2011

Implementation of the conjugate heat transfer code in KIVA-4

Edward Ng
Michigan Technological University

Implementation of the Conjugate Heat Transfer Code in KIVA-4

By

Edward Ng

A Report

Submitted in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

Department of Mechanical Engineering-Engineering Mechanics

Michigan Technological University

2011

This report, “Implementation of the Conjugate Heat Transfer Code in KIVA-4", is hereby approved in partial fulfillment of the requirements for the Degree of

Master of Science in Mechanical Engineering

Department of Mechanical Engineering – Engineering Mechanics, MTU

Report Advisor: _____

Dr. Song-Lin Yang

Department Chair: _____

Dr. William W. Predebon

Date: _____

ACKNOWLEDGEMENTS

First of all I would like to thank my academic advisor, Dr. Song-Lin Yang, who encouraged me to pursue this research in graduate studies and guided me throughout my Master's degree. I cherish all the invaluable knowledge that I gained from him.

I would also like to thank Dr. Seong-Young Lee and Dr. Franz X. Tanner for serving in my defense committee who provided all their support in completing my report with their valuable suggestions.

I would like to offer very special thanks to Dr. F.C. Ting and Dr. James Yi of Ford Motor Company who provided me the opportunity to work on this project and for supporting me financially through the University Research Programs (URP).

My project would not be complete without the work of Dr. Egel Urip who developed the original CHT code and I feel very grateful to him.

Last but not least, I would thank my parents and family who gave me immense support which helped me to overcome every hurdle and to succeed; also I extend my thanks to all my friends at MTU who added colors to my graduate student life.

ABSTRACT

KIVA is an open Computational Fluid Dynamics (CFD) source code that is capable to compute the transient two and three-dimensional chemically reactive fluid flows with spray. The latest version in the family of KIVA codes is the KIVA-4 which is capable of handling the unstructured mesh. This project focuses on the implementation of the Conjugate Heat Transfer code (CHT) in KIVA-4. The previous version of KIVA code with conjugate heat transfer code has been developed at Michigan Technological University by Egel Urip and is be used in this project. During the first phase of the project, the difference in the code structure between the previous version of KIVA and the KIVA-4 has been studied, which is the most challenging part of the project. The second phase involves the reverse engineering where the CHT code in previous version is extracted and implemented in KIVA-4 according to the new code structure. The validation of the implemented code is performed using a 4-valve Pentroof engine case. A solid cylinder wall has been developed using GRIDGEN which surrounds $3/4^{th}$ of the engine cylinder and heat transfer to the solid wall during one engine cycle (0-720 Crank Angle Degree) is compared with that of the reference result. The reference results are nothing but the same engine case run in the previous version with the original code developed by Egel. The results of current code are very much comparable to that of the reference results which verifies that successful implementation of the CHT code in KIVA-4.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Scope of the Project

The main objective of this project is to implement the existing Conjugate Heat Transfer (CHT) code from the KIVA-3VR2 into the latest version of the KIVA family, KIVA-4. The reason for implementing in KIVA-4 is that it can handle the unstructured mesh and because of which the CHT model can be used for the most complex geometries. The other important objective or significance of this project is to completely understand the structure of the KIVA-4 code and all its features so to transform any existing code or model from the previous structured mesh code KIVA-3VR2 to the latest code. This first half of the report deals with the structure and features of the KIVA-4 and the important things to be taken care while implementing any scheme or model in KIVA-4. The later part explains the detailed procedure on how the CHT model can be simulated using the KIVA-4 and also explains various options available in the CHT code for different applications. The CHT code is customizable for various applications which are also explained in this report. Finally the CHT model is applied to the cylinder wall and the simulation results are compared with the results of the CHT code in KIVA-3VR2.

Chapter 2

Introduction

The family of 'KIVA codes' was first released in 1985 by Los Alamos National Laboratory. It was developed based on arbitrary Lagrangian – Eulerian (ALE) methodology and is written in FORTRAN. The first release, called KIVA, was capable of computing the compressible flows in simple geometries [Amsden, 1985]. A few years later, KIVA-II [Amsden, 1989] was released with further improvements in 1989 with improved capability of computing the transient two and three-dimensional chemically reactive fluid flows with spray. These codes integrated the essential underlying physics models (spray, combustion, turbulence, etc.) and moving boundaries in a computational fluid dynamics code to simulate internal combustion engines and thereafter, all its development was directed towards improving features to simulate the engine flows.

Later in KIVA-3, development was made to compute complex geometries through multi-block structured mesh. To make the generation of computational model easier, the complex geometries were split into several blocks with regular simple geometry for generating the mesh, which are then combined together to obtain the mesh for the whole geometry.

KIVA-3 had its application focusing on the piston, combustion chamber, etc. Due to the need of simulating complex movements of valves for which the KIVA-3 was not so efficient in, the capability of the KIVA-3 code was enhanced by addition of an effective model of valves using the same technique as that used for piston movements, called 'snapper.' The earlier

snappers ‘SNAPB’ and ‘SNAPT’ are used as snappers for the piston whereas two new snappers ‘SNAPVTOP’ and ‘SNAPVFCE’ were used as snappers for the valves. All the other features of the KIVA-3 have been retained as in KIVA-3V other than the naming conventions of files being changed to give more clarity on file usage.

The latest version in the family of KIVA codes is the KIVA-4, which has enhanced features as compared to KIVA-3V. The major change in KIVA-4 is the capability to handle unstructured mesh. Other physical changes encountered in the KIVA-4 will be discussed in a later section.

Chapter 3

CHT code in KIVA-3V

At MTU, a conjugate heat transfer mode (CHT) was implemented in KIVA-3V to monitor the temperature variation and in the engine metal components. Before this, there were publications regarding how the engine metal components were monitored, see Liu and Reitzs' [Liu, 1998]. They developed a heat conduction solver for the KIVA-II code by using both finite element and finite difference method. The information from fluid domain was spatially interpolated into a separate finite element code for the solid domain. However, the MTU's CHT code applied the finite volume approach to model temperature distribution inside the engine metal components. The CHT solver is capable of solving structured and unstructured solid domain. Moreover, the solid phase calculation is coupled with fluid phase thus the solution is more realistic. The CHT code was also successfully validated by using a FORD engine [Urip, 2006]. The code has the capability to solve the solid phase systematically with the fluid phase (a.k.a. Hydro Code), using the information such as wall heat flux history at the fluid-solid interface to proceed for the heat conduction calculation. Since KIVA-3V can only handle structured mesh, the grid points at gas-solid interface require much care. It should be noted that the grid points at the gas-solid interface need to be matched within the given tolerance precisely, in order to transfer the information from fluid domain to solid domain. Therefore, when the solid domain has degenerated elements, necessary interpolation has to be done with care at the fluid-solid interfaces.

Chapter 4

The Changes in KIVA-4

The major change in KIVA-4 is that the numeric for grid/ mesh generation has been generalized to unstructured mesh grids. In a structured mesh used in KIVA-3V, a regular connectivity is maintained using a 2D/3D array with only quadrilateral elements for 2D structures and hexahedral elements for a 3D structure. Unlike this, the unstructured meshes don't have a regular connectivity but any type of elements can be used to define the grid that includes prisms, pyramids, tetrahedral, etc. Though the neighborhood connectivity has to be explicitly stored consuming more storage, handling complex geometry will be a lot simpler. But the unstructured mesh has added some limitations on the snapping procedure and also requires changes in the Lagrangian phase of computation in the ALE methodology. These modifications lead to a few changes in the calculation of pressure solutions and the momentum fluxing. The detailed discussions can be obtained from the paper by Torres and Trujillo [Torres, 2006].

In this section, we are going to discuss about the changes in code structure and the representation of different variables. The code structure of KIVA-4 is different from the previous versions in order to facilitate the handling of unstructured mesh. The major changes in variables, is that the indexing of different components of the mesh becoming more specific in terms of addressing. In the previous versions, there was only node indexing used for defining all the

components of the mesh like the cells, faces, etc., but in KIVA-4, every components have their unique address like cell indices, node indices, edge indices, face indices, etc. This component specific indexing helps keeping the code in a more structured and organized way. Also, it is easier to access the properties of vertex or cell or face without much hassle. This also makes the code debugging easier, but the downside is that it requires more number of arrays/-variables than the previous versions [Torres, 2006].

In spite of having increased number of arrays and variables, taking the advantage of the latest FORTRAN programming language capability, almost all the arrays in the KIVA-4 code are dynamic, resulting in a very efficient memory allocation and management. Previously, the variables and the size of arrays were declared in those files with extension "i", whereas in KIVA-4, the variables are declared in subroutines and allocated later in another subroutines using "use" command. The primitive challenge in implementing the existing CHT code on to the unstructured KIVA-4 is the code structure and changes in handling the variables. The primary focus of this project was to modify the existing CHT code to adapt for the new variables and code structure so as to handle the unstructured mesh and the degeneration of nodes. The key differences in the KIVA-4 are discussed in the following sections.

4.1 Indexing

As mentioned above, the previous versions of KIVA has only node indexing as the unique address, such as i4, and all other components of the control volume are defined relative to the node address, such as relative to i4. Also there are three arrays used to define the front, left and bottom faces of a computational cell (bcl, bcf, bcb) [Amsden et al., 1989]. Any active cells

and nodes are defined by a flag associated with the bookkeeping arrays f(i4) & fv(i4). In KIVA-4 only the active cells, nodes, and faces are listed and stored in the respective array. So every calculation is done only for the active cells/ nodes. Every cell, node and face is addressed with a unique index and all of them are inter-related/ associated when read from the input file [Torres, 2006].

For a given cell, all the associated information is stored in different arrays with more generic representation. For example the face addresses of a given cell can be obtained by the following array, ***numfaces(iface,i4c,)*** *where* ***iface*** *is face index (1-left,2-front,3-bottom,4-right,5-derrire,6-top) and the* ***i4c*** *is the cell address. Similarly,* ***nodes(inode,i4c)*** *is for the node address i4c cell* [Torres, 2006].

4.2 Element Types

One of the important changes is that the element types are classified in a more detailed way than the previous version. The only element type in the previous version of the KIVA is the node type and additionally there were classifications for the boundary face. In KIVA-4, element types are available for every element of the mesh like for cells, nodes, faces, edges. For example in KIVA-3V, if we need the length of all the 12 edges of a cell, first need to identify all the nodes and then distance nodes on various combination has to determined using the x, y, z co-ordinates and this procedure has to be repeated every time length of the edge is required. But in KIVA-4, the edges associated with each cell are stored in an array and the length of those edges can also be retrieved from the array directly.

Table 1 - Node Types

KIVA-3VR2	**KIVA-4**
flfluid = 1.0	squishnodes = 2
flface = 2.0	domenodes = 3
flbowl = 3.0	bowlnodes = 5
flsqsh = 4.0	topbowlnodes = 7
fldome = 5.0	portnodes = 11
flhead = 6.0	axisnodes = 37
	inoutflownodes = 41
	presnodes = 43
	movingnodes = 47
	movingnodesvalve = 53
	solidnodes = 59
	solidhnodes = 61
	solidbnodes = 67

Table 2 - Face Boundary types

KIVA-3VR2	**KIVA-4**
moving = 1.0	moving = 10
solid = 2.0	The bottom surface of the first valve = 11
axis = 3.0	The top surface of the first valve = 12
fluid = 4.0	The bottom surface of the second valve = 13
periodf = 5.0	The top surface of the second valve = 14
periodd = 6.0	The bottom surface of the third valve = 15
inflow = 7.0	The top surface of the third valve = 16
outflow = 8.0	The bottom surface of the fourth valve = 17
presin = 9.0	The top surface of the fourth valve = 18
persout = 10.0	solid = 20
	solidh = 21 (for the cylinder head)
	axis = 30
	fluid = 40
	periodf = 50 (for a front periodic boundary)
	periodd = 60 (for a derriere periodic boundary)
	inflow = 70 (for a velocity inflow boundary)
	outflow = 80 (for a continuative outflow boundary)
	presin = 90 (for a pressure inflow boundary)
	presout = 100 (for a pressure outflow boundary)

The following sample code illustrates the difference in code logic between the KIVA-3VR2 and KIVA-4

KIVA-4	**KIVA-3VR2**
do i4c = 1, ncellsa	do i4 = ifirst, ncells
i1= nodes (1, i4c)	i1 = i1tab (i4)
i3 = nodes (3, i4c)	i3 = i3tab (i4)
enddo	enddo

Although KIVA-4 still includes the i1tab, i3tab and i8tab bookkeeping arrays, the algorithm above clearly shows that KIVA-4 has a new approach to locate a node of the fluid(gird) cell. It is important to know that, for each computational cell, KIVA-4 no longer uses left, bottom, and front faces for surface area and outward normal vector identification.

Chapter 5

Conjugate Heat Transfer Model

The main objective of this project is to implement the conjugate heat transfer (CHT) model into the KIVA-4 code. Finite volume method is used to model the temperature distribution in the engine metal components. The code can solve a combination of four different types of elements such as tetrahedron, pyramid, prism, and hexahedron. These meshes are used to transmit information from the fluid domain (KIVA code) to the solid domain. The nodes at the gas-solid interface should match perfectly in order for the code to obtain any information from the fluid domain [Urip, 2006].

To better illustrate the variables (Table 3) of the CHT code, two figures are shown below for this purpose. Figure 1 represents a four cell (*ncellsld = 4*) mesh with proper cell labeling whilst figure 2 shows the node addressing (*nvertsld = 18*) for this mesh.

In addition to that, it can also be observed that the computational mesh has four interior faces (*niface = 4*), 16 boundary faces (*nbface = 16*). It should be noted that thermal boundary must be defined in *itape5*. For instance, a constant temperature is boundary type 1 and convection is boundary type 2. If any unassigned boundaries occur, the program will treat those as adiabatic faces. Moreover, when running the KIVA-4 with the CHT model, a file known as kiva4solid must be used, simply because it contains all the element and node connectivity at the gas-solid interface and is described in Table 3 [Urip, 2006].

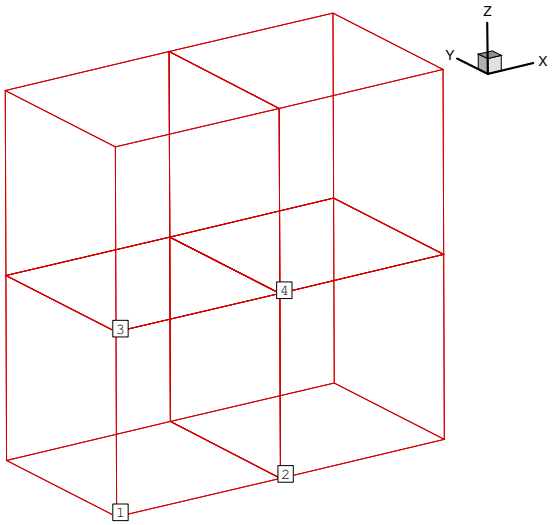


Figure 1 - Cell Indexing of Solid Mesh

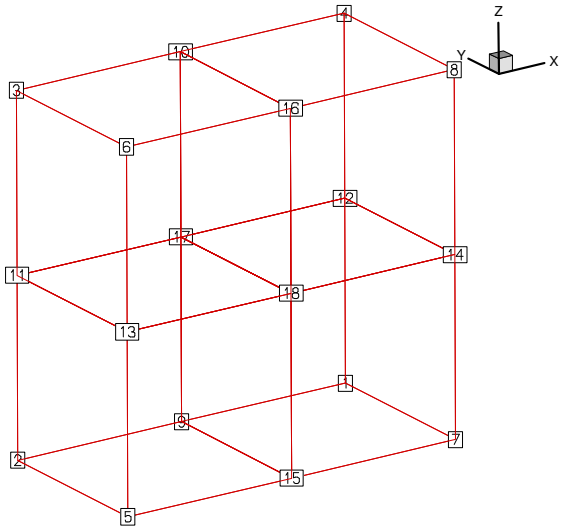


Figure 2 - Node Index of Solid Mesh

Table 3 - Solid Domain Arrays

ielement(*nc*,*10*)	• *nc*: cell index • *nc*: spans from one to *ncellsld* (total number of cells), $1 \le nc \le ncellsld$ • *1-10*: *1*: type of element (*1*-tetrahedron), (*2*-hexahedron), (*3*-prism), (*4*-pyramid) *2*: region identifier used to assign thermodynamic property *3-6*: node index of tetrahedron element, ielement(*nc*,*1*) = *1* *3-7*: node index of pyramid element, ielement(*nc*,*1*) = *4* *3-8*: node index of prism element, ielement(*nc*,*1*) = *3* *3-10*: node index of hexahedron element, ielement(*nc*,*1*) = *2*
kface(*ni*,*7*)	• Table of interior face (an interior face is shared by two elements with index: kface(*ni*,*6*) and kface(*ni*,*7*)) • *ni*: interior face index • *ni* spans from one to *niface* (total number of interior faces), $1 \le ni \le niface$ *1-3*: node index of triangle interior face

	1-4: node index of quadrilateral interior face *5*: type of interior face (*3*-triangle), (*4*-quadrilateral) *6*: cell index of the first element *7*: cell index of the second element
ibndry(*nb*,*7*)	• Table of boundary face • *nb*: boundary face index • *nb*: spans from one to *nbface* (total number of boundary faces), *1* ≤ *nb* ≤ *nbface* *1*: thermal boundary type *2*: type of boundary face (*3*-triangle), (*4*-quadrilateral) *3-5*: node index of triangle boundary face *3-6*: node index of quadrilateral boundary face *7*: cell index of the boundary face in question

Chapter 6

Procedure for CHT Simulation

Conjugate Heat Transfer (CHT simulation) is similar to the engine simulation including various pre-processing and then running the hydro code with post processing as the final process. The CHT code requires independent pre-processing similar to that of the fluid domain and apart from that there is additional pre-processing required, connecting the solid and fluid domain together. The detailed procedure to run the complete CHT model is explained below starting with the pre-processing.

6.1 Pre-Processing

As mentioned earlier, pre-processing involves a series of processes for generating all the input files required for running the hydro code. There are four different stages of the pre-processing in which the input data for the solid mesh and its connectivity with the fluid mesh is generated and finally converted to the input format of KIVA-4.

6.1.1 Step 1 - k3prepfluid

The first step is the conventional pre-processing for the fluid mesh which is usually used for any regular engine simulation. This process produces the actual mesh with all the nodes and all required information about the mesh based on the input. Usually the input consists of a manually prepared data set of different blocks of the given geometry with all the boundary conditions and the connectivity information. Then, during this process the data of small individual blocks are combined together to generate the overall mesh. In addition to that, the k3prepfluid has been modified [Urip, 2006] and therefore, while preparing iprep additional input parameters must be included. Those parameters are explained below in the input parameters section.

Primary Code used: ***k3prepfluid.f***

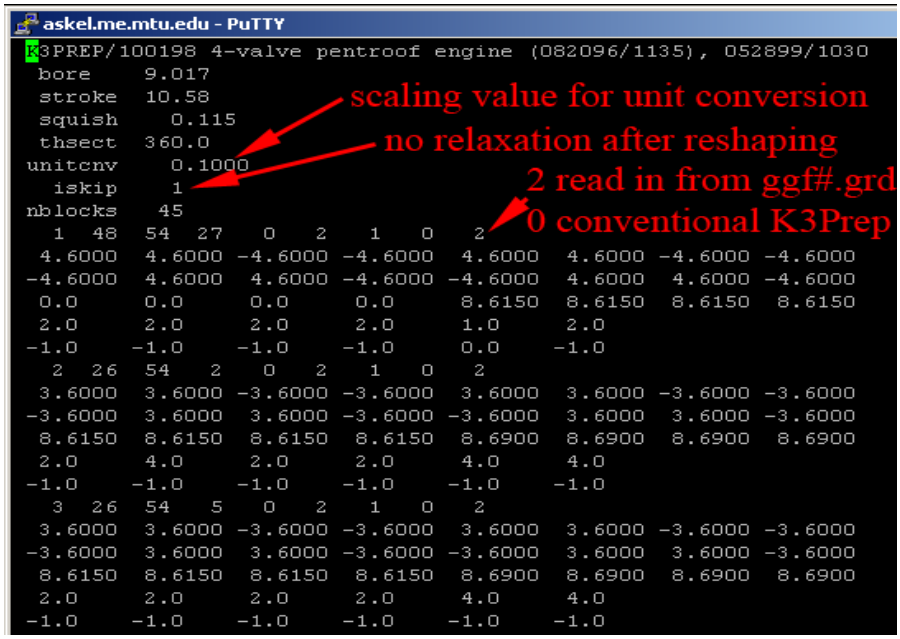
Supporting codes required: ***plotghost.f, utility.f***

Input files needed: ***iprep, comprep.i, ggf#.grd (if GRIDGEN is used)***

Output files produced: ***otape17, cornerlogfluid***

Input Parameters:

"***iprep***" is the input file which contains the data of different blocks of geometry. This is manually written in a formatted file. If any other software is used to build the geometry, then the input file should be in a specific format and some of the input parameters have to be changed. For example, if the GRIDGEN is used to generate the mesh, then the fluid mesh file should be named as "*ggf**#**.grd*" where # is the file number. Then the following input parameters has to be entered as shown on the console screen shot,



Unitcnv is simply a scaling factor for unit conversion. ***Iskip*** allows skipping the grid relaxation after the reshaping process [Urip, 2006].

6.1.2 Step 2 - k3prepsolid

This step is similar to the previous one but deals with various files for the solid domain/ mesh. The solid domain for an engine might include the cylinder wall, cooling water jacket, cylinder head, and piston. The CHT code is capable of dealing with all the above mentioned components. The moving and stationary parts are grouped and processed separately. This process itself has several sub stages which are explained in detail below.

a) **Stage 1:**

The first stage is exactly the same as the k3prepfluid, except the files contain different names and deals with all the solid parts, except the moving solid parts.

Primary Code used: ***k3prepsolid.f***

Supporting codes required: ***none***

Input files needed: ***iprepsolid, comsol.i***

Output files produced: ***solid17***

Input Parameters: *choose option* **= '1',** ***unitconv*** **= 0.1, id# = 999(large #)**

The *iprepsolid* can also be replaced by the input files generated from the GRIDGEN . For the GRIDGEN, the input file name has to be **"ggb#.grd"**

b) **Stage 2:**

Stage 2 is exactly the same as stage 1, but this process is for the moving solid domain such as piston. All the file requirements are same as before with little difference which is explained below,

Primary Code used: ***k3prepsolid.f***

Supporting codes required: ***none***

Input files needed: ***iprepsolid, comsol.i***

Output files produced: ***piston17***

Input Parameters: *choose option* **= '2',** ***unitconv*** **= 0.1, id# = 999(large #)**

The input file produced by GRIDGEN should be named as **"ggp#.grd".** If there is piston included in the solid domain, then after the *'piston17'* is produced, the same code is run again with the option **'3'** and the input files required are ***'piston17'*** and ***'solid17'.*** This process will produce a combined output file **"solid17"**

c) Stage 3:

In stage 3, the most important fluid-solid connectivity is established. This process generates the connectivity at the conjugate faces, i.e., the fluid-solid interface.

Primary Code used: ***k3prepsolid.f***

Supporting codes required: ***none***

Input files needed: ***solid17, itape17 (output from Step I renamed), cornerlogfluid, iprepsolid***

Output files produced: ***cornerlogsolid, conjugateface.dat***

Input Parameters: *choose option* **= '4', *unitconv* = 0.1, id# = 9999(large #)**

d) Stage 4:

This stage is meant for the additional format conversion as explained in k3prepsolid.f

The additional variables in the cornerlogsolid are explained in the below table, [Urip, 2006],

Table 4 –Element and Node connectivity

i4slr(*n*), i4flr(*n*), i4kmlr(*n*)	• *n*: conjugate face/fluid-solid interface index associated with left/right face • *n*: spans from one to *ncjgtlr* (total number of left and right conjugate faces), $1 \le n \le ncjgtlr$

	• i4slr(*n*): boundary face index of solid domain, see Table 4.1.1. • i4flr(*n*): face index of fluid domain • i4kmlr(*n*): boundary face index of solid domain directly below i4slr(*n*)
i4sfd(*n*), i4ffd(*n*), i4kmfd(*n*)	• associated with front/derriere face
i4sbt(*n*), i4fbt(*n*)	• associate with bottom/top face

6.1.3 Step 3 – Converter

The converter is the program used to convert the output files produced from the preprocessor to KIVA-4 input file format so as to use it for running the flow solver code. The converter basically changes the input format and in some cases changes the input variables to a different form.

Primary Code used: ***converter.f***

Supporting codes required: ***none***

Input files needed: ***itape17, cornerlogsolid***

Output files produced: ***kiva4grid, kiva4solid***

Input Parameters: ***none***

With the converter, all the pre-processing required to run the flow solver is completed. At the end of the pre-processing, there will be three files generated to be used as input files for the hydro code/ flow solver and they are,

1) kiva4grid

2) solid17

3) kiva4solid

The following flow chart explains the pre-processing in a more convenient way,

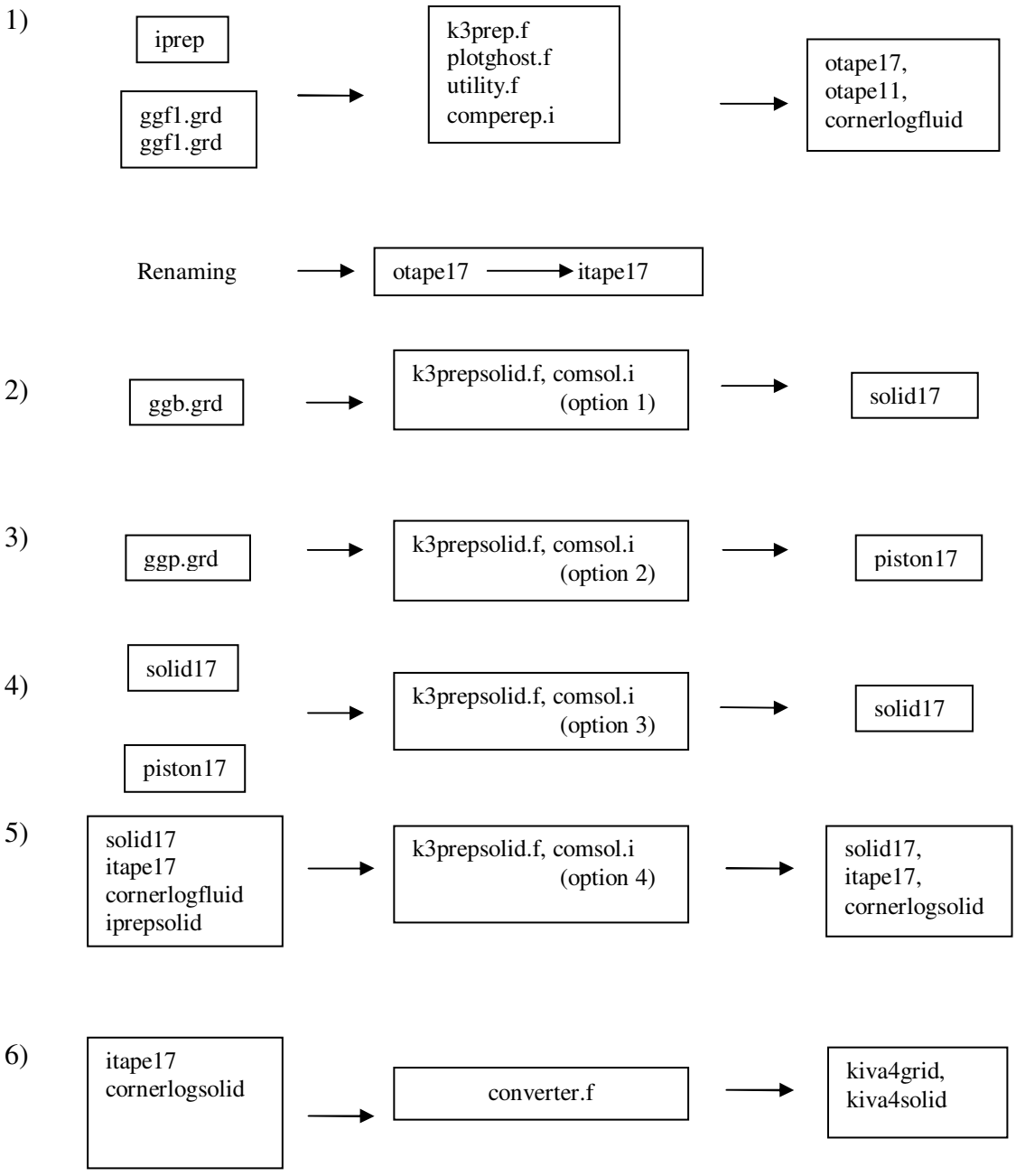


Figure 3 - Preparation Processes Flow chart

6.2 CHT code Options

An input parameter known as *icjgtht,* is used to define the mode needed for the run. If it is just solving the fluid phase alone, *icjgtht = 0*; for solving heat conduction equation alone *icjgtht = 1*; for solving both gas and solid phase, the so called conjugate heat transfer, *icjgtht = 2* and with *icjgtht = 3*, the code will not only solve the gas and solid phase but also record all the wall heat transfer history so that it can be later used for solving heat conduction equation alone. Let's consider case 1 for *icjgtht = 0,* case 2 for *icjgtht = 1,* and case 3 for *icjgtht = 2&3*.

The following figure illustrates the various processes to be run for the CHT code, the files required and the files produced.

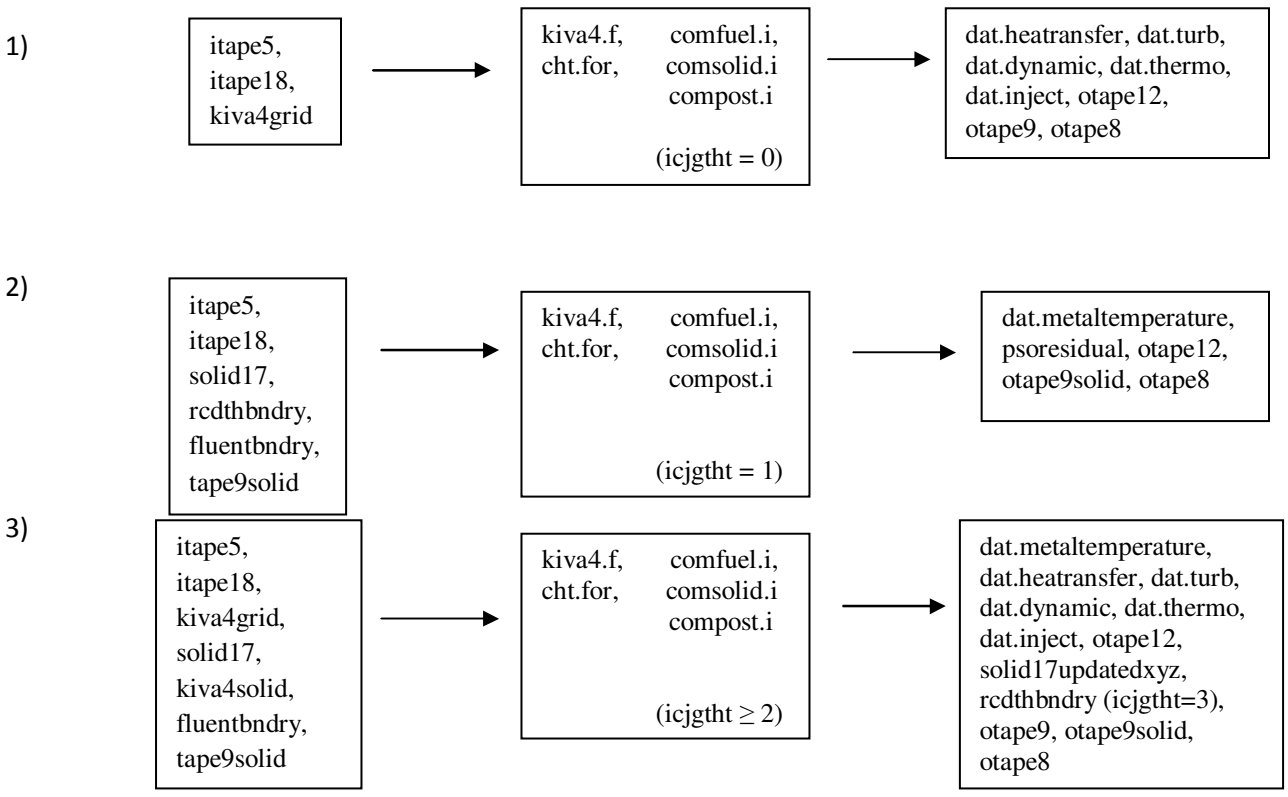


Figure 4 - Flow Chart for the CHT Code

There are several new variables introduced in the itape5 which are tabulated below, [Urip, 2006],

Table 5 – New Variables in itape5

icjgtht	• *0* – solving the gas phase alone • *1* – solving the solid phase alone • *2* – solving both the gas and the solid phases together, so called conjugate heat transfer • *3* – similar to *2* in addition the code will record the wall heat transfer history at the gas-solid interface to be used later on for solving the heat conduction equation alone
dtrcd	• Time step interval, unit is in crank angle, to record the heat transfer history. It is used in conjunction with *icjgtht = 3*.
atdcrcd	• Starting point of crank angle to record the heat transfer history. It is used in conjunction with *icjgtht = 3*.
epstsld	• Allowed relative error in implicit solution of heat diffusion in the solid domain, typically 1.0e-5
ksteady	• Used in conjunction with *icjgtht = 1*. • 0 – Solving transient heat conduction equation. • 1 – Solving steady state heat conduction equation. Transient thermal boundary, heat transfer history at conjugate interface, will be time averaged.
kbcjgt	• Used in conjunction with *icjgtht = 1*. • Thermal boundary condition, obtained by running the KIVA code with conjugate heat transfer mode, to be applied to the combustion chamber surface • 0 – Using heat flux history, $q_w''(\vec{x},t)$, subscript *w* for wall in the combustion chamber • 1 – Using convection history, $h_f(\vec{x},t)\left(T_f(\vec{x},t)-T_w\right)$, subscript *f*

	for fluid/gas
kbclnt	• Used in conjunction with *icjgtht = 1*. • Thermal boundary condition, obtained from FLUENT solver, to be applied to the cooling jacket surface • 0 – Using heat flux cooling boundary, $q''_{wc}(\vec{x})$, subscript *wc* for wall in the cooling jacket • 1 – Using convection cooling boundary, $h_{wc}(\vec{x})(T_c(\vec{x})-T_{wc})$, subscript *c* for coolant liquid
nregs	• Number of regions of solid domain for which *tempsldi* and *material* are to be read in.
relax	• Used in conjunction with *icjgtht = 1* and *ksteady = 1*. • Relaxation parameter used to speed up convergence rate. Its value ranging from unity to two. When *relax = 1*, point Gauss-Seidel iteration scheme is restored.
dtcastr, dtcaend, distrib	• Used in conjunction with *icjgtht = 1*, *ksteady* = 0, and transient thermal boundary, *rcdthbndry*, is provided. • Use time step size = $\frac{dtcaend - dtcastr}{distrib}$ when the current crank angle is between *dtcastr* and *dtcaend*. These flags are designed so that the code will use the above time step size when significant heat transfer to the wall occurs [Heywood, 1989], e.g., *dtcastr* = 350, *dtcaend* = 720, and distrib = 370 will use 1 degree crank angle time step size if the current crank angle falls between 350 and 720. Note that this condition applies to all engine cycles.
dtfast	• Used in conjunction with *icjgtht = 1* and *ksteady* = 0. • 1 – Using the most optimum time step size in addition to the time step size based on *dtcastr*, *dtcaend*, and *distrib*.
tempsldi(n)	• Initial cell temperature of solid domain for region *n*
material(n)	• Cell material type of solid domain for region *n*

	• Currently there are eight material types namely, Pure Aluminum (Al), Aluminum 2024-T6 Al2024), Copper (Cu), Pure Iron (Fe), Iron 99.75% Pure (Fe99), Artificial1 (Arti1), Artificial2 (Arti2), and Artificial3 (Arti3)
nsldbdry	• Total number of thermal boundary to be assigned for solid domain for which *aidxbdry* and *bdry* are to be read in.
aidxbdry(n)	• Boundary type specified in Gridgen, see Table 4.1.4.
bdry(n)	• Thermal boundary condition for solid domain: 14.0 – Convection $h*(T-T_\infty)$, $h\ T_\infty$ (2f8.3) h: convective heat transfer coefficient (ergs/s-K-cm^2) T_∞: free stream temperature (K) 15.0 – Constant temperature, *ct* (f8.3) *ct*: constant temperature (K) 16.0 – Sine wave temperature $ct + at * sin(omega * t)$, *ct*, *at*, *omega*, *ion_off* (3f8.3, i4) *ct*: constant wall temperature (K) *at*: amplitude wall temperature (K) *omega*: frequency of fluctuation (radians/s) *ion_off*: 1 – zero out negative temperature 0 – allow negative temperature 17.0 – Constant heat flux $q'' = cflux$, $cflux$ (f8.3) *cflux*: wall heat flux (ergs/s-cm^2) 18.0 – Sine wave heat flux $q'' = cflux + aflux * sin(omega * t)$, *cflux*, *aflux*, *omega*, *ion_off* (3f8.3, i4) 19.0 – Periodic-In Boundary 20.0 – Periodic-Out Boundary

6.3 Running the CHT code

The input parameter known as *itype* is added into itape5 in KIVA-4 to identify the mesh type whether structured (*itype* = 1) or unstructured (*itype* = 2). Additionally *itype* = 2 can also be used for the structured mesh where improved accuracy on the momentum fluxing is required.

There are also several other input parameters added in itape5, namely *numdiv, scf, diameterinjector, nsp and nspl.* More information about these variables can be obtained from the KIVA-4 user manual and has to thoroughly be understood before proceeding with the flow solver code. There are three cases explained below with different options of running the CHT code using the KIVA-4.

(a) Case 1- *icjgtht = 0*

It only solves the fluid phase; input files such as *itape5, itape18* and *itape17* are required. Since there are modifications in KIVA-4, an additional output file *dat.heatransfer* is generated when running case 1. This file basically contains the heat transfer rate of surface averaged of cylinder wall, cylinder head, and piston heat transfer rate.

(b) Case 2- *icjgtht = 1*

Running case 2 requires four additional input files compared to case 1. They are *rcdthbndry*, *fluentbndry*, *solid17* and *tape9solid*. *Solid17* is the only mandatory file for this run. It contains all the solid mesh data such as the element connectivity, thermal boundary face, etc., The other files are optional depending upon the availability of data and requirements. *Rcdthbndry* contains the wall heat flux history at the gas-solid interface and it is mainly used as one of the boundary conditions for the solid domain. The user has the option to choose whether

the history is time averaged (*ksteady = 1*) or to be treated as a transient boundary (*ksteady = 0*). *Fluentbndry* is obtained from FLUENT data and contains heat loss information due to the coolant flow. *Tape9solid* is very similar like *otape9*, it contains the temperature distribution of the solid phase. This file is used as an initial condition for the heat conduction calculation. An output file, *dat.metaltemperature* will be generated after running case 2. It contains the temperature distribution in engine metal components. If the parameter *ksteady* and *icjgtht* are set to unity, an output file called *psoresidual* will be produced. This file is actually used to monitor the convergence rate or to search an optimum relaxation parameter.

(c) Case 3- *icjgtht = 2 or 3*

In this case the KIVA-4 hydro code is run along with the heat conduction and solving for the temperature distribution through the solid domain. It consumes more computational time than previous cases.

After obtaining all the output files, the next step is post-processing to convert the results in the *otape9* and *otape9solid* files to the Tecplot format.

Chapter 7

Validation of the CHT Code

The CHT code implemented in KIVA-4 is validated using a "Four Valve case", one of the examples provided in the KIVA package. The reason for choosing this case is that, this is the only 3D case with a complex geometry available on both KIVA-3VR2 and KIVA-4. The solid part for this case is the cylinder wall of thickness 1.5mm and height of 76.625mm which covers the cylinder wall of the fluid mesh from the bottom to the edge of cylinder head. The solid mesh does not cover the cylinder head due to the complexity of the shape beyond this height. The solid cylinder wall is bounded by the conjugate face on the inner side at the fluid interface and the outer side of the solid wall is the coolant boundary with constant temperature as the initial condition. This case has been run on the KIVA-3V version first without fuel injection and then run with the KIVA-4 CHT code. The validation is performed with the major focus on the temperature distribution in the solid wall for two full cycles. First, the results of the fluid mesh is compared for various properties to the normal run without the CHT code to ensure that the implementation of the CHT code doesn't interfere with the calculations on fluid mesh. Later, the temperature distribution on the solid wall is compared between the results from KIVA-3V and KIVA-4.

The pre-processing is done based on the procedure explained in the pre-processing section of this report. The detailed procedure of running the CHT hydro code is provided below,

1. *cht.f*, *kiva4.f*, *comfuel.i*, *compost.i*, *comsolid.i*, *combdk3prep.i*, *itape5*, *itape18*, *kiva4grid*, *solid17*, and *kiva4solid* are needed for the hydro code run.
2. Run the hydro code by setting *icjgtht = 3* in *itape5*. This will allow you to record the wall heat transfer history at the gas-solid interface.
3. Several files are produced after running this HydroCode such as *rcthbndry* and *dat.metaltemperature*. *Rcthbndry* contains wall heat transfer history at the gas-solid interface. It will be used as a heat addition boundary condition for a steady and transient state calculation. If the HydroCode is split into for example 2 cycles (0 – 1440 degrees), the user has to rename the wall heat flux history for the first 720 degrees to *rcthbndry.1* so that when the code is running for the second cycle, the recorded wall heat flux history won't overlap. *Dat.temperature* contains the temperature distribution of the engine metal components.
4. And by now, the user should have *dat.temperature*, *rcthbndry.1* and *rcthbndry.2*.

7.1 Compiling Wall Heat Transfer Boundary

The code is purposely designed to combine all the separated files of wall heat transfer history into useful boundary information. Below is a step- by- step guide on how to run the code and consider the wall heat transfer history for the first engine cycle, which is saved in *rcthbndry.1* while the second engine cycle is recorded in *rcthbndry.2* [Egel, 2006].

1. Make sure that files such as *coolant.f*, *coolant.i* and all the *rcthbndry.#* are inside the folder.
2. Rename all the *rcthbndry.#* to *rcdthbndry#,* for instance *rcthbndry.1* to *rcdthbndry1* and *rcthbndry.2* to *rcdthbndry2*.

3. Run the thermal boundary compiler code and select option 10.
4. The code will pop up a message and enter 2 for *No* simply because you want the crank angle at the end of file *rcdthbndry2* to be 15.
5. A new message will pop up asking similar question and this time crank angle at the end of *rcdthbndry2* is 15 so enter 1 for *Yes*.
6. The code will produce a file known as *Summary.dat*. It contains crank angles at which wall heat transfer history was stored in *rcdthbndryn*.
7. Once you have the *rcdthbndryn* file, re-run the code and select option number 11 for compiling a transient boundary condition between 720 and 1440 crank angles for Hydro Code run.
8. Enter 721 for the starting zone and 1441 for the ending zone. The code will read in *rcdthbndry* and produce two files *rcdthbndry* and *summary.dat* where *rcdthbndry* contains transient thermal boundary condition for the heat conduction calculation.

7.2 Post Processing

The post processing code is specially written for converting the data obtained from the flow solver code into Tecplot format. The code has the capability to convert both the fluid and solid phase results.

1. *3dkivaconjugate.f* and *otape9solid* or *otape9* are needed for this step.
2. Rename *otape9solid* to *tape9solid*.
3. Run the post processing code and select option 3 which is plotting the solid phase only.

4. After that, user is asked to enter number zones to be written, user can either enter a large number e.g. 100 (to obtain full cycle of result) or at user specified zones numbers.
5. Then, user will be asked to enter whether the results are based on crank angle or time (seconds). Kindly select option 1 which is crank angle.
6. An output which is known as *k3postsolid.dat* will be generated and user may open it in Tecplot which is shown in the result section.

7.3 Results and Discussion of the Validation

Figure 5 shows the fluid and solid results of the temperature distribution being compared between KIVA-3V and KIVA-4. The temperature distribution is compared for the Crank Angle Degrees (CAD) 375 in KIVA3v and at the CAD 375 in KIVA-4. The peak temperature of the cycle is attained around this crank angle as there is no fuel injection involved and hence no combustion. Therefore, the CAD 360 at the end of compression should typically have the peak temperature for the cycle. We can see from the figures that the temperature distribution in the KIVA-4 results is slightly less than that of the KIVA-3V. Also, this temperature difference also contributed to the minor temperature difference in the solid domain. Apart from this difference there is no other significant difference found between the KIVA-4 and KIVA3V results, which validates the correctness of the CHT code implemented in the KIVA-4 code. Thus the validation of the implemented code is considered to be successful.

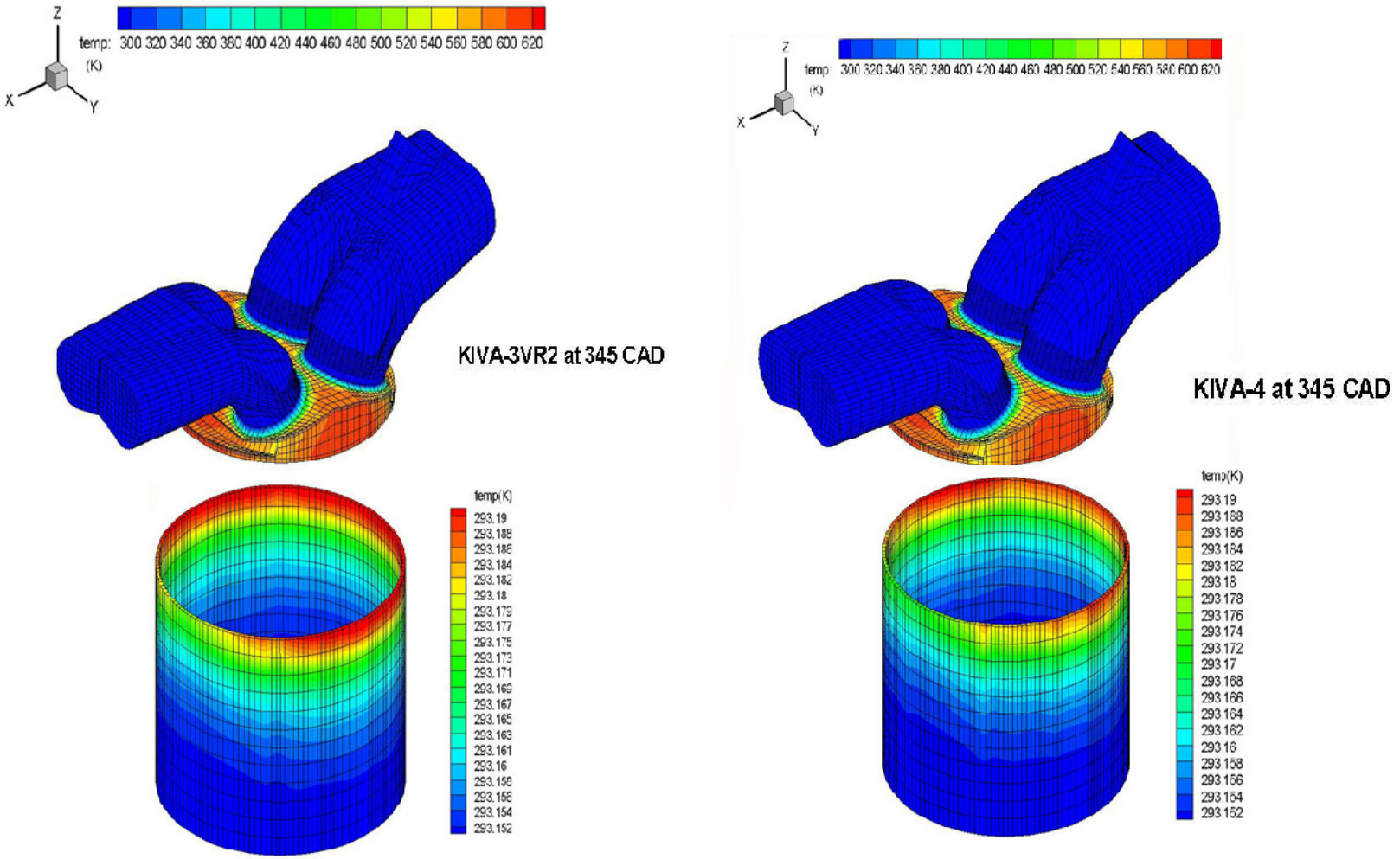


Figure 5 Comparison of first cycle Results - before ignition (CAD 345)

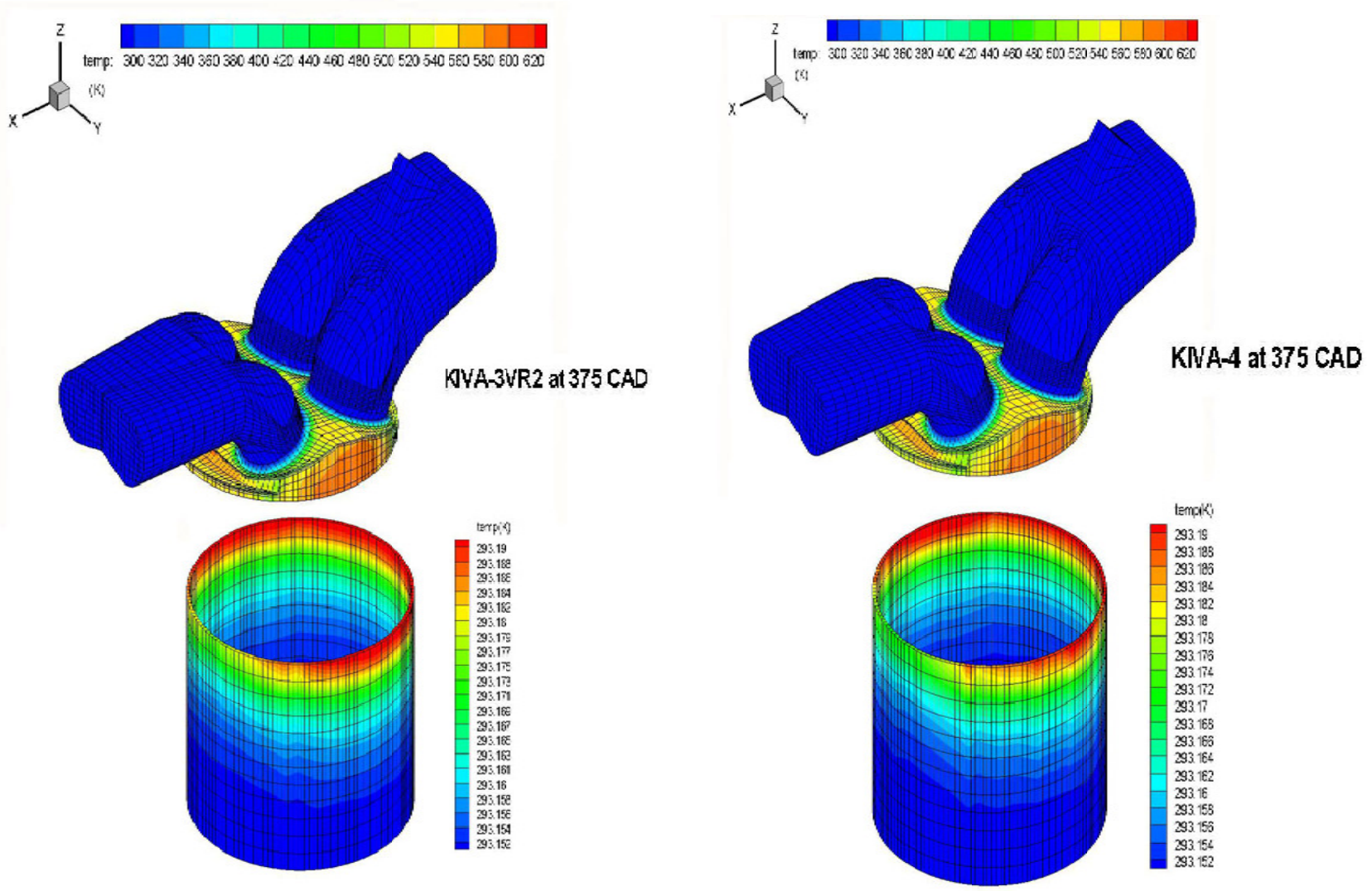


Figure 6 Comparison of first cycle Results - ater ignition (CAD 375)

Chapter 8

Conclusion and Future work

The Conjugate Heat Transfer code has been successfully implemented in the KIVA-4 with the capability to handle the unstructured mesh. The code has also been validated with a four valve case and found to be correct. But it completes only up to the code validation stage. For thorough verification of the implemented code, there are still more cases required to be run to verify the accuracy of the results which demand experimental data to compare with the results. Also the validation case has a structured mesh, so it is also required to evaluate this code with at least a simple case of unstructured mesh so as to ensure that the code can handle the unstructured mesh. But since the calculations and logic for the code is still based on unstructured mesh, which can account for the degeneration of nodes in the cell, it can be relied upon for handling the unstructured mesh based on the validation results. In addition to that, the 'wallfilm' subroutine is not validated simply because the example case does not include particles injection therefore, to achieve the code's optimum result, user would have to evaluate this code with an example case that includes particles injection.

A few drawbacks in the existing code have been found during various validation runs which will be included towards the improvement required for the future work. The hydro run for a full cycle of four stroke engine can be split into several runs instead of running it from CAD 0 to CAD 720. Usually depending upon the user requirement the cycle can be break at any

required crank angle. But it's been observed that if the cycle has been broke at CAD less than 180(end of suction), then it leads to the non-convergence of the solution for temperature. Further research is required to find out the source of the problem. One possible source could be the data output at the breaking point is not accurately transferred to the further cycle/ runs. The memory storage issue arises if the variables are properly allocated the array size which is another drawback.

Table 6 - Definitions of files used in CHT code

cornerlogfluid	- Contains node index of corners of each fluid block before the patching and dimensions/number of grid points in I-, J-, and K-direction of each fluid block.
solid17	- Contains node information, interior face table, boundary face table and element connectivity table.
gg#.grd	- Mesh file that produced from GRIDGEN. The file contains all the x, y, and z coordinates for the geometry. - ggb stands for "gridgenboundary" - ggp stands for "gridgenpiston" - ggf stands for "gridgenfluid"
iprepsolid	- The K3prepUntructured program reads in input file iprepsolid to generate fluid-solid element connectivity.
cornerlogsolid	- Contains connectivity tables for all conjugate faces.
k3prepunstructured.f	- This code is used for generate several tables required by the KIVA-Solid program (KIVA4 with CHT code)
com.i	- Global variable declaration
itape5	- Input parameters for KIVA code
itape18	- Valve lift history
itape17	- Contains all the x,y, and z coordinates, connectivity tables, boundary condition, etc. It is an input file for the flow solver code (KIVA).
rcdthbndry	- Contains the wall heat flux history at the gas-solid interface. This information will be used as one of the boundary conditions for the solid domain.
Fluentbndry	- Contains heat loss information due to the coolant flow. It can be obtained from FLUENT data.
tape9solid	- Renamed otape9solid, is very similar to otape9, except it only contains temperature solution of the solid phase.

eurip.f	- Contains the CHT code.
dat.heatransfer	- Contains surface averaged of cylinder wall, cylinder head, and piston heat transfer rate.
dat.thermo	- Is has been improved to include fuel vapor mass.
dat.metaltemperature	- Contains the wall temperature changes. The wall temperature in this case is defined as an ensemble average of temperature of all solid elements, which one of the faces is a conjugate interface.
psoresidual	- Contains residual value from the point successive over-relaxation iteration scheme and the total number of iteration when the steady state heat conduction solver is invoked (icjgtht = 1 and kstead = 1).

REFERENCES

Amsden, A. A., O'Rourke, P. J., and Butler, T. D., "KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays," *Los Alamos National Laboratory Report LA-11560-MS, Los Alamos, New Mexico, 1989.*

Amsden, A. A., "KIVA-3: A KIVA Program with Block-Structured Mesh for Complex Geometries," *Los Alamos National Laboratory Report LA-12503-MS, Los Alamos, New Mexico, 1993.*

Amsden, A. A., "KIVA-3V: A Block Structured KIVA Programs for Engines with Vertical or Canted Valves, *Los Alamos National Laboratory Report LA-13313-MS, Los Alamos, New Mexico, 1997.*

Amsden, A. A., "KIVA-3V, Release 2, Improvements to KIVA-3V," Los Alamos National Laboratory Report LA-13608-MS, *Los Alamos, New Mexico, 1999.*

Liu, Y. and Reitz, R. D., "Modeling of Heat Conduction within Chamber Walls for Multidimensional Internal Combustion Engine Simulation," International Journal of Heat and Mass Transfer, Vol. 41, pp. 859-869, 1998.

Torres, D. J., Mario F. Trujillo, "KIVA-4: An unstructured ALE code for compressible gas flow with sprays", *Elsevier, 2006.*

Torres,D. J., "KIVA-4 Manual,",*Los Alamos, New Mexico, 2006.*

Urip, Egel, "THE KIVA CODE WITH CONJUGATE HEAT TRANSFER MODEL FOR IC ENGINE SIMULATION", *Michigan Technological University, 2006.*