definition of a container type array/structure which stores 1 block filled with solid control volume objects.</td>
</tr>
<tr>
<td>My Cool Array Container Struct (see My Coolant Array Container.f90)</td>
<td>Definition of a container type array/structure which stores 1 block filled with coolant control volume objects.</td>
</tr>
<tr>
<td>GridModule (see Grid.f90)</td>
<td>Geometric calculations related for the grid of the solid body (volumes, face areas, metric derivatives, etc.).</td>
</tr>
<tr>
<td>Ghost_Cell_Subroutines (see Ghost Cells.f90)</td>
<td>Subroutines used to exchange information between adjacent blocks of the solid body for multi-block computations.</td>
</tr>
<tr>
<td>CGNS (see CGNS.f90)</td>
<td>Module reads in the CGNS file of the solid grid. The module also saves the heat transfer results to the “output.cgns” file for post-processing with Tecplot.</td>
</tr>
<tr>
<td>VTK (see VTK.f90)</td>
<td>Writes out results to *.vtk files for post-processing with ParaView [Ref.26].</td>
</tr>
<tr>
<td>FOGO_restart (see FOGO_restart.f90)</td>
<td>Creates restart files for the simulation so that the simulation can be stopped and resumed at a later time.</td>
</tr>
<tr>
<td>global_arrays (see global_array.f90)</td>
<td>Defines some arrays with global scope which are used throughout FOGO.</td>
</tr>
<tr>
<td>Load Input File (see Load Input File.f90)</td>
<td>Used to load the information stored in the FOGO.inp input file.</td>
</tr>
<tr>
<td>Coolant Interface (see coolant interface.f90)</td>
<td>Subroutines used to exchange wall temperatures and heat flux information between the coolant zone and the abutting solid zone.</td>
</tr>
<tr>
<td>FOGO_Convergence (see FOGO_Convergence.f90)</td>
<td>Checks to see if the heat transfer within the solid body has approached steady-state.</td>
</tr>
<tr>
<td>Conjugate Interface (see conjugate Interface.f90)</td>
<td>Module contains subroutines that are used to exchange heat flux and wall temperatures with an existing flow solver to perform the conjugate heat transfer analysis.</td>
</tr>
<tr>
<td>Module</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FOGO_init</td>
<td>Initializes all FOGO variables; also sets the boundary conditions and initial conditions defined in FOGO.inp file.</td>
</tr>
<tr>
<td>(see FOGO_initial_setup.f90)</td>
<td></td>
</tr>
<tr>
<td>FOGO_clean_mem</td>
<td>Used to deallocate FOGO's dynamic arrays.</td>
</tr>
<tr>
<td>(see FOGO_clean_mem.90)</td>
<td></td>
</tr>
<tr>
<td>FOGO</td>
<td>This module contains the FOGO_RUN subroutine instructs FOGO to perform all the required calculations for 1 iteration.</td>
</tr>
<tr>
<td>(see FOGO.f90)</td>
<td></td>
</tr>
</tbody>
</table>
program main
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! Import required modules:

!! All FOGO internal variables and subroutines:
use FOGO,ONLY : FOGO_RUN, FOGO_initial_setup, FOGO_Deallocate, CV_Container

!! Module with interface subroutines
use Conjugate_Interface

implicit none
integer :: i
integer :: i_max,j_max,k_max
integer :: CGNS_Zone
character(32) :: CGNS_Face

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! SETTING UP VARS FROM CFD USER PERSPECTIVE USING F90:

i_max = 119 !! Number of elements along the i-direction
j_max = 1   !! Number of elements along the j-direction
k_max = 1   !! Number of elements along the k-direction
CGNS_Zone = 1 !! We will apply the heat flux to the CGNS solid model for block number 1
CGNS_Face = "JMIN" !! heat flux will be applied @ JMIN face of the solid (J=1)
!! OPTION FOR CGNS_FACE ARE: IMAX,JMAX,KMAX,IMIN,JMIN,KMIN -> !!CASE SENSITIVE!
allocate(TW_array_J_max(i_max,j_max,k_max), Heat_Flux_Array_J_max(i_max,j_max,k_max), &
stat=alloc_status)
if (alloc_status /= 0) then
  print *, 'Allocation failed.'
stopt
end if

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! Setting up FOGO:
!! Loads all modules, initializes all required variables, call this before the first iteration!
call FOGO_initial_setup()
!! Get the wall temperature distribution before first CFD iteration so we can estimate the first heat flux
!! distribution from the CFD code; this will load the temperature distribution of the solid from
!! the restart.cgns file or a constant temperature distribution defined in the FOGO.inp. To instruct
!! FOGO to use the restart.cgns file, set the Load_Restart_File variable in FOGO.inp to .TRUE.

call get_wall_temp_from_FOGO(CGNS_Zone,CGNS_Face,TW_array_J_max)

!!! Now performing a dummy conjugate heat transfer analysis
do  i = 1,100

   ! Now giving a dummy value for the flux:
   Heat_Flux_Array_J_max(:,:,?) = 1000000.0

   !! First we would pass the heat flux @ Jmax face to FOGO
   !! (which corresponds to JMIN to the FOGO)
   call pass_heat_flux_to_FOGO(CGNS_Zone,CGNS_Face,Heat_Flux_Array_J_max)

   !! Next we run FOGO
   call FOGO_RUN() !! Run Fogo for 1 iteration

   !! Then we grab the wall temperature @ the Jmax face
   call get_wall_temp_from_FOGO(CGNS_Zone,CGNS_Face,TW_array_J_max)
end do

!!! CLEAN MEMORY SINCE WE HAVE FINISHED:
call FOGO_Deallocate() !! Clear allocated arrays!
deallocate(TW_array_J_max,Heat_Flux_Array_J_max, stat=alloc_status)
if (alloc_status /= 0) then
   print *, 'Deallocation failed.'
   stop
end if
Appendix C: FOGO.inp File for Case 1, Run 1

&INPUT
InputFile = Nozzle_SI_units.cgns,
Load_Restart_File = .TRUE.,
Plot_Cell_Center_as_Node = .FALSE.,
Save_Ghost_Cells_to_Output = .FALSE.,
estimate_delta_tau = .FALSE.,
use_delta_tau_ratio = 0.9,
delta_tau = 1.0,
n_Save_restart_file = 1000,
n_Save_output = 500,
k_coeff = 367.0,
rho = 8890.0,
cp = 385.0,
iMin_Insulated = .TRUE.,
iMax_Insulated = .TRUE.,
jMin_Insulated = .FALSE.,
jMax_Insulated = .FALSE.,
kMin_Insulated = .TRUE.,
kMax_Insulated = .TRUE.,
iMin_Temp_Given = .FALSE.,
iMax_Temp_Given = .FALSE.,
jMin_Temp_Given = .FALSE.,
jMax_Temp_Given = .FALSE.,
kMin_Temp_Given = .FALSE.,
kMax_Temp_Given = .FALSE.,
iMin_Flux_Given = .FALSE.,
iMax_Flux_Given = .FALSE.,
jMin_Flux_Given = .FALSE.,
jMax_Flux_Given = .FALSE.,
kMin_Flux_Given = .FALSE.,
kMax_Flux_Given = .FALSE.,
/
&IC
Coolant_M_dot = 5.234,
Coolant_Inlet_Temp = 309.0,
Internal_Temp = 309.0,
iMin_Temp = 10000.0,
iMax_Temp = 10000.0,
jMin_Temp = 10000.0,
jMax_Temp = 10000.0,
kMin_Temp = 10000.0,
kMax_Temp = 10000.0,
iMin_Flux = 1000000.0,

!! name of the grid file
!! resume simulation from a restart file?
!! output cell centered temperature at vertices in the output.cgns
!! save ghost cell data to output.cgns
!! let FOGO estimate the appropriate time step
!! let this at 0.9 to run at 90% of the stable time step
!! specify a time-step manually, to use this, set estimate_delta_tau to FALSE
!! save a restart file every N iterations
!! save output every N iterations
!! thermal conductivity of the solid
!! density of the solid
!! specific heat
!! is IMIN face of the solid insulated?
!! is IMAX face of the solid insulated?
!! is JMIN face of the solid insulated?
!! is JMAX face of the solid insulated?
!! is KMIN face of the solid insulated?
!! is KMAX face of the solid insulated?
!! wall temperature boundary condition specified at IMIN face of the solid?
!! wall temperature boundary condition specified at IMAX face of the solid?
!! wall temperature boundary condition specified at JMIN face of the solid?
!! wall temperature boundary condition specified at JMAX face of the solid?
!! wall temperature boundary condition specified at KMIN face of the solid?
!! wall temperature boundary condition specified at KMAX face of the solid?
!! wall heat flux boundary condition specified at IMIN face of the solid?
!! wall heat flux boundary condition specified at IMAX face of the solid?
!! wall heat flux boundary condition specified at JMIN face of the solid?
!! wall heat flux boundary condition specified at JMAX face of the solid?
!! wall heat flux boundary condition specified at KMIN face of the solid?
!! wall heat flux boundary condition specified at KMAX face of the solid?
!! coolant mass flow rate
!! coolant inlet temperature
!! initial temperature of the solid
!! specified B.C. temperature at IMIN face of the solid
!! specified B.C. temperature at IMAX face of the solid
!! specified B.C. temperature at JMIN face of the solid
!! specified B.C. temperature at JMAX face of the solid
!! specified B.C. temperature at KMIN face of the solid
!! specified B.C. temperature at KMAX face of the solid
!! specified B.C. heat flux at IMIN face of the solid
iMax_Flux = 1000000.0, !! specified B.C. temperature at IMAX face of the solid
jMin_Flux = 1000000.0, !! specified B.C. heat flux at JMIN face of the solid
jMax_Flux = 1000000.0, !! specified B.C. heat flux at JMAX face of the solid
kMin_Flux = 1000000.0, !! specified B.C. heat flux at KMIN face of the solid
kMax_Flux = 1000000.0, !! specified B.C. heat flux at KMAX face of the solid
/
&Nucleate_Boiling
Use_Nucleate_Boiling = .TRUE. !! To turn on Nucleate Boiling, set this to .TRUE.
T_sat = 557.91 !! Saturation Temperature of the coolant, kelvin
h_fg = 1512000.0 !! (enthalpy gas - enthalpy fluid) @ T_sat
rho_f = 741.69 !! Density of the fluid @ T_sat
rho_g = 35.885 !! Density of the gas @ T_sat
surface_tension = 0.017882 !! @ T_sat
Csf_coeff = 0.027 !! Coefficient that depends on fluid and wall material
s_coeff = 1.0 !! Coefficient that depends on fluid and wall material
/

50
Appendix D: FOGO.inp File for Case 2

&INPUT
GridFile = "solid_coarse.cgns",
GridFile_Units = "METER",
Load_Restart_File = .TRUE.,
subiterate = .TRUE.,
subiterate_tolerance = 1E-3,
n_Save_restart_file = 1000,
n_Save_output = 1000,
Plot_Cell_Center_as_Node = .FALSE.,
Save_Ghost_Cells_to_Output = .FALSE.,
estimate_delta_tau = .TRUE.,
use_delta_tau_ratio = 0.7,
delta_tau = 1.0,
k_coeff = 27.0,
rho = 7800.0,
cp = 460.0,
iMin_Insulated = .TRUE.,
iMax_Insulated = .TRUE.,
jMin_Insulated = .FALSE.,
jMax_Insulated = .FALSE.,
kMin_Insulated = .TRUE.,
kMax_Insulated = .TRUE.,
iMin_Temp_Given = .FALSE.,
iMax_Temp_Given = .FALSE.,
jMin_Temp_Given = .FALSE.,
jMax_Temp_Given = .FALSE.,
kMin_Temp_Given = .FALSE.,
kMax_Temp_Given = .FALSE.,
iMin_Flux_Given = .FALSE.,
iMax_Flux_Given = .FALSE.,
jMin_Flux_Given = .TRUE.,
jMax_Flux_Given = .FALSE.,
kMin_Flux_Given = .FALSE.,
kMax_Flux_Given = .FALSE.,
/
&IC
Internal_Temp = 400.0,
iMin_Temp = 10000.0,
iMax_Temp = 10000.0,
jMin_Temp = 366.48,
jMax_Temp = 366.48,
kMin_Temp = 10000.0,
kMax_Temp = 10000.0,
iMin_Flux = 1000000.0,
iMax_Flux = 1000000.0,
jMin_Flux = 1000000.0,
jMax_Flux = 1000000.0,
kMin_Flux = 1000000.0,
kMax_Flux = 1000000.0,
/
Appendix E: FOGO.inp File for Case 3

&INPUT
GridFile = "panels.cgns",
GridFile_Units = "INCH",
Load_Restart_File = .FALSE.,
subiterate = .TRUE.,
subiterate_tolerance = 1E-16,
n_Save_restart_file = 1000,
n_Save_output = 1000,
Plot_Cell_Center_as_Node = .FALSE.,
Save_Ghost_Cells_to_Output = .FALSE.,
estimate_delta_tau = .TRUE.,
use_delta_tau_ratio = 0.9,
delta_tau = 1.0,
k_coeff = 353.0,
rho = 8954.0,
cp = 383.1,
iMin_Insulated = .TRUE.,
iMax_Insulated = .TRUE.,
jMin_Insulated = .TRUE.,
jMax_Insulated = .FALSE.,
kMin_Insulated = .TRUE.,
kMax_Insulated = .TRUE.,
iMin_Temp_Given = .FALSE.,
iMax_Temp_Given = .FALSE.,
jMin_Temp_Given = .FALSE.,
jMax_Temp_Given = .FALSE.,
kMin_Temp_Given = .FALSE.,
kMax_Temp_Given = .FALSE.,
iMin_Flux_Given = .FALSE.,
iMax_Flux_Given = .FALSE.,
jMin_Flux_Given = .FALSE.,
jMax_Flux_Given = .TRUE.,
kMin_Flux_Given = .FALSE.,
kMax_Flux_Given = .FALSE.,
/
&IC
Internal_Temp = 700.0,
iMin_Temp = 10000.0,
iMax_Temp = 10000.0,
jMin_Temp = 366.48,
jMax_Temp = 1000.0,
kMin_Temp = 10000.0,
kMax_Temp = 10000.0,
iMin_Flux = 10000000.0,
&Coolant_Data
Use_Coolant = .TRUE., !! Does the current simulation involve any active cooling? (We assume coolant flows along i-dir)
N_zones_with_coolant = 3, !! If so, how many zones with active cooling?
Zones_with_coolant = 2, 4, 6, !! Which zones have coolant flowing through their boundaries? -> Based on CGNS zone numbering
N_Coolant_Channels = 8, 8, 8, !! How many coolant channels do we have in each zone specified in Zones_with_coolant array?
Solid_Cooled_Face = "JMIN", !! Which face of the solid zone is being cooled? -> Options: JMIN, JMAX ONLY!
Depth_Coolant_Channel = 0.002032, !! Cooling passage depth
Width_Coolant_Channel = 0.003175, !! Cooling passage width
Coolant_M_dot = 0.0844811375, 0.082780175, 0.0816462, !! Mass flow rate of the coolant for each coolant channel panel
Coolant_Inlet_Temp = 300.0,
Use_Nucleate_Boiling = .TRUE., !! To turn on Nucleate Boiling, set this to .TRUE.
Use_Film_Boiling = .TRUE., !! To turn on Film Boiling, set this to .TRUE.
T_sat = 446.57, !! Saturation Temperature of the coolant, kelvin
h_fg = 2128660, !! (enthalphy gas - enthalphy fluid) @ T_sat (J/kg)
rho_f = 921.48, !! Density of the fluid @ T_sat
rho_g = 2.2518, !! Density of the vapor @ T_sat
surface_tension = 0.043653, !! @ T_sat
Csf_coeff = 0.013, !! Coefficient that depends on fluid and wall material
s_coeff = 1.0, !! Coefficient that depends on fluid and wall material
/
&WIND_INTERFACE
WIND_CFL_file = "case3.cfl", !! What CFL file are we writing the wall temp. dist. to?
WIND_zone = 5, !! Which zone of the CFD grid are we changing this wall temp BC?
WIND_face = 3, !! Which face of the WIND_zone block are we specifying this temperature? (use 3 for Jmin, 4 for Jmax)
Wind_N_i = 121, !! Number of NODES along the i-direction on the zone we want to exchange wall temp and heat flux
Wind_N_k = 1, !! Number of NODES along the k-direction on the zone we want to exchange wall temp and heat flux
Which_FOGO_Face = "JMAX", !! WIND_face var corresponds to what face on FOGO's grid (options: JMAX OR JMIN -> case sensitive!)
WIND_h_flux_file = "qwall.dat" !! File where WIND places all heat flux information
/
REFERENCES