Hydrogen Mixing Studies (HMS) User's Manual

Manuscript Completed: August 1994
Date Published: December 1994

Prepared by
K. L. Lam, T. L. Wilson, J. R. Travis*

Los Alamos National Laboratory
Los Alamos, NM 87545

Prepared for
Division of Systems Research
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
NRC Job Code W6100

*Presently with
Science Applications International Corporation
2109 Air Park Rd., SE
Albuquerque, NM 87106

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

yu

DISCLAIMER

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

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Abstract

Hydrogen Mixing Studies (HMS) is a best-estimate analysis tool for predicting the transport, mixing, and combustion of hydrogen and other gases in nuclear reactor containments and other facilities. It can model geometrically complex facilities having multiple compartments and internal structures. The code can simulate the effects of steam condensation, heat transfer to walls and internal structures, chemical kinetics, and fluid turbulence. The gas mixture may consist of components included in a built-in library of 20 species.

HMS is a finite-volume computer code that solves the time-dependent, three-dimensional (3D) compressible Navier Stokes equations. Both Cartesian and cylindrical coordinate systems are available. Transport equations for the fluid internal energy and for gas species densities are also solved. The implicit, iterative computation of pressure in HMS's algorithm for solving the coupled governing equations, which uses an efficient preconditioned conjugate gradient matrix solver, allows simulation of both supersonic and subsonic fluid flows without severe limitation on time-step size due to the fluid sound speed.

HMS was originally developed to run on Cray-type supercomputers with vector-processing units that greatly improve the computational speed, especially for large, complex problems. Recently the code has been converted to run on Sun workstations. Both the Cray and Sun versions have the same built-in graphics capabilities that allow 1D, 2D, 3D, and time-history plots of all solution variables. Other code features include a restart capability and flexible definitions of initial and time-dependent boundary conditions.

This manual describes how to use the code. It explains how to set up the model geometry, define walls and obstacles, and specify gas species and material properties. Definitions of initial and boundary conditions are also described. The manual also describes various physical model and numerical procedure options, as well as how to turn them on. The reader also learns how to specify different outputs, especially graphical display of solution variables. Finally sample problems are included to illustrate some applications of the code. An input deck that illustrates the minimum required data to run HMS is given at the end of this manual.

Executive Summary

This document is the third of a series of manuals on the Hydrogen Mixing Studies (HMS) program, which is being sponsored by the US Nuclear Regulatory Commission (NRC) as a best-estimate tool for nuclear containment analyses involving hydrogen and cooling issues. This manual describes the input needed to run HMS and provides guidance on how to use the code. The previous two manuals are the Theory Manual (Ref. 1), which discusses the theoretical foundation and physical models in the code, and the Assessment Manual (Ref. 2), which reports some calculations that were used to assess the code with both analytical solutions and experimental data.

HMS can model geometrically complex facilities having multiple compartments and internal structures. The code can simulate the effects of steam condensation, heat transfer to walls and internal structures, chemical kinetics, and fluid turbulence. The gas mixture modeled may consist of components included in the built-in library of 20 species.

HMS is a finite-volume computer code that solves the time-dependent, three-dimensional (3D) compressible Navier Stokes equations. Both Cartesian and cylindrical coordinate systems are available. Transport equations for the fluid internal energy and for gas species densities are also solved. The implicit, iterative computation of pressure in HMS's algorithm for solving the coupled governing equations allows simulation of both supersonic and subsonic fluid flows without severe limitation on time-step size because of the fluid sound speed.

The code version described in this manual is designated HMS-93. HMS currently is maintained at the Los Alamos National Laboratory as a subset of a larger code package called GASFLOW, which is supported by the US Department of Energy (DOE) to address various nuclear and non-nuclear facility safety issues. Therefore, this code version is also called GASFLOW 1.0, to properly identify the version for configuration control. We maintain the DOE- and NRC-sponsored code versions under the same system because this is a more effective use of government resources.

Features of the HMS code include a restart capability and a graphic package that allows 1D, 2D, 3D, and time-history plots of all solution variables. Definition of initial conditions is rather flexible—fluid temperature, pressure, and composition at arbitrary regions of the mesh and temperature and material of solid thermal structures may be specified. Different types of boundary condition, which may be time dependent, can be specified on various portions of the computational domain boundaries and on internal wall and obstacle surfaces.

HMS was originally developed to run efficiently on Cray supercomputers so that large, complex problems can be solved in a reasonable time. To provide easier

access to the code by the general users, we have improved the portability of the code so that it will run on Sun workstations. This user's manual is applicable to both versions.

This manual explains how to set up a 3D mesh, either in Cartesian or cylindrical geometry, and define walls and obstacles. It describes how to specify gas species and material properties. Definitions of initial and boundary conditions are also explained. The manual also describes various physical model and numerical procedure options, and how to activate them. The reader should also be able to specify different outputs, especially graphical display of solution variables. Sample problems are included to illustrate some applications of the code. All input variables, their description, and default values are summarized in Appendices 1-4, which the user could use for quick reference. Appendix 5 shows an input file that contains the minimum required data to run a calculation, which should give new users a quick introduction on how to set up simple HMS problems.

Acknowledgements

The authors would like to thank Eric Haytcher for his effort in developing the workstation version of HMS. We also thank George Niederauer and Peter Royle for their helpful comments during preparation of this document.

CONTENTS

	Page
ABSTRACT	iii
EXECUTIVE SUMMARY	v
ACKNOWLEDGEMENTS	vii
I. INTRODUCTION	1
A. Code Capabilities	1
B. Computational Method	1
C. General Code Features	3
II. INPUT DATA FORMAT	4
III. GEOMETRY DEFINITION	8
A. Cell Labeling Convention	8
B. Mesh Generation	9
1. Direct Input of Grid Locations	9
2. Automatic Mesh Generation	10
C. Definition of Walls and Obstacles	14
1. Walls	14
2. Obstacles	15
D. Checking Geometric Model	16
IV. SPECIFICATION OF GAS SPECIES AND PROPERTIES	20
A. Definition of Gas Species	20
B. Definition of Transport Properties	22
V. INITIAL AND BOUNDARY CONDITIONS	23
A. Specification of Initial Conditions	24
1. Fluid Composition and State	24
2. Fluid Velocities	25
B. Specification of Boundary Conditions	25
1. Global Definition	26
2. Local Definition	28
VI. DEFINITION OF SOLID HEAT STRUCTURES	31
A. Wall Heat Structure	32
B. Slab Heat Structure	32
C. Sink Heat Structure	34
VII. PHYSICAL MODEL OPTIONS	37
A. Body Forces	37
B. Diffusion of Mass, Energy, and Momentum	38
C. Turbulence	39
1. Algebraic Model	40
2. Subgrid Scale Model	40
3. $k-\epsilon$ Model	42
D. Combustion	43
E. Heat Transfer	44
VIII. OPTIONS ON NUMERICAL SOLUTION PROCEDURE	44
A. Pressure Iteration	44
B. Time-Step Control	46

	C. Advection Scheme	48
IX.	OUTPUT AND RESTART	48
	A. Graphical Output	48
	1. Basic Plots	49
	2. Geometry Plots	49
	3. Time-History Plots	49
	4. Profile Plots	52
	5. Contour Plots	54
	6. Velocity Vector Plots	56
	B. Printed Output	57
	C. Output To Terminal	58
	D. Restart	58
X.	CODE MODIFICATION FOR SPECIAL USE	59
XI.	GENERAL USER GUIDANCE	60
XII.	SAMPLE PROBLEMS	66
	A. Von Karman Vortex Street	66
	B. Hydrogen Burn	67
	C. HDR Containment Facility Test T31.5	68
	REFERENCES	105
	Appendix 1. Summary of Variables in NAMELIST Group xput	106
	Appendix 2. Summary of Variables in NAMELIST Group meshgn	112
	Appendix 3. Summary of Variables in NAMELIST Group rheat	113
	Appendix 4. Summary of Variables in NAMELIST Group grafic	114
	Appendix 5. Sample Input Deck with Minimum Data Required	117

I. INTRODUCTION

A. Code Capabilities

Hydrogen Mixing Studies (HMS) is a best-estimate computer code developed at the Los Alamos National Laboratory (LANL) for the US Nuclear Regulatory Commission (NRC) for predicting the transport, mixing, and combustion of hydrogen and other gases in nuclear reactor containments and other facilities. The code can model geometrically complex facilities having internal structures and multiple compartments. The gas mixture modeled may consist of components included in a built-in library of 20 species (see Table III on p. 21). The fluid flow modeled may be laminar or turbulent, subsonic or supersonic. Momentum, heat, and mass transfer within the fluid is determined by physical mechanisms such as diffusion (molecular and/or turbulent) and convection. Heat conduction in solid structures is calculated and is coupled to the fluid dynamics through the wall temperatures and heat fluxes at the fluid-solid interfaces. If steam is present, the code predicts its rate of condensation based on the local wall temperature and bulk fluid conditions. The (simplified) chemical kinetics of the burn of a hydrogen-air-steam mixture can be solved simultaneously with the fluid dynamics to predict flame propagation and acceleration.

A typical application of HMS may be in predicting stratification of hydrogen distribution in a nuclear reactor containment building during the course of a severe accident in which large amounts of the flammable gas are produced. In analyzing containment designs (such as the AP-600 Passive Containment Cooling System) that lack active mixing mechanisms such as fans and internal sprays, but rather rely on natural circulation for cooling and mixing the containment atmosphere, the three-dimensional (3D), multi-species, variable-density capabilities of an analysis tool such as HMS are useful. The calculation will identify *local* regions of high hydrogen concentration within the multicompartment containment geometry where steam condensation also is occurring. The maximum hydrogen concentration can then be compared against flammability and detonation limits established experimentally to assess the risk of a hydrogen burn. The calculation can be carried one step further by assuming that the hydrogen gas mixture is ignited to determine the resulting pressure and temperature loads on the containment structures and safety-related equipment. Combustion modes that can be calculated include diffusion flames and propagating deflagrations. HMS is especially useful in predicting local pressure spikes in narrow passages between subcompartments where the interaction between fluid turbulence, temperature, and fuel concentration in the accelerated jet can be expected to give rise to instantaneous ignition.

B. Computational Method

In this section, we briefly summarize the computational method adopted in HMS. This is included so that the code user can quickly review the numerical

approach and models. Further details on the theoretical aspects of HMS are given in Ref. 1.

HMS is a finite-volume code that solves the time-dependent, 3D, compressible Navier-Stokes equations. Transport equations for the internal energy and for multiple gas species are also solved. The computational domain is discretized by a mesh of regular orthogonal cells in either Cartesian or cylindrical geometry. Primary hydrodynamic variables such as density, internal energy, and pressure are defined at cell centers, whereas the components of vector quantities such as velocity and mass flux are defined at the appropriate cell faces. A linearized Arbitrary-Lagrangian-Eulerian method is used for approximating the solution to the coupled mass, momentum, and energy conservation equations. The implicit, iterative pressure computation in this method, which uses an efficient preconditioned conjugate gradient matrix solver, allows simulation of both high- and low-speed (low-Mach-number) flows without the time-step restrictions that are caused by the fluid sound speed. The computational time-step size, however, is controlled automatically in the code so that the material Courant limit and numerical stability criteria resulting from various diffusion processes are not violated.

To model fluid turbulence, HMS provides an option for three turbulent models. These are, in the order of increasing complexity, the algebraic, subgrid-scale, and $k-\epsilon$ models, which are the respective zero-, one-, and two-transport equation models that compute the turbulent velocity and length scales required to determine the turbulent diffusivity. Turbulent diffusivity, together with its molecular counterpart, is used to determine gradient diffusion fluxes in the momentum, the internal energy, and the species mass transport equations.

Heat conduction within walls and structures is one-dimensional. The solid heat conduction equations are approximated with an implicit, finite-difference formulation that results in solution of tridiagonal matrices. Rates of heat transfer and condensation to walls and structures are calculated from the Reynolds analogy between momentum, heat, and mass transfer. A model is available to account for the enhanced mass- and heat-transfer rates in the presence of high mass fluxes toward the wall (e.g., during steam condensation). A term that accounts for the cooling effect caused by gas expanding into the volume space vacated by steam condensation is included in the energy equation.

Chemical energy of combustion involving hydrogen or other fuels provides a source of energy within the gaseous region, in addition to changing the composition of the gas mixture. HMS uses a one-step, global, chemical-kinetics model to simplify the actual chemical processes. (In the case of a hydrogen-nitrogen-oxygen-steam system, detailed chemical kinetics may involve up to about 50 intermediate reactions.) The model is based on a modified Arrhenius rate law that calculates the local fuel and oxidizer concentrations. The finite-rate chemical equation is solved implicitly for the fuel concentration when the fuel-oxidizer mixture is fuel-lean and

for the oxidizer or reactant concentration when the fuel-oxidizer mixture is fuel-rich. The procedure ensures that combustion gas components will never be driven negative, regardless of the time-step size.

C. General Code Features

HMS is a FORTRAN computer code originally developed to run on the Cray supercomputers at LANL. It has been optimized to take advantage of the vector processing units in such computer architecture to gain performance speed-up of almost an order of magnitude (compared to running the code in scalar mode) for some problems. Recently, the code has been converted to run on Sun workstations so that it can be readily installed at various research sites in the US and around the world.

A code calculation may be started from prescribed initial conditions or from the solution of a previous run. The restart capability is very useful when performing large-scale computations where a complete run may require many hours of CPU time on a supercomputer. Definition of initial conditions is rather flexible—fluid temperature, pressure, and composition at arbitrary regions of the mesh and temperature and material of solid thermal structures may be specified. Different types of boundary condition, which may be time dependent, can be specified on various portions of the computational domain boundaries and on internal wall and obstacle surfaces.

In HMS, the centimeter-gram-second (cgs) system is used for the units of dimensional quantities. Therefore, the user should use the following units when preparing input data for the code:

Length	Mass	Time	Pressure	Temperature	Energy
<i>cm</i>	<i>g</i>	<i>sec</i>	$\frac{\text{dynes}}{\text{cm}^2}$ (0.1 Pa)	<i>K</i>	<i>ergs</i> (10^{-7} J)

If the cylindrical coordinate system is used to set up the mesh for the computation, then input values for azimuthal coordinates, if required, must be in degrees (rather than in radians).

HMS has a built-in graphics package for displaying different views of the mesh and the computational results. In this regard, it is different from many codes that require a separate postprocessor to display the results. Actually, the code can be conveniently used as a graphics postprocessor to plot additional results after an analysis run has been completed. Available options include 1D profile, 2D contour and vector, 3D vector, and time-history plots for all hydrodynamic variables and for temperatures in all heat-conducting solid structures. Although the graphics have been developed as the primary tool for analyzing the computed data, several printed output files are written by the code to provide the user additional information about the run. Table I lists all the files used or written by HMS.

Table I. Description of Input and Output Files in HMS

File Name	Description
ingf	Input data text file.
pgf	Graphics metafile.
gfout	Output listing text file.
meshmap	Text file containing cell status and neighbor list information.
cyclinfo	Text file containing time-step and iteration information for each cycle of calculation.
tapel6	Text file echoing all NAMELIST input variables.
gfd1, gfd2, ...	Binary dump files used for restart calculations. The number of dump files produced can be controlled through user input.
pth1, pth2, ...	Binary file containing time history data. The number of files generated depends on the number of time history plots being asked for.

II. INPUT DATA FORMAT

To run HMS, the user must prepare an input file that contains data required for the problem calculation and for specifying any desirable output options. The input file is called `ingf`. The user must limit the input file to 80 columns wide except for optional comments. Figures 1a-b show the listing of the input for a sample problem. The first three lines contain alphanumeric data for problem identification purposes. These input data are

Line	Data Format	Description
1	A80	Title of problem to appear on all pages of graphical output and printed output
2	A10	Label to appear on printed output
3	A64	Special plot file label

HDR T31.5 Test Simulation, 23 APR 1993.

TSA-8 LAM

HMS-HDR

NOTES: Cylindrical mesh generated by directly entering the mesh line coordinates. Many definitions of walls and obstacles (mobs) and heat sinks have been eliminated in this sample input to conserve space. In addition, helium has been substituted for light gas. Initial temperature = 32 C.
Run 1: 0-60 sec

\$xput

```
autot      = 1.0,
cyl        = 1.0,
delt0      = 0.002,
deltmin    = 1.000e-04,      deltdtmax = 1.000e-00,
epsi0      = 1.000e-05,
epsimax    = 1.000e-05,      epsimin  = 1.000e-05,
gz         = -980.0,
itmax      = 1000,
maxcyc     = 9999999,
ittyfreq   = 100,
nu         = 0.15,      prandtl  = 0.7,      schmidt  = 0.45,
idiffmom    = 1,      idiffme   = 1,
tmodel     = 'alg',
pltdt      = 40.0,      prtdt    = 5000.0,
twfin      = 60.0,
tddt       = 1.000e+06,
velmx      = 5.0,
ibb        = 1,      ibt       = 1,
ibs        = 4,      ibn       = 4,
ibw        = 1,      ibe       = 1,
```

mat = 'air', 'h2o', 'he',

```
gasdef(1,1) = 1, 10, 1, 24, 1, 31, 1,
              1.00000e6, 305.0, 1, 0., 0.,
              1, 1.0000,
```

```
mobs = 1, 2, 1, 24, 1, 3, 1, 1,
        3, 4, 1, 6, 1, 2, 1, 1,
        8, 9, 3, 6, 16, 17, 1, 1,
```

```
walls = 5, 6, 5, 7, 22, 22, 1, 1,
         9, 9, 1, 2, 7, 10, 1, 2,
```

\$end

H'E A T - T R A N S F E R

\$rheat

```
ihflag = 1,
nhteslab = 10,      nhtewall = 10,
tsink0 = 305.,      tslab0 = 305.,      twall0 = 305.,
```

```
walldef(1,1) = 1, 100.,
walldef(1,2) = 2, 3.,
```

```
sinkdef = 1, 9, 1, 24, 1, 2, 1, 2, 8.454058e+05, 0.94,
           1, 9, 1, 24, 3, 4, 1, 2, 1.948350e+06, 0.64,
```

\$end

Lines after the third one, and lines between NAMELIST groups are ignored, so they can be used for comments.

Fig. 1a. Sample HMS input listing, showing two input NAMELIST groups, xput and rheat.

```

-----
      M E S H
-----
$meshgn
iblock = 1,

xgrid =
  0.0,      150.0,      230.0,      350.0,      500.0,      650.0,
  800.0,      950.0,      1000.0,      1060.0,

ygrid =
  0.0,      020.0,      036.0,      052.0,      068.0,      84.0,
  094.0,      109.0,      124.0,      139.0,      154.0,      169.0,
  184.0,      197.0,      213.0,      227.0,      241.0,      255.0,
  270.0,      289.0,      305.0,      322.0,      340.0,      360.0,

zgrid =
  0.0,      120.0,      220.0,      400.0,      500.0,      600.0,
  700.0,      980.0,      1080.0,      1180.0,      1280.0,      1440.0,
  1540.0,      1700.0,      1800.0,      2050.0,      2175.0,      2275.0,
  2455.0,      2555.0,      2725.0,      2900.0,      3050.0,      3300.0,
  3550.0,      3750.0,      3885.0,      4200.0,      4800.0,      5500.0,
  5600.0,

$end
-----
      G R A P H I C S
-----
$grafic
thdt = 0.1,

pnt = 1, 1, 23, 1,
      10, 24, 23, 1,
      1, 1, 24, 1,
      10, 24, 24, 1,
      1, 20, 30, 1,
      10, 20, 30, 1,

htltp = 6, 7, 22, 1, 'wall', 'top',
        3, 10, 2, 1, 'slab', 'west',
htthp = 7, 9, 2, 1, 'sink', 0,

p1d = 5, 6, 'tk', 0,
c2d = 1, 2, 'tk', 0,
      3, 4, 'tk', 0,
      1, 2, 'pn', 0,
      3, 4, 'pn', 0,
      1, 2, 'vf', 2,
      3, 4, 'vf', 2,
v2d = 1, 2, 1,
      3, 4, 1,
thp = 7, 11, 24, 1, 'pn', 0,
      7, 11, 24, 1, 'tk', 0,
      3, 22, 30, 1, 'vf', 1,
      3, 22, 30, 1, 'vf', 2,

$end
$special
$end
$parts
$end

```

Blanks between variables
and between lines within
a NAMELIST group
record are ignored

Fig. 1b. Continuation of sample input listing, showing four additional NAMELIST groups: meshgn, grafic, special, and parts. Note that the last two groups are blank.

The rest of the input data are read into the code via six groups of NAMELIST variables.

The main purpose of each NAMELIST group is listed below:

NAMELIST Group	Purpose
xput	Definition of physical properties, initial and boundary conditions, code control, and numerical solution option data.
meshgn	Specification of computational mesh.
rheat	Specification of heat-transfer data.
grafic	Definition of graphical output options.
parts	Specification of variables related to particle transport (not available in HMS).
special	Definition of miscellaneous 3D plotting variables.

Note that the NAMELIST groups `parts` and `special` are included but left blank in the sample input deck shown in Fig. 1b. Input variables in these two groups are not used in HMS, but the input file must contain the group names (though no data) so the input data will be processed successfully. All variables for the other four NAMELIST groups are described in Appendices 1-4.

The NAMELIST feature offers an easy way of specifying input data. Within each NAMELIST group, both scalar and array variables can be defined conveniently with their desired values. The order of appearance of the variables is unimportant. As can be seen in Figs. 1a-b, all input data values are clearly associated with the corresponding variable names, which makes it very simple for a user to modify the input deck to run other problems.

An input NAMELIST group record can consist of one or more lines (physical records). Columns 1 and 81 and beyond are ignored. In the first line, `$name` (the dollar sign delimiter followed immediately by the name of the NAMELIST group) must appear beginning in column 2 and then be followed by one or more blanks. The remaining portion of the input record may contain as many variables as needed, with their assigned values, and in any order. Commas are used to separate items and to separate input values for elements of the same array. Input items take the following forms:

```
variable=value  
array=value[,value,] . . .  
array(subscripts)=value[,value,] . . .
```

where *subscripts* are integer constants identifying particular elements of the array. (Brackets indicate optional entries.) Multi-dimensional array values are assigned in storage order. Any value can be repeated by $n*value$, where n is the

repetition count. A delimiter (\$) terminates the NAMELIST group record. The three characters "end" after the delimiter, used in the input deck shown in Figs. 1a-b, are not required to specify end of a group record. They are included to improve readability of the input.

Blanks can be used to improve legibility, but must not be embedded in names, values, or between an array name and the open (left) parenthesis that encloses the array indices. For example,

`gasdef (1,1) = . . .` is correct, whereas

`gasdef (1,1) = . . .` will lead to input processing errors.

Optional comments can appear between input NAMELIST group records. They can also be placed within a NAMELIST group. A comment within the record must be preceded by a semicolon. No input data can be specified after a comment on the same line, i. e., entries after a semicolon on the same line will be ignored. (See sample input decks in Sec. XII.) An input NAMELIST group record may contain only comments or may be entirely blank.

III. GEOMETRY DEFINITION

A. Cell Labeling Convention

In HMS two coordinate systems are available. In the Cartesian or rectangular system, the coordinate axes are x , y , and z , and their corresponding logical indices are i , j , and k . If the cylindrical system is used, then the logical coordinate indices i , j , and k correspond, respectively, to the radial (r), azimuthal (θ), and axial (z) directions. To define regions in the computational domain where initial and boundary conditions are to be applied, the user must understand the cell numbering scheme. The same scheme is used as the basis for specifying regions (lines, surfaces, or volumes) where graphical displays of the calculated results are desired.

The finite-difference mesh used for discretizing the geometry consists of computational cells that are ordered logically in three dimensions with indices i , j , and k . The maximum number of cells in each direction is designated $imax$, $jmax$, or $kmax$, depending on the direction. In HMS, a layer of fictitious cells is used just beyond each boundary of the computational domain to accommodate general boundary condition treatments. Therefore, in the z -direction, for example, $k = 1$, $kmax$ indicate the fictitious boundary cells while only cells with k indices from 2 to $kmax-1$ are active or real. So the total number of real cells in the entire mesh is the product $(imax-2)(jmax-2)(kmax-2)$.

Besides labeling cells, it is useful sometimes to refer to the cell faces between them, which form the grid lines. The HMS convention is that the i^{th} grid line refers to the cell face between a cell with index i and the next cell with index $i+1$. Figure 2

illustrates the numbering scheme with a 1D example using $i_{\max} = 7$. Similarly, this cell-face numbering convention applies to the other two (j and k) dimensions.

Fig. 2. HMS numbering convention for cells (or cell centers) and cell faces using the i -direction with $i_{\max} = 7$ for illustration. The fictitious boundary cells are shaded, i. e., cell numbers 1 and 7. The real fluid cells have numbered from 2 to 6. The physical computational volume ranges from cell-face number 1 to cell-face number 6.

B. Mesh Generation

Before generating a mesh, the user must specify which coordinate system is to be used for the computation. The input variable for this is `cy1` in the NAMELIST group `xput`. Set `cy1 = 0` (default) to use Cartesian coordinates or `cy1 = 1` to use cylindrical coordinates. (Note that if `cy1 = 1` and the full 360° is modeled, the user must specify the periodic condition at the azimuthal boundaries; see p. 27.) Then the computational mesh is defined by one of two methods available. Input variables for both methods are in NAMELIST group `meshgn`. Note that the user only defines geometry for the *real*, physical domain. Fictitious boundary cells are assigned automatically by the code.

1. Direct Input of Grid Locations

The first method of defining the mesh is simply direct entering of the coordinate value of each grid point in each direction. The input array variables `xgrid`, `ygrid`, and `zgrid` are used to specify grid point locations in the x -, y -, and z -directions in Cartesian coordinates. The length unit must be in centimeters. For example,

```
xgrid = 0., 1., 2., 3., 4., 5., 6., 7., 8., 10.
```

specifies that the mesh in the x -direction goes from 0 to 10 cm and has nine cells. The first eight cells have a cell-width of 1 cm, and the last one is 2 cm wide. Note that `xgrid`, `ygrid`, and `zgrid` values define the coordinates of cell faces.

If cylindrical coordinates are used, then `xgrid` refers to grid point locations in the radial (r) direction, and `ygrid` and `zgrid` refer respectively to the azimuthal (θ) and axial (z) directions. The measure of θ should be in degrees. For example,

```
ygrid = 0., 15., 30., 45., 60., 75., 90.
```

specifies a mesh that is a quadrant of a cylinder and has six layers of cells in the azimuthal direction, all evenly spaced at 15° .

2. Automatic Mesh Generation

The above method of directly entering grid coordinates is useful when such information is available, for example, from a separate mesh-generation program. In many cases, it is more convenient to use the second method offered by the code, which uses an automatic mesh generator. This method allows easy generation of a mesh composed of cells with either fixed or variable sizes. The basic idea is to build a mesh by stacking together a series of submeshes in each coordinate direction. For example, consider the x -direction. The x -dimension of the problem to be solved is subdivided into a set of nkx intervals. The k^{th} interval extends from its left (lower) end, $x_l(k)$, to the left end of the next interval, $x_l(k+1)$. Within each interval there is a location, $x_c(k)$, where the mesh cells will be smallest. In other words, the grid lines in the k^{th} submesh converge to location $x_c(k)$. The number of cells between $x_l(k)$ and $x_c(k)$ is specified as $nxl(k)$, and the number from $x_c(k)$ to $x_l(k+1)$ is specified as $nxr(k)$. The minimum cell size, which is located at $x_c(k)$, is specified as $dxmn(k)$.

Using the above information, the mesh generator expands cell sizes from a value of $dxmn(k)$ at $x_c(k)$ in a quadratic manner such that the required number of cells will lie on each side of $x_c(k)$ and fill the subinterval. If $dxmn(k)$ is larger than the cell size corresponding to uniform zoning, then the generator will produce a uniformly spaced mesh in the x -direction.

The number of cells defined on either side of $x_c(k)$ can be any value, including zero. A choice of zero is often useful when the minimum cell size is desired at the beginning or end of a subinterval. This is often done in problems in which it is required to have fine mesh resolution in the vicinity of a surface where steep gradients in the temperature or velocity profile are expected.

HMS supports definition of up to 49 mesh subintervals in each of the three dimensions. Thus, it is possible to generate complicated meshes with locally fine resolution around any number of points. Furthermore, because the minimum cell sizes are specified as part of the input data, there should be no unexpected cell-size related numerical stability difficulties.

In summary, the input parameters for mesh subdivision k in the x -direction are as follows:

nkx defines the total number of subintervals in the x -direction.
 $x1(k)$ sets the location of the left boundary of subdivision k .
 $xc(k)$ sets the "convergence point" where the minimum cell spacing occurs in subdivision k .
 $nx1(k)$ specifies the number of cells to the left of $xc(k)$, i. e., between locations $x1(k)$ and $xc(k)$ in subdivision k .
 $nxr(k)$ specifies the number of cells to the right of $xc(k)$, i. e., between locations $xc(k)$ and $x1(k+1)$ in subdivision k .
 $dxmn(k)$ specifies the minimum cell size in the x -direction in subdivision k .

A similar treatment is used in the y - and z -directions. In addition, the input variables used for the r - and θ -directions, when cylindrical coordinates are chosen, are the same as those for the x - and y -directions, respectively. A list of the variables for all the directions is given below:

x - or r -direction: $nkx, x1(k), xc(k), nx1(k), nxr(k), dxmn(k)$
 y - or θ -direction: $nky, y1(k), yc(k), ny1(k), nyr(k), dymn(k)$
 z -direction: $nkz, z1(k), zc(k), nz1(k), nzr(k), dzmn(k)$

Consider the following two examples that illustrate the use of the automatic mesh generator.

Cartesian Mesh. The first example involves Cartesian geometry ($cyl = 0$). Here we show how to generate a uniform mesh in the z -direction extending from 0 to 12 cm containing 10 cells. In the x -direction, the mesh also extends from 0 to 12 cm and consists of 10 cells, but has a minimum cell size of 0.2 cm on both sides of the line $x = 5$ cm. The following input specifications in NAMELIST group meshgn will generate such a mesh, as depicted in Fig. 3.

```

nkx = 1, nkz = 1,
z1(1)=0., zc(1)=0., nz1(1)=0, nzr(1)=10, dzmn(1)=1.e9, z1(2)=12.,
x1(1)=0., xc(1)=5., nx1(1)=5, nxr(1)=5, dxmn(1)=0.2, x1(2)=12.

```

Cylindrical Mesh. In the second example, the coordinate system chosen is cylindrical ($cyl = 1$ specified in NAMELIST group xput). Figure 4 shows the mesh in two dimensions generated by the following input in NAMELIST group meshgn:

```

nkx = 1, nky = 1,
x1(1)=0., xc(1)=15., nx1(1)=10, nxr(1)=0, dxmn(1)=0.5, x1(2)=15.,
y1(1)=0., yc(1)=0., ny1(1)=0, nyr(1)=24, dymn(1)=1.e9, y1(2)=360.

```

In the azimuthal (θ) direction, there are 24 cells, which are evenly spaced because the minimum cell size, $dymn$, is greater than the average cell width obtained by uniform zoning (i.e., $10^9 > 360/24$). In the radial (r) direction, there are 10 cells that discretize

Fig. 3. Two-dimensional x - z view of a mesh generated by the HMS automatic mesh generator. Note that the vertical mesh lines converge toward the $x = 5$ line.

the total radius of 15 cm. The minimum cell size, dx_{mn} , is specified as 0.5 cm, which is smaller than the "uniform" cell width given by $15 \text{ cm}/10$. Therefore, the cell size gradually expands from this minimum value at $r = 10 \text{ cm}$ to $r = 0$.

In many problems, it is useful to know the largest, as well as the smallest, cell sizes to have a feel for the computational length scales as compared to the physical scales. Note that the maximum cell size, $\delta_{\max}$ in any submesh generated by this automatic mesh generator can be easily determined from the relation

Fig. 4. Two-dimensional, r - θ view of a mesh generated by the HMS automatic mesh generator. The cell spacing becomes finer as r increases, but is uniform in the θ -direction.

$$(\delta_{\min} + \delta_{\max})/2 = \delta_{\text{avg}} ,$$

or

$$\delta_{\max} = 2 \delta_{\text{avg}} - \delta_{\min} ,$$

where δ_{avg} is the average cell size corresponding to uniform zoning. Therefore, $\delta_{\max}$ in the x -direction in the mesh generated in the above example is

$$2 \times [(5 - 0)/5] - 0.2 = 1.8 \text{ cm}$$

on the left side of $x = 5$ cm. On the right side, the maximum cell size is

$$2 \times [(12 - 5)/5] - 0.2 = 2.6 \text{ cm} .$$

C. Definition of Walls and Obstacles

The previous section describes how to generate a computational mesh that represents a discretized model of the region over which the conservation equations for the fluid are solved. In most practical problems, the fluid flow region is more complex than an empty rectangular box. There may be flow obstacles or interconnected subcompartments. In HMS, walls and obstacles can be defined within the mesh to model complex flow paths. The nomenclature used by the code is that a *wall* is a surface dividing two adjacent layers of fluid cells that forbids flow across it. An obstacle is a volume consisting of an arbitrary number of cells, namely *obstacle cells*, through which no fluid flow is allowed. In other words, obstacle cells are blocked out from the fluid-dynamics calculations. (However, in problems involving heat transfer, conduction inside the obstacle cells is calculated.)

1. Walls

To define a wall means specifying a surface normal to any of the three orthogonal dimensions with logical indices *i*, *j*, and *k*. This is done via the input array variable `walls` in the NAMELIST group `xput`. The array `walls` is 2D with the second index identifying the wall definition, and the first index specifying eight numbers that are required to define the wall surface:

<code>walls(1,*)</code>	Beginning <i>i</i> mesh index (cell-face number).
<code>walls(2,*)</code>	Ending <i>i</i> mesh index (cell-face number).
<code>walls(3,*)</code>	Beginning <i>j</i> mesh index (cell-face number).
<code>walls(4,*)</code>	Ending <i>j</i> mesh index (cell-face number).
<code>walls(5,*)</code>	Beginning <i>k</i> mesh index (cell-face number).
<code>walls(6,*)</code>	Ending <i>k</i> mesh index (cell-face number).
<code>walls(7,*)</code>	Block number (must be set to 1).
<code>walls(8,*)</code>	Integer to identify the type of wall (thickness and material), used only for heat transfer only. Ignored if heat transfer is not invoked.

(The asterisk `*` should be replaced by an integer that identifies the particular wall definition.)

The last element `walls(8,*)` is reserved for specifying an input that is only required if heat transfer is invoked (by setting `ihtflag = 1` in NAMELIST group `rheat`), but otherwise ignored. This entry is explained in Sec. VI. The variable `walls(7,*)` is reserved for use in GASFLOW, that has *multiblock* capability, to identify the block number. Because HMS does not support multiblock, this variable must be set to 1. The rest of the input, `walls(1,*)` to `walls(6,*)`, specify the location and extent of the wall. Because a surface has only two dimensions, one of the three pairs of beginning and ending mesh indices must be the same. Consider the following input which defines two walls:

```
walls = 2, 2, 1, 2, 2, 10, 1, 0,
        2, 9, 1, 2, 2, 2, 1, 0
```

Because the input data are read in consecutively in the order of memory storage, no indices have to be explicitly written for the two-dimensional array `walls`. (In other words, the first input line above defines values for `walls(1,1)`, `walls(2,1)`, etc., up to `walls(8,1)`.) The 16 numbers will be used correctly by the code to define two walls. An equivalent way to write the above input is

```
walls(1,2) = 2, 9, 1, 2, 2, 2, 1, 0
```

In this example, the first line defines a wall at the *i* cell-face index 2, and extends from *j*-index 1 to *j*-index 2, and from *k*-index 2 to *k*-index 10. The second line defines a wall that has a normal in the *k*-direction, or perpendicular to the first wall. If the input is applied to the mesh shown in Fig. 4, then the two walls will appear on the *x-z* plane of the mesh as shown in Fig. 5.

2. Obstacles

Figure 5 also shows obstacle cells that further restrict fluid flow in the computational domain. These "mesh obstacles" are specified by the input array variable `mobs` in NAMELIST group `xput`:

<code>mobs(1,*)</code>	Beginning <i>i</i> mesh index (cell-face number).
<code>mobs(2,*)</code>	Ending <i>i</i> mesh index (cell-face number).
<code>mobs(3,*)</code>	Beginning <i>j</i> mesh index (cell-face number).
<code>mobs(4,*)</code>	Ending <i>j</i> mesh index (cell-face number).
<code>mobs(5,*)</code>	Beginning <i>k</i> mesh index (cell-face number).
<code>mobs(6,*)</code>	Ending <i>k</i> mesh index (cell-face number).
<code>mobs(7,*)</code>	Block number (must be set to 1).
<code>mobs(8,*)</code>	Integer to identify the material that the solid obstacle is made of, used only for heat transfer. Ignored if heat transfer is not invoked.

The elements in the array `mobs` have the same meaning as those in `walls`, except for the last element, which is explained in Sec. VI, where solid heat conduction is discussed. However, mesh obstacles refer to a *volume* region where no flow is allowed to penetrate. Therefore, the beginning and ending *i*, *j*, and *k* mesh indices should define any two vertices of a three-dimensional volume that are diagonal to each other. The following two `mobs` definitions specify obstacle regions that are

Fig. 5. Two-dimensional x - z view of a mesh containing two wall surfaces and two obstacle regions generated by `walls` and `mobs` input definitions, respectively.

shown on Fig. 5:

```
mobs = 3, 7, 1, 2, 9, 10, 1, 0,
        6, 9, 1, 2, 7, 9, 1, 0
```

HMS supports 500 definitions each for `walls` and `mobs`.

D. Checking Geometric Model

Once the mesh has been generated and any walls and obstacles have been defined, the geometry of the computational domain is completely specified. The user may then specify the constituents of the gas mixture to be calculated, impose appropriate initial and boundary conditions, turn on various desired models, and

specify any parameters with regard to running the calculation. However, when setting up a new problem, especially one with a complex geometry, it is often helpful to review the mesh before the actual, desired computation is carried out. Knowing the mesh indices at all computational boundaries and where walls and obstacles are will help minimize errors in defining initial and boundary conditions. This will also make it easy to specify graphical output of the solution at regions of interest, so the calculation can be monitored right from the beginning.

After the input geometry and mesh definition have been read in and processed, HMS writes a file called meshmap that contains a list of all computational cells. Information is given for each cell on its *i*, *j*, and *k* index values, as well as a single index that the code uses for storage in memory (called "master" index, *m*), and the nature of the cell. The master cell index is related to the logical indices as follows:

$$m = (k-1)*imax*jmax + (j-1)*imax + i$$

In other words, *m* lists all cells consecutively, going over the *i*-index first, followed by *j*, and then *k*. The fictitious cells beyond the physical domain boundaries are termed *boundary* cells, while a cell within the domain is either a *fluid* or an *obstacle* cell, depending on whether it has been blocked out with a mobs definition. Also shown in the file meshmap is whether a fluid cell is open to flow in each of the three directions and the *m*-index of its neighboring cells in all directions. A section extracted from a meshmap file is shown below:

M	K	J	I	Cell Type	MFLAG	Velocity Comps.	----- -J	Neighbors +J	----- -K	----- +K	M
2196	8	1	12	B	24	none	2184	2208	1884	2508	2196
2197	8	2	1	B	24	none	2185	2209	1885	2509	2197
2198	8	2	2	F	6	vw	2186	2210	1886	2510	2198
2199	8	2	3	F	7	uvw	2187	2211	1887	2511	2199
2200	8	2	4	F	7	uvw	2188	2212	1888	2512	2200
2201	8	2	5	F	7	uvw	2189	2213	1889	2513	2201
2202	8	2	6	F	6	vw	2190	2214	1890	2514	2202
2203	8	2	7	O	8	none	2191	2215	1891	2515	2203
2204	8	2	8	O	8	none	2192	2216	1892	2516	2204
2205	8	2	9	O	8	none	2193	2217	1893	2517	2205
2206	8	2	10	F	7	uvw	2194	2218	1894	2518	2206
2207	8	2	11	F	6	vw	2195	2219	1895	2519	2207
2208	8	2	12	B	24	none	2196	2220	1896	2520	2208
2209	8	3	1	B	24	none	2197	2221	1897	2521	2209
2210	8	3	2	F	7	uvw	2198	2222	1898	2522	2210
2211	8	3	3	F	7	uvw	2199	2223	1899	2523	2211
2212	8	3	4	F	7	uvw	2200	2224	1900	2524	2212

Under the Velocity Comps. column is information on whether the cell has a velocity component (i. e., whether there can be flow) across the positive cell face in each of the three directions. The cell with a *m*-index of 2202, for example, is a fluid cell (Cell Type = F), and has velocity components across the positive *j*- and *k*-

faces (v and w , respectively) but not across the positive i -face (the u -component is not printed), because the next cell in that direction is an obstacle cell (Cell Type for cell 2203 is O). Naturally, all obstacle cells have no velocity components across them. For a fictitious boundary cell (Cell Type = B), there may or may not be flow across any of its faces, depending on the boundary conditions specified there. (In HMS, there are only three types of cells, F, B, and O.) The four columns under Neighbors give the m -indices of adjacent cells in the positive and negative j - and k -directions. Neighbors in the i -direction for an internal cell m have master indices $m-1$ and $m+1$. Under the MFLAG column is a number that is used by the code (in binary format) to describe the cell's status, i.e., whether it is open to flow and whether the associated surfaces, if any, have been specified as *no-slip* or *free-slip*, etc. This information, however, is primarily intended for code developers or advanced users who work with the code at the debugger level.

More useful to general code users are graphical displays of the mesh, rather than the tabular listing of each cell as given in the file meshmap. In HMS, five plots of the mesh are generated automatically before the calculation begins. These include 2D slices of the mesh in the i - j , j - k , and i - k planes, as well as 3D perspective views of the mesh from the inside and outside of the computational domain. However, these plots only show the discretization and dimensions of the mesh. Any walls or obstacle cells that may have been defined are not shown on these default plots. The code provides an option for plotting the mesh (and any embedded walls and obstacles) along planes cutting across all cell faces and cell centers. This is done via the variable maxcyc in NAMELIST group xput. This input variable normally sets the maximum number of computational cycles or time steps up to which the calculation is allowed to run. However, if maxcyc is set to a negative value, the code will not perform any physical calculation but will instead produce a set of plots that show 2D slices of the mesh. How many plots are generated or which views of the mesh are shown depends on the value of maxcyc, according to the keys listed in Table II.

As shown in Table II, it is possible to plot six sets of 2D grid plots—one for each i , j , or k cell-face location, and one for each i , j , or k cell-center location. The set of i cell-face plots, for example, are j - k surfaces ($1 \leq j \leq j_{\max}-1$, $1 \leq k \leq k_{\max}-1$) drawn at every i mesh line. The sets of k cell-center plots, for example, are i - j surfaces ($1 \leq i \leq i_{\max}-1$, $1 \leq j \leq j_{\max}-1$) drawn at successive k -direction cell-center locations. Depending on the value of maxcyc, one or more sets can be plotted. An entry of 1 in the table means the set is plotted while an entry of zero means that the set is not plotted. For example, if only cell-center plots of the mesh on all j -planes are wanted, then maxcyc = -16 should be specified. Cell-center plots are often more useful because they show the exact locations of obstacles. At interfaces between fluid and obstacle cells, the cell-face plots would be ambiguous.

Table II. Specification of 2D Grid Plots at Cell-Face and/or Cell-Center Locations via Input Variable maxcyc in NAMELIST Group xput

maxcyc	i Face	j Face	k Face	i Center	j Center	k Center
-1	1	0	0	0	0	0
-2	0	1	0	0	0	0
-3	1	1	0	0	0	0
-4	0	0	1	0	0	0
-5	1	0	1	0	0	0
-6	0	1	1	0	0	0
-7	1	1	1	0	0	0
-8	0	0	0	1	0	0
-9	1	0	0	1	0	0
-10	0	1	0	1	0	0
-11	1	1	0	1	0	0
-12	0	0	1	1	0	0
-13	1	0	1	1	0	0
-14	0	1	1	1	0	0
-15	1	1	1	1	0	0
-16	0	0	0	0	1	0
-17	1	0	0	0	1	0
-18	0	1	0	0	1	0
-19	1	1	0	0	1	0
-20	0	0	1	0	1	0
-21	1	0	1	0	1	0
-22	0	1	1	0	1	0
-23	1	1	1	0	1	0
-24	0	0	0	1	1	0
-25	1	0	0	1	1	0
-26	0	1	0	1	1	0
-27	1	1	0	1	1	0
-28	0	0	1	1	1	0
-29	1	0	1	1	1	0
-30	0	1	1	1	1	0
-31	1	1	1	1	1	0
-32	0	0	0	0	0	1
-33	1	0	0	0	0	1
-34	0	1	0	0	0	1
-35	1	1	0	0	0	1
-36	0	0	1	0	0	1
-37	1	0	1	0	0	1
-38	0	1	1	0	0	1
-39	1	1	1	0	0	1
-40	0	0	0	1	0	1
-41	1	0	0	1	0	1
-42	0	1	0	1	0	1
-43	1	1	0	1	0	1
-44	0	0	1	1	0	1
-45	1	0	1	1	0	1
-46	0	1	1	1	0	1
-47	1	1	1	1	0	1
-48	0	0	0	0	1	1
-49	1	0	0	0	1	1
-50	0	1	0	0	1	1
-51	1	1	0	0	1	1
-52	0	0	1	0	1	1
-53	1	0	1	0	1	1
-54	0	1	1	0	1	1
-55	1	1	1	0	1	1
-56	0	0	0	1	1	1
-57	1	0	0	1	1	1
-58	0	1	0	1	1	1
-59	1	1	0	1	1	1
-60	0	0	1	1	1	1
-61	1	0	1	1	1	1
-62	0	1	1	1	1	1
-63	1	1	1	1	1	1

IV. SPECIFICATION OF GAS SPECIES AND PROPERTIES

A. Definition of Gas Species

In HMS, the basic thermodynamic properties of all gas species are assumed to be governed by the ideal gas law. The ideal gas equation of state also applies to a multicomponent gas mixture. In other words, Dalton's Law of partial pressures is assumed to be valid. Therefore, within any volume V , we have the following relation for the gas mixture (or for each component):

$$pV = nRT \quad ,$$

where p is the pressure of the mixture (or partial pressure of a gas component), n is the total number of gram-moles (or number of moles of a gas component), R is the universal gas constant equal to 8.3144 ergs/mole-K, and T is the absolute temperature of the gas mixture. The above relation can also be written in terms of the mass density, ρ , which is given by nM/V , where M is the molecular weight:

$$pM = \rho RT \quad .$$

Therefore, the molecular weight alone is sufficient to define the pressure-density relationship of a gas species.

HMS solves the energy conservation equations in terms of the specific internal energy, i , which is related to the absolute temperature, for an ideal gas, by

$$i = \int_{T_{ref}}^T C_v dT' \quad ,$$

where C_v is the specific heat capacity at constant volume having units of ergs/g-K and T_{ref} is a reference temperature. In general, C_v is a function of temperature and one can approximate this function by polynomials of various degrees depending on the accuracy required. In HMS, the approximation is made that C_v is a constant for each gas species that is independent of temperature, and T_{ref} is chosen to be zero. Such an approximation allows us to write that

$$i = C_v T \quad .$$

Therefore, the internal energy of each gas species is directly proportional to the absolute temperature through a constant C_v . Hence, the thermodynamic behavior (pressure-density-temperature and energy-temperature relations) of each gas species is fully specified through two numbers representing M and C_v .

The built-in gas component library in HMS has 20 species with properties given in Table III. HMS does not support user-defined gas species. The user must choose from this library the gas species to be calculated.

Table III. Properties of Gas Species Available in HMS

Gas Species Common Name	Species Symbol Used in HMS	Molecular Weight <i>M</i> (g/mole)	Specific Heat Capacity <i>C_v</i> (10 ⁷ ergs/g-K)
Carbon atoms (soot)	c	12.0110	0.7099
Carbon monoxide	co	28.0110	0.7432
Carbon dioxide	co2	44.0110	0.6544
Hydrogen	h2	2.0160	0.0849
Water vapor	h2o	18.0160	1.4108
Nitrogen	n2	28.0130	0.7448
Nitrous oxide	n2o	44.0130	0.6912
Oxygen	o2	32.0000	0.6618
Air	air	28.9700	0.7168
Argon	ar	39.9480	0.3107
Helium	he	4.0030	3.1140
Ammonia	nh3	17.0300	1.5344
Methane	ch3	16.0430	1.7048
Hydroxyl radicals	ho	17.0080	1.2742
Hydrogen atoms	h	1.0080	12.3727
Hydrogen dioxide	ho2	33.0080	0.8050
Nitric oxide	no	30.0060	0.7175
Oxygen atoms	o	16.0000	0.8498
N-H radicals	nh	15.0140	1.3874
H-N-O radicals	hno	31.0140	0.8488

The input array variable `mat` in NAMELIST group `xput` is used to define the gas species in a calculation. Any one or all of the species listed in Table III can be chosen. For example, in a problem involving air, steam, and hydrogen, the input will be

```
mat = 'air', 'h2o', 'h2'
```

where the character string within each pair of quotes represents the symbol for the corresponding species as given in Table III. The order in which the gas names are listed in the definition of `mat` is arbitrary. However, this order determines the gas component number that identifies each species involved. Therefore, in this example, air is component 1, water vapor is component 2, and hydrogen is component 3 in the

gas mixture. These identification numbers will be used in subsequent input specifications where reference to particular components of the mixture is required.

In the example above, we treat air in the gas mixture as a single species, specified as 'air'. In reality, air is itself a mixture consisting of nitrogen, oxygen, and trace amounts of carbon dioxide and inert gases. However, modeling air as a single species simplifies the input specification and analysis of the calculated results a great deal, and reduces the computational time required. The user should follow this approach whenever possible. In problems where nitrogen, oxygen, carbon dioxide, etc., have to be calculated explicitly, such as combustion of hydrogen in oxygen and in nitrous oxide, then air should be specified as consisting of the individual gases at appropriate concentrations. At sea level, the composition of dry air by volume is approximately 78.2% N₂, 20.9% O₂, 0.9% Ar, and 0.03% CO₂. Here we only discuss identifying of the gases to be involved in the calculation. The concentration, in mole or volume fraction, of each gas component will be specified with the variable *gasdef*, which is discussed in the section on initial and boundary conditions (Sec. V).

Some of the species listed in Table III are not stable molecules. They are included in the gas library because the code (GASFLOW) has been used, and can be used, to study the interaction between the fluid dynamics and *detailed* chemical kinetics of turbulent flame propagation and acceleration. Detailed kinetics of even the "simple" H₂-O₂-H₂O chemical system involves about 50 reaction steps in which many intermediate reaction products are produced and destroyed. However, because it is intended primarily for practical problems, HMS uses one global reaction to model the entire hydrogen combustion kinetics.

B. Definition of Transport Properties

In this section, we discuss how to specify the *physical* transport properties for the gas mixture. These properties determine the rates at which mass, energy, and momentum are transported within the gas by the action of molecular diffusion. (Other mechanisms for mass, energy, and momentum transport include advection and turbulent mixing, both of which depend on the local, instantaneous velocity of the fluid.) In HMS, the diffusion process is modeled by the Fick's Law, which states that the diffusive flux is proportional to some gradient quantity that represents a driving potential. The proportionality constant is called the diffusion coefficient. In momentum transfer, the gradient is in the velocity vector, and the diffusion coefficient is the kinematic viscosity, ν . In mass diffusion, the gradient of species density is used, and the diffusion coefficient is called the mass diffusivity, D . For the diffusion of heat, the heat flux is proportional to the product of the temperature gradient and the thermal diffusivity, α . These diffusivities, in general, depend on temperature, mixture composition, and (to a lesser extent) pressure.

HMS assumes that all of the three diffusion coefficients are constant throughout the calculation. Furthermore, in the case of mass diffusion in a mixture consisting of three or more components, the same mass diffusivity is used for modeling the diffusion of every species into the mixture. (A more rigorous treatment would be to have a separate diffusivity for modeling the diffusion of each mass species into the mixture. For a binary mixture, equimolar counterdiffusion normally applies, hence a single mass diffusivity is adequate.) Again, this approximation of constant diffusivities has been taken in the current code because it is mainly for practical applications. In many situations, the detail dependence of these molecular transport properties on temperature and composition is either not known or the additional accuracy due to detail modeling does not justify the increase in complexity of the calculation. Furthermore, in many practical problems, the fluid flow is turbulent, so the turbulent diffusive fluxes dominate their molecular counterparts.

Therefore, input specification of ν , D , and α will require three numbers. For the kinematic viscosity ν , the input variable `nu` is used, which has units of cm^2/s . For the mass and thermal diffusivities, we use respectively the nondimensional quantities Sc and Pr (Schmidt and Prandtl numbers) to define them:

$$Sc \equiv \nu/D$$

$$Pr \equiv \nu/\alpha.$$

The Schmidt and Prandtl numbers are represented by the input variables `schmidt` and `prandtl`. All of these variables are in NAMELIST group `xput`. For example, an input line that reads

```
nu = 0.2, prandtl = 0.7, schmidt = 0.4
```

specifies that the kinematic viscosity is $0.2 \text{ cm}^2/\text{s}$, the thermal diffusivity is $0.286 \text{ cm}^2/\text{s}$, and the mass diffusivity is $0.5 \text{ cm}^2/\text{s}$. The default value for `nu` is $0.15 \text{ cm}^2/\text{s}$, while those for `prandtl` and `schmidt` are both 1 (i. e., $\alpha = D = \nu = 0.15 \text{ cm}^2/\text{s}$).

Note that the above input is only required or used if the model options requesting calculation of diffusion of momentum, mass, and energy are turned on. These options will be discussed in Sec. VII.B.

V. INITIAL AND BOUNDARY CONDITIONS

HMS solves the Navier-Stokes equations of motion and the energy conservation equations for a fluid in a specified computational domain. The governing equations are time dependent, partial differential equations. To complete the mathematical formulation, we must specify initial and boundary conditions. In problems where heat conduction in solid structures is calculated, the initial and

boundary solid temperatures also have to be specified. In this section, we discuss how to define initial and boundary conditions for the *fluid*. Those for the solid thermal structures will be discussed in the next section.

A. Specification of Initial Conditions

1. Fluid Composition and State

Except for a restart run (discussed in Sec. IX.D), the user must define the pressure, temperature, and composition of the fluid everywhere in the computational domain at the beginning of the calculation. This can be accomplished via the input array variable `gasdef` in NAMELIST group `xput`. Although initial conditions are defined with `gasdef`, the input variable has more general use. For example, the user must define, with `gasdef`, the fluid condition for all fictitious boundary cells that are expected to exchange fluid with adjacent physical cells. (More on that later when we discuss boundary conditions.) The variable is a 2D array. The second index identifies the particular "gas definition." For each `gasdef` specification, a minimum of 14 numbers is required, which are input through the elements of the first array dimension with the following meaning:

<code>gasdef (1, *)</code>	Beginning i mesh index (cell-face number).
<code>gasdef (2, *)</code>	Ending i mesh index (cell-face number).
<code>gasdef (3, *)</code>	Beginning j mesh index (cell-face number).
<code>gasdef (4, *)</code>	Ending j mesh index (cell-face number).
<code>gasdef (5, *)</code>	Beginning k mesh index (cell-face number).
<code>gasdef (6, *)</code>	Ending k mesh index (cell-face number).
<code>gasdef (7, *)</code>	Block number (must be set to 1).
<code>gasdef (8, *)</code>	Pressure (dynes/cm ²) in defined volume.
<code>gasdef (9, *)</code>	Temperature (K) in defined volume.
<code>gasdef (10, *)</code>	Option flag for specification of gas composition: 1 for mass fraction, 2 for volume fraction.
<code>gasdef (11, *)</code>	Time (s) at which "gas definition" begins.
<code>gasdef (12, *)</code>	Time (s) at which "gas definition" ends.
<code>gasdef (13, *)</code>	Gas species component number (determined by the order in the gas species list defined by <code>mat</code>). Gas species component can alternatively be specified by its symbol as given in Table III, e. g., 'h2'.
<code>gasdef (14, *)</code>	Mass or volume fraction of above gas species in defined volume.
<code>gasdef (15, *)</code>	Second gas species component number, if needed.
<code>gasdef (16, *)</code>	Mass or volume fraction of second gas species in defined volume, if needed.

From the above, we can see that `gasdef` defines the pressure, temperature, and composition of a specified fluid region. These conditions are imposed on the fluid volume over a specified range of time. Variables `gasdef (15, *)` and beyond are

only necessary if the user wants to define a fluid region of multiple gas species. Compositions of up to 20 gas species may be defined. At least one gas species must be defined, and the sum of all mass or volume fractions defined in each `gasdef` specification must be 1.

Note that at least one definition of `gasdef` is required to fully specify the initial fluid conditions. A common use of `gasdef` is to first specify initial conditions globally, then override them with following definitions for local conditions.

For specification of initial conditions, the beginning and end time should both be set to 0. Consider the following input:

```
mat = 'h2o', 'air',
gasdef(1,1) = 1, 6, 1, 6, 1, 6, 1, 1.013e6, 298.,
              2, 0., 0., 1, 0.1, 2, 0.9
```

which can also be input in the form:

```
mat = 'h2o', 'air',
gasdef(1,1) = 1, 6, 1, 6, 1, 6, 1, 1.013e6, 298.,
              2, 0., 0., 'air', 0.9, 'h2o', 0.1
```

If the coordinate system chosen is Cartesian, and the mesh in each of the three directions is the same as that shown on Fig. 2 (i. e., $i_{max} = j_{max} = k_{max} = 7$), then the above `gasdef` input specifies the initial condition of the fluid throughout the entire physical domain. The fluid is initially composed of 10% water vapor and 90% air by volume at room temperature and atmospheric pressure.

2. Fluid Velocities

For initial fluid velocities, default is that the fluid is initially at rest everywhere in the mesh. However, the user can change the default by setting a constant value for each component of the velocity vector. Then the code will set the initial fluid velocity everywhere in the mesh according to the specified component values. The input variables for defining initial velocity components, in NAMELIST group `xput`, are

<code>ui</code>	Initial fluid velocity in <i>i</i> - (<i>x</i> - or <i>r</i> -) direction, cm/s.
<code>vi</code>	Initial fluid velocity in <i>j</i> - (<i>y</i> - or θ -) direction, cm/s.
<code>wi</code>	Initial fluid velocity in <i>k</i> - (<i>z</i> -) direction, cm/s.

B. Specification of Boundary Conditions

HMS offers two methods, one "global" and the other "local," for specifying boundary conditions. These two methods will be described separately.

1. Global Definition

The first method applies boundary conditions on entire boundaries of the computational domain. There are six surfaces that bound the 3D mesh discretized by logical indices (i, j, k) . Consequently, the boundary condition on each of these surfaces can be specified through any one of the following variables. These and all input other variables discussed subsequently are in the NAMELIST group `xput`.

<code>ibw</code>	Boundary condition type indicator for the $+i$ (west) boundary.
<code>ibe</code>	Boundary condition type indicator for the $-i$ (east) boundary.
<code>ibs</code>	Boundary condition type indicator for the $+j$ (south) boundary.
<code>ibn</code>	Boundary condition type indicator for the $-j$ (north) boundary.
<code>ibb</code>	Boundary condition type indicator for the $+k$ (bottom) boundary.
<code>ibt</code>	Boundary condition type indicator for the $-k$ (top) boundary.

The boundary conditions on these six boundaries can be specified according to the following key:

Type	Boundary Condition
1	Rigid free-slip wall
2	Rigid no-slip wall
3	Continuative
4	Periodic
5	Specified pressure

Rigid Free-Slip. The default boundary condition is Type 1. Therefore, if no boundary conditions are specified, the code will assume that the entire computational volume is enclosed within rigid, impenetrable walls at which there is free slip, or the gradient of the tangential velocity components is zero. This is the most common boundary condition used. In many practical problems, a large portion of the computational boundaries are solid surfaces (for example, the walls of a room or a containment building), and the mesh resolution is not fine enough to represent the near-wall velocity gradients so the free-slip condition is the best approximation there. This is also the boundary condition at the $-i$ boundary, or at $r = 0$, if cylindrical coordinates are used. This is not a poor numerical representation at the $r = 0$ boundary because that surface has a zero area, hence there is no severe flow limitation caused by the free-slip wall condition at the centerline.

Rigid No-Slip. The no-slip condition, Type 2, is another option with which the user can define a boundary as an impenetrable surface. No slip means that the fluid "sticks" to the solid wall and all velocity components are zero there. (In the current version, the code does not support moving wall boundary conditions.) This boundary condition is used in problems where the velocity gradients near solid surfaces are important, and the mesh is sufficiently fine to resolve them. For example, if the classical Hagen-Poiseuille flow, i. e., laminar flow through a circular

pipe, is to be simulated, then the no-slip condition must be applied at the pipe wall to be able to calculate a parabolic velocity profile in the steady solution.

Continuative. Type 3 is the continuative boundary condition. This condition is usually applied at outflow boundaries, where the fluid is to flow smoothly out of the mesh, causing minimum upstream effects. With this boundary condition, the gradients of pressure, internal energy, density, velocity, etc., across the specified boundary are set to zero.

Periodic. The periodic boundary condition, Type 4, specifies that the fluid conditions at the beginning and ending boundaries in a particular direction are identical. Periodic boundaries must be specified in pairs, i. e., both the + and - boundaries must be specified as periodic. This condition is most commonly used for defining the θ -boundaries when the mesh covers the full 360° in the azimuthal direction. (Note that the default boundary condition is Type 1, or a rigid free-slip wall.) Therefore, if cylindrical coordinates are chosen and the mesh is defined to extend from 0° to 360° , then the following input must be used to specify the appropriate boundary condition at the $-\theta$ ($-j$) and $+\theta$ ($+j$) boundaries:

```
ibs = 4, ibn = 4
```

The periodic boundary condition is also sometimes used in problems where the computational domain represents part of a much larger physical volume. An example of this is the direct numerical simulation of turbulence. Although only part of the physical volume is modeled, the computational volume chosen is large enough to contain all the relevant scales of motion, so that the flow field calculated is representative of the entire fluid domain and periodic boundary conditions are thus good approximations at the mesh boundaries.

Specified Pressure. The pressure boundary condition, Type 5, specifies the fluid pressure at a particular boundary. The pressure value at the boundary will be that of the fluid in the adjacent fictitious boundary cells. Therefore, for complete specification of the pressure boundary condition, the input array `gasdef` must also be used to give a pressure value in the boundary cells adjacent to the boundary surface. Consider a problem in which `imax = 11`, `jmax = kmax = 7`. To define a pressure boundary condition at the $+i$ boundary of 1 bar (10^6 dynes/cm²), the user would write the following input:

```
ibe = 5,
gasdef(1,1) = 10, 11, 1, 7, 1, 7, 1, 1.e6, 300.,
              1, 0., 9.e99, 1, 1.
```

The additional information, temperature and gas composition, will be used to define the properties of the fluid flowing into the computational domain across the specified-pressure boundary, if that occurs during the calculation.

2. Local Definition

In the following, we shall describe an alternative method of defining boundary conditions. The second method of specifying boundary conditions complements the first method by allowing flexibility in imposing the boundary conditions at arbitrary parts of the mesh and within arbitrary time intervals. While the first method applies boundary conditions to the *entire* extreme surfaces of the mesh in each direction at all time, the second method is capable of imposing boundary conditions on selective surfaces, which can be external or internal, over a specified time range. If a surface has boundary conditions defined by both methods, then only that condition defined by the second method will take effect. Therefore, the user can, for instance, define an entire boundary surface with the first, global method, such as specifying Type 2 to indicate a no-slip wall. Then the user can overwrite a portion of that surface with a velocity boundary condition, specified via the second, local method to simulate a wall with an opening through which fluid is injected.

However, only four of the five types of boundary conditions discussed above can be specified with this local method. The periodic boundary condition must still be specified with the first method, i. e., this condition must be imposed on the whole surface of each of the pair of boundaries in a particular direction. The other boundary conditions can be applied to any surface by specifying the appropriate beginning and ending mesh indices in all three directions. Moreover, the continuative, pressure, and velocity boundary conditions can be imposed over a specific time range. This method is also used to specify velocity boundary conditions, which cannot be done with the first method. The input variables required for defining each of the boundary conditions are described below.

Free-Slip and No-Slip Walls. Since any impenetrable surface is *free-slip* by default, there is no need to explicitly request this boundary option. However, the default free-slip condition can be changed to no-slip via the `nslipdef` variable, which requires 8 entries per definition:

<code>nslipdef(1,*)</code>	Beginning i mesh index (cell-face number).
<code>nslipdef(2,*)</code>	Ending i mesh index (cell-face number).
<code>nslipdef(3,*)</code>	Beginning j mesh index (cell-face number).
<code>nslipdef(4,*)</code>	Ending j mesh index (cell-face number).
<code>nslipdef(5,*)</code>	Beginning k mesh index (cell-face number).
<code>nslipdef(6,*)</code>	Ending k mesh index (cell-face number).
<code>nslipdef(7,*)</code>	Block number (must be set to 1).
<code>nslipdef(8,*)</code>	The side of the surface that is no-slip. Options are
	'lower' negative side,
	'upper' positive side,
	'both' both sides.

In most cases, the beginning and ending i , j , and k mesh indices will define a surface that is coincident with a solid wall, which is to have the no-slip boundary condition. However, `nslipdef` can also be used to specify that all faces of an obstacle volume be no-slip. To do this the user sets `nslipdef(8,*)` to 'both', and specifies the beginning and ending mesh indices that define the volume occupied by the obstacle. Similar to other 2D input array variables such as `walls`, `mobs`, and `gasdef`, the second index of `nslipdef` is used to allow more than one input specification. Consider the following examples:

```
nslipdef(1,1) = 2, 2, 3, 5, 1, 9, 1, 'lower',
nslipdef(1,2) = 3, 3, 3, 5, 1, 9, 1, 'both',
nslipdef(1,3) = 4, 6, 2, 7, 4, 8, 1, 'both'
```

The first definition sets the lower side of the $i = 2$ ($j = 3-5$, $k = 1-9$) surface to no-slip. The second definition sets both the positive and negative sides of the $i = 3$ ($j = 3-5$, $k = 1-9$) surface to no-slip. The last definition sets all the surfaces bounding the volume defined by $i = 4-6$, $j = 2-7$, and $k = 4-8$ to no-slip.

Continuative. The continuative boundary condition can be specified via the variable `cbc`, which requires 9 numbers per definition:

```
cbc(1,*) Beginning i mesh index (cell-face number).
cbc(2,*) Ending i mesh index (cell-face number).
cbc(3,*) Beginning j mesh index (cell-face number).
cbc(4,*) Ending j mesh index (cell-face number).
cbc(5,*) Beginning k mesh index (cell-face number).
cbc(6,*) Ending k mesh index (cell-face number).
cbc(7,*) Block number (must be set to 1).
cbc(8,*) Start time (s).
cbc(9,*) End time (s).
```

For example, the following input

```
cbc = 21, 21, 1, 15, 1, 15, 1, 0.0, 9.e99
```

will specify that the boundary $i = 21$ has a continuative boundary condition, i. e., gradients of pressure, density, etc., across the boundary are zero. Because of the large end time, which exceeds practically all physical problem time, this boundary condition is effective throughout the calculation.

Pressure. The specified pressure boundary condition can be invoked with the input variable `pbc`:

```
pbc(1,*) Beginning i mesh index (cell-face number).
pbc(2,*) Ending i mesh index (cell-face number).
pbc(3,*) Beginning j mesh index (cell-face number).
```


`pbk(4,*)` Ending j mesh index (cell-face number).
`pbk(5,*)` Beginning k mesh index (cell-face number).
`pbk(6,*)` Ending k mesh index (cell-face number).
`pbk(7,*)` Block number (must be set to 1).
`pbk(8,*)` Start time (s).
`pbk(9,*)` End time (s).

Similar to the first method, the values of pressure to be specified at the boundary are taken from the fictitious boundary cells, the fluid conditions of which must be defined with `gasdef`. As an example, consider a mesh that is the same as the example used above for illustrating how to globally specify the pressure boundary condition, i. e., a mesh in which $i_{max} = 11$, $j_{max} = k_{max} = 7$. Now the $+i$ boundary (i. e., the surface where $i = i_{max} - 1 = 10$, refer to Fig. 2 for convention) is by default a free-slip wall. Suppose there is a hole at the center of the wall that is open to some ambient condition on the outside. Furthermore, the pressure will have a step change from 1 to 2 atmospheres (plus other changes in the ambient condition) at 100 s. The appropriate input would be

```

pbk(1,1) = 10, 10, 3, 4, 3, 4, 1, 0.0, 9.e99,
gasdef(1,1) = 10, 11, 3, 4, 3, 4, 1, 1.0132e6, 300.,
              2, 0., 100., 1, 1.,
gasdef(1,2) = 10, 11, 3, 4, 3, 4, 1, 2.0265e6, 400.,
              2, 100., 9.e99, 1, 0.5, 2, 0.5
  
```

If inflow occurs at the $i = 10$ boundary during the calculation, the properties (pressure, temperature, composition) of the fluid entering will be those defined in the boundary cells by `gasdef`.

Velocity. The input variable `vbc` can be used to specify velocity boundary conditions. Each definition requires 10 numbers:

`vbc(1,*)` Beginning i mesh index (cell-face number).
`vbc(2,*)` Ending i mesh index (cell-face number).
`vbc(3,*)` Beginning j mesh index (cell-face number).
`vbc(4,*)` Ending j mesh index (cell-face number).
`vbc(5,*)` Beginning k mesh index (cell-face number).
`vbc(6,*)` Ending k mesh index (cell-face number).
`vbc(7,*)` Block number (must be set to 1).
`vbc(8,*)` Element of the `vvalue` array that will define the velocity value.
`vbc(9,*)` Start time (s).
`vbc(10,*)` End time (s).

Note that `vbc(8,*)` does not directly specify what the velocity value is; rather, it specifies an integer that points to the corresponding element in the array `vvalue` that stores velocity values. The sign convention used for the velocity is that positive velocity indicates flow in the direction of increasing i , j , or k index. The following

example illustrates the use of `vbc` and `vvalue` to define inflow and outflow conditions for a mesh in which `imax = 11`, `jmax = kmax = 7`:

```
vbc(1,1) = 1, 1, 1, 7, 1, 7, 1, 3, 1.0, 2.0,
vbc(1,2) = 1, 1, 1, 7, 1, 7, 1, 2, 2.0, 5.0,
vbc(1,3) = 10, 10, 1, 7, 1, 7, 1, 2, 2.0, 5.0,
gasdef(1,8) = 0, 1, 1, 7, 1, 7, 1, 1.0132e6, 298.,
              2, 0., 9.e99, 1, 1.,
vvalue = 10., 30., 50., 20.
```

The first two `vbc` definitions specify a velocity of 50 cm/s during the time interval between 1 and 2 s, followed by a lower velocity of 30 cm/s from 2 to 5 s at the $-i$ boundary. The third `vbc` definition specifies a velocity of 30 cm/s during the time interval between 2 and 5 s at the $+i$ boundary. (Since all boundaries are by default free-slip walls, the $-i$ and $+i$ boundaries are closed until the beginning time of the respective `vbc` definitions.) Because these velocities are positive, the boundary condition at the $-i$ boundary represents an inflow condition, whereas that at the $+i$ boundary represents an outflow condition. For inflow conditions, the user must also define the fluid condition in the boundary cells adjacent to the inflow boundary. This is done in the above example with the 8th `gasdef` definition, which states that the incoming fluid is at atmospheric pressure and room temperature, and consists of pure gas component 1. (The gas component number is defined by the order in which the gas species are listed in the definition of the `mat` array.)

VI. DEFINITION OF SOLID HEAT STRUCTURES

In Sec. III.C, we discuss how to define solid walls and obstacles, with the variables `walls` and `objs` in NAMELIST group `xput`, in the computational mesh to restrict the fluid flow path. Our convention is that walls are surfaces and obstacles are volumes within the mesh across which the fluid is not allowed to penetrate. If heat transfer is invoked (by setting the variable `ihflag = 1` in NAMELIST group `rheat`), then all the defined walls and obstacles, as well as all the closed computational boundaries, will exchange heat with the fluid cells across the solid-fluid interfaces. A time dependent heat conduction equation is solved for each solid structure, with an implicit scheme. As an approximation, which greatly improves computational efficiency and speed, we assume 1D heat conduction, i. e., heat conducts only in a direction perpendicular to the interface between the solid and fluid. (In other words, if three orthogonal faces of an obstacle are exposed to fluid, then the code solves a 1D heat conduction equation for each of the directions, independently.) Another simplification is that the solid properties (conductivity, density, and heat capacity) have negligible dependence on temperature changes.

For the purpose of heat conduction calculations, we distinguish the solid surfaces where energy exchange with the fluid occurs into two types, wall heat structures and slab heat structures, depending on the depth of solid material behind the surfaces.

A. Wall Heat Structure

A solid surface is a *wall* surface if the depth of solid material behind it is *thin*, and the other side of the solid is also exposed to fluid. Therefore a wall heat structure is always two-sided, with its temperature profile determined by the adjacent fluid cell temperatures on both sides and its heat capacity and conductivity. There are two cases in which we have wall heat structures:

1. All impenetrable surfaces defined by `walls` in NAMELIST group `xput` will be considered wall heat structures, because by definition these are infinitely thin surfaces between adjacent fluid cells. (However, as discussed later, these surfaces will be assigned some effective thickness for the heat conduction calculation.)
2. For obstacle mesh cells (defined by `mobs` in NAMELIST group `xput`) surrounded by fluid cells on opposite sides, whether the heat structure type is a wall depends on the thickness between the two opposite sides which are exposed to fluid. If the thickness is smaller than `slabthk`, an input variable in NAMELIST group `rheat`, then the code treats the opposing surfaces as the two sides of a wall heat structure. If the thickness is greater than or equal to `slabthk`, then each of the opposing surfaces will be treated as a slab surface. Figure 6 illustrates some possibilities for obstacle surfaces treated either as those belonging to a wall or a slab heat structure.

B. Slab Heat Structure

A solid surface belongs to an infinitely thick *slab* heat structure if the solid material behind it is thicker than the `slabthk` value input by the user. For a slab heat structure, we assume that it is so thick that the heat or temperature wave due to exchange with the fluid never penetrate deep enough to affect the temperature profile near its back side, within the problem time scale. Therefore, if its backside is also exposed to fluid, then the backside surface will be treated as belonging to a separate slab heat structure, and the temperature distribution within each slab will only be affected by the temperature of the fluid in contact with its front side. There are two cases where we have slab heat structures:

1. All boundaries of the computational domain not open to flow will be treated as slabs (see Fig. 6).
2. For obstacles (defined by `mobs`), each surface exposed to fluid will be treated as an independent slab surface if the depth behind the surface is greater or equal to the input variable `slabthk`. Refer to Fig. 6 for some examples.

Fig. 6. Illustration of an obstacle surface belonging to a wall heat structure or to a slab heat structure, depending on the value of `slabthk`. Also shown are nonflow boundaries (of the mesh) treated as slab surfaces.

Note that the user does not directly define or specify slab or wall heat structures. The code automatically determines the heat structure type of surfaces corresponding to closed computational boundaries and those of the solid structures defined by `walls` and `mobs`. The user only determines the thickness criterion via the input variable `slabthk`.

Regardless of whether a heat structure is a slab or wall, the conduction calculation requires some physical properties and a spatial dimension. The spatial dimensions of obstacle cells are defined by the mesh. However, for surfaces defined by `walls`, the user will have to input an effective physical thickness, even though mathematically surfaces between adjacent fluid cells in the mesh have no thickness. Furthermore, the user must define the material in the wall. These definitions are accomplished through the 8th element of the `walls` array in NAMELIST group `xput`, and the `walldf` array in NAMELIST group `rheat`:

walls(8,*) Integer to identify the type of wall through the walldef array that stores material identification number and thickness for each wall type.

walldef(1,*) Material identification number. Options are
1 concrete,
2 steel,
3 superconducting material.

walldef(2,*) Thickness of wall (cm).

HMS offers a choice among three solid conducting materials with the following properties:

Material Number	Material Name	Density (g/cm ³)	Specific Heat Capacity (ergs/g-K)	Thermal Conductivity (ergs/s-cm-K)
1	Concrete	2.40	1.004×10^7	2.0×10^5
2	Steel	7.85	4.904×10^6	5.0×10^6
3	Superconductor	10.0	1.000×10^9	1.0×10^{20}

Because the code will calculate the obstacle thicknesses, the user only has to input the material which the obstacle is made of. This can be done via the 8th element of the mobs array:

mobs(8,*) Material identification number. Options are
1 concrete,
2 steel,
3 superconducting material.

C. Sink Heat Structure

In some practical problems where the computational mesh is not fine enough to represent the details of all internal structures, it is desirable to have the capability of modeling the heat transfer effects of these "subgrid" structures. That is, we need a method of modeling heat transfer between the fluid in a computational cell and the structures embedded within the cell. We accomplish this by defining a third type of heat structure, which we call distributed heat sinks.

Sinks are heat structures defined by the user which are assumed to be distributed within the fluid cells. Each sink is characterized by the simple model illustrated in Fig. 7. Similar to the other heat structure types, 1D heat conduction is calculated across the sink structure thickness. Both sides of a sink heat structure are exposed to the same fluid cell, and it is assumed that the structure temperature profile is symmetric about the centerline, so that only conduction in half the

structure thickness needs to be calculated. Definition of sink heat structures is done with the input array `sinkdef` in NAMELIST group `rheat`:

```

sinkdef(1,*)    Beginning i mesh index (cell-face number).
sinkdef(2,*)    Ending i mesh index (cell-face number).
sinkdef(3,*)    Beginning j mesh index (cell-face number).
sinkdef(4,*)    Ending j mesh index (cell-face number).
sinkdef(5,*)    Beginning k mesh index (cell-face number).
sinkdef(6,*)    Ending k mesh index (cell-face number).
sinkdef(7,*)    Block number (must be set to 1).
sinkdef(8,*)    Material identification number. Options are
                  1    concrete,
                  2    steel,
                  3    superconducting material.
sinkdef(9,*)    Total material volume (cm3).
sinkdef(10,*)   Average material thickness (cm).

```

Note that each sink definition can cover a fluid region (specified by the starting and ending *i*, *j*, and *k* mesh indices) consisting of multiple fluid cells. If such is the case, then the code will distribute the sink material to each fluid cell according to the cell volume, i. e., a fluid cell having twice the volume as another one will get twice as much sink material. According to our model, depicted in Fig. 7, specifying the volume and thickness of the sink material will also give the surface area through which heat exchange with the fluid occurs.

Examples. Consider the following input, which illustrates the use of `walls`, `mobs`, `walldef`, and `sinkdef` in a problem where heat transfer calculations are invoked:

```

$xput
...
walls(1,1) = 3, 3, 6, 7, 4, 7, 1, 5,
walls(1,2) = 2, 5, 8, 8, 4, 5, 1, 4,
...
mobs(1,1) = 3, 5, 2, 4, 2, 5, 1, 1,
...
$rheat
ihtflag = 1,
slabthk = 100.,
...
walldef(1,4) = 1, 20.,
walldef(1,5) = 2, 5.0,
...
sinkdef(1,1) = 3, 5, 10, 11, 7, 8, 1, 2, 8000., 10.,
sinkdef(1,2) = 5, 6, 10, 11, 7, 8, 1, 2, 4000., 10.,
...

```

Fig. 7. Schematic of the distributed heat sink used in HMS.

All the obstacle cells defined by the first mobs definition are made of concrete because $\text{mobs}(8,1) = 1$. The code will determine whether each surface of the individual obstacle cells exposed to fluid is a slab or wall surface, depending on the dimensions of the obstacle cells as compared to slabthk . Because $\text{walls}(8,1) = 5$, surfaces defined by the this walls definition are of type 5, which has been defined by walldef to be made of steel (material number 2) and having a thickness of 5 cm. Similarly, because $\text{walls}(8,2) = 4$, surfaces defined by the second walls definition are made of concrete (material number 1) and are 20 cm thick.

Regarding distributed heat sinks, the first sinkdef assigns sinks to two fluid cells with (i,j,k) indices of $(4,11,8)$ and $(5,11,8)$. (Refer to Fig. 2 for the celling numbering convention used in HMS.) The sink material is steel because $\text{sinkdef}(8,1) = 2$. If the two cells have the same size, then the distributed heat sink in each cell will have a volume of 4000 cm^3 , a thickness of 10 cm, and a surface area (on each side) of 400 cm^2 . The second sinkdef specifies only one fluid cell,

(6,11,8), and the heat sink in that cell also has a volume of 4000 cm^3 and a thickness of 10 cm.

Other Heat Structure Input. The heat conduction equations are solved by finite difference. The user can specify the number of nodes used for each type of heat structure. Note that all heat structures of the same type will have the same number of nodes. The input variables for this purpose are in NAMELIST group rheat, and are explained below:

nhtesink	Number of discretized elements to be used for calculating heat conduction in a sink heat structure. Default = 2.
nhtewall	Number of discretized elements to be used for calculating heat conduction in a wall heat structure. Default = 2.
nhteslab	Number of discretized elements to be used for calculating heat conduction in a slab heat structure. Default = 2.

In HMS, all solid heat structures are assumed to have a uniform temperature distribution in the beginning of a problem. These initial heat structure temperatures can be specified with the following input variables in NAMELIST group rheat:

tsink0	Initial temperature in sink heat structures (K). Default = 300.
twall0	Initial temperature in wall heat structures (K). Default = 300.
tslab0	Initial temperature in slab heat structures (K). Default = 300.

If any initial temperature is set negative, then the corresponding heat structures are assumed to be in thermal equilibrium with the contacting fluid cells. For example, if $tsink0 = -1$, and the initial fluid temperature is 298 K, then a sink heat structure will also have a temperature of 298 K in the beginning of the calculation.

VII. PHYSICAL MODEL OPTIONS

A. Body Forces

The momentum conservation equations solved by HMS include body forces, or forces given by the product of the fluid density and a constant acceleration. The most common body force is that due to gravity. Because the user can orient the computational mesh arbitrarily with respect to the gravity vector, the code by default sets the body force acceleration term to zero in all three directions. To specify the acceleration vector due to gravity, the user has to define the values of the following variables in NAMELIST group xput:

gx	Acceleration due to gravity in the i- (x- or r-) direction (cm/s^2). Default = 0.
----	---

- gy Acceleration due to gravity in the j- (y - or θ -) direction (cm/s^2).
 Default = 0.
- gz Acceleration due to gravity in the k- (z -) direction (cm/s^2).
 Default = 0.

For example, in a problem where the z -direction is vertically upward, normal earth gravity is activated with the following input:

$$gx = 0., \quad gy = 0., \quad gz = -980.$$

Besides specifying the gravity term, the code offers the option of starting a calculation with a pressure gradient in the fluid that is in equilibrium with its own body weight. This option is specified through the `ihystat` variable in NAMELIST group `xput`:

- `ihystat` Option flag for imposing an initial hydrostatic pressure gradient in the fluid cells according to the acceleration components `gx`, `gy`, and `gz`: 1 means ON, 0 means OFF (default).

B. Diffusion of Mass, Energy, and Momentum

In HMS, transport due to gradient diffusion is modeled by Fick's Law-type fluxes in the conservation equations for mass, energy, and momentum. The default is no gradient diffusion. The user must explicitly specify them using the following input variables in NAMELIST group `xput`:

- `idiffmae` Option flag for mass and energy diffusion:
 1 means ON, 0 means OFF (default).
- `idiffmom` Option flag for momentum diffusion:
 1 means ON, 0 means OFF (default).

Note that these option flags apply to the diffusion terms, which include both molecular and turbulent transport. However, by default, the code does not calculate turbulent diffusion; the user must explicitly specify that model option using the `tmodel` input variable (see discussion on turbulence in the next section).

For momentum diffusion, if `idiffmom` is set to 1, and no turbulence model is activated (`tmodel = 'none'`) there is an additional option of how the molecular diffusion coefficient is calculated. The user can specify whether the input dynamic viscosity, μ (in units of poise or g/cm-s), or the input kinematic viscosity, $\nu \equiv \mu/\rho$ (cm^2/s), is to be used for calculating the diffusional momentum fluxes or viscous stresses. This is done via the following variables in NAMELIST `xput`:

- `muoption` Option specifying whether the input variable `nu` or `cmug` is to be used to for viscous stress calculation:

	1 means nu is used (default), 2 means cmug is used.
nu	Kinematic viscosity, ν (cm ² /s). Default = 0.
cmug	Dynamic viscosity, μ (g/cm-s). Default = 1.8×10^{-4} .

For the diffusion of mass species and energy, the code uses the input variables `schmidt` and `prandtl` (see Sec. IV.B), respectively, to determine the appropriate diffusion coefficient.

C. Turbulence

The Navier-Stokes equations are nonlinear, and when the flow speed exceeds certain criteria (as measured by the Reynolds or Grashof number) they become unstable in the sense that the solution (fluid velocity, pressure, temperature, etc.) exhibits oscillatory or fluctuating behavior. These fluctuations can be calculated directly if the computational mesh is so fine and the time advancement increments are so small that even the smallest eddies (scales of motion) can be resolved. However, in practically all problems, this is not the case. Furthermore, in most problems, there is no need to trace the *exact* behavior of every fluid element at every instant. It is often adequate to calculate the fluid behavior averaged over some time interval and space region that are large compared to the turbulent fluctuation scale, but small compared to the problem transient time scale. (Such an averaging procedure is called Reynolds averaging.)

Therefore, the equations solved in HMS are actually Reynolds averaged, with the dependent variables being mean quantities after the turbulent fluctuations are removed. However, the fluctuations cannot be simply ignored. The equations retain terms that are averages of products like $u'v'$ and $u'\phi'$, where u' , v' are fluctuating velocity components and ϕ' is a fluctuating scalar quantity such as internal energy or a mass species concentration. These terms represent transport fluxes due to turbulent fluctuations and must be determined by some turbulence model that relates these fluxes to the calculated quantities.

In HMS, all turbulent fluxes are modeled like a molecular diffusion process. Turbulent momentum fluxes (stresses) are modeled as

$$\tau_{t,ij} \equiv \mu_t \partial u_i / \partial x_j, ,$$

where $\partial u_i / \partial x_j$ is the velocity gradient tensor. This is the so-called Boussinesq's hypothesis, which states that turbulent transport can be modeled as a diffusion process with an eddy viscosity, μ_t . In HMS, we further assume that the turbulent diffusion coefficients for mass and energy are both the same as that for momentum, and is given by μ_t / ρ , where ρ is the fluid density.

Therefore the task of modeling turbulence is reduced to determining the eddy viscosity . In HMS, three turbulence models are available. The choice can be specified via the input character variable `tmodel` in NAMELIST group `xput`:

<code>tmodel</code>	Symbol designating the type of turbulence model to be used:
<code>'none'</code>	No turbulence model, i. e., only molecular diffusion.
<code>'alg'</code>	Algebraic turbulence model.
<code>'sgs'</code>	Subgrid scale turbulence model.
<code>'ke'</code>	k - ϵ turbulence model.

Note that the turbulence model invoked via `tmodel` is only in effect when diffusion calculations have also been specified, i. e., when `idiffmom` and/or `idiffmae` has been set to 1.

1. Algebraic Model

The algebraic turbulence model used in HMS assumes that the turbulent viscosity can be written as

$$\mu_t = c_\mu \rho \sqrt{k} l ,$$

where k is the turbulent kinetic energy (per unit mass), ρ is the fluid density, l is a turbulent length scale, and c_μ is a constant coefficient. In this model, it is assumed that the turbulent kinetic energy is a specified fraction of the fluid mean kinetic energy, i. e.,

$$k = f \cdot \frac{1}{2} (u^2 + v^2 + w^2) .$$

The parameters to be input for the algebraic turbulence model are c_μ , l and f , which are represented by the following variables in NAMELIST group `xput`:

<code>cmu</code>	Constant c_μ for the algebraic turbulence model. Default = 0.05.
<code>clength</code>	Length scale, l , for the algebraic turbulence model (cm). Default = 30.48.
<code>fracke</code>	Fraction of the mean kinetic energy, f , used in the algebraic turbulence model to determine the turbulent kinetic energy. Default = 0.1.

2. Subgrid Scale Model

The subgrid scale turbulence model, similar to the algebraic model, also assumes that the turbulent viscosity can be written as

$$\mu_t = c_\mu \rho \sqrt{k} l ,$$

where k is the turbulent kinetic energy (per unit mass), ρ is the fluid density, l is a turbulent length scale, and c_μ is a constant coefficient. However, this model attempts to model the effects of the turbulent eddies that are not resolved by the mesh (the so-called subgrid eddies). Hence l is not an input parameter, but instead it is determined by the code from the average cell size:

$$l = (\Delta x^2 + \Delta y^2 + \Delta z^2)^{1/2}.$$

The turbulent kinetic energy, instead of being calculated from the mean velocities algebraically, is determined by solving the following transport equation:

$$\begin{aligned} & \frac{\partial(\rho k)}{\partial t} + \nabla \cdot (\rho k \mathbf{u}) \\ &= -\frac{2}{3} \rho k \nabla \cdot \mathbf{u} + \tau : \nabla \mathbf{u} + \nabla \cdot (\mu \nabla k) - \rho k^{3/2} / l. \end{aligned}$$

Consequently, there is only one input parameter, c_μ , for this model, which is represented by the following variable in NAMELIST `xput`:

`cmusgs` Constant c_μ for the subgrid scale turbulence model.
Default = 0.05.

Because a time-dependent transport equation for k is solved, the user must define the initial values of k in all fluid cells. FIMS uses an approach similar to `gasdef` for this purpose, i. e., the user defines initial turbulence quantities in a volume region for a specified time range. The array variable for this is in NAMELIST group `xput`:

<code>turbdef(1,*)</code>	Beginning i mesh index (cell-face number).
<code>turbdef(2,*)</code>	Ending i mesh index (cell-face number).
<code>turbdef(3,*)</code>	Beginning j mesh index (cell-face number).
<code>turbdef(4,*)</code>	Ending j mesh index (cell-face number).
<code>turbdef(5,*)</code>	Beginning k mesh index (cell-face number).
<code>turbdef(6,*)</code>	Ending k mesh index (cell-face number).
<code>turbdef(7,*)</code>	Block number (must be set to 1).
<code>turbdef(8,*)</code>	Integer pointer to location in <code>tkeval</code> array for value of turbulent kinetic energy.
<code>turbdef(9,*)</code>	Integer pointer to location in <code>epsval</code> array for value of turbulence dissipation rate, ϵ .
<code>turbdef(10,*)</code>	Integer pointer to location in <code>sclval</code> array for value of turbulence length scale.
<code>turbdef(11,*)</code>	Time (s) at which "turbulence definition" begins.
<code>turbdef(12,*)</code>	Time (s) at which "turbulence definition" ends.

The user must input a valid pointer in `turbdef(8,*)` to locate the `tkeval` array element that stores the turbulent kinetic energy value. Note that `turbdef(9,*)` and `turbdef(10,*)` will be ignored for the subgrid scale model. The following input illustrates the use of `turbdef` to define an initial condition and a boundary condition for solving the k transport equation:

```
turbdef(1,1) = 1, 50, 1, 32, 1, 45, 1, 2, 0, 0, 0.0, 0.0,
turbdef(1,2) = 0, 1, 2, 5, 4, 6, 1, 1, 0, 0, 0.0, 1.e9,

tkeval      = 1.e3, 0.0,
```

The first `turbdef` defines an initial condition for k . Because it points to the second element in `tkeval`, the value of k is 0 initially. The second `turbdef` specifies a boundary condition at the -i boundary, between j-indices of 2 and 5, and between k-indices of 4 and 6. The boundary value of k specified is 1000 (cm^2/s^2).

3. k - ε Model

In the k - ε model, it is assumed that the turbulent viscosity can be written as

$$\mu_t = c_\mu \rho k^2 / \varepsilon ,$$

where k is the turbulent kinetic energy (per unit mass), ρ is the fluid density, ε is the rate of dissipation of turbulent kinetic energy, and c_μ is a constant coefficient. The turbulent kinetic energy is determined from solution of its transport equation:

$$\begin{aligned} & \frac{\partial(\rho k)}{\partial t} + \nabla \cdot (\rho k \mathbf{u}) \\ &= \nabla \cdot \left(\frac{\mu}{\sigma_k} \nabla k \right) + \tau : \nabla \mathbf{u} + \mu \alpha \mathbf{g} \cdot \nabla T - \rho \varepsilon + k S . \end{aligned}$$

The terms on the right-hand side are, in order, diffusional transport, production by viscous stresses, production by buoyancy, viscous dissipation, and generation from other sources. Similarly, the rate of dissipation, ε , is determined from solution of its transport equation:

$$\begin{aligned} & \frac{\partial(\rho \varepsilon)}{\partial t} + \nabla \cdot (\rho \varepsilon \mathbf{u}) \\ &= \nabla \cdot \left(\frac{\mu}{\sigma_\varepsilon} \nabla \varepsilon \right) + c_1 \frac{\varepsilon}{k} \tau : \nabla \mathbf{u} + c_1 \frac{\varepsilon}{k} \mu \alpha \mathbf{g} \cdot \nabla T - c_2 \rho \frac{\varepsilon^2}{k} + \varepsilon S . \end{aligned}$$

The following variables in NAMELIST group `xput` are used to set the k - ϵ model parameters:

<code>cmuke</code>	c_μ used in the k - ϵ turbulence model. Default = 0.09.
<code>c1ke</code>	c_1 used in the k - ϵ turbulence model. Default = 1.44.
<code>c2ke</code>	c_2 used in the k - ϵ turbulence model. Default = 1.92.
<code>sigmak</code>	σ_k used in the k - ϵ turbulence model. Default = 1.
<code>sigmae</code>	σ_ϵ used in the k - ϵ turbulence model. Default = 1.3

Because the k - ϵ model solves time-dependent transport equations for both k and ϵ , the user must define initial conditions for these quantities. The `turbdef` array described above with the subgrid scale model is also used for defining initial and boundary conditions for the k - ϵ model. The user must define a valid pointer for the array `tkeval` in `turbdef(8,*)`, and a valid pointer for the array `epsval` in `turbdef(9,*)`. The input in `turbdef(10,*)` will be ignored. The following input illustrates the use of `turbdef` to define an initial condition and a boundary condition for solving the k and ϵ transport equations:

```
turbdef(1,1) = 1, 50, 1, 32, 1, 45, 1, 2, 1, 0, 0.0, 0.0,
turbdef(1,2) = 0, 1, 2, 5, 4, 6, 1, 1, 2, 0, 0.0, 1.e9,

tkeval      = 1.e3, 0.0,
epsval      = 0.0, 800.,
```

The first `turbdef` defines an initial condition for k and ϵ . The values of both k and ϵ are initially 0. The second `turbdef` specifies a boundary condition at the `-i` boundary, between `j`-indices of 2 and 5, and between `k`-indices of 4 and 6. The boundary values of k and ϵ specified are $1000 \text{ cm}^2/\text{s}^2$ and $800 \text{ cm}^2/\text{s}^3$, respectively.

D. Combustion

HMS models the combustion of hydrogen in air with the following single, global reaction:

where the coefficients a, b, c, d, e, f , and g are molar densities (moles/ cm^3) of the respective reactants and products. The reaction rate, $\dot{\omega}$, is determined from a modified Arrhenius law. Computationally, during each time cycle, the change in density of the fuel, δ_H , is calculated implicitly first when the fuel-oxidizer mixture is fuel lean. Then the new molar densities of H_2 , O_2 , and H_2O are determined from the old values, respectively, as $f = a - \delta_H$, $g = b - \delta_H/2$, $e = d + \delta_H$. When the mixture is fuel rich,

then the density change for the oxidizer (O_2), δ_O , is calculated first, and the new molar densities are determined as $g=b-\delta_O$, $f=a-2\delta_O$, $e=d+2\delta_O$. This will ensure that unphysical negative species densities will not result from the chemical reaction. In addition to changing the reactant and product species densities, the reaction rate is used to compute the increase in energy due to the combustion process.

This global reaction is the only combustion model in HMS. The reaction rate parameters and heat of reaction are fixed in the code. Therefore, the user only has to specify the following variable, in NAMELIST group `xput`, in order to turn on the hydrogen combustion model in the calculation:

`iburn` Option flag for the hydrogen burn model:
1 means ON, 0 means OFF (default).

E. Heat Transfer

Although the energy conservation equation is solved by the code under all circumstances, heat transfer between the fluid and any solid structures are by default not calculated. Therefore, in a problem that, for instance, involves hydrogen combustion (`iburn = 1`) but the heat transfer model is not explicitly requested, a temperature rise will still be calculated, which is only valid if the process is adiabatic or if the problem time scale is fast compared to the heat transfer time scale.

To activate heat transfer and steam condensation calculations, using models based on the argument of analogies between momentum, heat, and mass transfer, the user must specify the following variable in NAMELIST group `rheat`:

`rheat` Option flag to turn on heat transfer and condensation calculations:
1 means ON, 0 means OFF (default).

The user should also refer to Sec. VI, on defining heat structures, for complete specification of a heat transfer problem.

VIII. OPTIONS ON NUMERICAL SOLUTION PROCEDURE

A. Pressure Iteration

The numerical algorithm used in HMS for solving the coupled fluid mass, momentum, and energy equations includes an implicit pressure iteration phase, which enables the code to simulate compressible flows without the computational time step being limited by the speed of sound. The implicit, iterative calculation is the solution of matrix equations arising from discretization of Poisson type pressure equations by the preconditioned conjugate residual (PCR) method (see Ref. 1). The PCR algorithm is constructed such that the true solution is obtained after a *finite* number of iterations. However, that number is roughly the total number of

computational cells, so in order to complete most practical problems in a reasonable amount of computer time, we have to specify certain error acceptance criteria to terminate the iteration procedure and move on to the next calculation phase. Note that the matrix solution has to be performed at each time cycle, so it is important to keep the iteration numbers reasonably low for complex, long transient problems.

The accuracy of an iterative solution is indicated by a residual vector r , which would be zero if the solution is exact. In HMS, the iteration will stop if all the components in r are less than a specified error value. This convergence criterion or tolerance, ϵ , as defined for the actual matrix equations in the code, is a dimensionless quantity. The code has a default value for this parameter in the beginning of the calculation, and a default set of criteria for changing its value during the calculation, depending on the number of iterations required for convergence in the previous time cycle. The user usually does not have to change these default settings. However, to allow flexibility in controlling the iteration procedure for a wide range of problems, the code provides the following input variables in NAMLIST group `xput`:

<code>epsi0</code>	Initial value of ϵ . Default = 1×10^{-5} .
<code>epsimin</code>	Minimum value of ϵ . Default = 1×10^{-6} .
<code>epsimax</code>	Maximum value of ϵ . Default = 1×10^{-3} .
<code>itdownt</code>	If the current number of iterations required for convergence is less than this parameter, then the value of ϵ in the next time cycle will be decreased by 5% (i. e., the error tolerance will be tightened). This iteration number is also used to adjust the time step, as described in the next section. Default = 50.
<code>itupdt</code>	If the current number of iterations required for convergence exceeds this parameter, then the value of ϵ in the next time cycle will be increased by 5% (i. e., the convergence criterion will be less stringent). Default = 100.

Therefore ϵ will increase, decrease, or remain the same as the solution advances in time, depending on whether the current iteration number exceeds `itupdt`, or if it is less than `itdownt`. (Note that `itdownt` is also used to adjust the time step size according to the number of iterations necessary for convergence. This will be explained further below.) However, the change in ϵ is limited by `epsimin` and `epsimax`. If the user wants to use a fixed convergence criterion, he should set `epsi0`, `epsimin`, and `epsimax` the same.

The user can also specify the maximum number of iterations after which the iteration stops and the calculation continues, regardless of whether the current matrix solution satisfies the convergence criterion. This is done via the following variable in NAMELIST group `xput`:

itmax Maximum number of iterations (per time cycle) allowed.
Default = 20.

The code prints out iteration and time step information for each time cycle to a file called `cyclinfo`. (The same information is output to the terminal at a specified frequency. Section IX describes how to set this frequency.) The following is part of a `cyclinfo` file that illustrates the information reported:

TIME	CYCLE	PITER	DELT	DMAX	EPSI
2.601E-02	1	38	2.601E-02	6.811E-06	1.000E-05
4.548E-02	2	66	1.947E-02	6.909E-06	1.000E-05
5.793E-02	3	62	1.245E-02	6.764E-06	1.000E-05
7.063E-02	4	62	1.270E-02	8.009E-06	1.000E-05
8.358E-02	5	57	1.295E-02	6.708E-06	1.000E-05
9.679E-02	6	58	1.321E-02	9.230E-06	1.000E-05
1.103E-01	7	57	1.348E-02	8.370E-06	1.000E-05
1.240E-01	8	57	1.375E-02	8.589E-06	1.000E-05
1.380E-01	9	57	1.402E-02	7.250E-06	1.000E-05
1.523E-01	10	57	1.430E-02	7.985E-06	1.000E-05

Here the first 10 computational time cycles are reported. The `TIME` column gives the problem time (s), and `DELT` is the time-step size. `EPSI` is the value of ϵ and it can be seen that it remains unchanged. `PITER` is the number of pressure iterations that have been carried out. The pressure matrix solution in each cycle may or may not have converged. Whether convergence has been achieved is indicated by comparing `EPSI` to `DMAX`, which represents the largest component in the residual vector $\mathbf{r}$. In this example, convergence has been achieved in every cycle.

B. Time-Step Control

Because HMS solves the time-dependent conservation equations, a calculation proceeds in finite time steps (also called cycles) until the problem end time or the specified maximum number of cycles has been reached. The end time and maximum number of cycles are defined by the following input variables (all variables discussed in this section are in NAMELIST group `xput`):

twfin Time (s) at which the problem is finished. Default = 10.
maxcyc Maximum allowable number of cycles. Default = 10.

Hence, if the user does not specify the above variables, the calculation will stop at a problem time of 10 s or after 10 cycles have been carried out, whichever occurs first.

How fast a problem can be completed depends on the time-step size Δt chosen. The initial, minimum, and maximum values of Δt to be used are defined with the following input variables:

delt0	Beginning time-step size (s). Default = 0.02.
deltmin	Minimum time-step size (s). Default = 1×10^{-4} .
deltmax	Maximum time-step size (s). Default = 1×10^{30} .

The maximum and minimum values are used to define a range within which Δt can be varied during the calculation. Hence the maximum Δt allowed is `deltmax`, and if Δt goes below `deltmin`, the problem will terminate. HMS has algorithm for adjusting the time increment during a calculation, or the user can force the code to use a fixed time step size.

The numerical method used in HMS treats most physical processes implicitly in time, except the advection and diffusion terms. Explicit treatment of these terms leads to the fact that the solution procedure is only stable if the time-step size Δt satisfies both the Courant criterion and the diffusion limit. To ensure numerical stability, the code limits Δt as follows:

$$\Delta t \leq \frac{1}{4} \min \left(\frac{\Delta_i}{u_i}, \frac{\Delta_j}{u_j}, \frac{\Delta_k}{u_k} \right) \quad (\text{Courant Limit})$$

$$\Delta t \leq 0.9 \left(\frac{\Delta_{\min}^2}{6 \nu_{\text{eff}}} \right) \quad (\text{Diffusion Limit})$$

For the Courant limit, Δ_j and u_j , etc., are local cell spacing and velocity in all three directions. For the diffusion limit, ν_{eff} is the effective diffusivity in a cell, $\Delta_{\min}$ is the smallest dimension of that cell. The limit is applied, of course, only if the diffusion option is turned on (`idiffmom` and/or `idiffmae` set to 1). In a problem where all diffusion processes and turbulence modeling are turned on, the effective diffusivity used in the diffusion limit will be the sum of the turbulent diffusivity and the largest of the molecular diffusivities for mass, energy, and momentum.

HMS by default will try to adjust Δt to achieve maximum efficiency while satisfying the stability limits. The code increases or decreases Δt for the next computational cycle according to the number of pressure iterations required for convergence in the current cycle. If the number is greater than the input variable `itdowndt` (default = 50), then Δt will be decreased by 5%. If the iteration number is less than `itdowndt`, then Δt will be increased by 5% in the next cycle.

If the user wants a fixed time-step size, the following input variable should be specified:

autot	Option flag for automatic control of time-step size Δt :
	1. means Δt is adjusted by code during calculation (default).

0. means the input `del t 0` is used for Δt throughout calculation.

C. Advection Scheme

Each of the conservation equations solved by HMS contains a convective flux term $\nabla \cdot (\phi \mathbf{u})$, where ϕ is the conserved quantity (mass density, internal energy, or momentum) and $\mathbf{u}$ is the velocity vector. The default numerical method for discretizing this term in space is the first-order donor-cell method. The donor-cell scheme is simple and fast, and does not suffer from the spurious oscillations caused by some higher-order schemes. However, the solution obtained has larger numerical diffusion error than those given by higher-order schemes. Therefore, the code provides an alternative advection scheme, which was originally developed by van Leer (see Ref. 3). The van Leer scheme is a second-order, slope-limiting method that has the *monotone* property. (A monotone scheme does not have the unphysical undershoots and overshoots exhibited by many higher-order methods.) In a problem where numerical diffusion errors need to be minimized, the more sophisticated van Leer scheme should be substituted for the default simple donor-cell method to calculate the convective fluxes. This can be done via the following input variable in NAMELIST group `xput`:

`ifvl` Option flag for turning on the van Leer advection scheme:
 0 means the donor-cell method will be used (default).
 1 means van Leer scheme will be used.

IX. OUTPUT AND RESTART

A. Graphical Outputs

A typical problem solved by HMS usually involves many computational cells and time steps. To analyze the calculation results by studying printed tables of numbers is useful in special cases, but inefficient in many situations. Therefore, graphical display of results has been the primary focus of the HMS code development effort with regard to user outputs. Graphical outputs have the advantage of showing data in a compact and comprehensive form so that important trends and phenomena can be easily identified.

The following is a description of various plotting capabilities available in the code. Most input variables regarding graphical outputs are in NAMELIST group `graphic`. One exception is the time interval between plots, which control the plotting frequency. This variable is in NAMELIST group `xput`:

`pltdt` Time interval (s) between successive sets of 1D profile, 2D contour, 2D and 3D vector plots are generated, if such plots are requested.
 Default = 1.

All input variables discussed in the rest of this section are in NAMELIST group `rgraphic`, except otherwise noted.

Sample plots are not given in this section. However, in Sec. XII (SAMPLE PROBLEMS), extensive examples are shown of the graphics generated by HMS. The plot file generated is called `pgf`.

1. Basic Plots

HMS always produces five basic plots (each on a separate page or frame) of the mesh and places them in the beginning of the plot file. These plots show, in order, 2D *j-k*, *i-k*, and *i-j* views and 3D "internal" and "external" perspective views of the mesh. In addition, toward the end of the plot file and in front of all *requested* time history plots are five basic time-history plots. The first plot is the total mass time history, and it appears on a single page. The next four are energy time-history plots, which all appear on one page. The quantities plotted are total internal energy, total kinetic energy, total energy loss due to steam condensation, and total internal plus kinetic energy. The basic time-history plots will be generated if the problem end time, `twfin`, is greater than `thdt`, the time interval at which time-history data are written (see discussion of time-history plots below).

2. Geometry Plots

In Sec. III.D, we discuss how to use HMS to just plot the mesh plus all defined walls and obstacles. This is done by setting `maxcyc` in NAMELIST `xput` to a negative number. The set of geometry plots obtained depend on the actual value of the negative number according to Table II.

3. Time-History Plots

Time-history plots of selected solution variables can be requested with the following input array variable:

<code>thp(1,*)</code>	<code>i</code> mesh index (cell number or cell-face number).
<code>thp(2,*)</code>	<code>j</code> mesh index (cell number or cell-face number).
<code>thp(3,*)</code>	<code>k</code> mesh index (cell number or cell-face number).
<code>thp(4,*)</code>	Block number (must be set to 1).
<code>thp(5,*)</code>	Solution variable to be plotted. Choose one of the character strings (enclosed in single quotes) given in Table IV.
<code>thp(6,*)</code>	Gas species name (symbol representing one of the species defined by <code>mat</code> in NAMELIST group <code>xput</code>) enclosed in single quotes. This variable is used only if <code>thp(5,*)</code> has been set to <code>'rsn'</code> , <code>'mf'</code> , or <code>'vf'</code> . Instead of a character string representing the species name, a component number (based on the order in which the species is defined in the <code>mat</code> array) can alternatively be entered here.

Table IV. Solution Variables Available for Plotting

Symbol	Quantity to be plotted
'pn'	Pressure.
'rn'	Mixture density.
'rsn'	Species density.
'cmn'	Cell mass.
'sien'	Specific internal energy.
'un'	i- (x- or r-)velocity component.
'vn'	j- (y- or θ -)velocity component.
'wn'	k- (z-) velocity component.
'tk'	Fluid temperature.
'mf'	Species mass fraction.
'vf'	Species volume fraction.
'vmag'	Velocity magnitude.
'mdotx'	Mass flow rate in i- (x- or r-) direction.
'mdoty'	Mass flow rate in j- (y- or θ -) direction.
'mdotz'	Mass flow rate in k- (z-) direction.
'vdotx'	Volume flow rate in i- (x- or r-) direction.
'vdoty'	Volume flow rate in j- (y- or θ -) direction.
'vdotz'	Volume flow rate in k- (z-) direction.
'delpx'	Pressure difference in i- (x- or r-) direction.
'delpy'	Pressure difference in j- (y- or θ -) direction.
'delpz'	Pressure difference in k- (z-) direction.
'delt'	Time-step size (does not depend on spatial location).
'diffp'	Cell pressure minus an ambient or reference pressure defined by pamb0 (default = 1.01325×10^6 dynes/cm ²) in NAMELIST group xput.
'pstag'	Stagnation pressure ($\rho u ^2/2$) minus pamb0.
'mu'	<i>Effective</i> (molecular and turbulent) viscosity.
'nu'	<i>Effective</i> viscosity divided by fluid density.
'tke'	Turbulent kinetic energy, only valid for if tmodel has been set to 'sgs' or 'ke'.
'eps'	Rate of dissipation of turbulent kinetic energy, only valid if tmodel has been set to 'ke'.

The second dimension in thp array allows more than one definition of time-history plot request, and the first dimension consists of 6 elements that define a particular time-history plot. The variables thp(1,*), thp(2,*), and thp(3,*) are i-, j-, and k-indices, respectively, that define the spatial location where a solution quantity is to be plotted as a function of time. The logical indices can either represent cell number or cell-face number, depending on the quantity being plotted. The reason for this is that in HMS components of velocity, mass flow rate, volume flow rate, and

pressure gradient are defined at cell faces in the corresponding direction, while all scalar quantities such as densities, pressure, temperature, etc., are defined at cell centers. (See Fig. 2 for cell numbering convention.) Consider the following examples:

```
thp(1,1) = 4, 8, 2, 1, 'sien', 0,
thp(1,2) = 3, 4, 5, 1, 'un', 0,
thp(1,3) = 3, 4, 5, 1, 'vn', 0,
thp(1,4) = 3, 4, 5, 1, 'vdotz', 0,
thp(1,5) = 3, 4, 5, 1, 'mf', 'h2o',
thp(1,6) = 4, 8, 2, 1, 'vf', 1,
```

The first thp definition asks for the time-history plot of internal energy at cell (4,8,2). The fifth time-history plot is that of the mass fraction of water vapor at cell (3,4,5). The sixth time-history plot is that of the volume fraction of fluid component 1 (component identification numbers are determined by the order in which the species are listed in the definition of the mat array in NAMELIST group xput) at cell (4,8,2). The second, third, and fourth time-history plots are those of components of vector quantities, and therefore the location should indicate a cell face. For example, the second plot is that of the i velocity component at the i = 3 face of the cell with a j-index of 4 and a k-index of 5. Similarly, the fourth plot is that of the volume flow rate in the z-direction at the k = 5 face of the cell with an i-index of 3 and a j-index of 4.

To conserve space, four time-history plots are grouped on each page or frame. The data to be plotted are first stored in binary files called pth1, pth2, ... (one file for four time histories), then read back in and plotted after the calculation is completed. The frequency at which time-history data are written and subsequently plotted is controlled by the following input variable:

thdt Time (s) interval at which time-history data are written and plotted.
Default = 1×10^{100} .

The time history plots by default have grid lines for both the horizontal (time) and vertical (solution quantity) axes. To turn the grid lines off, the user should set the following input variable:

gline On/off switch for grid lines on time-history plots. The options are:
'on' (default) or 'off'.

Besides time histories of the fluid solution quantities given in Table IV, the user can also plot the surface temperature of a solid heat structure as a function of time. This is done via the following input variable:

```
hthp(1,*)      i-index of fluid cell in contact with the heat structure.
hthp(2,*)      j-index of fluid cell in contact with the heat structure.
hthp(3,*)      k-index of fluid cell in contact with the heat structure.
hthp(4,*)      Block number (must be set to 1).
```

htthp(5,*)	Heat structure type. Choices are: 'slab' Slab heat structure. 'sink' Sink heat structure. 'wall' Wall heat structure.
htthp(6,*)	The side of the fluid cell which coincides with the solid surface whose temperature is to be plotted. This entry is only used if htthp(5,*) has been set to 'slab' or 'wall', because sink structures are assumed to be distributed in the fluid cell. Choices are: 'west' +i side of fluid cell. 'east' -i side of fluid cell. 'south' +j side of fluid cell. 'north' -j side of fluid cell. 'bottom' +k side of fluid cell. 'top' -k side of fluid cell.

Of course heat transfer calculations have to be invoked (by setting ihtflag = 1 in NAMELIST group rheat) for these definitions to be effective. To illustrate the use of htthp definitions, consider the following examples:

```

htthp = 9, 5, 7, 1, 'slab', 'top',
        2, 4, 8, 1, 'sink', 0,
        4, 5, 4, 1, 'wall', 'east',

```

The first heat transfer time-history plot is the temperature at the surface of slab heat structure that coincides with the +k face of the fluid cell (9,5,7). The second plot is that of the surface temperature of the distributed sink heat structure in fluid cell (2,4,8). The third plot will show the time history of the surface temperature of the wall heat structure that is on the +i side of fluid cell (4,5,4).

4. Profile Plots

The user can plot all of the solution variables listed in Table IV (except 'delt') as a function of any one of the three spatial coordinates through an arbitrary region of the mesh. To define the line along which the profile of the quantity of interest is to be plotted, HMS uses the concept of *points*. A line parallel to any of the axes can be defined by two points with the same spatial coordinates in two directions. For example, points with mesh indices (3,4,1) and (3,4,10) define the line going from k = 1 to k = 10, at i = 3 and j = 4. (As described in the following paragraphs, 2D contour plots, and 2D and 3D vector plots also need points to define the region over which the solution quantity is plotted.) Points for plotting purposes can be defined with the following input variable:

pnt(1,*)	i-mesh index.
pnt(2,*)	j-mesh index.
pnt(3,*)	k-mesh index.

pnt (4, *) Block number (must be set to 1).

Note that the first dimension of the array pnt contains four elements to define the point location, and the second dimension identifies the point. Once the points have been defined, the user can specify what the 1D profile plots are via the following input variable:

p1d(1, *) Identification number of the first point (second index of the corresponding pnt definition).
p1d(2, *) Identification number of the second point (second index of the corresponding pnt definition).
p1d(3, *) Solution variable whose 1D profile is to be plotted. Choose one of the symbols (enclosed in single quotes) listed on Table IV, except 'delt'.
p1d(4, *) Gas species name (symbol representing one of the species defined by mat in NAMELIST group xput) enclosed in single quotes. This variable is used only if p1d(3, *) has been set to 'rsn', 'mf', or 'vf'. Instead of a character string representing the species name, a component number (based on the order in which the species is defined in the mat array) can alternatively be entered here.

Note that the first point should not have higher mesh index values than the second point, or an error will result. Consider the following input which illustrates how to use point definitions to define 1D profile plots:

```
pnt(1,1) = 3, 4, 1, 1,  
pnt(1,2) = 3, 4, 10, 1,  
pnt(1,3) = 2, 6, 7, 1,  
pnt(1,4) = 15, 6, 7, 1,  
  
p1d(1,1) = 1, 2, 'pn', 0  
p1d(1,2) = 1, 2, 'rsn', 'h2',  
p1d(1,3) = 3, 4, 'tk', 0,
```

Here four points are defined, with the first two and the last two points being "co-linear" pairs. Therefore the two pairs of points, 1 and 2, and 3 and 4, can be used to define 1D profile plots. The first profile plot is that of the fluid pressure along the line going from point 1 to point 2. The second profile plot is that of the hydrogen species density along the same line. The third profile plot is that of the fluid temperature along the line defined by points 3 and 4.

Similar to time-history plots, the code by default put vertical and horizontal grid lines on 1D profile plots. This can be changed via the following input variable:

glinep1d On/off switch for grid lines on 1D profile plots. The options are 'on' (default) or 'off'.

In problems involving heat transfer, the user can request plotting of the temperature profile in the solid heat structure via the following input variable:

ht1dp(1,*)	i-index of fluid cell in contact with the heat structure.
ht1dp(2,*)	j-index of fluid cell in contact with the heat structure.
ht1dp(3,*)	k-index of fluid cell in contact with the heat structure.
ht1dp(4,*)	Block number (must be set to 1).
ht1dp(5,*)	Heat structure type. Choices are:
	'slab' Slab heat structure.
	'sink' Sink heat structure.
	'wall' Wall heat structure.
ht1dp(6,*)	The side of the fluid cell which is in contact with the heat structure whose temperature profile is to be plotted. This entry is only used if ht1dp(5,*) has been set to 'slab' or 'wall', because sink structures are assumed to be distributed in the fluid cell. Choices are:
	'west' +i side of fluid cell.
	'east' -i side of fluid cell.
	'south' +j side of fluid cell.
	'north' -j side of fluid cell.
	'bottom' +k side of fluid cell.
	'top' -k side of fluid cell.

For slab and wall heat structures, the temperature profile along the entire depth of the structure is plotted. For sink heat structures, only half of the profile is plotted, because it is assumed in the calculation that the temperature distribution is symmetric about the centerline. Consider the following input:

```
ht1dp = 9, 5, 7, 1, 'slab', 'top',
        2, 4, 8, 1, 'sink', 0,
        4, 5, 4, 1, 'wall', 'east',
```

The first heat transfer 1D profile plot is that of the temperature in the slab heat structure that is in contact with the +k face of fluid cell (9,5,7). The second plot is that of the surface temperature profile inside the distributed sink heat structure in fluid cell (2,4,8). The third plot will show the temperature distribution within the wall heat structure that is on the +i side of fluid cell (4,5,4).

5. Contour Plots

It is often useful to plot the contour of a solution quantity on a plane (for example, to identify "hot spots" in certain calculations). 2D contour plots are defined in basically the same way as 1D profile plots. Two points with the same mesh index in one direction (i. e., a pair of so-called "co-planar" points) are used to define the

plane where data are to be taken for the contour plot. Once some (co-planar) points have been defined, contour plots can be requested via the following input variable:

- `c2d(1,*)` Identification number of the first point (second index of the corresponding `pnt` definition).
- `c2d(2,*)` Identification number of the second point (second index of the corresponding `pnt` definition).
- `c2d(3,*)` Solution variable for the 2D contour plot. Choose one of the symbols (enclosed in single quotes) listed on Table IV, except `'delt'`.
- `c2d(4,*)` Gas species name (symbol representing one of the species defined by `mat` in NAMELIST group `xput`) enclosed in single quotes. This variable is used only if `c2d(3,*)` has been set to `'rsn'`, `'mf'`, or `'vf'`. Instead of a character string representing the species name, a component number (based on the order in which the species is defined in the `mat` array) can alternatively be entered here.

Similar to 1D profile plots, the two points specified in `c2d(1,*)` and `c2d(2,*)` should be chosen such that the mesh index values increase in the direction from the first to the second point, or an error will occur. The following input illustrates how to define points and how to specify 2D contour plots:

```
pnt(1,1) = 3, 1, 1, 1,
pnt(1,2) = 3,12, 10, 1,
pnt(1,3) = 2, 1, 7, 1,
pnt(1,4) = 15,12, 7, 1,

c2d(1,1) = 1, 2, 'pn', 0
c2d(1,2) = 1, 2, 'rsn', 'h2',
c2d(1,3) = 3, 4, 'tk', 0,
```

The first two points have the same *i*-index so they define a plane normal to the *i*-direction, at *i* = 3, ranging from *j* = 1 to 12 and from *k* = 1 to 10. The third and fourth points have the same *k*-index so they define a plane normal to the *k*-direction, at *k* = 7, ranging from *i* = 2 to 15 and *j* = 1 to 12. The first contour plot is that of the fluid pressure on a plane defined by points 1 and 2. The second contour plot is that of the hydrogen species density on the same plane. The third profile plot is that of the fluid temperature on the plane defined by points 3 and 4.

The contour lines are plotted in color. HMS uses seven colors; these are, in the order of decreasing contour values, red, yellow, green, cyan, blue, magenta, and black. The user can specify the number of contour levels via the following input variable:

`ncontur` Number of contour levels. Default = 7.

6. Velocity Vector Plots

Using the concept of points, as discussed above for profile and contour plots, the user can also specify velocity vector plots. There are two types of vector plots available. 2D velocity vector plots show the velocity magnitude and direction on a plane defined by two co-planar points. 3D velocity vector plots show the velocity magnitude and direction in a volume, which can be specified by defining two points that locate its diagonal vertices. The length of the shaft, the size of the arrowhead, and the color of the vector are made proportional to the velocity magnitude. Therefore, the vectors in regions with tiny velocities will show up as black dots.

To specify 2D velocity vector plots, the user should define the following:

- `v2d(1,*)` Identification number of the first point (second index of the corresponding pnt definition).
- `v2d(2,*)` Identification number of the second point (second index of the corresponding pnt definition).
- `v2d(3,*)` Flag for frame advance. This option (if set to 0) can be used to overlay the vector plot with the next plot for special presentation. However, it is advised that this flag be set to 1 so that the vector plot will appear by itself on a single frame.

Note that the two points should have the same mesh index in one direction, and the second point should have larger coordinates than the first, or an error will occur. The following is an example showing the use of pnt and v2d to define a 2D velocity vector plot:

```
pnt(1,1) = 1, 1, 2, 1,  
pnt(1,2) = 9, 9, 2, 1,  
  
v2d(1,1) = 1, 2, 1,
```

In this example, velocity vectors will be plotted on the plane $k = 2$, ranging from $i = 1$ to 9, $j = 1$ to 9.

To specify 3D velocity vector plots, the user should define the following:

- `v3d(1,*)` Identification number of the first point (second index of the corresponding pnt definition).
- `v3d(2,*)` Identification number of the second point (second index of the corresponding pnt definition).
- `v3d(3,*)` Flag for frame advance. This option (if set to 0) can be used to overlay the vector plot with the next plot for special presentation. However, it is advised that this flag be set to 1 so that the vector plot will appear by itself on a single frame.
- `v3d(4,*)` Number of the 3D view coordinates definition, i. e., second

index of the corresponding `viewcrds` definition, if `viewcrds` has been specified. If there is no definition of `viewcrds`, this number should be set to 1, in order to use the default 3D viewing coordinates, which are calculated by the code based on the dimensions of the mesh.

Note that the two points specified above should be chosen such that the second point has higher coordinate values than the first, or an error will occur. The following is an example showing the use of `pnt` and `v3d` to define a 3D velocity vector plot:

```
pnt(1,1) = 1, 1, 1, 1,  
pnt(1,2) = 9, 9, 9, 1,  
  
v3d(1,1) = 1, 2, 1, 1,
```

In this example, velocity vectors will be plotted in a volume ranging from $i = 1$ to 9, $j = 1$ to 9, $k = 1$ to 9. Because `v3d(4,1)` is set to 1 (and there is no definition of `viewcrds`), the viewing coordinates for the 3D plot will be determined by the code based on the dimensions of the computational domain.

If there are too many computational cells in the region defined by either `v2d` or `v3d`, the velocity vectors, which by default are plotted at each cell location, will appear cluttered. In this case, the user can use the following input variable to reduce the number of vectors that are plotted:

<code>iinc</code>	Increment in i -index between adjacent vectors. Default = 1.
<code>jinc</code>	Increment in j -index between adjacent vectors. Default = 1.
<code>kinc</code>	Increment in k -index between adjacent vectors. Default = 1.

For example, to plot velocity vectors at every *other* cell location, one should set `iinc`, `jinc`, and `kinc` all to 2.

B. Printed Output

In addition to graphical outputs, HMS provides printed outputs for each calculation. The text file `meshmap`, which is discussed in Sec. III.D, contains cell-number, cell-status (i. e., whether each cell is a fluid or boundary cell, or if its faces are open to flow, etc.) and neighbor-list information. Another printed output file is `cyclinfo`, which lists iteration and time step information at each computational time cycle. This file is discussed in Sec. VII.A. A third printed output file prints out the input NAMELIST variables used in the code, and is called `tape16`.

The main printed output file is `gfout`. In the beginning of the file, the code version number and the date of the run are printed. Then the values of main input variables are listed, followed by tables showing mesh coordinates and cell spacings

(edge-to-edge and center-to-center). The plotting output specifications are then echoed. Next, the calculated fluid velocity (all three components), pressure, and density at each cell are listed, at selected time intervals. This time interval is defined by the following variable in NAMELIST group `xput`:

`prtdt` Time interval (s) between printing of the fluid solution field (all velocity components, pressure, and density) to the output file `gfout`. Default = 1000.

Because in most 3D problems the listing of fluid solution at all cells can be quite long, the default printed output interval has been chosen to be reasonably large (1000 s) to avoid unintended, excessively long output listing.

The `gfout` file also prints out the time, cycle number, and the file name (`gfdn`, where n is an integer) whenever a restart dump file is written. At the end of `gfout`, the total CPU time used is reported, as well as the per cell, per cycle CPU time. Also provided is a table listing, for each of about 16 important subroutines, the CPU seconds used (and its percent contribution to the total CPU time), and the computational speed measured in megaflops. The actual number of subroutines reported depends on the physical models chosen. This information is useful to the user who is interested in the programming and computational aspects of the code.

C. Output to Terminal

Besides graphical and text file outputs, HMS also writes output to a terminal (or the FORTRAN standard output unit that, under a UNIX-type operating system, can be "piped" to a specified file). This output is intended to help the user monitor the calculation as it is being carried out. Any error messages will also be given here. After some banner messages that include identification of code version, the time-step and pressure iteration information is printed. The information given is the same as that in the `cyclinfo` file (see Sec. VII.A); however, instead of printing at every computational time cycle, the terminal output is printed at a selected frequency, which is defined by the following input variable in NAMELIST group `xput`:

`ittyfreq` Number of cycles between printing of time-step and pressure iteration information to the terminal. Default = 20.

When calculation is finished, the code prints to the terminal the same timing information as in the output file `gfout` (discussed at the end of the above section). In addition, it reports the number of restart dump files written and the number of pages (or frames) generated in the plot file `pgf`.

D. Restart

Because HMS is capable of solving complex, large problems, it may take a large amount of computer time to finish a problem. Therefore, the code provide a

restart capability so that a long calculation can be divided into a series of shorter runs. A restart dump file is always produced at the end of each run. However, the user can specify that additional restart files be written at selected time intervals. This is done via the following input variable in NAMELIST group `xput`:

`tddt` Time interval (s) at which restart dump files are written.
Default = 10.

Therefore, one restart dump file, called `gfd1`, will be written if the problem end time (specified by `twfin` in NAMELIST group `xput`) is less than `tddt`. If `twfin` is larger than `tddt`, then `gfd1` will be the restart file written at time `tddt`. The next restart files, `gfd2`, `gfd3`, etc., will be written at times that are multiples of `tddt`. Hence, the restart file that contain the final solution will have the name `gfdn`, where n is the total number of restart files produced.

To specify that a run is to begin from the solution stored in a restart dump file, the user should define the following variable in NAMELIST group `xput`:

`nrsdump` Number that appears in the name of the restart dump file
that is to be read in. For example:
0 New problem, not a restart run (default).
1 Read from restart file `gfd1`.
2 Read from restart file `gfd2`.

X. CODE MODIFICATION FOR SPECIAL USE

If a user wants to define new gas species, change physical models, or modify numerical schemes, the changes must be made to the source code directly. (At some future time, "user subroutines" could be provided to make this task easier.) In the following, we give an example of how to modify the code so that it can simulate a large-break blowdown experiment.

One of the assessment problems discussed in Ref. 2 is simulation of the HDR T31.5 experiment. The HDR containment building is a large facility with 11 000-m³ free volume in 72 interconnected compartments. In the experiment, a steam-water mixture was first injected into the containment to simulate a large-break blowdown of a pressure vessel, and then superheated steam was injected that was followed by a release of helium-hydrogen light gas. Because HMS is a single-phase fluid code, it is unable to model the two-phase critical flow mixture entering the containment. Therefore, we have developed a blowdown jet-expansion model to extract the correct mass flow rate of steam source into the containment. The steam-water mixture either partially flashes or condenses as it enters the containment, depending on the thermodynamic conditions. See Ref. 2 for the details of this model.

Figures 8a-e show a listing of the code changes in "update" format. (The actual software maintenance tool used is called HISTORIAN, a commercial product.) There are four major changes:

1. Replace the 'c' species in the code with a new 'lg' species to simulate the "light gas," which is made up of 15% hydrogen and 85% helium by volume.
2. Read from the file `sortam` (FORTRAN I/O Unit 98) the blowdown two-phase mixture source condition and flow rate (as well as those during the superheated steam and light gas injection phases). Add the source in the appropriate conservation equations.
3. Implement the two-phase jet expansion model. The entropy-temperature-pressure data (from a steam table) are read in from the file `inpsor` (Unit 97). Basically, the specific entropy of the two-phase jet before exiting the blowdown pipe is calculated from the measured temperature, pressure and quality upstream of the pipe exit. Then the jet is expanded isentropically into the blowdown room of the containment building, which is at a lower pressure. The amount of flashing or condensation that occurs in the expanded jet is governed by its quality, which can be determined from the liquid and vapor specific entropies at the lower containment pressure via the "lever rule." See Ref. 2 for model details.
4. Write additional information to the terminal (Unit 6) and to files (Units 17, 99). This printed information includes heat transfer surface types and numbers, source rates after jet expansion, cumulative mass of light gas injected, and pressure solution at a monitor location.

XI. GENERAL USER GUIDANCE

Before setting up a complex problem, it is always helpful for the user to run some similar but simpler problems first. Doing so will allow the user to quickly gain insight to the problem and verify the majority of the input deck. Common ways of simplifying a problem include the following:

1. Use a coarser mesh. In a heat transfer problem, also coarsen the nodalization in all heat conducting solids.
2. Use default physical models, which are normally the simplest options.
3. Reduce the problem time, i. e., `twfin`.
4. Relax the pressure iteration error criterion, i. e., increase `epsi0`.

Through a series of runs in which complexities are added successively, the user will become more familiar with the problem, which should help in analyzing results from the final calculations.

For large-scale, long-running problems, it is advisable to use the restart capability of the code to break the problem into a series of shorter runs. The user should check the calculations with extensive graphical display of the solution both as a function of time and space. Note that the code can be used as a graphics

```

*ident modhdr
*/ HISTORIAN mod file for the HDR problem.
*/ -----
*/ Count and write out heat transfer surfaces
*i setheat.48
  real aconslab(40),aconwall(40),ashelk(40)
*i setheat.171
  ksink = isink
  asinkk = 0.0
*i setheat.183
  asinkk = asinkk + areasink(isink)*2.0
*i setheat.186
  nsinkk = isink - ksink
  write (6,1) k, nsinkk, asinkk
  1 format (' k=',i3,' Total no. sinks=',i4,' Total exposed area=',
    &lpe15.6)
*i setheat.496
  write (6,2) nhtsinks, nhtwalls, nhtslabs
  2 format (' no. sinks=',i6,' no. walls=',i6,' no. slabs=',i6)
  do 6001 islab=1,nhtslabs
    m = mslab(islab)
    k = int(m/imaxjmax) + 1
    j = int((m-(k-1)*imaxjmax)/imax) + 1
    i = m - (j-1)*imax - (k-1)*imaxjmax
    aconslab(k) = aconslab(k)+areaslab(islab)
6001 continue
  do 6002 iw1=1,nhtwalls
    m1 = mwall(1,iw1)
    m2 = mwall(2,iw1)
    k1 = int(m1/imaxjmax) + 1
    k2 = int(m2/imaxjmax) + 1
    if (matwall(iw1).eq. 1) then ! concrete wall
      aconwall(k1) = aconwall(k1)+areawall(iw1)
      aconwall(k2) = aconwall(k2)+areawall(iw1)
    else if (matwall(iw1).eq.2) then ! steel wall (shell)
      ashelk(k1) = ashelk(k1) + areawall(iw1)
      ashelk(k2) = ashelk(k2) + areawall(iw1)
    end if
6002 continue
    sumac = 0.0
    do 6003 k = 2, km1
      aconck = aconwall(k) + aconslab(k)
      sumac = sumac+aconck
      write (6,*) k, sumac
      write (6,4) k, aconwall(k),aconslab(k),aconck,ashelk(k)
  4 format (' k=',i3,t8,'Concrete wall area=',lpe15.6,t50,'Concrete sl
    &ab area=',lpe15.6/t8,'Total concrete area=',lpe13.6/t8,'Shell area
    &=',lpe15.6)
6003 continue
*i setheat.140
  do 6000 k = 2, km1
    aconslab(k) = 0.0
    aconwall(k) = 0.0
    ashelk(k) = 0.0
6000 continue
    numsw = 0
    numcw = 0
    do 6102 iw=1,iw1
      m1 = mwall(1,iw)
      m2 = mwall(2,iw)
      k1 = int(m1/imaxjmax) + 1
      k2 = int(m2/imaxjmax) + 1
      j1 = int((m-(k1-1)*imaxjmax)/imax) + 1
      i1 = m - (j1-1)*imax - (k1-1)*imaxjmax
      if (matwall(iw).eq. 1) then ! concrete wall
        aconwall(k1) = aconwall(k1)+areawall(iw)
        aconwall(k2) = aconwall(k2)+areawall(iw)
        numcw = numcw + 2
      else if (matwall(iw).eq.2) then ! steel wall (shell)
        if (k1.eq.24.and.k2.eq.25) goto 6102 ! exclude impingement plate
        ashelk(k1) = ashelk(k1) + areawall(iw)
        ashelk(k2) = ashelk(k2) + areawall(iw)
        numsw = numsw + 2
      end if
6102 continue
    write (6,*) numcw, numsw
    sumashel = 0.0

```

Fig. 8a. Beginning of code modification listing for the HDR problem.

```

        sumacon = 0.0
        do 6103 k = 2, km1
            sumashel = sumashel + ashelk(k)
            sumacon = sumacon + aconwall(k)
            write (6,5) k, sumashel, sumacon
5 format (' k=',i4,' cumulative shell area=',1pe15.6
        &      /' cumulative con. wall area=',1pe15.6)
            write (6,4) k, aconwall(k),aconslab(k),aconwall(k),ashelk(k)
6103 continue
*/ -----
*/
*/ Write total mass to file fort.96
*/ i outhist.193
        write (96,*) t, tmass
*/ -----
*/
*/ Change species carbon (C12) to light gas (LG).
*/ d gprop.15
        &      , lho2 , lno , lnh , lhno , llg
*/ d gprop.19
        &      , llho2 , llno , llnh , llhno , lllg
*/ d rinput.225
        if ( mat(i) .eq. 'lg' ) llg = i
*/ d setcode.306,314
        llLG = i
        lLG = 0
        gasname(i) = 'Light Gas LG'
        gname (i) = 'LG'
        hf0g (i) = 0.0000000000E+00 ! Light Gas LG
        wmoieg (i) = 3.705000 ! Light Gas LG

        coef1 ( 0, i) = 0.0000000000E+00 ! Light Gas LG
        coef1 ( 1, i) = 3.6830000000E+07 ! Light Gas LG
*/ d setcode.74,75
        data (gasprop(i, 1), i = 1,5 ) /
        & 3.7050, 1.532, 3.683000000e7, 0.0, 0.0 / ! LG
*/
*/ ident mblowdn
*/ Special mod set to implement the blowdown model (based on isentropic
*/ expansion) for the HDR problem.
*/ Also write out the quality of the expanded mixture and the total
*/ steam inventory in the containment vs. time from 0 to 50 sec, and
*/ from 1255 to 2888 sec.
*/ af, ,zzzend
*/ cd csource

        common / csource /
        & msor,
        & sore, sorm(3),
        & rif, tif, vif, volif,
        & gammaml(3),csv(3),
        & pb(50),tb(50),hfa(50),hfga(50),sfa(50),sfga(50)
*/ i gfdrive.23
*/ ca csource

c Read csv and gammaml special for this subroutine

        data (csv(i),i=1,3) / 7.168017241e6, 1.404069571e7, 3.6830000e7 /
        data (gammaml(i),i=1,3) / 0.4, 0.329, 0.532 /
c Define the Computational Cell that receives the blowdown mass,
c momentum, and energy

        data isor,jsor,ksor / 07, 11, 24 /
*/ i gfdrive.25

c Read in Steam Tables from file INPSOR

        open (97,file='inpsor')
        do 14 i=1,50
14 read(97,15) pb(i),tb(i),hfa(i),hfga(i),sfa(i),sfga(i)
15 format(6f10.5)

        msor = isor + (jsor-1)*imax + (ksor-1)*imaxjmax
        msortop = isor + (jsor-1)*imax + (ksor+1-1)*imaxjmax

c Read Initial Values from the Source Tape

```

Fig. 8b. Continuation of code modification listing for the HDR problem.

```

open(98,file='sortam')

read(98,2) dum
read(98,2) dum
2 format(a8)

read(98,3) time1,pres1,th2o1,frh2o1,qual1,frh21
read(98,3) time2,pres2,th2o2,frh2o2,qual2,frh22
3 format(6f13.6)

*i gfdrive.35

c Read Values from the Source Tape so that t is bounded

time = t

4 continue
if((time.lt.time2).and.(time.ge.time1)) go to 5
time1=time2
th2o1=th2o2
frh2o1=frh2o2
pres1=pres2
qual1=qual2
frh21=frh22
read(98,3) time2,pres2,th2o2,frh2o2,qual2,frh22
go to 4
5 continue

c Time interpolate SORTAM (bounded values) with current time t

tot=(time-time1)/(time2-time1)
th2o=th2o1+tot*(th2o2-th2o1)
frh2o=1000.0*(frh2o1+tot*(frh2o2-frh2o1))
pres=pres1+tot*(pres2-pres1)
qual=qual1+tot*(qual2-qual1)
frh2=1000.0*(frh21+tot*(frh22-frh21))
tfr=frh2o+frh2

c Check to ensure blowdown source (experimental data) is positive

if(tfr.gt.0.0) go to 6
xh2o=0.0
xh2=0.0
tif=th2o
rif=0.0
vif=0.0
go to 7
6 continue

c
c Interpolate Steam Tables for Pipe end (blowdown) pressure.
c
if ( t .gt. 1249.0 ) then
qual = 0.5
end if

c pres = pipe blowdown pressure

do 41 j=1,49
if((pres.ge.pb(j)).and.(pres.lt.pb(j+1)))go to 42
41 continue
stop10
42 continue

c
pratio=(pres-pb(j))/(pb(j+1)-pb(j))

c
sf=sfa(j)+pratio*(sfa(j+1)-sfa(j))
sfg=sfga(j)+pratio*(sfga(j+1)-sfga(j))
s=sf+qual*sfg

c
c pre = containment reference pressure

pre=pn(msortop)*1.0e-06

c
c Check to insure blowdown pressure (experimental data) is greater
c than the containment reference pressure.
c
c Also skip isentropic expansion calculation for light gas

```

Fig. 8c. Continuation of code modification listing for the HDR problem.


```

c injection phase.
      if(pres.ge.pre .and. t.lt.2153.9) go to 49
      tif=th2o
      quality=qual
      go to 53
49 continue
c
c Interpolate Steam Tables for Containment (reference) pressure
      do 51 j=1,49
      if((pre.ge.pb(j)).and.(pre.lt.pb(j+1)))go to 52
51 continue
      stop11
52 continue
c
      pratio=(pre-pb(j))/(pb(j+1)-pb(j))
c
      sf=sfa(j)+pratio*(sfa(j+1)-sfa(j))
      sfg=sfga(j)+pratio*(sfga(j+1)-sfga(j))
c
c Determine the saturation temperature from the reference pressure, pre,
c using the Clausius-Clapeyron Equation
      tif=-2263.0/(0.434*log(pre)-6.064)
c
c Expand blowdown mass flow rate (isentropically) to a quality
c based on the containment reference pressure
      quality=(s-sf)/sfg
c
53 continue
c
c tfr = total mass flow rate entering containment from steam and h2
      tfr=quality*frh2o+frh2
c
c define mass fractions of steam and hydrogen
      xh2o=quality*frh2o/tfr
      xh2=frh2/tfr
c
c const = mass flux = mass flow rate / flow area
      ij = isor + (jsor-1)*imax
      const=tfr/(dely(ij)*delx(ij))
c
c compute mass fractions times Cv and Cv times gamma-1
      cvbar=xh2o*csv(2)+xh2*csv(3)
      xcg=xh2o*csv(2)*gammaml(2)+xh2*csv(3)*gammaml(3)
c
c define blowdown inflow values, vif=velocity, rif=density,
c volif=volumetric flow rate, and siegif=specific internal energy
      siegif=cvbar*tif
      rif=pn(msor)/(tif*xcg)
      vif=const/rif
      volif=tfr/rif
7 continue
c
c compute the source terms for the energy and mass transport equations
      sorm(1)=0.0
      sorm(2)=xh2o*tfr
      sorm(3)=xh2*tfr
      sore=(sorm(2)*csv(2)+sorm(3)*csv(3))*tif
c
c Write out some info.
c
      if (t.le.50. .or. (t.gt.1255. .and. t.lt.2148.)) then
      tmh2o = 0.0
      do 54 m = m222, mrrr
      m2 = m + (lh2o-1)*mst
      tmh2o = tmh2o + cvmgz(xsn(m2)*vol(m), 0.0, and(mflag(m),mskbo))
54 continue
      write (99,55) t, quality, tmh2o, sorm(2), pres, tif

```

Fig. 8d. Continuation of code modification listing for the HDR problem.


```

55 format (f9.3,1pe13.6,1pe13.6,1pe13.6,1pe13.6,1pe13.6)
end if
if ( t.gt.2154. .and. t.lt.2894. ) then
  tmlg = 0.0
  do 56 m = m222, mrrr
    m2 = m + (1lg-1)*mst
    tmlg = tmlg + cvmgz(rsn(m2)*vol(m), 0.0, and(mflag(m),mskbo))
56 continue
  write (99,55) t, tmlg, sorm(3), tif
end if

*b phasea.86
*ca csource

*i phasea.117

c add volumetric flow rate, volif, to the computational cell receiving
c the blowdown source. Note: since the blowdown source is received
c on the negative (bottom) side of the cell, it must be added with
c a minus sign

      divun(msor) = divun(msor) - volif

*b phasec.81
*ca csource

*i phasec.134

c add mass flow rate, sorm(i), to the computational cell receiving
c the blowdown source. Note: since the blowdown source is received
c on the negative (bottom) side of the cell, it must be added with
c a minus sign

      divru(msor) = divru(msor) - sorm(n)

*i phasec.173

c add energy flow rate, sore, to the computational cell receiving
c the blowdown source. Note: since the blowdown source is received
c on the negative (bottom) side of the cell, it must be added with
c a minus sign

      divreu(msor) = divreu(msor) - sore

*b phasecvl.81
*ca csource

*i phasecvl.168

c add mass flow rate, sorm(i), to the computational cell receiving
c the blowdown source. Note: since the blowdown source is received
c on the negative (bottom) side of the cell, it must be added with
c a minus sign

      divru(msor) = divru(msor) - sorm(n)

*i phasecvl.231

c add energy flow rate, sore, to the computational cell receiving
c the blowdown source. Note: since the blowdown source is received
c on the negative (bottom) side of the cell, it must be added with
c a minus sign

      divreu(msor) = divreu(msor) - sore

*/
*ident mod40
*b update.15
*ca com2
  data monpt / 6717 /
*d update.25,29
  &' TIME CYCLE PITER DELT DMAX EPSI PN')
  if ( mod(cycle, ittyfreq) .eq. 0 )
    & write ( 6, 220 ) t, cycle, iter, delt, dmax, epsi, pn(monpt)
    write (17, 220 ) t, cycle, iter, delt, dmax, epsi, pn(monpt)
220 format ( 1pe10.3, i9, i7, 1pe10.3, 1pe10.3, 1pe10.3, 1pe16.9 )

```

Fig. 8e. End of code modification listing for the HDR problem.

postprocessor for the data in a restart dump file, which is always written at the end of a run.

In case a calculation gives unphysical results, the user should review the input deck. Most problems arise from incorrectly specifying initial and boundary conditions. The user should ask herself or himself the following:

1. Have all fluid cells in the entire domain been given the correct initial conditions via defining the `gasdef` array appropriately?
2. Are there any open boundaries across which inflow can occur? If so, do the boundary cells have the appropriate fluid conditions defined via `gasdef`?
3. Does reducing Δt (by decreasing `delt0` and/or `deltmax`) give the same unphysical results?
4. Does tightening the pressure iteration convergence criterion (i. e., reducing `epsi0`) give the same unphysical results?

Finally, the user should note that HMS always solves the time-dependent conservation equations. A *steady-state* calculation option is not available in the code. However, this should not prevent the user from solving steady-state (time-independent) problems with the code. The initial conditions in such a calculation will constitute an initial guess, and each time cycle will represent an iteration toward the steady solution. Time-history plots of the relevant solution quantities will indicate if and when steady state is attained. (Even in codes which provide the steady state calculation option, "false time stepping" is sometimes used on a particular equation to improve convergence if there is knowledge of the time scale over which the variable changes. In some high-speed compressible or multiphase flow problems, it may be necessary to solve the steady-state problem as a transient one, with small time steps in the beginning.)

XII. SAMPLE PROBLEMS

(All figures showing sample input decks and output plots, Figs. 9 to 35, are placed at the end of this section.)

A. Von Karman Vortex Street

The von Karman vortex street problem is discussed in the Assessment Manual (Ref. 2). Here, we show the input listing and some of the graphical outputs obtained for one case of the calculations. This is the case of flow past a rectangular block at a Reynolds number of 30. This Reynolds number is less than the critical value (around 40) above which vortex streets are formed behind the flow obstacle.

Figures 9a-b list the input used in the calculation. Note that the van Leer advection scheme is chosen to reduce numerical diffusion errors. The positive and negative j mesh boundaries (which model the wind tunnel wall, if this calculation is compared to an experiment) are free-slip. However, surfaces of the obstacle are

specified with no-slip velocity boundary conditions (`nslipdef`). Note also definition of a perturbed inlet velocity during the initial one second. Figure 10 shows the mesh used, and Figs. 11 to 14 are selected output plots that show some of the computed results.

B. Hydrogen Burn

This sample problem illustrates how to set up a premixed hydrogen-air combustion problem. Figures 15a-b show the input listing for this problem. Note that no boundary conditions are specified here, so the default free-slip wall condition is used for all boundaries. The initial condition is a stagnant, premixed hydrogen-air mixture (10% hydrogen, 19% oxygen, and 71% nitrogen by volume) at atmospheric pressure and 300 K. The 2D computational domain is 100 cm long (x -direction, discretized into 20 cells) and 250 cm high (y -direction, 50 cells). All computational cells are uniform in size, 5 x 5 cm. Three compartments, which are defined by a `walls` and a `mobs` input specification, are modeled, as can be seen in Fig. 16. The `walls` definition specifies a thin partition (which has no thickness mathematically) between the upper and middle compartments. This partition extends from $x = 0$ to 75 cm, leaving an opening of 25 cm between the two compartments. The `mobs` definition specifies a 5-cm-thick physical barrier to separate the middle and lower compartments. This barrier extends from $x = 25$ to 100 cm, so that fluid flow between the two compartments is only allowed between x -coordinates of 0 and 25 cm.

It is assumed that the mixture is ignited at a location on the left side of the domain ($x = 0$) and 50 cm from the bottom. Ignition by a spark is considered. The electrical energy deposited instantaneously by the spark is simulated by defining the initial temperature of the cell where ignition occurs to 1000 K. The problem is then run to a time of 5 s. The hydrogen burn model is, of course, activated (`iburn = 1`). Note that the burn model simply calculates the reaction rate and the subsequent change in species densities and increase in energy. No ad hoc model for the flame propagation is needed. Progression of the flame front is determined by the interaction between chemical kinetics and fluid dynamics (diffusion, convection, turbulence, etc.).

Diffusion of mass, energy, and momentum, as well as the algebraic turbulence model, are turned on. Gravity is zero in all directions by default. In this problem, because the positive y -direction is vertically upward, earth gravity can be activated by setting `gy = -980 cm/s2`. Other code options that are turned on include specification of graphical outputs for pressure, temperature, H_2 , and H_2O volume fraction time histories, velocity vectors, and temperature contours. Figures 16 to 25 show some of the plots produced. Because the flame propagation speed is relatively low (on the order of 1 m/s), the hydrogen burn can be characterized as a slow deflagration.

C. HDR Containment Facility Test T31.5

This problem is discussed in the Assessment Manual (Ref. 2). The particular calculation presented here simulates the first 60 s of the HDR test T31.5. In the experiment, a large-break blowdown of the pressure vessel injected about 3×10^7 kg of steam-water mixture into the containment during the first 50 s. Superheated steam and a helium-hydrogen mixture (termed "light gas") were also injected during the latter part of the transient. The code has to be modified to read in the blowdown source and implement a two-phase jet expansion model for this calculation (see Sec. X). Figures 26a-h show the input listing for the run. Figures 27 to 35 show some of the graphical outputs that illustrate the calculated results.

```

Flow past block, Re = 30                                HMS-93
TSA-8 Lam
HMS-vk.block
NOTES: Obstacle (2cm tall, 1cm thick) in channel 24cm tall, 20 cm long.
       Obstacle to channel height ratio is 1/12
       Variable mesh spacing: 2 zones in x and 3 zones in y.
       Number of real cells = 44 x 44 x 1.
-----
$Xput
  nrsdump = 0,
  ifvl = 1,           ; Turn on van Leer advection scheme
  autot = 1.0,
  delt0 = 0.025,
  deltmin = 1.000e-04,
  deltmax = 1.000e-00,
  epsi0 = 1.000e-05,
  epsimax = 1.000e-05,
  epsimin = 1.000e-05,
  gz = -000.0,
  iobpl = 1,
  itdowndt = 500,
  itupdt = 500,
  itmax = 500,
  lpr = 1,
  maxcyc = 900056,
  ittyfreq = 100,
  nu = 0.153,
  muoption = 1,
  pltdt = 50.0,
  prtdt = 40.0,
  twfin = 40.0,
  tddt = 500.0,
  velmx = 1.5,
  ibb = 1,
  ibn = 1,
  ibs = 1,
  ibw = 1,
  ibe = 1,
  ibt = 1,

  idiffmom = 1,

  nslipdef(1,1) = 5, 9, 19, 19, 1, 2, 1, 'both',
  nslipdef(1,2) = 5, 9, 27, 27, 1, 2, 1, 'both',
  nslipdef(1,3) = 5, 5, 19, 27, 1, 2, 1, 'both',
  nslipdef(1,4) = 9, 9, 19, 27, 1, 2, 1, 'both',

  vvalue = 2.3, 1.2, ; Inlet velocity, Perturbed velocity

  vbc(1,1) = 01, 01, 1, 23, 1, 02, 1, 1, 0.0, 2.5e+03, ; Inlet, upper half
  vbc(1,2) = 01, 01, 23, 45, 01, 02, 1, 2, 0.0, 1.0e+00, ; Perturb., lower half
  vbc(1,3) = 01, 01, 23, 45, 01, 02, 1, 1, 1.0, 2.5e+03, ; Inlet, lower half

  pbc(1,1) = 45, 45, 1, 45, 01, 02, 1, 00.0, 2.5e+03,

  mat = 'air',

  gasdef(1,1) = 1, 45, 1, 45, 1, 2, 1,
               1.0132500000e6, 300.00, 1, 0., 0.,
               'air', 1.00000,

  gasdef(1,2) = 0, 1, 1, 45, 1, 2, 1,
               1.0132500000e6, 300.00, 1, 0., 2.5e+03,
               'air', 1.00000,

  gasdef(1,3) = 45, 46, 1, 45, 1, 2, 1,
               1.0132500000e6, 300.00, 1, 0., 2.5e+03,
               'air', 1.00000,

  mobs = 5, 9, 19, 27, 1, 2, 1,

$end
-----
M E S H
-----
$meshgn

  iblock = 1,

```

Fig. 9a. First half of the input listing for the von Karman sample problem.

```

nkx = 02,
xl(1) = 0.0, xc(1) = 0.0, nxl(1) = 0, nxr(1) = 32, dymn(1) = 1000.0,
xl(2) = 8.0, xc(2) = 8.0, nxl(2) = 0, nxr(2) = 12, dymn(2) = 0.3,
xl(3) = 20.0,

nky = 03,
yl(1) = 0.0, yc(1) = 9.0, nyl(1) = 10, nyx(1) = 0, dymn(1) = 0.3,
yl(2) = 9.0, yc(2) = 9.0, nyl(2) = 0, nyx(2) = 24, dymn(2) = 9999.9,
yl(3) = 15.0, yc(3) = 15.0, nyl(3) = 0, nyx(3) = 10, dymn(3) = 0.3,
yl(4) = 24.,

nkz = 01,
zl(1) = 0.0, zc(1) = 0.0, nzl(1) = 0, nzx(1) = 01, dymn(1) = 1000.0,
zl(2) = 10.0,

$end
-----
G R A P H I C S
-----
$graphic

igrd = 1,

thdt = 0.05,

thp(1,1) = 18, 3, 2, 1, 'vn', 0,
thp(1,2) = 18, 6, 2, 1, 'vn', 0,
thp(1,3) = 36, 3, 2, 1, 'vn', 0,
thp(1,4) = 36, 6, 2, 1, 'vn', 0,
thp(1,5) = 41, 3, 2, 1, 'vn', 0,
thp(1,6) = 41, 6, 2, 1, 'vn', 0,
thp(1,7) = 45, 3, 2, 1, 'vn', 0,
thp(1,8) = 45, 6, 2, 1, 'vn', 0,
thp(1,9) = 17, 4, 2, 1, 'un', 0,
thp(1,10) = 17, 6, 2, 1, 'un', 0,
thp(1,11) = 17, 24, 2, 1, 'un', 0,
thp(1,12) = 37, 4, 2, 1, 'un', 0,
thp(1,13) = 37, 6, 2, 1, 'un', 0,
thp(1,14) = 37, 24, 2, 1, 'un', 0,
thp(1,15) = 42, 6, 2, 1, 'un', 0,
thp(1,16) = 42, 24, 2, 1, 'un', 0,

pnt(1,1) = 1, 1, 2, 1,
pnt(1,2) = 45, 45, 2, 1,
pnt(1,3) = 9, 1, 2, 1,
pnt(1,4) = 9, 45, 2, 1,
pnt(1,5) = 21, 1, 2, 1,
pnt(1,6) = 21, 45, 2, 1,
pnt(1,7) = 37, 1, 2, 1,
pnt(1,8) = 37, 45, 2, 1,
pnt(1,9) = 42, 1, 2, 1,
pnt(1,10) = 42, 45, 2, 1,
pnt(1,11) = 45, 1, 2, 1,
pnt(1,12) = 45, 45, 2, 1,
          9, 15, 2, 1,
          37, 31, 2, 1,

v2d(1,1) = 1, 2, 1,
          13, 14, 1,

iinc = 2,
jinc = 2,

$end
-----
P A R T I C L E S
-----
$parts

$end
$reheat

$end
$special

$end

```

Fig. 9b. Second half of the input listing for the von Karman sample problem.

Fig. 10. Mesh used for the sample von Karman vortex street problem. The automatic mesh generator has been used to define two zones in the x -direction and three zones in the y -direction. This is a basic plot generated by automatically by the code.

Fig. 11. Velocity vector plot at 40 s over the entire computational domain. Note the nearly symmetric velocity distribution about the centerline. This plot is produced by the first line of the following input (definition of points 1, 2, 13, and 14 is not shown):

```
v2d(1,1) = 1, 2, 1,
           13,14, 1,
```

Fig. 12. Velocity vector plot at 40 s over a region just behind the block. Note the pair of weak, counter-rotating eddies above and below the centerline. This plot is produced by the second line of the following input (definition of points 1, 2, 13, and 14 is not shown):

$$v2d(1,1) = 1, 2, 1, \\ 13, 14, 1,$$

Frame No. 11

12 DEC 93 20:31:52 GASFLOW v1.0

Fig. 13. Time history plots of the normal velocity component at four locations. Note the gradual decay of the oscillation amplitudes, which indicates steady state is being approached. These plots are produced by the following input lines:

thp(1,1)	=	18, 3, 2, 1, 'vn', 0,
thp(1,2)	=	18, 6, 2, 1, 'vn', 0,
thp(1,3)	=	36, 3, 2, 1, 'vn', 0,
thp(1,4)	=	36, 6, 2, 1, 'vn', 0,

Frame No. 13

12 DEC 93 20:31:52 GASFLOW v1.0

Fig. 14. Time-history plots of the streamwise velocity component at four locations. Note that the oscillation amplitudes are decaying, which indicate steady state is being approached. These plots are produced by the following input lines:

```

thp(1,9) = 17, 4, 2, 1, 'un', 0,
thp(1,10) = 17, 6, 2, 1, 'un', 0,
thp(1,11) = 17, 24, 2, 1, 'un', 0,
thp(1,12) = 37, 4, 2, 1, 'un', 0,

```

```

3-Compartment H2 Burn Test Problem
N6 jrt
HMS-Burn
HMS-93
NOTES: 2-D domain 100 cm x 250 cm with deltax = deltax = deltax = 5 cm
Number of fluid cells = 20 x 50 x 1 for the
coordinate dimension x, y, and z, respectively.

Two horizontal walls separating computational domain.
Top wall defined by "walls" definition.
Bottom wall defined by "mobs" definition.

-----
$Xput

    iburn = 1,                ; Turn on hydrogen combustion

    idiffmom = 1,             ; Momentum diffusion on
    idiffme = 1,             ; Mass and energy diffusion on
    nu = 0.150 ,
    prandtl = 0.70,
    schmidt = 0.25,

    tmodel = 'alg',          ; Algebraic turbulence model

    autot = 1.0,
    delt0 = 00.01000 ,
    deltmin = 1.000e-06,
    deltmax = 1.000e-02,

    epsi0 = 1.000e-05,
    epsimax = 1.000e-05,
    epsimin = 1.000e-05,

    gy = -980.0,             ; Gravity in -y direction

    itdowndt = 100,
    itupdt = 100,
    itmax = 500,

    lpr = 1,
    ittyfreq = 100,
    pltddt = 0.50,
    prtdt = 500.0,
    tddt = 100.0,
    velmx = 1.5,

    maxcyc = 10000,
    twfin = 5.000,           ; Total problem time 5 seconds

    mobs = 6, 21, 20, 21, 1, 2, 1, 0, ; Obstacle definition

    walls = 1, 16, 36, 36, 1, 2, 1, 0, ; Wall definition

    mat = 'h2', 'o2', 'n2', 'h2o',

; Initial Condition Throghout domain:
    gasdef(1,1) = 1, 21, 1, 51, 1, 2, 1,
                1.013500e6, 300.0, 2, 0., 0.,
                'h2', 0.1000000, 'o2', 0.1900000, 'n2', 0.7100000,

; Initial Condition at Ignition Location:
    gasdef(1,2) = 1, 02, 11, 12, 1, 2, 1,
                1.013500e6, 1000.0, 2, 0., 0.,
                'h2', 0.1000000, 'o2', 0.1900000, 'n2', 0.7100000,

$end

-----
M E S H
-----
$meshgn

    iblock = 1,

    nkx=1,
    xl(1)= 0.0, xc(1) = 0.000 , nxl(1)= 0, nxr(1)=20 , dxmn(1)= 9999.,
    xl(2)= 100.0,

    nky=1,
    yl(1)= 0., yc(1) = 0.000 , nyl(1)= 0, nyx(1)=50 , dymn(1)= 9999.,
    yl(2)= 250.0,

```

Fig. 15a. First part of the input listing for the hydrogen burn sample problem.

```

nkz=1,
  z1(1)=0.0000, zc(1) = 0.0000, nzl(1)= 0, nzt(1)= 1 , dzmn(1)= 9999.,
  z1(2)= 5.,

$end
-----
      G R A P H I C S
-----
$grafic

  thdt = 0.050,

  thp = 10, 25, 2, 1, 'pn', 0,          ; pressure time history
        10, 25, 2, 1, 'tk', 0,          ; temperature time history
        10, 25, 2, 1, 'vf', 'h2',       ; H2 volume fraction t. h.
        10, 25, 2, 1, 'vf', 'h2o',     ; H2O vol. frac. t. h.

; Define two points that would cover the entire physical x-y domain.
pnt(1, 1) = 1, 1, 2, 1,
           21,51, 2, 1,

v2d      = 1,02, 1,          ; velocity vector plot on plane
           ; defined by points 1 and 2.

c2d      = 1,02, 'tk' , 0,    ; temperature contour plot on plane
           ; defined by points 1 and 2.

$end
-----
      P A R T I C L E S
-----
$parts
$end
-----
      H E A T   T R A N S F E R   &   C O N D E N S A T I O N
-----
$heat
$end
-----
      S P E C I A L
-----
$special
$end

```

Fig. 15b. Second part of the input listing for the hydrogen burn sample problem.

Fig. 16. Temperature contours at 0.5 s. Note the rising hot plume.

Fig. 17. Velocity vectors at 0.5 s showing upward motion near the ignition location.

Fig. 18. Temperature contours at 1.0 s. The hydrogen flame is split between the lower and middle compartment.

Fig. 19. Velocity vectors at 1.0 s, showing fluid motion spreading to other locations from the ignition point.

Fig. 20. Temperature contours at 2.0 s. The flame has spread to most areas, except in regions below the bottom and top compartments.

Fig. 21. Velocity vectors at 2.0 s.

Fig. 22. Temperature contours at 3.0 s. The hydrogen burn is complete, as indicated by the relatively small temperature gradients throughout the domain. (The difference between maximum and minimum temperatures is roughly 400 K, compared to about 1100 K at 2.0 s.)

Fig. 23. Velocity vectors at 3.0 s.

Frame No. 29

12 DEC 93 20:41:43 GASFLOW v1.0

Fig. 24. Basic time-history plots showing four total energy quantities. The lower plot at the left represents energy change due to steam condensation, which is shown to be 0 for this problem. These plots are generated automatically by HMS.

3-Compartment H2 Burn Test Problem

HMS-93

Frame No. 31

12 DEC 93 20:41:43 GASFLOW v1.0

Fig. 25. Time-history plots showing pressure, temperature, and volume fractions of hydrogen and water vapor at cell location (10,25,2). Note that at this location, the chemical reaction is essentially complete at about 2.0 s.

```

HDR T31.5 Test Simulation                HMS-93
TSA-8 LAM
HMS-HDRT31.5
NOTES:
    Initial temperature = 32 C.
    Run 1: 0-60 sec
-----
$xput
nrsdump = 0,
autot   = 1.0,
cyl     = 1.0,
delt0   = 0.002,
deltmin = 1.000e-04,
deltmax = 1.000e-00,
epsi0   = 1.000e-05,
epsimax = 1.000e-05,
epsimin = 1.000e-05,
gz      = -980.0,
iobpl   = 1,
itdowndt = 500,
itupdt  = 500,
itmax   = 1000,
lpr     = 1,
maxcyc  = 9000001,
ittyfreq = 100,
nu      = 0.15,
prandtl = 0.7,
schmidt = 0.45,
muoption = 1,
idiffmom = 1,
idiffme  = 1,
tmodel  = 'alg',
clength = 50.,
pltdt   = 40.00,
prtdt   = 5000.00,
twfin   = 60.000,
tdt     = 5000.00,
velmx   = 5.0,
ibb     = 1,
ibn     = 4,      ; Periodic boundary condition in
ibs     = 4,      ; azimuthal direction.
ibw     = 1,
ibe     = 1,
ibt     = 1,

mat = 'air', 'h2o', 'lg', ; 'lg' is only valid species in modified code.

gasdef(1,1) = 1,10 , 1,24 , 1,31 , 1 ,      ; Only one "gasdef"
              1.00000000000e6, 305.00, 1, 0., 0., ; for global i. c.
              'air', 1.00000,

mobs = 1, 2,      1, 24,      1, 3,      1, 1,
        3, 4,      1, 6,      1, 2,      1, 1,
        3, 4,      7, 19,     1, 2,      1, 1,
        3, 4,      20, 24,     1, 2,      1, 1,
        6, 10,     1, 24,     1, 3,      1, 1,
        3, 10,     1, 17,     2, 3,      1, 1,
        3, 10,     18, 19,     2, 3,      1, 1,
        3, 10,     20, 23,     2, 3,      1, 1,
        3, 4,      23, 24,     2, 3,      1, 1,
        5, 10,     23, 24,     2, 3,      1, 1,
        3, 5,      17, 18,     2, 3,      1, 1,
        3, 5,      19, 20,     2, 3,      1, 1,
        3, 4,      1, 2,      3, 4,      1, 1,
        3, 4,      3, 6,      3, 4,      1, 1,
        3, 4,      7, 18,     3, 4,      1, 1,
        3, 4,      20, 24,     3, 4,      1, 1,
        7, 10,     1, 24,     3, 6,      1, 1,
        5, 6,      1, 2,      4, 6,      1, 1,
        5, 6,      23, 24,     4, 6,      1, 1,
        1, 4,      1, 2,      4, 5,      1, 1,
        1, 4,      3, 24,     4, 5,      1, 1,
        1, 2,      2, 3,      4, 5,      1, 1,
        1, 2,      2, 3,      4, 5,      1, 1,
        3, 4,      2, 3,      4, 5,      1, 1,
        3, 4,      1, 24,     5, 6,      1, 1,
        1, 2,      19, 24,     5, 6,      1, 1,
        8, 10,     1, 24,     6, 7,      1, 1,

```

Fig. 26a. Continuation of the input listing for the HDR sample problem.

1, 3,	19, 24,	6, 7,	1, 1,
3, 4,	1, 24,	6, 7,	1, 1,
4, 5,	1, 24,	6, 7,	1, 1,
5, 6,	1, 19,	6, 7,	1, 1,
5, 6,	20, 24,	6, 7,	1, 1,
6, 7,	1, 16,	6, 7,	1, 1,
6, 7,	17, 19,	6, 7,	1, 1,
6, 7,	21, 24,	6, 7,	1, 1,
7, 8,	1, 5,	6, 7,	1, 1,
7, 8,	6, 20,	6, 7,	1, 1,
7, 8,	22, 24,	6, 7,	1, 1,
8, 9,	2, 19,	7, 10,	1, 1,
3, 4,	1, 18,	7, 8,	1, 1,
3, 4,	20, 24,	7, 8,	1, 1,
4, 5,	8, 9,	7, 8,	1, 1,
1, 2,	1, 24,	8, 9,	1, 1,
2, 3,	1, 5,	8, 9,	1, 1,
2, 3,	8, 24,	8, 9,	1, 1,
3, 4,	1, 24,	8, 9,	1, 1,
4, 6,	8, 9,	8, 10,	1, 1,
3, 4,	1, 24,	9, 10,	1, 1,
3, 4,	1, 24,	10, 11,	1, 1,
4, 5,	1, 4,	10, 11,	1, 1,
4, 5,	5, 13,	10, 11,	1, 1,
4, 5,	14, 24,	10, 11,	1, 1,
5, 6,	1, 4,	10, 11,	1, 1,
5, 6,	5, 19,	10, 11,	1, 1,
5, 6,	20, 24,	10, 11,	1, 1,
6, 7,	1, 13,	10, 11,	1, 1,
6, 7,	14, 19,	10, 11,	1, 1,
6, 7,	21, 24,	10, 11,	1, 1,
7, 8,	1, 2,	10, 11,	1, 1,
7, 8,	3, 15,	10, 11,	1, 1,
7, 8,	16, 18,	10, 11,	1, 1,
7, 8,	19, 20,	10, 11,	1, 1,
7, 8,	22, 24,	10, 11,	1, 1,
8, 9,	1, 20,	10, 11,	1, 1,
8, 9,	22, 24,	10, 11,	1, 1,
3, 4,	1, 20,	11, 12,	1, 1,
3, 4,	21, 24,	11, 12,	1, 1,
4, 6,	12, 13,	11, 12,	1, 1,
5, 6,	7, 11,	11, 13,	1, 1,
6, 8,	8, 9,	11, 13,	1, 1,
7, 9,	5, 6,	11, 12,	1, 1,
7, 8,	12, 13,	11, 12,	1, 1,
8, 9,	1, 5,	11, 13,	1, 1,
8, 9,	7, 19,	11, 13,	1, 1,
8, 9,	22, 24,	11, 13,	1, 1,
4, 8,	12, 13,	12, 13,	1, 1,
1, 2,	1, 24,	12, 13,	1, 1,
2, 3,	1, 15,	12, 13,	1, 1,
2, 3,	17, 24,	12, 13,	1, 1,
3, 4,	1, 24,	12, 13,	1, 1,
3, 4,	1, 24,	14, 15,	1, 1,
4, 5,	1, 2,	14, 15,	1, 1,
4, 5,	3, 6,	14, 15,	1, 1,
4, 5,	7, 8,	14, 15,	1, 1,
4, 5,	12, 24,	14, 15,	1, 1,
5, 7,	1, 8,	14, 15,	1, 1,
5, 7,	12, 17,	14, 15,	1, 1,
5, 6,	17, 18,	14, 15,	1, 1,
5, 7,	18, 19,	14, 15,	1, 1,
5, 6,	20, 21,	14, 15,	1, 1,
5, 7,	21, 22,	14, 15,	1, 1,
5, 9,	22, 24,	14, 15,	1, 1,
7, 8,	1, 2,	14, 15,	1, 1,
7, 8,	3, 8,	14, 15,	1, 1,
7, 8,	12, 15,	14, 15,	1, 1,
7, 8,	16, 18,	14, 15,	1, 1,
7, 8,	19, 20,	14, 15,	1, 1,
8, 9,	1, 20,	14, 15,	1, 1,
3, 4,	1, 23,	13, 14,	1, 1,
8, 9,	1, 5,	13, 14,	1, 1,
8, 9,	7, 19,	13, 14,	1, 1,
8, 9,	22, 24,	13, 14,	1, 1,
4, 8,	7, 13,	13, 14,	1, 1,
3, 4,	1, 2,	15, 16,	1, 1,
3, 4,	3, 10,	15, 16,	1, 1,

Fig. 26b. Continuation of the input listing for the HDR sample problem.

3, 4,	11, 20,	15, 16,	1, 1,
3, 4,	21, 24,	15, 16,	1, 1,
4, 7,	7, 8,	15, 16,	1, 1,
4, 8,	12, 13,	15, 16,	1, 1,
7, 9,	22, 23,	15, 16,	1, 1,
8, 9,	3, 6,	15, 16,	1, 1,
8, 9,	7, 20,	15, 16,	1, 1,
8, 9,	3, 6,	16, 17,	1, 1,
8, 9,	7, 20,	16, 17,	1, 1,
1, 2,	1, 24,	15, 20,	1, 1,
2, 3,	1, 3,	16, 17,	1, 1,
2, 3,	5, 9,	16, 17,	1, 1,
2, 3,	11, 16,	16, 17,	1, 1,
2, 3,	18, 21,	16, 17,	1, 1,
2, 3,	23, 24,	16, 17,	1, 1,
3, 4,	1, 24,	16, 17,	1, 1,
4, 5,	7, 8,	16, 17,	1, 1,
7, 9,	7, 8,	16, 17,	1, 1,
4, 8,	12, 13,	16, 20,	1, 1,
3, 5,	1, 24,	17, 18,	1, 1,
5, 6,	1, 4,	17, 18,	1, 1,
5, 6,	9, 11,	17, 18,	1, 1,
5, 6,	20, 24,	17, 18,	1, 1,
6, 7,	1, 5,	17, 18,	1, 1,
6, 7,	21, 24,	17, 18,	1, 1,
7, 9,	1, 6,	17, 18,	1, 1,
7, 9,	22, 24,	17, 18,	1, 1,
8, 9,	7, 20,	17, 18,	1, 1,
7, 8,	19, 20,	17, 18,	1, 1,
5, 8,	13, 18,	17, 18,	1, 1,
5, 7,	18, 19,	17, 18,	1, 1,
5, 6,	5, 8,	17, 18,	1, 1,
6, 8,	7, 8,	17, 18,	1, 1,
8, 9,	3, 6,	18, 21,	1, 1,
8, 9,	7, 19,	18, 21,	1, 1,
3, 5,	1, 9,	18, 19,	1, 1,
3, 5,	11, 24,	18, 19,	1, 1,
6, 9,	7, 8,	18, 19,	1, 1,
5, 6,	17, 18,	18, 19,	1, 1,
7, 9,	17, 18,	18, 19,	1, 1,
3, 5,	1, 9,	19, 20,	1, 1,
3, 5,	11, 24,	19, 20,	1, 1,
3, 4,	1, 24,	19, 20,	1, 1,
5, 6,	7, 12,	19, 20,	1, 1,
6, 7,	7, 10,	19, 20,	1, 1,
6, 7,	11, 12,	19, 20,	1, 1,
6, 7,	13, 14,	19, 23,	1, 1,
6, 7,	14, 15,	19, 21,	1, 1,
7, 9,	7, 12,	19, 20,	1, 1,
5, 9,	17, 18,	19, 21,	1, 1,
3, 4,	1, 24,	20, 21,	1, 1,
2, 3,	2, 6,	20, 21,	1, 1,
2, 3,	8, 12,	20, 21,	1, 1,
2, 3,	14, 18,	20, 21,	1, 1,
2, 3,	20, 23,	20, 21,	1, 1,
4, 5,	1, 9,	20, 21,	1, 1,
4, 5,	11, 24,	20, 21,	1, 1,
5, 9,	7, 8,	20, 21,	1, 1,
8, 9,	2, 3,	18, 21,	1, 1,
1, 2,	1, 24,	20, 24,	1, 1,
3, 5,	1, 15,	21, 22,	1, 1,
3, 5,	17, 24,	21, 22,	1, 1,
5, 6,	1, 5,	21, 22,	1, 1,
5, 6,	17, 19,	21, 22,	1, 1,
5, 6,	20, 24,	21, 22,	1, 1,
6, 7,	1, 3,	21, 22,	1, 1,
6, 9,	3, 5,	21, 22,	1, 1,
6, 7,	14, 15,	21, 23,	1, 1,
6, 7,	17, 19,	21, 22,	1, 1,
6, 7,	21, 24,	21, 22,	1, 1,
7, 8,	1, 3,	21, 22,	1, 1,
7, 8,	17, 18,	21, 22,	1, 1,
7, 8,	19, 20,	21, 22,	1, 1,
7, 8,	22, 24,	21, 22,	1, 1,
5, 9,	7, 8,	21, 22,	1, 1,
8, 9,	1, 3,	21, 27,	1, 1,
8, 9,	7, 20,	21, 22,	1, 1,
8, 9,	22, 24,	21, 27,	1, 1,

Fig. 26c. Continuation of the input listing for the HDR sample problem.

8, 9,	9, 19,	21, 27,	1, 1,
8, 9,	7, 9,	21, 23,	1, 1,
3, 4,	1, 6,	22, 24,	1, 1,
3, 9,	2, 3,	22, 24,	1, 1,
3, 6,	2, 5,	22, 24,	1, 1,
3, 5,	2, 6,	22, 24,	1, 1,
3, 5,	7, 20,	22, 23,	1, 1,
3, 6,	7, 8,	22, 23,	1, 1,
7, 9,	7, 8,	22, 23,	1, 1,
5, 9,	17, 18,	22, 23,	1, 1,
3, 4,	21, 24,	22, 23,	1, 1,
4, 5,	21, 23,	22, 23,	1, 1,
4, 9,	22, 23,	22, 23,	1, 1,
3, 5,	2, 15,	23, 24,	1, 1,
5, 6,	7, 9,	23, 24,	1, 1,
3, 7,	11, 15,	23, 24,	1, 1,
3, 4,	15, 24,	23, 24,	1, 1,
4, 5,	17, 23,	23, 24,	1, 1,
5, 6,	17, 19,	23, 24,	1, 1,
5, 6,	20, 23,	23, 24,	1, 1,
6, 7,	16, 19,	23, 24,	1, 1,
6, 7,	21, 23,	23, 24,	1, 1,
7, 8,	16, 18,	23, 24,	1, 1,
7, 9,	19, 20,	23, 24,	1, 1,
7, 8,	22, 23,	23, 24,	1, 1,
6, 9,	7, 8,	23, 24,	1, 1,
3, 9,	2, 3,	24, 26,	1, 1,
3, 6,	3, 5,	24, 26,	1, 1,
3, 5,	5, 7,	24, 26,	1, 1,
3, 4,	7, 11,	24, 26,	1, 1,
3, 9,	7, 8,	25, 26,	1, 1,
3, 6,	7, 8,	24, 25,	1, 1,
7, 8,	7, 8,	24, 25,	1, 1,
3, 5,	9, 11,	24, 26,	1, 1,
6, 7,	11, 15,	24, 26,	1, 1,
6, 9,	17, 18,	25, 26,	1, 1,
3, 4,	15, 24,	24, 26,	1, 1,
4, 5,	15, 23,	24, 26,	1, 1,
5, 9,	17, 18,	24, 25,	1, 1,
5, 9,	22, 23,	24, 26,	1, 1,
1, 4,	1, 24,	26, 27,	1, 1,
4, 5,	2, 5,	26, 27,	1, 1,
5, 9,	2, 5,	26, 27,	1, 1,
8, 9,	7, 20,	26, 27,	1, 1,
8, 9,	22, 24,	26, 27,	1, 1,
4, 5,	2, 23,	26, 27,	1, 1,
5, 9,	7, 18,	26, 27,	1, 1,
5, 7,	18, 19,	26, 27,	1, 1,
5, 6,	20, 23,	26, 27,	1, 1,
6, 7,	21, 23,	26, 27,	1, 1,
7, 9,	19, 20,	26, 27,	1, 1,
7, 9,	22, 23,	26, 27,	1, 1,
7, 9,	22, 23,	26, 27,	1, 1,
8, 9,	11, 15,	27, 28,	1, 1,
8, 9,	17, 21,	27, 28,	1, 1,
7, 8,	12, 14,	27, 28,	1, 1,
7, 8,	17, 18,	27, 29,	1, 1,
7, 8,	19, 21,	27, 29,	1, 1,
8, 9,	17, 21,	28, 29,	1, 1,
6, 7,	18, 19,	11, 14,	1, 1,
walls =	5, 6,	5, 7,	22, 22,
	7, 9,	3, 6,	22, 22,
	6, 7,	3, 5,	22, 22,
	7, 9,	5, 6,	24, 24,
	5, 6,	5, 7,	24, 24,
	6, 7,	7, 8,	24, 24,
	4, 9,	8, 9,	24, 24,
	2, 3,	1, 24,	24, 24,
	7, 8,	18, 19,	29, 29,
	1, 9,	1, 24,	30, 30,
	6, 7,	7, 8,	25, 25,
	4, 7,	8, 9,	25, 25,
	5, 6,	10, 14,	9, 9,
	6, 7,	11, 14,	9, 9,
	6, 7,	10, 11,	24, 24,
	4, 6,	3, 3,	3, 6,
	6, 7,	4, 4,	3, 6,

Fig. 26d. Continuation of the input listing for the HDR sample problem.

5, 6,	5, 5,	3, 6,	1, 1,
5, 6,	8, 8,	3, 6,	1, 1,
6, 7,	10, 10,	3, 6,	1, 1,
5, 7,	13, 13,	3, 6,	1, 1,
6, 7,	18, 18,	3, 6,	1, 1,
4, 7,	22, 22,	3, 6,	1, 1,
4, 6,	3, 3,	6, 7,	1, 1,
6, 8,	4, 4,	6, 10,	1, 1,
7, 8,	5, 5,	6, 7,	1, 1,
7, 8,	6, 6,	6, 7,	1, 1,
6, 7,	16, 16,	6, 7,	1, 1,
6, 7,	17, 17,	6, 7,	1, 1,
4, 8,	22, 22,	6, 7,	1, 1,
4, 5,	2, 2,	7, 10,	1, 1,
6, 8,	7, 7,	7, 10,	1, 1,
7, 8,	9, 9,	7, 10,	1, 1,
5, 6,	10, 10,	7, 10,	1, 1,
6, 7,	11, 11,	7, 9,	1, 1,
5, 8,	14, 14,	7, 10,	1, 1,
7, 8,	18, 18,	7, 29,	1, 1,
7, 8,	19, 19,	8, 29,	1, 1,
5, 9,	22, 22,	7, 10,	1, 1,
5, 6,	24, 24,	7, 10,	1, 1,
6, 7,	18, 18,	7, 8,	1, 1,
6, 7,	10, 10,	8, 10,	1, 1,
5, 7,	18, 18,	8, 10,	1, 1,
7, 8,	24, 24,	11, 14,	1, 1,
5, 7,	2, 2,	11, 14,	1, 1,
5, 7,	4, 4,	11, 14,	1, 1,
5, 7,	7, 7,	11, 12,	1, 1,
5, 8,	7, 7,	12, 13,	1, 1,
5, 6,	11, 11,	11, 13,	1, 1,
5, 7,	14, 14,	11, 12,	1, 1,
7, 8,	15, 15,	11, 18,	1, 1,
7, 8,	16, 16,	11, 18,	1, 1,
5, 7,	16, 16,	11, 14,	1, 1,
5, 8,	17, 17,	11, 14,	1, 1,
6, 7,	18, 18,	11, 14,	1, 1,
6, 7,	16, 16,	13, 14,	1, 1,
7, 8,	5, 5,	12, 14,	1, 1,
5, 8,	14, 14,	12, 14,	1, 1,
5, 7,	2, 2,	15, 17,	1, 1,
6, 9,	3, 3,	15, 17,	1, 1,
5, 6,	15, 15,	15, 17,	1, 1,
6, 7,	16, 16,	15, 17,	1, 1,
5, 7,	19, 19,	15, 17,	1, 1,
5, 9,	23, 23,	15, 17,	1, 1,
5, 6,	5, 5,	17, 18,	1, 1,
6, 8,	7, 7,	17, 18,	1, 1,
5, 7,	19, 19,	17, 18,	1, 1,
6, 9,	2, 2,	18, 21,	1, 1,
6, 8,	3, 3,	18, 21,	1, 1,
5, 8,	5, 5,	18, 21,	1, 1,
6, 7,	16, 16,	18, 21,	1, 1,
6, 9,	23, 23,	18, 21,	1, 1,
5, 6,	3, 3,	19, 21,	1, 1,
6, 9,	5, 5,	21, 22,	1, 1,
6, 7,	5, 5,	22, 23,	1, 1,
8, 9,	5, 5,	22, 23,	1, 1,
6, 9,	5, 5,	23, 24,	1, 1,
6, 7,	5, 5,	24, 25,	1, 1,
8, 9,	5, 5,	24, 25,	1, 1,
5, 8,	9, 9,	24, 26,	1, 1,
5, 6,	11, 11,	24, 26,	1, 1,
5, 6,	15, 15,	24, 26,	1, 1,
6, 8,	16, 16,	24, 26,	1, 1,
6, 9,	5, 5,	25, 26,	1, 1,
4, 6,	15, 15,	23, 24,	1, 1,
5, 6,	2, 2,	9, 10,	1, 1,
6, 6,	4, 5,	3, 4,	1, 1,
5, 5,	5, 6,	3, 4,	1, 1,
5, 5,	7, 8,	3, 4,	1, 1,
6, 6,	3, 5,	4, 6,	1, 1,
5, 5,	5, 8,	4, 6,	1, 1,
6, 6,	8, 9,	3, 6,	1, 1,
5, 5,	11, 14,	3, 6,	1, 1,
6, 6,	15, 18,	3, 6,	1, 1,
6, 6,	1, 9,	7, 10,	1, 1,

Fig. 26e. Continuation of the input listing for the HDR sample problem.

7, 7,	9, 10,	7, 10,	1, 1,
6, 6,	10, 11,	7, 9,	1, 1,
5, 5,	14, 17,	7, 8,	1, 1,
7, 7,	18, 19,	7, 29,	1, 1,
5, 5,	10, 12,	7, 8,	1, 1,
7, 7,	11, 14,	7, 9,	1, 1,
5, 5,	23, 24,	7, 8,	1, 1,
9, 9,	1, 2,	7, 10,	1, 2,
9, 9,	19, 24,	7, 10,	1, 2,
6, 6,	9, 10,	8, 10,	1, 1,
5, 5,	10, 18,	8, 10,	1, 1,
5, 5,	22, 24,	8, 9,	1, 1,
9, 9,	20, 22,	10, 11,	1, 2,
5, 5,	2, 4,	11, 14,	1, 1,
7, 7,	4, 5,	11, 14,	1, 1,
7, 7,	1, 2,	12, 14,	1, 1,
9, 9,	6, 7,	11, 12,	1, 2,
9, 9,	5, 7,	12, 14,	1, 2,
9, 9,	19, 22,	11, 14,	1, 2,
5, 5,	14, 15,	11, 12,	1, 1,
5, 5,	13, 16,	12, 14,	1, 1,
5, 5,	16, 17,	13, 14,	1, 1,
7, 7,	15, 16,	11, 18,	1, 1,
9, 9,	20, 22,	14, 15,	1, 2,
9, 9,	1, 3,	14, 17,	1, 2,
9, 9,	6, 7,	15, 21,	1, 2,
9, 9,	20, 22,	15, 17,	1, 2,
9, 9,	22, 23,	15, 17,	1, 2,
9, 9,	21, 22,	16, 17,	1, 2,
5, 5,	1, 2,	16, 17,	1, 1,
5, 5,	23, 24,	15, 17,	1, 1,
7, 7,	23, 24,	15, 17,	1, 1,
7, 7,	1, 2,	15, 17,	1, 1,
6, 6,	2, 3,	16, 17,	1, 1,
5, 5,	14, 15,	15, 16,	1, 1,
6, 6,	15, 16,	15, 17,	1, 1,
5, 5,	17, 19,	15, 16,	1, 1,
5, 5,	13, 19,	16, 17,	1, 1,
9, 9,	23, 24,	15, 17,	1, 2,
6, 6,	16, 17,	18, 21,	1, 1,
6, 6,	23, 24,	18, 21,	1, 1,
6, 6,	1, 3,	19, 21,	1, 1,
9, 9,	20, 22,	17, 18,	1, 2,
9, 9,	1, 2,	18, 21,	1, 2,
9, 9,	19, 24,	18, 21,	1, 2,
6, 6,	5, 7,	17, 18,	1, 1,
9, 9,	3, 7,	21, 26,	1, 2,
9, 9,	20, 22,	21, 24,	1, 2,
9, 9,	19, 22,	22, 23,	1, 2,
9, 9,	8, 9,	23, 26,	1, 2,
9, 9,	7, 8,	24, 25,	1, 2,
9, 9,	19, 22,	24, 26,	1, 2,
7, 7,	5, 9,	24, 25,	1, 1,
7, 7,	5, 8,	25, 26,	1, 1,
6, 6,	16, 17,	24, 26,	1, 1,
7, 7,	5, 7,	26, 27,	1, 1,
9, 9,	5, 7,	26, 27,	1, 2,
9, 9,	20, 22,	26, 27,	1, 2,
9, 9,	1, 11,	27, 28,	1, 2,
9, 9,	15, 17,	27, 28,	1, 2,
9, 9,	21, 24,	27, 28,	1, 2,
9, 9,	1, 24,	29, 30,	1, 2,
9, 9,	1, 17,	28, 29,	1, 2,
9, 9,	21, 24,	28, 29,	1, 2,

\$end

H E A T - T R A N S F E R

\$rheat

ihtflag = 1,

nhteslab = 10,

nhtewall = 10,

tsink0 = 305.,

Fig. 26f. Continuation of the input listing for the HDR sample problem.


```

tslab0 = 305.,
twallo = 305.,

walldef(1,1) = 1, 100.,
walldef(1,2) = 2, 3.,

sinkdef = 1, 9, 1, 24, 1, 2, 1, 2, 8.454058e+05, 0.94,
          1, 9, 1, 24, 3, 4, 1, 2, 1.948350e+06, 0.64,
          1, 9, 1, 24, 4, 5, 1, 2, 8.053100e+05, 0.60,
          1, 9, 1, 24, 5, 6, 1, 2, 9.050338e+05, 0.60,
          1, 9, 1, 24, 6, 7, 1, 2, 9.356719e+04, 0.58,
          1, 9, 1, 24, 7, 8, 1, 2, 5.117654e+06, 0.74,
          1, 9, 1, 24, 8, 9, 1, 2, 1.219332e+06, 0.74,
          1, 9, 1, 24, 9, 10, 1, 2, 1.299153e+06, 0.76,
          1, 9, 1, 24, 10, 11, 1, 2, 1.027740e+05, 0.72,
          1, 9, 1, 24, 11, 12, 1, 2, 2.461295e+06, 0.72,
          1, 9, 1, 24, 12, 13, 1, 2, 1.417796e+06, 0.56,
          1, 9, 1, 24, 13, 14, 1, 2, 1.205490e+06, 0.56,
          1, 9, 1, 24, 14, 15, 1, 2, 1.288242e+06, 1.28,
          1, 9, 1, 24, 15, 16, 1, 2, 3.688611e+06, 0.88,
          1, 9, 1, 24, 16, 17, 1, 2, 3.301155e+06, 0.88,
          1, 9, 1, 24, 17, 18, 1, 2, 2.400162e+06, 2.26,
          1, 9, 1, 24, 18, 19, 1, 2, 3.377140e+06, 1.04,
          1, 9, 1, 24, 19, 20, 1, 2, 3.223832e+06, 1.04,
          1, 9, 1, 24, 20, 21, 1, 2, 2.305958e+06, 0.78,
          1, 9, 1, 24, 21, 22, 1, 2, 1.608571e+06, 0.84,
          1, 9, 1, 24, 22, 23, 1, 2, 4.448468e+06, 1.04,
          1, 9, 1, 24, 23, 24, 1, 2, 3.241484e+06, 1.18,
          1, 9, 1, 24, 24, 25, 1, 2, 4.588547e+06, 1.42,
          1, 9, 1, 24, 25, 26, 1, 2, 3.605405e+06, 1.84,
          1, 9, 1, 24, 27, 28, 1, 2, 1.284570e+07, 1.44,
          1, 9, 1, 24, 28, 29, 1, 2, 1.400817e+07, 1.44,

Send
-----
M E S H
-----
$meshgn

iblock = 1,

xgrid =
0.0,      150.0,      230.0,      350.0,      500.0,      650.0,
800.0,      950.0,      1000.0,      1060.0,

ygrid =
0.0,      020.0,      036.0,      052.0,      068.0,      84.0,
094.0,      109.0,      124.0,      139.0,      154.0,      169.0,
184.0,      197.0,      213.0,      227.0,      241.0,      255.0,
270.0,      289.0,      305.0,      322.0,      340.0,      360.0,

zgrid =
0.0,      120.0,      220.0,      400.0,      500.0,      600.0,
700.0,      980.0,      1080.0,      1180.0,      1280.0,      1440.0,
1540.0,      1700.0,      1800.0,      2050.0,      2175.0,      2275.0,
2455.0,      2555.0,      2725.0,      2900.0,      3050.0,      3300.0,
3550.0,      3750.0,      3885.0,      4200.0,      4800.0,      5500.0,
5600.0,

Send
-----
G R A P H I C S
-----
$grafic

thdt = 0.1,

igrid = 0,      ; Axes will not be drawn on contour and vector plots.

pnt = 1, 1, 23, 1,
      10, 24, 23, 1,
      1, 1, 24, 1,
      10, 24, 24, 1,
      1, 20, 30, 1,
      10, 20, 30, 1,
      1, 1, 25, 1,
      10, 24, 25, 1,
      1, 10, 1, 1,

```

Fig. 26g. Continuation of the input listing for the HDR sample problem.

```

10, 10, 31, 1,
1, 11, 1, 1,
10, 11, 31, 1,
1, 12, 1, 1,
10, 12, 31, 1,

1, 6, 1, 1,
10, 6, 31, 1,
1, 7, 1, 1,
10, 7, 31, 1,

p1d = 5, 6, 'tk', 0,

ht1dp = 9, 20, 30, 1, 'wall', 'east',
6, 11, 24, 1, 'slab', 'west',

7, 11, 24, 1, 'wall', 'top',
7, 11, 24, 1, 'slab', 'north',

c2d = 1, 2, 'tk', 0,
3, 4, 'tk', 0,
7, 8, 'tk', 0,
9,10, 'tk', 0,
11,12, 'tk', 0,
13,14, 'tk', 0,
1, 2, 'pn', 0,
3, 4, 'pn', 0,
7, 8, 'pn', 0,
9,10, 'pn', 0,
11,12, 'pn', 0,
13,14, 'pn', 0,
1, 2, 'vf', 'h2o'
3, 4, 'vf', 'h2o'
7, 8, 'vf', 'h2o'
9,10, 'vf', 'h2o'
11,12, 'vf', 'h2o'
13,14, 'vf', 'h2o'

v2d = 1, 2, 1,
3, 4, 1,
7, 8, 1,
9,10, 1,
11,12, 1,
13,14, 1,
15,16, 1,
17,18, 1,

thp(1,1) = 6, 18, 28, 1, 'pn', 0,
7, 11, 24, 1, 'pn', 0,
7, 11, 25, 1, 'pn', 0,
7, 11, 24, 1, 'tk', 0,
7, 11, 25, 1, 'tk', 0,
7, 10, 23, 1, 'tk', 0,
2, 7, 30, 1, 'tk', 0,
9, 21, 12, 1, 'tk', 0,
6, 14, 4, 1, 'tk', 0,
3, 22, 30, 1, 'vf', 'air',
3, 22, 30, 1, 'vf', 'h2o',
3, 22, 30, 1, 'vf', 'lg',
7, 20, 20, 1, 'vf', 'lg',
7, 20, 8, 1, 'vf', 'lg',
7, 20, 12, 1, 'vf', 'lg',
7, 20, 12, 1, 'vf', 'lg',

htthp = 7, 11, 24, 1, 'wall', 'top',
7, 11, 24, 1, 'sink', 0,
6, 11, 16, 1, 'sink', 'west',
5, 11, 2, 1, 'slab', 'top',

$end
$special
$end
$parts
$end

```

Fig. 26h. End of the input listing for the HDR sample problem.

Fig. 27. 1D profile plot of fluid temperature. Note that the "x"-axis is actually the radial coordinates, because cylindrical geometry is chosen. The steel shell is at a radius of 1000 cm. The annular region outside the steel shell is considerably colder than inside the containment. This plot is generated by a p1d definition.

Fig. 28. Temperature profiles in some wall and slab heat structures. Note that temperature gradients in the walls are less than those in the slabs because the walls are made of steel and are much thinner than the concrete slabs (3 cm compared to 100 cm). These plots are generated by `ht1dp` definitions.

Fig. 29. Velocity vectors on a horizontal (r - θ) plane. The equal beginning and ending k -indices (23, as shown on the plot) indicate the axial level of this plane.

Fig. 30. Velocity vectors on a vertical (r - z) plane. The same beginning and ending j -indices (11, as shown on the plot) indicate the azimuthal location of this plane.

Frame No. 63

12 DEC 93 17:50:39 GASFLOW v1.0

Fig. 31. Basic time-history plots showing four total energy quantities. The lower plot on the left represents total energy change due to steam condensation. These plots are generated automatically by HMS.

Frame No. 64

12 DEC 93 17:50:39 GASFLOW v1.0

Fig. 32. Basic time-history plot showing the total fluid mass. The blowdown increases the total mass by about 7.5×10^6 g at around 40 s, after then condensation dominates and gradually reduces fluid mass in the containment atmosphere.

Frame No. 65

12 DEC 93 17:50:39 GASFLOW v1.0

Fig. 33. Time-history plots showing fluid pressure and temperature at some locations. Note the immediate temperature jump at the blowdown cell location (7,11,24).

Fig. 34. Time-history plots showing fluid temperature and species volume fraction at some locations. Note a more gradual temperature rise in cell (6,14,4) as compared to the blowdown cell location (7,11,24) shown in the previous figure. No "light gas" is released during the initial two-phase blowdown.

Frame No. 69

Fig. 35. Time-history plots showing surface temperature of some heat structures. The lower right plot shows the surface temperature of a slab heat structure at the bottom of the containment increases very little during the first minute. These plots are produced by `htthp` definitions.

REFERENCES

1. T. L. Wilson and J. R. Travis, "Hydrogen Mixing Studies (HMS): Theory and Computational Model," Los Alamos National Laboratory report LA-12459-MS (NUREG/CR-5948) (December 1992).
2. K. L. Lam, J. R. Travis, and T. L. Wilson, "Hydrogen Mixing Studies (HMS): Assessment Manual," Los Alamos National Laboratory report LA-12593-M (NUREG/CR-6060) (June 1993).
3. B. van Leer, "Towards the Ultimate Conservation Difference Scheme V. A Second-Order Sequel to Godunov's Method," *J. Comp. Phys.* **32**, 101 (1979).

Appendix 1. Summary of Variables in NAMELIST Group `xput`

(Variables superscripted with a double dagger (‡) are not used in HMS.)

Variable	Default	Description
<code>alpha</code> [‡]	1.0	Upstream weighting factor in advection scheme. Not used in HMS.
<code>autot</code>	1.0	Automatic time-step control flag: 1.0 means ON; 0.0 means OFF (fixed time-step size). See VIII.B.
<code>bbc(*,*)</code> [‡]	None	Buoyancy boundary condition definition. Not used in HMS.
<code>bcid(*,*)</code> [‡]	None	Boundary condition i. d. array. Not used in HMS.
<code>c1ke</code>	1.44	Parameter for k - ϵ turbulence model. See VII.C.3.
<code>c2ke</code>	1.92	Parameter for k - ϵ turbulence model. See VII.C.3.
<code>cbc(*,*)</code>	None	Continuative boundary condition definition array. See V.B.2.
<code>cbc(1,*)</code>	None	Beginning <i>i</i> mesh index (cell-face number).
<code>cbc(2,*)</code>	None	Ending <i>i</i> mesh index (cell-face number).
<code>cbc(3,*)</code>	None	Beginning <i>j</i> mesh index (cell-face number).
<code>cbc(4,*)</code>	None	Ending <i>j</i> mesh index (cell-face number).
<code>cbc(5,*)</code>	None	Beginning <i>k</i> mesh index (cell-face number).
<code>cbc(6,*)</code>	None	Ending <i>k</i> mesh index (cell-face number).
<code>cbc(7,*)</code>	None	Block number (must be set to 1).
<code>cbc(8,*)</code>	None	Start time (s).
<code>cbc(9,*)</code>	None	End time (s).
<code>clength</code>	30.48	Length scale for algebraic turbulence model. See VII.C.1.
<code>cmu</code>	0.05	Constant for algebraic turbulence model. See VII.C.1.
<code>cmug</code>	1.8E-04	Constant value of dynamic viscosity, to be used with <code>muoption = 2</code> . See VII.B.
<code>cmuke</code>	0.09	Parameter for k - ϵ turbulence model. See VII.C.3.
<code>cmusgs</code>	0.05	Constant for subgrid scale turbulence model. See VII.C.2.
<code>cyl</code>	0.0	Coordinate system option. See III.B. 0.0 means Cartesian; 1.0 means cylindrical.
<code>delt0</code>	0.02	Initial time increment size. See VIII.B.
<code>deltmax</code>	1.0E+30	Maximum allowable time increment size. See VIII.B.
<code>deltmin</code>	1.0E-04	Minimum allowable time increment size. Run will be terminated if this exceeds the time-step size calculated by code. See VIII.B..
<code>epsamb0</code> [‡]	372.68	Ambient or initial turbulence dissipation rate. Not used in HMS.
<code>epsi0</code>	1.0E-05	Initial pressure iteration error criterion. See VIII.A.
<code>epsimax</code>	1.0E-03	Maximum pressure iteration error criterion. See VIII.A.
<code>epsimin</code>	1.0E-06	Minimum pressure iteration error criterion. See VIII.A.

epsval (*)	None	Array to store values for turbulence dissipation rate. Used with turbdef. See VII.C.
fractke	0.1	Fraction of mean kinetic energy, used to calculate turbulent kinetic energy in algebraic turbulence model. See VII.C.1.
gasdef (*, *)	None	Gas definition array, used for defining initial and boundary conditions for fluid. See V.A.1.
gasdef (1, *)	None	Beginning i mesh index (cell-face number).
gasdef (2, *)	None	Ending i mesh index (cell face number).
gasdef (3, *)	None	Beginning j mesh index (cell-face number).
gasdef (4, *)	None	Ending j mesh index (cell-face number).
gasdef (5, *)	None	Beginning k mesh index (cell-face number).
gasdef (6, *)	None	Ending k mesh index (cell-face number).
gasdef (7, *)	None	Block number (must be set to 1).
gasdef (8, *)	None	Pressure (dynes/cm ²) in defined volume.
gasdef (9, *)	None	Temperature (K) in defined volume.
gasdef (10, *)	None	Option flag for specification of gas composition: 1 for mass fraction, 2 for volume fraction.
gasdef (11, *)	None	Time (s) at which "gas definition" begins.
gasdef (12, *)	None	Time (s) at which "gas definition" ends.
gasdef (13, *)	None	Gas species component number (determined by the order in the gas species list defined by mat). Gas species component can alternatively be specified by its symbol as given in Table III, e.g., 'h2'.
gasdef (14, *)	None	Mass or volume fraction of above gas species in defined volume.
gasdef (15, *)	None	Second gas species component number, if needed.
gasdef (16, *)	None	Mass or volume fraction of second gas species in defined volume, if needed
gasdef (40, *)	None	Last index in first dimension is 40 $\Rightarrow$ compositions of a maximum of $(40-12)/2 = 14$ species can be defined.
gx	0.0	Acceleration due to gravity in the i- (x- or r-) direction (cm/s ²).
gy	0.0	Acceleration due to gravity in the j- (y- or θ -) direction (cm/s ²).
gz	0.0	Acceleration due to gravity in the k- (z-) direction (cm/s ²).
ibb	1	Boundary condition indicator for -k (bottom) mesh boundary. See V.B.1. Options are: 1. Rigid free-slip. 2. Rigid no-slip. 3. Continuative. 4. Periodic. 5. Specified pressure.
ibe	1	Boundary condition indicator for +i (east) mesh boundary. See ibb description.
ibn	1	Boundary condition indicator for +j (north) mesh boundary. See ibb description.
ibs	1	Boundary condition indicator for -j (south) mesh boundary. See ibb description.

ibt	1	Boundary condition indicator for +k (top) mesh boundary. See ibb description.
iburn	0	Option flag for hydrogen combustion: 0 means OFF; 1 means ON.
ibw	1	Boundary condition indicator for -i (west) mesh boundary. See ibb description.
idiffme	0	Option flag for mass and energy diffusion. See VII.B. 0 means OFF; 1 means ON.
idiffmom	0	Option flag for momentum diffusion. See VII.B. 0 means OFF; 1 means ON.
ieopt [†]	1	Order of curve fit used for computing internal energy i as function of temperature T . Not used in HMS. Options are: $1 \Rightarrow i = a_0 + a_1 \cdot T$ $2 \Rightarrow i = a_0 + a_1 \cdot T + a_2 \cdot T^2$ $3 \Rightarrow i = a_0 + a_1 \cdot T + a_2 \cdot T^2 + a_3 \cdot T^3$ For options 2 and 3, see trange.
ifsurf [†]	None	Option flag. Not used in HMS.
ifvl	0	Option flag to activate van Leer advection scheme. See VII.C. 0 means donor-cell method; 1 means van Leer method.
ihystat	0	Option flag for imposing hydrostatic pressure gradient in fluid cells according to acceleration components g_x , g_y , and g_z . See VII.A. 0 means OFF; 1 means ON.
iobpl [†]	1	Option flag for perspective obstacle plotting. 1 means ON; 0 means OFF. Not used in HMS.
iorder [†]	1	Order of spatial discretization accuracy. Not used in HMS.
ipropt [†]	1	Option flag. Not used in HMS.
itdowndt	50	Number used for tightening pressure iteration (see VII.A). Also used to adjust time-step size (see VII.B)
itmax	20	Maximum number of pressure iterations per cycle. See VII.A.
ittyfreq	20	Frequency of printing iteration/cycle information to terminal. See IX.C.
itupdt	100	Number used for relaxing pressure iteration convergence criterion. See VII.A.
iwalls(*,*) [†]	None	i-wall definition array. Not used in HMS.
jwalls(*,*) [†]	None	j-wall definition array. Not used in HMS.
kwalls(*,*) [†]	None	k-wall definition array. Not used in HMS.
lpr	1	Control flag for printing to metafile. Must be set to 1 in HMS.
mat(*)	None	List of gas species symbols (enclosed in single quotes) for specifying gas components. See IV.A.
maxcyc	10	Maximum number of cycles allowed. See VIII.B. If set to negative, no fluid calculation will be done, but instead a set of mesh plots are produced, according to the keys in Table II. See III.D.
mobs(*,*)	None	Mesh obstacle definition array. See III.C.2.
mobs(1,*)	None	Beginning i mesh index (cell-face number).
mobs(2,*)	None	Ending i mesh index (cell-face number).
mobs(3,*)	None	Beginning j mesh index (cell-face number).

mobs(4,*)	None	Ending j mesh index (cell-face number).
mobs(5,*)	None	Beginning k mesh index (cell-face number).
mobs(6,*)	None	Ending k mesh index (cell-face number).
mobs(7,*)	None	Block number (must be set to 1).
mobs(8,*)	None	Material identification number: 1 means concrete; 2 means steel; 3 means superconductor. Only used if ihtflag = 1. See VI.B.
muoption	1	Option specifying whether nu or cmug is to be used for viscous stress calculation. See VII.B. 1 means nu is used; 2 means cmug is used.
np [†]	0	Number not used in HMS.
nrsdump	0	Number that appears in the name of the restart dump file to be read in. See IX.D. Negative or zero values indicate a new run.
nslipdef(*,*)	None	Array to define no-slip surfaces. See V.B.2.
nslipdef(1,*)	None	Beginning i mesh index (cell-face number).
nslipdef(2,*)	None	Ending i mesh index (cell-face number).
nslipdef(3,*)	None	Beginning j mesh index (cell-face number).
nslipdef(4,*)	None	Ending j mesh index (cell-face number).
nslipdef(5,*)	None	Beginning k mesh index (cell-face number).
nslipdef(6,*)	None	Ending k mesh index (cell-face number).
nslipdef(7,*)	None	Block number (must be set to 1).
nslipdef(8,*)	None	Side of the surface that is no-slip. Options: 'lower' means negative side; 'upper' means positive side; 'both' means both negative and positive sides.
nu	0.0	Kinematic viscosity, ν (cm ² /s). See VII.B.
omg [†]	1.7	Over-relaxation factor. Not used in HMS.
pamb0	1.0E+06	Ambient pressure value for plotting purposes. See IX.A.3.
pb(*,*)	None	Array for defining pressure boundary conditions. See V.B.2.
pb(1,*)	None	Beginning i mesh index (cell-face number).
pb(2,*)	None	Ending i mesh index (cell-face number).
pb(3,*)	None	Beginning j mesh index (cell-face number).
pb(4,*)	None	Ending j mesh index (cell-face number).
pb(5,*)	None	Beginning k mesh index (cell-face number).
pb(6,*)	None	Ending k mesh index (cell-face number).
pb(7,*)	None	Block number (must be set to 1).
pb(8,*)	None	Start time (s).
pb(9,*)	None	End time (s).
pltdt	1.0	Time interval (s) between successive sets of 1D profile, 2D contour, 2D and 3D velocity vector plots are generated. See IX.A.

prandtl	1.0	Fluid Prandtl number, used to determine thermal diffusivity. See VII.B.
prrdt	1000	Time interval (s) between printing of the fluid solution field(all velocity components, pressure, and density) to file <code>gfout</code> .
pvalue(*) [†]	None	Array for defining pressure values. Not used in HMS.
qbc(*,*) [†]	None	Specified volumetric flow rate boundary condition. Not used in HMS.
qvalue(*) [†]	None	Array for defining volumetric flow rate values. Not used in HMS.
revps [†]	0.0	Rotation rate of mesh. Not used in HMS.
schmidt	1.0	Fluid Schmidt number, used to determine mass diffusivity. See VII.B.
sclamb0 [†]	30.0	Ambient or initial turbulence length scale. Not used in HMS.
sclval(*)	None	Array to store values for turbulence length scale. Used with <code>turbdef</code> . See VII.C.
sigmae	1.3	Parameter for k - ϵ turbulence model. See VII.C.3.
sigmak	1.0	Parameter for k - ϵ turbulence model. See VII.C.3.
solatype [†]	0	Option to activate particle transport. Not used in HMS.
tddt	10	Time interval (s) at which restart dump files are written. See IX.D.
tkeamb0 [†]	500.0	Ambient or initial turbulent kinetic energy. Not used in HMS.
tkeval(*)	None	Array to store values for turbulent kinetic energy. Used with <code>turbdef</code> . See VII.C.
tlimd [†]	1.0	Variable not used in HMS.
tmodel	'none'	Turbulence model option: 'none' means no turbulence model; 'alg' means algebraic model (see VII.C.1); 'sgs' means subgrid scale model (see VII.C.2); 'ke' means k - ϵ model (see VII.C.3).
trange [†]	'low'	Temperature range for determining internal energy vs. temperature relations. Not used in HMS. Options are 'low' for temperatures less than 3000 K; 'high' for temperatures higher than 3000 K but less than 6000K.
turbdef(*,*)	None	Array to define turbulence quantities for subgrid scale and k - ϵ models. See VII.C.2.
turbdef(1,*)	None	Beginning <i>i</i> mesh index (cell-face number).
turbdef(2,*)	None	Ending <i>i</i> mesh index (cell-face number).
turbdef(3,*)	None	Beginning <i>j</i> mesh index (cell-face number).
turbdef(4,*)	None	Ending <i>j</i> mesh index (cell-face number).
turbdef(5,*)	None	Beginning <i>k</i> mesh index (cell-face number).
turbdef(6,*)	None	Ending <i>k</i> mesh index (cell-face number).
turbdef(7,*)	None	Block number (must be set to 1).
turbdef(8,*)	None	Integer pointer to location in <code>tkeval</code> array for value of turbulent kinetic energy.
turbdef(9,*)	None	Integer pointer to location in <code>epsval</code> array for value of turbulent kinetic energy.

turbdef(10,*)	None	Integer pointer to location in sclval array for value of turbulent kinetic energy.
turbdef(11,*)	None	Start time (s).
turbdef(12,*)	None	End time (s).
twfin	10	Problem end time (s).
ui	0.0	Initial velocity in i-direction (cm/s). See V.A.2.
vbc(*,*)	None	Array for defining velocity boundary conditions. See V.B.2.
vbc(1,*)	None	Beginning i mesh index (cell-face number).
vbc(2,*)	None	Ending i mesh index (cell-face number).
vbc(3,*)	None	Beginning j mesh index (cell-face number).
vbc(4,*)	None	Ending j mesh index (cell-face number).
vbc(5,*)	None	Beginning k mesh index (cell-face number).
vbc(6,*)	None	Ending k mesh index (cell-face number).
vbc(7,*)	None	Block number (must be set to 1).
vbc(8,*)	None	Element of vvalue that will define the velocity value.
vbc(9,*)	None	Start time (s).
vbc(10,*)	None	End time (s).
veldef(*,*)	None	"Equivalenced" to vbc.
velmx	2.0	Scaling factor for velocity vector plots (multiplies internally scaled vectors).
vi	0.0	Initial velocity in j-direction (cm/s). See V.A.2.
vvalue(*)	None	Array to store values for velocity, used with vbc. See V.B.2.
walls(*,*)	None	Wall definition array. See III.C.1.
walls(1,*)	None	Beginning i mesh index (cell-face number).
walls(2,*)	None	Ending i mesh index (cell-face number).
walls(3,*)	None	Beginning j mesh index (cell-face number).
walls(4,*)	None	Ending j mesh index (cell-face number).
walls(5,*)	None	Beginning k mesh index (cell-face number).
walls(6,*)	None	Ending k mesh index (cell-face number).
walls(7,*)	None	Block number (must be set to 1).
walls(8,*)	None	Integer to identify the type (material and effective thickness) of wall through the walldef array. Only used if ihtflag = 1. See VI.A.
wi	0.0	Initial velocity in k-direction (cm/s). See V.A.2.

Appendix 2. Summary of Variables in NAMELIST Group `meshgn`

(Refer to Section III.B for detailed explanation of variables.)

Variable	Default	Description
<code>dxmn (*)</code>	None	Array to store minimum cell size (cm) in <i>i</i> -direction for each submesh. Used for automatic mesh generation.
<code>dymn (*)</code>	None	Array to store minimum cell size (cm) in <i>j</i> -direction for each submesh. Used for automatic mesh generation.
<code>dzmn (*)</code>	None	Array to store minimum cell size (cm) in <i>k</i> -direction for each submesh. Used for automatic mesh generation.
<code>iblkxp (*, *)</code>	None	Multiblock connectivity information array. Not used in HMS.
<code>iblock</code>	None	Block identification number. Must be set to 1 in HMS.
<code>nkx</code>	None	Number of submesh in <i>i</i> -direction.
<code>nky</code>	None	Number of submesh in <i>j</i> -direction.
<code>nkz</code>	None	Number of submesh in <i>k</i> -direction.
<code>nxl (*)</code>	None	Array to store number of cells on the <i>-i</i> side for each submesh. Used for automatic mesh generation.
<code>nxr (*)</code>	None	Array to store number of cells on the <i>+i</i> side for each submesh. Used for automatic mesh generation.
<code>nyl (*)</code>	None	Array to store number of cells on the <i>-j</i> side for each submesh. Used for automatic mesh generation.
<code>nyr (*)</code>	None	Array to store number of cells on the <i>+j</i> side for each submesh. Used for automatic mesh generation.
<code>nzl (*)</code>	None	Array to store number of cells on the <i>-k</i> side for each submesh. Used for automatic mesh generation.
<code>nzr (*)</code>	None	Array to store number of cells on the <i>+k</i> side for each submesh. Used for automatic mesh generation.
<code>xc (*)</code>	None	Array to store location (cm) of edge of smallest cell in <i>i</i> -direction. Used for automatic mesh generation.
<code>xgrid (*)</code>	None	Array to store mesh coordinates in <i>i</i> -direction. Used for direct mesh definition.
<code>xl (*)</code>	None	Array to store <i>i</i> -coordinate (cm) of starting location of a submesh, which is the same as the ending location of the previous submesh. Used for automatic mesh generation.
<code>yc (*)</code>	None	Array to store location (cm) of edge of smallest cell in <i>j</i> -direction. Used for automatic mesh generation.
<code>ygrid (*)</code>	None	Array to store mesh coordinates in <i>j</i> -direction. Used for direct mesh definition.
<code>yl (*)</code>	None	Array to store <i>j</i> -coordinate (cm) of starting location of a submesh, which is the same as the ending location of the previous submesh. Used for automatic mesh generation.
<code>zc (*)</code>	None	Array to store location (cm) of edge of smallest cell in <i>k</i> -direction. Used for automatic mesh generation.
<code>zgrid (*)</code>	None	Array to store mesh coordinates in <i>k</i> -direction (cm). Used for direct mesh definition.
<code>zl (*)</code>	None	Array to store <i>j</i> -coordinate (cm) of starting location of a submesh, which is the same as the ending location of the previous submesh. Used for automatic mesh generation.

Appendix 3. Summary of Variables in NAMELIST Group rheat

(Variables superscripted with a double dagger (‡) are not used in HMS.)

Variable	Default	Description
ihtflag	1	Option flag to activate heat transfer and steam condensation. 1 means ON; 0 means OFF.
mobsdef(*,*) [‡]	none	Obstacle type definition array. Not used in HMS.
nhtemobs [‡]	2	Number of heat conduction elements in obstacle cells. Not used in HMS.
nhtesink	2	Number of heat conduction elements in a sink heat structure.
nhteslab	2	Number of heat conduction elements in a slab heat structure.
nhtewall	2	Number of heat conduction elements in a wall heat structure.
sinkdef(*,*)	None	Array to define distributed heat sinks. See VI.C.
sinkdef(1,*)	None	Beginning i mesh index (cell-face number).
sinkdef(2,*)	None	Ending i mesh index (cell-face number).
sinkdef(3,*)	None	Beginning j mesh index (cell-face number).
sinkdef(4,*)	None	Ending j mesh index (cell-face number).
sinkdef(5,*)	None	Beginning k mesh index (cell-face number).
sinkdef(6,*)	None	Ending k mesh index (cell-face number).
sinkdef(7,*)	None	Block number (must be set to 1).
sinkdef(8,*)	None	Material identification number: 1 means concrete; 2 means steel; 3 means superconductor.
sinkdef(9,*)	None	Total material volume (cm ³).
sinkdef(10,*)	None	Average material thickness (cm)
slabthk	100.0	Thickness (cm) of a slab heat structure. Also used to determine whether obstacle is a wall or slab heat structure. See VI.A.
tmobs0 [‡]	None	Initial temperature (K) of obstacles. Not used in HMS.
tsink0	300.0	Initial temperature (K) of sink heat structures. Negative values indicate temperature of adjacent fluid cell is to be used.
tslab0	300.0	Initial temperature (K) of slab heat structures. Negative values indicate temperature of adjacent fluid cell is to be used.
twall0	300.0	Initial temperature (K) of wall heat structures. Negative values indicate temperature of adjacent fluid cell is to be used.
walldef(*,*)	None	Wall type definition array. See VI.B.
walldef(1,*)	None	Material identification number: 1 means concrete; 2 means steel; 3 means superconductor.
walldef(2,*)	None	Effective thickness of wall (cm).

Appendix 4. Summary of Variables in NAMELIST Group **grafic**

(Variables superscripted with a double dagger (‡) are not used in HMS.)

Variable	Default	Description
c2d(*,*)	None	Array for defining 2D contour plots. See IX.A.5.
c2d(1,*)	None	Identification number for first point (second index in pnt).
c2d(2,*)	None	Identification number for second point (second index in pnt).
c2d(3,*)	None	Solution variable. Choose from list in Table IV.
c2d(4,*)	None	Gas species name or component number. Required only if c2d(3,*) is 'rsn', 'mf', or 'vf'.
gline	'on'	Option flag for drawing grid lines on time history plots. 'on' means grid lines drawn; 'off' means grid lines not drawn.
glinepld	'on'	Option flag for drawing grid lines on 1D profile plots. 'on' means grid lines drawn; 'off' means grid lines not drawn.
htldp(*,*)	None	Array for defining heat structure temperature profile plots. See IX.A.4.
htldp(1,*)	None	i-index of fluid cell in contact with heat structure.
htldp(2,*)	None	j-index of fluid cell in contact with heat structure.
htldp(3,*)	None	k-index of fluid cell in contact with heat structure.
htldp(4,*)	None	Block number (must be set to 1)
htldp(5,*)	None	Heat structure type. Choices are: 'slab' Slab heat structure; 'sink' Sink heat structure; 'wall' Wall heat structure.
htldp(6,*)	None	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are: 'west' +i side of fluid cell. 'east' -i side of fluid cell. 'south' +j side of fluid cell. 'north' -j side of fluid cell. 'bottom' +k side of fluid cell. 'top' -k side of fluid cell.
htthp(*,*)	None	Array for defining heat structure surface temperature time history plots. See IX.A.3
htthp(1,*)	None	i-index of fluid cell in contact with heat structure.
htthp(2,*)	None	j-index of fluid cell in contact with heat structure.
htthp(3,*)	None	k-index of fluid cell in contact with heat structure.
htthp(4,*)	None	Block number (must be set to 1)
htthp(5,*)	None	Heat structure type. Choices are: 'slab' Slab heat structure; 'sink' Sink heat structure; 'wall' Wall heat structure.

htthp(6,*)	None	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are: 'west' +i side of fluid cell. 'east' -i side of fluid cell. 'south' +j side of fluid cell. 'north' -j side of fluid cell. 'bottom' +k side of fluid cell. 'top' -k side of fluid cell.
igrid	1	Option flag for drawing grid box (axes) on 2D contour and vector plots. 1 means ON; 0 means OFF.
iinc	1	i-direction cell increment between velocity vectors.
ippka(*) [‡]	1	Particle type array for particle plots. Not used in HMS
ipvew(*) [‡]	1	View point array for particle plots. Not used in HMS.
jinc	1	j-direction cell increment between velocity vectors.
kinc	1	k-direction cell increment between velocity vectors.
nap [‡]	1	Number of film frames advanced between particle plots. Not used in HMS.
ncontur	7	Number of contour levels. See IX.A.5.
np1dpp [‡]	1	Number of 1D profile plots per page (frame). Not used in HMS.
npplts [‡]	0	Number of particle plots. Not used in HMS.
nthppp [‡]	4	Number of time history plots per page (frame). Not used in HMS.
p1d(*,*)	None	Array for defining 2D contour plots. See IX.A.5.
p1d(1,*)	None	Identification number for first point (second index in pnt).
p1d(2,*)	None	Identification number for second point (second index in pnt).
p1d(3,*)	None	Solution variable. Choose from list in Table IV.
p1d(4,*)	None	Gas species name or component number. Required only if p1d(3,*) is 'rsn', 'mf', or 'vf'.
p1ddt [‡]	None	Time (s) between 1D profile plots. Not used in HMS.
pnt(*,*)	None	Array to define points for profile, contour, and vector plots. See IX.A.4.
pnt(1,*)	None	i-mesh index
pnt(2,*)	None	j-mesh index
pnt(3,*)	None	k-mesh index
pnt(4,*)	None	Block number (must be set to 1)
thdt	1.0E100	Time interval (s) at which time-history data are stored. See IX.A.3.
thp(*,*)	None	Array for defining time-history plots. See IX.A.3.
thp(1,*)	None	i-mesh index (cell number or cell-face number).
thp(2,*)	None	j-mesh index (cell number or cell-face number).
thp(3,*)	None	k-mesh index (cell number or cell-face number).
thp(4,*)	None	Block number (must be set to 1).
thp(5,*)	None	Solution variable. Choose from list in Table IV.

thp(6,*)	None	Gas species name or component number. Required only if thp(5,*) is 'rsn', 'mf', or 'vf'.
v2d(*,*)	None	Array for defining 2D vector plots. See IX.A.6.
v2d(1,*)	None	Identification number for first point (second index in pnt).
v2d(2,*)	None	Identification number for second point (second index in pnt).
v2d(3,*)	None	Film advance flag. 0 means NO. 1 means YES.
v3d(*,*)	None	Array for defining 3D vector plots. See IX.A.6.
v3d(1,*)	None	Identification number for first point (second index in pnt).
v3d(2,*)	None	Identification number for second point (second index in pnt).
v3d(3,*)	None	Film advance flag. 0 means NO. 1 means YES.
v3d(4,*)	None	Number of 3D view coordinates definition (second index of corresponding viewcrds definition).
viewcrds(*,*)	None	Array for defining 3D viewing coordinates.
viewcrds(1,*)	None	i object coordinate (cm).
viewcrds(2,*)	None	j object coordinate (cm).
viewcrds(3,*)	None	k object coordinate (cm).
viewcrds(4,*)	None	i eye point coordinate (cm).
viewcrds(5,*)	None	j eye point coordinate (cm).
viewcrds(6,*)	None	k eye point coordinate (cm).

Appendix 5. Sample Input Deck with Minimum Data Required

To set up an HMS problem, one must, at the minimum, define a mesh, specify what fluid species are involved, and prescribe any appropriate initial and boundary conditions. In addition desired model options have to be activated. Many of the input variables used by HMS have default values. For example the default boundary condition is free-slip rigid wall, and the default initial velocity is zero. However, variables like `mat`, `gasdef`, and those in NAMELIST group `meshgn` (for mesh generation) have no defaults. Therefore, for a problem in which the fluid is initially at rest and is enclosed by free-slip solid boundaries, the minimum input would be that required to define the fluid species (`mat`), the initial fluid thermodynamic condition (`gasdef`), and the mesh (NAMELIST group `meshgn`). An input deck that has such minimum required data is shown in Fig. 36, which should help the new user to set up a simple problem quickly.

The fluid in the problem is air, which is initially at 300 K and 1×10^6 dynes/cm² pressure. Because the problem specifies no inflow or outflow and does not activate any physical models (such as heat transfer and gravity), the uniform pressure, temperature, and velocity fields should persist as the calculation advances in time. (In this case, the initial condition is the steady solution.) The accuracy of the calculation (measured by deviation of the velocity from zero, for example) is controlled by the pressure iteration convergence criterion (`epsi0`, default = 1×10^{-5}) and by the maximum iteration number allowed per cycle (`itmax`, set to 40 in this problem). The default initial time-step size (`delt0`) is 0.02 s, and the problem end time (`twfin`) is specified as 0.5 s. By default, automatic time-step control is chosen. Specification of the graphic NAMELIST variables is not strictly required, but contour and vector plots are graphics that are commonly asked for. With these problem specifications, the maximum velocity magnitude at the end of the calculation is 4×10^{-6} cm/s.


```

Fictitious Problem with Minimum Input Data                                HMS-93
TSA-8 Lam
HMS-Play
NOTES: 2-D domain 100 cm x 250 cm with deltax = deltax = deltax = 5 cm
Number of fluid cells = 20 x 50 x 1 for the
coordinate dimension x, y, and z, respectively.
This problem is a closed box of pure air experiencing no artificial
perturbation. The solution should show no deviation from the
initial conditions (300 K, 1 bar, no flow) as problem time increases.
-----
M A I N      I N P U T
-----
$xput
  twfin   = 1.0,      ; Default problem finish time is 10.0, so this
                      ; definition is not necessary for the code
                      ; to run. However, default pltdt (plotting
                      ; interval) is 1.0, so we set the problem time
                      ; to be the same so that only one set of
                      ; contour and vector plots are produced.
                      ; Default delto is 0.02, so there will be
                      ; a few cycles carried out.

  itmax    = 40,      ; Default is 20, which is not enough to get
                      ; an accurate solution for this problem, as
                      ; the mesh is not trivially small.

  maxcyc   = 9999999, ; Set maximum time cycle to large number
                      ; to ensure problem end time is reached.
                      ; Default maximum number is cycles is 10.

  mat = 'air',

; Initial Condition Throughout domain:

  gasdef(1,1) = 1 , 21, 1 , 51, 1 , 2, 1 ,
                1.000e6, 300.0, 2, 0., 0.,
                'air', 1.0,

$end
-----
M E S H      G E N E R A T I O N
-----
$meshgn
  iblock = 1,

  nkx=1,
    xl(1)= 0.0, xc(1) = 0.000 , nxl(1)= 0, nxr(1)=20 , dxmn(1)= 9999.,
    xl(2)= 100.0,

  nky=1,
    yl(1)= 0., yc(1) = 0.000 , nyl(1)= 0, nyx(1)=50 , dymn(1)= 9999.,
    yl(2)= 250.0,

  nkz=1,
    zl(1)=0.0000, zc(1) = 0.0000, nzl(1)= 0, nzx(1)= 1 , dzmn(1)= 9999.,
    zl(2)= 5.,

$end
-----
G R A P H I C S
-----
$grafic
  thdt = 0.1      ; Want to get the basic time history plots, because
                  ; default value for thdt (interval between time
                  ; history data are written and plotted) is 1.e100.
                  ; With thdt = 0.1, and twfin = 0.5, each basic time
                  ; history plot will have six data points, including
                  ; beginning and end times.

; Define two points that would cover the entire physical x-y domain.
  pnt(1, 1) = 1, 1, 2, 1,
              21, 51, 2, 1,

  v2d      = 1, 02, 1,      ; velocity vector plot on plane
                          ; defined by points 1 and 2.

  c2d      = 1, 02, 'tk' , 0, ; temperature contour plot on plane
                          ; defined by points 1 and 2. $end

$reheat
$end
$parts
$end
$special
$end

```

Fig. 36. Input deck showing minimum set of data required to run HMS.

NRC FORM 335 (2-89) NRCM 1102, 3201, 3202		U.S. NUCLEAR REGULATORY COMMISSION		1. REPORT NUMBER (Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.) NUREG/CR-6180 LA-12741-M							
BIBLIOGRAPHIC DATA SHEET											
2. TITLE AND SUBTITLE Hydrogen Mixing Studies (HMS): User's Manual				3. DATE REPORT PUBLISHED							
				<table border="1"> <tr> <td>MONTH</td> <td>YEAR</td> </tr> <tr> <td>December</td> <td>1994</td> </tr> </table>		MONTH	YEAR	December	1994		
MONTH	YEAR										
December	1994										
				4. FIN OR GRANT NUMBER W6100							
5. AUTHOR(S) K. L. Lam, T. L. Wilson, and J. R. Travis*				6. TYPE OF REPORT Technical							
				7. PERIOD COVERED (Inclusive Dates)							
8. PERFORMING ORGANIZATION - NAME AND ADDRESS (If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.) <table border="0"> <tr> <td>Los Alamos, National Laboratory</td> <td>*Science Applications International Corporation</td> </tr> <tr> <td>Los Alamos, New Mexico 87545</td> <td>2109 Air Park Rd., SE</td> </tr> <tr> <td></td> <td>Albuquerque, NM 87106</td> </tr> </table>						Los Alamos, National Laboratory	*Science Applications International Corporation	Los Alamos, New Mexico 87545	2109 Air Park Rd., SE		Albuquerque, NM 87106
Los Alamos, National Laboratory	*Science Applications International Corporation										
Los Alamos, New Mexico 87545	2109 Air Park Rd., SE										
	Albuquerque, NM 87106										
9. SPONSORING ORGANIZATION - NAME AND ADDRESS (If NRC, type "Same as above"; if contractor, provide NRC Division, Office or Region, U. S. Nuclear Regulatory Commission, and mailing address.) Division of Systems Research Office of Nuclear Regulatory Research U. S. Nuclear Regulatory Commission Washington, DC 20555											
10. SUPPLEMENTARY NOTES											
11. ABSTRACT (200 words or less) Hydrogen Mixing Studies (HMS) is a best-estimate analysis tool for predicting the transport, mixing, and combustion of hydrogen and other gases in nuclear reactor containments and other facilities. It can model geometrically complex facilities having multiple compartments and internal structures. The code can simulate the effects of steam condensation, heat transfer to walls and internal structures, chemical kinetics, and fluid turbulence. The gas mixture may consist of components included in a built-in library of 20 species. This manual describes how to use the code. It explains how to set up the model geometry, define walls and obstacles, and specify gas species and material properties. Definitions of initial and boundary conditions are also described. The manual also describes various physical model and numerical procedure options, as well as how to turn them on. The reader also learns how to specify different outputs, especially graphical display of solution variables. Finally sample problems are included to illustrate some applications of the code. An input deck that illustrates the minimum required data to run HMS is given at the end of this manual.											
12. KEY WORDS/DESCRIPTORS (List words or phrases that will assist researchers in locating the report.) Hydrogen, mixing, containment systems, computational modeling				13. AVAILABILITY STATEMENT Unlimited							
				14. SECURITY CLASSIFICATION <table border="1"> <tr> <td>(This Page)</td> </tr> <tr> <td>Unclassified</td> </tr> <tr> <td>(This Report)</td> </tr> <tr> <td>Unclassified</td> </tr> </table>		(This Page)	Unclassified	(This Report)	Unclassified		
				(This Page)							
				Unclassified							
(This Report)											
Unclassified											
15. NUMBER OF PAGES											
16. PRICE											