KIT SCIENTIFIC REPORTS 7711

GASFLOW-MPI: A Scalable Computational Fluid Dynamics Code for Gases, Aerosols and Combustion

Volume 2: Users' Manual (Revision 1.0)

Jianjun Xiao, Jack Travis, Peter Royl, Gottfried Necker
Anatoly Svishchev, Thomas Jordan

KIT Scientific Publishing

Jianjun Xiao, Jack Travis, Peter Royl, Gottfried Necker
Anatoly Svishchev, Thomas Jordan

GASFLOW-MPI: A Scalable Computational Fluid Dynamics Code for Gases, Aerosols and Combustion

Volume 2: Users' Manual (Revision 1.0)

Karlsruhe Institute of Technology
KIT SCIENTIFIC REPORTS 7711

GASFLOW-MPI: A Scalable Computational Fluid Dynamics Code for Gases, Aerosols and Combustion

Volume 2: Users' Manual (Revision 1.0)

by
Jianjun Xiao, Jack Travis, Peter Royl, Gottfried Necker
Anatoly Svishchev, Thomas Jordan

KIT Scientific Publishing

Report-Nr. KIT-SR 7711

Impressum

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

www.ksp.kit.edu

Print on Demand 2016

ISBN 978-3-7315-0448-1 (Vol. 1)
ISBN 978-3-7315-0449-8 (Vol. 2)
ISBN 978-3-7315-0447-4 (Set)
DOI: 10.5445/KSP/1000050394

Executive Summary

Karlsruhe Institute of Technology (KIT) is developing the parallel computational fluid dynamics code GASFLOW-MPI as a best-estimate tool for predicting transport, mixing, and combustion of hydrogen and other gases in nuclear reactor containments and other facility buildings. The code can model geometrically complex facilities with multiple compartments and internal structures. It can simulate the effects of two-phase dynamics with the homogeneous equilibrium model (HEM), two-phase heat transfer to walls and internal structures, chemical kinetics, catalytic recombiners, and fluid turbulence. An analysis with the GASFLOW-MPI code will result in the complete fluid dynamics description of gas species and discrete particle distribution and pressure, and temperature loadings on the walls and internal structures participating in an event.

GASFLOW sequential version has been used to calculate the distribution and control of hydrogen and noxious gases in complicated nuclear containment and confinement buildings and in nonnuclear facilities. It has been applied to situations involving transporting and distributing combustible gas mixtures. It has been used to study gas behavior in complicated containment systems with low-speed buoyancy-driven flows, with diffusion-dominated flows, and during deflagrations. The effects of controlling such mixtures by safety systems can be analyzed.

GASFLOW-MPI is a finite-volume code based on proven computational fluid dynamics methodology that solves the compressible Navier-Stokes equations for three-dimensional volumes in Cartesian or cylindrical coordinates. Wall shear stress models are provided for bulk laminar and turbulent flow. GASFLOW-MPI has transport equations for multiple gas species and one for internal energy. The two turbulence models available in GASFLOW-MPI are the algebraic and κ–ε model which provide zero- and two-transport-equation models that determine turbulent velocity and length scales needed to compute the turbulent viscosity. Terms for turbulent diffusion of different species are included in the mass and internal energy equations.

Heat conduction within walls and structures is one dimensional. Heat and mass transport to walls and structures is based on a modified Reynolds-Chilton-Colburn analogy, which accounts for increased heat transfer and condensation when the mass fraction of steam becomes a relatively large fraction of the mass of the gas mixture. Vaporization of fluid films is included with an inhibiting function as water vapor concentrations in fluid volumes adjacent to structures increase. Two-phase dynamics can occur in the fluid mixture volumes according to a classical homogeneous equilibrium model.

Chemical energy of combustion involving hydrogen provides a source of energy within the gaseous regions. A one-step global chemical kinetics model based on a modified Arrhenius law accounts for local hydrogen and oxygen concentrations. Hydrogen is ignited using a generalized ignitor model that represents both spark- and glow-plug-type designs. A catalytic hydrogen combination with oxygen is modeled using data from both the Nonproliferation and International Security division (NIS) and Siemens recombiner box designs.

The aerosol model comprises the following models: Lagrangian discrete particle transport, stochastic turbulent particle diffusion, particle deposition, particle entrainment, and particle cloud. These models incorporate the physics of particle behavior to model discrete particle phenomena and allow the code user to track the transport, deposition, and entrainment of discrete particles as well as clouds of particles.

In GASFLOW-MPI, the computational domain is discretized by a mesh of rectangular parallelepiped cells in either Cartesian or cylindrical geometry where primary hydrodynamic variables are cell-face-centered normal velocity and cell-centered density, internal energy, and pressure. A linearized Arbitrary-Lagrangian-Eulerian method is used for approximating the solution to the mass, momentum, and energy conservation equations.

The code version described in this manual is designated GASFLOW-MPI 1.0. In the 1980s, the name Hydrogen Mixing Studies, or HMS, was applied to any of a series of codes developed to solve special problems in HMS using a common theoretical basis. The latest version of HMS (HMS-93, for the year 1993, also known as HMS 1.0 for the first integrated version) integrated the best features of all the older versions into a single software package. This work was sponsored by the US Nuclear Regulatory Commission (NRC) as a best-estimate tool for nuclear containment analyses involving hydrogen and cooling issues. HMS 1.0 is the initial version of a larger code package called GASFLOW, which is supported by the US Department of Energy (DOE) to address various nuclear and nonnuclear facility safety issues. HMS 1.0 for the NRC is the same as GASFLOW 1.0 for the DOE.

Previous versions of HMS were applied to the following facilities and standard problems:

- EPRI/HDR International Standard Problems.
- Sandia FLAME and VGES Facilities.
- Nevada Hydrogen Tests.
- NRC Containment Loads Working Group Standard Problems.
- HCOG 1/4 Scale Test Facility.
- CSNI Hydrogen Distribution Benchmark Problems.
- Hydrogen Rule for Large Dry Containments.
- PHDR Large-Scale Hydrogen Mixing Experiment.
- PHDR Fire Experiments.

GASFLOW 2.0 has been extended beyond GASFLOW 1.0 with the following developments:

- Independent multiblock computational domains.
- Independent multiblocks connected on external boundaries by a ventilation system.
- Implemention of a fraction area treatment to model flow areas smaller than a cell face area.
- Accurate internal energy as a function of temperature to 4th degree polynomials.
- Gas properties library of thermochemical and transport extended to 30 species.
- Homogeneous equilibrium model for fluid mixture.
- Droplet depletion or “rainout”.
- Two-phase heat and mass transfer to structural components.

- Both spark- and glow-plug ignitor models.
- Empirical hydrogen combustion limits.
- Hydrogen recombiner models.
- Transport, deposition, and entrainment of discrete particles.

GASFLOW-MPI 1.0 has been extended beyond the GASFLOW serial version 3.5 using the paradigms of Message Passing Interface (MPI) and domain decomposition. The data structure, parallel linear solvers and preconditioners in Portable Extensible Toolkit for Scientific Computing (PETSc) has been employed. GASFLOW-MPI 1.0 has been parallelized based on GASFLOW serial version 3.5 with major changes in the following:

- The data structure in GASFLOW serial code has been completely replaced by using the distributed arrays in PETSc library.
- The Precontioned Conjugate Residual method used in GASFLOW serial version has been replaced by the parallel preconditioners and linear solvers in PETSc library.
- Multiblock computational domains and multiblocks connected on external boundaries by a ventilation system is not supported in the current GASFLOW-MPI release. In order to keep the backward compitability, the block number in input variable arrays in ingf file, such as gasdef(7,*), mobs(7,*) and walls(7,*), were not removed. However, currently it must be always 1 for GASFLOW-MPI applications.
- Transport, deposition, and entrainment of discrete particles are currently not supported. These features will be implemented in the next release of GASFLOW-MPI.
- Only algebraic and κ-ε turbulence models are currently supported. More turbulence models will be implemented in future release of GASFLOW-MPI.
- Models for turbulent combustion have been extended in GASFLOW-MPI.
- Post-processing: pscan and graphic library, cgs, will not be used as post-processing tool in GASFLOW-MPI. Data format for GMV, Opendx, AVS and VISART are not suppoted. Instead, we provide Python tools, pyscan and create3D, for visualization purposes. The data can be read by the most popular 3D visualization tools, such as Paraview, Visit, Tecplot and Ensight.

Acknowledgements

We would like to thank all GASFLOW users for their bug reports and suggestions during all these years. We thank the financial supports from U.S. NRC and U.S. DOE for the development of GASFLOW serial version in 1980s and 1990s. We also would like to thank J. W. Spore, K. L. Lam, C. Müller, B. D. Nichols, T. L. Wilson of the Los Alamos National Laboratory for all the previous work that has been accomplished in the GASFLOW serial version. We greatly acknowledge B. Smith and the PETSc development team of Argonne National Laboratory for their support and help during the development of parallel version GASFLOW-MPI.

Contents

1 Introduction

1.1 Code Capabilities

GASFLOW-MPI is a scalable best-estimate computer code developed at the Karlsruhe Institute of Technology (KIT) for predicting the transport, mixing, and combustion of hydrogen and other gases, liquid water droplets, and aerosols in nuclear reactor containments and other nonnuclear buildings. The code can model geometrically complex facilities having internal structures and multiple compartments and is useful for facilities that are required to address various nuclear and nonnuclear safety issues. The gas/liquid mixture modeled may consist of components included in a built-in library of 25 species. The aerosols modeled can be of different material densities and sizes. The fluid flow modeled may be laminar or turbulent, subsonic or supersonic, single phase or two phase, and with or without aerosols. 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 GASFLOW-MPI code 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 European Power Reactor (EPR) or 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), multispecies, variable-density capabilities of an analysis tool such as GASFLOW-MPI are useful. The calculation will identify local regions of high hydrogen concentration within the multicompartment containment geometry where steam condensation also is occurring. GASFLOW-MPI modeling capability allows for the positioning of catalytic recombiners at various locations throughout the containment. The GASFLOW model for recombiners simulates a slow recombination of hydrogen and oxygen to form water even though the mixture is not flammable. 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 on 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 slow deflagrations. GASFLOW-MPI 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.

1.2 Computational Method

In this section, we briefly summarize the computational method adopted in GASFLOW-MPI. This is included so that the code user can quickly review the numerical approach and models. Further details on the theoretical aspects are given in the GASFLOW-MPI Theory and Computational Model Manual, Vol. 1.

GASFLOW-MPI is a parallelized finite-volume code that solves the time-dependent, 3D, compressible Navier-Stokes equations. Transport equations for the internal energy and for multiple gas species, a liquid droplet species, and multiaerosol sizes are also solved. The computational domain is discretized by a mesh of regular orthogonal cells in either Cartesian or cylindrical geometry. The computational domain is a single 3D block. 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 efficient matrix solvers and pre-conditioners in PETSc library, 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, GASFLOW-MPI currently provides an option of two turbulence models. These are the algebraic and κ-ε models, which are the zero- 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 the solution of tridiagonal matrices. The GASFLOW-MPI mesh expansion capability allows for small nodes at the surface and expanding larger nodes within the heat structure. This will result in accurate calculations of the surface temperatures. 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. GASFLOW-MPI 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.

The GASFLOW-MPI code aerosol model comprises a Lagrangian discrete particle transport model, a stochastic turbulent particle diffusion model, a particle deposition model, a particle entrainment model, and a particle cloud model. These models incorporate the physics of particle behavior to model discrete particle phenomena and allow the code user to track the transport, deposition, and entrainment of discrete particles, as well as clouds of particles, in nuclear systems.

1.3 General Code Features

GASFLOW serial code is a Fortran 90 computer code originally developed to run on the Cray supercomputers at LANL. It has been parallelized using the Message Passing Interface (MPI) and domain decomposition. The distributed array data structure in PETSc library has been employed in GASFLOW-MPI.

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 in which 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 as well as temperature and material of solid thermal structures may be specified. Different types of boundary conditions, which may be time dependent, can be specified on various portions of the computational domain boundaries and on internal wall and obstacle surfaces, and many of the boundary conditions can be changed during restart runs.

Please note that the centimeter-gram-second (cgs) system is used in GASFLOW-MPI for the units of dimensional quantities. Therefore, the user should carefully use the following units when preparing input data for the code:

Length	Mass	Time	Pressure	Temperature	Energy
cm	g	sec	dyn/cm^2	K	ergs (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).

The old-fasion built-in graphics package for displaying different views of the mesh and the computational results in GASFLOW serial code has been removed from the GASFLOW-MPI code. It means that pscan, cgs library and pgf which have been used for many years in GASFLOW serial code will not be available in GASFLOW-MPI. Instead, we developed Python tools, namely pyscan and create3D, for post-processing. 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 can be visualized using pyscan. Create3D is used to convert GASFLOW-MPI NETCDF-4 dump files for 3D visualization using either xdmf files for Paraview, Visit and Ensight, or .plt files for Tecplot.

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 1-1 lists all the files used or written by GASFLOW-MPI.

Table 1-1 Description of input and output files in GASFLOW-MPI

File Name	Description
ingf	Input data text file.
gfout	Output listing text file.
cyclinfo	Text file containing time-step and iteration information for each cycle of calculation.
gfd000*.nc	NETCDF-4 dump files used for restarting calculations and 3D visualization. The number of dump files produced can be controlled through user input. The data format can be converted by create3D for 3D visualization.
plothist.nc	NETCDF-4 file containing all time history data which can be plotted by pyscan.
profiles.nc	NETCDF-4 file containing all 1D profile, 2D contour and vector, 3D vector which can be plotted by pyscan.

2 Input Data Format

To run GASFLOW-MPI, 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. The first three lines contain alphanumeric data for problem identification purposes. These input data follow:

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.

The input data are read into the code via eight 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.
innet	Ventilation system description (not needed in GASFLOW-MPI).
meshgn	Specification of computational mesh (only one $meshgn is allowed in GASFLOW-MPI).
rheat	Specification of heat-transfer data.
grafic	Definition of graphical output options.
parts	Specification of variables related to particle transport.
special	Definition of miscellaneous 3D plotting variables.
specialp	Definition of additional 3D plotting variables.

Since multi-block with ventilation system is currently not supported in GASFLOW-MPI, it should be noted that $innet must be blank and only one $meshgn is allowed. The input file must contain the group names (though no data), so the input data will be processed successfully. All variables for the NAMELIST groups are described in Appendices.

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. 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). Column 1 and column 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.) Multidimensional array values are assigned in storage order. Any value can be repeated by n*value, where n is the repetition count. A delimiter ($end) terminates the NAMELIST group record.

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(1:14,1) = ...,is correct, whereas

gasdef (1:14,1) = ...,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. An input NAMELIST group record may contain only comments or may be entirely blank.

3 Geometry Definition

3.1 Cell Labeling Convention

In GASFLOW-MPI 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 GASFLOW-MPI, 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 and k = kmax are 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. The edges of these cell faces, form the grid lines. The GASFLOW-MPI 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. This computational cell labeling scheme is shown in Figure 3–1. GASFLOW numbering convention for cells (or cell centers) and cell faces using the i-direction with imax = 7 for illustration. The fictitious boundary cells are shaded, i. e., cell numbers 1 and 7. The real fluid cells are numbered from 2 to 6. The physical computational volume ranges from cell face number 1 to cell face number 6.

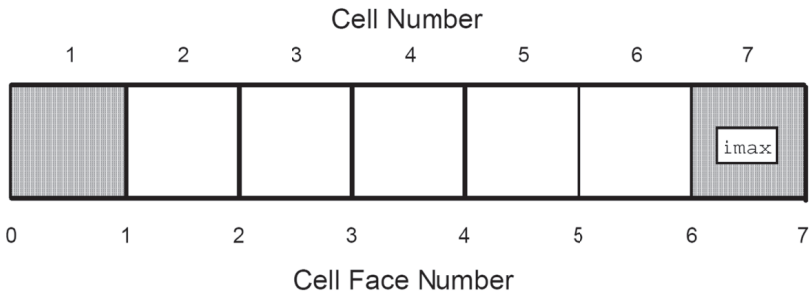


Figure 3-1 GASFLOW-MPI computational cell labeling scheme

3.2 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 cyl in the NAMELIST group xput. Set cyl = 0 (default) to use Cartesian coordinates or cyl = 1 to use cylindrical coordinates. Then the computational mesh is defined by one of two methods available. Input variables for both methods are in the NAMELIST group meshgn. Note that the user only defines geometry for the real physical domain. Fictitious boundary cells are assigned automatically by the code.

3.2.1 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 15˚ apart.

3.2.2 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. This method uses an automatic mesh generator which 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, **xl(k)**, to the left end of the next interval, **xl(k+1)**. Within each interval there is a location, **xc(k)**, where the mesh cells will be smallest. In other words, the grid lines in the k^{th} interval converge to location **xc(k)**. The number of cells between **xl(k)** and **xc(k)** is specified as **nxl(k)**, and the number from **xc(k)** to **xl(k+1)** is specified as **nxr(k)**. The minimum cell size, which is located at **xc(k)**, is specified as **dxmn(k)**.

Using the above information, the mesh generator expands cell sizes from a value of **dxmn(k)** at **xc(k)** in a quadratic manner such that the required number of cells will lie on each side of **xc(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.

Any number of cells can be defined on either side of **xc(k)**, including zero. A choice of zero is often useful when the minimum cell size is desired at the beginning or end of an interval. This is often done in problems in which a fine mesh resolution is required, in the vicinity of a surface where steep gradients in the temperature or velocity profile are expected.

GASFLOW supports definition of up to 49 mesh intervals 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.
xl(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.
nxl(k) specifies the number of cells to the left of **xc(k)**, i. e., between locations **xl(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 **xl(k+1)** in subdivision k.
dxmn(k) specifies the minimum cell size in the *x*-direction in subdivision k.

The maximum number of mesh subdivisions allowed is 50. 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, xl(k), xc(k), nxl(k), nxr(k), dxmn(k)**
y- or θ-direction	**nky, yl(k), yc(k), nyl(k), nyr(k), dymn(k)**
z-direction	**nkz, zl(k), zc(k), nzl(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** = 0.0). 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 for 3D block number 1, as depicted in Figure 3–2.

iblock = 1,

nkx = 1, **nkz** = 1,

xl(1)= 0., **xc(1)**= 5., **nxl(1)**= 5, **nxr(1)**= 5, **dxmn(1)**= 0.2, **xl(2)**= 12.

zl(1)= 0., **zc(1)**= 0., **nzl(1)**= 0, **nzr(1)**= 10, **dzmn(1)**= 1.e9, **zl(2)**= 12.,

Cylindrical Mesh. In the second example, the coordinate system chosen is cylindrical (**cyl** = 1.0 specified in NAMELIST group xput). Figure 3-3 shows the mesh in two dimensions generated by the following input in NAMELIST group meshgn:

iblock = 1,

nkx = 1, **nky** = 1,

xl(1)= 0., **xc(1)**= 15., **nxl(1)**= 10, **nxr(1)**= 0, **dxmn(1)**= 0.5, **xl(2)**= 15.,

yl(1)= 0., **yc(1)**= 0., **nyl(1)**= 0, **nyr(1)**= 24, **dymn(1)**= 1.e9, **yl(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 the total radius of 15 cm. The minimum cell size, **dxmn**, is specified as 0.5 cm, which is smaller than the "uniform" cell width given by 15 cm/10. Therefore, the cell size gradually expands from this minimum value at *r* = 10 cm to a maximum value at *r* = 0.

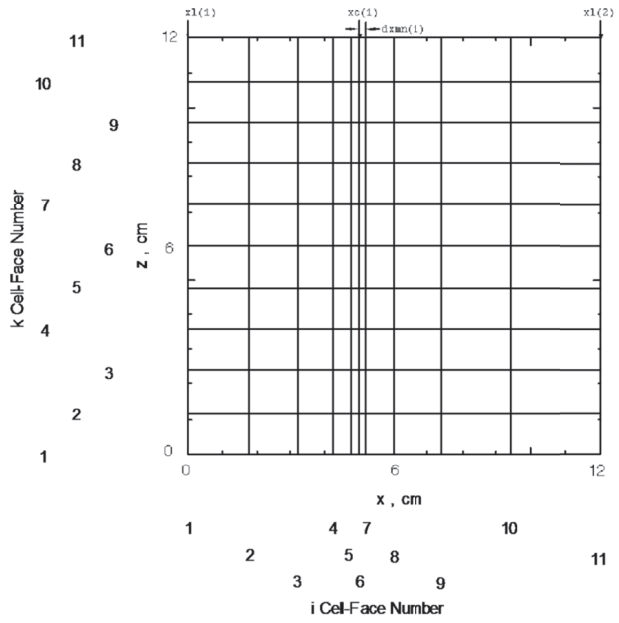


Figure 3-2 Two-dimensional *x*-*z* view of a mesh generated by the GASFLOW-MPI automatic mesh generator. Note that the vertical mesh lines converge on the *x* = 5 line.

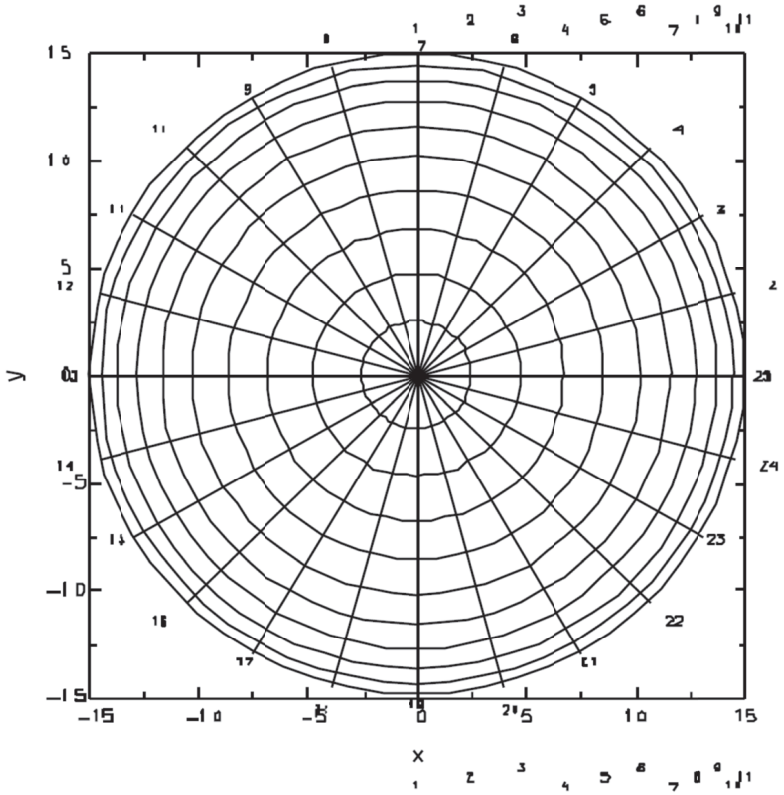


Figure 3-3 Two-dimensional, r-θ view of a mesh generated by the GASFLOW-MPI automatic mesh generator. The cell spacing becomes finer as r increases, but is uniform in the θ-direction.

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, δ_{max} in any submesh generated by this automatic mesh generator can be easily determined from the relation

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

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

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

2 x [(5 - 0)/5] - 0.2 = 1.8 cm

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

2 x [(12 - 5)/5] - 0.2 = 2.6 cm.

3.3 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 GASFLOW-MPI, 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.) Once *walls* and *obstacles* have been placed within the computational mesh, the user has the option of putting *holes* through both *wall* s and *obstacles*. This allows a more general construction of the complexities involved in developing complicated geometries.

3.3.1 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:

walls(1,*) Beginning i mesh index (cell face number).
walls(2,*) Ending i mesh index (cell face number).
walls(3,*) Beginning j mesh index (cell face number).
walls(4,*) Ending j mesh index (cell face number).
walls(5,*) Beginning k mesh index (cell face number).
walls(6,*) Ending k mesh index (cell face number).
walls(7,*) Block number (must be 1 for GASFLOW-MPI).
walls(8,*) Integer to identify the type of wall (thickness and material). Used only for heat transfer; ignored if heat transfer is not invoked.

The asterisk (*) should be replaced by an integer that identifies the particular wall definition (≤ 500). 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 is otherwise ignored. This is explained in Section 7.5. The rest of the input, **walls(1,*)** to **walls(6,*)**, specifies the location and extent of the wall. **walls(7,*)** is the block number which must be always 1 for GASFLOW-MPI. 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 in the example 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(1:8,1) = 2, 2, 1, 2, 2, 10, 1, 0,
walls(1:8,2) = 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 vector in the k-direction, or perpendicular to the first wall. If the input is applied to the mesh shown in Figure 3-3, then the two walls will appear on the *x-z* plane of the mesh, as shown in Figure 3-4.

3.3.2 Obstacles

Figure 3–4 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:

mobs(1,*)	Beginning i mesh index (cell face number).
mobs(2,*)	Ending i mesh index (cell face number).
mobs(3,*)	Beginning j mesh index (cell face number).
mobs(4,*)	Ending j mesh index (cell face number).
mobs(5,*)	Beginning k mesh index (cell face number).
mobs(6,*)	Ending k mesh index (cell face number).
mobs(7,*)	Block number (must be 1 for GASFLOW-MPI).
mobs(8,*)	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 asterisk (*) should be replaced by an integer that identifies the particular **mobs** definition (≤ 3000). The elements in the array **mobs** have the same meaning as those in **walls**, except for the last element, which is explained in Section 1, 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 orthogonal to each other. The following two **mobs** definitions specify obstacle regions:

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

GASFLOW supports 500 definitions for walls and 3000 definitions for mobs.

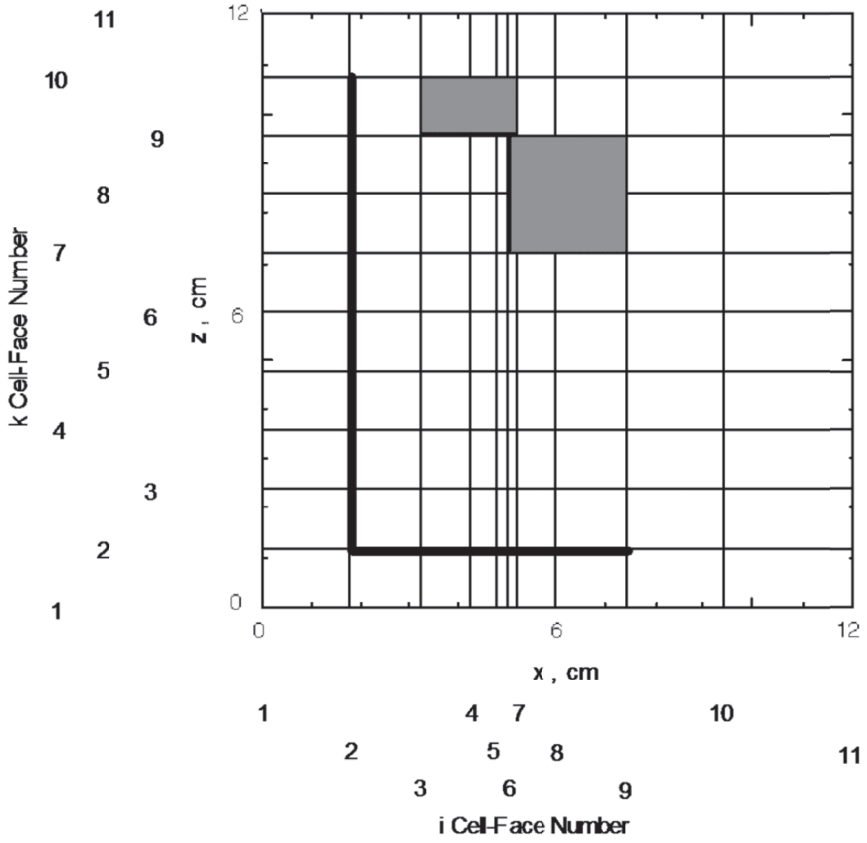


Figure 3-4 Two-dimensional *x-z* view of a mesh containing two wall surfaces and two obstacle regions generated by walls and mobs input definitions, respectively.

3.3.3 Geometric Modeler

GASFLOW-MPI includes a generalized built in geometric modeler to help develop complex geometrical obstacles and wall shapes. Included are all real quadric surfaces and additionally toroidal bodies of revolution about the z-axis. There are capabilities of coupling the GASFLOW-MPI geometric modeler to initial and boundary conditions through **gasdef** statements (see Section 5.1 below), which allows the user to specify complex initial conditions in a relatively easy manner.

Our generalized quadratic equation in three space variables is

$$F = a\cdot(x-x_o)^2 + b\cdot(y-y_o)^2 + c\cdot(z-z_o)^2 +$$

$$2\cdot f\cdot(y-y_o)\cdot(z-z_o) + 2\cdot g\cdot(z-z_o)\cdot(x-x_o) + 2\cdot h\cdot(x-x_o)\cdot(y-y_o) +$$
$$2\cdot p\cdot(x-x_o) + 2\cdot q\cdot(y-y_o) + 2\cdot r\cdot(z-z_o) + 2\cdot s\sqrt{(x-x_0)^2+(y-y_0)^2} + \qquad \text{Equ.3-1}$$

$$2\cdot t\sqrt{(x-x_0)^2+(z-z_0)^2} + 2\cdot u\sqrt{(y-y_0)^2+(z-z_0)^2} + d$$

where the input for this GASFLOW geometric modeler is accomplished in the NAMELIST input block XPUT. The input variable array is **geomodel(i,*)**, where i = 1,26, and is defined for the n^{th} **geomodel** where (*) is replaced by an integer that identifies the particular n^{th} **geomodel** definition as follows:

geomodel(1,*) F, the general quadratic equation defined by Equ.3-1
F > 0, obstacles outside
F = 0, walls define surface
F < 0, obstacles inside

geomodel(2,*) Block number that curve is applied (must be 1 for GASFLOW-MPI).

geomodel(3,*) Flag for heat transfer
For obstacles: = **mobs(8,*)** (See Section 3.3.2)
For walls: = **walls(8,*)** (See Section 3.3.1)

geomodel(4,*) x_o in Equ.3-1

geomodel(5,*) y_o in Equ.3-1

geomodel(6,*) z_o in Equ.3-1

geomodel(7,*) a in Equ.3-1

geomodel(8,*) b in Equ.3-1

geomodel(9,*) c in Equ.3-1

geomodel(10,*) d in Equ.3-1

geomodel(11,*) f in Equ.3-1, default = 0

geomodel(12,*) g in Equ.3-1, default = 0

geomodel(13,*) h in Equ.3-1, default = 0

geomodel(14,*) p in Equ.3-1, default = 0

geomodel(15,*) q in Equ.3-1, default = 0

geomodel(16,*) r in Equ.3-1, default = 0

geomodel(17,*) s in Equ.3-1, default = 0

geomodel(18,*) t in Equ.3-1, default = 0

geomodel(19,*) u in Equ.3-1, default = 0

geomodel(20,*) lower x limiter, default = -10^{+50}

geomodel(21,*) upper x limiter, default = $+10^{+50}$

geomodel(22,*) lower y limiter, default = -10^{+50}

geomodel(23,*) upper y limiter, default = $+10^{+50}$

geomodel(24,*) lower z limiter, default = -10^{+50}

geomodel(25,*) upper z limiter, default = $+10^{+50}$

geomodel(26,*) = 0 (default) activates geometric modeler to define obstacles and walls within the computational mesh.
> 0 couples the geometric modeler to the **gasdef** input variable, where the value refers to the **gasdef** statement number.

The idea behind the limiters is that the quadratic function Equ.3-1 is only constructed between the limits:

geomodel(20,*) < x < **geomodel(21,*)**
geomodel(22,*) < y < **geomodel(23,*)**
geomodel(24,*) < z < **geomodel(25,*)**

GASFLOW-MPI is currently limited to 50 total geomodel statements.

3.3.3.1 Torus Model

When a curve such as a circle is revolved about a line lying in the same plane as the circle, the surface obtained is a circular torus or torus of revolution (Figure 3-5). Let d be the radius of the revolving circle and let D be the distance from its center to the axis of rotation, then the equation for a translated circle in the x-z plane, shown at the top of Figure 3-5 is

$$F=(x-D)^2+z^2-d^2 \qquad \text{Equ. 3-2}$$

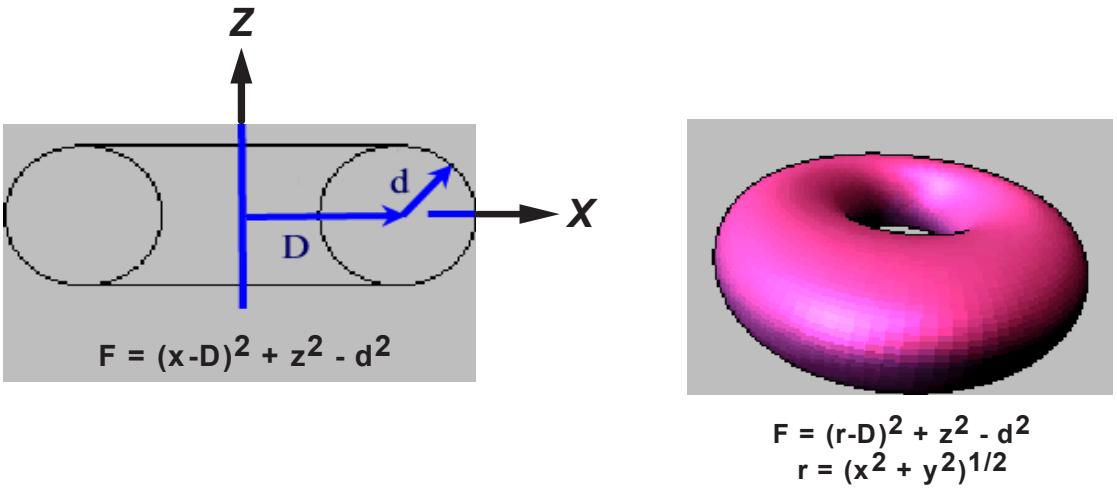


Figure 3-5 A torus of revolution

When we revolve this translated circle around the z-axis, we rewrite Equ. 3-2 as

$$F=(r-D)^2+z^2-d^2 \qquad \text{Equ. 3-3}$$

where

$$r=\sqrt{x^2+y^2} \qquad \text{Equ. 3-4}$$

Equ. 3-3 then becomes

$$F=x^2+y^2+z^2-2D\sqrt{x^2+y^2}+\left(D^2-d^2\right) \qquad \text{Equ. 3-5}$$

It's clear that for a torus, when equating $s = -D$ and d in Equ.3-1 is $\left(D^2 - d^2\right)$ as shown in Equ. 3-5. The surface area of the torus is $4\,\pi^2 Dd$, and its volume is $2\,\pi^2 Dd^{\,2}$.

We present two examples of GASFLOW generated Toroids using the geometric modeler input variable array **geomodel**. In the first example, we will generate a torus in Cartesian geometry and in the second example, we will generate the same torus in cylindrical geometry. At any rate, we present a full torus with radius, d = 50 cm, and the distance from the torus center to the axis of rotation, D = 150 cm.

3.3.3.2 Cartesian Coordinates

Included here is a listing of the relevant input parameters and variables for the GASFLOW generated torus in a Cartesian Coordinate computational domain:

```
$xput
   cyl                 = 0.0, ; Cartesian Coordinates
   geomodel(1:26,1)  =-1.0, 1.0, 0.0, 0.0, 0.0, 000.0,
                       1.0, 1.0, 1.0, 20000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -150.0, 0.0, 0.0,
                       -1.0e+50, +1.0e+50, -1.0e+50, +1.0e+50, -1.0e+50, +1.0e+50,  0,
   gasdef(1:40,1)      =  1,'im1',  1,'jm1',  1,'km1',  1,  1.000e+06, 413.15, 2,
                       0.,  0., 'n2', 0.790, 'o2', 0.210, 'h2', 0.000, 22*0.0,
$end

$meshgn
   iblock   = 1,
   nkx      = 1,
   xl(1)    = -250.0, xc(1) = -250.0, nxl(1) = 0, nxr(1) = 50,  dxmn(1) = 9999.,
   xl(2)    = 250.0,
   ky       = 1,
   yl(1)    = -250.0, yc(1) = -250.0, nyl(1) = 0, nyr(1) = 50,  dymn(1) = 9999.,
   yl(2)    = 250.0,
   nkz      = 1,
   zl(1)    = -100.0, zc(1) = -100.0, nzl(1) = 0, nzr(1) = 20,  dzmn(1) = 9999.,
   zl(2)    = 100.0,
$end
```

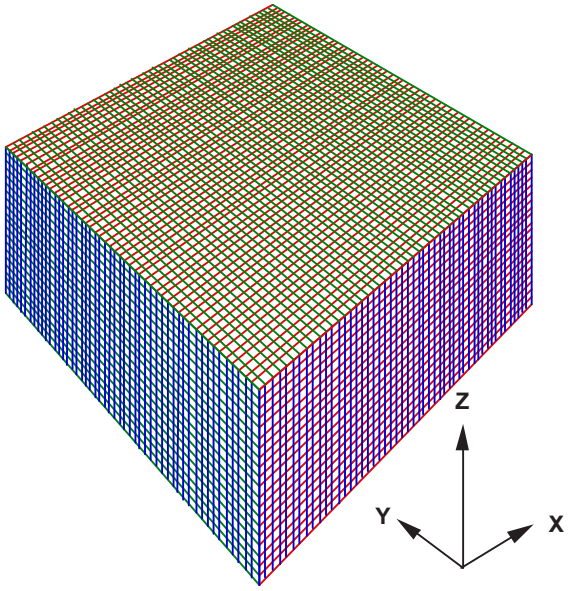


Figure 3-6 Computational block used to generate torus in Cartesian Coordinates.

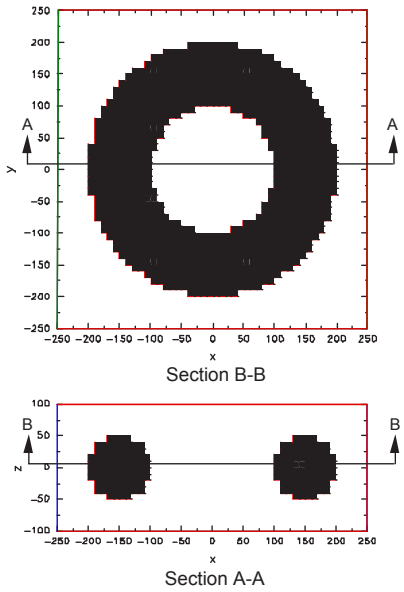


Figure 3-7 Generated torus in Cartesian Coordinates.

3.3.3.3 Cylindrical Coordinates

Included here is a listing of the relevant input parameters and variables for the GASFLOW generated torus in a Cylindrical Coordinate computational domain:

```
$xput
    cyl                 = 1.0, ; Cylindrical Coordinates
    geomodel(1:26,1)  =-1.0, 1.0, 0.0, 0.0, 0.0, 000.0,
                        1.0, 1.0, 1.0, 20000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -150.0, 0.0, 0.0,
                        -1.0e+50, +1.0e+50, -1.0e+50, +1.0e+50, -1.0e+50, +1.0e+50,  0,
    gasdef(1:40,1)      =  1,'im1',  1,'jm1',  1,'km1',  1,  1.000e+06, 413.15, 2,
                        0.,  0., 'n2', 0.790, 'o2', 0.210, 'h2', 0.000, 22*0.0,
$end

$meshgn
    iblock   = 1,
    nkx      = 1,
    xl(1)    = 0.0, xc(1) =    0.0, nxl(1) = 0, nxr(1) = 25,  dxmn(1) = 9999.,
    xl(2)     = 250.0,
    nky      = 1,
    yl(1)    = 0.0, yc(1) =    0.0, nyl(1) = 0, nyr(1) = 50,  dymn(1) = 9999.,
    yl(2)    = 360.0,
    nkz      = 1,
    zl(1)    = -100.0, zc(1) = -100.0, nzl(1) = 0, nzr(1) = 20,  dzmn(1) = 9999.,
    zl(2)    = 100.0,
$end
```

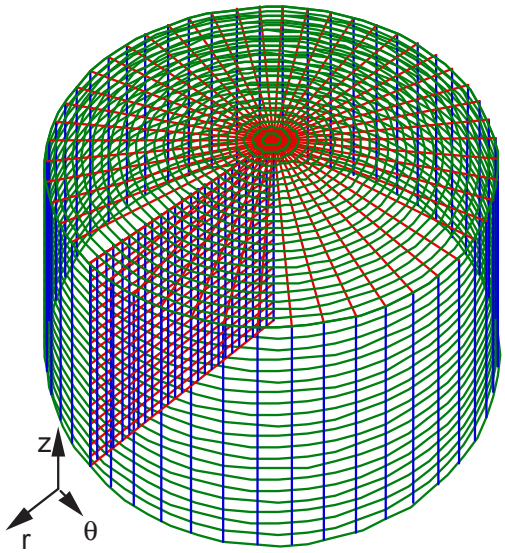


Figure 3-8 Computational block used to generate torus in Cylindrical Coordinates.

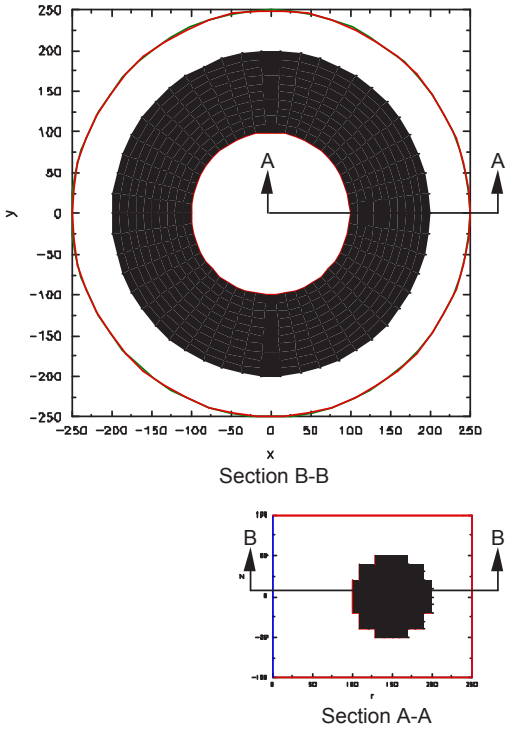


Figure 3-9 Generated torus in Cylindrical Coordinates.

3.3.3.4 General Quadric Surfaces

One can construct a quadric surface having one of the following relationships:

1. Real ellipsoid: $a'x^2 + b'y^2 + c'z^2 = 1$
2. Imaginary ellipsoid: $a'x^2 + b'y^2 + c'z^2 = -1$
3. Hyperboloid of one sheet: $a'x^2 + b'y^2 - c'z^2 = 1$
4. Hyperboloid of two sheets: $a'x^2 + b'y^2 - c'z^2 = -1$
5. Real quadric cone: $a'x^2 + b'y^2 - c'z^2 = 0$
6. Imaginary quadric cone: $a'x^2 + b'y^2 + c'z^2 = 0$
7. Elliptic paraboloid: $a'x^2 + b'y^2 + 2z = 0$
8. Hyperbolic paraboloid: $a'x^2 - b'y^2 + 2z = 0$
9. Real elliptic cylinder: $a'x^2 + b'y^2 = 1$
10. Imaginary elliptic cylinder: $a'x^2 + b'y^2 = -1$
11. Hyperbolic cylinder: $a'x^2 - b'y^2 = 1$
12. Real intersecting planes: $a'x^2 - b'y^2 = 0$

13. Imaginary intersecting planes: $a'x^2 + b'y^2 = 0$

14. Parabolic cylinder: $x^2 + 2y = 0$

15. Real parallel planes: $x^2 = 1$

16. Imaginary parallel planes: $x^2 = -1$

17. Coincident planes: $x^2 = 0$

Relationships 2, 6, 10, and 16 have no real solutions, so they don't describe surfaces in real three-dimensional space, andEqu. 3-16 is of very little interest. However, GASFLOW-MPI is able to automatically generate all curves except 2, 6, 10, 16 and 17. This allows us to easily represent any of the curves with respect to a reference point (x_0,y_0,z_0) in the GASFLOW mesh. Another aspect of this relationship, Equ.3-1, is F. For example, when F > 0, we fill the outside of desired curve with obstacles as shown for the real ellipsoid in Figure 3-10.

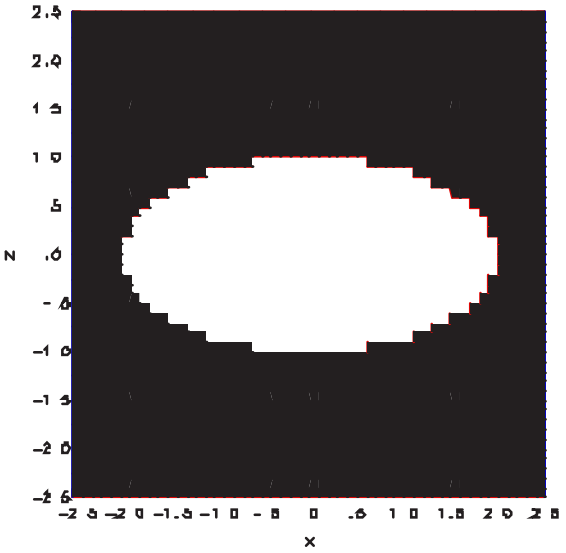


Figure 3-10 A real ellipsoid centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F > 0.0.

The GASFLOW-MPI geometric modeler requires the following input for this curve (Figure 3-10):

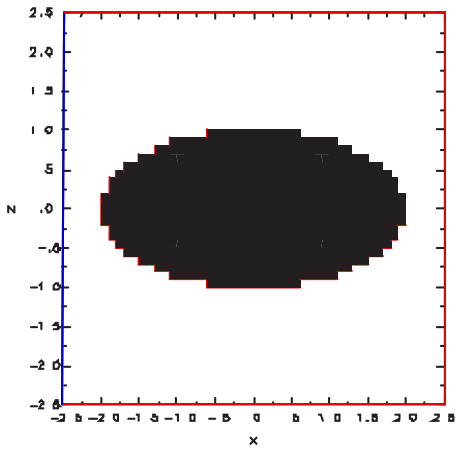
geomodel(1:10,1) = +1.0, 1.0, 1.0, 0.0, 0.0, 0.0,0.25, 1.0, 1.0,-1.0,

When F < 0, we fill the inside of desired curve with obstacles as shown for the real ellipsoid in Figure 3-11, and when F = 0, we construct the desired curve with walls as shown for the real ellipsoid in Figure 3-12. The GASFLOW geometric modeler requires the following input for this curve (Figure 3-11):

geomodel(1:10,1) = -1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.25, 1.0, 1.0,-1.0,

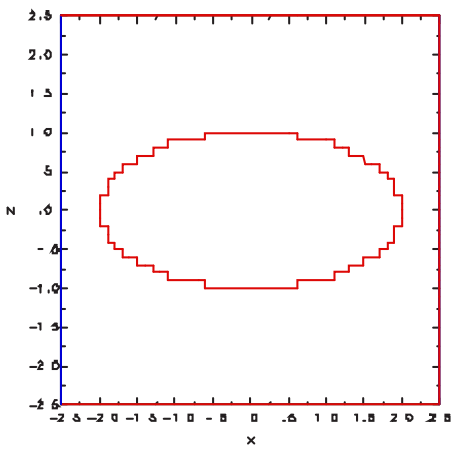
The GASFLOW-MPI geometric modeler requires the following input for this curve (Figure 3-12):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.25, 1.0, 1.0,-1.0,



$F = 0.25x^2 + y^2 + z^2 - 1 < 0$

Figure 3-11 A real ellipsoid centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F < 0.0.



$F = 0.25x^2 + y^2 + z^2 - 1 = 0$

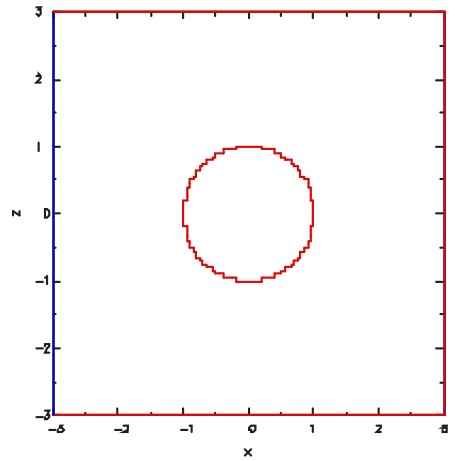
Figure 3-12 A real ellipsoid centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.

The GASFLOW-MPI geometric modeler requires the following input for this curve (Figure 3-13):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,-1.0,

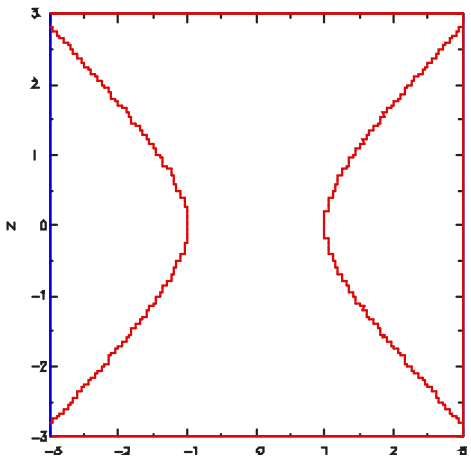
The GASFLOW-MPI geometric modeler requires the following input for this curve (Figure 3-14):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 1.0,-1.0,-1.0,



$F = x^2 + y^2 + z^2 - 1 = 0$

Figure 3-13 A real ellipsoid (sphere in this case) centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.



$F = x^2 + y^2 - z^2 - 1 = 0$

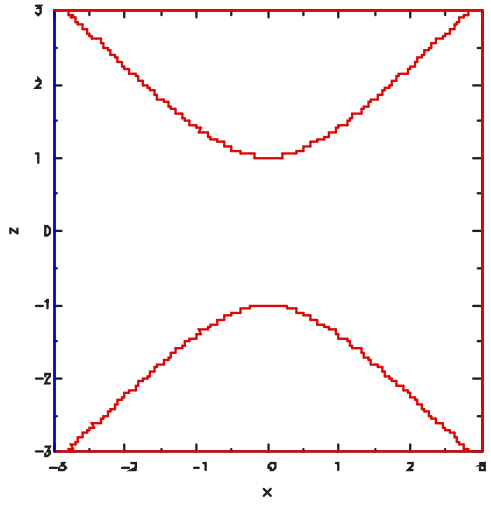
Figure 3-14 A Hyperboloid of one sheet centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.

The GASFLOW-MPI geometric modeler requires the following input for this curve (Figure 3-15):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0,-1.0, 1.0,

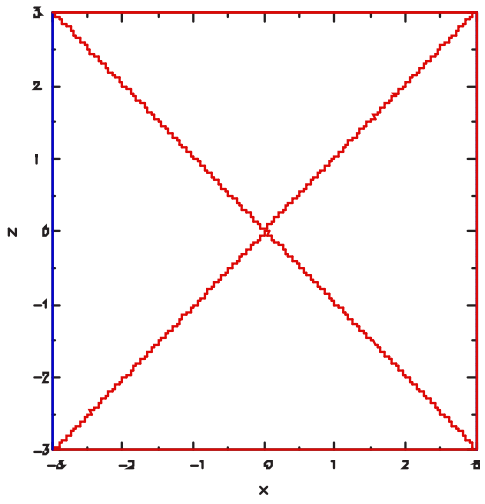
The GASFLOW geometric modeler requires the following input for this curve (Figure 3-16):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 1.0,-1.0, 0.0,



$F = x^2 + y^2 - z^2 + 1 = 0$

Figure 3-15 A Hyperboloid of two sheet centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.



$F = x^2 + y^2 - z^2 = 0$

Figure 3-16 A real quadric cone centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.

The GASFLOW geometric modeler requires the following input for this curve (Figure 3-17):

geomodel(1:16,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0,-1.0,

The GASFLOW geometric modeler requires the following input for this curve (Figure 3-18):

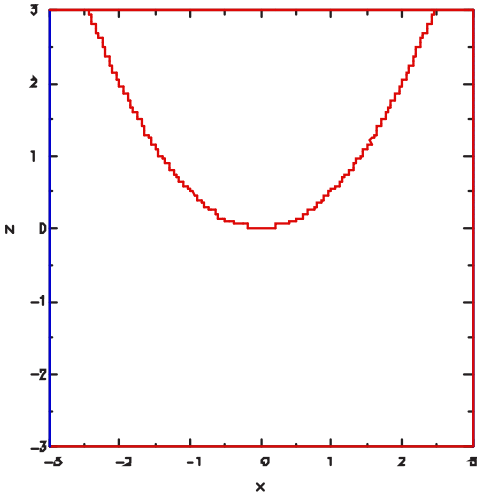
geomodel(1:16,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0,-1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0,-1.0,

The GASFLOW geometric modeler requires the following input for this curve (Figure 3-19):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 0.0, 1.0,-1.0,

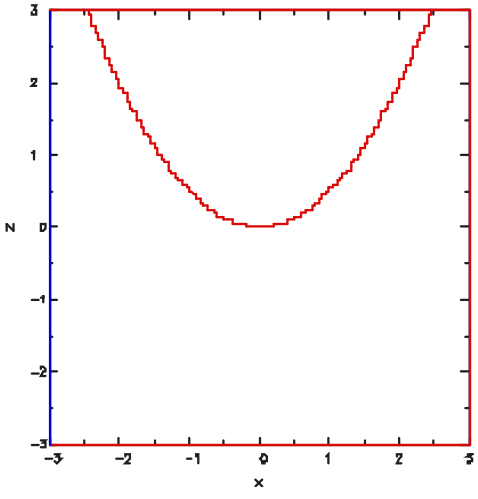
The GASFLOW geometric modeler requires the following input for this curve (Figure 3-20):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 0.0,-1.0,-1.0,



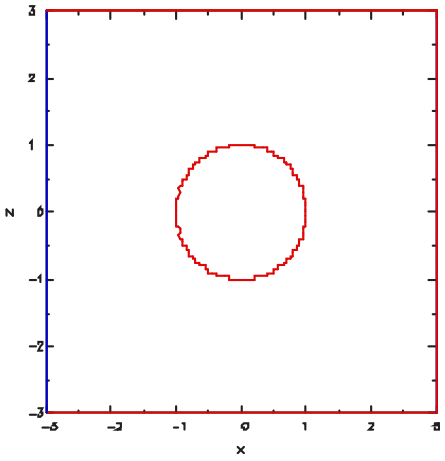
F = x2 + y2 - 2z = 0

Figure 3-17 An elliptic paraboloid centered at x_O = 0,0, y_O = 0,0, and z_O =0,0, where F = 0.0.



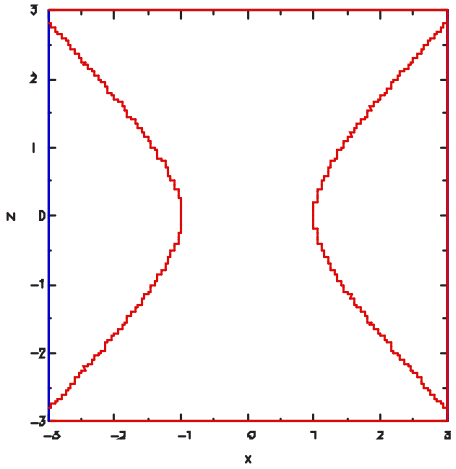
F = x^2 - y^2 - 2z = 0

Figure 3-18 A hyperbolic paraboloid centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.



F = x^2 + z^2 - 1 = 0

Figure 3-19 A real elliptic cylinder (right circular cylinder in this case) centered at x_O = 0,0, y_O = 0,0, and zo = 0,0, where F = 0.0.



F = x^2 - z^2 - 1 = 0

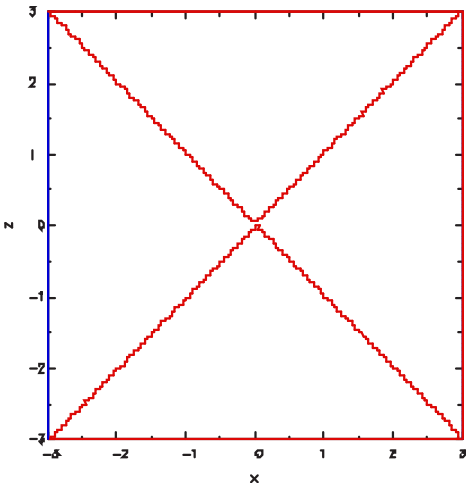
Figure 3-20 A hyperbolic cylinder centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.

The GASFLOW geometric modeler requires the following input for this curve (Figure 3-21):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 0.0,-1.0, 0.0,

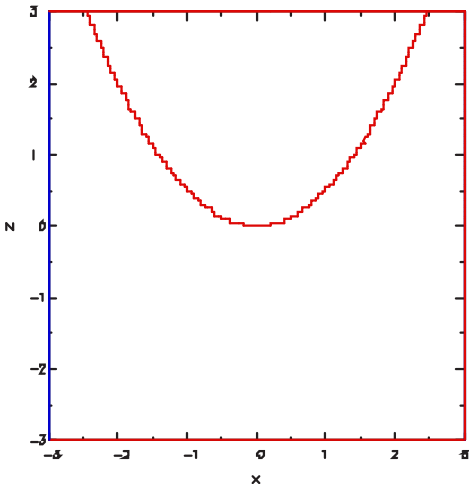
The GASFLOW geometric modeler requires the following input for this curve (Figure 3-22):

geomodel(1:16,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,-1.0,



$F = x^2 - z^2 = 0$

Figure 3-21 Real intersecting planes centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.



$F = x^2 - 2z = 0$

Figure 3-22 A parabolic cylinder centered at x_O = 0,0, y_O = 0,0, and z_O = 0,0, where F = 0.0.

The GASFLOW-MPI geometric modeler requires the following input for this curve (Figure 3-23):

geomodel(1:10,1) = 0.0, 1.0, 1.0, 0.0, 0.0, 0.0,1.0, 0.0, 0.0,-1.0,

More capabilities are shown in Figure 3-24 to

Figure 3-26 where building complex geometries with little effort on the part of the user is demonstrated. Figure 3-24 demonstrates how a spherical annulus geometry can be generated using only two input statements. Figure 3-25 provides a computational volume consisting of a hemi-spherical cap, a cylindrical mid-section, and an hemi-elliptical base with three input statements.

Figure 3-26 builds upon the geometric volume constructed in Figure 3-24 by including a central cylinder, two square obstacles created by using spherical objects and invoking the limiters, and a kind of a tear drop object constructed with a hemi-sphere and a right circular cone.

The GASFLOW-MPI geometric modeler requires the following input for these objects (Figure 3-24):

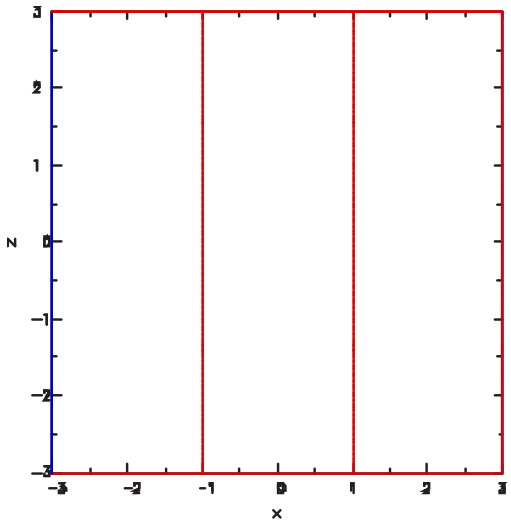
geomodel(1:10,1) = -1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,-4.0, ; solid sphere
geomodel(1:10,2) = +1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,-9.0,; hollow sphere

The GASFLOW-MPI geometric modeler requires the following input for these objects (Figure 3-25):

geomodel(1:26,1) = +1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,-9.0, 0.0, 0.0, ; hemi-sphere
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -3.0,+3.0,-3.0,+3.0, 0.0,+3.0, 0.0,
geomodel(1:26,2) = +1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0,-9.0, 0.0, 0.0,; cylinder
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,-3.0,+3.0,-3.0,+3.0,-2.0, 0.0, 0.0,
geomodel(1:26,3) = +1.0, 1.0, 1.0, 0.0, 0.0,-2.0, .11, .11, 1.0,-1.0, 0.0, 0.0, ; hemi-ellipse
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,-3.0,+3.0,-3.0,+3.0,-3.0,-2.0, 0.0,

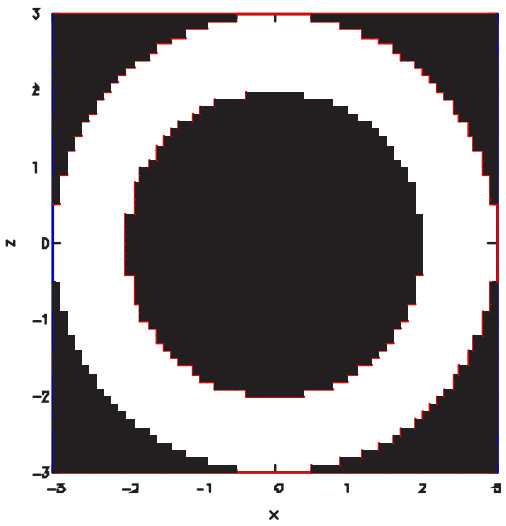
The geometric modeler requires the following input for these objects (Figure 3–26):

geomodel(1:26,1) = +1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,-9.0, 0.0, 0.0,; hemi-sphere
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,-3.0,+3.0,-3.0,+3.0, 0.0,+3.0, 0.0,
geomodel(1:26,2) = +1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0,-9.0, 0.0, 0.0,; cylinder
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, , 0.0,-3.0,+3.0,-3.0,+3.0,-2.0, 0.0, 0.0,
geomodel(1:26,3) = +1.0, 1.0, 1.0, 0.0, 0.0,-2.0, .11, .11, 1.0,-1.0, 0.0, 0.0,;hemi-ellipse
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, , 0.0, -3.0,+3.0,-3.0,+3.0,-3.0,-2.0, 0.0,
geomodel(1:26,4) = -1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0,-0.5, 0.0, 0.0,; solid cylinder
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,-3.0,+3.0,-3.0,+3.0,-3.0,-1.0, 0.0,
geomodel(1:26,5) = -1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0,-1.0, 0.0, 0.0,; solid hemi-sphere
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -3.0,+3.0,-3.0,+3.0,-3.0, 1.0, 0.0,
geomodel(1:26,6) = -1.0, 1.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0,-1.0, 0.0, 0.0, 0.0,; solid cone
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -3.0,+3.0,-3.0,+3.0, 1.0, 2.0, 0.0,
geomodel(1:26,7) = -1.0, 1.0, 1.0, 2.0, 0.0,-1.0, 1.0, 1.0, 1.0,-4.0, 0.0, 0.0,; cube
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.5, 2.5,-0.5,+0.5,-1.5,-0.5, 0.0,
geomodel(1:26,8) = -1.0, 1.0, 1.0,-2.0, 0.0,-1.0, 1.0, 1.0, 1.0,-4.0, 0.0, 0.0,; cube
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -2.5,-1.5,-0.5,+0.5,-1.5,-0.5, 0.0,



$F = x^2 - 1 = 0$

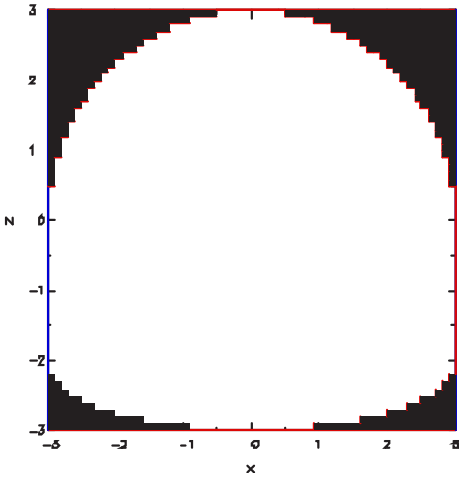
Figure 3-23 Real parallel planes centered at x_o = 0,0, y_o = 0,0, and z_o = 0,0, where F = 0.0.



F1 = x2 + y2 + z2 -4 < 0

F2 = x2 + y2 + z2 -9 > 0

Figure 3-24 Spherical annulus centered at x_o = 0,0, y_o = 0,0, and z_o = 0,0, where F_1 < 0.0 and F_2 > 0.0.

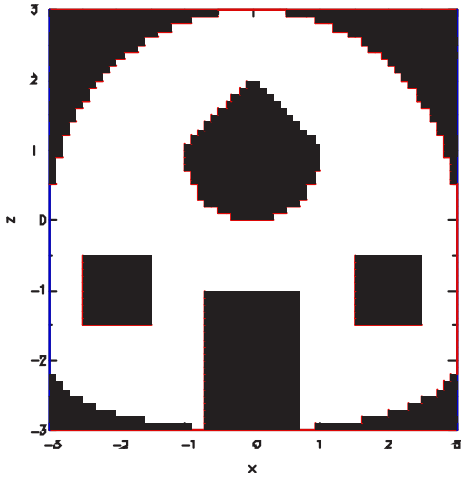


$F_1 = x^2 + y^2 + z^2 - 9 > 0;\ z > 0$

$F_2 = x^2 + y^2 - 9 > 0;\ -2 < z < 0$

$F_3 = x^2/9 + y^2/9 + (z+2)^2 - 1 > 0\ ;\ z < -2$

Figure 3-25 Volume with hemi-spherical cap, cylindrical mid-section, and hemi-elliptic bottom.



$F_1 = x^2 + y^2 + z^2 - 9 > 0\ ;\ z > 0$

$F_2 = x^2 + y^2 - 9 > 0\ ;\ -2 < z < 0$

$F_3 = x^2/9 + y^2/9 + (z+2)^2 - 1 > 0\ ;\ z < -2$

$F_4 = x^2 + y^2 - 0.5 < 0\ ;\ -3 < z < -1$

$F_5 = x^2 + y^2 + (z-1)^2 - 1 < 0\ ;\ 0 < z < 1$

$F_6 = x^2 + y^2 - (z-2)^2 < 0\ ;\ 1 < z < 2$

$F_7 = (x-2)^2 + y^2 + (z+1)^2 - 4 < 0\ ;$

$1.5 < x < 2.5,\ -0.5 < y < 0.5,\ -1.5 < z < -0.5$

$F_8 = (x+2)^2 + y^2 + (z+1)^2 - 4 < 0\ ;$

$-2.5 < x < -1.5,\ -0.5 < y < 0.5,\ -1.5 < z < -0.5$

Figure 3-26 Volume with hemi-spherical cap, cylindrical mid-section, hemi-elliptic bottom, and some internals of various shapes.

3.3.3.5 Coupling the geomodel and gasdef input variables

It is convenient at times to be able to easily input complex initial conditions. To this end, we have generalized the use of the **gasdef** input variable (see Section 5.1) by coupling it to the **geomodel** input variable. As the user will see in Section 5.1, usually the **gasdef** statement specifies fluid conditions in rectangular regions in two space dimensions and hexahedral volumes in three space dimensions. However, as we demonstrate in the following example, we are now able to input any initial conditions that are consistent with the GASFLOW-MPI geometric modeler. In the two space dimensions shown in Figure 3-27, we input a quarter circle with radius 500 cm of stoichiometric hydrogen-air concentrations in an air medium. Actually this analysis is based a three-dimensional problem involving a hemispherical balloon with radius 300 cm, but for the purposes of display, we have reduced it to two-space dimensions by collapsing the y-coordinate dimension and increased the stoichiometric hydrogen-air concentrations to 500 cm.

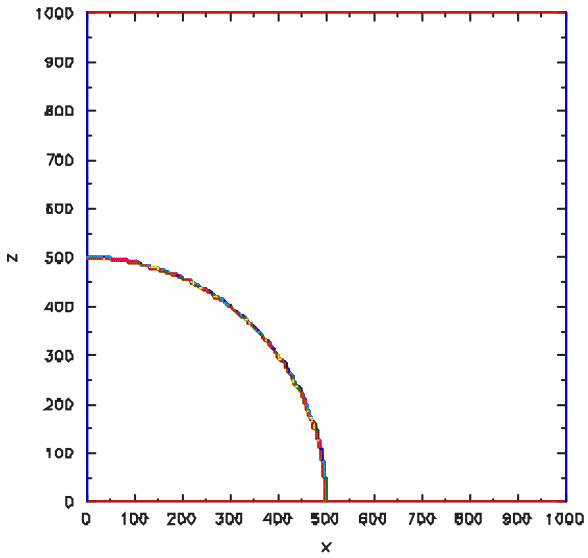


Figure 3-27 Example of coupling the geomodel and gasdef input variables to define complex initial conditions. In this case we define a quarter circle with radius 500 cm of stoichiometric hydrogen-air concentrations in an air medium.

```
$xput
    cyl = 0.0, ; Cartesian Coordinates
    geomodel(1:26,1) =-1.0, 1.0, 0.0, 0.0, 0.0, 0.0,
                      1.0, 1.0, 1.0, -25.0e+04, 0.0, 0.0, 0.0, 0.0, 0.0,0.0,0.0, 0.0,0.0,
                      1.0e+50,+1.0e+50,-1.0e+50,+1.0e+50,-1.0e+50,+1.0e+50, 2,
    mat                = 'h2', 'h2o', 'n2', 'o2',
    gasdef(1:40,1)     = 1, 'imax', 1, 'jmax', 1,'kmax', 1, 1.015e+06, 300.00, 2,
                      0., 0., 'n2', 0.79, 'o2', 0.21, 24*0.0,
    gasdef(1:40,2)     = 1, 2, 01,  02, 1,  2, 1, 1.015e+06, 300.00, 2,
                      0., 0., 'h2', 0.29, 'n2', 0.565, 'o2', 0.145, 22*0.0,
$end

$meshgn
    iblock = 1,
    nkx    = 1,
    xl(1)   = 000.0,xc(1) = 000.0,nxl(1) = 0,nxr(1) =200, dxmn(1) = 9999.,
    xl(2)  = 1000.0,
    nky    = 1,
    yl(1)   = 000.0,yc(1) = 000.0,nyl(1) = 0,nyr(1) = 1, dymn(1) = 9999.,
    yl(2)  =  10.0,
    nkz    = 1,
    zl(1)  = 000.0, zc(1) = 000.0, nzl(1) = 0,nzr(1) =200, dzmn(1) = 9999.,
    zl(2)  = 1000.0,
$end
```

In connection with activating the geometric modeler, there are two XPUT namelist variables that the user should be aware. Because the geometric modeler can generate copious numbers of walls and obstacles, it could be that GASFLOW-MPI may not predict and therefore allocate sufficient memory arrays for all the heat transfer surfaces. If that is the case, GASFLOW-MPI will stop and tell the user that he should increase the size of either or both **nobsgeo** and/or **nwallsgeo**. These two variables are respectively an additional amount of memory allocation required because of the geometric modeler for obstacles and walls, respectively. Remember, GASFLOW-MPI will stop and tell the user to increase the value of these variables and will give an indication of how much one should increase them.

3.3.4 Holes

In order to model complicated geometries, GASFLOW-MPI has a generalized input to represent *obstacles* (solid structures which eliminate fluid cells in the computational mesh) and *walls* (which provide a zero flux condition on any computational fluid face). These surfaces described by *obstacles* and *walls* may, at the user's discretion, provide two-phase heat-transfer regions within the computational mesh. It is convenient to add another geometry modeling capability, and we call this input *holes*. This allows regions that have been removed as fluid cells by using *obstacles* to be set back to fluid cells. It must be noted that all *obstacles* (**mobs**) and *walls* (**walls**) are processed first, and then the *holes* (**holes**) are positioned. In the current version of GASFLOW-MPI, the number of *holes* is limited to the PARAMETER variable MXA (MXA=300). Each *hole* is defined by thirteen entries in the *holes* input array in the NAMELIST input block xput. The *holes* input variable array, [**holes(i,*)**, i = 1,13], is defined for the n^{th} hole where (*) is replaced by an integer that identifies the particular n^{th} hole definition:

holes(1,*) Beginning i mesh index (cell face number).

holes(2,*) Ending i mesh index (cell face number).

holes(3,*) Beginning j mesh index (cell face number).

holes(4,*) Ending j mesh index (cell face number).

holes(5,*) Beginning k mesh index (cell face number).

holes(6,*) Ending k mesh index (cell face number).

holes(7,*) Block number (must be 1 for GASFLOW-MPI).

holes(8,*) Flag for fluxing on the beginning i mesh face [**holes(1,*)**]:
= -1 implies mixed fluxing condition allowed;
= 0 implies no-fluxing condition allowed;
= +1 implies fluxing condition allowed.

holes(9,*) Flag for fluxing on the ending i mesh face [**holes(2,*)**]:
= -1 implies mixed fluxing condition allowed;
= 0 implies no-fluxing condition allowed;
= +1 implies fluxing condition allowed.

holes(10,*) Flag for fluxing on the beginning j mesh face [**holes(3,*)**]:
= -1 implies mixed fluxing condition allowed;
= 0 implies no-fluxing condition allowed;
= +1 implies fluxing condition allowed.

holes(11,*) Flag for fluxing on the ending j mesh face [**holes(4,*)**]:
= -1 implies mixed fluxing condition allowed;
= 0 implies no-fluxing condition allowed;
= +1 implies fluxing condition allowed.

holes(12,*) Flag for fluxing on the beginning k mesh face[**holes(5,*)**]:
= -1 implies mixed fluxing condition allowed;
= 0 implies no-fluxing condition allowed;
= +1 implies fluxing condition allowed.

holes(13,*) Flag for fluxing on the ending k mesh face [**holes(6,*)**]:
= -1 implies mixed fluxing condition allowed;
= 0 implies no-fluxing condition allowed;
= +1 implies fluxing condition allowed.

The asterisk (*) should be replaced by an integer that identifies the particular holes definition (≤ 300).

An example of the input for **holes** follows:

Consider a computational mesh, i.e., *block* 1, that has 10 fluid cells in each of the three coordinate directions *x*, *y*, and *z*. This means that including the boundary cells, there are 12 cells in each coordinate direction. There is an obstacle block that penetrates the entire computing domain that is 3 cells high, 7 cells wide, and 10 cells deep made of material 1; there is a horizontal wall (ceiling) 8 cells wide, 2 cells above the bottom boundary, and 10 cells deep made of material 2; and there is a vertical wall 4 cells high, 1 cell from the west or left side boundary, and 10 cells deep made of material 2. The input is as follows:

```
$xput
   …
   mobs = 3,10,  1,11,  5, 8,  1,  1,  ; solid obstacle
   walls = 2,10,  1,11,  3, 3,  1,  2,  ; horizontal wall
           2, 2,  1,11,  6,10,  1,  2,  ; vertical wall
   …
$end
```

This configurations is plotted as shown in Figure 3-28. We wish to position 3 holes in the existing geometric model:

1. a hole 2 vertical cells high penetrating the obstacle from the top,
2. a vertical hole penetrating through the entire obstacle, and
3. a hole penetrating the horizontal wall.

This new configuration is shown in Figure 3-29. The entire input stream follows:

```
$xput
   …
   mobs =  3,10,  1,11,  5, 8,  1,  1,  ; solid obstacle
   walls =  2,10,  1,11,  3, 3,  1,  2,  ; horizontal wall
            2, 2,  1,11,  6,10,  1,  2,  ; vertical wall
   holes =  5, 7,  4, 7,  6, 8,  1,0,0,0,0,0,1,; top hole
            8, 9,  5, 6,  5, 8,  1,0,0,0,0,1,1,; thru hole
            8, 9,  5, 6,  2, 4,  1,1,1,1,1,1,1,; wall hole
   …
$end
```

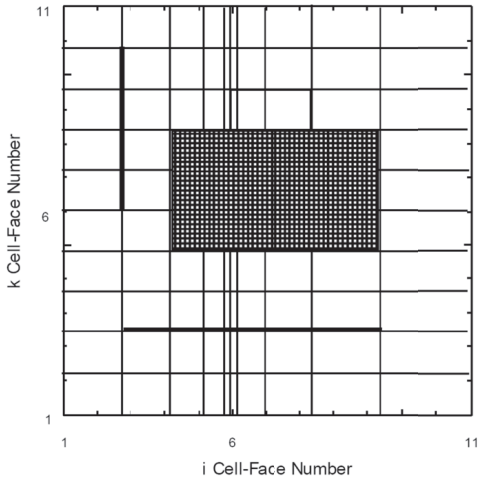


Figure 3-28 A geometric model with one obstacle and 2 walls to demonstrate the holes option

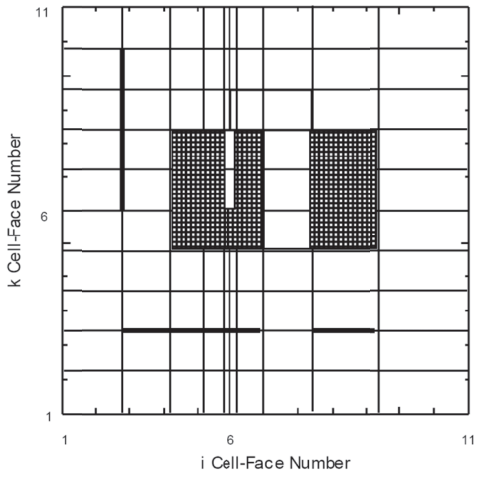
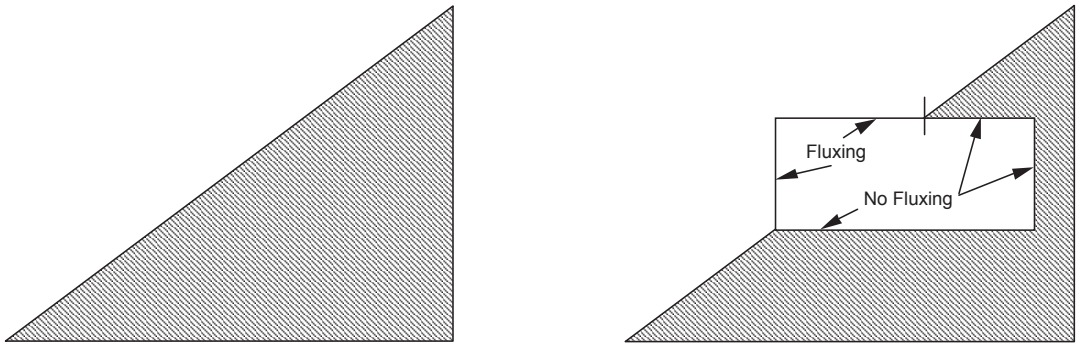


Figure 3-29 A geometric model with one obstacle (mobs) and 2 walls (walls) with 3 penetrations cut through the j=6 plane (*y*-plane cut) to demonstrate the holes option

Coupling the **holes** and **geomodel** statements is best served with an example. In Figure 3-30 we display an obstacle generated by the **geomodel** function. In Figure 3-30, we show the same obstacle with a hole generated with the **holes** input variable. The **holes** input variable, as applied to any given surface must be able to specify more than either a fluxing boundary condition or a no fluxing boundary condition. As seen in this example, a single **holes** statement is unable to define the hole in the appropriate manner because the upper surface has a mixed condition where part of the surface is no fluxing and part of it fluxing.

The mixed surface fluxing condition is invoked with a -1 input for the fluxing condition for the specified surfaces in the **holes** statement, which means entry locations 8 through 13 for respectively beginning i face, ending i face, beginning j face, ending j face, beginning k face, and ending k face. Otherwise, the pure no-flux surface condition is input as 0 (zero) and the pure fluxing surface condition is input as 1.



a. Obstacle generated with the **geomodel** statement

b. Hole in obstacle generated with the **holes** statement

Figure 3-30 Example of using the holes input variable with a geomodel generated obstacle.

This means that GASFLOW-MPI examines computational cells with common surfaces to the holes cells and determines if the fluxing or no-fluxing condition is activated. For example, if the common surface cell just outside the hole cell is an obstacle, then the no-fluxing condition is set, whereas if it's a fluid cell, then the fluxing condition is set. In previous versions of GASFLOW, the entire surface was set as fluxing with a +1 input value and no-fluxing with a 0 (zero) input value.

A better example is shown in Figure 3-31, where we present a 20 degree section of a torus who's radius, d = 404 cm, and the distance of the torus center to the axis of rotation, D = 376.2 cm. We are interested in constructing three holes in the outside obstacle part displayed in Figure 3-31. To do that, we use the new mixed surface condition for the **holes** option, and we present the results in Figure 3-32.

We will discuss the details of all changes in the input stream in the following section: however, for completeness, we list the relevant input values that we used to generate this example below. Please note that in this listing the **holes** input arrays have been deactivated with the leading semicolon (;), but to generate the results in Figure 3-32, one only needs to activate these holes statements by removing the leading semicolon.

```
$xput
   cyl                  = 1.0,  ; Cylindrical Geometry
   geomodel(1:26,1) = +1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, -21684,
                        0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -404.0,-1.0e+50, 0.0,0.0,
                        +1.0e+50, -1.0e+50, +1.0e+50, -1.0e+50, +1.0e+50,  0.0,
   gasdef(1:40,1)       = 1,'im1',  1,'jm1',  1,'km1',  1,  1.000e+06, 413.15, 2,
                        0.,  0., 'n2', 0.790, 'o2', 0.210, 'h2', 0.000, 22*0.0,
   ; holes(1:13,01)    = 1, 68,   2,   6, 01, 14,   1,   0, 0,   -1, -1,  0, -1,
   ; holes(1:13,02)    = 1, 61,   2,   6, 24, 38,   1,   0, 0,   -1, -1, -1, -1,
   ; holes(1:13,03)    = 10, 63,   2,   6, 54, 61,   1,  -1, 0,   -1, -1, -1,  0,
   mobs(1:8,1)          = 1,  2,   1, 7,   55, 61,    1, 0,
   mobs(1:8,2)          = 2,  4,   1, 7,   56, 61,    1, 0,
   mobs(1:8,3)          = 4,  5,   1, 7,   57, 61,    1, 0,
   mobs(1:8,4)          = 5,  7,   1, 7,   58, 61,    1, 0,
```

```
    mobs(1:8,5)          = 7,  8,   1, 7,   59, 61,    1, 0,
    mobs(1:8,6)          = 8, 10,   1, 7,   60, 61,    1, 0,
$end

$meshgn
    iblock = 1,
    nkx = 5,
    xl(1) =  404.0, xc(1) =  404.0, nxl(1) = 0, nxr(1) = 40,  dxmn(1) = 9999.,
    xl(2) =  834.0, xc(2) =  834.0, nxl(2) = 0, nxr(2) = 10,  dxmn(2) =  12.,
    xl(3) = 1006.5, xc(3) = 1179.0, nxl(3) =10, nxr(3) =  0,  dxmn(3) =  12.,
    xl(4) = 1179.0, xc(4) = 1179.0, nxl(4) = 0, nxr(4) =  2,  dxmn(4) = 9999.,
    xl(5) = 1204.0, xc(5) = 1204.0, nxl(5) = 0, nxr(5) =  5,  dxmn(5) = 9999.,
    xl(6) = 1274.0,
    nky = 1,
    yl(1) =  0.0,  yc(1) =  0.0,  nyl(1) = 0,  nyr(1) = 6,  dymn(1) = 9999.,
    yl(2) =  20.0,
    nkz = 1,
    zl(1) =  -430.0, zc(1) = -430.0, nzl(1) = 0, nzr(1) = 60,  dzmn(1) = 9999.,
    zl(2) =  430.0,
$end
```

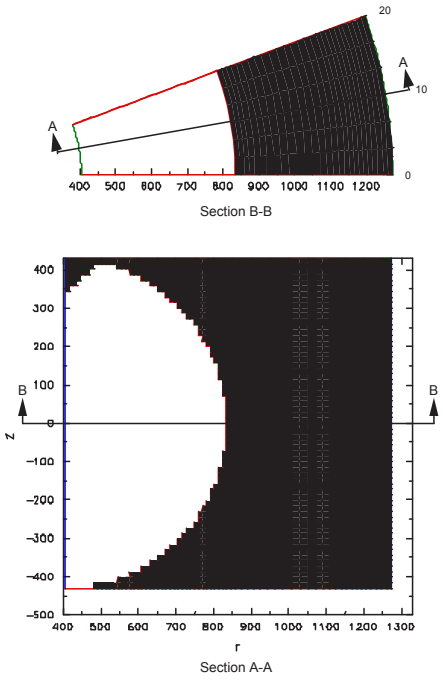


Figure 3-31 Complex example for demonstrating need for mixed fluxing condition using the holes input variable.

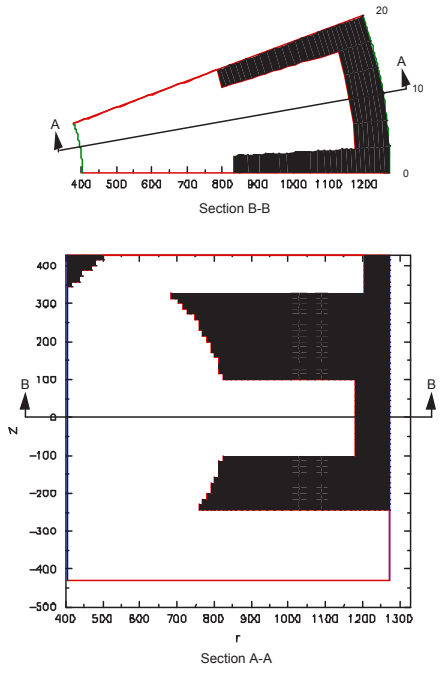


Figure 3-32 Complex example for demonstrating the mixed fluxing condition when using the holes input variable.

3.3.5 Fractional Areas, Flow Resistances, and Sub-grid Mass Flow Rates

Often when modelling complex geometries, the actual flow area is different from the structure that the computational mesh allows. For example, in Figure 3-29, the small volume that we created with a **holes** option in the large obstacle has an inlet flow area that is actually 10% of the flow area shown in the mesh. GASFLOW-MPI has the capability to reduce the flow area and volumes of selected cells. The user can input a fractional area at the cell edge between two adjacent cells and GASFLOW-MPI will automatically include a sharp edge orifice flow loss at that location and calculate velocities based on the actual flow area, or alternatively, the user can activate a sub-grid mass flow rate model. Fractional area reductions, flow resistances, or sub-grid mass flow rates are input with the **areardef** array in the xput NAMELIST group. The first 7 locations in the **areardef(1:7,*)** array are defined in the usual way, where the asterisk (*) is replaced by a sequential integer (≤ 600); that is, GASFLOW-MPI supports 600 definitions for **areardef**.

areardef(1,*) Beginning i mesh index (cell face number).
areardef(2,*) Ending i mesh index (cell face number).
areardef(3,*) Beginning j mesh index (cell face number).
areardef(4,*) Ending j mesh index (cell face number).
areardef(5,*) Beginning k mesh index (cell face number).
areardef(6,*) Ending k mesh index (cell face number).
areardef(7,*) Block number.

With the location

areardef(12,*) > 0 activating a standard flow resistance model
< 0 activating the sub-grid mass flow rate model (default = 1)

For fractional area flow with resistance [**areardef(12,*)** > 0], we review the following:

The drag term is given by

$$\mathbf{D}_d = \frac{1}{2}\mathbf{C}_D(\rho_m \mathbf{u})^A \left|\mathbf{u}^n\right| . \qquad \text{Equ. 3-6}$$

where the drag coefficient for incompressible flows, $\mathbf{C}_D$, is given by

$$\mathbf{C}_\mathrm{D} = \left[1+0.707\bullet(1-\mathbf{A})^{\frac{1}{2}} - \mathbf{A}\right]^2 \qquad \text{Equ. 3-7}$$

which is only valid for an orifice geometry. The development can proceed as follows. Instead of the drag coefficient being only specified as an orifice coefficient, we generalize the treatment by assuming that

$$\mathbf{C}_D = \sum_k \mathbf{C}_k = \mathbf{C}_{form} + \mathbf{C}_{frictional} , \qquad \text{Equ. 3-8}$$

where $\mathbf{C}_{form}$ is "classical" form or pressure loss coefficient given in

$$\mathbf{D}_d = \frac{1}{2}\mathbf{C}_{form}(\rho_m \mathbf{u})^A |\mathbf{u}^n| \qquad \text{Equ. 3-9}$$

and $\mathbf{C}_{frictiona}$ is the Moody loss coefficient

$$\mathbf{C}_{frictional} = f\frac{L}{D_e} \qquad \text{Equ. 3-10}$$

involving the friction factor, f, the length over which the loss occurs, L, and the equivalent diameter, $D_e = 4\frac{Flow\ Area}{Wetted\ Perimeter}$. For laminar flow, the friction factor for pipes is

$$f = \frac{64}{\mathrm{Re}}, \qquad \text{Equ. 3-11}$$

for flow between parallel plates

$$f = \frac{96}{\mathrm{Re}},\ D_e = 2 \cdot Z, \qquad \text{Equ. 3-12}$$

where Z is the distance between the plates, and for rectangular cross section ducts

$$f = \frac{64}{\phi \cdot \mathrm{Re}},\ D_e = \frac{2(Z_1 Z_2)}{Z_1 + Z_2}, \qquad \text{Equ. 3-13}$$

We can generalize the friction factor by writing

$$f = \frac{\phi}{\mathrm{Re}} = \frac{\phi \cdot \mu}{|\mathbf{u}|\rho D_e} \qquad \text{Equ. 3-14}$$

which makes use of the absolute fluid viscosity, μ, the fluid density, ρ, the equivalent diameter as defined before, and ϕ, the geometry factor: 64 for pipes, 96 for parallel plates, and 64/ϕ rectangular channels.

Substituting Equ. 3-14 into Equ. 3-10, then into Equ. 3-8, and finally into Equ. 3-6, yields for the "total" drag function

$$\mathbf{D}_d = \frac{1}{2}\left[\mathbf{C}_{form}|\mathbf{u}| + \frac{\phi \cdot \mu \cdot L}{\rho D_e^2}\right](\rho_m \mathbf{u}) \qquad \text{Equ. 3-15}$$

Note that in this expression, we have both a linear term and a quadratic term in the velocity. The solution procedure in GASFLOW has been extended to account for both of these terms.

With this in mind, we define

areardef(8,*) Fraction of geometric flow area available for fluid flow. If areardef(8,*) is less than zero and areardef(9,*) equals zero , then no flow loss coefficient is included at this location. If areardef(8,*) is less than zero and areardef(10,*) equals zero , then no drag loss is included at this location.

areardef(9,*) User input flow loss coefficient. If areardef(9,*)is zero, then correlations for flow through an orifice are used to calculate the flow loss coefficient. If areardef(8,*) is less than zero and areardef(9,*) equals zero, then no flow loss coefficient is included at this location. Default is zero.

areardef(10,*) Hydraulic diameter to be used for the laminar drag loss. If areardef(10,*) equals zero, then no laminar drag loss is included at this location. Default is zero.

areardef(11,*) Coefficient in the laminar drag loss formula. Default is 64.

areardef(12,*) Fraction of the cell-center distance between neighboring cells to be used in the laminar drag loss formula. Default is one.

There are some implications involving the combination of these input values that we will discuss here. Basically, **areardef(8,*)**, **areardef(9,*)**, and **areardef(10,*)** control the terms in Equ. 3-15. For example, when

1. **1.areardef(8,*)** < 0, **areardef(9,*)** = 0, **areardef(10,*)** = 0
   The flow loss function Equ. 3-15 is identically zero, but GASFLOW solves the conservation equation based upon a reduced area specified by |**areardef(8,*)**|.

2. **areardef(8,*)** /= 0
   areardef(9,*) = 0, then C_{form} is computed based upon the orifice coefficient of Eq. (3-7).
   areardef(9,*) > 0, then C_{form} = **areardef(9,*)**
   areardef(10,*) = 0, then $C_{frictional}$ = 0.
   areardef(10,*) > 0, then $C_{frictional}$ then the default values of
   areardef(11,*) = 64.0
   areardef(12,*) = 1.0

   are used to calculate the frictional flow loss in Equ. 3-10, unless either or both of these input values are changed by the user.

3. **areardef(8,*)** = 0

The resistance of the flow loss approaches infinity, i.e., there will be no flow through such a plane specified by this input value. If the beginning and ending mesh index is the same in for the i, j, or k input, then the flow area fraction is applied only at the cell edge identified by the repeated mesh index. For example, the flow area into the room we created by the **holes** option in Figure 3-29 can be reduced to 10% of the geometric flow area by the following statement:

areardef(1:12,1) = 5,7,4,7,8,8,1,0.1, 4*0.0,

If the user requires that the volume and flow area be reduced, then the beginning and ending mesh indices for i, j, and k must be different. For example, the following input reduces geometric volumes and flow areas by 50% for the hole that we drilled through the entire obstacle in Figure 3-33:

areardef(1:12,2) = 7,8,5,6,5,8,1,0.5, 4*0.0,

The following input reduces geometric volumes and flow areas by 75% for flow through the horizontal wall or ceiling in Figure 3-33:

areardef(1:12,3) = 7,8,5,6,3,3,1,0.75, 4*0.0,

We can create the entire geometric model in Figure 3-33, including the fractional areas and volumes with the following input:

```
$xput
   mobs     = 3,10,  1,11,  5, 8,  1,  1,  ; solid obstacle
   walls    = 2,10,  1,11,  3, 3,  1,  2,  ; horizontal wall
              2, 2,  1,11,  6,10,  1,  2,  ; vertical wall
   holes    = 5, 7,  4, 7,  6, 8,  1,  0,0,  0,0,  0,1, ; top hole
              8, 9,  5, 6,  5, 8,  1,  0,0,  0,0,  1,1, ; thru hole
              8, 9,  5, 6,  2, 4,  1,  1,1,  1,1,  1,1, ; wall hole
   areardef = 5, 7,  4, 7,  8, 8,  1,  0.1, 4*0.0,  ; fract area top hole
              8, 9,  5, 6,  5, 8,  1,  0.5, 4*0.0,  ; fract area thru hole
              8, 9,  5, 6,  3, 3,  1, 0.75, 4*0.0,  ; fract area wall
$end
```

To activate the sub-grid mass flow rate model [**areardef(12,*)** < 0], a frictionless adiabatic solution for this problem is provided as follows:

$$\dot{m} = A \cdot C_d \cdot \begin{cases} \sqrt{2 p_1 \rho_1 \left(\frac{\gamma}{\gamma-1}\right)\left[\left(\frac{p_2}{p_1}\right)^{\frac{2}{\gamma}} - \left(\frac{p_2}{p_1}\right)^{\frac{\gamma+1}{\gamma}}\right]} & ; \ \frac{p_2}{p_1} \geq \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}} \\ \sqrt{\gamma p_1 \rho_1 \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} & ; \ \frac{p_2}{p_1} < \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}} \end{cases} \qquad \text{Equ. 3-16}$$

where $\dot{m}$ is the mass flow rate through the hole with area A and discharge coefficient C_d. Some relevant static pressures on either side of the hole, with p_1 and ρ_1 corresponding to the high pressure and gas density, respectively, are specified, plus the ratio of specific heats γ.

We define the additional variables for the **areardef** input variable:

areardef(8,*) Fraction of defined cell face area open for flow. We normally set this entry < zero with **areardef(9,*)** and **areardef(10,*)** either equal to zero or negative to indicate i,j,k,iblock index packing.

areardef(9,*) Computational index for evaluating Equ. 3-16
= 0.0 for the use of local indexes to compute the mass flow rate.
< 0.0 for the use of i,j,k,iblock index packing to compute the mass flow rate.
See examples below.

areardef(10,*) Computational index for evaluating Equ. 3-16
= 0.0 for the use of local indexes to compute the mass flow rate.
< 0.0 for the use of i,j,k,iblock index packing to compute the mass flow rate. See example below.

areardef(11,*) Directions of computed mass flow rate allowed
< 0.0 allows only negative flows
= 0.0 allows both positive and negative flows.
> 0.0 allows only positive flows.

areardef(12,*) Flow loss coefficient, C_d, in Equ. 3-16 for a given flow area, but must be < 0 to activate this model; otherwise, all input using **areardef** is consistent with the original definitions in the 1998 GASFLOW Users Manual. For example, -1.0 is no flow loss, while -0.5 limits the flow by half.

3.3.6 Sub-grid Mass Flow Rate Model Examples

1. In the following example shown in Figure 3-33, we demonstrate a fractional cell face open for flow of 25%, only in the positive flow direction, with a 0.75 flow loss coefficient; using adjacent hole cells used for the flow computation as indicated by the large solid dots in the following figure.

   areardef(1:12,1) = 10, 10, 12, 13, 22, 23, 1, -0.25, 0.0, 0.0, +1.0, 0.75,

2. In the following example shown in Figure 3-34, we demonstrate a fractional cell face open for flow of 10%, only in the negative flow direction, with a 0.60 flow loss coefficient; using nearly static pressure cells for the flow computation as shown by the large solid dots in the following figure.

   areardef(1:12,1) = 10,10, 12,13, 22,23, 1,
   -0.10, -07132301, -14132301, 1.00, -0.60,

3. In the following example shown in Figure 3-35, we demonstrate a fractional cell face open for flow of 50%, with both positive and negative flow directions, with a 0.85 flow loss coefficient; using nearly static pressure cells for the flow computation as presented by the large solid dots in the following figure.

   areardef(1:12,1) = 10,10, 12,13, 22,23, 1,
   0.50, -10102301, -11102301, 0.00, -0.85,

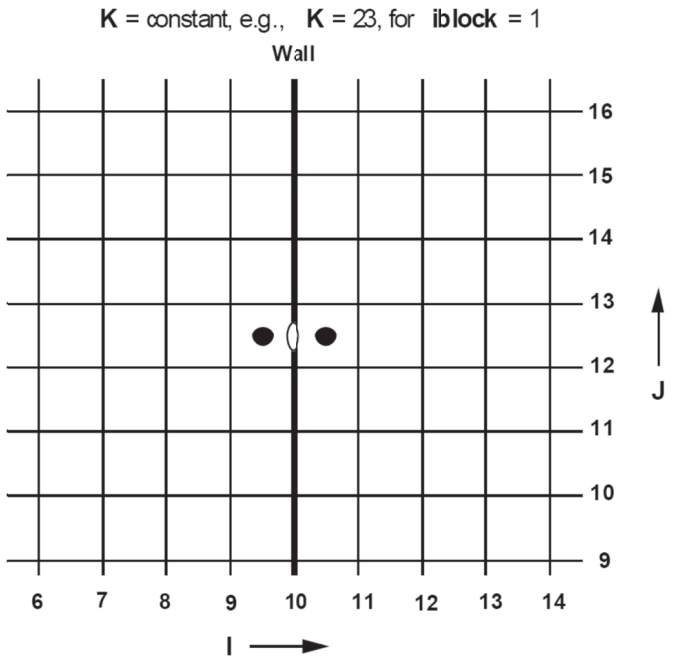


Figure 3-33 areardef example using adjacent hole computational cells to compute critical or sub-critical flows.

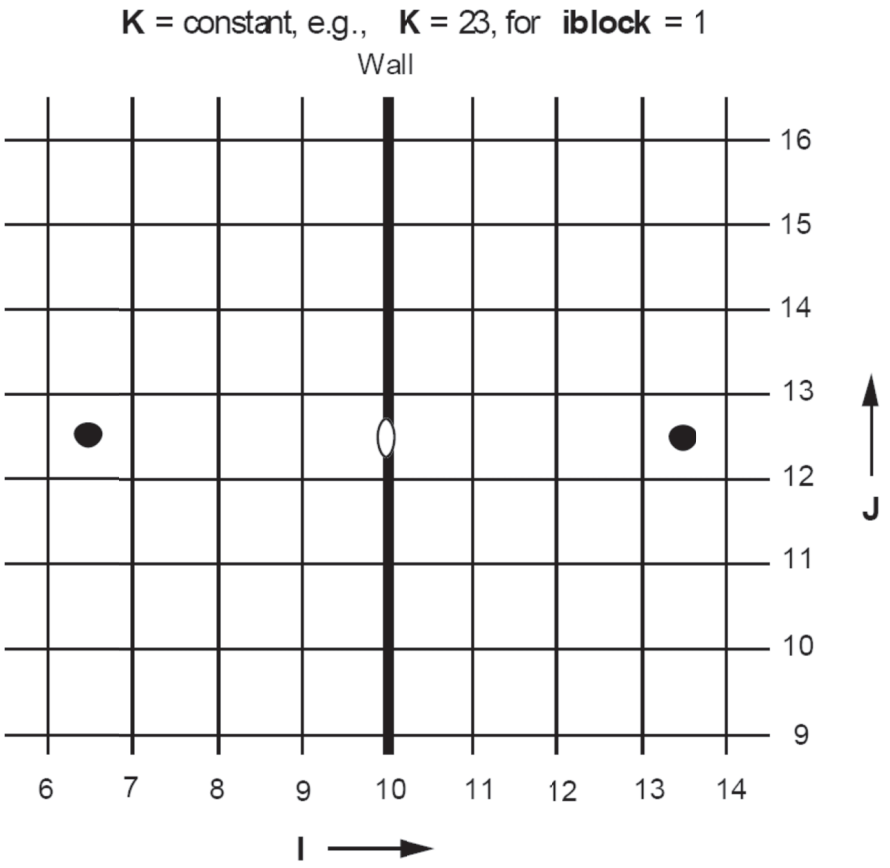


Figure 3-34 areardef example with nearly static pressure computational cells used to compute critical or sub-critical flows.

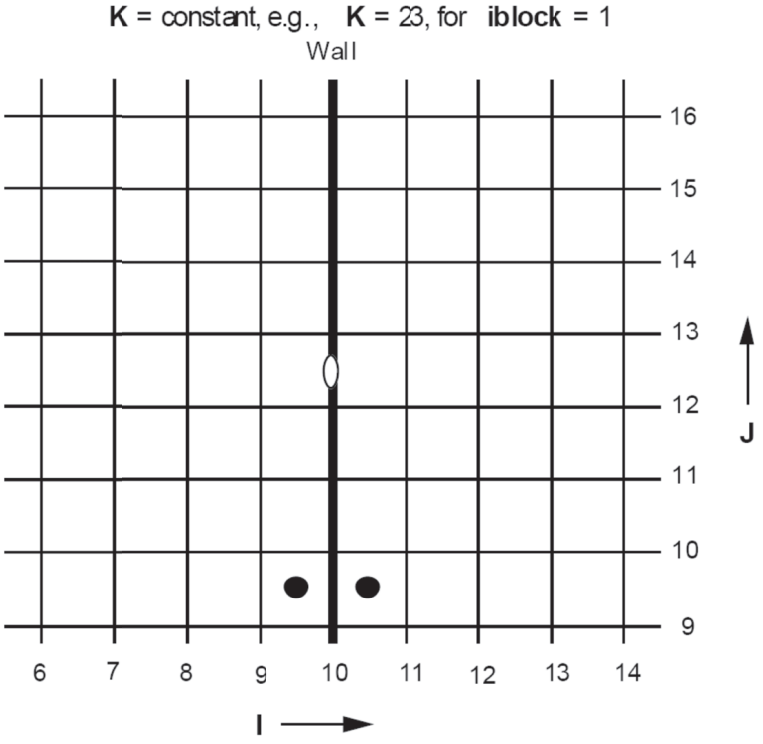


Figure 3-35 areardef example with largely removed static pressure computational cells used to compute critical or sub-critical flows.

3.3.7 Rupture Disks or Blowaway Panels

A rupture disk model to simulate any defined "wall" that can break or blowaway under a pressure load is available. It must be noted that the rupture disk model is always associated with a defined wall. When the heat transfer option is activated, i.e., **ihtflag** = 1 in NAMELIST RHEAT, the user does not have to exclude the wall section from the defining **walldef**. The code will automatically exclude any wall section that is also defined as a rupture disk from any heat transfer considerations.

The specifications for the rupture disks are required in input NAMELIST XPUT as follows:

rupdkdef(1,*) Beginning i mesh index (cell face number).
rupdkdef(2,*) Ending i mesh index (cell face number).
rupdkdef(3,*) Beginning j mesh index (cell face number).
rupdkdef(4,*) Ending j mesh index (cell face number).
rupdkdef(5,*) Beginning k mesh index (cell face number).
rupdkdef(6,*) Ending k mesh index (cell face number).
rupdkdef(7,*) Block number.
rupdkdef(8,*) Fraction of geometric flow area available for fluid flow after the rupture disk has failed.
rupdkdef(9,*) If rupdkdef(9,*) < 1.e+10: failure criterion provided as a pressure difference in dynes/cm^2

If 1.e+10 <rupdkdef(9,*)< 1.e+20: failure criterion provided as 1.e+10 times absolute failure pressure in dynes/cm^2
If 1.e+20 <rupdkdef(9,*)< 1.e+30: failure criterion provided as 1.e+20 times gas temperature in K

Note that the first 8 entries are identical to the fractional flow area definition **areardef** in NAMELIST XPUT. An example specifying 3 rupture disks, with fractional area open to flow of 0.1 after failing with a pressure difference greater than 0.01 bar (10^4 dynes/cm^2) would be input as follows:

```
$xput
   ...
   walls (1:8,1)      = 6, 6,   1,11,   1,11,  1,  1,
   walls (1:8,2)      = 1,11,   6, 6,   1,11,  1,  1,
   walls (1:8,3)      = 1,11,   1,11,   6, 6,  1,  1,
   rupdkdef(1:9,1)  = 6, 6,   3, 4,   3, 4,  1,  0.1, 1.0e+4,
   rupdkdef(1:9,2)  = 3, 4,   6, 6,   3, 4,  1,  0.1, 1.0e+4,
   rupdkdef(1:9,3)  = 3, 4,   3, 4,   6, 6,  1,  0.1, 1.0e+4,   .
   ...
$end
```

GASFLOW supports 300 definitions for **rupdkdef**.

3.4 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, GASFLOW-MPI 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	Neighbors -J	Neighbors +J	Neighbors -K	Neighbors +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 an 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 GASFLOW , 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. Pyscan can plot the mesh together with obstacles and walls.

4 Specification of Gas Species and Properties

4.1 Definition of Gas Species

In GASFLOW , 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):

$$pV = nRT \qquad \text{Equ. 4-1}$$

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:

$$pM = \rho RT. \qquad \text{Equ. 4-2}$$

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

$$I = \int_{T_{ref}}^{T} C_v dT \qquad \text{Equ. 4-3}$$

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. GASFLOW-MPI gives the user the following options for the calculation of the internal energies:

ieopt = 1 1st order polynomial
ieopt = 2 2nd order polynomial

These fits are provided over two ranges of temperatures, and the user can select which range is more appropriate for the application.

trange = 'low' *T* up to 3000 K
trange = 'high' *T* up to 5000 K

In addition, the specific heat capacity is required in the evaluation correlations for heat transfer and fluid flow. The user is given the following options for evaluation of the specific heat capacity:

icopt = 0 Derivative of specific internal energy.
icopt = 1 Constant value.

icopt = 2 2nd order polynomial (T < 750 K).

icopt = 3 Gordon & McBride approximation.

Note that the conservation equations for mass, energy, and momentum are solved consistently with the user-selected values for **ieopt** and **trange**. The recommended selection for **icopt** is **icopt** = 0, which ensures that correlations for heat transfer and fluid flow transport properties are evaluated with a consistent specific heat capacity.

The built-in gas component library in GASFLOW-MPI has 25 species with properties given in Table 4-1. In current GASFLOW-MPI, user-defined species are not supported. The user must choose the species to be calculated from this library.

Table 4-1 Properties of gas species available in GASFLOW-MPI

Gas Species – Common Name	Species – Symbol Used
Carbon atoms (soot)	c
Carbon monoxide	co
Carbon dioxide	co2
Hydrogen	h2
Water vapor	h2o
Nitrogen	n2
Nitrous oxide	n2o
Oxygen	o2
Air	air
Argon	ar
Helium	he
Ammonia	nh3
Methane	ch3
Hydroxyl radicals	ho
Hydrogen atoms	h
Hydrogen dioxide	ho2
Nitric oxide	no
Oxygen atoms	o
N-H radicals	nh
H-N-O radicals	hno
Hydrogen Peroxide	h2o2
Amidogen	nh2
Light gas	lg
Xenon	xenon
Liquid water	h2ol

The input array variable **mat** in NAMELIST group xput is used to define the species in a calculation. Any one or all of the species listed in Table 4-1 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 4-1. 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, as well as 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% N_2, 20.9% O_2, 0.9% Ar, and 0.03% CO_2. Here, we discuss identifying only 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 (Section 0).

Some of the species listed in Table 4-1 are not stable molecules. They are included in the gas library because the code 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" H_2-O_2-H_2O 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, GASFLOW-MPI uses one global reaction to model the entire hydrogen combustion kinetics.

4.2 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 GASFLOW-MPI, the diffusion process is modeled by 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. If the user selects no temperature dependence for the transport properties, then input specification of v, D, and α will require three numbers. For the kinematic viscosity v, 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:

$$Sc = v/D = \rho\mu/D \qquad \text{Equ. 4-4}$$

$$Pr = v/\alpha = \rho\mu/\alpha \qquad \text{Equ. 4-5}$$

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 which reads

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

specifies constant values of the kinematic viscosity (v, **nu**) as 0.2 cm^2/s, the thermal diffusivity (α) is 0.286 cm^2/s, and the mass diffusivity (D) is 0.5 cm^2/s. The default value for **nu** is 0.15 cm^2/s, while those for **prandtl** and **schmidt** are both 1 (i. e., $\alpha = D = v = 0.15$ cm^2/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 Section 7.2. However, if the user requires that the transport properties be functions of the temperature, then the following input options are available:

itopt = 0	Nonmechanistic calculation of the transport properties using input data.
muoption = 0	Properties are computed from the local density, the input kinematic viscosity *nu*, and the Prandtl and Schmidt numbers.
muoption = 1	Properties are computed from the local density, the input kinematic viscosity *nu*, and constant values for the thermal conductivity and diffusion coefficient.
muoption = 2	Properties are given by constant values for dynamic viscosity, thermal conductivity, and diffusion coefficient (see Section 7.2).
itopt ≠ 0	Mechanistic calculation of the transport properties, depending on temperature, pressure, and mixture composition.
itopt = 1	Constant value*, $T < 500$ K
itopt = 2	Linear approximation*, $T < 1000$ K
itopt = 3	Quadratic approximation*, $T < 3000$ K
itopt = 4	Cubic approximation*, $T < 3000$ K
itopt = 5	Quadratic approximation*, $T < 5000$ K
itopt = 6	Using approximation formula from CHEMKIN*, $T < 5000$ K

* regarding temperature dependence for the species' properties

If **itopt** ≠ 0, no specification of **muoption** is required; it is set to the default value 0. The property fits available in the GASFLOW-MPI were obtained from the CHEMKIN computer code.

5 Initial and Boundary Conditions

GASFLOW-MPI 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.

5.1 Specification of Initial Conditions

5.1.1 Fluid Composition and State

Except for a restart run (discussed in Section 9.2), 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 two-dimensional array. The second index identifies the particular "gas definition." For each **gasdef** specification, there are a minimum of 14 numbers required, which are input through the elements of the first array dimension with the following meaning:

gasdef(1,*) Beginning i mesh index (cell face number).

gasdef(2,*) Ending i mesh index (cell face number).

gasdef(3,*) Beginning j mesh index (cell face number).

gasdef(4,*) Ending j mesh index (cell face number).

gasdef(5,*) Beginning k mesh index (cell face number).

gasdef(6,*) Ending k mesh index (cell face number).

gasdef(7,*) Block number (must be 1 for GASFLOW-MPI).

gasdef(8,*) Pressure (dynes/cm^2) in defined volume. If **gasdef(8,*)** is less than zero, then the |**gasdef(8,*)**|points to the column number in the SORTAM file and the pressure will be obtained from this column in the SORTAM file. If **gasdef(8,*)** is less than zero and |**gasdef(8,*)**| is larger than 1,000,000, then |**gasdef(8,*)**|is a packed **i,j,k,iblk** location and the pressure will be obtained from the cell at **i,j,k,iblk**.

gasdef(9,*) Temperature (K) in defined volume. If **gasdef(9,*)** is less than zero, then the INT(ABS(**gasdef(9,*)**) points to the column number in the SORTAM file and the temperature will be obtained from this column in the SORTAM file.

gasdef(10,*) Option flag for specification of gas composition and time-dependent BC **(Boundary Condition)**: 1 for mass fraction, 2 for volume fraction, or > 9 implies that a time-dependent function for the pressure and temperature will be specified.

gasdef(11,*) Time(s) at which “gas definition” begins.

gasdef(12,*) Time(s) at which “gas definition” ends.

gasdef(13,*) 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 4-1, e. g., 'h2'.

gasdef(14,*) Mass or volume fraction of above gas species in defined volume. If **gasdef(14,*)** is less than zero, then the INT(ABS(**gasdef(14,*)**) points to the column number in the SORTAM file and the mass/volume fraction will be obtained from this column in the SORTAM file.

gasdef(15,*) Second gas species component number, if needed.

gasdef(16,*) Mass or volume fraction of second gas species in defined volume, if needed. If **gasdef(16,*)** is less than zero, then the INT(ABS(**gasdef(16,*)**) points to the column number in the SORTAM file and the mass/volume fraction will be obtained from this column in the SORTAM file.

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 23 gas species, 1 liquid component (water liquid), and 1 solid component (carbon soot) 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. Currently, pressure and temperature boundary conditions are only allowed at computational mesh boundaries. Therefore, the specification of boundary condition pressures with **gasdef** must currently be only at computational mesh boundaries.

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

mat = 'h2o', 'air',
gasdef(1:16,1) = 1, 6, 1, 6, 1, 6, 1, 1.013e6, 298.0,2, 0., 0., 1, 0.1, 2, 0.9,

which can also be input in the form:

mat = 'h2o', 'air',
gasdef(1:16,1) = 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 Figure 3–1 (i. e., imax = jmax = kmax = 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.

gasdef's time-dependent functions can be defined for the pressure and temperature. When **gasdef(10,*)** is greater than 9, GASFLOW-MPI will use time-dependent functions for the pressure and temperature for the cells specified in the **gasdef** definition. The function type is determined by the ten's digit in the **gasdef(10,*)** input. The constants to be used in the function are determined from the hundreds digit in the **gasdef(10,*)** input. The one's digit for **gasdef(10,*)** is still used as the option flag to indicate whether the input for species is volume fraction or mass fraction. For example, if **gasdef(10,*)** is 321, then the one's digit is 1, the ten's digit is 2, and the hundred's digit is 3. The 1 in the one's place indicates species concentrations will be input as a mass fraction. The 2 in the ten's place indicates a function type of 2 (see list below), and the 3 in the hundred's place indicates that the constants in function type 2 will be obtained from **pfunc(*,3)** and **tfunc(*,3)**.

If **gasdef(10,*)** is greater than 9, let the ten's digit in **gasdef(10,*)** be **ifunc** and the hundred's digit in **gasdef(10,*)** be **icons**. Then the following function types are available for the pressure and temperature, and the constants in these functions are determined from **gasdef(8,*), gasdef(9,*)**, **pfunc(*,icons)**, and **tfunc(*,icons)**. If the function type selected is table look-up, then the user input time versus pressure table is **ptab,** and the user input time versus temperature table is **ttab**.

ifunc = 0 implies constant function.
ifunc = 1 implies linear function, f = a + b * ts.
ifunc = 2 implies quadratic function, f = a + b * ts**2.
ifunc = 3 implies cubic function, f = a + b * ts**2 + c * ts**3.
ifunc = 4 implies 4th order function, f = a + b * ts**2 + c * ts**3 +d*ts**4.
ifunc = 5 implies table look-up function, f = table(tr, f).
ifunc = 6 implies exponential function, f = a + b * exp(tr/c).
ifunc = 7 implies sinusoidal function, f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e).

where,

a = **gasdef(8,*)** or **gasdef(9,*)**
b = **pfunc(1,icons)** or **tfunc(1,icons)**
c = **pfunc(2,icons)** or **tfunc(2,icons)**
d = **pfunc(3,icons)** or **tfunc(3,icons)**
e = **pfunc(4,icons)** or **tfunc(4,icons)**
tr = time – tstart
ts = (time – tstart) /(tend – tstart)
tstart = **gasdef(11,n)**
tend = **gasdef(12,n)**
table(tr,f) = **ptab(1,maxp,icons)** or **ttab(1,maxp,icons)**
maxp = 20, currently

The same function type is used for both pressure and temperature evaluation, but a different set of function constants are available. The user has available the following sets of constants for the functions defined above:

pfunc(4,*) Sets of constants for pressure functions.

ptab(2,*,9) Set of *x-y* pair tables that define time-dependent pressure boundary condition functions. **ptab(1,m,n)** is the time (s) for m^{th} point of n^{th} pressure table, and **ptab(2,m,n)** is the pressure (dynes/cm^2) for m^{th} point of n^{th} pressure table. Currently, the maximum number of points is 20 and the maximum number of tables is 9.

tfunc(4,*) Sets of constants for temperature boundary condition functions.

ttab(2,*,9) Set of *x-y* pair tables that define time-dependent temperature boundary condition functions. **ttab(1,m,n)** is the time (s) for m^{th} point of n^{th} temperature table, and **ttab(2,m,n)** is the temperature (K) for m^{th} point of n^{th} temperature table. Currently, the maximum number of points is 20 and the maximum number of tables is 9.

Note that for time-dependent tables, linear interpolation is performed between points in the table. The following example illustrates a quadratic increase in pressure from 1 bar to 2 bars from time 100 seconds to 200 seconds. For the example given below, **gasdef(1,1)** will be using the third set of **pfunc** and **tfunc** constants for the quadratic equation. From 100 to 200 seconds, the function for pressure will be

P = 106 + 106((time – 100) / 100)2

pbc(1:9,1) = 1, 2, 1, 2, 3, 4, 4, 100.0, 200.0,
gasdef(1:14,1) = 1, 2, 1, 2, 3, 4, 4, 1.0e+06, 300.0, 321, 100.0, 200.0, 'air', 1.0,
pfunc(1,1) = 100.0,
pfunc(1,2) = 100.0,
pfunc(1:2,3) = 0.0, 1.0e+06,
tfunc(1,1) = 0.0
tfunc(1,2) = 0.0
tfunc(1:2,3) = 0.0, 0.0,

The temperature will not vary but will be held at a constant value of 300 K, since the constants for the temperature function are zero. Note that for the example given above, **gasdef(10,1)** = 321: **iopt** = 1, which implies mass fraction concentrations; **ifunc** = 2, which implies a quadratic function; and **icons** = 3, which implies the third set of constants in the **pfunc** and **tfunc** tables of constants.

The following example illustrates a table look-up for both pressure and temperature. Note that linear interpolation will be performed between points in the table. The second table for both temperature and pressure will be used, and this **gasdef** will be applied from 10.0 to 99999.0 seconds. Note that the **gasdef** comes on at 10.0 seconds. The times in the table are relative to when the **gasdef** turns on. So the first point in the pressure table is at 50.0 + 10.0 = 60 seconds into the transient. From 10.0 to 60.0 seconds, the first point in the tables will be used (i.e., pressure = 1.1e + 06). Also, note that the times in tables do not have to be same for both the pressure and temperature. The pressure table ends at 200 + 10 = 210 seconds. For time greater than 210 seconds, the last point in the table will be used.

For this example, the pressure in bars as a function of time (*t*) is given below:

P = 1.0	for 0 < t < 10
P = 1.1	for 10 < t < 60
P = 1.1 - 0.3*(t-60)/100	for 60 < t < 160
P = 0.8 + 1.2*(t-160)/50	for 160 < t < 210
P = 2.0	for 210 < t < 99999

The temperature as a function of time (t) for this example is

T = 300	for 0 < t < 60
T = 300 - 5*(t-60)/200	for 60 < t < 260
T = 295 + 15*(t-260)/50	for 260 < t < 310
T = 310	for 310 < t < 99999

pbc(1:9,1) = 1, 2, 1, 2, 3, 4, 4, 0.0, 99999.0,
gasdef(1:14,1) = 1, 2, 1, 2, 3, 4, 4, 1.0e+06, 300.0, 1, 0.0, 10.0, 'air', 1.0,
gasdef(1:14,2) = 1, 2, 1, 2, 3, 4, 4, 1.0e+06, 300.0, 251, 10.0, 99999.0, 'air', 1.0,
ptab(1,1,1) = 0.0, 1.0e+06,
100.0, 2.0e+06,
ptab(1,1,2) = 0.0, 1.1e+06,
50.0, 1.1e+06,
150.0, 0.8e+06,
200.0, 2.0e+06,
ttab(1,1,1) = 0.0, 300.0,
100.0, 1000.0,
ttab(1,1,2) = 0.0, 300.0,
50.0, 300.0,
250.0, 295.0,
300.0, 310.0,

If negative values are input for pressure, temperature, and/or composition in the **gasdef** input, then time-dependent values for pressure, temperature, and/or composition can be obtained from the SORTAM file (see Section 5.2.4). When the pressure, temperature, or composition is a negative number in the **gasdef** input, the absolute value of the number points to a column in the SORTAM file. For the following example, the **gasdef** input is

gasdef(1:18,1) = 1 , 'im1', 1 , 'jm1' , 1 , 'km1', 1 , ; Initial condition
1.0e6, 300.0, 1, 0., 0., 'n2', 0.80, 'h2', 0.10, 'o2', 0.10,
gasdef(1:18,2) = 0, 1, 1, 2, 1, 2, 1 , ; Boundary condition
-2.0, -3.0, 2, 0., 9.e99, 'n2', -4.0, 'o2', -5.0, 'h2', -6.0,

and the SORTAM file is given below.

SORTAM file for ignitor test problem.
NCOLS
6
IVVALUES IVTYPES
1 1
0 1
0 1
0 1
0 1
0 1

time(s)	air(g/s)	p(dynes/cm^2)	t(K)	xf(n_2)	xf(o_2)	xf(h_2)
0.0000E+00	0.0000E+00	1.0000E+06	3.0000E+02	7.9000E-01	2.1000E-01	0.0000E+00
5.0000E+00	1.0000E+01	1.1000E+06	3.1000E+02	7.9000E-01	2.1000E-01	0.0000E+00
1.0000E+01	1.0000E+01	1.1000E+06	3.1000E+02	7.0000E-01	2.0000E-01	1.0000E-01
2.0000E+01	2.0000E+01	1.0000E+06	3.2000E+02	7.0000E-01	2.0000E-01	1.0000E-01
5.0000E+01	2.0000E+01	1.0000E+06	3.0000E+02	7.0000E-01	2.0000E-01	1.0000E-01
5.0000E+02	2.0000E+01	1.0000E+06	3.0000E+02	7.0000E-01	2.0000E-01	1.0000E-01

The pressure is given in column two of the SORTAM file, since **gasdef(8,2)** = -2; the temperature is column three in the SORTAM file, since **gasdef(9,2)** = -3; the nitrogen mass fraction is in column four of the SORTAM file, since **gasdef(14,2)** = -4; the oxygen mass fraction is in column five of the SORTAM file, since **gasdef(16,2)** = -5; and the hydrogen mass fraction is in column six of the SORTAM file, since **gasdef(18,2)** = -6.

If the user requires that the pressure in the **gasdef** be obtained from the pressure calculated for a cell inside the mesh, then **gasdef(8,*)** must be input as a negative number with the absolute value of the negative number equal to a packed i, j, k, iblk location. A packed i, j, k, iblk location is defined as:

n i,j,k,iblk packed location = IIJJKKBB.
i Number of millions in *n*.
j Number of ten thousands in *n* – i*1,000,000.
k Number of hundreds in *n* – i*1,000,000 – j*10,000.
iblk Number of ones in *n* – i*1,000,000 – j*10,000 – k*100. (must be 1 for GASFLOW-MPI)

For example,

gasdef(1:14,1) = 1,2, 3,4, 5,6, 1, –02030401, 300.0, 1 , 0. , 1.0e+99, ‘air’, 1.0,

The pressure used for this **gasdef** statement will be obtained from the pressure at cell i = 2, j = 3, k = 4, iblk = 1.

5.1.2 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

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

5.2 Specification of Boundary Conditions

GASFLOW-MPI offers two methods, one “global” and the other “local,” for specifying boundary conditions. These two methods will be described separately.

5.2.1 Global Definition

The first method applies boundary conditions on entire boundaries of the computational domain of all 3D blocks. For each 3D block, there are six surfaces that bound the three-dimensional 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 other input variables discussed subsequently are in the NAMELIST group xput.

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

These boundary condition types are currently applied uniformly to all 3D blocks in the input model. 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. 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.

Wall Functions. When rigid no-slip conditions are specified and turbulence is activated, the user may wish to use wall functions rather than resolving the boundary layers. The options are as follows:

iwallfunc = 0 ; default value, no wall functions are active.
iwallfunc = 1 ; no-slip conditions must to active, assumes smooth walls.
iwallfunc = 2 ; no-slip conditions must to active,
assumes rough walls and **krough** must be specified.

See Section 5.2.1 and Section 0 for further context concerning the wall functions.

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. Therefore, if cylindrical coordinates are chosen and the mesh is defined to extend from 0˚ to 360˚, then the following input should be used to specify the appropriate boundary condition at the -θ(-j) and +θ(+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/cm^2), the user would write the following input:

ibe = 5,
gasdef(1:14,1) = 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.

5.2.2 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.

Note that the **vbc**, **pbc**, **cbc**, and **mbc** boundary conditions can only currently be applied to the extreme surfaces of the mesh in each direction. 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. In addition, the mass flow rate can be specified as a boundary condition. With the mass flow rate specified as a boundary condition, the donor cell density is used to determine the actual velocity that will be specified as a boundary. 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:

nslipdef(1,*) Beginning i mesh index (cell face number).
nslipdef(2,*) Ending i mesh index (cell face number).
nslipdef(3,*) Beginning j mesh index (cell face number).
nslipdef(4,*) Ending j mesh index (cell face number).
nslipdef(5,*) Beginning k mesh index (cell face number).
nslipdef(6,*) Ending k mesh index (cell face number).
nslipdef(7,*) Block number (must be 1 for GASFLOW-MPI).
nslipdef(8,*) 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 two-dimensional 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(1:8,1) = 2, 2, 3, 5, 1, 9, 1, 'lower',
nslipdef(1:8,2) = 3, 3, 3, 5, 1, 9, 1, 'both',
nslipdef(1:8,3) = 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 1 for GASFLOW-MPI).
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).
pbc(4,*) Ending j mesh index (cell face number).
pbc(5,*) Beginning k mesh index (cell face number).
pbc(6,*) Ending k mesh index (cell face number).
pbc(7,*) Block number .
pbc(8,*) Start time (s).
pbc(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 imax = 11, jmax = kmax = 7. Now the +i boundary (i. e., the surface where i = imax –1 = 10, refer to Figure 3–1 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

pbc(1,1) = 10, 10, 3, 4, 3, 4, 1, 0.0, 9.e99,
gasdef(1:14,1) = 10, 11, 3, 4, 3, 4, 1, 1.0132e6, 300., 2, 0., 100., 1, 1.,
gasdef(1:16,2) = 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**.

Since **gasdef** allows for time-dependent functions for pressure and temperature the example given above can be made to impose a time-dependent pressure that changes from 1 to 2 atms over a 100 second time interval.

pbc(1:9,1) = 10, 10, 3, 4, 3, 4, 1, 0.0, 9.e99,
gasdef(1:14,1) = 10, 11, 3, 4, 3, 4, 1, 1.0132e6, 300., 151, 0., 100., 1, 1.,
ptab(1,1,1) = 0.0, 1.0132e+06,
100.0, 2.0265e+06,

ttab(1,1,1) = 0.0, 300.0,
100.0, 300.0,

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,*) Beginning i mesh index (cell face number).
vbc(3,*) Beginning j mesh index (cell face number).
vbc(4,*) Beginning j mesh index (cell face number).
vbc(5,*) Beginning k mesh index (cell face number).
vbc(6,*) Beginning k mesh index (cell face number).
vbc(7,*) Block number (must be 1 for GASFLOW-MPI).
vbc(8,*) If **vbc(8,*)** < 100, then **vbc(8,*)** is the index in the **vvalue** array that will define a constant velocity from the start time to the end time for the **vbc** definition. If **vbc(8,*)** > 100, then **vbc(8,*)** points to a time-dependent function which will be used to determine the velocity as a function of time (see example below).
vbc(9,*) Start time(s).
vbc(10,*) End time(s).

Note that **vbc(8,*)** < 100 does not directly specify what the velocity value is; rather, it specifies an integer that points to the corresponding element in the user input 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. Note that for ducts, the direction of increasing i is from the west end to the east end of the duct. 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(1:10,1) = 1, 1, 1, 7, 1, 7, 1, 3, 1.0, 2.0,
vbc(1:10,2) = 1, 1, 1, 7, 1, 7, 1, 2, 2.0, 5.0,
vbc(1:10,3) = 10, 10, 1, 7, 1, 7, 1, 2, 2.0, 5.0,
gasdef(1:14,8) = 0, 1, 1, 7, 1, 7, 1, 1.0132e6, 298.0, 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.)

If **vbc(8,*)** is larger than 100, then a time-dependent function for the velocity boundary condition will be specified. When **vbc(8,*)** is larger than 100, then the one's digit (let the one's digit be **ivv**) points to the value in the **vvalue** array to be used for the time = 0.0 constant in the velocity time-dependent functions; the ten's digit (let the ten's digit be **ifunc**) is the function type; and the hundred's digit (let the hundred's digit be **icons**) points to the constants to be used in the **vfunc** constants table. The function types available for velocity boundary conditions are the same as for the pressure and temperature functions defined in a time-dependent **gasdef** definition.

ifunc	function
0	constant function.
1	linear function, f = a + b * ts.
2	quadratic function, f = a + b * ts**2.
3	cubic function, f = a + b * ts**2 + c * ts**3.
4	4th order function, f = a + b * ts**2 + c * ts**3 +d*t**4.
5	table look-up function, f = table(tr, f).
6	exponential function, f = a + b * exp(tr/c).
7	sinusoidal function, f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e).

where

a = **vvalue(ivv)**
b = **vfunc(1,icons)**
c = **vfunc(2,icons)**
d = **vfunc(3,icons)**
e = **vfunc(4,icons)**
tr = time - tstart
ts = (time - tstart)/(tend - tstart)
tstart = **vbc(9,n)**
tend = **vbc(10,n)**
table(tr,f) = **vtab(1,maxp,icons)**

For time-dependent velocity boundary conditions, the user must either supply a table of constants (i.e., **vfunc**) if **ifunc** is not equal to 5 or a table of *x*-*y* pairs (i.e., **vtab**) if **ifunc** is equal to 5.

vfunc(4,maxtb) Sets of constants for velocity boundary condition functions.
vtab(2,maxp,9) Set of *x*-*y* pair tables that define velocity boundary condition functions.

Note that with a parameter statement in GASFLOW-MPI, **maxp** (i.e., the maximum number of points in a table) is currently set to 20 and **maxtb** (i.e., the maximum number of tables) is set to 20. The following example illustrates a linear increase in velocity from 100 cm/s to 200 cm/s from time 50 seconds to 150 seconds.

vvalue = 100.0, 250.0, 0.0, -225.0,
vbc(1:10,1) = 1, 1, 1, 0, 0, 0, 0, 111, 50.0, 150.,
vfunc(1,1) = 100.0,

Mass Flow Rate. The input variable **mbc** can be used to specify mass flow rate boundary conditions. The actual velocity used as the boundary condition is determined from the user-supplied mass flow rate, the donor cell density, and the flow area of the cell edge where the boundary condition is to be supplied, calculated according to the following formula:

vbc = **mbc**/(ρA)

where

vbc	Velocity boundary condition.
mbc	Mass flow rate boundary condition.
ρ	Donor cell density.
A	Flow area.

Each **mbc** definition requires 10 numbers:

mbc(1,*)	Beginning i mesh index (cell face number).
mbc(2,*)	Beginning i mesh index (cell face number).
mbc(3,*)	Beginning j mesh index (cell face number).
mbc(4,*)	Beginning j mesh index (cell face number).
mbc(5,*)	Beginning k mesh index (cell face number).
mbc(6,*)	Beginning k mesh index (cell face number).
mbc(7,*)	Block number (must be 1 for GASFLOW-MPI).
mbc(8,*)	If **mbc(8,*)** < 100, then **mbc(8,*)** is the index in the **mvalue** array that will define a constant mass flow rate from the start time to the end time for the **mbc** definition. If **mbc(8,*)** > 100, then **mbc(8,*)** points to a time-dependent function which will be used to determine the mass flow rate as a function of time (see example given below).
mbc(9,*)	Start time(s).
mbc(10,*)	End time(s).

Note that **mbc(8,*)** < 100 does not directly specify what the mass flow rate value is; rather, it specifies an integer that points to the corresponding element in the array **mvalue** that stores mass flow rate values. The sign convention used for the mass flow rate is that positive mass flow rate indicates flow in the direction of increasing i, j, or k index. Note that for ducts, i is increasing from the west end to the east end of the duct. The examples for **vbc** and **vvalue** are equivalent to **mbc** and **mvalue** with **vbc** replaced by **mbc** and **vvalue** replaced by **mvalue**.

If **mbc(8,*)** is larger than 100, then a time-dependent function for the mass flow rate boundary condition will be specified. When **mbc(8,*)** is larger than 100, then the one's digit (let the one's digit be **imv**) points to the value in the **mvalue** array to be used for the time = 0.0 constant in the velocity time-dependent functions; the ten's digit (let the ten's digit be **ifunc**) is the function type; and the hundred's digit (let the hundred's digit be **icons**) points to the constants to be used in the **vfunc** constants table. The function types available for mass flow rate boundary conditions are the same as for the pressure and temperature functions defined in a time-dependent **gasdef** definition and for the time-dependent velocity defined in the **vbc** array.

ifunc	function
0	constant function.
1	linear function, f = a + b * ts.
2	quadratic function, f = a + b * ts**2.
3	cubic function, f = a + b * ts**2 + c * ts**3.
4	4th order function, f = a + b * ts**2 + c * ts**3 +d*t**4.
5	table look-up function, f = table(tr, f).
6	exponential function, f = a + b * exp(tr/c).
7	sinusoidal function, f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e).

where,

a = **mvalue(imv)**
b = **mfunc(1,icons)**
c = **mfunc(2,icons)**
d = **mfunc(3,icons)**
e = **mfunc(4,icons)**
tr = time - tstart
ts = (time - tstart)/(tend - tstart)
tstart = **mbc(9,n)**
tend = **mbc(10,n)**
table(tr,f) = **mtab(1,maxp,icons)**

For time-dependent mass flow rate boundary conditions, the user must either supply a table of constants (i.e., **mfunc**) if **ifunc** is not equal to 5, or a table of *x-y* pairs (i.e., **mtab**) if **ifunc** is equal to 5.

mfunc(4,maxtb) Sets of constants for mass flow rate boundary condition functions.
mtab(2,maxp,9) Set of *x-y* pair tables that define mass flow rate boundary condition functions.

Note that with a parameter statement in GASFLOW, **maxp** is currently set to 20 and **maxtb** is set to 20. The following example illustrates a linear increase in mass flow rate from 1 g/s to 2 g/s from time 50 seconds to 150 seconds.

mvalue = 1.0,
mbc(1:10,1) = 1, 1, 1, 0, 0, 0, 0, 111, 50.0, 150.,
vfunc(1,1) = 1.0,

5.2.3 Diffusion Cutoff and Mass Balance with Source Reservoirs

In the following, we shall describe a method of defining internal boundary conditions to represent the sources of mass, momentum, and energy within the boundaries of the computational mesh. The idea is to generate a reservoir where we can define the fluid condition and velocity so that the fluid from this reservoir can be convected into the mesh.

The procedure is a little complicated and therefore can be best served with an example. The geometry from Figure 3-29 will be used with the addition of the reservoir, conditions in the reservoir, flow conditions from the reservoir to the rest of the computational domain, a method to subtract the reservoir mass and energy out of the normal mass and energy balance, and a method to insure that mass and energy are not diffused from either the reservoir to the computing volume of interest or in the reverse direction.

First, we will define two additional input quantities. The first is used to subtract from the mass and energy balances the mass and energy from the reservoir volume. This can be accomplished via the input array variable **subsodef** in NAMELIST group xput. The variable is a two-dimensional array. The second index identifies the particular "reservoir volume." For each **subsodef** specification, there are seven numbers required, which are input through the elements of the first array dimension with the following meaning:

subsodef(1,*) Beginning i mesh index (cell face number).
subsodef(2,*) Beginning i mesh index (cell face number).
subsodef(3,*) Beginning j mesh index (cell face number).
subsodef(4,*) Beginning j mesh index (cell face number).
subsodef(5,*) Beginning k mesh index (cell face number).
subsodef(6,*) Beginning k mesh index (cell face number).
subsodef(7,*) Block number (must be 1 for GASFLOW-MPI).

To prevent nondesired diffusion of gases from the reservoir into the computing region of interest and the reverse process as well, we introduce a method of forcing all diffusion coefficients at the reservoir boundary to zero. We define 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 **zeroddef** in the NAMELIST group xput.

The array **zeroddef** is 2D with the second index identifying the particular surface definition and the first index specifying seven numbers that are required to define the **zeroddef** surface:

zeroddef(1,*) Beginning i mesh index (cell face number).
zeroddef(2,*) Beginning i mesh index (cell face number).
zeroddef(3,*) Beginning j mesh index (cell face number).
zeroddef(4,*) Beginning j mesh index (cell face number).
zeroddef(5,*) Beginning k mesh index (cell face number).
zeroddef(6,*) Beginning k mesh index (cell face number).
zeroddef(7,*) Block number (must be 1 for GASFLOW-MPI).

The asterisk (*) should be replaced by an integer that identifies the particular **subsodef** and **zeroddef** definitions. GASFLOW-MPI supports 300 definitions each for subsodef and zeroddef. The example is as follows: We assume the same geometry as the example from Figure 3-29. The reservoir is located at cell i=8, j=6, k=10, and walls are constructed around the reservoir such that it is open in the positive *x* direction. An air mixture of nitrogen and oxygen at 1 bar and 300 K is established throughout the entire computational volume as an initial condition. The reservoir is filled with hydrogen also at 1 bar and 300 K and will remain at that condition for 10 s. Hydrogen from the

reservoir is injected into the computational volume at 100 cm/s for a period of 10 s. The mass and energy of the reservoir is subtracted from the overall mass and energy balance using the **subsodef** input variable. Diffusion of hydrogen from the reservoir and nitrogen and oxygen into the reservoir is prohibited by the use of the **zeroddef** input variable. The input stream is shown here:

```
$xput
…
 mobs
 walls      = 3, 10, 1, 11, 5, 8, 1, 1,  ; solid obstacle
            = 2,10, 1, 11, 3, 3, 1, 2,  ; horizontal wall
              2, 2, 1, 11, 6, 10, 1, 2,  ; vertical wall
              7, 8, 5, 6, 10, 10, 1, 2,  ; reservoir top wall
              7, 8, 5, 6, 9, 9, 1, 2,    ; reservoir bottom wall
              7, 8, 5, 5, 9, 10, 1, 2,   ; reservoir south wall
              7, 8, 6, 6, 9, 10, 1, 2,   ; reservoir north wall
              7, 7, 5, 6, 9, 10, 1, 2,   ; reservoir north wall
 holes      = 5, 7, 4, 7, 6, 8, 1, 0, 0, 0, 0, 0, 1,  ; top hole
              8, 9, 5, 6, 5, 8, 1, 0, 0, 0, 0, 1, 1,  ; thru hole
              8, 9, 5, 6, 2, 4, 1, 1, 1, 1, 1, 1, 1,  ; wall hole
 mat        = 'h2', 'n2', 'o2',  ; problem components
 gasdef(1:16,1) = 1, 11, 1, 11, 1, 11, 1, 1.0e+06, 300.0, 2, 0.0, 0.0, 'n2', 0.79, 'o2', 0.21, ;
              initial conditions
 gasdef(1:14,2) = 7, 8, 5, 6, 9, 10, 1, 1.0e+06, 300.0, 2, 0.0, 10.0, 'h2', 1.00,  ; reservoir conditions
 vbc(1:10,1)    = 8, 8, 5, 6, 9, 10, 1, 1, 0.0,10.0,  ; reservoir U
 vvalue         = 100.0, ; reservoir inflow velocity value (cm/s)
 subsodef(1,1)  = 7, 8, 5, 6, 9,10, 1, ; subtract mass & energy
 zeroddef(1:7,1) = 8, 8, 5, 6, 9,10, 1, ; zero reservoir diffusion
...

$end
```

A display of the geometry including the reservoir is presented in Figure 5–1.

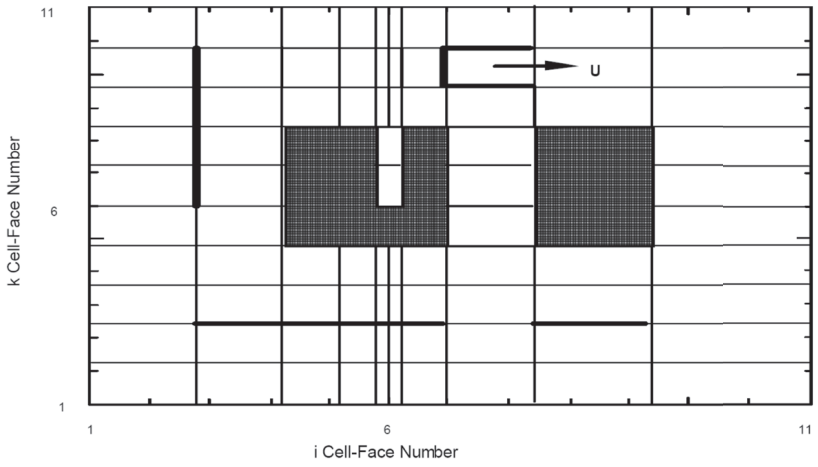


Figure 5-1 Example of applying an internal mass, momentum, and energy source

5.2.4 Boundary Conditions from SORTAM File

The **vvalue** used in a **vbc** definition can be specified as a time-dependent table if a SORTAM input file is provided by the user. The SORTAM file is a table of velocities, mass flow rates, and/or volumetric flow rates that can be applied at different locations in the GASFLOW-MPI mesh. The SORTAM velocities, mass flow rates, and/or volumetric flow rates can be applied at the boundaries of the mesh or at the boundary of an internal mass, momentum, and energy source described in Section 5.2.3. The locations where these time-dependent flow rate boundary conditions will be applied is identified through the **vbc** input as described in Section 5.2.2. The element in the **vvalue** array referenced by the **vbc** array will be changed with time according to the input in the SORTAM file. GASFLOW-MPI expects a SORTAM file if **sortami** is 1 in the xput NAMELIST group. The format of the SORTAM file is described below.

Record #	Description
1	Title, (a80) for the file.
2	Comment, (a80).
3	**ncols(*)** number of columns of data not counting the first column which is always time. The column number for the time column is zero.
4	Comment, (a80).
5	**ivvalues**(1), **ivtypes**(1), (*), where **ivvalues** (j) is the **ivvalues** for the j^{th} column in the sortam table. **ivvalues** is the index pointing to the array element in the **vvalue** array that will be time-dependent (i.e. **vvalue**(**ivvalues** (j)) will be defined by the j^{th} column of data in the sortam file). An **ivvalues** (j) = 0 implies that the j^{th} column of data in the sortam file will not be used as a flow rate boundary condition, but is retained for documentation or for other graphics programs. **ivtypes** (j) indicates the type of data in the j^{th} column of data in the SORTAM file. **IVTYPES (j)** = 0, the j^{th} column of data is a velocity, cm/s. **IVTYPES (j)** = 1, the j^{th} column of data is a mass flow rate, g/s. **IVTYPES (j)** = 2, the j^{th} column of data is a volumetric flow rate, cm^3/s.
6	**ivvalues**(2), **ivtypes(**2), (*).
4+**ncols**	ivvalues(ncols), ivtypes(ncols), (*).
5+**ncols**	Header for SORTAM table, (a80).
6+**ncols**	TIME, (SORTAB(*,j),j=1, NCOLS), (*), where time is the time in seconds for the flow rates that will be read into the **sortab** array in columns j = 1 through NCOLS. **sortab** is the container array that holds the columns of data read in from the SORTAM file. Note that the first data element read in after time is counted as column 1.

A partial listing of a SORTAM file is given below:

SORTAM file for GX6.
NCOLS
5
IVVALUES IVTYPES
1 1
2 1
3 1
5 1
0 1

time(s)	r_5 h_2o(g/s)	r_6 h_2o(g/s)	r_8 h_2o(g/s)	h_2 (g/s)	vreco (cm/s)
0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
3.0000E+01	1.8000E+01	9.0000E+00	9.0000E+00	0.0000E+00	0.0000E+00
2.2680E+03	1.8000E+01	9.0000E+00	9.0000E+00	0.0000E+00	0.0000E+00
2.2980E+03	2.8000E+01	1.4000E+01	1.4000E+01	0.0000E+00	0.0000E+00
3.2400E+03	2.8000E+01	1.4000E+01	1.4000E+01	0.0000E+00	0.0000E+00
3.2700E+03	5.0000E+01	2.1500E+01	2.1500E+01	0.0000E+00	0.0000E+00
5.2920E+03	5.0000E+01	2.1500E+01	2.1500E+01	0.0000E+00	0.0000E+00
5.3220E+03	5.0000E+01	2.9000E+01	2.9000E+01	0.0000E+00	0.0000E+00

In the partial listing of a SORTAM file given above, you can see that GASFLOW-MPI will expect 5 columns of data in addition to the time column. Also, the last or fifth column of flow rate data will be read, but not used. The first four columns of flow rate data are all mass flow rate data. The first three columns of data will be used to set **vvalue(1)**, **vvalue(2)**, and **vvalue(3)**, respectively as a function of time. The fourth column of data will be used to set **vvalue(5)** as a function of time. To find flow rates for time between time points given in the table, linear interpolation is used.

For mass flow rate and volumetric flow rate to be converted to velocity requires a flow area. The flow area is determined from the **vbc** input that refers to the identified for a given column of data in the SORTAM file. For example, the **vbc** input given below

vbc(1:10,1) = 08, 10, 08, 10, 09, 09, 1, 1, 0., 1.e+99,
vbc(1:10,2) = 08, 10, 08, 10, 04, 04, 1, 2, 0., 1.e+99,
vbc(1:10,3) = 08, 10, 16, 18, 04, 04, 1, 3, 0., 1.e+99,
vbc(1:10,4) = 04, 04, 15, 26, 3, 4, 1, 5, 0., 1.e+99,

and the SORTAM file given above would result in the flow area to be used for the conversion of the first column of mass flow rates to velocity, defined by the *z*-direction flow area on the top face of the following i, j, k cells:

i	j	k
9	9	9
10	9	9
9	10	9
10	10	9

The density required to convert the mass flow rate to a velocity will be obtained from the cell identified in the **vbc** input as the last i, j, k cell in the **vbc** definition. For the example given above, the last cell in the **vbc** definition is at i=10, j=10, k=9. The gas species convected into the GASFLOW mesh by the SORTAM flow rate boundary conditions will be of the normal donor cell densities and species. Therefore, the locations where the **vbc** velocities are convecting from, must be defined with the appropriate **gasdef**, **subsodef**, and **zeroddef** (See Section 5.2.3).

5.2.4.1 Time shift for the sortam file

It is convenient at times to have a means of executing GASFLOW-MPI using an applied time shifting algorithm to the **sortam** file. This has been implemented as

tshift_sortam = 0.0(default)
tshift_sortam > 0.0(any positive floating point value to shift the execution of the **sortam** file)

which is read by GASFLOW-MPI in the xput into stream.

Please be cautioned that this time shift only applies to the **sortam** file and does not apply to any other time dependent input variables, such as **gasdef**, **vbc**, **pbc,** or any other variable that appears in the GASFLOW-MPI input stream and has a tstart and tend associated with it.

5.2.4.2 Faster execution when using the sortam file

The execution time can slow significantly when using the SORTAM input file when this file has many piece wise continuous segments. For example in Figure 5–2, when the XPUT namelist input variable **sortami** = 1, the time step is adjusted to coincide exactly with the end points of the piece wise continuous intervals. This often leads to a severe reduction in the time-step for no other reason than to meet the intervals specified by the SORTAM file. A sawtooth behavior can result in the time step as the sudden reduction occurs, often several orders of magnitude less than necessary, and then the recovery until the end of the next interval is reached.

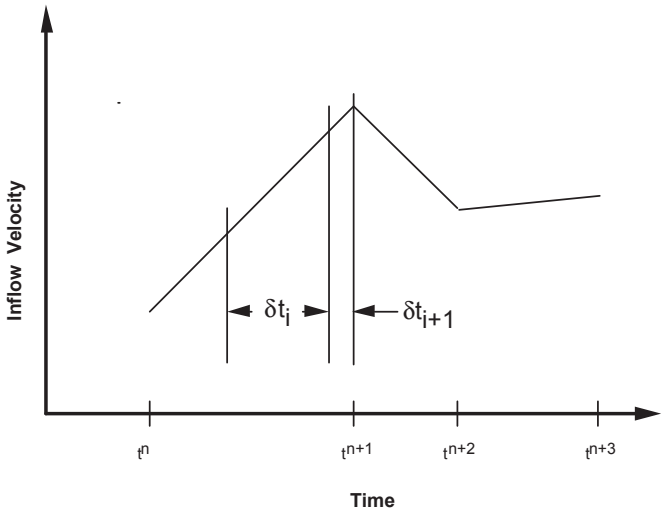


Figure 5-2 Example of potential time step reduction caused in the evaluation of the SORTAM file when sortami = 1.

By specifying **sortami** = 2 in the XPUT namelist input, GASFLOW-MPI will ignore the end of intervals as shown in Figure 5–2 by the dashed lines. There is some error induced by this option, but often it's self compensating by the nature of the integration as clearly seen by the area between the dashed and solid lines, i.e., a positive/negative effect. Note that in this case $\delta t_{i+2} > \delta t_{i+1} > \delta t_i$.

5.2.4.3 Normal expansions using the sortam file

The SORTAM File is often used by GASFLOW to incorporate complicated time dependent source term data. For example, a typical SORTAM file can appear as shown below.

GKN GRS Data unexpanded original GRS source with consistent water/steam sat. data, superh. melcor data
ncols
7
Ivvalues ivtypes
0 1
0 1
1 1
0 1
0 1
0 1
0 1

Time(s)	p (dynes/cm^2) #1	T (K) #2	mdot (g/s) #3	Xi (h2o) #4	Xi (h2ol) #5	Xi (h2) #6	Xi (xe) #7
0.0000E+00	1.5678E+08	6.0299E+02	3.9920E+06	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
1.0000E+00	1.3577E+08	5.9257E+02	5.7820E+06	6.485645E-02	9.351435E-01	0.000000E+00	0.000000E+00
4.0000E+00	1.2662E+08	5.8764E+02	5.3090E+06	6.404219E-02	9.359578E-01	0.000000E+00	0.000000E+00
8.0000E+00	1.0723E+08	5.7622E+02	4.9500E+06	6.363636E-02	9.363636E-01	0.000000E+00	0.000000E+00
1.2000E+01	1.0258E+08	5.7325E+02	4.8832E+06	5.938667E-02	9.406133E-01	0.000000E+00	0.000000E+00
1.4000E+01	9.7246E+07	5.6972E+02	4.8699E+06	5.544290E-02	9.445570E-01	0.000000E+00	0.000000E+00
1.8000E+01	1.0400E+08	5.7417E+02	4.1741E+06	7.211092E-02	9.278890E-01	0.000000E+00	0.000000E+00
2.2000E+01	1.0174E+08	5.7271E+02	3.9460E+06	6.073596E-02	9.392640E-01	0.000000E+00	0.000000E+00
2.4000E+01	9.8974E+07	5.7088E+02	3.8920E+06	5.369989E-02	9.463001E-01	0.000000E+00	0.000000E+00
3.2000E+01	9.3389E+07	5.6707E+02	3.8250E+06	3.267973E-02	9.673202E-01	0.000000E+00	0.000000E+00

The left most column is the time in seconds. Each of the numbered columns are:

#1 Pressure in dynes/cm2,
#2 Temperature in degrees Kelvin,
#3 Mass flow rate in g/s,
#4 Water vapor mass fraction of the mass flow rate,
#3*#5 Water liquid mass fraction of the mass flow rate,
#3*#6 Hydrogen mass fraction of the mass flow rate,
#3*#7 Xenon mass fraction of the mass flow rate.

The corresponding **gasdef** input statement can appear as:

gasdef(1:24,2) = 32, 35, 39, 42, 7, 8, 1, -1, -2, 1, 0., 1.e+99,
'n2', 0.0, 'o2', 0.0, 'h2', -6, 'h2o', -4, 'h2ol', -5, 'xenon', -7,

Note that this input through the **gasdef** statement allows the thermodynamic conditions of the sortam file as specified with the pressure (column #1) and temperature (column #2) to expand naturally to the conditions of the containment. In this case at time equal zero, we have liquid water at 156 bars and 603 K expanding to containment conditions through a "flashing" process. Below we will discuss other options available for this expansion process.

In fact, this SORTAM and **gasdef** statement are from the input for a reactor containment analysis. By using a linear piece wise continuous function in time, the mass flow rate (g/s), column 3 in the SORTAM File (note that the time column is not counted as a data column because GASFLOW-MPI always expects time to be located there), and the mass fractions, columns 4-7 in the SORTAM File, one can provide GASFLOW-MPI with time dependent source terms.

In order to generalize the use of the SORTAM file to multiple sources, we must relate which mass fraction columns correspond to what total mass flow rate column. Therefore, we have modified the **gasdef** input when a SORTAM File is to be read in the following way:

gasdef(1:24,2) = 32, 35, 39, 42, 7, 8, 1, -1, -2, 1, 0., 1.e+99,
'n2', 0.0, 'o2', 0.0, 'h2', -306, 'h2o', -304, 'h2ol', -305, 'xenon', -307,

Note that when either the species mass fraction or volume fraction is to be obtained from a specific column of the SORTAM file, the entry is always < -100. This allows, for example the hydrogen mass fraction in this example to be associated with the total mass fraction of column

INT(ABS(**gasdef**(18,2)/100)),

while the actual hydrogen mass fraction is

INT(ABS(**gasdef**(18,2)))-100*INT(ABS(**gasdef**(18,2)/100)).

When **gasdef**(18,2) = -306,

INT(ABS(**gasdef**(18,2)/100)) = 3, and

INT(ABS(**gasdef**(18,2)))-100*INT(ABS(**gasdef**(18,2)/100)) = 6.

Hence, the hydrogen mass fraction located in column 6 of the SORTAM File is associated with the total mass flow rate column 3. When volumetric flow rates and volume fractions are used, **gasdef**(10,*) = 2, then the methodology is equally exact. There is only one restriction, and it really isn't a restriction, and that is for any given **gasdef** statement referring to SORTAM File columns, they must be consistently mass related (mass flow rates and mass fractions) or volume related (volumetric flow rates and volume fractions).

5.2.4.4 Other expansion options using the sortam file

It is often desirable to, instead of allowing GASFLOW-MPI to provide the expansion from the conditions in a reservoir or source cell volume, pre-calculate the expansion process and then use this pre-expanded source as the source. In Figure 5–3, we show the available two-phase pre-expansions in the classical temperature-entropy diagram for the conditions at time equal zero in the sortam file expanded to 1 bar containment pressure:

A-B:Isentropic (constant entropy),
A-C:Isenthalpic (constant enthalpy), and
A-D:Isenergetic (constant internal energy).

In general, we're interested in the production of the vapor component during the expansion or flashing process. Displaying the vapor production is shown in Figure 5–4 where we plot vapor mass fraction as a function of Temperature. The least amount of vapor is produced by the isentropic expansion, while the maximum amount is produced by the isenergetic process.

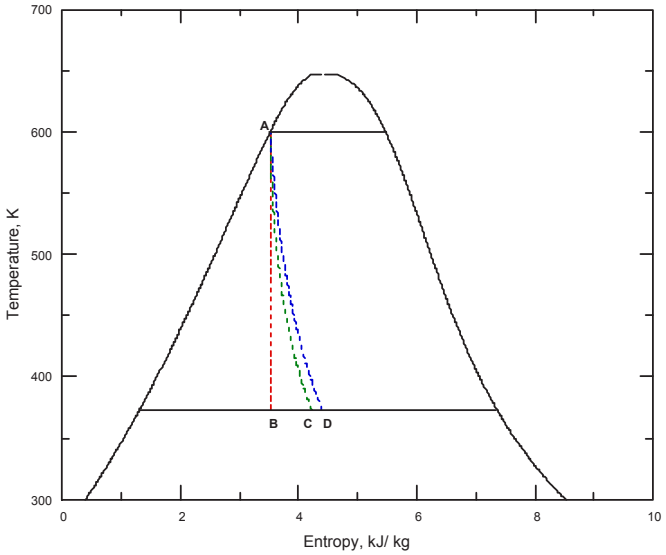


Figure 5-3: Path in a temperature-entropy diagram for a two-phase expansion from saturated liquid water at 157 bars to 1 bar for Isentropic (A-B), Isenthalpic (A-C), and Isenergetic (A-D) processes.

Steam table data has been introduced into the GASFLOW-MPI code to accurately predict the above mentioned expansions from saturation conditions of up to 200 bars (639 K) down to 0.01 bars (280 K). The equation governing the expansion is

$$\phi_A = \phi_{f,A} + x_A \cdot \phi_{fg,A} \qquad \text{Equ. 5-1}$$

where ϕ_A is the desired expansion property at condition **A**, $\phi_{f,A}$ is the saturated liquid value of the same property, $\phi_{fg,A}$ is the difference between the saturated vapor and liquid values for the same property, and x_A is quality or vapor mass fraction. Expanding now to the second state at the containment pressure, actually the saturation temperature for the containment pressure, gives

$$\phi_{f,A} + x_A \cdot \phi_{fg,A} = \phi_{f,i} + x_i \cdot \phi_{fg,i} \qquad \text{Equ. 5-2}$$

where the subscript i refers to the property associated with expansion **A-B**, **A-C**, or **A-D**. Solving for the quality yields

$$x_i = \frac{\phi_{f,A} + x_A \cdot \phi_{fg,A} - \phi_{f,i}}{\phi_{fg,i}}$$ Equ. 5-3

Equ. 5-3 is used to generate the results shown in Figure 5–4.

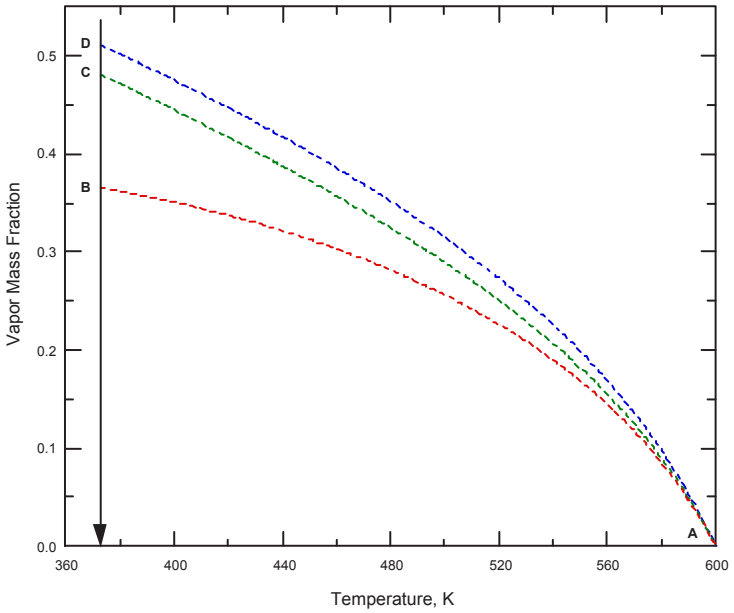


Figure 5-4 Path in a temperature-mass fraction diagram for a two-phase expansion from saturated liquid water at 157 bars to 1 bar for Isentropic (A-B), Isenthalpic (A-C), and Isenergetic (A-D) processes.

The user can specify which expansion they want by using the tenth (10^{th}) entry in the **gasdef** statement

gasdef(1:24,2) = 32, 35, 39, 42, 7, 8, 1, -1, -2, 1, 0., 1.e+99,
'n2', .0, 'o2', .0, 'h2', -306, 'h2o', -304, 'h2ol', -305, 'xenon', -307,

The meaning of 10th entry is defined by

gasdef(10,*) Option flag for specification of gas composition: 1 for mass fraction, 2 for volume fraction, > 9 implies a time dependent function for the pressure and temperature will be specified.
In addition, < 0 values imply pre-expansions:
-101 is an isentropic expansion with sortam data specified in terms of mass fractions;
-201 is an isenthalpic expansion with sortam data specified in terms of mass fractions;
-301 is an Isenergetic expansion with sortam data specified in terms of mass fractions;
-102 is an isentropic expansion with sortam data specified in terms of volume fractions;

-202 is an isenthalpic expansion with sortam data specified in terms of volume fractions;
-302 is an Isenergetic expansion with sortam data specified in terms of volume fractions;

When using the pre-expansion option, it is assumed that the expansion will occur from the pressure specified in the sortam file, column #2 in the above example, to a value characteristic of the containment pressure. As written in the new users manual, |**gasdef**(8,*)| > 1,000,000, then it is a packed i, j, k, iblk location for a reference pressure located in cell i, j, k, iblk.

In the following **gasdef** statement, we specify a isenergetic expansion from the data in the sortam file to the reference pressure located in cell i=36, j=43, k=08, and iblk=1.

gasdef(1:24,2) = 32, 35, 39, 42, 7, 8, 1, -36430801, -2, -301, 0., 1.e+99,
'n2', .0, 'o2', .0, 'h2', -306, 'h2o', -304, 'h2ol', -305, 'xenon', -307,

Since GASFLOW-MPI solves the internal energy equation as one of the primitive variable equations, it isn't too surprising to see that the results obtained with no pre-expansion (a natural expansion) and the isenergetic pre-expansion are nearly identical in the pressurization of the containment.

5.2.4.5 Rules and examples for using the sortam file.

The use of the SORTAM file for describing complicated input is a powerful technique from an internal source which allows fractional values from any defined source. The rules for using this capability are presented here, a review of the input variables involved, and several examples are presented below.

Rules for using SORTAM file for internal sources:

1. **sortami**: Flag of notification to signal GASFLOW to expect an input SORTAM file.

   0 (default): No SORTAM file expected.
   1 SORTAM file expected with integration coinciding exactly with piece wise continuous segments.
   2 SORTAM file expected with integration independent of piece wise continuous segments.

2. **walls**: Define the volume within the computational mesh where the internal source is located, specify the direction of the inflow, and isolate all source cells from each other with non heat exchange walls.

   walls(1,*) Beginning i mesh index (cell face number).
   walls(2,*) Ending i mesh index (cell face number).
   walls(3,*) Beginning j mesh index (cell face number).
   walls(4,*) Ending j mesh index (cell face number).
   walls(5,*) Beginning k mesh index (cell face number).
   walls(6,*) Ending k mesh index (cell face number).
   walls(7,*) Block number (must be 1 for GASFLOW-MPI).

walls(8,*) Integer to identify the type of wall (thickness and material). Used only for heat transfer; ignored if heat transfer is not invoked or value is set to zero. It is recommended that for internal source walls, this value is set to zero.

3. **subsodef** : Subtract mass and energy from the global mass and energy balances where the internal source is located.

subsodef(1,*) Beginning i mesh index (cell face number).
subsodef(2,*) Ending i mesh index (cell face number).
subsodef(3,*) Beginning j mesh index (cell face number).
subsodef(4,*) Ending j mesh index (cell face number).
subsodef(5,*) Beginning k mesh index (cell face number).
subsodef(6,*) Ending k mesh index (cell face number).
subsodef(7,*) Block number (must be 1 for GASFLOW-MPI).

4. **zeroddef** : Zero mass diffusion across all inflow surfaces.

zeroddef(1,*) Beginning i mesh index (cell face number).
zeroddef(2,*) Ending i mesh index (cell face number).
zeroddef(3,*) Beginning j mesh index (cell face number).
zeroddef(4,*) Ending j mesh index (cell face number).
zeroddef(5,*) Beginning k mesh index (cell face number).
zeroddef(6,*) Ending k mesh index (cell face number).
zeroddef(7,*) Block number (must be 1 for GASFLOW-MPI).

5. **gasdef** : Define source thermodynamic state and relationship to the SORTAM file.

gasdef (1,*) Beginning i mesh index (cell face number).
gasdef (2,*) Ending i mesh index (cell face number).
gasdef (3,*) Beginning j mesh index (cell face number).
gasdef (4,*) Ending j mesh index (cell face number).
gasdef (5,*) Beginning k mesh index (cell face number).
gasdef (6,*) Ending k mesh index (cell face number).
gasdef (7,*) Block number (must be 1 for GASFLOW-MPI).
gasdef (8,*) Pressure (dynes/cm^2) in defined volume. If **gasdef(8,*)** is less than zero, then the |**gasdef(8,*)**| points to the column number in the SORTAM file and the pressure will be obtained from this column in the SORTAM file. If **gasdef(8,*)** is less than zero and |**gasdef(8,*)**| is larger than 1,000,000, then |**gasdef(8,*)**| is a packed **i, j, k, iblk** location and the pressure will be obtained from the cell at **i, j, k, iblk**.
gasdef (9,*) Temperature (K) in defined volume. If **gasdef(9,*)** is less than zero, then the INT(ABS(**gasdef(9,*)**) points to the column number in the SORTAM file and the temperature will be obtained from this column in the SORTAM file.

gasdef (10,*) Option flag for specification of gas composition, thermodynamic pre-expansion option or time-dependent boundary condition:
1 for mass fraction,
2 for volume fraction, or
> 9 implies that a time-dependent function for the pressure and temperature will be specified. In addition,
< 0 values implies thermodynamic pre-expansions:
-101 is an *Isentropic* expansion with sortam data specified in terms of mass fractions.
-201 is an *Isenergetic* expansion with sortam data specified in terms of mass fractions.
-301 is an *Isenergetic* expansion with sortam data specified in terms of mass fractions.
-102 is an *Isentropic* expansion with sortam data specified in terms of volume fractions.
-202 is an *Isenergetic* expansion with sortam data specified in terms of volume fractions.
-302 is an *Isenthalpic* expansion with sortam data specified in terms of volume fractions.

gasdef (11,*) Time(s) at which “gas definition” begins.

gasdef (12,*) Time(s) at which “gas definition” ends.

gasdef (13,*) 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, e. g., 'h2', 'n2', 'h2o', etc.

gasdef (14,*) Mass or volume fraction of above gas species in defined volume. If **gasdef(14,*)** is less than zero, but > -100, then the INT(ABS(**gasdef(14,*)**) points to the column number in the SORTAM file and the mass/volume fraction will be obtained from this column in the SORTAM file. When **gasdef(14,*)** < -100, then the hundreds digits refer to the mass or volume fraction part (SORTAM column) in the SORTAM file while the ones digit refers to the component fraction in the SORTAM file.

gasdef (15,*) Second gas species component number, if needed.

gasdef (16,*) Mass or volume fraction of second gas species in defined volume, if needed.

6. **vbc**: Define the inflow surface for the internal source.

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 1 for GASFLOW-MPI).

vbc(8,*) If **vbc(8,*)** < 100, then **vbc(8,*)** is the index in the **vvalue** array that will define a constant velocity from the start time to the end time for the **vbc** definition. If **vbc(8,*)** > 100, then **vbc(8,*)** points to a time-dependent function which will be used to determine the velocity as a function of time. When **vbc(8,*)** < 0, then this flags the interaction with the SORTAM file and the vvalue location to provide a negative direction inflow for an internal source.

vbc(9,*) Start time(s).

vbc(10,*) End time(s).

7. **vvalue**: Define the velocity on the inflow surface for the internal source.

8. A third input column has been added to the sortam file with the ivvalues - ivtypes input. This column is called the source_fraction and it refers to the fraction of that particular column (in both a positive and negative sense) inflowing through that ivvalue surface. The input FORMAT is (2i9,f9.4) for the ivvalues - ivtypes - source_fraction input.

These rules are demonstrated in the following five examples.

EXAMPLE 1:

This is an example showing 100% positive direction internal source inflow through a single surface. Note the last entry in the ivvalues - ivtypes - source_fraction input columns; the 1.0 indicates that 100% of the specified mass flow rate will be used for the **vvalue(1)** entry associated with **vbc(1,1)**. The result of this input stream and the sortam file shown in **Table 5-1** is presented in Figure 5–5.

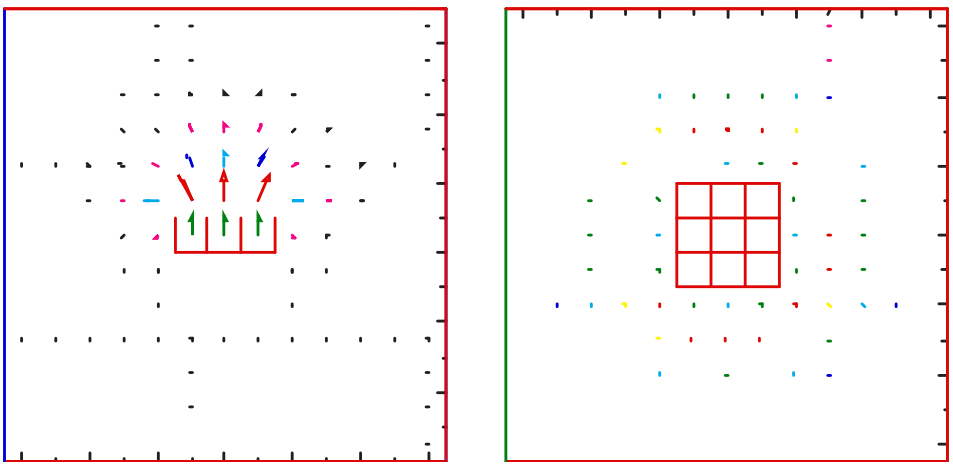


Figure 5-5: Example 1 sortam demonstration. Left side shows a vertical slice through the internal source, while the right side shows a horizontal slice through the internal source.

$xput

…

mat = 'n2', 'h2', 'h2o', 'h2ol',

gasdef(1:14,1) = 1, 'im1', 1, 'jm1', 1, 'km1', 1, 1.015e+06, 300.00, 2, 0., 0., 'n2', 1.0,

gasdef(1:14,2) = 6, 9, 6, 9, 7, 8, 1, -02020801, -2, 1, 0.0, 800.0,
'n2', 0.0, 'h2o', -603, 'h2ol', -604, 'h2', -605,
sortami = 2,
walls = 6, 9, 6, 6, 7, 8, 1, 0,
6, 9, 7, 7, 7, 8, 1, 0,
6, 9, 8, 8, 7, 8, 1, 0,
6, 9, 9, 9, 7, 8, 1, 0,
6, 6, 6, 9, 7, 8, 1, 0,
7, 7, 6, 9, 7, 8, 1, 0,
8, 8, 6, 9, 7, 8, 1, 0,
9, 9, 6, 9, 7, 8, 1, 0,
6, 9, 6, 9, 7, 7, 1, 0,
subsodef(1:7,1) = 6, 9, 6, 9, 7, 8, 1,
zeroddef(1:7,1) = 6, 9, 6, 9, 8, 8, 1,
vbc(1:10,1) = 6, 9, 6, 9, 8, 8, 1, 1, 0.0, 800.0,
vvalue = 0.0,
...

$end

Table 5-1 SORTAM file for Example 1

Example SORTAM file for 1 surface + outflow
ncols
6
ivvalues ivtypes source_fraction
0 0
0 0
0 0
0 0
0 0
1 1 1.0

time(s)	P (d/cm^2)	T (K)	xh_2o	xh_2ol	Xh_2	mdot (g/s)
0	5.6E6	4.04E2	4.87E-1	5.50E-2	4.58E-1	1.5E3
100	5.1E6	4.00E2	4.10E-2	3.32E-1	6.27E-1	1.2E3
200	5.3E6	3.98E2	6.00E-2	3.60E-1	5.80E-1	1.1E3
300	5.3E6	3.95E2	3.00E-3	4.54E-1	5.43E-1	9.9E2
400	5.5E6	3.92E2	5.63E-1	1.22E-1	3.15E-1	1.2E3
500	5.8E6	3.92E2	6.21E-1	1.48E-1	2.31E-1	1.1E3
600	5.9E6	3.92E2	5.31E-1	3.14E-1	1.55E-1	8.8E2
700	5.5E6	3.90E2	3.21E-1	6.75E-1	4.00E-3	5.6E2
800	5.5E6	3.90E2	3.19E-1	6.78E-1	3.00E-3	5.6E2

EXAMPLE 2:

This is an example showing 100% positive direction internal source inflow through a single surface but instead of using a solid bottom on the source volume, we use **zeroddef(1,2)**, **vbc(1,2)** and **vvalue(2)** to effectively provide a zero fluxing boundary condition. Note again that the last entry in the ivvalues - ivtypes - source_fraction input columns; the 1.0 indicates that 100% of the specified mass flow rate will be used for the **vvalue(1)** entry associated with **vbc(1,1)**. The result of this input stream and the sortam file shown in Table 5-2 is presented in Figure 5-6. The calculation is identical to Example 1.

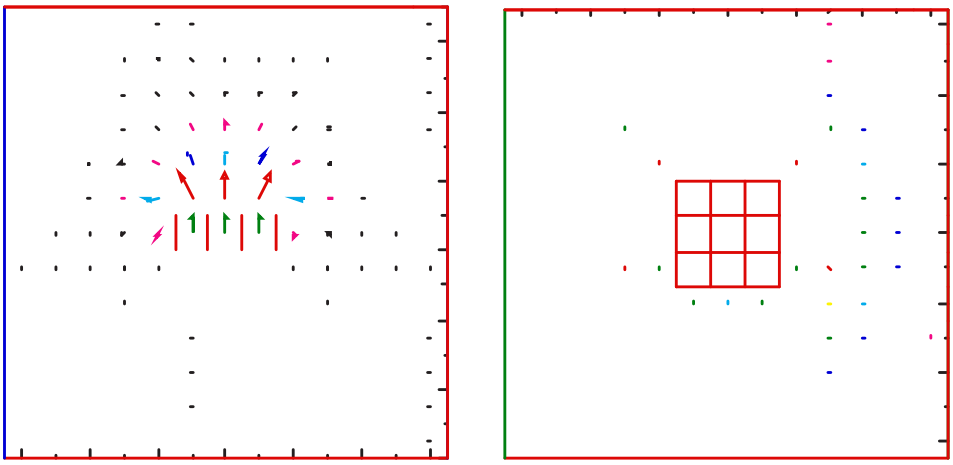


Figure 5-6 Example 2 sortam demonstration. Left side shows a vertical slice through the internal source, while the right side shows a horizontal slice through the internal source.

$xput
...
mat = 'n2', 'h2', 'h2o', 'h2ol',
gasdef(1:14,1) = 1, 'im1', 1, 'jm1', 1, 'km1', 1, 1.015e+06, 300.00, 2, 0., 0., 'n2', 1.0,
gasdef(1:14,2) = 6, 9, 6, 9, 7, 8, 1, -02020801, -2, 1, 0.0, 800.0,
'n2', 0.0, 'h2o', -603, 'h2ol', -604, 'h2', -605,
sortami = 2,
walls = 6, 9, 6, 6, 7, 8, 1, 0,
6, 9, 7, 7, 7, 8, 1, 0,
6, 9, 8, 8, 7, 8, 1, 0,
6, 9, 9, 9, 7, 8, 1, 0,
6, 6, 6, 9, 7, 8, 1, 0,
7, 7, 6, 9, 7, 8, 1, 0,
8, 8, 6, 9, 7, 8, 1, 0,
9, 9, 6, 9, 7, 8, 1, 0,
6, 9, 6, 9, 7, 7, 1, 0,
subsodef(1:7,1) = 6, 9, 6, 9, 7, 8, 1,

zeroddef(1:7,1) = 6, 9, 6, 9, 8, 8, 1,
zeroddef(1:7,2) = 6, 9, 6, 9, 7, 7, 1,
vbc(1:10,1) = 6, 9, 6, 9, 8, 8, 1, 1, 0.0, 800.0,
vbc(1:10,2) = 6, 9, 6, 9, 7, 7, 1, 2, 0.0, 800.0,
vvalue = 0.0, 0.0,
...
$end

Table 5-2 SORTAM file for Example 2

Example SORTAM file for 1 surface + outflow/0 bottom
ncols
6
ivvalues ivtypes source_fraction
0 0
0 0
0 0
0 0
0 0
1 1 1.0

time(s)	P (d/cm^2)	T (K)	xh_2o	xh_2ol	xh_2	mdot (g/s)
0	5.6E6	4.04E2	4.87E-1	5.50E-2	4.58E-1	1.5E3
100	5.1E6	4.00E2	4.10E-2	3.32E-1	6.27E-1	1.2E3
200	5.3E6	3.98E2	6.00E-2	3.60E-1	5.80E-1	1.1E3
300	5.3E6	3.95E2	3.00E-3	4.54E-1	5.43E-1	9.9E2
400	5.5E6	3.92E2	5.63E-1	1.22E-1	3.15E-1	1.2E3
500	5.8E6	3.92E2	6.21E-1	1.48E-1	2.31E-1	1.1E3
600	5.9E6	3.92E2	5.31E-1	3.14E-1	1.55E-1	8.8E2
700	5.5E6	3.90E2	3.21E-1	6.75E-1	4.00E-3	5.6E2
800	5.5E6	3.90E2	3.19E-1	6.78E-1	3.00E-3	5.6E2

EXAMPLE 3:

This is an example showing 100% negative direction internal source inflow through a single surface but instead of using a solid top surface on the source volume, we use **zeroddef(1,1)**, **vbc(1,1)** and **vvalue(1)** and the 6th column entry in the sortam file to effectively provide a top zero fluxing boundary condition. Note again that the last entry in the ivvalues - ivtypes - source_fraction input column 6 ; the +0.0 indicates that none of the specified mass flow rate will be used for the **vvalue(1)** entry associated with **vbc(1,1)** and the 7th column entry -1.0 indicates that 100% of the specified mass flow rate will be used for the **vvalue(2)** entry associated with **vbc(1,2)** where a negative value for **vbc(8,2)** is required to provide the inflow in the negative coordinate direction. The result of this

input stream and the sortam file shown in Table 5-3 is presented in Figure 5-7. This simulation is very similar to Examples 1 and 2 except the source inflow is in the negative coordinate direction.

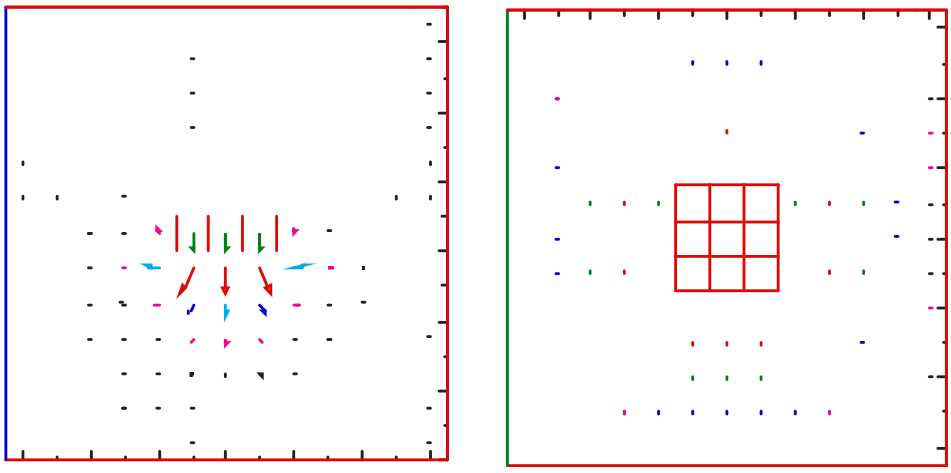


Figure 5-7 Example 3 SORTAM demonstration. Left side shows a vertical slice through the internal source, while the right side shows a horizontal slice through the internal source.

```
$xput
...
 mat              = 'n2', 'h2', 'h2o', 'h2ol',
 gasdef(1:14,1)   = 1, 'im1', 1, 'jm1', 1, 'km1', 1, 1.015e+06, 300.00, 2, 0., 0., 'n2', 1.0,
 gasdef(1:14,2)   = 6, 9, 6, 9, 7, 8, 1, -02020801, -2, 1, 0.0, 800.0,
                    'n2', 0.0, 'h2o', -603, 'h2ol', -604, 'h2', -605,
 sortami          = 2,
 walls            = 6, 9, 6, 6, 7, 8, 1, 0,
                    6, 9, 7, 7, 7, 8, 1, 0,
                    6, 9, 8, 8, 7, 8, 1, 0,
                    6, 9, 9, 9, 7, 8, 1, 0,
                    6, 6, 6, 9, 7, 8, 1, 0,
                    7, 7, 6, 9, 7, 8, 1, 0,
                    8, 8, 6, 9, 7, 8, 1, 0,
                    9, 9, 6, 9, 7, 8, 1, 0,
                    6, 9, 6, 9, 7, 7, 1, 0,
 subsodef(1:7,1)  = 6, 9, 6, 9, 7, 8, 1,
 zeroddef(1:7,1)  = 6, 9, 6, 9, 8, 8, 1,
 zeroddef(1:7,2)  = 6, 9, 6, 9, 7, 7, 1,
 vbc(1:10,1)      = 6, 9, 6, 9, 8, 8, 1, 1, 0.0, 800.0,

 vbc(1:10,2)      = 6, 9, 6, 9, 7, 7, 1, -2, 0.0, 800.0,
 vvalue           = 0.0, 0.0,
...
$end
```

Table 5-3 SORTAM file for Example 3

Example SORTAM file for 1 surface + outflow
ncols
7
Ivvalues ivtypes source_fraction
0 0
0 0
0 0
0 0
0 0
1 1 0.0
2 1 -1.0

time(s)	P (d/cm^2)	T (K)	xh_2o	xh_2ol	xh_2	mdot1 (g/s)	mdot2 (g/s)
0	5.6E6	4.04E2	4.87E-1	5.50E-2	4.58E-1	1.5E3	1.5E3
100	5.1E6	4.00E2	4.10E-2	3.32E-1	6.27E-1	1.2E3	1.2E3
200	5.3E6	3.98E2	6.00E-2	3.60E-1	5.80E-1	1.1E3	1.1E3
300	5.3E6	3.95E2	3.00E-3	4.54E-1	5.43E-1	9.9E2	9.9E2
400	5.5E6	3.92E2	5.63E-1	1.22E-1	3.15E-1	1.2E3	1.2E3
500	5.8E6	3.92E2	6.21E-1	1.48E-1	2.31E-1	1.1E3	1.1E3
600	5.9E6	3.92E2	5.31E-1	3.14E-1	1.55E-1	8.8E2	8.8E2
700	5.5E6	3.90E2	3.21E-1	6.75E-1	4.00E-3	5.6E2	5.6E2
800	5.5E6	3.90E2	3.19E-1	6.78E-1	3.00E-3	5.6E2	5.6E2

EXAMPLE 4:

This is an example showing 50% negative direction and 50% positive direction internal source inflow through two surfaces, we use **zeroddef(1,1)**, **vbc(1,1)** and **vvalue(1)** and the 6th column entry in the SORTAM file to provide the positive top inflow condition and **zeroddef(1,2)**, **vbc(1,2)** and **vvalue(2)** and the 7th column entry in the SORTAM file to provide the negative bottom inflow condition. Note again that the last entry in the IVVALUES - IVTYPES - SOURCE_FRACTION input column 6 ; the +0.5 indicates that 50% of the specified mass flow rate will be used for the **vvalue(1)** entry associated with **vbc(1,1)** and the 7th column entry -0.5 indicates that 50% of the specified mass flow rate will be used for the **vvalue(2)** entry associated with **vbc(1,2)** where a negative value for **vbc(8,2)** is required to provide the inflow in the negative coordinate direction. The result of this input stream and the SORTAM file shown in Table 5–4 is presented in Figure 5-8.

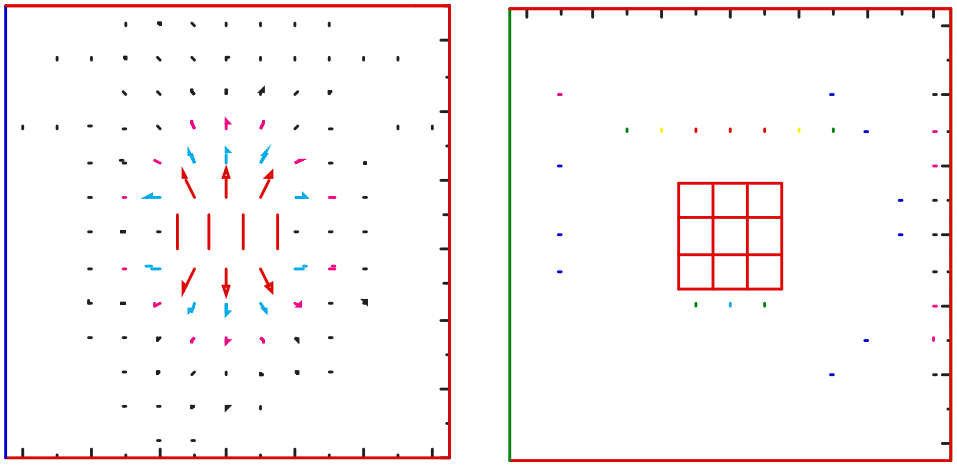


Figure 5-8 Example 4 SORTAM demonstration. Left side shows a vertical slice through the internal source, while the right side shows a horizontal slice through the internal source.

```
$xput
...
 mat              = 'n2', 'h2', 'h2o', 'h2ol',
 gasdef(1:14,1)   = 1, 'im1', 1, 'jm1', 1, 'km1', 1, 1.015e+06, 300.00, 2, 0., 0., 'n2', 1.0,
 gasdef(1:14,2)   = 6, 9, 6, 9, 7, 8, 1, -02020801, -2, 1, 0.0, 800.0,
                    'n2', 0.0, 'h2o', -603, 'h2ol', -604, 'h2', -605,
 sortami          = 2,
 walls            = 6, 9, 6, 6, 7, 8, 1, 0,
                    6, 9, 7, 7, 7, 8, 1, 0,
                    6, 9, 8, 8, 7, 8, 1, 0,
                    6, 9, 9, 9, 7, 8, 1, 0,
                    6, 6, 6, 9, 7, 8, 1, 0,
                    7, 7, 6, 9, 7, 8, 1, 0,
                    8, 8, 6, 9, 7, 8, 1, 0,
                    9, 9, 6, 9, 7, 8, 1, 0,
                    6, 9, 6, 9, 7, 7, 1, 0,
 subsodef(1:7,1)  = 6, 9, 6, 9, 7, 8, 1,
 zeroddef(1:7,1)  = 6, 9, 6, 9, 8, 8, 1,
 zeroddef(1:7,2)  = 6, 9, 6, 9, 7, 7, 1,
 vbc(1:10,1)      = 6, 9, 6, 9, 8, 8, 1, 1, 0.0, 800.0,
 vbc(1:10,2)      = 6, 9, 6, 9, 7, 7, 1, -2, 0.0, 800.0,
 vvalue           = 0.0, 0.0,
...
$end
```

Table 5-4 SORTAM file for Example 4

Example SORTAM file for 1 surface +/- outflow
ncols
7
Ivvalues ivtypes source_fraction
0 0
0 0
0 0
0 0
0 0
1 1 +0.5
2 1 -0.5

time(s)	P (d/cm^2)	T (K)	xh_2o	xh_2ol	xh_2	mdot1 (g/s)	mdot2 (g/s)
0	5.6E6	4.04E2	4.87E-1	5.50E-2	4.58E-1	1.5E3	1.5E3
100	5.1E6	4.00E2	4.10E-2	3.32E-1	6.27E-1	1.2E3	1.2E3
200	5.3E6	3.98E2	6.00E-2	3.60E-1	5.80E-1	1.1E3	1.1E3
300	5.3E6	3.95E2	3.00E-3	4.54E-1	5.43E-1	9.9E2	9.9E2
400	5.5E6	3.92E2	5.63E-1	1.22E-1	3.15E-1	1.2E3	1.2E3
500	5.8E6	3.92E2	6.21E-1	1.48E-1	2.31E-1	1.1E3	1.1E3
600	5.9E6	3.92E2	5.31E-1	3.14E-1	1.55E-1	8.8E2	8.8E2
700	5.5E6	3.90E2	3.21E-1	6.75E-1	4.00E-3	5.6E2	5.6E2
800	5.5E6	3.90E2	3.19E-1	6.78E-1	3.00E-3	5.6E2	5.6E2

EXAMPLE 5:

This is an example showing inflow on all 6 faces of the defined internal source. Note that for the current model, we need to specify 6 mass flow rate columns - one for each of the source volume surfaces. This example is a direct extension of the previous examples. The result of this input stream and the SORTAM file shown in Table 5–5 is presented in Figure 5-9.

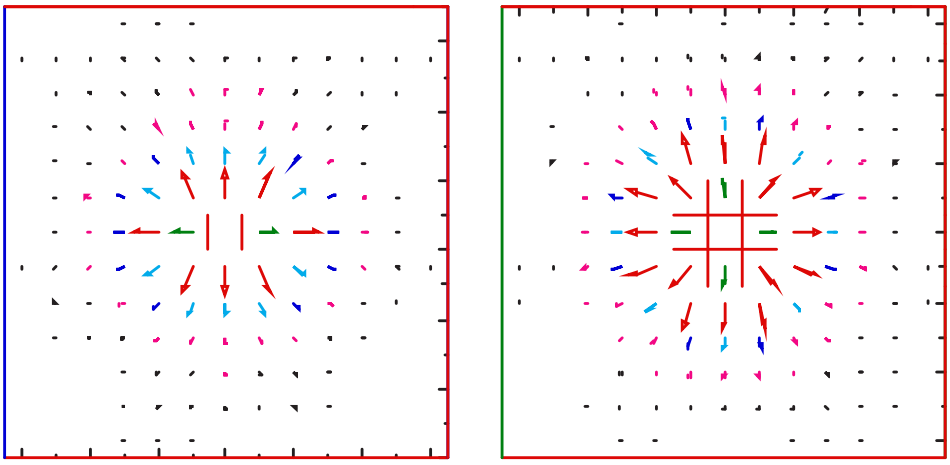


Figure 5-9. Example 5 SORTAM demonstration. Left side shows a vertical slice through the internal source, while the right side shows a horizontal slice through the internal source.

```
$xput
...
 mat                = 'n2', 'h2', 'h2o', 'h2ol',
 gasdef(1:14,1)     = 1, 'im1', 1, 'jm1', 1, 'km1', 1, 1.015e+06, 300.00, 2, 0., 0., 'n2', 1.0,
 gasdef(1:14,2)     = 6, 9, 6, 9, 7, 8, 1, -02020801, -2, 1, 0.0, 800.0,
                      'n2', 0.0, 'h2o', -603, 'h2ol', -604, 'h2', -605,
 sortami            = 2,
 walls              = 6, 9, 6, 6, 7, 8, 1, 0,
                      6, 9, 7, 7, 7, 8, 1, 0,
                      6, 9, 8, 8, 7, 8, 1, 0,
                      6, 9, 9, 9, 7, 8, 1, 0,
                      6, 6, 6, 9, 7, 8, 1, 0,
                      7, 7, 6, 9, 7, 8, 1, 0,
                      8, 8, 6, 9, 7, 8, 1, 0,
                      9, 9, 6, 9, 7, 8, 1, 0,
                      6, 9, 6, 9, 7, 7, 1, 0,
 subsodef(1:7,1)    = 6, 9, 6, 9, 7, 8, 1,
 zeroddef(1:7,1)    = 6, 9, 6, 9, 8, 8, 1,
 zeroddef(1:7,2)    = 6, 9, 6, 9, 7, 7, 1,
 zeroddef(1:7,3)    = 6, 9, 6, 6, 7, 8, 1,
 zeroddef(1:7,4)    = 6, 9, 9, 9, 7, 8, 1,
 zeroddef(1:7,5)    = 6, 6, 6, 9, 7, 8, 1,
 zeroddef(1:7,6)    = 9, 9, 6, 9, 7, 8, 1,
 vbc(1:10,1)        = 6, 9, 6, 9, 8, 8, 1, 1, 0.0, 800.0,
 vbc(1:10,2)        = 6, 9, 6, 9, 7, 7, 1, -2, 0.0, 800.0,
 vbc(1:10,3)        = 6, 9, 6, 6, 7, 8, 1, 3, 0.0, 800.0,
 vbc(1:10,4)        = 6, 9, 9, 9, 7, 8, 1, -4, 0.0, 800.0,
 vbc(1:10,5)        = 6, 6, 6, 9, 7, 8, 1, 5, 0.0, 800.0,
 vbc(1:10,6)        = 9, 9, 6, 9, 7, 8, 1, -6, 0.0, 800.0,
 vvalue             = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
...
$end
```

Table 5-5 SORTAM file for Example 5

Example SORTAM file for 1 surface + outflow
ncols
11
ivvalues ivtypes source_fraction
0 0
0 0
0 0
0 0
0 0
1 1 +0.3
2 1 -0.3
3 1 +0.1
4 1 -0.1
5 1 +0.1
6 1 -0.1

time(s)	P (d/cm^2)	T (K)	xh_2o	xh_2ol	xh_2	mdot1 (g/s)	mdot2 (g/s)	mdot3 (g/s)	mdot4 (g/s)	mdot5 (g/s)	mdot6 (g/s)
0	5.6E6	4.04E2	4.87E-1	5.50E-2	4.58E-1	1.5E3	1.5E3	1.5E3	1.5E3	1.5E3	1.5E3
100	5.1E6	4.00E2	4.10E-2	3.32E-1	6.27E-1	1.2E3	1.2E3	1.2E3	1.2E3	1.2E3	1.2E3
200	5.3E6	3.98E2	6.00E-2	3.60E-1	5.80E-1	1.1E3	1.1E3	1.1E3	1.1E3	1.1E3	1.1E3
300	5.3E6	3.95E2	3.00E-3	4.54E-1	5.43E-1	9.9E2	9.9E2	9.9E2	9.9E3	9.9E3	9.9E3
400	5.5E6	3.92E2	5.63E-1	1.22E-1	3.15E-1	1.2E3	1.2E3	1.2E3	1.2E3	1.2E3	1.2E3
500	5.8E6	3.92E2	6.21E-1	1.48E-1	2.31E-1	1.1E3	1.1E3	1.1E3	1.1E3	1.1E3	1.1E3
600	5.9E6	3.92E2	5.31E-1	3.14E-1	1.55E-1	8.8E2	8.8E2	8.8E3	8.8E3	8.8E3	8.8E3
700	5.5E6	3.90E2	3.21E-1	6.75E-1	4.00E-3	5.6E2	5.6E2	5.6E2	5.6E2	5.6E2	5.6E2
800	5.5E6	3.90E2	3.19E-1	6.78E-1	3.00E-3	5.6E2	5.6E2	5.6E2	5.6E2	5.6E2	5.6E2

6 Definition of Solid Heat Structures

In Section 3.3, we discussed how to define solid walls and obstacles in the computational mesh (with the variables **walls** and **mobs** in NAMELIST group xput) to restrict the fluid flow path. Our convention is that within the mesh, walls are surfaces and obstacles are volumes which the fluid is not allowed to penetrate. If heat transfer is invoked (by setting the variable **ihtflag** = 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 (backward Euler **teta** = 1.0) or a semi-implicit scheme (Crank Nicholson **teta** = 0.5). As an approximation, which greatly improves computational efficiency and speed, we assume that heat conduction is one-dimensional, 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. In addition, we can include so-called distributed heat sinks, which are planar walls of a given volume thickness and material. These sinks do not occupy any fluid volume; one can think of them as having an area per unit flow volume. They are exposed to the fluid on the negative side of the sink. The inside of the sink (the positive side of the sink structure) can be simulated with various boundary conditions that will be explained later.

6.1 Wall/Slab Heat Structures

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. In general a wall heat structure is two-sided, with its temperature profile determined by the adjacent fluid cell temperatures on both sides and by 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. In GASFLOW serial version, **slabthk,** an input variable in NAMELIST group rheat, is used to determin if a heat structure is a wall or a slab. If the thickness is smaller than **slabthk**, 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. In current GASFLOW-MPI, we don’t support **slabthk**. It means **walls** can be only treated as wall heat structures, and **mobs** can be only treated as slab heat structures.

A solid surface belongs to a slab heat structure if the solid material is defined in **mobs**. In the default option, a slab is considered infinitely thick so that within the problem time scale, the heat or temperature wave due to exchange with the fluid never penetrates deep enough to affect the temperature profile near its back side. 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. The initial option has been extended to also define slabs with boundary conditions on the back side. This will be discussed in a later part. 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, if a material number is specified for the boundary material (**matbdy** >0). If **matbdy** is set to zero, the boundary of the computational domain is adiabatic.
2. For obstacles (defined by **mobs** and associated with a material number > 0), each surface exposed to fluid will be treated as an independent slab surface.

Note that the user does not directly define obstacles as slab or wall heat structures. The code automatically determines the heat structure type of surfaces corresponding to closed computational boundaries, which are generally treated as slabs, and the heat structure type of the solid structures defined by **walls** and **mobs**.

6.2 Heat Conduction in Wall Heat Structures

Regardless of whether a heat structure is a slab or wall, the conduction calculation requires some information about 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 mathematical 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 through the **walldef** array in NAMELIST group rheat:

walls(8,*) Integer to identify the type of wall through the **walldef** array that stores the material identification number and thickness for each wall type (see Section 3.3.1).

walldef(1,*) Material identification number. Table 6-1 shows the available pre-defined data from the material data base. See also **wltabsink** and **rcptabsink.**

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

walldef(3,*) Sets boundary condition for first fluid/wall surface (BC#1).
walldef(3,*) = 0.0 implies a fluid-wall heat exchange (default);
walldef(3,*) > 0.0 implies a constant wall temperature boundary condition with T = **walldef(3,*);** and
walldef(3,*) < 0.0 implies an adiabatic wall boundary condition.

walldef(4,*) Sets BC#2 for last fluid/wall surface.
walldef(4,*) = 0.0 implies a fluid-wall heat exchange (default);
walldef(4,*) > 0.0 implies a constant wall temperature boundary condition with T =

walldef(4,*); and
walldef(4,*) < 0.0 implies an adiabatic wall boundary condition.

walldef(5,*) δx for the first node in the wall.
walldef(5,*) = 0.0 implies a uniform mesh spacing for heat-conduction nodes; and
walldef(5,*) > 0.0 implies a variable mesh spacing for heat-conduction nodes with **walldef(5,*)** = δx of surface heat-conduction node on both sides of the wall.

walldef(6,*) Fraction of wall area from mesh surface that is used for heat transfer.

walldef(7,*) Flag for further specification of BC on negative side of wall:
walldef(7,*) = 0 no further modification;
walldef(7,*) > 0 gives table number from **surftab** that specifies time-dependent surface temperature, with the initial temperature at *t*=0 taken from **walldef(3,*)**;
walldef(7,*) = –1 applies heat flux **walldef(9,*)** and/or heat transfer with coefficient **walldef(10,*)** and applies fluid temperature **walldef(3,*)** on negative side.

walldef(8,*) Flag for further specification of BC on positive side of wall:
walldef(8,*) = 0 no further modification;
walldef(8,*) > 0 gives table number from **surftab** that specifies time dependent surface temperature, with the initial temperature at *t*=0 taken from **walldef(4,*)**;
walldef(8,*) = –1 applies heat flux from **walldef(9,*)** and/or heat transfer with coefficient **walldef(10,*)** and applies fluid temperature **walldef(4,*)** on positive side.

walldef(9,*) Heat flux [erg/cm^2·s] applied as BC by **walldef(7,*)** or **walldef(8,*)**. Positive flux means add heat to the wall (i.e., condensation), which is the same convention as in the fluid wall condensation/vaporization heat flux from GASFLOW-MPI.

walldef(10,*) Heat-transfer coefficient [erg/(cm^2·s·K)] applied as BC by **walldef(7,*)** or **walldef(8,*)**.

GASFLOW-MPI offers a choice among the 20 solid conducting materials given below. Selecting a value of 0 for the material number causes the structure to be disregarded from heat conduction. Material numbers 7 to 12 and 13 to 18 in the property library denote identical structure materials. This was intentionally made to allow the user to assign identical material properties and give structures for 3D visualization different material numbers to selectively only display structures with selected material numbers (see parameter **matdef** and **matpanel**) . The user can of course redefine these properties by tabular input to his needs.

Table 6-1 Heat Transfer Material Data Base

No.	material	ρ (g/cm^3)	C_P (erg/g·cm·K)	k (erg/s·cm·K)	α (cm^2·s)	emissivity
1	US concrete	2.4	1.00E+07	2.00E+05	8.30E-03	1
2	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
3	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
4	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
5	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
6	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
7	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
8	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
9	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
10	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
11	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
12	GRS concrete	2.225	8.79E+06	2.10E+05	1.07E-02	1
13	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
14	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
15	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
16	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
17	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
18	Steel	7.85	4.90E+06	5.00E+06	1.30E-01	1
19	superconduct	10	1.00E+09	1.00E+20	1.00E+10	1
20	insulator	1.00E-10	1.00E-10	1	1.00E+20	1

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 (see Section 3.3.2):

mobs(8,*) Material identification number. Options are
= 0: disregard heat conduction,
> 0: number selects material from table 6-1

If a zero value is specified for the material number, this particular structure is not included in the heat-conduction and heat-transfer simulation; it is counted neither as a slab nor a wall for heat transfer but only serves as a wall or an obstacle for the flow simulation. When heat transfer is turned on (**ihtflag** > 0), the zero material number allows GASFLOW-MPI to selectively leave out certain

structures from heat exchange (for instance walls from rupture disks which cannot be easily removed otherwise) but still consider those with nonzero material numbers.

The GASFLOW-MPI code also allows the specification of additional heat-conducting materials, which can be defined explicitly in the input. The input parameter (**mpreset** default 6) automatically loads the data from material number 1 to mpreset from Table 6-1 into a property library. Every material number > **mpreset** requires a table input for the thermal conductivity and the product of ρC_P. This table input must then be specified for walls, slabs, and sinks (see definition below) according to the input numbers **nhteslab**, **nhtesink**, and **nhtewall** which describe heat-conducting elements in slabs, walls, and sinks. This table input allows the simulation of composite structures (i.e., a liner on top of a concrete structure) in which thermal conductivities and heat capacities can vary from node to node. Note that these tables must always be specified for all structure types even though some material numbers may be applied only in a wall. The extended options above **walldef(6,*)** require the specification of the following additional data in block rheat:

ntotmat	Total number of structure materials. (**ntotmat** <=20, Default =6).
mpreset	Materials from **imat** = 1 to **mpreset** in Table 6-1 are automatically loaded into the property tables for each of the nhteslab, nhtewall, and nhtesink elements used in the 1D heat conduction simulation (Default =6).
nhteslab	Number of 1D heat conduction elements in a slab heat structure (<100).
nhtewall	Number of 1D heat condution elements in a wall heat structure (<100).
nhtesink	Number of 1D heat conduction elemenst in a sink heat structure (<100).
wltabslab(*,imat)	Thermal conductivity table for material **imat** for each of the **nhteslab** elements of the slab structure. Tables for Materials 1 to **mpreset** are automatically filled with constant values from Table 6-1. Tables must be input nodewise only for material numbers from **mpreset+1** to **ntotmat**.
wltabwall(*,imat)	Thermal conductivity for material **imat** for each of the **nhtewall** elements of the wall structure. Tables for Materials 1 to **mpreset** are automatically filled with constant values from Table 6-1. Tables must be input nodewise only for material numbers from **mpreset+1** to **ntotmat**.
wltabsink(*,imat)	Thermal conductivity for material **imat** for each of the **nhtesink** elements of the sink structure. Tables for Materials 1 to **mpreset** are automatically filled with constant values from Table 6-1. Tables must be input nodewise only for material numbers from **mpreset+1** to **ntotmat**.
rcptabslab(*,imat)	ρC_P for material **imat** for each of the **nhteslab** elements of the slab structure. Tables for Materials 1 to **mpreset** are automatically filled with constant values from Table 6-1. Tables must be input nodewise only for material numbers from **mpreset+1** to **ntotmat**.
rcptabwall(*,imat)	ρC_P for material **imat** for each of the **nhtewall** elements of the wall structure. Tables for Materials 1 to **mpreset** are automatically filled with constant values from Table 6-1. Tables must be input nodewise only for material numbers from **mpreset+1** to **ntotmat**

rcptabsink(*,imat) ρC_P for material **imat** for each of the **nhtesink** elements of the sink structure. Tables for Materials 1 to **mpreset** are automatically filled with constant values from table 9-1. Tables must be input nodewise only for material numbers from **mpreset+1** to **ntotmat**.

surftab(2,j,i) Pair of time [s] and temperature [K] at time t_j for temperature table i. Maximum number of points per table is 50 and the maximum number of tables is 30. (The problem time must never exceed the maximum of the table time.)

twall0 Initial wall surface temperature on the wall side for which **walldef** does not define the surface temperature (**walldef(3,*)** and/or **walldef(4,*)** = 0). If **twall0** < 0, then the surface temperature from the adjacent fluid node is applied on the undefined sides of the wall.

Note that one must not input the fluid conditions and/or heat flux for the side of the wall. This would overspecify the problem and the code would automatically stop with a message. It is possible however to specify the temperature on both sides of the wall, thus defining a thermal boundary condition for fluid heating or cooling. Defining a wall with specified fluid conditions on one side or making the boundary condition adiabatic on one side makes sense only if the input conditions apply to a side of the wall that does not face a fluid. This could be the inner side of a wall on top of an obstacle or a wall side facing the boundary of the computational mesh. The code is currently set up in such a way that any wall on top of a slab replaces that particular slab, which allows quite a flexible definition of the structure boundary conditions.

The initial temperature profile across the wall is evaluated from the surface temperatures on the two sides of the wall, taking **twall0**, the temperature of the adjacent GASFLOW-MPI fluid node, or the input surface temperature [**walldef(3,*)** and/or **walldef(4,*)**], whichever applies. If one side of the wall is adiabatic, the code initiates a flat temperature profile across the wall with the wall surface temperature on the fluid side. The steady-state profile includes the effect of varying thermal conductivities. A flat steady-state temperature profile under nonadiabatic conditions on any wall side requires input so that either **twall0** > 0 or **twall0** < 0.0 with a uniform fluid temperature on both sides of the wall, leaving the input values **walldef(3,*)** and **walldef(4,*)** zero.

When specifying a heat flux and/or heat-transfer coefficient with a fluid temperature on one side of the wall, the code determines the wall surface temperature on this side from the input data and calculates a steady-state heat flux through the wall from this data. The surface temperature on the other (fluid) side is applied according to the fluid or **twall0** specification. The surface temperature on the fluid side and the calculated steady-state heat flux are used to calculate the steady-state temperature profile across the wall again, accounting for varying thermal conductivities from layered structures, if the wall material chosen is greater than 3.

When cylindrical coordinates are chosen (**cyl** = 1), heat conduction through walls in a radial direction (walls in plane 1) is calculated in cyclindrical coordinates under both steady-state and transient conditions.

Examples:

Wall of 10 cm thickness with 29 heat-conducting elements, fluid onboth sides, and composite material. (All but the 15th structure element is made of concrete; the 15th structure element is made of steel.) Use backward Euler scheme (**teta** = 1.0 = default). Use of dynamic mesh expansion with a small surface node of 0.01 cm on both sides of the wall (example shown in Figure 6–1 for 10 elements only).

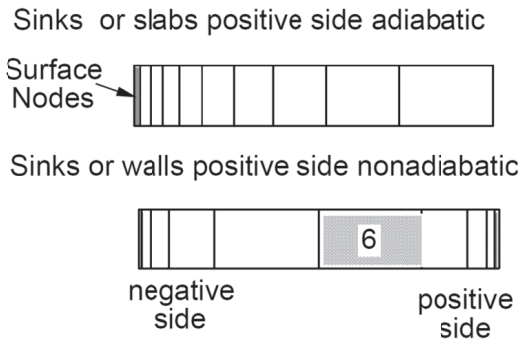


Figure 6-1 Noding scheme for 10 heat-conducting elements with small surface nodes and internal mesh expansion in sinks, slabs, and walls. Input defines thickness of surface nodes. (6 indicates the sixth structural volume counting from negative toward positive)

(Test1 input)
$xput
…
gasdef(1:14,1) = 1, 4, 1, 2, 1, 4, 1,1.0e6, 400.0, 2, 0.0, 0.0, 'air', 1.0,
walls(1:8,1) = 3, 3, 1, 2, 1, 2, 1, 1,
...
$end

$rheat
...
teta = 1.0,
ihtflag = 1,
ntotmat = 4,
nhteslab = 20,
nhtesink = 10,
nhtewall = 29,
twall0 = -1,
walldef(1:5,1) = 4, 10.0, 0.0, 300.0, 0.01,
wltabwall(1:29,4) = 14*2.0e+5, 5.0e+6, 14*2.0e+5,
wltabslab(1:20,4) = 10*2.0e+5, 5.0e+6, 9*2.0e+5,
wltabsink(1:10,4) = 5*2.0e+5, 5.0e+6, 4*2.0e+5,

rcptabwall(1:29,4) = 14*6.25924e+7, 3.84964e+7, 14*6.25924e+7,
rcptabslab(1:20,4) = 10*6.25924e+7, 3.84964e+7, 9*6.25924e+7,
rcptabsink(1:10,4) = 5*6.25924e+7, 3.84964e+7, 4*6.25924e+7,
…
$end

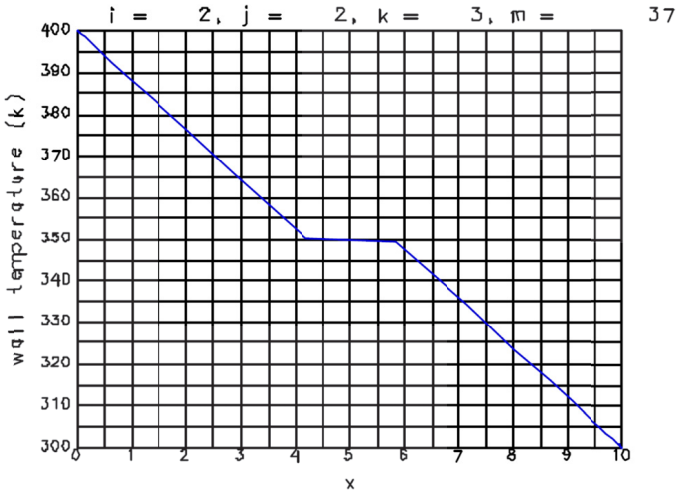


Figure 6-2 Steady-state solution for Example 1 (the 10 cm thick composite structure made up of a sandwich of concrete, steel, andconcrete)

1. Concrete wall on top of an obstacle with specified heat flux of 10^{10} erg/cm^2-s on the obstacle side of the wall and fluid conditions on the fluid side. Solve with Crank Nicholson scheme (**teta** = 0.5).

   (Test2 input)
   $xput
   …
   gasdef(1:14,1) = 1, 4, 1, 2, 1, 4, 1, 1.0e6, 400.0, 2, 0.0, 0.0, 'air', 1.,
   mobs(1:8,1) = 2, 3, 1, 2, 2, 3, 1, 1,
   walls(1:8,1) = 2, 2, 1, 2, 2, 3, 1, 1,
   …
   $end

   $rheat
   …
   ihtflag = 1,
   teta = 0.5,
   nhtewall = 29,
   twall0 = -1.0,
   walldef(1:9,1) = 1, 10.0, 0.0, 1.0, 0.01, 1.0, 0, -1, 1.0e+10,
   …
   $end

In Example 2, a steady-state temperature profile is calculated across the wall with the surface temperature 400 K on the fluid side. The steady-state wall surface temperature on the obstacle side is determined from the input heat flux. Note that the fluid conditions on the obstacle side must be specified by a number greater than zero if the heat flux is to continue to be applied during the transient also. If the heat flux is only applied to set up a steady-state temperature profile, one can set **walldef(4,1)** to zero. In this case, a steady-state profile is set up, but the obstacle side of the wall is defaulted to adiabatic conditions.

2. If one wants to start from a flat profile of 400 K across the wall before applying the input heat flux, one can use the **walldef** statement only, with the boundary temperature on the obstacle side specified to be equal to the fluid temperature in the startup and with the heat flux specified only in the restart.

   (Test3 start)

   walldef(1:5,1) = 1, 10., 0., 400., 0.01,

   (Test3 restart)

   walldef(1:9,1) = 1, 10., 0., 400., 0.01,1.,0,-1,1.e+10,

We have applied this procedure in Example 3 to check the transient heat-conduction solution from GASFLOW with analytical results for a 10 cm steel and concrete wall.

Figure 6-3 compares the predicted surface temperature vs. time development on the obstacle side with analytical predictions. Results from a calculation of the same transient with 10 heat-conduction nodes are also displayed in

Figure 6-3, which documents that this discretization is not sufficient for this strong heat flux.

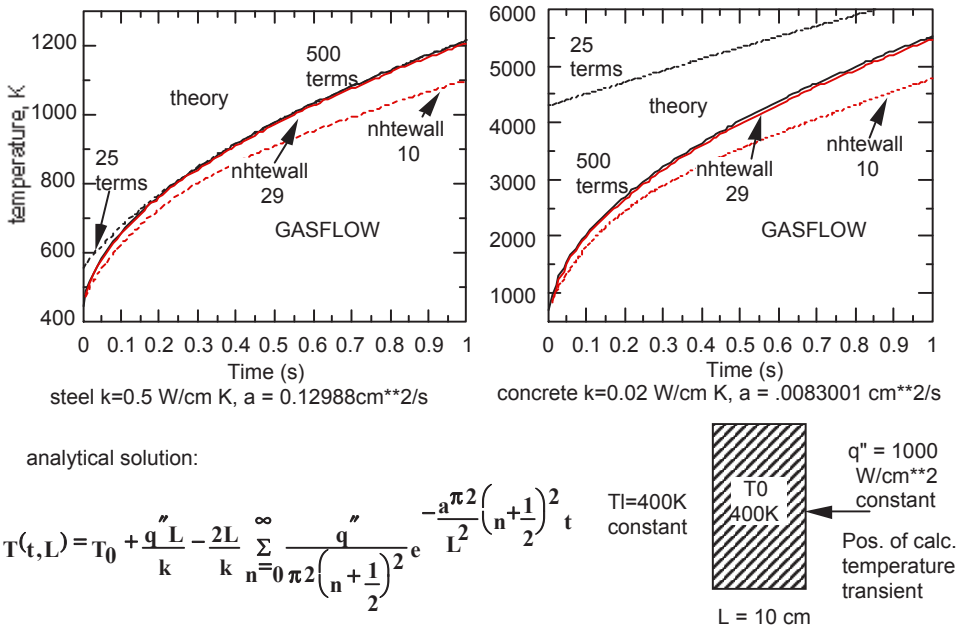


Figure 6-3 wall surface temperature

Figure 6-3 shows GASFLOW calculated the wall surface temperature on the positive side and an analytical solution for 1D heat conduction through a 10 cm wall of steel or concrete starting at 400 K isothermal condition. A constant surface temperature of 400 K was applied on the negative side and a constant heat flux of 1000 W/cm^2 was given on the positive side. Results are for 29 and 10 heat-conduction elements and internal mesh expansion with equal surface elements of 0.01cm. Only very small difference can be found between Crank Nicholson (teta = 0.5) and backward Euler (teta = 1.0).

3. The same wall but with input of a temperature transient of the **surftab** array for the surface of the positive side, simulated with the input:

```
(Test4 input)
$xput
…
 twfin              = 1.0,
 gasdef(1:14,1)     = 1, 4, 1, 2, 1, 4, 1, 1.0e6, 400.0, 2, 0.0, 0.0, 'air', 1.0,
 mobs(1:8,1)        = 2, 3, 1, 2, 2, 3, 1, 1,
 walls(1:8,1)       = 2, 2, 1, 2, 2, 3, 1, 1,
…
$end

$rheat
…
 ihtflag            = 1,
 teta               = 0.5,
 nhtewall           = 29,
 twall0             = -1.0,
 walldef(1:8,1)     = 1, 10.0, 0.0, 400.0, 0.01, 0.9, 0, 2,
 surftab(1,1,2)     = 0.0, 500.0,
                      1.0, 800.0,
…
$end
```

In this example, the wall surface temperature on the positive side increases linearly from 500 K to 800 K within 1 s. The steady-state profile and the steady-state surface temperature on the obstacle side have been set to 400 K. The table values are only applied during the transient. Thus this input assumes that the surface temperature will jump to the initial value of 500 K at the beginning of the transient for a flat internal profile of 400 K from the steady-state initialization. Note that the problem time **twfin** must be within the maximum time specified in **surftab**. We have additionally decreased the wall area for the heat transfer by a factor of 0.9 relative to the wall area defined by the mesh surface.

6.3 Heat Conduction in Sink Heat Structures

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 Figure 6–4. 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 1 for GASFLOW-MPI).

sinkdef(8,*) Material identification number (defaults Table 6-1).

sinkdef(9,*) Total material volume volsink (cm^3).

sinkdef(10,*) Average material thickness avgthick (cm).

sinkdef(11,*) Sets BC for sink/fluid surface (BC#1).
sinkdef(11,*) = 0.0 implies that the BC#1 will be sink-fluid heat exchange (default).
sinkdef(11,*) > 0.0 implies a wall temperature boundary condition of T = **sinkdef(11,*)** on the fluid side.

sinkdef(12,*) Sets the sink centerline BC (BC#2).
sinkdef(12,*) = –1.0 implies an adiabatic BC(default) will be applied at the sink centerline.
sinkdef(12,*) > 0.0 implies a temperature boundary condition will be applied at the sink centerline of T = **sinkdef(12,*)**.

sinkdef(13,*) δx for the first node in the sink.
sinkdef(13,*) = 0.0 implies uniform mesh spacing for heat-conduction nodes.
sinkdef(13,*) > 0.0 implies variable mesh spacing for heat-conduction nodes, with **sinkdef(13,*)** = *dx* of first heat-conduction node if internal BC is adiabatic and **sinkdef(13,*)** = *dx* on outer and inner sink surface nodes if internal BC is nonadiabatic.

sinkdef(14,*) Flag for further specification of BC on negative side:
= 0 no further modifications (default);
> 0 gives table number from **surftab** that specifies time-dependent surface temperature. The initial temperature at *t*=0 is taken from **sinkdef(11,*)**.

sinkdef(15,*) Flag for further specification of BC on positive side (centerline) of slab:
= 0 no further modification (default);
> 0 gives table number from **surftab** that specifies time-dependent surface temperature, with the initial temperature at *t*=0 taken from **sinkdef(12,*)**;
= –1 applies heat flux from **sinkdef(16,*)** and/or heat transfer with coefficient **sinkdef(17,*)** and fluid temperature **sinkdef(12,*)** on positive side (positive heat flux means add energy to the sink).

sinkdef(16,*) Heat flux [erg/cm^2·s] applied as BC by **sinkdef(15,*)**.

sinkdef(17,*) Heat-transfer coefficient [erg/cm^2·s·K] applied as BC by **sinkdef(15,*)**.

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 of another one will get twice as much sink material. According to our model, depicted in

Figure 6-3, specifying the volume and thickness of the sink material will also give the surface area through which heat exchange with the fluid occurs. The following relation exists between the sink volume volsink, the input thickness avgthick, and the total surface to which the fluid is exposed:

thickness = avgthick/2
area= volsink / thickness

thickness denotes the actual thickness of the sink structure in the heat-conduction solution.

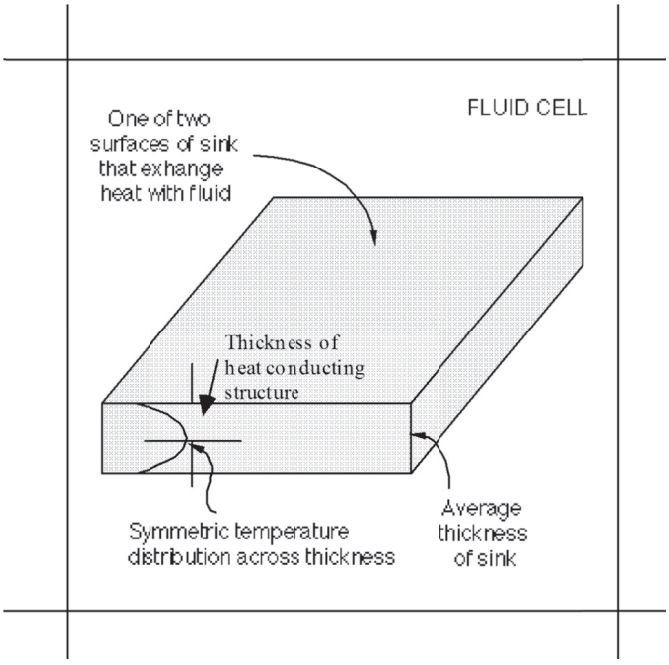


Figure 6-4 Schematic representation of the distributed heat sink used in GASFLOW-MPI

Example.

Sink structure of steel with positive (central) BC defined by heat transfer from an outside fluid of 373 K. Apply 10 elements for heat conduction and a small surface node of 0.01 cm on both sides. Second sink applies surface temperature table 1 for temperature transient on the positive (central side).

```
$rheat
…
 ihtflag          = 1,
 nhtesink         = 10,
 tsink0           = -1,
 sinkdef(1:17,1)  = 2, 4, 1, 2, 1, 2, 1, 2, 8000., 10., 0.0, 873., 0.01, 0, -1, 0., 10000.,
 sinkdef(1:15,2)  = 2, 3, 1, 2, 2, 3, 1, 2, 4000., 10., 0.0, 400., 0.01, 0, 1,
 sinkdef(1:13,3)  = 2, 4, 1, 2, 2, 3, 1, 2, 4000., 10., 0.0, -1.0, 0.01,
 surftab(1,1,1)   = 0.0, 500., 1., 800.,
…
$end
```

Regarding distributed heat sinks, the first **sinkdef** assigns sinks to two fluid cells with (i,j,k) indices of (3,2,2) and (4,2,2). (Refer to Figure 3–1 for the cell numbering convention used in GASFLOW-MPI) The sink material is steel because **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 5 cm, and a total surface area 800 cm^2. The second **sinkdef** specifies only one fluid cell, (3,2,3), and the heat sink in that cell also has a volume of 4000 cm^3 and a thickness of 5 cm. The third sink is adiabatic at the sink centerline. In this case, the dynamic mesh expansion only defines the small input node of 0.01cm on the fluid side and monotonically increases the mesh size to the inner boundary at 5 cm thickness (see also Figure 6–1). The third sink partially overlaps the fluid node of the second sink. Because sinks do not directly interfere with each other, one can specify more than one sink statement to apply to the same fluid node. All sinks have the surface temperatures on their negative sides (fluid sides) specified by the fluid temperature because no **tsink0** > 0 has been specified in the input. The surface temperature at the positive (inner) sink side of the first sinkdef statement is calculated with the temperature from an outside fluid and the heat-transfer coefficient of 10,000 ergs/cm^2K specified in the input. A steady-state profile is then set up with a linear increase from the temperature on the fluid side to the calculated surface temperature on the inner side of the sink.

6.4 Other Heat Structure Input

The heat-conduction equations are solved by finite difference. The finite differences are applied to a one-dimensional mesh that is either planar or cylindrical depending upon the value of **cyl** (**cyl** = 0 implies planar conduction and **cyl** = 1 implies cylindrical conduction in the i-coordinate direction and planar conduction in the j- and k-coordinate directions). 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. (Max < 100)

nhtewall Number of discretized elements to be used for calculating heat conduction in a wall heat structure. Default = 2. (Max < 100)

nhteslab Number of discretized elements to be used for calculating heat conduction in a slab heat structure. Default = 2. (Max < 100)

In GASFLOW-MPI, all solid heat structures can be defined 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. This default initialization of the structure temperatures can change when the boundary conditions specify different temperatures on both sides of the structures or when heat transfer from external fluid conditions or heat fluxes are specified on one side of the structure at the onset of the analysis.

6.5 Heat Conduction in Slab Heat Structures (Boundary Cells)

The slab temperature boundary conditions for all blocks and all boundary cells can either be input as a single number (see below) or as the **slabdef** array which allows a slab's thickness, initial temperatures, boundary conditions, and material type to be dependent upon the block number and upon whether the boundary cell is on the east, west, north, south, top, or bottom boundary. **tslabbc** and **slabdef** are described below:

tslabbc Inner slab temperature boundary condition (default –9.123) for all blocks and boundaries (uniform). Can be overridden individually for each of the 6 slab planes using **slabdef**. **tslabbc** < 0 implies adiabatic BC. **tslabbc** > 0.0 implies fixed temperature BC on the inside of the slab, with *T*(bc) = **tslabbc**.

tslab0 Initial slab temperature (default 300 K) can be overridden using **slabdef**. If **tslab0** < 0, use temperature of the adjacent fluid node to define initial slab temperature. A flat temperature profile with **tslab0** between the fluid side and the inner slab side is initiated. If **tslabbc** > 0 and **islablin** = 1, a linear steady-state profile from the fluid surface to **tslabbc** at the inner surface of the slab is set up. If **tslabbc** > 0 , **islablin** = 1, and **hslablin** > 0, a linear profile between the slab surface temperature on the fluid side and the inner slab temperature **tslabbc** is set up.

The slab surface temperature on the fluid side is then calculated from steady-state fluid to slab heat transfer on the fluid side with **tslab0** being the fluid temperature and **hslablin** the heat-transfer coefficient on the fluid side of the slab. **tslabbc** is the surface temperature on the inside (positive side) of the slab.

slabdef(25,*) Sets **tslab0**, **tslabbc**, and **slabthk** for specific boundaries and blocks. Conditions on the east boundary generally apply to the slabs on the west side of all obstacles. Correspondingly, the west side boundary conditions are applied to all slabs on the east side of obstacles, and so on similarly for the other four planes.

slabdef(1,n) Block number (must be 1 for GASFLOW-MPI).

slabdef(2,n) Thickness of slabs on east boundary.

slabdef(3,n) Thickness of slabs on west boundary.

slabdef(4,n) Thickness of slabs on north boundary.

slabdef(5,n) Thickness of slabs on south boundary.

slabdef(6,n) Thickness of slabs on top boundary.

slabdef(7,n) Thickness of slabs on bottom boundary.

slabdef(8,n) Initial temperature for slabs on east boundary (applied instead of **tslab0**).

slabdef(9,n) Initial temperature for slabs on west boundary(applied instead of **tslab0**).

slabdef(10,n) Initial temperature for slabs on north boundary(applied instead of **tslab0)**.

slabdef(11,n) Initial temperature for slabs on south boundary(applied instead of **tslab0**).

slabdef(12,n) Initial temperature for slabs on top boundary(applied instead of **tslab0**).

slabdef(13,n) Initial temperature for slabs on bottom boundary(applied instead of **tslab0)**.

slabdef(14,n) Temperature BC for slabs on east boundary(applied instead of **tslabbca**).

slabdef(15,n) Temperature BC for slabs on west boundary(applied instead of **tslabbca**).

slabdef(16,n) Temperature BC for slabs on north boundary(applied instead of **tslabbca**).

slabdef(17,n) Temperature BC for slabs on south boundary(applied instead of **tslabbca**).

slabdef(18,n) Temperature BC for slabs on top boundary(applied instead of **tslabbca**).

slabdef(19,n) Temperature BC for slabs on bottom boundary(applied instead of **tslabbca)**.

slabdef(20,n) Material type for slabs on east boundary(applied instead of **matbdy**).

slabdef(21,n) Material type for slabs on west boundary(applied instead of **matbdy)**.

slabdef(22,n) Material type for slabs on north boundary(applied instead of **matbdy**).

slabdef(23,n) Material type for slabs on south boundary(applied instead of **matbdy**).

slabdef(24,n) Material type for slabs on top boundary(applied instead of **matbdy)**.

slabdef(25,n) Material type for slabs on bottom boundary(applied instead of **matbdy**).

islablin Flag to set an approximate linear initial temperature profile in each slab cell based on temperature of slab BC#2 and fluid cell contiguous with BC#1 (default is 0, no linear initial temperature profile).

hslablin Heat-transfer coefficient on slab BC#1. Used in evaluating the linear slab cell temperature profile (default is 1000).

dxslabc **dxslabc** = 0.0 results in uniform mesh spacing for all concrete slabs. **dxslabc ≠ 0** results in a variable mesh spacing with for all concrete slabs. The first δx for each concrete slab will be **dxslabc** (default is zero).

dxslabs — **dxslabs** = 0.0 results in uniform mesh spacing for all nonconcrete slabs. **dxslabs ≠ 0** results in a variable mesh spacing with for all nonconcrete slabs. The first δx for each nonconcrete slab will be **dxslabs** (default is 0).

matbdy — Material type (uniform) for the mesh boundary slabs in all blocks. Can be overridden for specific blocks and boundaries using **slabdef**. **matbdy** = 0 results in no boundary slabs.

A material number > 0 specified for a boundary of the computational mesh implies that this particular boundary side is simulated as a slab of the material represented by the specified number. If nothing more is specified, a flat temperature profile with **tslab0** (default 300 K) is initiated and the inner side of the slab is maintained adiabatic. If **tslab0** < 0, a flat profile with the adjacent fluid temperature is specified and an adiabatic BC is applied at the inside. If a value greater than zero is input for **tslabbc,** the BC on the inner (positive) surface of the slab uses a constant surface temperature **tslabbc**. The analysis still starts from a flat profile with **tslab0,** which can cause a sudden jump at the inner slab surface if **tslabbc** is specified different from **tslab0**. Only if **islablin** is set to 1 is a linear steady-state profile set up between the slab surface temperature **tslab0** on the fluid side and the inner slab surface temperature **tslabbc**. If **hslablin** is also specified, the slab surface temperature on the fluid side is calculated with a steady-state fluid slab heat transfer assuming that **tslab0** is the fluid temperature on the fluid side and **tslabbc** is the surface temperature on the inner side of the slab.

The **slabdef** statements allow different definitions for each of the six boundaries of the computational mesh of the block they are defined for. But they apply at the same time also to the obstacle planes inside. The conditions for the computational boundary on the east side define the west side of the boundary slab and are also used to simulate the west side of the obstacle slabs inside. In the same way, the conditions for the top boundary define the bottom side of the top boundary slabs and the bottom side of the internal obstacle slabs.

Examples:

1. Obstacle slab made of steel with specified inner temperature of 400 K and with the initial temperature taken from the adjacent fluid node. To avoid a jump in temperature at the inner surface, we set the temperature of the initial fluid to the inner slab surface temperature.

```
$xput
…
 gasdef(1:14,1)    = 1, 4, 1, 2, 1, 4, 1, 1.0e6, 400., 2, 0., 0.0, 'air', 1.0,
 mobs(1:8,1)       = 2, 3, 1, 2, 2, 3, 1, 2,
…
$end

$rheat
…
 ihtflag           = 1,
```

```
 nhteslab      = 20,
 tslab0        = -1,
 tslabbc       = 400,
…
$end
```

2. Obstacle in block 1 made of concrete with specified inner temperature of 400 K in the west slab, 500 K in the east slab, and 600 K in the top, bottom, south, and north slab. Initial fluid temperature is 300 K. The surface node on the fluid side has a size of 0.01 cm. A steady-state profile is set up with a heat-transfer coefficient of 1000 erg/cm^2·s·K on the fluid sides of the six slabs that bound the obstacle. The boundary cells are considered adiabatic (material numbers for boundaries of the mesh all set to zero).

```
$meshgn
…
 iblock            = 1,
 xgrid             = 0., 50., 100.0, 150.,
…
$end

$xput
…
 gasdef(1:14,1)    = 1, 4, 1, 2, 1, 4, 1, 1.0e6, 400., 2, 0., 0., 'air', 1.,
 mobs(1:8,1)       = 2, 3, 1, 2, 2, 3, 1, 1,
…
$end


$rheat
…
 ihtflag           = 1,
 nhteslab          = 20,
 slabdef(1:25,1)   = 1, 6*30.0, 6*-1.0, 400.0, 500.0, 4*600.0, 6*0,
 dxslabc           = 0.01,
 islablin          = 1,
 hslablin          = 1000.0,
…
$end
```

Because the obstacle extends to the mesh boundary on the south and north side, no slabs are defined for the south and north side of this obstacle. The heat-conduction equations for this example are solved for four slabs only.

Note that the user has the option to choose different obstacle materials and different materials for each boundary of the block. The code is currently not set up, however, to specify different sizes of the surface nodes for different obstacles or to apply other than the specified boundary conditions. These are defined for each plane within one block but must be used in each obstacle at this particular plane. It is thus not possible to use different inner boundary conditions for different obstacles in the same plane. Note also that the decision whether an obstacle is considered a slab or a wall is always made based on the input parameter **slabthk** only. Therefore, this parameter must be specified together with **slabdef(2–7,1)**, which applies only to the mesh boundaries. But it has been shown before that defining walls on top of an obstacle side allows the user to vary the boundary conditions on any desired obstacle side and thus to overcome the restrictions that currently exist in the simulation of slab heat structures.

6.6 Background for Defining Steady-State Temperature Profiles

The steady-state temperature profile in the heat-conducting structures can either be flat or have an initial gradient. If one side of the structure is modeled as adiabatic, the initial temperature profile is always flat and defined from the surface temperature of the opposite side. Steady-state temperature profiles are only simulated for nonadiabatic boundary conditions on either side of the structure. Different thermal conductivities for composite layers must be accounted for in the initial profile, if a gradient is simulated. The temperature gradient across the structure is calculated from a steady-state heat flux q''_{ss} for planar walls or linear heat rating q'_{ss} for cyclindrical structures.

Planar wall

$$T_j = \begin{cases} T_1 & for \quad j = 1 \\ T_{j-1} - \dfrac{\delta x_j}{k_j} q''_{ss} & for \quad j = 2, NHTE \\ T_{NHTE+1} & for \quad j = 1 \end{cases} \qquad \text{Equ. 6-1}$$

Cylindrical wall

$$T_j = \begin{cases} T_1 & for \quad j = 1 \\ T_{j-1} - \dfrac{\ln\left(x_j / x_{j-1}\right)}{k_j} q'_{ss} & for \quad j = 2, NHTE \\ T_{NHTE+1} & for \quad j = 1 \end{cases} \qquad \text{Equ. 6-2}$$

The steady-state heat ratings and the surface temperatures depend on the selected boundary conditions.

6.6.1 Steady-State Fluid Conditions Input for the Structure

GASFLOW-MPI allows input of some outer fluid condition on one structure side by specifying an outer fluid temperature, a heat-transfer coefficient, and a heat flux. This applies to the center of a sink but is also possible on either side of a wall. If the fluid conditions are specified on the positive

side (positive side implies the high number side of the heat structure or right side where increasing node numbers go from left to right), the steady-state heat flux for planar geometry is calculated from.

$$q''_{ss} = -\frac{\left(T_{fe} - T_1\right) + q_{ce} / h_e}{\frac{1}{h_e} + \sum_{j=2}^{NHTE+1} \frac{\delta x_j}{k_j}} \qquad \text{Equ. 6-3}$$

For the same conditions, the correlation for the linear heat rate in cylindrical coordinates is

$$q'_{ss} = -\frac{\left(T_{fe} - T_1\right) + q_{ce} / h_e}{\frac{1}{x_{NHTE+1} h_e} + \sum_{j=2}^{NHTE+1} \frac{\ln\left(x_j / x_{j-1}\right)}{k_j}} \qquad \text{Equ. 6-4}$$

In these correlations T_{fe}, q_{ce}, and h_e are the input values for the fluid temperature, the steady-state heat flux, and the convective heat-transfer coefficient on the east side. Note that the heat flux going into the wall on either side is assigned a positive sign in GASFLOW-MPI. This means that q_{ce} has to be entered with a negative sign because the heat flux on the east/positive/right side must have the sign of the positive *x*-direction also. Using the total heating rate, the structure surface temperature on the east/positive/right side is defined as

$$T_{NHTE+1} = \begin{cases} T_{fe} + \left(\frac{q''_{ss}}{h_e} + \frac{q_{ce}}{h_e}\right) & for\ planar\ wall \\ T_{fe} + \left(\frac{q'_{ss}}{x_{NHTE+1} h_e} + \frac{q_{ce}}{h_e}\right) & for\ cylindrical\ wall \end{cases} \qquad \text{Equ. 6-5}$$

When the fluid conditions are defined on the west/negative/left side of the structure, the heat flux and linear heat rating have the correlations for planar structures

$$q''_{ss} = -\frac{\left(T_{NHTE+1} - T_{fw}\right) - q_{cw} / h_w}{\frac{1}{h_w} + \sum_{j=2}^{NHTE+1} \frac{\delta x_j}{k_j}} \qquad \text{Equ. 6-6}$$

and for cylindrical structures

$$q'_{ss} = -\frac{\left(T_{NHTE+1} - T_{fw}\right) - q_{cw} / h_w}{\frac{1}{x_1 h_e} + \sum_{j=2}^{NHTE+1} \frac{\ln\left(x_j / x_{j-1}\right)}{k_j}} \qquad \text{Equ. 6-7}$$

The structure surface temperatures on the west/negative/left side are then calculated from Equ. 6-6 and Equ. 6-7 as

$$T_1 = \begin{cases} T_{fw} - \left(\dfrac{q''_{ss}}{h_w} - \dfrac{q_{cw}}{h_w} \right) & for\ planar\ wall \\ T_{fw} - \left(\dfrac{q'_{ss}}{x_1 h_w} - \dfrac{q_{cw}}{h_w} \right) & for\ cylindrical\ wall \end{cases} \qquad \text{Equ. 6-8}$$

The calculated surface temperature and the heating rates for these boundary conditions are applied to calculate the steady-state temperature profile using Equ. 6-1 or Equ. 6-2.

The convention for calculating steady-state profiles in slabs is slightly different. In this case, with **islablin** = 1 and **hslablin** > 0 (as explained in Section 6.5), the initial fluid temperature is specified on the negative side of the slab and is only considered as an initial condition. The surface temperature **tslabbc** is defined on the positive side and held constant as a boundary condition. The slab surface temperature on the negative side is then adjusted to be consistent with the steady-state heat flux.

6.6.2 Direct Input of Steady-State Heat Flux on the Structure

One can also specify only the heat flux and the surface temperature on one side of the structure. In this case, the steady-state surface temperature on the side of the heat flux is consistently defined for this heat flux using the given surface temperature on the opposite side. Equ. 6-1 and Equ. 6-2 are then applied again with the known surface temperatures to determine the steady state temperature profiles in the different structure elements. If the heat flux and one surface temperature are specified, then the surface temperature on the opposite side is calculated consistently. It can no longer be set by input if a steady-state temperature profile is assumed to exist prior to the analysis. The input heat fluxes q_{ce} and q_{cw} determine the heat flux for planar walls by

$$q''_{ss} = \begin{cases} q_{cw} & \text{when flux is specified on negative side} \\ -q_{ce} & \text{when flux is specified on positive side} \end{cases} \qquad \text{Equ. 6-9}$$

For cylindrical walls, the linear heat rate is determined from the input heat fluxes by

$$q'_{ss} = \begin{cases} x_1 q_{cw} & \text{when flux is specified on negative side} \\ -x_{NHTE+1} q_{ce} & \text{when flux is specified on positive side} \end{cases} \qquad \text{Equ. 6-10}$$

The input currently requires the user to specify the steady-state surface temperature on the opposite side as the heat flux. The missing surface temperature on the heat flux side is then calculated from Equ. 6-3 or Equ. 6-4, respectively, with the following correlation:

$$T_{NHTE+1} - T_1 = -\begin{cases} q''_{ss} \sum_{j=2}^{NHTE+1} \dfrac{\delta x_j}{k_j} & \text{for planar walls} \\ q'_{ss} \sum_{j=2}^{NHTE+1} \dfrac{\ln\left(x_j / x_{j-1}\right)}{k_j} & \text{for cylindrical walls} \end{cases} \qquad \text{Equ. 6-11}$$

Instead of inputting the surface temperature on the opposite side, it would be possible to specify the steady-state surface temperature on the same side as the heat flux. But this condition would require prior hand calculations to arrive at the desired surface temperature on the fluid side; therefore, it has not been put into the code.

6.7 Heat Fluxes into Slabs, Walls, and Sinks

The heat rate absorbed in the different structures is evaluated in subroutine outheat and printed for each slab, wall, and sink at the user-specified time interval **prtdt**. This subroutine also evaluates the heat fluxes which the structure takes out of the GASFLOW-MPI fluid cells. These heat fluxes are summed up for each structure type and are also included in the general plot output from PlotHist.nc. They can be applied in an overall energy balance if the system is closed. The transient temperature changes are also evaluated separately in the surface elements of both structure sides. This allows the calculation of transient heat fluxes associated with boundary conditions that only specify structure surface temperatures. The rate at which heat is absorbed in the structure is evaluated from the rate of temperature change at each temperature node, and the heat absorption rate equals the sum of the heat fluxes on both surfaces of the structure, which is an integral check that the transient evaluation of the structure temperatures does indeed conserve energy.

6.8 Balancing of Heat Structure Surfaces

The code automatically sums up all structure surfaces for slabs, walls, and sinks, split up into concrete, steel, and total surface. This information is printed out to the file gfout. Additionally, by setting the input parameter in rheat to

lprarea =1,

a data set area is defined at the start of the calculation under Unit 99 that contains the following: all affected m nodes

m	type	mat	Thickness (cm)	Area (cm^2)
4601	1	1	5.0000E+01	1.4000E+04
4661	1	1	5.0000E+01	1.4000E+04

the type (1=slab, 2=sink, 3=wall); the material number; the thickness applied for heat conduction; and the surface area of the respective structure. This information can be applied from other programs for a room wise balancing of structure surfaces as needed in the comparison of GASFLOW-MPI heat structure input to corresponding input from lumped parameter models.

There are instances when the GASFLOW-MPI calculated heat transfer surface areas that are computed from the internal geometry are not exactly in agreement with plant, experiment, or other code data. For example, in the current GKN analysis, we are attempting to match the heat transfer surface area of the GRS/RALOC analysis. In the following Table, we see the GRS data and the GASFLOW-MPI calculated areas:

Table 6-2: Comparison of Heat Transfer Areas used by GRS/RALOC and Computed by GASFLOW-MPI

Heat transfer surface	GRS/RALOC Area (m^2)	Calculated area (m^2)
Concrete area	31140	35119
Steel area (not reco)	17485	17485
Steel area (reco)	366	366
Steel shell	7919	8074

The third column of this Table, the GASFLOW area, is found from the screen output when **ihtflag** > 0. This information is shown in Figure 6-5. Note that GASFLOW's units are cm^2. The first entry in GASFLOW column of the Table is 35119 m^2 which is shown in Figure 6-5 as the total concrete surface. The next two entries, 17485 m^2 and 366 m^2, are shown as material 2 as sinks and walls, respectively. The last GASFLOW entry in Table 6-2, 8074 m^2, is found as material 4 in Figure 6-5.

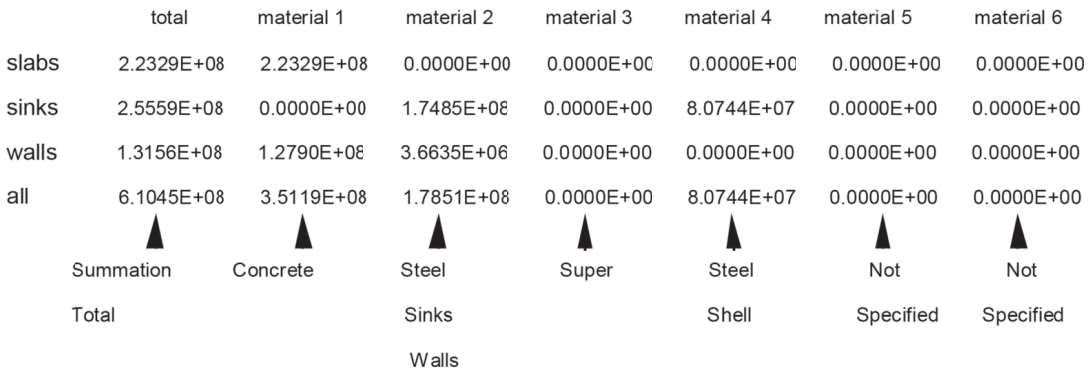


Figure 6-5 Balance of heat transfer surfaces before rebalancing **when** ihtflag > 0

We have added an option in GASFLOW-MPI that will allow the user to exactly balance the heat transfer areas by heat transfer surface type and material number. This option is best demonstrated by continuing the use of the example already started above. We can signal the heat transfer setup routines that we wish to rebalance the heat transfer surfaces by providing the following input in NAMELIST rheat:

iareabal Flag must be set > 0 to activate heat transfer rebalancing option (default = 0).
refaslab(*) Reference area for slabs by material type.
refasink(*) Reference area for sinks by material type.
refawall(*) Reference area for walls by material type.

To produce exactly the heat transfer surface area as the GRS/RALOC data, GASFLOW-MPI would therefore require the following input:

```
$rheat
...
 ihtflag      = 1,
 iareabal     = 1,
 refaslab     = 1.97991e+08,
 refasink     = 0.00000e+00, 1.74850e+08, 0.00000e+00, 0.79190e+08,
 refawall     = 1.13409e+08, 3.66350e+06,
…
$end
```

Note that we have partitioned the concrete area between slabs and walls. When this input is used, GASFLOW-MPI makes the required modification to all heat transfer surface areas and prints as standard output to gfout as shown in in Figure 6–6. It is now clear that an exact heat transfer surface area representation of the input data is obtained.

6.9 Modify Material Numbers of Slab Structures

It is mostly for 3D visualization that the user wants to omit certain structures to get a better view on results in inner regions hidden by obstacles. The user can open up a view into such regions by giving the hidden structures the same properties as the visible ones but associate them to a different material number. In most cases, the user distinguishes visible and invisible structures with different material numbers already when setting up the geometry model with obstacles and walls.

To allow the user to make certain structures invisible also during the run one can change the material number of certain slabs even during the run with the parameter **matdef** from $rheat.

Balance of heat transfer surfaces before rebalancing

	total	material 1	material 2	material 3	material 4	material 5	material 6
slabs	2.2329E+08	2.2329E+08	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
sinks	2.5559E+08	0.0000E+00	1.7485E+08	0.0000E+00	8.0744E+07	0.0000E+00	0.0000E+00
walls	1.3156E+08	1.2790E+08	3.6635E+06	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
all	6.1045E+08	3.5119E+08	1.7851E+08	0.0000E+00	8.0744E+07	0.0000E+00	0.0000E+00

Reference areas used to rebalance the heat transfer surfaces

	material 1	material 2	material 3	material 4	material 5	material 6
slabs	1.9799E+08	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
sinks	0.0000E+00	1.7485E+08	0.0000E+00	7.9190E+07	0.0000E+00	0.0000E+00
walls	1.1341E+08	3.6635E+06	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00

Balance of heat transfer surfaces after rebalancing

	total	material 1	material 2	material 3	material 4	material 5	material 6
slabs	1.9799E+08	1.9799E+08	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
sinks	2.5404E+08	0.0000E+00	1.7485E+08	0.0000E+00	7.9190E+07	0.0000E+00	0.0000E+00
walls	1.1707E+08	1.1341E+08	3.6635E+06	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
all	5.6910E+08	3.1140E+08	1.7851E+08	0.0000E+00	7.9190E+07	0.0000E+00	0.0000E+00

Figure 6-6 **GASFLOW output when** ihtflag **> 0 and** iareabal **> 0**

This parameter works for slabs only and is defined on the surfaces of the affected slabs as

matdef(1,*) Beginning i mesh of the slab surface.
matdef(2,*) Ending i mesh of the slab surface.
matdef(3,*) Beginning j mesh of the slab surface.
matdef(4,*) Ending j mesh of the slab surface.
matdef(5,*) Beginning k mesh of the slab surface.
matdef(6,*) Ending k mesh of the slab surface.
matdef(7,*) Block number (must be 1 for GASFLOW-MPI).
matdef(8,*) Material Number attributed to this slab.

Note that the slab surface is always a cell face, i.e like for a wall one pair of i, j, k should have the same index. If an obstacle is to be made fully invisible matdef statements have to be issued for each slab side of the obstacle that has contact with a fluid cell. The matdef statement can only refer to material numbers defined in $rheat. It cannot be used to remove slabs and obstacles.

Example:

The input matbdy defines the material on the computational boundary. One can define a different material number for the boundary slabs of a cylindrical mesh in the angle between 90 and 270 degrees:

$rheat
...
ihtflag = 1,
matbdy = 6,
matdef = 30, 30, 16, 46, 01, 26, 1, 7, ; opens 180 degree view
i1,i2 = 30, 30, ; defines the outer radial boundary in the mesh
j1,j2 = 16, 46, ; defines the angle with the new material 90 and 270 degrees from ygrid
k1,k2 = 1, 26 defines the height of the opened up boundary cylinder
matpanel = 6, ;only displays material 6 and omits 7.
…
$end

7 Physical Model Options

7.1 Body Forces

The momentum conservation equations solved by GASFLOW-MPI 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 activeted with the following input:

gx = 0.0,

gy = 0.0,

gz = -980.0,

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).

The user may also specify the computational domain to be initialized according to the hydrostatic pressure gradient, he must set at least one component of the gravitational vector (**gx**,**gy**,**gz**) and **ihystat** = 1 in XPUT namelist. When these options are selected, the pressure that the user has provided with the **gasdef** input statements will not be used. Instead, the analytical hydrostatic solution for the pressure field with be set in all computational cells and **gasdef** statements at the beginning of the simulation according to

$$0 = -\nabla p + \mathbf{g}\rho \qquad \text{Equ. 7-1}$$

The solution of this equation requires a reference point to which the integration may be associated, an initial pressure at an exact location. We provide this information in the XPUT namelist variables reference pressure (pamb0) and locations of reference point (xamb0, yamb0, zamb0). The default values of these variables are as follows:

pamb0 = 1.013253e+06,
xamb0 = 0.0,
yamb0 = 0.0,
zamb0 = 0.0,

If an initial hydrostatic pressure gradient is required, it is important the user investigates the ramifications of using these initial variables. The reason is that the initial hydrostatic pressure is determined by an analytical evaluation of the above equation and not the numerical approximation to the entire momentum equations (Navier-Stokes equations) that GASFLOW-MPI solves, and there is likely to be a difference between the analytical and numerical methods.

7.2 Diffusion of Mass, Energy, and Momentum

In GASFLOW-MPI, 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 without gradient diffusion. The user must explicitly specify them using the following input variables in NAMELIST group xput:

idiffme 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 to calculate the molecular diffusion coefficient. The user can specify whether the input dynamic viscosity, μ (in units of poise or g/cm·s), or the input kinematic viscosity, $\nu = \mu/\rho$ (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 for viscous stress calculation:
1 means **nu** is used (default);
2 means **cmug** is used.
nu Kinematic viscosity, ν (cm^2/s). Default = 0.15.
cmug Dynamic viscosity, μ (g/cm·s). Default = 1.8 x 10^{-4}.

For the diffusion of mass species and energy, the code uses the input variables **schmidt** and **prandtl**, respectively, if the user has selected the **itopt** = 0 option, to determine the appropriate diffusion coefficient. See Section 4.2 for information concerning temperature-dependent transport properties.

7.3 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 with the turbulent fluctuation scale, but small compared with the problem transient time scale. Such an averaging procedure is called Reynolds averaging.

Therefore, the equations solved in GASFLOW-MPI 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 GASFLOW-MPI, all turbulent fluxes are modeled like a molecular diffusion process. Turbulent momentum fluxes (stresses) are modeled as

$$\tau_{t,ij} = \mu_t \, \partial u_i / x_j, \qquad \text{Equ. 7-2}$$

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 GASFLOW, we further assume that the turbulent diffusion coefficients for mass and energy are both the same as that for momentum and are 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 GASFLOW, two turbulence models are available. The choice can be specified via the input character variable **tmodel** in NAMELIST group xput:

tmodel Symbol designating the type of turbulence model to be used:
'none', no turbulence model, i.e., only molecular diffusion;
'alg', algebraic turbulence model;
'ke', κ-ε turbulence model;
'ko', κ-ω turbulence model;
'sstko', SST κ-ω 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 **idiffme** has been set to 1.

Wall Functions. When rigid no-slip conditions are specified and turbulence is activated, the user may wish to use wall functions rather than resolving the boundary layers. The options are as follows:

iwallfunc = 0 ; default value
; no wall functions are active
iwallfunc = 1 ; no-slip conditions must to active
; assumes smooth walls
iwallfunc = 2 ; no-slip conditions must to active
; assumes rough walls and **krough** must be specified

See Section 5.2.1 and Section 7.5 for further context concerning the wall functions.

The initial turbulent conditions in the ambient can be defined by using two of the following variables,

tkeamb0 Initial turbulent kinetic energy in the ambient. Default = 10 (cm^2/s^2).
epsamb0 Initial turbulent dissipation rate in the ambient. Default = 10000 (cm^2/s^3) .
sclamb0 Initial turbulent scale in the ambient.

7.3.1 Algebraic Model

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

$$\mu_t = c_\mu \rho \sqrt{\kappa} l, \qquad \text{Equ. 7-3}$$

where κ 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.,

$$\kappa = f \cdot \frac{1}{2}(u^2 + v^2 + w^2) \qquad \text{Equ. 7-4}$$

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:

cmu Constant c_μ for the algebraic turbulence model. Default = 0.09.
clength Length scale, l, for the algebraic turbulence model (cm). Default = 30.48.
fractke Fraction of the mean kinetic energy, f, used in the algebraic turbulence model to determine the turbulent kinetic energy. Default = 0.1.

7.3.2 The *κ-ε* Model

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

$$\mu_t = c_\mu \rho \kappa^2 \varepsilon \qquad \text{Equ. 7-5}$$

where κ 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:

$$\frac{\partial(\rho\kappa)}{\partial t}+\nabla\cdot(\rho\kappa\mathbf{u})=\nabla\cdot\left(\frac{\mu_t}{\sigma_k}\nabla\kappa\right)+\tau:\nabla\mathbf{u}+\mu\alpha\mathbf{g}\cdot\nabla T-\rho\varepsilon+\kappa S_k \qquad \text{Equ. 7-6}$$

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:

$$\frac{\partial(\rho\varepsilon)}{\partial t}+\nabla\cdot(\rho\varepsilon\mathbf{u})=\nabla\cdot\left(\frac{\mu_t}{\sigma_\varepsilon}\nabla\varepsilon\right)+c_1\frac{\varepsilon}{\kappa}\tau:\nabla\mathbf{u}+c_1\frac{\varepsilon}{\kappa}\mu\alpha\mathbf{g}\cdot\nabla T-c_2\rho\frac{\varepsilon^2}{\kappa}+\varepsilon S_\varepsilon \qquad \text{Equ. 7-7}$$

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

cmuke c_μ used in the κ-ε turbulence model. Default = 0.09.
c1ke c_1 used in the κ-ε turbulence model. Default = 1.44.
c2ke c_2 used in the κ-ε turbulence model. Default = 1.92.
sigmak σ_κ used in the κ-ε turbulence model. Default = 1.
sigmae σ_ε used in the κ-ε turbulence model. Default = 1.3.

Because the κ-ε model solves time-dependent transport equations for both κ and ε, the user must define initial conditions for these quantities. The **turbdef** array described below is used for defining initial and boundary conditions for the κ-ε model and is in the NAMELIST group xput.

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

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 κ and ε transport equations:

turbdef(1:12,1) = 1, 50, 1, 32, 1, 45, 1, 2, 1, 0, 0.0, 0.0,
turbdef(1:12,2) = 0, 1, 2, 5, 4, 6, 1, 1, 2, 0, 0.0, 1.e9,
tkeval = 1.0e3, 0.0,
epsval = 0.0, 800.0,

The first **turbdef** defines an initial condition for κ and ε. The values of both κ 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 κ and ε specified are 1000 cm^2/s^2 and 800 cm^2/s^3, respectively.

7.3.3 SST κ-ω model

warning: The SST κ-ω model is under development as time permits. It is currently not available in GASFLOW-MPI 1.0. The following variables in NAMELIST group xput are used to set the SST κ-ω model parameters (see Section 2.7.3 in Theory Manual for more details):

sigmak3 σ_{k3} used in the SST κ-ω turbulence model.
sigmao3 $\sigma_{\omega 3}$ used in the SST κ-ω turbulence model.
alphao3 α_3 used in the SST κ-ω turbulence model.
betao3 β_3 used in the SST κ-ω turbulence model.

The coefficients of standard κ-ω model are multiplied by blending function F_1 and the coeeficients of the transformed κ-ε equations are multiplied by a function 1-F_1, the closure coefficients for the SST κ-ω turbulence model are calculated by,

$$\begin{aligned}
\sigma_{k3} &= F_1\sigma_{k1} + (1-F_1)\sigma_{k2}, \\
\sigma_{\omega 3} &= F_1\sigma_{\omega 1} + (1-F_1)\sigma_{\omega 2}, \\
\alpha_3 &= F_1\alpha_1 + (1-F_1)\alpha_2, \\
\beta_3 &= F_1\beta_1 + (1-F_1)\beta_2
\end{aligned} \qquad \text{Equ. 7-8}$$

sigmas σ_{k1} used in the κ-ω turbulence model. Default = 0.5.
sigmao $\sigma_{\omega 1}$ used in the κ-ω turbulence model. Default = 0.5.
alphao α_1 used in the κ-ω turbulence model. Default = 5/9.
betao β_1 used in the κ-ω turbulence model. Default = 0.075.
sigmak σ_{k2} used in the transformed κ-ε turbulence model. Default = 1.
sigmaeo $\sigma_{\omega 2}$ used in the transformed κ-ε turbulence model. Default = 0.856.
c1ke α_2 used in the transformed κ-ε turbulence model. Default = 0.44.
c2ke β_2 used in the transformed κ-ε turbulence model. Default = 0.0828.

The **turbdef** array used for defining initial and boundary conditions for the SST κ-ω model is identical as the **turbdef** array of κ-ε model. The user must define two valid pointers for the arrays **tkeval** in turdef(8,*), **epsval** in turdef(9,*) and **sclval** in turdef(10,*). Two of these three arrays must be defined. For the SST κ-ω model, the value of omega, **omgval**, can be calculated by

omgval = epsval/(cmuke*tkeval), when tkeval and epsval are defined;
omgval = (cmuke*$sclval^2$)/epsval, when epsval and sclval are defined;
omgval = $tkeval^{0.5}$/sclval, when tkeval and sclval are defined.

7.4 Combustion

7.4.1 One-Step Global Chemical Kinetics Model

In order to be compatible with the previous versions, we leave this section unchanged.

GASFLOW-MPI models the combustion of hydrogen in air with the following single, global reaction:

$$a(H_2)+b(O_2)+c(N_2)+d(H_2O)\overset{\dot{\omega}}{\leftrightarrow}e(H_2O)+f(H_2)+g(O_2)+c(N_2) \qquad \text{Equ. 7-9}$$

where the coefficients *a*, *b*, *c*, *d*, *e*, *f*, and *g* are generally referred to as the stoichiometric coefficients. The reaction rate, $\dot{\omega}$, is determined from a modified Arrhenius law, where

$$-\frac{1}{a}\frac{dC_{h_2}}{dt}=-\frac{1}{b}\frac{dC_{o_2}}{dt}=+\frac{1}{e}\frac{dC_{h_2o}}{dt}=\dot{\omega}\ , \qquad \text{Equ. 7-10}$$

and a = 2, b = 1, and e = 2.

The progress variable, $\dot{\omega}$, is

$$\dot{\omega}=k_f\left[H_2\right]^2\left[O^2\right]-k_b\left[H_2O\right]^2\ , \qquad \text{Equ. 7-11}$$

The forward and backward reactions rates are, respectively,

$$k_f(T)=a_fT^f\exp\left(-\frac{E_f}{RT}\right), \qquad \text{Equ. 7-12}$$

and

$$k_b(T)=a_bT^b\exp\left(-\frac{E_b}{RT}\right). \qquad \text{Equ. 7-13}$$

Computationally, during each time cycle, the change in molar density of the fuel, δc_{h_2} is calculated implicitly first when the fuel-oxidizer mixture is fuel lean. Then the molar densities at the advanced time level *n*+1 $c_{h_2}^{n+1}$, $c_{o_2}^{n+1}$, and $c_{h_2o}^{n+1}$ are determined respectively as

$$c_{h_2}^{n+1}=c_{h_2}^{n}+\delta c_{h_2}, \qquad \text{Equ. 7-14}$$

$$c_{o_2}^{n+1}=c_{o_2}^{n}+\frac{b}{a}\delta c_{h_2}, \qquad \text{Equ. 7-15}$$

$$c_{h_2o}^{n+1}=c_{h_2o}^{n}-\frac{e}{a}\delta c_{h_2}. \qquad \text{Equ. 7-16}$$

When the mixture is fuel rich, then the molar density change for the oxidizer δc_{o_2} is calculated first, and the molar densities are determined as

$$c_{o_2}^{n+1} = c_{o_2}^{n} + \delta c_{o_2}\,, \qquad \text{Equ. 7-17}$$

$$c_{h_2}^{n+1} = c_{h_2}^{n} + \frac{a}{b}\delta c_{o_2}\,, \qquad \text{Equ. 7-18}$$

$$c_{h_2o}^{n+1} = c_{h_2o}^{n} - \frac{e}{b}\delta c_{o_2}\,. \qquad \text{Equ. 7-19}$$

This will ensure that nonphysical 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. 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 = +1, Hydrogen burn model on (forward reaction only)
= 0, Hydrogen burn model off (default)
= -1, Hydrogen burn model on (both forward and reverse reactions)

Table 7-1 Coefficients and parameters for the forward reaction only (iburn = +1) global (1-step) chemical kinetics model.

Reaction Rate	A	E/R	T^n
forward, k_f	5.0000×10^{12}	7.8×10^{4}	n=0
backward, k_b	0.0	0.0	n=0

Table 7-2 Coefficients and parameters for the forward reaction and reverse reaction (iburn = -1) global (1-step) chemical kinetics model.

Reaction Rate	A	E/R	T^n
forward, k_f	5.0000×10^{17}	9.37514×10^{3}	n=0
backward, k_b	6.2132×10^{21}	6.96287×10^{4}	n=-1

To activate the hydrogen recombiner model, the **ircomb** input variable must be 1 and **rcombdef** input described in Chapter 7.7.2 must be provided.

7.4.2 Ignitior Model for Global Chemical Kinetics Model

Note: this ignitor model should only be used together with the combustion model in Chapter 7.4.1. The ignitor model is simple, but effective. At user-specified locations GASFLOW-MPI first checks the

gas composition to determine if the mixture is combustible, based on the Shapiro Diagram for hydrogen, oxygen, and steam mixtures. The locations where the ignitor model is applied are defined with the input array **ignitdef,** which is in the xput NAMELIST group. The parameter **ignitdef** is described below:

ignitdef(1,*) i cell index.
ignitdef(2,*) j cell index.
ignitdef(3,*) k cell index.
ignitdef(4,*) Block number (must be 1 for GASFLOW-MPI).
ignitdef(5,*) Ignitor type:
= 0, continuous ignitor;
> 0, spark ignitor.

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

When **ignitdef(5,*)** is greater than zero, then the ignitor types are defined in the input array **spxidef** and the input variable **spxigdt**, which are also in the xput NAMELIST group. Input array **spxidef** and input variable **spxigdt** are defined below:

spxigdef(1,igtyp) Time when ignitor is first turned on.
spxigdef(2,igtyp) Time when ignitor is turned off.
spxigdef(3,igtyp) Time interval between sparks.
spxigdt Time of the spark duration. Default = 0.001 s.

In the **spxigdef** definitions, **igtyp** is the ignitor type number defined by **ignitdef(5,*)**. The maximum number of ignitor definitions (i.e., **ignitdef(*,n)**) and the maximum number of ignitor types are both currently set with a parameter statement to 300.

For example, let's consider three continuously operating ignitors in block 1 at the following locations:

1st ignitor: (i = 12, j = 41, k = 17, number of block = 1)
2nd ignitor: (i = 21, j = 14, k = 06, number of block = 1)
3rd ignitor: (i = 31, j = 22, k = 25, number of block = 1)

and three sparking ignitors in block 2 at locations:

4th ignitor: (i = 21, j = 4, k = 71, number of block = 2)
5th ignitor: (i = 12, j = 14, k = 60, number of block = 2)
6th ignitor: (i = 13, j = 22, k = 25, number of block = 2)

where ignitors 4 and 6 start operating at time = 0.0, with a 10 s period, operating until 1000.0 s, while ignitor 5 starts operating at 2000.0 s with a 60.0 s period, operating until 8000.0 s.

The input stream would be input as follows:

```
$xput
…
 ignitdef     = 12, 41, 17, 1, 0,
                21, 14, 6, 1, 0,
                31, 22, 25, 1, 0,
                21, 14, 71, 1, 1,
                12, 14, 60, 1, 2,
                13, 22, 25, 1, 1,
 spxigdef     = 0.0, 1000.0, 10.0,
                2000.0, 8000.0, 60.0,
 spxigdt      = 0.01,
…
$end
```

NOTE: In this case, the spark duration variable **spxigdt** is specified a different value than the default value of 0.001 s. If the combustion model is activated, **iburn** ≠ 0, then the possibility of an spontaneous or auto ignition may occur provided the local temperature is greater than the auto ignition temperature and certain criteria concerning the gas concentrations are met. If the user wishes to allow auto ignition to occur, they must activate the following model by setting **ignitaut** > 0 in the namelist XPUT. The auto ignition temperature, **autotemp**, can also be set to a value other than it's default value of **autotemp** = 800.0 (Kelvin).

We have incorporated the Kumar flammability limits for hydrogen-oxygen-diluent mixtures. The gas concentration criteria proposed for this model is shown in the Table 7-3.

Table 7-3 Kumar Flammability Limits of Hydrogen-Oxygen-Diluent Mixtures.

Gas Concentration (molar fractions)	Gas Concentration Criteria (molar percentages)	Hydrogen or Oxygen Concentration (molar percentages)
Hydrogen "Lean" $[H_2]<2[O_2]$	$[O_2] + [\text{Diluent}] \leq 96\%$	$[H_2] > 4\%$
Hydrogen "Rich" $[H_2]>2[O_2]$	$[H_2] + [\text{Diluent}] \leq 95\%$	$[O_2] > 5\%$

The burning rate of hydrogen is generally determined from the Arrhenius correlation (See Chapter 7.4) if the combustion model is activated (iburn=1).

ignitaut = 0

With ignitaut =0, GASFLOW simulates hydrogen combustion in the limit of the Kumar criterion using the Arrhenius correlation with the physical temperature of each computational cell. The reference temperature T_{ref} applied in the Arrhenius correlation is set to the physical temperature in each

computational cell. An exception are active ignitor cells where the reference temperature applied in the Arrhenius correlation is increased to 2000 K.

ignitaut =1

Hydrogen burn is simulated only for gas temperatures above auto ignition with the parameter ignitaut =1 in cells that fulfill the Kumar criterion. The temperature T_{ref} (m) for cell m in the Arrhenius correlation is then set different from the cell temperature. In a loop over all cells the code first determines if the cell temperature is below the input value autotemp for the auto ignition temperature. In this case the temperature T_{ref}(m) for cell m in the Arrhenius correlation is set to 1 K only. The cell temperature T_{cell} (m) defines T_{ref} (m) only when it is above autotemp (see Figure 7-1).

ignitaut =2

An additional option ignitaut = 2 has been included (this option is not shown in Figure 7-1). It prevents auto ignition to occur in PAR cells defined by rcombdef. The T_{ref} values for PAR cells are always set to 1 when the user sets the value ignitaut =2.

Cells with ignitors

In active ignitor cells defined by ignitdef , spxigdef, and spxigdt the code resets the value of T_{ref} to 2000 K. A value of Tref (m) >1 then denotes all cells which fulfill necessary conditions for ignition. But hydrogen burn is only simulated in these cells if their hydrogen and oxygen concentrations are within the lean and rich flammability limits as defined by the Kumar criterion in Table 7-3.

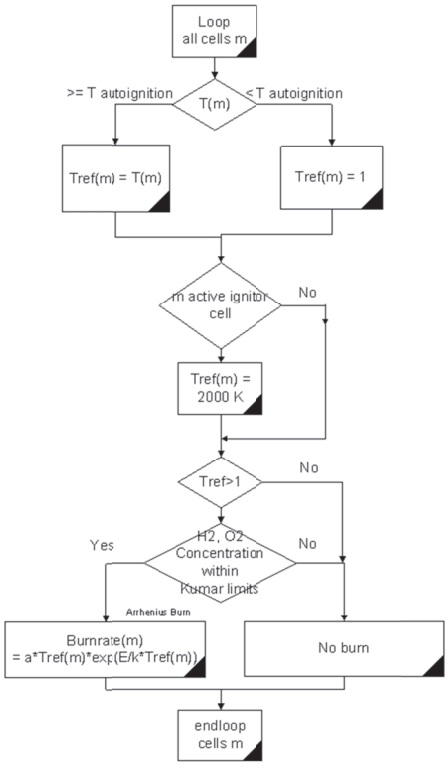


Figure 7-1 Scheme for burning hydrogen with Arrhenius Correlation for ignitaut = 1

7.4.3 Combustion Models Based on Reaction Progress Variable

(Warning: These combustion models are currently experimental, and should be used with caution. Please contact the developers for more information, in case you would like to use these models. More details of the models will be given in the next release.)

A general "combustion progress variable" transport equation has been solved to model the turbulent combustion in GASFLOW-MPI. Please refer to Chapter 2.8.2 of GASFLOW-MPI Theory Manual.

In order to turn on these hydrogen combustion models, the user has to first specify **iburn** = 4 in NAMELIST group xput. When **iburn** = 4, xi_ignitdef in Chapter 7.4.4 must be used.

Since the key to this modeling approach is the source term, the user then has to specify **isourcexi** in NAMELIST group xput to tell the code which model will be used to obtain the source term.

isourcexi = 0, source term off;
= 1, Arrhenius rate model;
= 2, model based on progress variable gradient (default);
= 3, under development;
= 4, under development;
= 5, Eddy dissipation model.

Please note currently the turbulent combustion models (**isourcexi** = 2, 3, 4 and 5) are only coupled with the κ-ε turbulence model. Therefore, κ-ε turbulence model must be turned on in order to obtain the turbulent kinetic energy, κ, and turbulent dissipation rate, ε, which are used to calculate the source term in the raction progress variable transport equation. The initial conditions of the turbulent properties could strongly influence the combustion process and the time step of the calculation. Therefore, the user should estimate the initial conditions of experiments or simulations and give reasonable initial values to the turbulence model. There are two ways to set up the initial turbulent conditions.

The first way is to use two of the variables: **tkeamb0**, **epsamb0** and **sclamb0**. For example, to set up the initial turbulent kinetic energy and dissipation rate in the whole computational domain, we could define

tkeamb0 = 1.0e2,
epsamb0 = 8.0e3,

Another more flexible way is to use **turbdef** which allows the user to define various conditions in different sub-domains. For example,

turbdef(1:12,1) = 2, 'im1', 2, 'jm1', 2, 'km1', 1, 1, 1, 0, 0.0, 0.0,
turbdef(1:12,2) = 2, 3, 2, 3, 2, 3, 1, 2, 2, 0, 0.0, 0.0,
tkeval = 1.0e2, 2.5e3,
epsval = 8.0e3, 2.0e4,

When **isourcexi** equals 2, the users has to define **iturbflame** in NAMELIST group xput to select a correlation for calculation of turbulent flame speed.

iturbflame = 0, use laminar flame speed;
= 1, use the correlation $S_T = S_L\left(1 + \frac{\sqrt{\kappa}}{S_L}\right)$;
= 2, use Kawanabe correlation;
= 3, use Peters correlation;
= 4, use Zimont correlation (default);
= 5, use Zimont-Mesheriakov correlation;
= 6, use Schmidt correlation.

To obtain the reference initial pressure, density and thermal diffusivity to calculate the Damkoehler number for the models above (**isourcexi = 2**), the user has to define a reference point to represent the unburnt mixture using the variables below.

iref_unburnt i cell index for reference point of unburnt mixture;
jref_unburnt j cell index for reference point of unburnt mixture;
kref_unburnt k cell index for reference point of unburnt mixture;
iblkref_unburnt Block number for reference point of unburnt mixture.

7.4.4 Ignitor model for Reaction Progress Variable

(Warning: These ignitor models are currently experimental, and should be used with caution. Please contact the developers for more information, in case you would like to use these models. More details of the models will be given in the next release.)

Note: this ignitor model should only be used together with the combustion models in Chapter 7.4.3. At user-specified locations GASFLOW-MPI first checks the gas composition to determine if the mixture is combustible, based on the Shapiro Diagram for hydrogen, oxygen, and steam mixtures. The cells where the xi_ignitor model is applied are defined with the input array **xi_ignitdef,** which is in the xput NAMELIST group. The parameter, **xi_ignitdef**, is described below:

xi_ignitdef(1,*) beginning i mesh index (cell face number).
xi_ignitdef(2,*) ending i mesh index (cell face number).
xi_ignitdef(3,*) beginning j mesh index (cell face number).
xi_ignitdef(4,*) ending j mesh index (cell face number).
xi_ignitdef(5,*) beginning k mesh index (cell face number).
xi_ignitdef(6,*) ending k mesh index (cell face number).
xi_ignitdef(7,*) block number (must be 1 for GASFLOW-MPI)
xi_ignitdef(8,*) time when ignitors are first turned on (s).
xi_ignitdef(9,*) time of the ignition duration (s).

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

The same Kumar flammability limits for hydrogen-oxygen-diluent mixtures, which is shown in the Table 7-3, is also used for **xi_ignitdef** definition.

For example, let's consider three continuously operating ignitors in block 1 at the following locations:

1st ignitor: (i = 2~4, j = 2~3, k = 3~4, number of block = 1, ignition starts at 0 s, ignition duration: 5 ms)

2nd ignitor: i = 6~7, j = 5~6, k = 6~7, number of block = 1, ignition starts at 1 ms, ignition duration: 2 ms)

3rd ignitor: (i = 9~10, j = 8~9, k = 9~10, number of block = 1, ignition starts at 2 ms, ignition duration: 4 ms)

The reference cell for the unburnt mixture is located at (2, 2, 2).

The input stream would be input as follows:

```
$xput
…
 iburn              = 4,
 isourcexi          = 2,
 iturbflame         =  4,
 iturb              =  'ke',
 idiffme            =  1,
 idiffmom           =  1,
 iref_unburnt       =  2,
 jref_unburnt       =  2,
 kref_unburnt       =  2,
 iblkref_unburnt    =  1,
 tkeamb0            =  100,
 epsamb0            =  10000,
 xi_ignitdef        =  2, 4, 2, 3, 3, 4, 1, 0, 0.005,
                       6, 7, 5, 6, 6, 7, 1, 0.01, 0.002,
                       9, 10, 8, 9, 9, 10, 0.02, 0, 0.004,
…
$end
```

7.5 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** > 0) 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 with 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:

ihtflag Option flag to turn on heat-transfer and steam-condensation calculations:
1 means ON; 0 means OFF (default).

When **ihtflag** > 0 other input values control heat transfer options through the **rheat** NAMELIST input stream for activating specific heat transfer relationships.

ihtcoef	Descriptions
0	constant heat transfer coefficient. Set equal to **cons1** **cons1=** 1.0 (default value in ergs/K·s)
1	laminar heat transfer coefficient assumes smooth walls.
2	von Karman analogy heat transfer coefficient. Assumes smooth walls.
3	Extended von Karman analogy heat transfer coefficient. Assumes smooth walls.
4	**Default model**. Reynolds analogy with Colburn correction for the heat transfer coefficient. Assumes smooth walls.
5	Reynolds analogy with Colburn correction for the heat transfer coefficient. Assumes rough walls and **krough** must be specified **Krough =** 0.1 (default surface roughness depth (cm))
6	Constant heat flux. Set equal to **cons1**. **cons1=** 1.0 (default value in ergs/cm^2·s)
7	Similar velocity-temperature profiles. Temperatures maybe directly scaled using **cons1** **cons1=** 1.0 (default value in ergs/cm^2·s)
8	Heat transfer coefficient = 0, same as **ihtcoef** = 0 when **cons1 = 0**

When the default model is used (ihtcoef = 4), the multiplication factor **cons1** is preset.

cons1 Multiplication factor for the heat-transfer coefficient (applies to convection and condensation/evaporation). Default = 3.6.

There are other heat and mass transfer parameters that the user can control:

crelax Relaxation factor for depleting liquid H_2O. Default = 0.01 (s^{-1})

icond Flag for condensation on structures: 1 means ON; 0 means OFF. Default = 1

ienh Film-model enhancement from Bird Stewart Lightfoot:
1 means ON; 0 means OFF; so far only for film condensation. Default = 1.

ifcca Flag for Chilton-Colburn analogy: 1 means ON; 0 means OFF. Default = 1.

When no-slip velocity boundary conditions are applied and turbulence is activated, we repeat again certain input options from Section 5.2.1 and Section 7.3 and these are the wall function options:

iwallfunc	Descriptions
0	default value. No wall functions are active.
1	no-slip conditions must to active assumes smooth walls.
2	no-slip conditions must to active assumes rough walls and **krough** must be specified.

Within the general heat transfer model, when **ihtflag** > 0, the user may activate a radiation heat transfer model. The radiation model is activated by the **rheat** NAMELIST variable **irad** > 0. There are a number of options:

irad	Descriptions
0	Radiation model is not activated (default value).
1	Radiation transport equation model is implicitly solved.

The recommended option for the solution of the radiation heat transfer transport equation is 1.

There are several other **rheat** NAMELIST variables connected with the radiation heat transfer model that the user has some control:

emismat(*) Material emissivity. Each material in the heat transfter material data base has a default emismat(*) = 1.

clamda radiation mean free path (cm). If clamda > 0, then the user is specifying a constant radiation mean free path; otherwise, the mean free path is computed using the method of Lechner, “Spectral and Total Emissivity of Water Vapor and Carbon Dioxide”, Combustion and Flame, Vol. 19, pp. 33-48, 1972.

itmaxr maximum number of iterations allowed for the solution of the radiation heat transfer equation (default = 500).

epsrad convergence criteria for the solution of the radiation heat transfer equation (default = 10^{-6}).

The user should also refer to Section 1, on defining heat structures, for complete specification of a heat-transfer problem.

7.6 Aerosol Model

(Note: aerosol model has not been included in GASFLOW-MPI 1.0. It will be developed in GASFLOW-MPI as time allows.)

To set up the GASFLOW-MPI Lagrangian discrete particle dynamics models, an understanding of the input parameters and their function in the model's is essential. In Appendix F the particle input parameters are listed and defined with default values. These are described in more detail in the following sections focusing on the four major functions of the particle model. These functions are particle initialization, transport, deposition, and entrainment. In addition, the particle cloud model is described. Each section presents and describes the input parameters, presents associated variables and arrays and presents in outline form the setup procedure for the model function. Next, descriptions of the parameter statements, restart procedure, and graphic and tabular output are given. A sample problem setup and output are discussed.

Generally, it is impractical, because of computer storage limitations and costs, to track each particle in an aerosol cloud. The total mass of the aerosol cloud must be represented by a finite number of discrete particles. In the following description of input parameters, the terms real and simulation are often used. The term real is used in reference to the particle number, mass, etc., that are being modeled. The term simulation refers to the discrete particles that are being used in the numerical model to simulate the real particles being modeled. Each simulation particle may represent thousands of real particles of the same size and density.

7.6.1 Description of Particle Initialization Input Parameters

The particulate materials are characterized by class and size. The material class corresponds to a material density. Each material class may have multiple particle sizes. The input parameter **pmass(lc,ls)** is the total mass of each particle class and size that is to be input over the total problem time. The class material density is set in **prhoin(lc)** and the particle diameters are specified for each class and size by **pdiamin(lc,ls)**. From this input data the number of *real* particles, npreal(lc,ls), for each class and size and the total number of *real* particles, npsum, are computed. Next, the volumes, volp(lc), within the computational mesh into which each particle class is to be set are determined from the input parameters specifying the maximum and minimum *x*-, *y*-, and *z*-coordinates of the particle volumes; these are **xpe(lc)**, **xpw(lc)**, **ypn(lc)**, **yps(lc)**, **zpt(lc)**, and **zpb(lc)**.

When the particle model is activated, the GASFLOW random number generator is also activated with a seed based on the day and time of execution. The input parameter **init_random** allows the initialization of the seed. The value of the seed is printed on the "**gfout**" file. Generally, the total number of real particles, determined from **pmass(lc,ls)**, **prhoin(lc),** and **pdiamin(lc,ls)**, would be prohibitively large to compute efficiently. The user selects the total number of simulation particles to be input and transported by the code. This input parameter is **npinpt** (number particles input total). From the total number of simulation particles input, **npinpt**, the number of real particles for each class and size, npreal(lc,ls), and the total number of real particles, npsum, the code computes the number of simulation particles for each class and size npsim(lc,ls). Using the information available, the real mass represented by each simulation particle, rmpp(lc,ls), is computed.

The input parameter **twpinp(lc)** designates the initial time when *simulation* particles of each class are to be input into the system. The total number of particles for each class may all be set in the mesh initially or may be injected as a function of time. This choice is governed by the input parameters **tinjt(lc)** (total injection time) and **pinpdt(lc)** (particle input Δt). To inject the particles as a function of time, set the initial time when particles are to be injected, **twpinp(lc)**, the total injection time, **tinjt(lc)**, and the time interval between particle injections, **pinpdt(lc).** From this input data the code computes the number of injection times and the number of *simulation* particles for each class and size, npinj(lc,ls), to be injected at the appropriate intervals. To inject a single volume of particles, input the time of injection, **twpinp(lc)**, and values for **tinjt(lc)** and **pinpdt(lc)** such that the ratio **tinjt(lc)**/**pinpdt(lc)** is less than 1.0. The code will set the number of injection times, ninjts(lc), to 1.0. The value of **pinpdt(lc)** also must be greater than the problem time.

When restarting a problem from a previous particle run, set the parameter **twpinp(lc)** to a value greater than the final problem time to inhibit the injection of additional particles. If injection of particles is wanted at the time determined in the previous run, do not set a value for **twpinp(lc)** in the restart input parameter file. The correct value will be read from the restart tape file.

The variables **itpcl**, **itpsz**, and **msp** are included in the xput NAMELIST group and are used to set array sizes in the particle model. Therefore, whenever **solatype** is greater than zero, these three inputs must be supplied in the xput NAMELIST group. A calculation can be started with **solatype** equal to zero and then restarted with **solatype** greater than zero. The restart must include the three inputs **itpcl**, **itpsz**, and **msp**. Once these three inputs have been used, all subsequent restarts must include the same values for these input variables. Initial particle velocities could be specified to each particle class with the part NAMELIST variable **partvel(:,lc)**.

The following aerosol input parameters are in the xput NAMELIST group:

itpcl	Total number of particle classes.
itpsz(lc)	Number of initial particle sizes in each particle class.
msp	Total number of simulation particles allowed in this calculation.
solatype	= 0.0 implies a hydrodynamic calculation without the particle model. = 1.0 implies a hydrodynamic calculation with the particle model. = 2.0 implies a particle-model-only calculation with gas velocities fixed in time.

The following input parameters are in the parts NAMELIST group:

init_random	Particle random number generator option: **init_random**=0 (default) implies the initialization of the seed. **init_random**=n implies n is the seed which allows the user to initialize the random number generator with the same seed regardless of the time of execution. This option is helpful when reproducing the same results for each separate calculation is necessary.
ipblkin(lc)	Mesh block in which particle class, index lc, is initially located (must be 1 for GASFLOW-MPI).
ipclin(lc)	Particle class number input.
npinpt	Total number of *simulation* particles to be input.

partvel(:,lc) Flag for specifying initial particle velocity (cm/s).
= 0; no initial particle velocity specified for particle class **lc** (default).
= 1; particle velocity specified for particle class **lc** and signaling GASFLOW to assign particle class initial velocities **partvel(2:4,lc).**
partvel(2,lc) specifies particle class **lc** u-component initial velocity (default = 0.0).
partvel(3,lc) particle class **lc** v-component initial velocity (default = 0.0).
partvel(4,lc) particle class **lc** w-component initial velocity (default = 0.0).

pinpdt(lc) Time interval between particle inputs for each particle class (s).

pdiamin(lc,ls) Particle diameter input for each class and size (cm).

pmass(lc,ls) Total *real* particle mass of each class and size (g).

prhoin(lc) Material density input for each particle class (g/cm^3).

tinjt(lc) Total injection time for each particle class (s).

twpinp(lc) Time when to input particles of each class (s).

xpe(lc) Maximum *x*-coordinate of initial particle block for each particle class (cm).

xpw(lc) Minimum *x*-coordinate of initial particle block for each particle class (cm).

ypn(lc) Maximum *y*-coordinate of initial particle block for each particle class (cm).

yps(lc) Minimum *y*-coordinate of initial particle block for each particle class (cm).

zpb(lc) Minimum *z*-coordinate of initial particle block for each particle class (cm).

zpt(lc) Maximum *z*-coordinate of initial particle block for each particle class (cm).

The following inputs are required:

1. Particle material density, **prhoin(lc)**, particle size, **pdiamin(lc,ls)**, and total mass of particulate material, **pmass(lc,ls)**.
2. Particle input volume and location:

   xpw(lc) **xpe(lc)**
   yps(lc) **ypn(lc)**
   zpb(lc) **zpt(lc)**

3. Particle injection time and rate:

   twpinp(lc) Initial time when to input particles.
   tinjt(lc) Total injection time.
   pinpdt(lc) Time increment between particle injections.

4. Total number of simulation particles to be input over the entire injection time: **npinpt.**
5. Determine the total number of simulation particles to be simulated, **msp**, the number of classes, **itpcl**, and the number of different sizes in each class, **itpsz**.

The following is the procedure for the preparation of input for particle initialization:

1. Define the location of the particles to be input for each class and size. The mesh block, **ipblkin(lc)**, in which the particles of class lc are placed, has a default value of 1. Specify the rectangular volume in the computational mesh into which the simulation particles are to be placed by inputting the maximum and minimum *x*-, *y*-, and *z*-coordinates of the particle volume for each

class; these are, respectively, **xpe(lc)**, **xpw(lc)**, **ypn(lc)**, **yps(lc)**, **zpt(lc)**, and **zpb(lc)**. **init_random** allows to select the way of initializing the seed for particle random number generator.

2. Define the number of aerosols to be modeled. Set the number of different particle materials, i.e., particle classes, **itpcl**. Specify the number of particle sizes to be modeled in each of these classes, **itpsz(lc)**. The default value for each of these is 1. The particle class number, **ipclin(lc)**, is currently set by default and does not need to be changed.
3. Characterize the *real* particle material to be modeled. Specify the total material mass of each class and size, **pmass(lc,ls)**. Set the density of the material of each class, **prhoin(lc)**, and specify the particle diameter for each class and size, **pdiamin(lc,ls)**. (These same particle densities and diameters are used for calculating the *simulation* particle dynamics in the code particle transport model.)
4. Define the simulation particle injection number and mode. Set the total number of simulation particles, which includes all the particles of all classes and sizes to be input over all time, **npinpt**. Select the initial time and duration for which particles of each class will be injected by setting the time when to input particles, **twpinp(lc)**, and the total injection time for each class **tinjt(lc)**. Set the time interval between particle injections during this period, **pinpdt(lc)**. To inject a single volume of particles, i.e, a one-time injection, input the time of injection, **twpinp(lc)**. Set the value of **pinpdt(lc)** to a value greater than the final problem time. Select a value for **tinjt(lc)** such that the ratio **tinjt(lc)**/**pinpdt(lc)** is less than 1.0.

To inject into the computational domain a selected simulation particle concentration each computational cycle, a nearly constant computational time step is necessary for this to result in a nearly constant particle concentration input. For particle number concentration input, the *simulation* particle mass is assumed equal to the *real* particle mass, **pmass(lc,ls)**. The standard input parameters are set with the exception of **npinpt** and **pmass(lc,ls)**, which must be computed by the user.

To initialize the particle input in this manner, choose the input parameters **xpe(lc)**, **xpw(lc)**, **ypn(lc)**, **yps(lc)**, **zpt(lc)**, and **zpb(lc)**, and compute the particle input volume. Choose a simulation particle number concentration and compute the number of particles to be injected each computational time step. Next, choose input parameters **tinjt(lc)** and **pinpdt(lc)** and compute the total number of injection times. Use this information to compute the input parameter **npinpt**, the total number of particles input. (If this number is prohibitively large, the number concentration or the input volume must be reduced.) Finally, select the particle size and density and compute the input parameter **pmass(lc,ls)**, the particle mass of each class and size to be input over the total problem time.

In more detail, the setup procedure is as follows:

1. Determine the input volume dimension in the inflow velocity direction to be compatible with $U_{\text{in}} \cdot dt$ (cm).
2. Compute the particle number concentrations, cpinj(lc,ls), for each particle class and size (no. of particles/cm^3).
3. Compute particle input volume, volp(lc), from **xpe(lc)**, etc., for each class (cm^3).
4. Compute the number of particles injected each injection time for each class and size, npinj(lc,ls) = cpinj(lc,ls) ·volp(lc).
5. Select a maximum injection time, **tinjt(lc)** (s), and the particle injection time interval, **pinpdt(lc)** (s).

6. Compute the number of injection times anticipated in the total problem time, ninjts(lc) = **tinjt(lc)** / **pinpdt(lc)**.
7. Compute the number of particles for each class and size, npsim(lc,ls)= **npinj(lc,ls)** ·**ninjts(lc)**.
8. Sum the total number of particles in each class and size. This gives the input parameter **npinpt**, which is the total number of particles to be injected.
9. Select the particle size, **pdiamin(lc,ls)** (cm), and the particle density, **prhoin(lc)** (g/cm^3).
10. Compute the mass of each *simulation* particle class and size injected. The product of this and the number of particles for each class and size is the input parameter **pmass(lc,ls)**, pmass(lc,ls)= [(π/6) • **pdiamin(lc,ls)**^{3} • **prhoin(lc)**] • npsim(lc,ls) (g).

7.6.2 Description of Particle Transport Input Parameters

If **imarker** = 1, the particles input are used as marker particles and moved at the local fluid velocity. Most of the particle physics is bypassed. The default value of **imarker** is 0. In this case, the complete particle dynamics algorithm is executed.

In the two-way momentum coupling, the continuous fluid phase is impacted by the discreted particle phase (and vice versa). To use the two-way coupling option in GASFLOW, **solatype** must be equal to 1 which allows the hydrodynamic calculation with the particle model. The parts NAMELIST variable **fpcoupling** should be set to 2 to activate two-way momentum coupling.

The particle transport model includes the inertial, drag, and gravitational forces. In cylindrical coordinates (**cyl** = 1 in NAMELIST group xput), the centrifugal force also is modeled. The drag coefficient in the Stokes flow region is a function of the flow Reynolds number and, consequently, of the relative particle-fluid velocity. This requires an iterative procedure (the Newton-Raphson method is used) to simultaneously solve for the drag coefficient and the particle velocity. **niterp** is input to set the maximum number of iterations. Generally, the smaller the particle size, the greater the number of iterations required. For example, for 0.5-µm-diameter (RC Guide uses hyphens in units) particles of unit density, it requires about 40 iterations to compute an accurate terminal settling velocity. It requires only 20 iterations for a 100-µm, unit-density particle to compute the correct terminal settling velocity. The default value for **niterp** is set to 30. The slip correction factor, **scfacin(mxpcl,mxpsz)**, reduces the drag force for small particles and should be used for particles with diameters less than about 1.0 µm. The value is always 1.0 or greater, and is 1.165 for 1.0-µm-diameter particles and 2.888 for 0.1-µm-diameter particles. The default value for **scfacin(mxpcl,mxpsz)** is 1.0.

The turbulent diffusion of the particles is characterized by the particle turbulent diffusion coefficient, **tdcp**. Note that there are currently three models in GASFLOW for the turbulent diffusion coefficient of the . The numerical model for turbulent diffusion computes the diffusion velocity components for each particle. In this model, the fluid velocity components at the locations of each particle are the sum of the mean velocity components and the random turbulent velocity component. The basic idea in determining the random turbulent velocity is to consider the particle as a point source that diffuses for a time Δ*t*. The probability of where the particle is likely to move is given by a Gaussian distribution, with a width distribution determined by the standard deviation (4 x Δt x **tdcp**)$^{1/2}$. A random number generator selects the location actually used within the distribution. A default value of 0.0 is set.

The user can specify the model used to move each particle by the parts NAMELIST variable **fluid_velocity**. The user has the option to interpolate two or four fluid velocities for each coordinate direction by specifying **fluid_velocity**.

The coefficient of kinematic viscosity, **visf**, is set by default to the value of **nu** input in NAMELIST group xput and typically does not need to be changed. In the evaluation of the particle Reynolds Number, the user now has the option to input constant values for fluid density and kinetic viscosity by inputting **rhogas** (default = 0.0012) in the $parts namelist, and **nu** (default = 0.1535) in the xput namelist. To generalize the fluid properties presented above, we have introduced two new input variables in the parts namelist: **local_rhogas** (default = 0) and **local_visf** (default = 0).

Similarly, the default value of 2.0 for **wmax**, the maximum argument of the error function used in the inverse error function table, does not need to be changed. The input parameter **inpvol** is not used for the Lagrangian particle setup. It may be used in the future for the combined Lagrangian-Eulerian scheme.

We are now able to exactly simulate Stokes flows by setting the Stokes coefficient in the $parts namelist input stream **stokes(lc,ls)** for each particle class and size.

The following input parameters for the particle transport model are in the parts NAMELIST group.

fluid_velocity Flag for methodology of computing local particle velocity in fluid cell.
= 0, uses local particle cell fluid velocities for computing the fluid velocity at the particle location.
= 1, uses all fluid velocities for computing the fluid velocity at the particle location.

fpcoupling Flag for two-way momentum coupling.
= 1 implies one-way momentum coupling,
= 2 implies two-way momentum coupling.

imarker Marker particle flag.
= 0, the complete particle dynamics algorithm is executed. Default = 0.
= 1, particles are moved at the local fluid velocity (most of the particle physics is bypassed).

inpvol Input parameter to choose actual or fictitious volume assigned to each particle.
= 0, actual volume is computed;
= 1, $\delta x \cdot \delta y \cdot \delta z$ of cell in which particle is located.

local_rhogas = 0 (default), the constant values for **rhogas** is used.
> 0, actual spatial and time dependent local fluid density is used.

local_visf = 0 (default), the constant values for **visf** is used.
> 0, actual spatial and time dependent local fluid kinetic viscosity is used.

niterp Number of iterations for the Newton iteration cycle for determining drag-induced particle velocity. Default = 30.

scfacin(mxpcl, mxpsz) Cunningham slip correction factor for each particle class and size. Default = 1.0.

tdcp Turbulent diffusion coefficient for particles (cm^2/s).
> 0, a constant particle turbulent diffusion coefficient.
= –1.0, then $\mu/(\rho \cdot Sc_p)$ is used for the turbulent diffusion coefficient for particles, where Sc_p is the particle Schmidt Number and μ is the total viscosity of the gas mixture including the turbulent viscosity.
= –2.0, turbulent kinetic energy is used to determine the turbulent diffusion coefficient. In this case, if **tmodel** = 'ke' or **tmodel** = 'alg'**,** the turbulent viscosity is used to determine the turbulent diffusion coefficient. See Section 2.12.2 in Theory Manual for more details

stokes(lc,ls) = 0, solve the equations of particle motion. See Section 2.12.1 in Theory Manual for more details.
> 0, simulate the Stokes flows.

visf Coefficient of kinematic viscosity of fluid.

wmax Maximum argument of error function used in inverse error function table. Default = 2.0.

The procedure for the particle transport model setup is as follows:

1. Set up particle input as described in Section 7.6.1.
2. 2.Select the value of **imarker**. The default value of 0 utilizes the full particle dynamics algorithm.
3. If the particle diameters are 1.0 μm or less, set the slip correction factor, **scfacin(mxpcl,mxpsz)**. The default value is 1.0.
4. If particle turbulent diffusion is to be modeled, set the coefficient **tdcp** to an estimated value. The default value is 0.0 cm^2/s.
5. Choose a value for **niterp**, typically between 10 and 40, depending on the particle size and accuracy required (default = 30).
6. Use the default values for **inpvol** and **wmax**. The values of **rhogas** and **visf** can be set as constant or dynamic by specifying **local_rhogas** and **local_visf**.

7.6.3 Description of Particle Deposition Model Input

The default value of the particle deposition flag, **ipdep(lc)**, is 0. If no particle adhesion upon impact for particle class lc is to be modeled, no further input is necessary. At the other extreme, if all particles adhere upon impact, an **ipdep(lc)** value of 2 is input. A deposition model based on the particle material properties (and some stochastic considerations) is invoked if **ipdep(lc)** is set equal to 1. For example, if the first class of particle material (class number 1) is a liquid mist, then **ipdep(1)** = 2 is appropriate because the impacting material will most likely not bounce. Then, if the second particle class is a harder material and the deposition model is to be used to choose whether a particle bounces or adheres on impaction, **ipdep(2)** = 1 is input.

The deposition model uses the Dahneke theory of rebound to compute a critical rebound velocity, at which there is a 50% probability of bounce. (The code randomly chooses which particles inside a window bracketing the critical rebound velocity actually bounce.) This critical rebound velocity is a function of the impacting particle mass and energy and the coefficient of restitution.

The particle energy is a function of the Hamaker constant, **hca**, the particle diameter, **pdiamin(lc,ls)**, the separation distance of the particle and substrate, **sdz**, and particle bulk mechanical properties, bmck(lc), which are a function of the material Poisson ratio, **poisrt(lc)**, and Young's modulus, **yngmod(lc)**.

The Hamaker constant has been determined from theory and experiments for many particle-surface combinations. The values range from about $5.0 \cdot 10^{-13}$ to $5.0 \cdot 10^{-12}$ erg. A typical value of $1.0 \cdot 10^{-12}$ erg is used for the default value of **hca**. The equilibrium separation distance, **sdz**, is estimated to be $4.0 \cdot 10^{-8}$ cm. This may vary according to surface roughness. The default value for **sdz** is this estimated value. However, the influence of surface roughness can be qualitatively modeled by increasing the value of **sdz** to the estimated roughness height.

The coefficient of restitution, **core**, has been experimentally determined for many particle-surface combinations. The values typically range from 0.90 to 0.99 for a hard particle impinging onto a hard surface. The input parameter **core** has a default value of 0.96. Experimental data shows the trend for the coefficient of restitution to reach this maximum value at a given incident velocity and then almost immediately the ratio of rebound velocity to incident velocity begins to decrease as the incident velocity increases. This phenomenon is programmed into the code and does not need any input data.

Values for the mechanical properties, Poisson ratio, **poisrt(lc)**, and Young's modulus, **yngmod(lc)**, are in appropriate handbooks. The Poisson ratio is always less than 0.5. It ranges from 0.42 for gold to 0.16 for fused quartz. The default value is 0.29, which is the value for steel. Youngs' modulus ranges from $2.10 \cdot 10^{+12}$ dynes/cm^2 for steel to $3.24 \cdot 10^{+10}$ dynes/cm^2 for plexiglass.

Deposition is in a real sense a stochastic process that follows the general trend of the theoretical and empirical models developed from and compared with experimental data. Because of this, it is a reasonable assumption that some small, unknown percentage of the particles that impact a surface will adhere. To account for this, the parameter **depperc(lc)** is input. The default value is set to 5.0%. If the substrate is dirty or there are other reasons to believe a smaller or larger percentage of the particles will adhere on impact, this value can be changed.

In general, the harder the particle and surface, the greater the surface roughness, the larger the particle, and the greater the velocity, the more likely it is that bounce will occur. Bounce is not a problem for liquids or easily deformed material, such as tar.

The following input parameters are in the parts NAMELIST group:

core	Coefficient of restitution of particle material.
depperc(lc)	Percentage of particles that immediately deposit upon impact.
hca	Hamaker constant (ergs).
ipdep(lc)	Particle deposition flag input for each class: if **ipdep(lc)**= 0, no adhesion; = 1, adhesion determined by deposition model; = 2, all impacting particles adhere.
ndxpd	Number of longitudinal sections in *x*-direction in which particle deposition (only on bottom of mesh) is monitored: if - 1, no plot.
pdiamin(lc,ls)	Particle diameter input for each class and size (cm).
poisrt(lc)	Poisson ratio of particle class material.

prhoin(lc) Material density input for each particle class (g/cm^3).

sdz Separation distance of particle and substrate (cm).

yngmod(lc) Young's modulus of particle material (dynes/cm^2).

The following are the setup procedures for the particle deposition model:

1. Set up particle input as described in Section 7.6.1, Particle Initialization, and in Section 7.6.2, Particle Transport. This includes the input for the diameter of each class and size particle, **pdiamin(lc,ls)**, and the particle material density for each particle class, **prhoin(lc)**.
2. Select a value of **ipdep(lc)** for each particle class being modeled. There is no deposition of particles when the default value of 0 is set.
3. Use the default values for the Hamaker constant, **hca**, the particle-substrate separation distance, **sdz**, and the coefficient of restitution, **core**. These input parameters may be changed if there is a reason to do so.
4. Input appropriate material property values, Poisson ratio, **poisrt(lc)**, and Young's modulus, **yngmod(lc)**, for all particle classes for which the deposition model is used, that is, when **ipdep(lc)** = 1. The default values are those for steel.
5. Use the default value (5%) for the percentage of particles that deposit on impact, **depperc(lc)**; or, considering the condition of the substrate and any other pertinent information, set it to a better guess.
6. If a plot is wanted of the total particle mass deposited on the "floor" of the mesh (i.e., the bottom face of the k = 2 layer of cells) as a function of distance in the *x*-direction, set the input parameter **ndxpd** to the number of sections into which the distance from the mesh minimum *x*-coordinate to the maximum *x*-coordinate is to be divided and in which the deposition is to be monitored. If this information is not wanted, do not set this parameter, and the default value is such that this computation and plot will be skipped.
7. If time-history plots are wanted for the computational mesh cell faces onto which particles have deposited, input the necessary information according to instructions in Section 9.1.5, Graphics and Tabular Particle Data Output.

7.6.4 Description of Entrainment Input Parameters

The default value for the particle entrainment flag, **intrn**, is 0. If particle entrainment is not to be modeled, no further input is required. If particle entrainment is to be modeled, set **intrn** to 1, and the entrainment model is used to suspend deposited particles when the fluid velocity equals or exceeds the computed particle threshold suspension velocity. Criteria for determining the fluid velocity at which a particle initially at rest on a surface will become suspended is determined from a force balance equation that includes gravity, adhesion, fluid lift, drag, and friction forces. A particle adhering to a surface will be dislodged when the removal forces equals or exceeds the force of particle adhesion. This force balance equation is modified by experimental data that takes into account the effects of particle interactions with other particles, and the, usually, nonspherical particle shape. The gravitational, buoyant, and adhesive forces depend only on the physical properties of the particle and gas densities, and they are independent of the gas stream velocity. The adhesive force is the van der Waals inter-surface molecular force and is a function of the Hamaker constant, **hca**, the particle diameter, **pdiamin(lc,ls)**, the separation distance of the particle and the

substrate, **sdz**, and particle bulk mechanical properties, bmck(lc), which is a function of the material Poisson ratio, **poisrt(lc)**, and Young's modulus, **yngmod(lc)**. These parameters are described in the discussion of particle deposition. The drag and lift forces depend on the gas stream velocity. The friction force is proportional to the coefficient of sliding friction and is set to a value of 0.45, which is the value suggested from experimental studies.

To compute the lift and drag forces, the velocity at the center of the stationary particle, which is typically in the viscous sublayer, must be known. The law-of-the-wall equation, which characterizes the fluid velocity in the turbulent boundary layer near the wall, is used to calculate the characteristic shear velocity and, subsequently, the velocity at the center of the particle. A no-slip wall boundary condition is assumed. In addition, the velocity gradient across the surface boundary layer must be known to compute the lift force. These computations require an estimate of the boundary layer thickness at the location of the deposited particle. If the input parameter **dbl** is set to a value less than 0, the code will compute a boundary layer thickness as a function of a Reynolds number and the distance along the wall from the point at which the turbulent fluid initially contacts the wall. It is assumed that this is the point at which the fluid enters the computational mesh. However, if this model is not compatible with the problem setup, an estimate of a constant boundary layer thickness may be set by the parameter **dbl**.

The input parameters **cdrf** and **sdzrf** are multipliers for the computed drag coefficient and the usual surface-substrate separation distance, respectively, that allow a qualitative representation of the roughness of the particle and substrate surfaces. The default values are 10.0 and 100.0, respectively. These values were determined by comparison of computations with one set of experimental data and require further evaluation.

The following input parameters are in the parts NAMELIST group:

intrn	Input parameter to choose particle entrainment option: = 0, no particle pickup (default and no more input is required.); = 1, pickup option is on.
dbl	Boundary layer thickness. Default = 10 cm.
cdrf	Aerodynamic drag coefficient roughness factor. Default =10.
sdzrf	Particle-substrate separation distance roughness factor. Default = 100.

The procedure for the entrainment model setup is given below:

1. Set up particle input and deposition input as described in Sections 7.6.1, 7.6.2, and 7.6.3. This includes the input for the diameter of each class and size particle, **pdiamin(lc,ls),** and the material property values, Poisson ratio, **poisrt(lc)**, and Young's modulus, **yngmod(lc)**, for all particle classes for which the entrainment model is used. The default values for the Hamaker constant, **hca,** and the particle-substrate separation distance, **sdz**, are set.
2. Select a value of **intrn**. There is no entrainment of particles for the default value of 0. Input a value of 1 to turn on the entrainment model.
3. Set **dbl** to –1.0 for the code to compute the boundary layer thickness, or set an estimate of a constant boundary layer thickness. The default value is 10.0 cm.
4. Use the default values of **cdrf** and **sdzrf**. These may be changed if better estimates are known.

7.6.5 Description of Particle Cloud Model Input Parameters

The concept of the discrete computational simulation particle representing a multitude of real particles, all of which are located at the same point in space as the simulation particle, is extended in the particle cloud model. This model permits each simulation particle to represent a multitude of real particles that disperse as a Gaussian cloud.

The default value of the particle cloud model flag, **icloud**, is 0. If the particle cloud model is not to be used, no further input is necessary. If the cloud model is to be used (**icloud** = 1), then a value for the particle cloud diffusion coefficient, **pcdc**, is chosen. The default value is 0. An estimated value for this input parameter is the value of the turbulent diffusion coefficient, **tdcp**.

The cloud model is designed to be used with particle monitors at selected locations in the computational domain. The total number of monitors is specified by the input parameter **ntmntr**, which has a default value of 0. The selected number of monitors must not be greater than the maximum number of monitors, mntrmx**,** which is set in a parameter statement. The current value is 20. The monitor locations are specified by the input parameters **xm**, **ym,** and **zm** for each monitor**.**

Time-history plots of the real particle mass detected by monitors is plotted using the procedure described in Section 9.1.1.

The input parameters are in the parts NAMELIST group:

icloud Particle cloud model flag.
= 0, cloud model off (default);
= 1, cloud model is on.

ntmntr Total number of particle cloud monitors.

pcdc Particle cloud diffusion coefficient. If **pcdc** is greater than zero, it is used for the particle cloud diffusion coefficient. If **pcdc** = –1.0, then the particle cloud diffusion coefficient is equal to Sc, where μ is the total viscosity of the gas mixture including the turbulent viscosity. Default = 0.

xm *x*-coordinate of particle cloud monitors.

ym *y*-coordinate of particle cloud monitors.

zm *z*-coordinate of particle cloud monitors.

The following is the setup procedure for the particle cloud model:

1. Choose **icloud** = 1 to turn on the particle cloud model.
2. Set **pcdc** to an estimated value.
3. Choose the total number of monitors, **ntmntr**, and their locations, **xm**, **ym**, and **zm**.
4. Specify in the graphics input parameter **pthp** 'pmntr' for time-history plots of the real particle cloud mass at each monitor.

7.6.6 Particle Model Restart

Setting the NAMELIST group xput input parameter **tddt** to the appropriate time interval between tape dumps will generate a GASFLOW-MPI restart file (see Section 9.2) file from which to restart a run. A common procedure in running the aerosol model is to set **solatype**, also in NAMELIST group xput, to a value of 0.0, for a fluid solution only, and generating a steady-state fluid flow field. Then, **solatype** = 2.0 is set for particle transport only, and particles are injected into this steady flow field. Parameters that may need changing when restarting from a tape dump include: **twfin, maxcyc, solatype, nrsdump, pthpt0 and twpinp.**

7.7 Special Containment Models

In order to accurately model modern nuclear containments, it was found that three special models were necessary. The two models (i.e., sump, and recombiner models) are described below.

7.7.1 Sump Model

GASFLOW-MPI allows the user to specify a sump, which is basically a water film that has a constant thickness and a time-dependent temperature. The relevant phenomenon is shown in Figure 7-2. We see that the sump mass is determined by five sources:

1. Droplet depletion or rainout of the liquid component of the fluid field directly into the sump;
2. Liquid films draining from slabs and walls directly into the sump;
3. Phase change (condensation and/or evaporation) at the sump surface;
4. Sump to Sump mass exchange, and
5. Direct mass addition.

Energy is also transferred to the sump by the above mentioned five mechanisms and in addition, there can be convection and radiation heat transfer at the sump surface plus the source of fission product heating or the energy of radio nuclide decay.

Currently for the sump model, we assume that there is only one sump, that is, the sump to sump mass and energy exchange shown in Figure 7-2 is not applicable for this version of the sump model. We further assume a lumped parameter approach where the sump is represented by one control volume, which contains the total sump mass and energy, and therefore a single temperature. We are not modeling a dynamically draining film from slab, wall and sink heat transfer structures, but instead, we set a maximum film thickness through input and when the film thickness exceeds this input value, we directly transfer the excess mass and it's associated energy directly to the sump.

To activate the sump model, the input parameter **ihtflag** in NAMELIST RHEAT must be none zero and positive, and the liquid (**h2ol**) and vapor (**h2o**) components of water must be specified with the **mat** input variable in NAMELIST XPUT.

The sump model or the old constant film thickness model is activated by inputting information in the NAMELIST RHEAT input stream with the **cfilmdef** input variable array. This input variable array, **cfilmdef(*,n)**, is defined for the n^{th} surface as:

cfilmdef(1,n) beginning i mesh index (cell face number).
cfilmdef(2,n) ending i mesh index (cell face number).
cfilmdef(3,n) beginning j mesh index (cell face number).
cfilmdef(4,n) ending j mesh index (cell face number).
cfilmdef(5,n) k mesh index.
cfilmdef(6,n) k mesh index.
cfilmdef(7,n) block number (must be 1 for GASFLOW-MPI)
cfilmdef(8,n) > 0, constant film thickness (must be less than 100 cm)
< 0, sump number (must be a negative integer)

The asterisk (*) should be replaced by an integer that identifies the particular **cfilmdef** definitions. GASFLOW supports 500 definitions for **cfilmdef**.

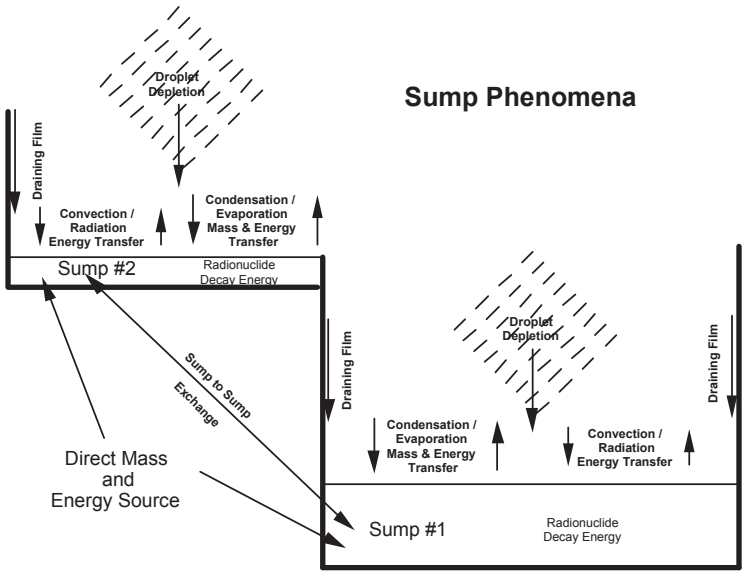


Figure 7-2 Schematic drawing of the relevant phenomenon represented in the GASFLOW-MPI Sump Model. Note that in this current version, there is no Sump to Sump Exchange.

We will see in the examples below that there are actually two sump model options:

1. A simple user controlled temperature, and
2. A more mechanistic model with mass and energy balances. **cfilmdef(8,n)** controls the actions of this model.

When **cfilmdef(8,n)** > 0 then the temperature of the sump, **sumptemp**, as a function of time, **sumptime**, can be controlled, and input in either degrees **Celsius** or Kelvin. This can be accomplished via the input variable **nsumppts** and the array variables **sumptemp** and **sumptime** in NAMELIST group rheat. The definitions are as follows:

nsumppts Number of sump temperatures in the **sumptemp** and **sumptime** arrays.
Absolute value of **nsumppts** must be less than 100.
< 0, **sumptemp** values are in °C.
> 0, **sumptemp** values are in K.

sumptemp Sump temperatures < 100 values.
sumptime Time for the sump temperatures < 100 values.

GASFLOW supports a table of 99 paired temperature vs. time values to describe the sump temperature in either degrees Celsius or Kelvin.

When specifying a sump, the user must

1. Define a horizontal wall (**walls**) or obstacles (**mobs**) that coincides exactly with the sump location.
2. Define a constant film thickness for the sump with the **cfilmdef** input variable that exactly coincides with the defined wall or obstacle from 1.
3. Make certain that 'h2o' is listed in the problem composition, **mat**, and additionally 'h2ol' if the two-phase homogeneous equilibrium model(HEM) is to be used.
4. Input a sump temperature as a function of time through the paired single-dimensioned arrays **sumptemp** and **sumptime**.

An example is as follows:

1. We assume the same geometry as the example from Figure 3-29.
2. The sump is located at cell edge k = 2 and covers the entire bottom of the computation domain, i.e., 1 < i < 11 and 1 < j < 11, and a wall is constructed to indicate this sump.
3. The HEM is activated.
4. A constant film thickness is established to exactly coincide with the wall from 2.
5. Eight paired temperature and time values controlling the sump temperature in degrees **Celsius or Kelvin** are defined.

The input stream is shown here:

```
$xput
…
 mobs        = 3, 10, 1, 11, 5, 8, 1, 1,  ; solid obstacle
 walls       = 2, 10, 1, 11, 3, 3, 1, 2,  ; horizontal wall
               2, 2, 1, 11, 6, 10, 1, 2,  ; vertical wall
               1, 11, 1, 11, 2, 2, 1, 2,  ; sump surface
 holes       = 5, 7, 4, 7, 6, 8, 1, 0, 0, 0, 0, 0, 1,  ; top hole
               8, 9, 5, 6, 5, 8, 1, 0, 0, 0, 0, 1, 1,  ; thru hole
               8, 9, 5, 6, 2, 4, 1, 1, 1, 1, 1, 1, 1,  ; wall hole
 mat         = 'h2', 'n2', 'o2', 'h2o', 'h2ol',  ; components -> HEM
 cfilmdef    = 1, 11, 1, 11, 2, 2, 1, 5.0,  ; sump film thickness
…
$end
$rheat
…
 nsumppts    = -8,  ; sump paired temperature and time values in Celsius
```

sumptemp = 25.0, 37.0, 50.0, 45.0, ; sump temperature
60.0, 62.0, 63.0, 63.0, ; sump temperature
sumptime = 0.0, 2800.0, 6200.0, 9480.0, ; sump time
12600.0, 14400.0, 18000.0, 30000.0, ; sump time
…
$end

A display of the geometry including the sump is presented in Figure 7–3.

When **cfilmdef(8,n)** < 0, for this version of the sump model, **cfilmdef(8,n)** = -1, we must provide additional sump parameters or characteristics. This is accomplished with the **sumpchar(*,n)** input variable array in the NAMELIST RHEAT input stream, where **n** is the sump number (1 for this version). So this input variable array, **sumpchar(*,n)**, is defined for the n^{th} sump as:

sumpchar(1,n) initial sump temperature
> 0 for degrees Kelvin and
< 0 for degrees Celsius.
sumpchar(2,n) initial sump depth (cm).
sumpchar(3,n) maximum sump depth (cm).

Additional important input in the NAMELIST RHEAT input stream is:

filmth > 0, maximum film thickness (default is 0.1 cm). When the film exceeds this value, the excess mass and energy is transferred to the sump.
< 0, a liquid film is initialized on all slab, wall, and sink surfaces with a thickness of |**filmth**|.

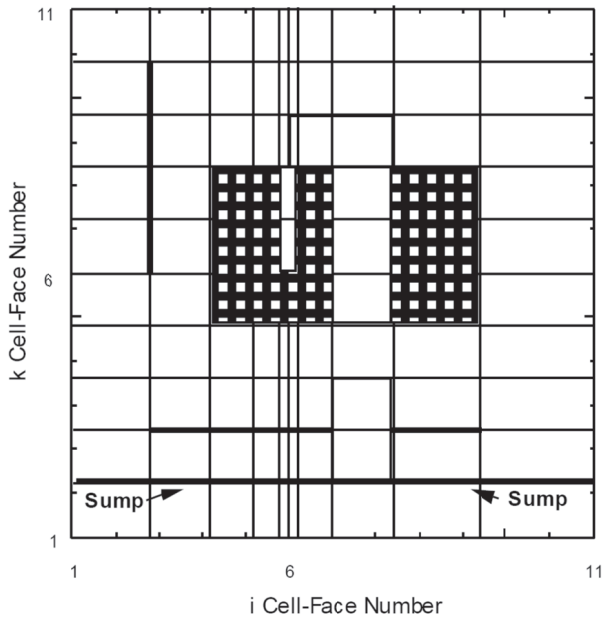


Figure 7-3 Example of a sump with a modification of Figure 3-29.

Several tables of sump sources can also be input in the NAMELIST RHEAT input stream. These include the direct source of mass (**sumpmas**) and energy (**sumpengy**) and the energy source due to radio nuclide decay energy (**sumprn**). These tables can be specified as functions of time with input variable array (**sumptime**). Each of these input sump source variable arrays may accept up to 99 values where the scalar variable **nsumppts** defines the actual number of table entries. For completeness we define these variables as:

sumptime number of input values for each of the sump source tables in seconds
> 0, **sumpengy** is in ergs/s
< 0, **sumpengy** is in ergs/(g·s)

sumpengy energy source associated with **sumpmas**, which is the direct energy source in ergs/s or ergs/(g·s), depending upon the sign of **sumptime**.

sumpmas mass source associated with **sumpengy**, which is the direct mass source in g/s.

sumprn energy source associated with the decay of fission products, radio nuclides, in ergs/s.

An example input of these options could be:

```
$rheat
…
 nsumppts   =  4,
 sumptime   =  0.0, 750.0, 1250.0, 3600.0,
 sumpmas    =  0.0, 1000.0, 2000.0, 0.0,
 sumpengy   =  0.0, 1.2e+10, 2.0e+10, 0.0,
 sumprn     =  0.0, 6.0e+12, 8.0e+12, 1.0e+13,
...
$end
```

7.7.2 Recombiner Model

Recombiner boxes of the NIS and Siemens design are modeled in the current version of GASFLOW-MPI. In addition, the GRS model for a single recombiner foil is included for completeness**.** We provide an example of the recombiner input, but first we'll discuss the details of the input variables.

The location in the i, j plane for each recombiner is given in the **rcombdef** array described below. The **rcombdef** array is in the rheat NAMELIST group.

If **rcombdef(1,*)** = i1, **rcombdef(3,*)** = j1, and **rcombdef(5,*)** = k1**,** then the location of the recombiner in the i, j plane is at i1+1, j1+1. The energy source term due to the recombiner will be at location i1+1, j1+1, k1+1. The location of the cell for which the inlet conditions for the recombiner are determined will be at i1+1, j1+1, k1+1+offset, where offset is given by **rcombdef(10,*)**,and the velocity boundary condition for the recombiner flow rate will be specified at the *z*-face of cell i1+1, j1+1, k1+1+offset. Therefore, to use inlet compositions from cells below the recombiner, offset must be negative. The inlet compositions are only used in the correlations for volumetric flow through the recombiner. For the Siemens type FR-90/1 designs, the offset is zero, so the composition, recombination, and energy release due to recombination are located in

the catalytic reaction volume. For the GRS foil, the reaction volumes are all fluid cells with common cell faces to the foil.

rcombdef(1,*) Beginning i mesh index (cell face number).

rcombdef(2,*) Ending i mesh index (cell face number).

rcombdef(3,*) Beginning j mesh index (cell face number).

rcombdef(4,*) Ending j mesh index (cell face number).

rcombdef(5,*) Beginning k mesh index (cell face number).

rcombdef(6,*) Ending k mesh index (cell face number).

rcombdef(7,*) Block number (must be 1 for GASFLOW-MPI).

rcombdef(8,*) Actual flow area into the recombiner irrespective of the computational mesh (not used with Siemens correlations 5 <= type<=9)

rcombdef(9,*) Recombiner type.
= 1, implies NIS recombiner model with chemistry based on compositions at the location identified by the offset level, and forced volumetric flow through the recombiner based on the correlation for the NIS recombiner;
= 2, implies Siemens recombiner model with chemistry based on compositions at the location identified by the offset level, and forced volumetric flow through the recombiner based on the correlation for the Siemens recombiner;

= 3, implies Siemens recombiner model with chemistry based on compositions in the catalytic reaction volume according to the correlations for Siemens Type FR 90/1 Type 3 allows early downflow from containment convection and flow reversal to upflow from catalytic recombination
= 4, implies GRS recombiner foil model with chemistry based on compositions in the fluid cells with common faces to the specified foil
= 5, implies Siemens correlation for box type FR90/1–100, this box was used also in GX tests
= 6, implies Siemens correlation for box type FR90/1–150
= 7, implies Siemens correlation for box type FR90/1-320
= 8, implies Siemens correlation for box type FR90/1-960
= 9, implies Siemens correlation for box type FR90/1-1500
= 10-15 , implies old NIS PARs from Biblis A LOCA simulation for reruns of archive case
= 16, PAR model for Reinecke net recombiner no longer used
= 17, temperature correction to Areva PAR correlation no longer used
= 18, New NIS PAR validated with test THAI HR 14
= 19, Areva PAR FR380 from THAI HR Tests

rcombdef(10,*) Offset. It can be only set to -1 due to the limitation of layers of ghost cells in the parallel code.

rcombdef(11,*) Threshold value for hydrogen concentration. Hydrogen concentration must be above this value before recombiner will operate. User can input this choice. Default is 0.03.
If **rcombdef(11,*)** is zero and **rcombdef(9,*)** is one, then **rcombdef(11,*)** will be set to 0.008.
If **rcombdef(11,*)** is zero and **rcombdef(9,*)** is two, then **rcombdef(11,*)** will be set to 0.031.
If **rcombdef(11,*)** is zero and recombdef(9,*) is >=5 and <=9 then **recombdef(11,*)** will be set to 0.02.
If **rcombdef(11,*)** is zero, and **rcombdef(9,*)** is >=10 and <=15 **rcombdef(11,*)** will be set to 0.008.
If **rcombdef(11,*)** is zero and **rcombdef(9,*)** is 18 **rcombdef(11,*)** will be set to 0.015 as in THAI test HR14.
If **(rcombdef(11,*)** is zero and **rcombdef(9*)** is 19 **rcombdef(11,*)** will be set to 0.02 as in THAI HR3.
If **rcombdef(11,*)** is less than zero, then **rcombdef(11*)** is the time in seconds when the recombinder will begin to operate.

rcombdef(12,*) The user can input any time constant above zero. Default value is set to 100s if the user inputs zero. If the time constant is input as zero for recombiner type 1, and 10 to 15 then the time constant is set to 1800.0 s. The time constant is not used for recombiner type 4.

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

GASFLOW-MPI supports 500 definitions for recombiners. The time history of the total recombination rate of all PARs is stored as variable **sumh2loss** in the plothist.nc file. The time history of the recombined hydrogen mass itemized for each PAR is stored as variable **recmass(ncomb)** in the plothist.nc file with ncomb being the total number of PARs. ncomb is automatically determined from the input rcombdef.

When specifying a recombiner, the user must

1. Define at least hydrogen, oxygen, and water vapor in the problem composition.
2. Define the recombiner box or foil.
3. Define the catalytic reaction volume, the type of recombiner, offset for the NIS and Siemens forced volumetric flow rate models, and other recombiner characteristics defined above.

An example is as follows:

1. We assume the same geometry as the example from Figure 3-2, but without the internal obstacles and walls other than the recombiner geometries.
2. Introduce a NIS recombiner box type 1 (see 1 in Figure 7–4).
3. Introduce a Siemens recombiner box type 2 (see 2 in Figure 7–4).
4. Introduce a Siemens Type 3 FR90/1 recombiner box (see 3 in Figure 7–4).
5. Introduce a catalytic zone type 4 GRS foil on each side of a defined wall.

The input stream is shown here:

```
$xput
…
 walls      =  9, 10, 6, 6, 6, 9, 1, 2,  ;NIS north wall (type 1)
               10, 10, 5, 6, 6, 9, 1, 2,  ;NIS east wall (type 1)
                9, 9, 5, 6, 6, 9, 1, 2,   ;NIS west wall (type 1)
                9,10, 5, 5, 6, 9, 1, 2,  ;NIS south wall (type 1)
                3, 4, 6, 6, 2, 8, 1, 2,   ;Siemens north wall(type 2)
                4, 4, 5, 6, 2, 8, 1, 2,   ;Siemens east wall (type 2)
                3, 3, 5, 6, 2, 7, 1, 2,   ;Siemens west wall (type 2)
                3, 4, 5, 5, 2, 8, 1, 2,   ;Siemens south wall (type 2)
                3, 4, 5, 6, 8, 8, 1, 2,   ;Siemens top wall(type 2)
                8, 9, 6, 6, 2, 5, 1, 2,   ;Siemens north wall(type 3)
                9, 9, 5, 6, 2, 5, 1, 2,   ;Siemens east wall (type 3)
                8, 8, 5, 6, 2, 5, 1, 2,   ;Siemens west wall (type 3)
                8, 9, 5, 5, 2, 5, 1, 2,   ;Siemens south wall (type 3)
                6, 6, 4, 8, 7,10, 1, 2,   ;GRS foil model (type 4)
 mat        =  'h2','n2','o2','h2o','h2ol',
…
$end


$rheat

…

 rcombdef   =  9, 10, 5, 6, 7, 8, 1, 1.0e+04, 1, -2, 0, 0,   ; NIS (type 1)
               3, 4, 5, 6, 4, 5, 1, 2.0e+02, 2, -3, 0, 0,    ; Siemens (type 2)
               8, 9, 5, 6, 3, 4, 1, 9.6e+03, 3, 0, 0, 0,     ; Siemens (type 3)
               5, 7, 4, 8, 7, 10, 1, 5.0e+04, 4, 0, 0, 0,    ;GRS foil (type 4)
 matcomb    = 2, ; par material for foil type 4
 walldef    = 2, 0.1, 0.0, 0.0, 0.0, ; GRS foil material 2
…

$end
```

A display of the geometry including the four types of recombiners is presented in Figure 7–4.

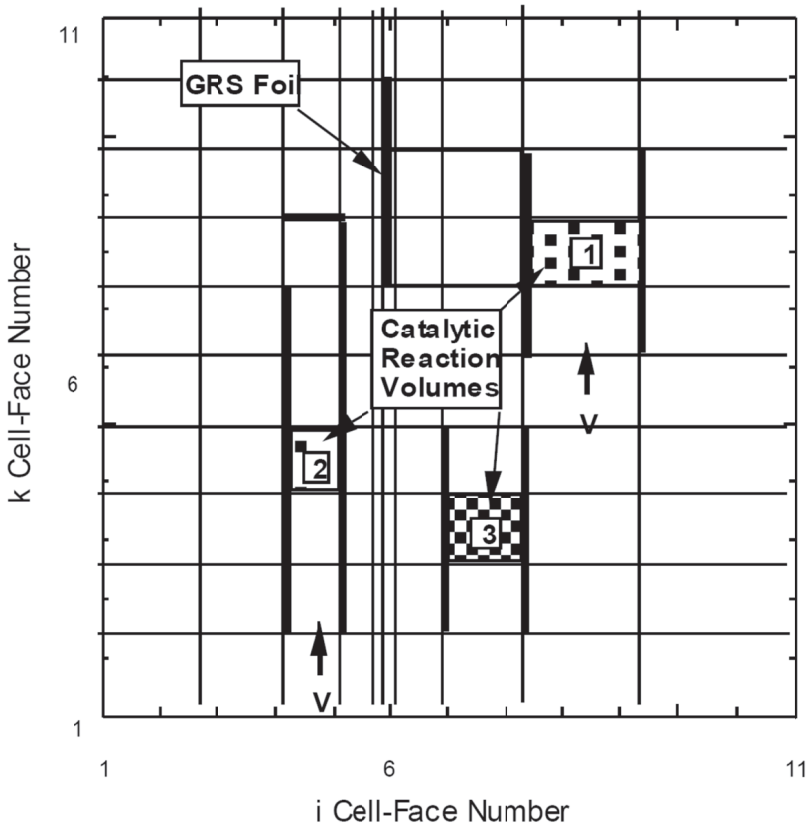


Figure 7-4 Examples of recombiner types and setup for a vertical cut through the computational mesh at j = 6. Type 1 is a NIS forced volumetric flow recombiner, type 2 is a Siemens forced volumetric flow recombiner, and type 3 is a Siemens FR90/1 design.

7.7.2.1 Recent Updates to the Box Reco Models

The above rcombdef definitions allow activation of two new box type PAR models. They were implemented for the GASFLOW analysis of the THAI HR recombiner tests and define the tested NIS and Areva PARs.

7.7.2.1.1 New NIS PAR correlation (rcombdef(9,*) = 18)

The THAI test HR14 used the new design of the recombiners marketed by Nuclear Ingenieur Service as a 1/8 module scaled down to the size of the THAI facility. The test had hydrogen injection from below and a phase during which nitrogen injection from the top simulated hydrogen recombination in the transition to oxygen starvation.

The suggested correlation from the vendor predicted the hydrogen removal rate quite well in test HR14. This test also covered the transition to oxygen starvation by adding some nitrogen in the late phase. The implemented correlation for hydrogen removal also includes correction factors that simulate the transition to oxygen starvation. For rcombde(9,*) = 18 the code determines the H_2 removal rate R_{H2} from the following correlation:

$$R_{H2}[g/s] = 671/3.6 * (conref)^{1.307} * p * (1 - 0.05 * (p-1))/T * Type * Stack * fak \qquad \text{Equ. 7-20}$$

The parameters are defined at the inlet of the box as

$$\psi_{H2in},\psi_{O2in} = \ Volume \ \ fractions \ \ hydrogen,oxygen \ (KCHin,KCOin)$$
$$conref = \min(\psi_{H2in},2\psi_{O2in})$$
$$p = pressure \ [bar] \ (DPA77H16)$$
$$T = \ Gas \ \ Temperature \ \ PAR \ \ inlet[K] \ (KTFin)$$
$$Stack = \ 1.25$$
$$Type \ = \ 11$$

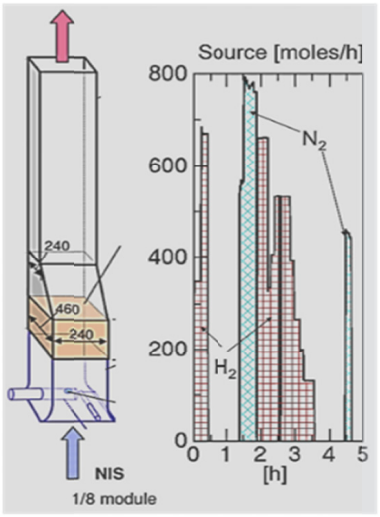


Figure 7-5 NIS Module with N_2, H_2 Source tested in THAI test HR14

The factor fak corrects for oxygen starvation when the oxygen volume fraction at the PAR inlet drops below 6 Vol%. It is defined as

$$fak = \begin{cases} = 1.0 \ for \ \psi_{O2in} > 0.06 \\ = \exp(-48.391*(0.06-\psi_{O2in})+540.25*(0.06-\psi_{O2in})^2) \ \ for \ \ \psi_{O2in} \le 0.06 \end{cases} \qquad \text{Equ. 7-21}$$

Stack and Type in the above correlation are lumped together in the parameter rcombdef(8,*). For test HR14 rcombdef(8,*) should be set to 11*1.25 = 13.75. With the known parameters Stack and Type, the correlation from Equ. 7-20 can be applied to all types of NIS PARs. The PAR efficiency η_{fit}, i.e. the fraction of the inflowing volume of hydrogen that is recombined has been determined from fits to the measured flow in test HR14 as

$$\eta_{fit} = 0.42 * fak_{\eta}$$
$$with$$
$$fak_{\eta} = \begin{bmatrix} 1.0 \ for \ \psi_{O2in} > 0.06 \\ \exp(-20.954*(0.06-\psi_{O2in})) \ \ for \ \psi_{O2in} \le 0.06 \end{bmatrix} \qquad \text{Equ. 7-22}$$

The PAR efficiency doesn't alter the rate of hydrogen removal, but it is used with the NIS box model to enhance the stationary flow through the PAR. More hydrogen than what the removal rate in Equ. 7-20 requires is ventilated into the box to maintain the measured PAR efficiency.

As shown in the theory manual the quasi stationary flow $\dot{V}0_{in}$, which is determined from the correlation in Equ. 7-20 with the efficiency is approached with a relaxation time constant τ using the correlation:

$$\frac{d\dot{V}_{in}}{dt} = \frac{1}{\tau} * (\dot{V}0_{in} - \dot{V}_{in}) \qquad \text{Equ. 7-23}$$

with $\dot{V}_{in}$ being the transient flow rate that is actually enforced by a velocity boundary condition at the PAR inlet.

The following sample input demonstrates the use of the NIS PAR correlation derived for the NIS 1/8 module from THAI test HR14. The NIS PAR type is defined by rcombdef(9,*)=18. Stack and type of the NIS PAR were lumped together into the multiplier rcombdef(8)=11*1.25=13.75. The PAR is simulated as a 4 wall box that is open at the top and bottom. Applied is a 2D Cartesian mesh with 4x, 2y and 5 z nodes and continuous inflow of a mixture of 90% nitrogen with 5% hydrogen and 5% oxygen from the bottom at 10 cm /s. Outflow is defined by a pressure boundary condition at the top defined by a gasdef statement. Default values for this PAR are a startup threshold rcombdef(11,*) of 1.5 Vol% H_2 and a time constant rcombdef(12,*) of 100 s.

```
$xput
...
 mat          = 'n2','o2','h2', 'h2o', 'h2ol',
 gasdef(1,1)  = 1, 'im1', 1, 'jm1', 1, 'km1', 1, 1.0e6, 325.0, 2, 0., 0.,
                'n2', 0.9, 'o2', 0.05, 'h2', 0.05,  ; initial condition

 gasdef(1,2)  = 1, 'im1', 1, 'jm1', 0, 1, 1, 1.0e6, 325.0, 2, 0., 1.e+99,
                'n2', 0.9, 'o2', 0.05, 'h2', 0.05, ; inflow from bottom
 gasdef(1,3)  = 1, 'im1', 1, 'jm1', 'km1', 'kmax', 1, 0.99E+6, 325.0, 2, 0., 1.e+99,
                'n2', 0.9, 'o2', 0.05, 'h2', 0.05, ; top boundary
 pbc(1,1)     = 1, 'im1', 1, 'jm1', 'kmax', 'kmax', 1, 0., 1.e+99,
 vbc(1,1)     = 1, 'im1', 1, 'jm1', 1, 1, 1, 1, 0.0, 1.e+99,
 vvalue       = 10.0,
 walls        = 1, 2, 1, 1, 3, 4, 1, 0,  ; par1
                1, 2, 2, 2, 3, 4, 1, 0,
                1, 1, 1, 2, 3, 4, 1, 0,
                2, 2, 1, 2, 3, 4, 1, 0,
...
$end
$meshgn
...
```

iblock = 1,
xgrid = 0.0, 100.0, 200.0, 300.0,
ygrid = 0.0, 100.0,
zgrid = 0.0, 100.0, 200.0, 300.0, 400.0,
…
$end
$rheat
…
ihtflag = 1,
ircomb = 1,
rcombdef(1:12,1) = 1, 2, 1, 2, 3, 4, 1, 13.75, 18, -1, 0.0, 0.0, ; NIS PAR
…
$end

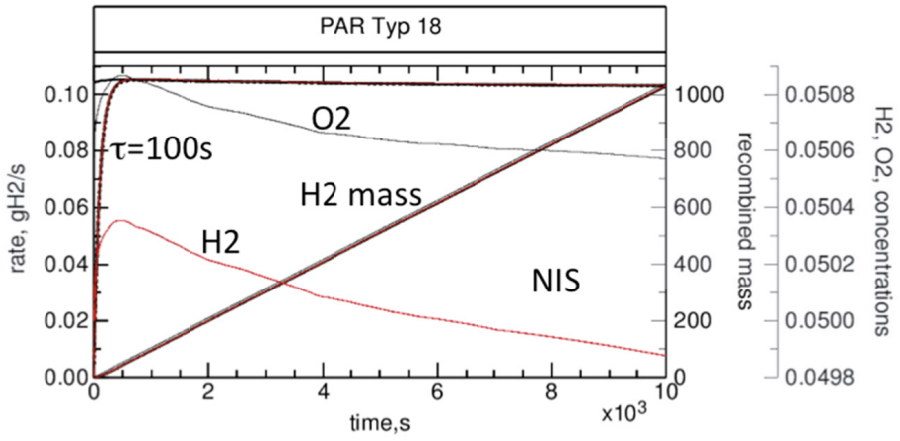


Figure 7-6 GASFLOW calculated NIS PAR rates with oxygen starvation and cumulated H_2 removal (red) compared to analytical evaluation (black) using calculated parameters at the PAR inlet.

Figure 7–6 gives the resulting PAR rate [g H_2/s] and the cumulated H_2 removal as calculated in GASFLOW and as evaluated directly from the correlations given above. With the correction for the applied time constant of 100 s the analytical evaluation applying a first order differential equation solver for Equ. 7-23 is demonstrated to match the calculated results.

7.7.2.1.2 Areva PAR FR380 from THAI HR tests (rcombdef(9,*)=19)

The THAI HR tests were performed with an Areva PAR of type FR380. The box was cut and only ½ of the active zone was left inside (Figure 7–7). Thus its capacity has been scaled down to 50%. The constants in the PAR correlation were additionally reduced by 10% to compensate for stronger wall effects and a reduced flow cross section.

The following correlation for hydrogen recombination is implemented for this PAR of type 19:

$$R_{H2}\left[gH2/s\right] = fak * \left(1.37 * p\left[bar\right] + 1.63\right) * conref * \tanh\left(100 * \psi_{H2in} - 0.5\right) \qquad \text{Equ. 7-24}$$

with $conref = \min(\psi_{H2in}, 2*\psi_{O2in}, 0.08)$

and further correction for oxygen starvation with

fak = 1 for $\psi_{H2in} \leq \psi_{O2in}$

fak = 0.6 for $\psi_{H2in} > \psi_{O2in}$

The PAR correlation for type 19 is applied with an average value of the PAR efficiency

$$\eta = (\psi_{H2in} - \psi_{H2out}) / \psi_{H2in} = 0.5 \qquad \text{Equ. 7-25}$$

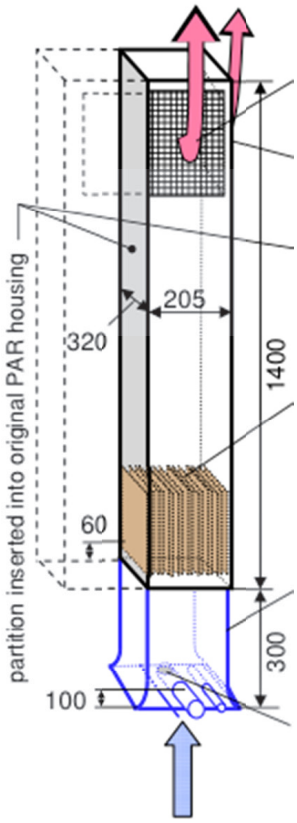


Figure 7-7 Areva 0.5 FR 380 PAR from THAI HR tests

The following sample input demonstrates the use of this correlation for the Areva PAR from the THAI HR experiments. This PAR type is defined by rcombdef(9,*)=19. The PAR cell is bounded by vertical walls on the side and a horizontal wall one mesh above the PAR cell that enforces outflow to the side (see Figure 7-8). Applied is a 2D Cartesian mesh with 4x, 2y and 6 z nodes and continuous inflow of a mixture of air with 5 Vol% hydrogen from the bottom at 10 cm /s. Outflow is defined by a pressure boundary condition at the top defined by a gasdef statement. Default values for this PAR are a startup threshold rcombdef(11,*) of 2 Vol% H_2 and a time constant rcombdef(12,*) of 100s. The measured startup threshold varies in the different THAI tests in a range between 0.8 to 4.2 Vol% hydrogen with the lowest startup value in pure air hydrogen mixtures without steam.

```
$xput
…
 mat              = 'n2', 'o2', 'h2', 'h2o', 'h2ol',
 gasdef(1:18,1)   = 1, 'im1', 1, 'jm1', 1, 'km1', 1, 1.0e6, 325.0, 2, 0.0, 0.0,
                    'n2', 0.7395, 'o2', 0.2105, 'h2', 0.05, ; initial condition
 gasdef(1:18,2)   = 1, 'im1', 1, 'jm1', 0, 1, 1, 1.0e6, 325.0, 2, 0., 1.0e+99,
                    'n2', 0.7395, 'o2', 0.2105, 'h2', 0.05, ; inflow from bottom
 gasdef(1:18,3)   = 1, 'im1', 1, 'jm1', 'km1', 'kmax', 1, 0.99E+6, 325.0, 2, 0., 1.0e+99,
                    'n2', 0.7395, 'o2', 0.2105, 'h2', 0.05, ; top boundary
 pbc(1:9,1)       = 1, 'im1', 1, 'jm1', 'kmax', 'kmax', 1, 0.0, 1.0e+99,
 vbc(1:10,1)      = 1, 'im1', 1, 'jm1', 1, 1, 1, 1, 0.0, 1.0e+99,
 vvalue           = 10.0,
 walls            = 1, 2, 1, 1, 3, 4, 1, 0,  ; par1
                    1, 2, 2, 2, 3, 4, 1, 0,
                    1, 1, 1, 2, 3, 4, 1, 0,
                    2, 2, 1, 2, 3, 4, 1, 0,
                    1, 2, 1, 2, 5, 5, 1, 0,
…
$end
$meshgn
 iblock           = 1,
 xgrid            = 0.0, 100.0, 200.0, 300.0,
 ygrid            = 0.0, 100.0,
 zgrid            = 0.0, 100.0, 200.0, 300.0, 400.0, 500.0,
$end
$rheat
…
 ihtflag            = 1,
 ircomb             = 1,
 rcombdef(1:12,1)   = 1, 2, 1, 2, 3, 4, 1, 1.0, 19, -1, 0.0, 0.0, ; FR380 THAI
…
$end
```

The resulting PAR rates and the cumulated H_2 removal are given in Figure 7–8.

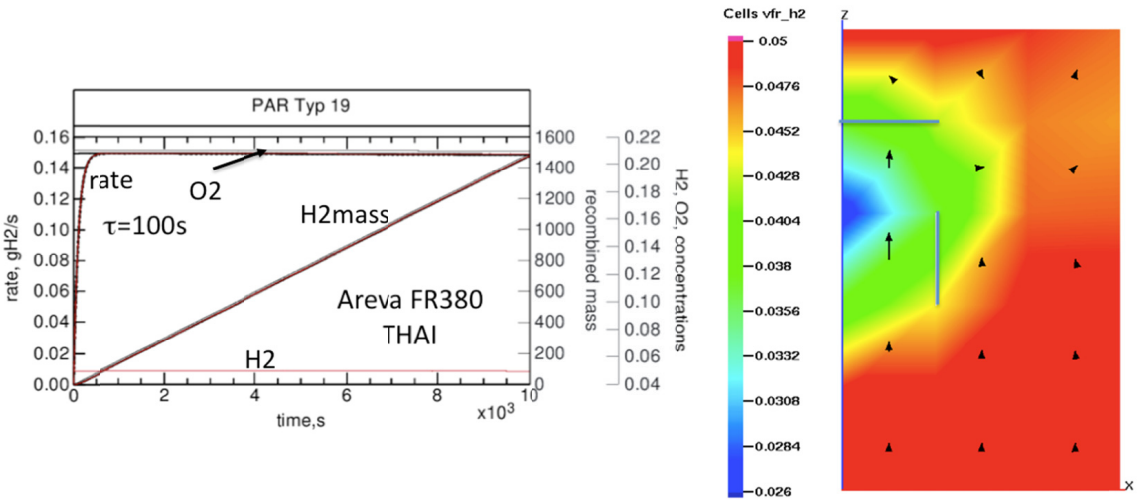


Figure 7-8 GASFLOW calculated PAR rates for FR380 THAI with 5 Vol% H_2 in air and cumulated H_2 removal (red) compared to analytical evaluation (black) using calculated parameters at the PAR inlet. (Analytical evaluation uses the formulas Equ. 7-24 and Equ. 7-25 with the time constant correction from Equ. 7-23. The graph to the right shows the simulated configuration with the PAR box and the local reduction of the H_2 volume fraction inside the box with the corresponding velocity vectors.)

Special Case: PAR box extends over more than one cell

The multiplier rcombdef(8,*) to the recombination effect has now been activated for all box type PARs. Earlier it couldn't effect the Areva PAR types 5 through 9 at all. If the user wants the full effect of the PAR he must always set rcombdef(8,*) to 1.0. In some cases like in the analysis of the THAI HR tests where the geometry of the PAR box extends over more than one computational cell the user may want to split up the PAR effect over several computational cells. The following sample splits up the recombination of the Areva PAR FR380 THAI over 3 computational cells. It could be used in the same way for splitting up the effect of any of the other box type PARs over different cells. These are the minor modifications to xput, meshgn, and rheat to split up the PAR effect from the above example over 3 cells.

```
$xput
...
 walls          = 1, 4, 1, 1, 3, 4, 1, 0,  ; par1
                  1, 4, 2, 2, 3, 4, 1, 0,
                  1, 1, 1, 2, 3, 4, 1, 0,
                  2, 2, 1, 2, 3, 4, 1, 0,
                  3, 3, 1, 2, 3, 4, 1, 0,
                  4, 4, 1, 2, 3, 4, 1, 0,
                  1, 4, 1, 2, 5, 5, 1, 0,
...
$end

$meshgn
...
```

```
 xgrid          =  0.0, 33.33, 66.67, 100.0, 200.0, 300.0,
 ygrid          =  0.0, 100.0,
 zgrid          =  0.0, 100.0, 200.0, 300.0, 400.0, 500.0,
…
$end
$rheat
…
 ircomb             = 1,
 rcombdef(1:12,1)   = 1, 2, 1, 2, 3, 4, 1, 0.3333, 19, -1, 0., 0., ; AREVA THAI PAR
 rcombdef(1:12,2)   = 2, 3, 01, 02, 03, 04, 1, 0.3333, 19, -1, 0., 0., ; AREVA THAI PAR
 rcombdef(1:12,3)   = 3, 4, 01, 02, 03, 04, 1, 0.3333, 19, -1, 0., 0., ; AREVA THAI PAR
…
$end
```

Instead of a single PAR, we split up its effect over 3 smaller PARs of the same type (in this case they are equally sized in area). Undesired mixing and diffusion is prevented by defining separate adiabatic walls around each PAR cell. Each of the newly defined PAR cells is given the fraction of the full PAR effect represented by the ratio of the cell area and the full area over which the single PAR would extend. Instead 1.0 rcombdef(8,*) is thus set to 0.3333 for the 3 subdivided PAR cells, so that the fractions add up to a total of 1.0. Figure 7–9 shows the setup in which the full PAR effect has been split up over 3 cells.

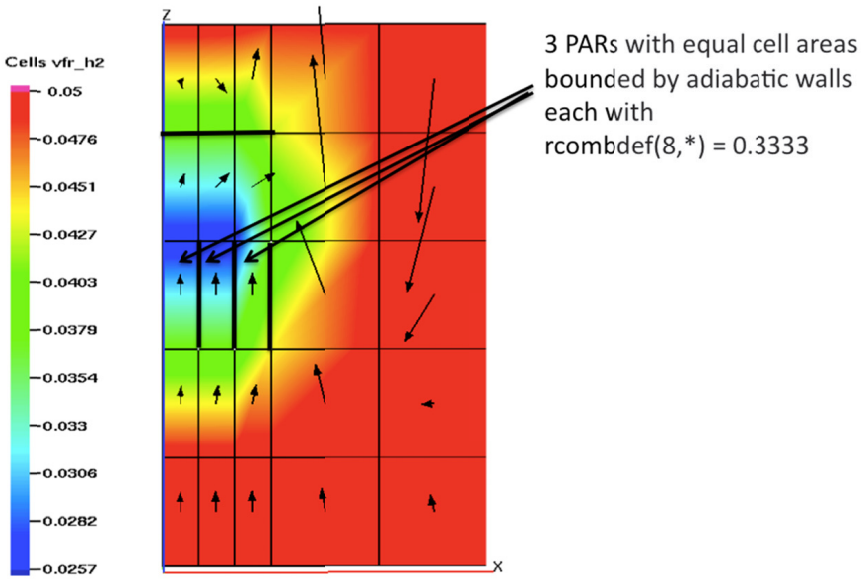


Figure 7-9 Setup for Areva FR380 THAI PAR extending over 3 computational cells

7.7.2.1.3 Updates to the standard Areva PAR Models

The Areva PAR correlations from chapter 2.9 in the theory manual were refined to also account for oxygen starvation. The recombination effect is determined as

$$\dot{m}_{H_2} = fak * (k_1 \cdot p + k_2) \cdot 100 * conref \cdot \tanh \cdot (100 * conref - 0.5)$$ Equ. 7-26

conref = min(vf_{H2},2*vf_{O2},0.08)

where

$\dot{m}_{H_2}$ [g/s]	recombination rate
vf_{H2}	volumetric concentration of hydrogen at PAR inlet
vf_{O2}	volumetric concentration of oxygen at PAR inlet
p [bar]	total pressure
k_1 [g/(s·bar)]	empirically determined recombiner constant
k_2 [g/s]	empirically determined recombiner constant

newly added is the correction for oxygen starvation and the cut off threshold for the recombination effect

fak = factor for oxygen starvation
fak = 1.0 for $vf_{h2} \le vf_{O2}$
fak = 0.6 for $vf_{h2} > vf_{O2}$
fak = 0 for $vf_{h2} < 0.005$ .or. $vf_{O2} < 0.0025$

Table 7-4 Parameters of the standard Areva PAR Types simulated in GASFLOW-MPI

Areva Catalytic Recombiner Constants				approx. inlet x Section [cm2]	GASFLOW PAR Type
Type	Chimney [cm]	k_1 [g/(s×bar)]	k_2 [g/s]		
FR1-150	100	4.800E-03	5.800E-03	1.800E+02	6
FR1-320	100	1.000E-02	1.200E-02	3.840E+02	7
FR1-960	100	3.100E-02	3.700E-02	1.152E+03	8
FR1-380T	140	3.100E-02	3.700E-02	1.312E+03	8
FR1-750T	140	6.100E-02	7.400E-02	2.624E+03	5
FR1-1500T	140	1.370E-01	1.670E-01	4.200E+03	9

A new input parameter etainp (default = 1.0) in $rheat allows the user to specify the PAR efficiency eta=(vfH2in –vfH2out)/vfH2in for the PAR depletion efficiency to be used with the standard correlations for the Areva PARs. Table 7-4 gives the applied parameters for the standard Areva PARs together with the approximate inlet flow cross sections and the heights of the PAR chimneys.

7.7.2.2 Recombination Energy Release into structures

7.7.2.2.1 Catalytic Foils (rcomdef(9,*)=4

As shown in chapter 2.9.4 of the theory manual GASFLOW also allows to release the recombination energy only into a defined structure and from there feed it back to the fluid by convection and radiation heat transfer.

The so called GRS model for catalytic foils (rcombdef(9,*) = 4,plate recombiners) (Figure 7–10) has not been used frequently, but continues to be operational also in the new GASFLOW-MPI release. It considers structure surfaces with special indices for the structure material to recombine hydrogen and oxygen from the fluid node adjacent to such structure. It removes mass and energy of oxygen and hydrogen from the fluid node (ΔH_2, ΔO_2), adds their heat of recombination (242 kJ/mol) to the heat conduction node on the surface of the structure (qrec), and returns 1 mole of steam per removed 1.5 moles of hydrogen and oxygen with the steam energy of the foil surface temperature (ΔH_2O).

1D Heat conduction into the structure and radiative and convective cooling of the structure surface (qrad, qconv) determine the structure surface temperature. In the orthogonal grid of GASFLOW a fluid node can be bounded in the limit by up to 6 different catalytically coated structure surfaces. For the GRS foil model (rcombdef(9,*)=4) allowed structures to recombine hydrogen and oxygen can be walls and slabs. The input example for the foil model from Figure 7–4 defines a wall on the positive side of x node 6 and defines this wall as a recombining surface on both sides by the rcombdef statement which covers the fluid nodes on both sides of the wall.

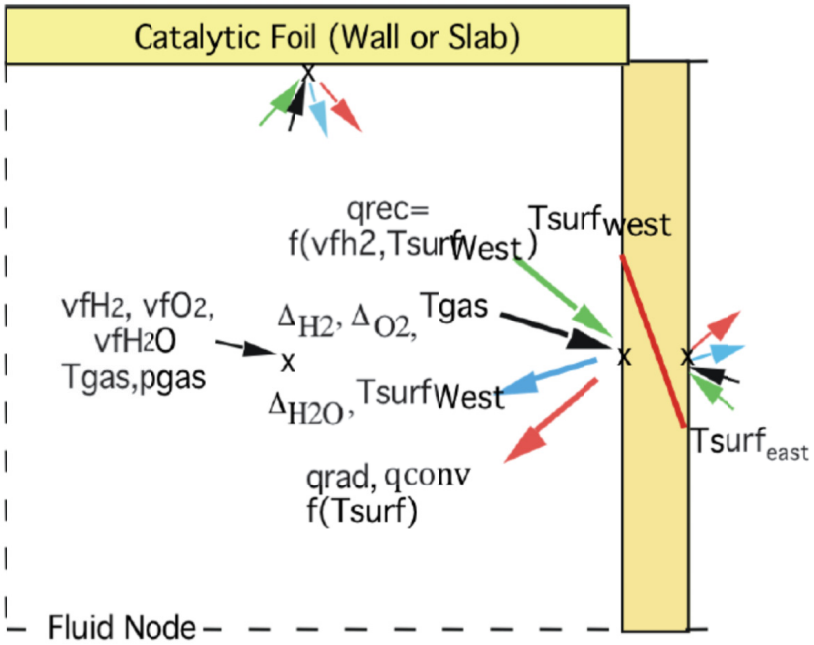


Figure 7-10 Catalytic foil Model for hydrogen recombination (rcombdef(9,*)=4)

```
$xput
…
 walls              =  6, 6, 4, 8, 7, 10, 1, 2,  ;GRS foil model (type 4)
 mobs               =  10, 11, 1, 2, 4, 5, 1, 3,
…
$end
$rheat
…
 rcombdef(1:12,1)   =  5, 7, 4, 8, 3, 4, 1, 1.0, 4, 0, 0, 0, ; GRS foil (type 4)
 rcombdef(1:12, 2)  =  9, 10, 1, 2, 4, 5, 1, 1.0, 4, 0, 0, 0,
 matcomb            =  2, 3, ; recombining materials  (only used for foil type 4)
 walldef            =  2, 0.1, 0.0, 0.0, 0.0,  ; GRS foil material 2
…
$end
```

Restricting the xnode (i-) range in rcombdef above from 5 to 6 would only allow the wall on the positive side of x node 6 to recombine hydrogen. The multiplier rcombdef(8,*) is not applied with the foil model. Setting a second wall = 5, 5, 4, 8, 7, 10, 1, 2, on the negative side of node 6 would make the rcombdef statement to recombine on both wall surfaces that bound the i node 6. The material index matcomb=2 in rheat defines material 2 as catalytic structure. As usual the walldef statement defines the structure properties in this case assuming a foil thickness of 0.1 cm. It is possible to also define selected sides of an obstacle as a recombining surface.

The mobs statement above has a material index 3. The second rcombdef statement would define the negative x side of this obstacle as recombining surface and the additional material number 3 added to the matcomb statement would add this side of the obstacle as a recombining surface.

GASFLOW-MPI simulates molecular and turbulent diffusion of hydrogen, oxygen and steam. In a simple geometry one could directly simulate hydrogen, oxygen diffusion near such recombiner foils also in a first principle approach without experimental correlations. This would require a detailed wall treatment with full resolution of the boundary layer and a consistent description of heat and mass transfer to the catatlytic foil. But currently the recombination rate on the catalytic foil is determined dependent on the hydrogen volume fraction and wall surface temperature from experimental data with a the GRS correlation given in the theory manual.

The foil model has been developed for analysis of the THINCAT concept, which plated component structures like pipings and steam generators with catalytic coatings instead of using box recombiners. The foil model was validated with the HDR foil reco test E11.8.1 and subsequently applied in a scoping analysis for a full reactor containment.

7.7.2.2.2 Ignition Model for Areva PARs

The THAI HR tests recorded the surface temperatures of the recombining catalytic foils inside the FR380 PAR at various locations. Different sensor locations gave essentially the same surface temperatures of the foils. During the tests the recorded foil temperatures often exceeded the auto ignition limit of 773 K. But auto ignition never occurred inside the PAR box in any of the THAI HR

tests. The finding was that auto ignition occurred outside the PAR box when the recorded foil surface temperatures exceeded a threshold of 1220K. Although the mechanisms for this auto ignition are not clearly understood a foil surfaces temperature exceeding a value of 1220 K was identified as a trigger for auto ignition for the Areva PAR tested in the THAI HR experiments.

The current box type PAR models implemented in GASFLOW-MPI all release the recombination energy into the gas only. The flow rate through the box resulting from the PAR correlation and the measured PAR efficiency then determine the increase of the gas temperature inside the box. If the bounding walls of the PAR cell are given as structure properties through walldef statements their cooling effect contributes to the gas temperature also. As in the experiments the calculated gas temperature generally does not exceed the auto ignition limit in the analyzed THAI HR tests with auto ignition. GASFLOW-MPI allows to simulate the PAR foil temperature for the box type Areva PARs.

We have used the option in GASFLOW-MPI to release the recombination energy into structures and have extended the box type PAR models to also predict foil surface temperatures which can be applied as trigger for auto ignition.

The PAR FR380 tested in the THAI HR tests has an array of catalytically coated plates of a certain height with a total active area of 1.44 m2 and a foil thickness of 0.2 mm. The box has an effective flow cross section of 498. 6 cm2. The foil array is located right above the entrance region (see Figure 7–7). The box type PAR models are generally set up and defined in GASFLOW around coarse single fluid cells from a large 3D mesh for the simulation of a full containment geometry. It is not possible to simulate local effects and detailed phenomena for the fluid flow through this cell. By defining distributed heat sinks in GASFLOW-MPI, releasing the recombination energy into the sink and letting the sink being cooled by the convective flow through the box one obtains a foil surface temperature which agrees quite well with experimental data.

With the specific foil surface area and thickness and the concentration dependent flow rate and velocity in the PAR box one can determine representative transient foil temperatures for all Areva PARs at their locations inside the containment mesh. Based on the findings from the THAI HR tests these temperatures can be used as trigger for auto ignition. The sample input models an Areva PAR box recombiner with calculation of the foil surface temperature. To include the simulation of the foil temperature use the sample input for the Areva PAR FR380 given above and define the following additional input parameters.

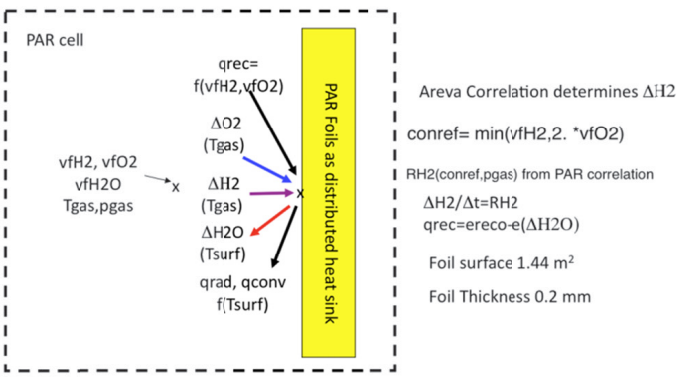


Figure 7-11 Simulation of Foil surface temperature in box type Areva PARs (rcombdef(9,*) =19 or between 5 and 9)

```
$xput
…
 areardef(1:8,1)      = 1, 2, 1, 2, 3, 3, 1, 0.049858,
 areardef(1:8,2)      = 1, 2, 1, 2, 4, 4, 1, 0.049858,
…
$end
$rheat
…
 ircomb               = 1,
 rcombdef(1:12,1)     = 1, 2, 1, 2, 3, 4, 1, 1., 19, -1, 0., 0., ; AREVA THAI PAR matrec = 2
 sinkdef(1:13,1)      = 1, 2, 1, 2, 3, 4, 1, 2, 144.0, 0.02, 0.0, -1.0, 0.0,
…
$end
$grafic
…
 thp                  = 2, 2, 4, 1, 'tk', 0,
 htthp                = 2, 2, 4, 1, 'sink', 0,
…
$end
```

The areardef statements at the upper and lower edge of the PAR cell reduce the PAR cell flow area in the coarse mesh of 10000 cm^2 to the real PAR box flow area 498.6 cm^2 to determine the convective heat transfer with the correct velocity. The sinkdef statement defines a sink with a total foil volume of 144 cm^3, i.e. 14400 cm^2* 0.01 cm (half foil thickness) and a total sink thickness of 0.02 cm and associates it with the material index 2 (steel). An adiabatic boundary condition is applied at the sink centerline. The new input parameter matrec tells the code to release the recombination energy into the defined sink for the PAR cell. The material index specified in the parameter matrec must match the material index of the sinkdef statement to release the PAR energy into the sink. Foil temperatures are evaluated for all box type PARs given the foil surface, foil thickness and the effective flow cross section of the PAR type. of the catalytic foil.

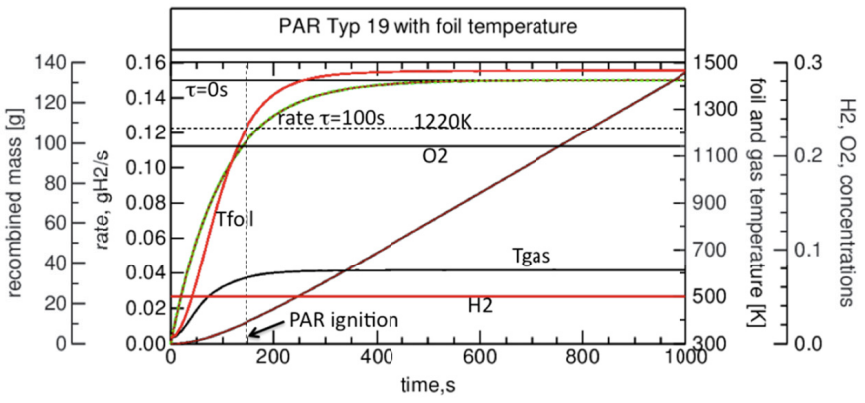


Figure 7-12 GASFLOW calculated PAR rates for FR380 THAI with 5 Vol% H_2 in air, cumulated H_2 removal and catalytic foil and gas temperatures in the PAR cell.

The sensors in the $grafic input block record the gas temperature inside the PAR cell and the foil surface temperature. Figure 7-12 gives the GASFLOW-MPI results for the sample case defined above. The evaluation of the foil temperature doesn't change the overall result of the simulation, the recombination rates are the same as those in Figure 7–8. Due to the low heat capacity of the foils the stationary gas temperatures are the same also. The above sample meets the experimentally determined foil temperature threshold of 1220K which can be used as criterion for PAR ignition at 150 s. Evaluation of the foil surface temperature should be done without radiation cooling. The complex foil arrangement cannot be adequately simulated with the GASFLOW-MPI radiation model and inclusion of radiation heat transfer would result in way too low foil temperatures.

7.7.3 Xenon Decay Model

GASFLOW-MPI has a Xenon gas component, which when activated allows decay energy to be added to the gas mixture in the containment atmosphere as a function of time and space according to the convective behavior of the Xenon gas component.

In the NAMELIST input block XPUT, the following variables have been added:

xecoef Xenon decay coefficient [ergs/(g·m·s)] (Default = 0.0)
xet0 Xenon decay time shift [s] (Default = 0.0)
xetau0 Xenon decay time constant [s] (Default = 1.0)
xepowert Power (-1<=**xepowert**<=1) for the (t-**xet0**) term in Equ. 7-27. Note that the absolute value is actually used in the evaluation of the equation. The reason for this is discussed below. (Default = 0.0).

$$\dot{e}_{Xe-decay}(t)\left\langle\frac{ergs}{g-s}\right\rangle=\mathbf{xecoef}\left\langle\frac{ergs}{g-s^{1+xepowert}}\right\rangle\times \left(t\langle s\rangle-\mathbf{xet0}\langle s\rangle\right)^{|\mathbf{xepowert}|}\cdot\exp\left[-\frac{\left(t\langle s\rangle-\mathbf{xet0}\langle s\rangle\right)}{\mathbf{xetau0}\langle s\rangle}\right] \quad \text{Equ. 7-28}$$

It has been pointed out that specifying the decay heating rate as a specific value, i.e., per mass, as ergs/(g·s) is not always very convenient as most codes like MAAP and MELCOR give the decay heating rate as an total energy rate, ergs/s. Internally in GASFLOW-MPI, we can normalize a function given as an energy rate, ergs/s to a specific energy rate, ergs/(g·s), by simply dividing by the time-dependent Xenon mass in the computational system. The individual cell mass can then be computed and multiplied by the normalized specific energy rate to compute, and what GASFLOW-MPI really needs, the energy source in every computational cell due to decay heating. We make use of **xepowert** as a flag to switch between specific energy rate input and total energy rate input as shown here:

xepowert > 0, for specific decay energy rate input, ergs/(g·s).
< 0, for total decay energy rate input, ergs/s.

Therefore, for **xepowert** < 0, the input function becomes

$$\dot{E}_{Xe-decay}(t)\left\langle\frac{ergs}{s}\right\rangle = \mathbf{xecoef}\left\langle\frac{ergs}{s^{1+xepowert}}\right\rangle\times \left(t\langle s\rangle - \mathbf{xet0}\langle s\rangle\right)^{|\mathbf{xepowert}|}\cdot\exp\left[-\frac{\left(t\langle s\rangle - \mathbf{xet0}\langle s\rangle\right)}{\mathbf{xetau0}\langle s\rangle}\right] \quad \text{Equ. 7-29}$$

It is convenient to allow tabular input to define the decay heating rate, either as a specific energy rate or as a total energy rate. We have added this capability by using **xetau0** as an input flag in the following fashion:

xetau0 > 0, for function input.
< 0, for tabular input.

When **xetau0** < 0, the tabular input is accomplished through the **xecoef** input variable. For the function input, **xecoef** is a scalar; however, when **xetau0** < 0 defines tabular input, **xecoef** is used as an array input dimensioned of size 50, where the odd elements represent the decay heating rate, either specific, ergs/(g·s), or total, ergs/s, as described above, and the even elements represent time, s. We will give an example for each of these decay heating input schemes, but before that, we summarize the 4 methods in the following Table.

Table 7-5 Possible methods for inputing the decay heating rate in GASFLOW-MPI

Decay Heating Input Method	xepowert	xetau0	xecoef
Function ergs/s	< 0	> 0	Scalar
Function ergs/(g·s)	>= 0	> 0	Scalar
Tabular ergs/s	< 0	< 0	Array
Tabular ergs/(g·s)	>= 0	< 0	Array

Demonstration Examples:

Example A. Tabular total decay energy rate, ergs/s, input:

The NAMELIST XPUT input stream is given here:

xetau0 = -5.68828e+03, ; Tabular
xepowert = -0.267, ; (ergs/s) input
xecoef ; VALUE TIME
= 0000.000e+10, 0.0000e+03,
0170.650e+10, 0.2010e+03,
1685.150e+10, 1.5210e+03,
2062.000e+10, 2.0010e+03,
1239.500e+10, 3.8010e+03,
0533.490e+10, 1.8200e+04,

This tabular function is shown in Figure 7-13.

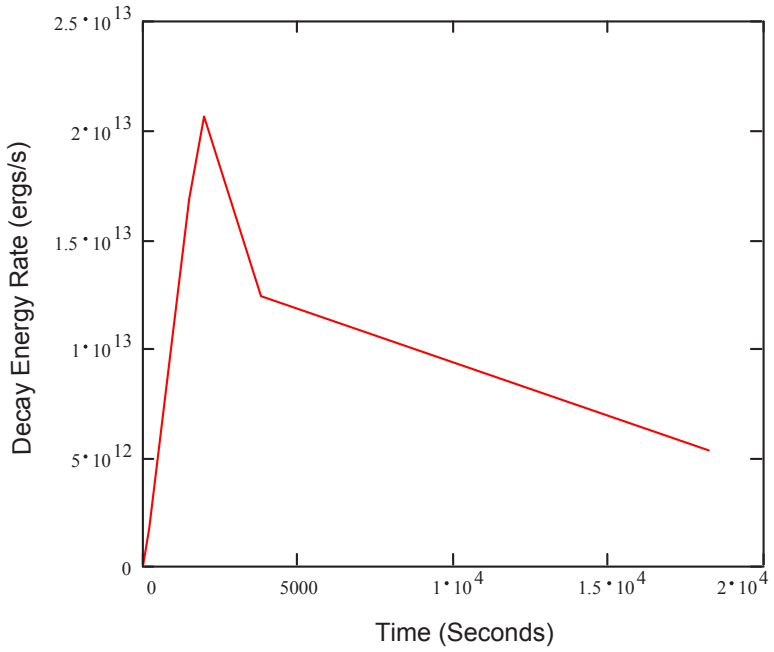


Figure 7-13 .Example of tabular total decay energy rate as listed in Example A.

Example B. Tabular specific decay energy rate, ergs/(g·s), input:

Consider the time dependent Xenon mass in the system as shown in Figure 7-14. Dividing the tabular input of Example A by this Xenon mass data yields the specific decay energy rate results shown in Figure 7-15, and presented as the NAMELIST XPUT input stream given here.

```
xetau0      = -5.68828e+03,  ;Tabular
xepowert    = +0.267,        ;(ergs/g·s) input
xecoef        VALUE      TIME
            = 0000.000e+04,  0.0000e+03,
              2171.000e+04,  0.2010e+03,
              2832.000e+04,  1.5210e+03,
              2635.000e+04,  2.0010e+03,
              1584.000e+04,  3.8010e+03,
              0682.000e+04,  1.8200e+04,
```

This tabular data is presented below in Figure 7-15.

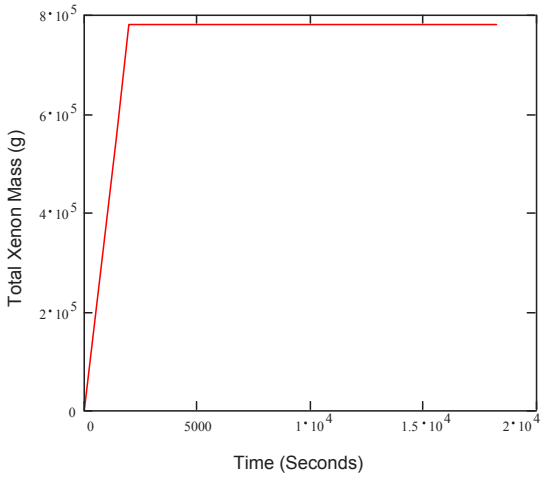


Figure 7-14 Xenon mass in the system for Example B.

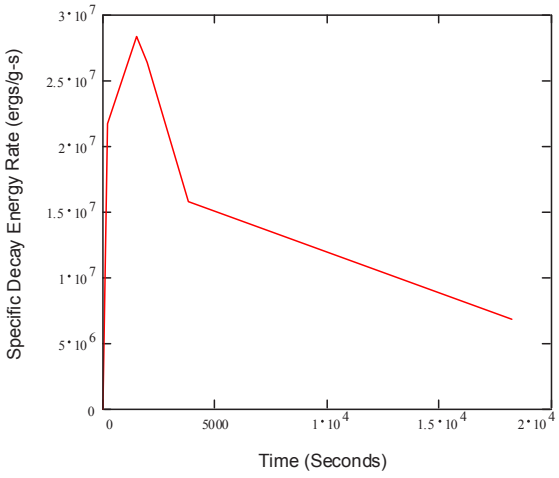


Figure 7-15 Example of tabular specific decay energy rate as listed in Example B.

Example C. Function of the total decay energy rate,ergs/s input:

The NAMELIST XPUT input stream is give here:

xetau0 = +4.64684e+03, ; Defined function
xepowert = -0.431, ; (ergs/s) input
xecoef = 120.214e+10, ; Coefficient
xet0 = 0.0e+00, ; Time shift

This function is presented below in Figure 7-16.

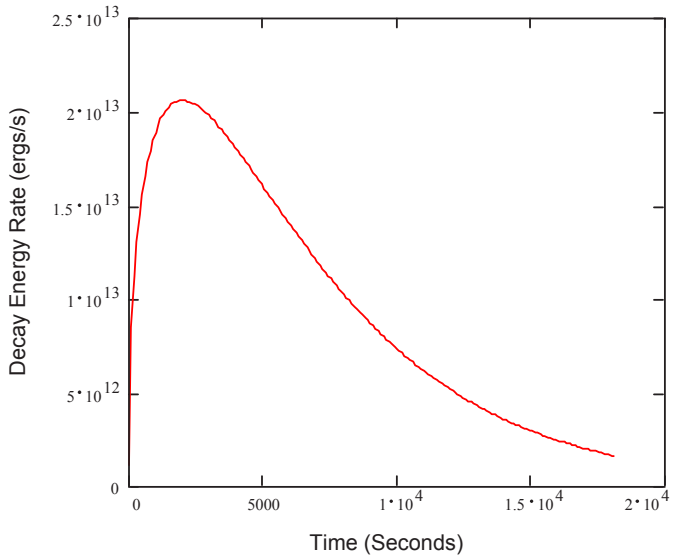


Figure 7-16 Example of input function for the total decay energy rate as listed in Example C.

One can now present on the same display the tabular data, Figure 7–13 and the functional approximation, Figure 7–16, on the same plot as shown in Figure 7–17.

The three unknowns in the function, Equ. 7-29, are, for example, derived from the tabular data in the following manner:

1. We have a maximum of the function at $t_{\max}$ where

$\dot{E}_{\max} = \dot{E}_{Xedecay,\max}\left(t_{\max}\right)$, or

$$xecoef \cdot \left(t_{\max} - xet0\right)^{|xepowert|} \cdot \exp\left[-\frac{\left(t_{\max} - xet0\right)}{xetau0}\right] = \dot{E}_{\max}$$

2. At $t_{\max}$ we impose a zero slope as well, i.e.,

$\frac{d}{dt}\dot{E}_{Xedecay}\left(t = t_{\max}\right) = 0$, or

$$t_{\max} - xet0 - xepowert \cdot xetau0 = 0$$

3. The time integration, total energy, of both tabular and functional input should be equal

$$\int_0^{t_{end}} \dot{E}_{Xedecay}\left(t\right)dt = \frac{1}{2}\sum_i \left(\dot{E}_i + \dot{E}_{i-1}\right)\cdot\left(t_i - t_{i-1}\right)$$

where the subscript denotes the i[th] interval of the tabular data.

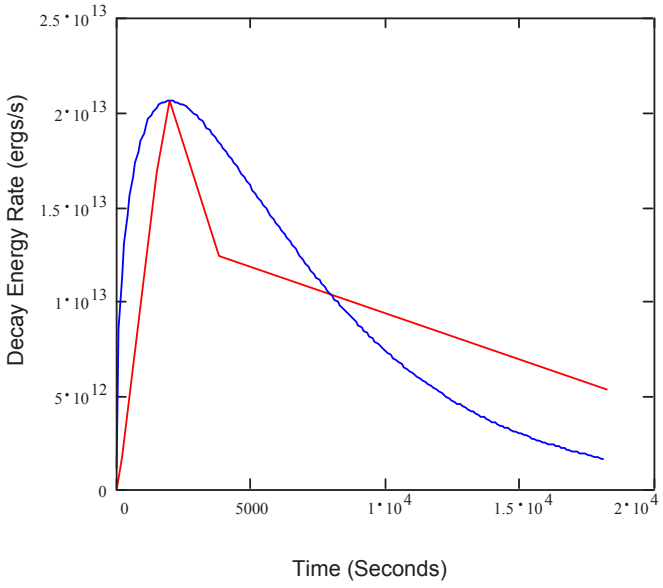


Figure 7-17 Example A using tabular input and Example C using the function approximation.

Example D: Function of the total decay energy rate,ergs/(g·s) , input: The NAMELIST XPUT input stream is give here:

xetau0 = +5.68828e+03, ; Defined function
xepowert = +0.267, ; (ergs/g-s) input
xecoef = 521.0e+04, ; Coefficient
xet0 = 0.0e+0, ; Time shift

This function is presented below in Figure 7–18.

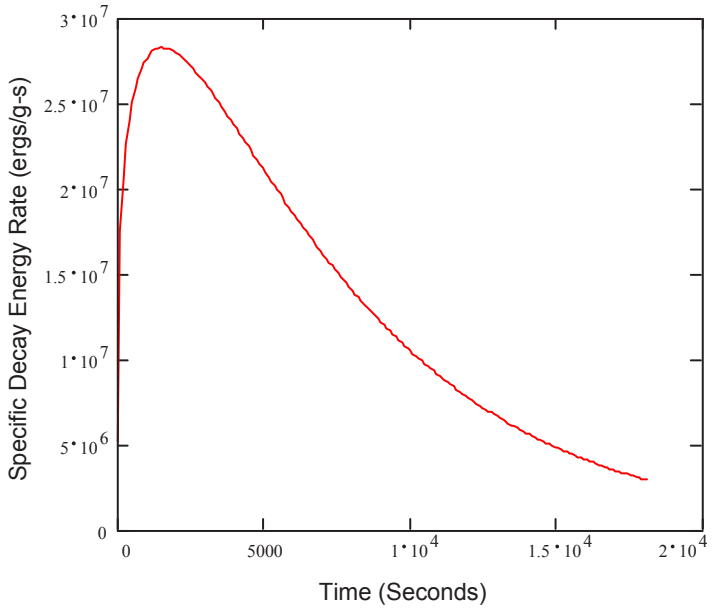


Figure 7-18 Example of input function for the specific decay energy rate as listed in Example D.

We can present the tabular data, Figure 7–15, and the functional approximation, Figure 7–18, on the same plot as shown in Figure 7–19. The 3 unknowns in the function, Equ. 7-28, are derived from the tabular data in a similar manner as before.

There is no real reason to use the functional relationships, Equ. 7-28 and Equ. 7-29 for describing the decay heating behavior in the gas phase. Usually data is provided in tabular form, and it is convenient to use the data as provided and not develop a functional approximation. The only exceptions would be if data are not provided and the user has limited knowledge of time dependencies or, of course, if the function is provided as a starting point.

When the tabular option is used, the array **xecoef** is currently limited to 50 elements, i.e., 25 pairs of decay heating energy rate and time. Extrapolation of the tabular data is not allowed, so the last time interval of the tabular data must at least bound the problem time, **twfin**.

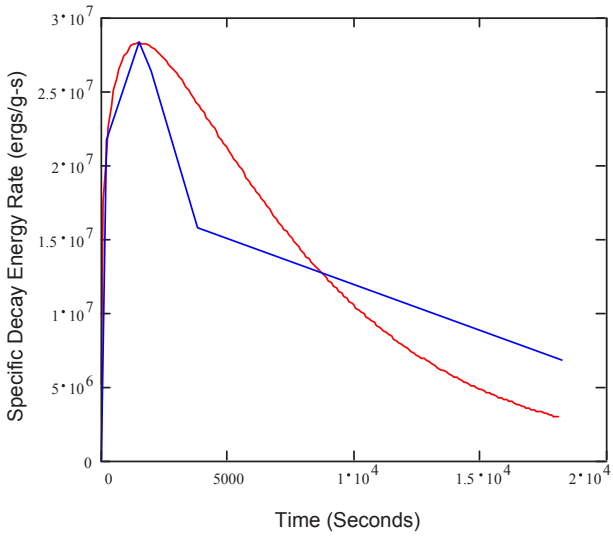


Figure 7-19 Example B using tabular input and Example D using the function approximation.

7.8 Generalized Fan Model

A generalized fan model maybe defined as a momentum source whose location coincides with its respective velocity component. To activate the generalized fan options, one must place fans within the 3-dimensional mesh by using the **fandef** statement to define the n^{th} fan within the NAMELIST XPUT:

fandef(1,n) Beginning i mesh index (cell face number).
fandef(2,n) Ending i mesh index (cell face number).
fandef(3,n) Beginning j mesh index (cell face number).
fandef(4,n) Ending j mesh index (cell face number).
fandef(5,n) Beginning k mesh index (cell face number).
fandef(6,n) Ending k mesh index (cell face number).
fandef(7,n) Block number (must be 1 for GASFLOW-MPI).
fandef(8,n) ITFAN table number for fan head versus volumetric flow (see **fantb** below).
Maximum number allowed is 20.
fandef(9,n) ITSPD table number for fan speed versus time (see **fspdtb** below)
Maximum number allowed is 20.
fandef(10,n) SPDR, rated fan blower speed, revolutions/s.
fandef(11,n) QR, rated fan volumetric flow rate, cm^3/s.
> 0, fan is directed in the positive coordinate direction.
< 0, fan is directed in the negative coordinate direction.
fandef(12,n) HR, rated fan head
> 0, fan table (**fantb**) is in $dynes/cm^2$.
< 0, fan table (**fantb**) is in cm H_2O.
fandef(13,n) SPD, fan speed at time = 0.0, revolutions/s.

There is a current limitation of 600 **fandef** statements.

There must be additional information to define the fan characteristics (pressure change as a function of volumetric flow rate), and to define the fan performance (fan speed as a function of time). These are provided to GASFLOW-MPI in the form of two tables containing paired values.

The fan characteristics table, which is usually provided by the fan manufacture, is a table of paired values in the NAMELIST XPUT for the array **fantb(1:2,maxp,maxtb)**, where maxp is limited to 30 paired values and maxtb is limited to 20 tables. The input is defined:

fantb(1,i,j) Volumetric flow of table pair i for table number j, cm^3/s.
fantb(2,i,j) Fan head of table pair i for table j.

The fan performance table, which is problem dependent, is provided to GASFLOW with a table of paired values in the NAMELIST XPUT for the array **fspdtb(1:2,maxp,maxtb)**, where again maxp is limited to 30 paired values and maxtb is limited to 20 tables. The input is defined:

fspdtb(1,i,j) Time of table pair i for table number j, s.
fspdtb(2,i,j) Fan speed of table pair i for table j, revolutions/s.

In the following example we specify two fans to fully operate for the first 40 seconds with then the first of the fans to fail and not be active beyond 40 seconds.

```
$xput
…
 fandef(1:13,1)     = 25, 25, 43, 46, 5, 8, 1, 1, 2, 2603.0, 2.469e+05, 4000.0, 2603.0,
 fandef(1:13,2)     = 25, 25, 43, 46, 10, 13, 1, 1, 1, 2603.0, 2.469e+05, 4000.0, 2603.0,
 fantb(1:2, 1,1)    = 0.000e+00,  5.8300e+03,
 fantb(1:2, 2,1)    = 3.740e+04,  5.5000e+03,
 fantb(1:2, 3,1)    = 7.516e+04,  5.3000e+03,
 fantb(1:2, 4,1)    = 1.117e+05,  5.1000e+03,
 fantb(1:2, 5,1)    = 1.473e+05,  4.9000e+03,
 fantb(1:2, 6,1)    = 2.012e+05,  4.5000e+03,
 fantb(1:2, 7,1)    = 2.469e+05,  4.0000e+03,
 fantb(1:2, 8,1)    = 2.818e+05,  3.5000e+03,
 fantb(1:2, 9,1)    = 3.114e+05,  2.9990e+03,
 fantb(1:2, 10,1)   = 3.657e+05,  1.9990e+03,
 fantb(1:2, 11,1)   = 3.910e+05,  1.4990e+03,
 fantb(1:2, 12,1)   = 4.131e+05,  0.9990e+03,
 fantb(1:2, 13,1)   = 4.349e+05,  0.4989e+03,
 fantb(1:2, 14,1)   = 4.551e+05,  0.000,
 fspdtb(1:2,1,1)    = 0.0e+0,   2603.0,
 fspdtb(1:2,2,1)    = 4.0e+1,   2603.0,
 fspdtb(1:2,3,1)    = 1.0e+5,   2603.0,
 fspdtb(1:2,1,2)    = 0.0e+0,   2603.0,
 fspdtb(1:2,2,2)    = 4.0e+1,   2603.0,
```

fspdtb(1:2,3,2) = 4.1e+1, 0000.0,
fspdtb(1:2,4,2) = 4.1e+9, 0000.0,
...
$end

7.9 Generalized Energy Source Term Model

We have generalized and expanded the existing energy source input, **esdef**, to a maximum of 500 NAMELIST XPUT statements and changed the actual input energy units of power, i.e., ergs/s where 1 Watt = 10^7ergs/s. This NAMELIST XPUT statements are now defined as follows:

esdef(1,*) Beginning i mesh index (cell face number).
esdef(2,*) Ending i mesh index (cell face number).
esdef(3,*) Beginning j mesh index (cell face number).
esdef(4,*) Ending j mesh index (cell face number).
esdef(5,*) Beginning k mesh index (cell face number).
esdef(6,*) Ending k mesh index (cell face number).
esdef(7,*) Block number (must be 1 for GASFLOW-MPI).
esdef(8,*) Total power in the computational volume defined by the first 7 elements of this array, ergs/s.
esdef(9,*) Time (s) at which "**esdef**" begins.
esdef(10,*) Time (s) at which "**esdef**" ends.

In the way of an example, we add statements to the NAMELIST XPUT of the above example (example from Section 7.8), to demonstrate how one can easily model fans in, for example CPU Boxes with heat sources:

fandef(1:13, 3) = 19, 19, 09, 10, 04, 06, 1, 2, 3, 3300.0, +15.00e+03, 0000.0, 3300.0,
; Box exit fans
fandef(1:13, 4) = 09, 09, 09, 10, 07, 08, 1, 2, 3, 3300.0, +15.00e+03, 0000.0, 3300.0,
; Box internal fans
fandef(1:13, 5) = 09, 09, 09, 10, 09, 10, 1, 2, 3, 3300.0, +15.00e+03, 0000.0, 3300.0,
; Box internal fans
fantb(1:2, 1,2) = 0.000e+00, 6.0000e+02,
fantb(1:2, 2,2) = 2.500e+03, 5.2000e+02,
fantb(1:2, 3,2) = 5.000e+03, 3.7500e+02,
fantb(1:2, 4,2) = 7.500e+03, 2.8000e+02,
fantb(1:2, 5,2) = 10.000e+03, 2.4000e+02,
fantb(1:2, 6,2) = 12.500e+03, 1.8000e+02,
fantb(1:2, 7,2) = 15.000e+03, 0.0000e+00,
fspdtb(1:2,1,3) = 000.0e+0, 3300.0,
fspdtb(1:2,2,3) = 004.0e+1, 3300.0,
fspdtb(1:2,3,3) = 001.0e+5, 3300.0,

and the power deposited in the gas from each CPU (120 W each), the CPU power supply (10 W), and the motherboard power (40 W):

esdef(1:10,1) = 10, 11, 09, 10, 07, 08, 1, 1.20e+9, 5.0, 1.0e+10, ; right side CPU
esdef(1:10,2) = 10, 11, 09, 10, 09, 10, 1, 1.20e+9, 5.0, 1.0e+10, ; left side CPU
esdef(1:10,3) = 13, 17, 09, 10, 04, 06, 1, 1.00e+8, 5.0, 1.0e+10, ; CPU power supply
esdef(1:10,4) = 13, 18, 09, 10, 07, 12, 1, 3.00e+8, 5.0, 1.0e+10, ; Motherboard

7.10 Spray Model

Warning: The GASFLOW-MPI Spray model is experimental and under development as time permits; and therefore, should be used with extreme caution.

GASFLOW-II solves the classical two-phase Homogenous Equilibrium Model (HEM) with the assumption that the liquid droplets are dispersed in a gaseous medium. This means that there is thermal and mechanical equilibrium (equal temperatures and velocities, respectively) between the phases and that the volume fraction of the gaseous components is close to unity.

The first version of the GASFLOW-II spray model assumes that there is still mechanical equilibrium between the phases (equal velocities) but non-equilibrium temperatures. This requires the solution of a specific internal energy equation for the liquid droplet phase where the gas temperature is now computed by subtracting the liquid droplet energy from the total energy equation and inverting this relationship for the gas temperature and also inverting the specific internal energy function to obtain the liquid temperature.

The pressure field is determined from the gaseous components only. Convective heat transfer and phase change mass transfer is modeled between liquid and vapor components to obtain the appropriate coupling phenomena.

Droplet depletion is also modeled to provide the effect of liquid sinks from the fluid field seen in droplet sedimentation or rainout and with a mechanistic impaction model.

7.10.1 Activation of the GASFLOW II Spray model

ispray = 1, ; Spray model activated

must occur in the **xput** input stream.

7.10.2 Some current restrictions

1. The liquid component must be last in the **mat** input list: **mat** = 'air', 'h2o', 'h2ol',
2. There must be some liquid "seed" in all **gasdef** lists:

gasdef(1:18,1) = 1,'im1', 1, 'jm1', 1, 'km1', 1, 1.015e+06, 300.00, 2, 0.0, 0.0,
'air', 0.999999, 'h2o', 0.0, 'h2ol', 0.000001,

7.10.3 General Spray input in the "xput" input stream

1. Droplet Advection Algorithm Options:
   Impsprayd = 0 ; Explicit droplet advection algorithm (default)
   Impsprayd = 1 ; Implicit droplet advection algorithm

2. When using a gasdef input variable to define droplet temperatures and diameters, the user can use the **spraygdef** input array with the standard **gasdef** input array to specify droplet temperature and average diameter:

   gasdef(1:18,1) = 1,'im1', 01,'jm1', 1, 'km1', 1, 1.015e+06, 300.00, 2, 0.0, 0.0,
   'air', 0.999999, 'h2o', 0.0, 'h2ol', 0.000001,
   spraygdef(1:2,1) = 300.0, 0.00001,

   The above Initial Condition example sets both the gas and liquid phases at 300 K with the droplet seed having an initial diameter equaling 0.1 microns.

   gasdef(1:18,2) = 6, 7, 6, 7, 0, 1, 1, 1.015e+06, 600.0, 2, 0.0, 50.0,
   'air', 0.0, 'h2o', 0.999, 'h2ol', 0.001,
   spraygdef(1:2,2) = 300.0, 0.1,

   In this example, a typical boundary condition active from 0.0 < time < 50.0 s, the gas is at 600 K while the liquid droplets are at 300 K with a diameter of 1 mm.

   There is a maximum of 500 spraygdef's possible, i.e., **spraygdef**(1:2,500) is the maximum allowed.

3. The user can define a liquid spray source any place in the computational mesh (for example, to locate a spray head or nozzle) by using the **spraydef** input arrays:

   spraydef(1:12,1) = 3, 4, 6, 7, 9, 10, 1, 2.0e+04, 300.0, 1.0e-02, 50.0, 99999.0,
   spraydef(1:12,2) = 9, 10, 6, 7, 9, 10, 1, 2.0e+04, 300.0, 1.0e-02, 50.0, 99999.0,
   spraydef(1:12,3) = 6, 7, 3, 4, 9, 10, 1, 2.0e+04, 300.0, 1.0e-02, 50.0, 99999.0,
   spraydef(1:12,4) = 6, 7, 9, 10, 9, 10, 1, 2.0e+04, 300.0, 1.0e-02, 50.0, 99999.0,

   In this example, there are 4 locations where spray heads are specified. Note that the first 7 entries are the same notation as for nearly all GASFLOW-MPI mesh wide input arrays, while entries 8 -> 12 are respectively, Spray mass flow rate (g/s), Droplet Temperature (K), Droplet Diameter (cm), Time of Activation ($Time_1$), and Time of Termination ($Time_2$).

   The action of the sprays, coupling the above **spraydef** input with the traditional **mbc** and **mvalue** input arrays, is shown here with the following example:

   mbc = 3, 4, 6, 7, 9, 9, 1, 1, 50.0, 99999.0,
   9, 10, 6, 7, 9, 9, 1, 2, 50.0, 99999.0,
   6, 7, 3, 4, 9, 9, 1, 3, 50.0, 99999.0,
   6, 7, 9, 10, 9, 9, 1, 4, 50.0, 99999.0,

mvalue = -2.000e+04, -2.000e+04,
-2.000e+04, -2.000e+04,

There is a maximum of 100 **spraydef**'s possible, i.e., **spraydef**(1:12,100) is the maximum allowed.

4. Another example, a single headed spray, with a time-dependent temperature could be as follows:

spraydef(1:12,1) = 1, 2, 1, 2, 84, 85, 1, 3.0, -152.0, 0.013, 0.0, 99999.0,
ttab(1:2, 1,1) = 0.0, 404.25,
ttab(1:2, 2,1) = 100.0, 404.25,
ttab(1:2, 3,1) = 311.0, 295.25,
ttab(1:2, 4,1) = 1000.0, 300.85,
ttab(1:2, 5,1) = 9999.0, 300.85,

In this example, we make use of the **ttab** option to specify a time dependent spray temperature.

7.10.4 Mechanistic droplet impaction model

This model can be activated with the input variable in the $RHEAT NAMELIST group **iliq**.

iliq	Description
1	(default) activates the parametric model.
-1	activates this mechanistic model.
-2	activates both parametric and mechanistic models.

7.10.5 General Spray input in the "grafic" input stream

We have added additional plotting capability when the spray model is active. The time-history plots (**thp**), the one-dimensional plots (**p1d**), and the two-dimensional contour plots (**c2d**) are all able to plot

1. 'td' -> the droplet temperature
2. 'tv' -> the vapor temperature
3. 'sied' -> the droplet specific internal energy
4. 'diad' -> the droplet average diameter

For example, we can plot the new time-history variables in the following manner:

thp = 7, 7, 5, 1, 'td', 0,
7, 7, 5, 1, 'tv', 0,
7, 7, 5, 1, 'sied', 0,
7, 7, 5, 1, 'diad', 0,

and then define points within the computational mesh

```
pnt        =   1, 7, 1, 1,
              'im1', 7, 'km1', 1,
                7, 1, 1, 1,
                7, 'jm1', 'km1', 1,
                7, 7, 1, 1,
                7, 7, 'km1', 1,
```

to plot one-dimensional profiles as:

```
p1d        = 5, 6, 'wn', 0,
             5, 6, 'sien', 0,
             5, 6, 'tk', 0,
             5, 6, 'pn', 0,
             5, 6, 'rsn', 'h2ol',
             5, 6, 'rsn', 'h2o',
             5, 6, 'td', 0,
             5, 6, 'tv', 0,
             5, 6, 'diad', 0,
```

and two-dimensional contours as:

```
c2d        = 1, 2, 'tk', 0,
             1, 2, 'rsn', 'h2o',
             1, 2, 'rsn', 'h2ol',
             3, 4, 'tk', 0,
             3, 4, 'rsn', 'h2o',
             3, 4, 'rsn', 'h2ol',
             1, 2, 'diad', 0,
             1, 2, 'sied', 0,
             1, 2, 'td', 0,
             1, 2, 'tv', 0,
             3, 4, 'td', 0,
             3, 4, 'tv', 0,
             3, 4, 'sied', 0,
             3, 4, 'diad', 0,
```

8 Options on Numerical Solution Procedure

8.1 Pressure Iteration

The numerical algorithm used in GASFLOW-MPI 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 linear solvers in PETSc library. The algorithm is constructed such that the true solution is obtained after a finite number of iterations. 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 GASFLOW-MPI, 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. However, to allow flexibility in controlling the iteration procedure for a wide range of problems, the code provides the following input variable in NAMELIST group xput:

epsi0 Initial value of ε. Default = 1×10^{-5}.

In GASFLOW serial version, the user can 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:

itmaxMaximum number of iterations (per time cycle) allowed. Default = 1000.

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 1 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	CODE	IBLK	i	j	k
1.000E-02	1	33	1.000E-02	8.543E-06	1.000E-05	MX	1	1	1	1
2.000E-02	2	48	1.000E-02	6.030E-06	1.000E-05	MX	1	1	1	1
3.000E-02	3	45	1.000E-02	7.538E-06	1.000E-05	MX	1	1	1	1
4.000E-02	4	39	1.000E-02	7.900E-06	1.000E-05	MX	1	1	1	1
5.000E-02	5	41	1.000E-02	9.453E-06	1.000E-05	MX	1	1	1	1
6.000E-02	6	40	1.000E-02	8.151E-06	1.000E-05	MX	1	1	1	1
7.000E-02	7	40	1.000E-02	6.678E-06	1.000E-05	MX	1	1	1	1
8.000E-02	8	39	1.000E-02	8.235E-06	1.000E-05	MX	1	1	1	1
9.000E-02	9	38	1.000E-02	9.919E-06	1.000E-05	MX	1	1	1	1
1.000E-01	10	39	1.000E-02	8.282E-06	1.000E-05	MX	1	1	1	1

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 with DMAX, which represents the largest component in the residual vector **r**. In this example, convergence has been achieved in every cycle. If the pressure iteration does not converge but the maximum error is still relatively large, the calculation will stop and the run will have to be repeated with a smaller time-step size. The printout under CODE provides the user with information on what is controlling the time-step size. An explanation for the CODE output is given below:

CU The fluid Courant limit based on the *x*-direction velocity is controlling the time-step size.
CV The fluid Courant limit based on the *y*-direction velocity is controlling the time-step size.
CW The fluid Courant limit based on the *z*-direction velocity is controlling the time-step size.
DA Energy diffusion is controlling the time-step size.
DD Species mass diffusion is controlling the time-step size.
DN Momentum diffusion is controlling the time-step size.
IG Energy source term is controlling the time-step size.
IN Indicates either the initial time-step size or that the user has fixed the time-step size with **autot** = 0.0.
IT Pressure iteration is controlling the time-step size.
MX Time-step size is at **deltmax**.
NA Nothing controlling the time-step size.
PJ The particle injection time-step size is controlling the time-step size.
PU The particle velocity in the *x*-direction is controlling the time-step size.
PV The particle velocity in the *y*-direction is controlling the time-step size.
PW The particle velocity in the *z*-direction is controlling the time-step size.

The cell controlling the time-step size is identified under the IBLK, I, J, K headings.

8.2 Time-Step Control

Because GASFLOW-MPI 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. GASFLOW has an algorithm for adjusting the time increment during a calculation, or the user can force the code to use a fixed time-step size.

GASFLOW-MPI does however allow the user to dynamically specify the maximum time step, **deltmax**, by making use of the fact that **deltmax** is an array dimensioned 20. When there is only one entry in **deltmax**, it is treated as a scalar constant, i.e., a single maximum time step value, but when there are multiple entries, they are associated in pairs with the odd elements representing the time interval or time step and the even elements representing the upper time limit for that particular time interval or time step. If the current time exceeds the last time specified in the array **deltmax**, the last maximum time step size will be held for the remaining simulation time up to **twfin**.

An example is shown here for the maximum time step, **deltmax**, as it is read in the NAMELIST XPUT:

```
$xput
…
                ; VALUE    TIME
deltmax         =  0.5,   10000.1,
                   0.6,   20000.1,
                   0.7,   30000.1,
                   0.8,   40000.1,
                   0.5,   50000.1,
                   0.6,   60000.1,
                   0.75,  70000.1,
                   1.0,   90000.0,
…
$end
```

For this example, the maximum time step is held constant at 0.5 s for (0 <= time < 10000), 0.6 s for (0 <= time < 20000), and so on.

Figure 8-1 shows the maximum time step behavior for this input specification.

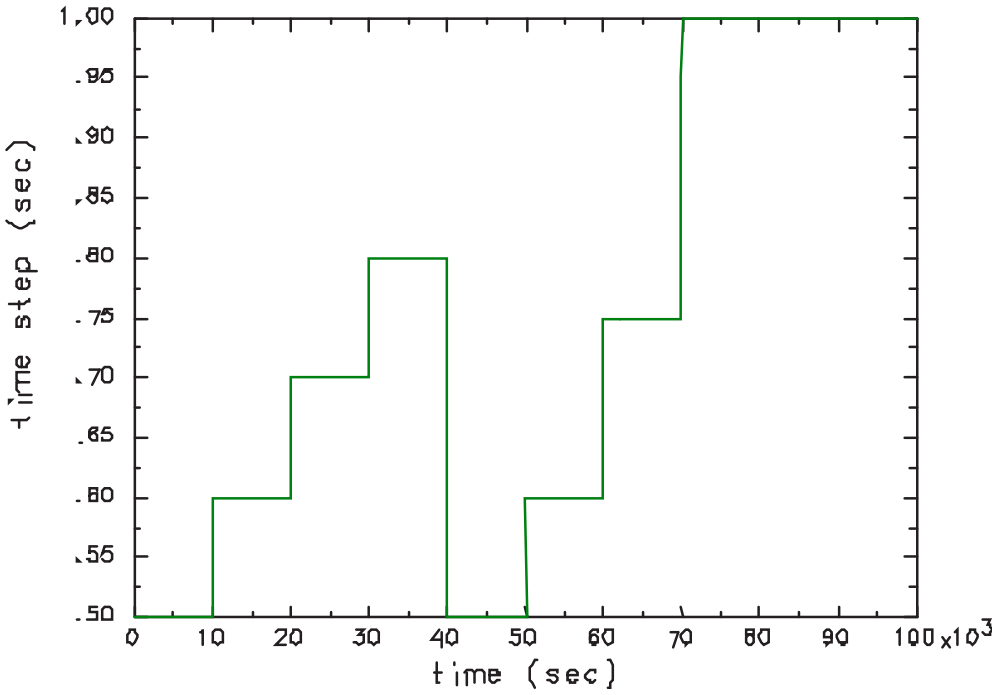


Figure 8-1 Maximum time step shown from this example.

Note that we've added a small amount of time, 0.1 seconds, to all the even entries in **deltmax**. This insures that the maximum time step allowed is updated at the requested time. See Section 8.4, Control of Time Interval Variables, for more information.

The numerical method used in GASFLOW-MPI 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 Δ*t* satisfies both the Courant criterion and the diffusion limit. To ensure numerical stability, the code limits Δ*t* as follows:

$$\Delta t \le \text{cflnum} \times \min\left(\frac{\Delta_i}{u_i}, \frac{\Delta_j}{u_j}, \frac{\Delta_k}{u_k}\right) \text{(Courant Limit)} \qquad \text{Equ. 8-1}$$

$$\Delta t \le 0.9\left(\frac{\Delta_{\min}^2}{6\,\nu_{\text{eff}}}\right) \quad \text{(Diffusion Limit)} \qquad \text{Equ. 8-2}$$

For the Courant limit, Δ_j and u_j, etc., are local cell spacing and velocity in all three directions. In GASFLOW-MPI, we have the option for users to define the maximum allowed CFL number using **cflnum**. The default value of **cflnum** is 0.25 which is also used in GASFLOW serial version. The user could increase **cflnum (< 0.9)** to allow a bigger time step. But be aware that too big time step may cause numerical instability. In this case, the user should reduce the **cflnum**. For the diffusion limit, ν_{eff} is the effective diffusivity in a cell, and $\Delta_{\min}$ is the smallest dimension of that cell. The limit is applied, of course, only if the diffusion option is turned on (**idiffmom** and/or **idiffme** 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. GASFLOW serial version by default will try to adjust Δ*t* to achieve maximum efficiency while satisfying the stability limits. The code increases or decreases Δ*t* 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 = 800), then Δ*t* will be decreased by 2%. If the iteration number is less than **itupdt** (default = 800), then Δ*t* will be increased by 2% in the next cycle. Since in GASFLOW-MPI the number of pressure iterations required for convergence may vary with various number of processes, the calculation results could be slightly different by using **itdowndt** and **itupdt**. Therefore, in GASFLOW-MPI we will not use these artifcial input variables to control the time step.

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 Δ*t* :
1.0 means Δ*t* is adjusted by code during calculation (default);
0.0 means the input **delt0** is used for Δ*t* throughout calculation.

8.3 Advection Scheme

Each of the conservation equations solved by GASFLOW-MPI contains a convective flux term $\nabla \cdot (\phi \mathbf{u})$, where ϕ is the conserved quantity (mass density, internal energy, or momentum) and **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. 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.

8.4 Control of Time Interval Variables.

There are certain input array variables that provide time-dependent control of the maximum time step, **deltmax**, plotting time **pltdt**, printing time, **prtdt**, restart dump and 3D visualization time, **tddt**, and plotting time history time, **thdt**. These variables have been extended to arrays with maximum number of elements equaling 20, and are shown here:

deltmax(1:20) maximum allowed time step (NAMELIST xput).
pltdt(1:20) plotting time interval to file profiles.nc (NAMELIST xput).
prtdt(1:20) printing time interval to file gfout (NAMELIST xput).
tddt(1:20) restart dump and 3D visualization time interval to file gfd*.nc (NAMELIST xput).
thdt(1:20) time-history plotting time interval to file plothist.nc (NAMELIST grafic).

When there is only one entry in the above variable arrays, it is treated as a scalar constant, but when there are multiple entries, they are associated in pairs with the odd elements representing the time interval or time step and the even elements representing the upper time limit for that particular time interval or time step. If the current time exceeds the last time specified in the array **deltmax**, the last maximum time step size will be held for the remaining simulation time up to **twfin**. This holds true for all the other variables list above as well.

The new interval values are calculated at the end of every cycle, but care should be taken. The interval considered is the time difference between the last time output was done and the current time. The fact that the time is not continuous in the simulation causes additional complications.

For example

```
$grafic
…
 tddt          = 0.5,2,
                 10, 50.0,
…
$end
```

How many dumps will be generated up to 50 seconds?

The answer is:

3 up to 2 seconds (at ~0.5, ~1.0, ~1.5) and 4 up to 50 (at ~11.5, ~21.5, ~31.5, ~41.5).

Why only three up to 2 seconds? Because the first output is done at time >= 0.5 seconds. The time step will rarely hit 0.5 seconds exactly. Therefore the first output is somewhat delayed. At times > 2 seconds, the new interval (10 seconds) is already active.

If the user wanted a restart dump at approximately 0.5, 1.0, 1.5, 2,0, 12,0, 22.0, 32.0, and 42.0 s, then the easiest way would be to add a small amount to the interval. In the above case, one could use:

```
$grafic
…
 tddt          = 0.5,2.1,
                 10, 50.0,
…
$end
```

and this would insure that the interval 2.0 is reached before the increment is updated.

An example is shown here for the maximum time step, **deltmax**, as it is read in the NAMELIST xput:

```
$grafic
…
 deltmax      = 0.5,10000.1,
                0.6,20000.1,
                0.7,30000.1,
                0.8,40000.1,
                0.5,50000.1,
                0.6,60000.1,
                0.75,70000.1,
                1.0,500000.0,
…
$end
```

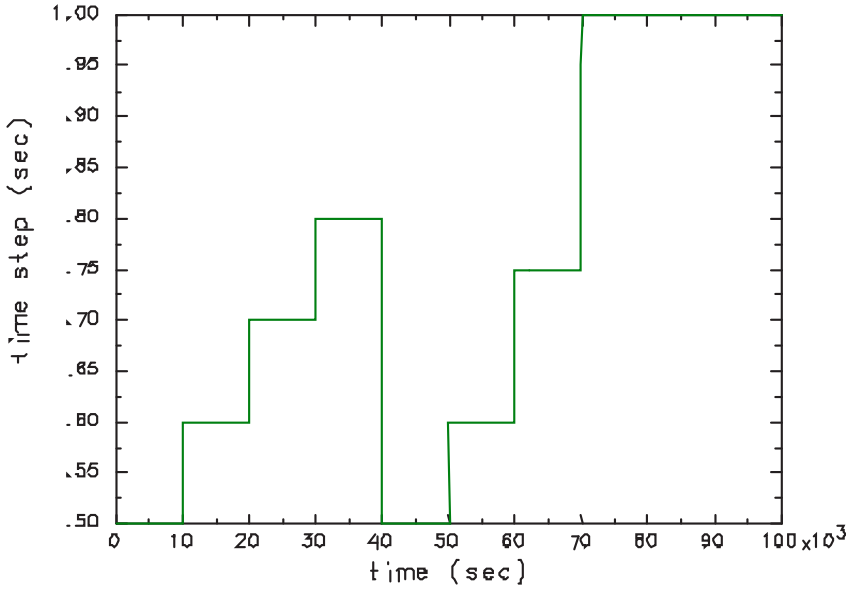


Figure 8-2 Maximum time step shown from the above example.

Two other variables in NAMELIST xput can be controlled in a similar fashion. These variables are **pltdt** and **prtdt**.

The time-history plot frequency interval **thdt** which appears in the NAMELIST graphic. In the very same fashion as the variables and arrays discussed above, we remind the reader that:

thdt array of dimension 20, allows 10 pairs of (time, time-history intervals) to control the time-history output (plothist.nc) time interval.

An example is shown below:

```
$grafic
…
 thdt          = 50.0,  100.1,
                 100.0,500.1,
                 500.0,   1000.1,
                 250.0,   2000.1,
                 500.0,   5000.1,
…
$end
```

In this example, information for the time history plots would occur at approximately the following times: 50 s, 100 s, 200 s, 300 s, 400 s, 1000 s, 1250 s, 1500 s, 1750 s, 2000 s, 3000 s, 3500 s, 4000 s, 4500 s, 5000 s.

9 Output And Restart

9.1 Graphical Outputs

The following is a description of various plotting capabilities available in the code. Most input variables regarding graphical outputs are in NAMELIST group grafic. 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, and 2D and 3D vector plots in profiles.nc, if such plots are requested. Default = 1.

GASFLOW-MPI does however allow the user to dynamically specify the time interval between output, **pltdt**, by making use of the fact that **pltdt** is an array dimensioned 20. When there is only one entry in **pltdt**, it is treated as a scalar constant, i.e., a single time interval, but when there are multiple entries, they are associated in pairs with the odd elements representing the time interval and the even elements representing the upper time limit for that particular time interval. If the current time exceeds the last time specified in the array **pltdt**, the last maximum time step size will be held for the remaining simulation time up to **twfin**.

An example is shown here for the maximum time step, **pltdt**, as it is read in the NAMELIST XPUT:

```
$xput
…
                    ;  VALUE    TIME
pltdt               =  0.5,    10000.1,
                       0.6,    20000.1,
                       0.7,    30000.1,
                       0.8,    40000.1,
                       0.5,    50000.1,
                       0.6,    60000.1,
                       0.75,   70000.1,
                       1.0,    90000.0,
…
$end
```

For this example, the plotting time interval is held constant at 0.5 s for (0 <= time < 10000), 0.6 s for (0 <= time < 20000), and so on. See Section 8.4 for more details.

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

9.1.1 Time-History Plots

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

thp(1,*) i mesh index (cell number or cell face number).
thp(2,*) j mesh index (cell number or cell face number).
thp(3,*) k mesh index (cell number or cell face number).
thp(4,*) Block number (must be 1 for GASFLOW-MPI).
thp(5,*) Solution variable to be plotted. Choose one of the character strings (enclosed in single quotes) given in
Table 9-1.
thp(6,*) 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 **thp(5,*)** 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.

The second dimension in the thp array allows more than one definition of time-history plot request, and the first dimension consists of six 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 GASFLOW-MPI 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 in Figure 3–1) for cell numbering convention. Consider the following examples:

thp(1:6,1) = 4, 8, 2, 1, 'sien', 0,
thp(1:6,2) = 3, 4, 5, 1, 'un', 0,
thp(1:6,3) = 3, 4, 5, 1, 'vn', 0,
thp(1:6,4) = 3, 4, 5, 1, 'vdotz', 0,
thp(1:6,5) = 3, 4, 5, 1, 'mf', 'h2o',
thp(1:6,6) = 4, 8, 2, 1, 'vf', 1,
thp(1:6,7) = 3, 2, 0, 0, 'un', 0,

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. The seventh time-history plot is for the velocity at the third cell face of the second duct.

Table 9-1 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, K.
'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 x 10^6 dynes/cm^2) in NAMELIST group xput.
'pstag'	Stagnation pressure ($\rho\|\mathbf{u}\|^2/2$) minus **pamb0**.
'mu'	Effective (molecular and turbulent) viscosity.
'nu'	Effective viscosity divided by fluid density.
'tke'	Turbulent kinetic energy, only valid if **tmodel** has been set to 'ke'.
'eps'	Rate of dissipation of turbulent kinetic energy, only valid if **tmodel** has been set to 'ke'.
'tsat'	Saturation temperature, K.
'psat'	Saturation pressure, dynes/cm^2.
'rh'	Percent relative humidity.
'tc'	Fluid temperature, °C.
'difft'	Fluid temperature minus the saturation temperature.
'insht'	Instrument cooling heat removal.

The frequency at which time-history data are written and subsequently plotted is controlled by the following input variable:

thdt Time interval (s) at which time-history data are written to plothist.nc.
Default = 1 x 10^{100}.

GASFLOW does however allow the user to dynamically specify the time interval between output, **thdt**, by making use of the fact that **thdt** is an array dimensioned 20. When there is only one entry in **thdt**, it is treated as a scalar constant, i.e., a single time interval, but when there are multiple entries, they are associated in pairs with the odd elements representing the time interval and the even elements representing the upper time limit for that particular time interval. If the current time exceeds the last time specified in the array **thdt**, the last maximum time step size will be held for the remaining simulation time up to **twfin**.

An example is shown here for the maximum time step, **thdt**, as it is read in the NAMELIST XPUT:

```
$xput
…
                    ;  VALUE    TIME
 thdt               =  0.5,    10000.1,
                       0.6,    20000.1,
                       0.7,    30000.1,
                       0.8,    40000.1,
                       0.5,    50000.1,
                       0.6,    60000.1,
                       0.75,   70000.1,
                       1.0,    90000.0,
…
$end
```

For this example, the time history plotting time interval is held constant at 0.5 s for (0 <= time < 10000), 0.6 s for (0 <= time < 20000), and so on. See Section 8.4 for more details. Besides time histories of the fluid solution quantities given in

Table 9-1, the user can also write the surface temperature of a solid heat structure as a function of time into plothist.nc. This is done via the following input variable:

htthp(1,*) i-index of fluid cell in contact with the heat structure.
htthp(2,*) j-index of fluid cell in contact with the heat structure.
htthp(3,*) k-index of fluid cell in contact with the heat structure.
htthp(4,*) Block number (must be 1 for GASFLOW-MPI).
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

'east', +i side of fluid cell;
'west', -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 the 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).

Time dependent liquid film thickness (cm) on all GASFLOW-MPI structures can be plotted in a similar manner. This allows the user to monitor the time dependence of condensation and/or vaporization through dryout from any surface type in the form of a time-history plot. The input is located in NAMELIST rgrafic, and it is defined with the **filmthp** statement:

filmthp(1,*) i-index of fluid cell in contact with heat transfer structure.
filmthp(2,*) j-index of fluid cell in contact with heat transfer structure.
filmthp(3,*) k-index of fluid cell in contact with heat transfer structure.
filmthp(4,*) block number (must be 1 for GASFLOW-MPI).
filmthp(5,*) Heat structure type. Choices are:
'slab': slab heat structure;
'sink': sink heat structure;
'wall': wall heat structure.
filmthp(6,*) Side of fluid cell in contact with heat transfer structure (not needed for sink heat structures). Choices are:
'east', +i side of fluid cell;
'west', -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.

Time dependent energy fluxes due to condensation/vaporization (ergs/(cm^2·s)) on all GASFLOW-MPI structures can be plotted in a similar manner. This allows the user to monitor the time dependence of phase-change heat transfer to or from any surface type in the form of a time-history plot. The input is located in NAMELIST rgrafic, and it is defined with the **condfthp** statement:

condfthp(1,*) i-index of fluid cell in contact with heat transfer structure.
condfthp(2,*) j-index of fluid cell in contact with heat transfer structure.
condfthp(3,*) k-index of fluid cell in contact with heat transfer structure.
condfthp(4,*) block number (must be 1 for GASFLOW-MPI).
condfthp(5,*) Heat structure type. Choices are:
'slab': slab heat structure;
'sink': sink heat structure;
'wall': wall heat structure.
condfthp(6,*) Side of fluid cell in contact with heat transfer structure (not needed for sink heat structures). Choices are:
'east', +i side of fluid cell;
'west', -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.

Time dependent energy fluxes due to convective heat transfer (ergs/(cm^2·s)) on all GASFLOW-MPI structures can be plotted in a similar manner. This allows the user to monitor the time dependence of convection heat transfer to or from any surface type in the form of a time-history plot. The input is located in NAMELIST rgrafic, and it is defined with the **convfthp** statement:

convfthp(1,*) i-index of fluid cell in contact with heat transfer structure.
convfthp(2,*) j-index of fluid cell in contact with heat transfer structure.
convfthp(3,*) k-index of fluid cell in contact with heat transfer structure.
convfthp(4,*) block number (must be 1 for GASFLOW-MPI).
convfthp(5,*) Heat structure type. Choices are:
'slab': slab heat structure;
'sink': sink heat structure;
'wall': wall heat structure.
convfthp(6,*) Side of fluid cell in contact with heat transfer structure (not needed for sink heat structures). Choices are:
'east', +i side of fluid cell;
'west', -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.

Time dependent energy fluxes due to radiation heat transfer (ergs/(cm^2·s)) on all GASFLOW-MPI structures can be plotted in a similar manner. This allows the user to monitor the time dependence of radiation heat transfer to or from any surface type in the form of a time-history plot. The input is located in NAMELIST rgrafic, and it is defined with the **radfthp** statement:

radfthp(1,*) i-index of fluid cell in contact with heat transfer structure.
radfthp(2,*) j-index of fluid cell in contact with heat transfer structure.
radfthp(3,*) k-index of fluid cell in contact with heat transfer structure.
radfthp(4,*) block number (must be 1 for GASFLOW-MPI).
radfthp(5,*) Heat structure type. Choices are:
'slab': slab heat structure;
'sink': sink heat structure;
'wall': wall heat structure.
radfthp(6,*) Side of fluid cell in contact with heat transfer structure (not needed for sink heat structures). Choices are:
'east', +i side of fluid cell;
'west', -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.

Time dependent energy fluxes due to recombination heat transfer (ergs/(cm^2·s)) on all GASFLOW-MPI structures can be plotted in a similar manner. This allows the user to monitor the time dependence of recombination heat transfer to or from any surface type in the form of a time-history plot. The input is located in NAMELIST rgrafic, and it is defined with the **qrecfthp** statement:

qrecfthp(1,*) i-index of fluid cell in contact with heat transfer structure.
qrecfthp(2,*) j-index of fluid cell in contact with heat transfer structure.
qrecfthp(3,*) k-index of fluid cell in contact with heat transfer structure.
qrecfthp(4,*) block number (must be 1 for GASFLOW-MPI).
qrecfthp(5,*) Heat structure type. Choices are:
'slab': slab heat structure;
'sink': sink heat structure;
'wall': wall heat structure.
qrecfthp(6,*) Side of fluid cell in contact with heat transfer structure (not needed for sink heat structures). Choices are:
'east', +i side of fluid cell;
'west', -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.

All of the above time history plots are limited to 500 separate plots.

9.1.2 Profile Plots

The user can plot all of the solution variables listed in Table 9–1 (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, GASFLOW-MPI 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 1 for GASFLOW-MPI).

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 (the point number is the second index of the corresponding **pnt** definition).
p1d(2,*) Identification number of the second point (the point number is the 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 9-1, 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 “colinear” 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.

In a similar manner, 1D profile plots for certain structural surface characteristics can also be plotted. The user can specify what the 1D surface profile plots are via the following input variable:

p1dsurf(1,*) Identification number of the first point (the point number is the second index of the corresponding **pnt** definition).

p1dsurf(2,*) Identification number of the second point (the point number is the second index of the corresponding **pnt** definition).

p1dsurf(3,*) Surface solution variable whose 1D profile is to be plotted. Choose one of the following symbols:
‘condf’ Water vapour energy flux from condensation or evaporation from the given surface, ergs/(cm^2·s) (see next input variable).
‘convf’ Convective energy flux for a given surface, ergs/(cm^2·s) (see next input variable).
‘filmt’ Film of water on a given surface, cm (see next input variable).
‘htcoef’ Heat transfer coefficient for a given surface, ergs/(cm^2·K·s) (see next input variable).
‘massf’ Water vapour mass flux condensing or evaporating from the given surface, g/(cm^2·s) (see next input variable).
‘qrecf’ Energy flux for a given recombiner surface, ergs/(cm^2·s) (see next input variable).
‘radf’Radiation energy flux for a given surface, ergs/(cm^2·s) (see next input variable).

p1dsurf(4,*) Heat structure type. Choices are
'slab', slab heat structure;
'sink', sink heat structure;
'wall', wall heat structure.

p1dsurf(5,*) The side of the fluid cell which is in contact with the heat structure whose surface profile is to be plotted. This entry is only used if **p1ds(4,*)** has been set to 'slab' or 'wall', because sink structures are assumed to be distributed in the fluid cell. Choices are
'east',+i side of fluid cell;
'west',-i side of fluid cell;
'north',+j side of fluid cell;
'south',-j side of fluid cell;
'top',+k side of fluid cell;
'bottom', -k side of fluid cell.

Again 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 :4,1) = 3, 4, 10, 1,
pnt(1 :4,2) = 33, 4, 10, 1,
pnt(1 :4,3) = 15, 6, 7, 1,
pnt(1 :4,4) = 15, 66, 7, 1,

p1dsurf(1 :5,1) = 1, 2, ‘massf’, ‘wall’, ‘north’,
p1dsurf(1 :5,2) = 1, 2, ‘filmt’, ‘slab’, ‘bottom’,
p1dsurf(1 :5,3) = 3, 4, ‘convf’, ‘wall’, ‘east’,

Here again four points are defined, with the first two and the last two points being “colinear” pairs. Therefore the two pairs of points, 1 and 2, and 3 and 4, can be used to define 1D surface profile plots. The first surface profile plot is that of the mass flux along the line going from point 1 to point 2 on the wall surface located on the north side. The second surface profile plot is film thickness along the same line of fluid cells but on a slab surface located on the bottom of these fluid cells. The third surface profile plot is that of the convective heat flux along the line of fluid cells defined by points 3 and 4 on the wall located on the east side of the fluid cells.

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 1 for GASFLOW-MPI).
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
'east',+i side of fluid cell;
'west',-i side of fluid cell;
'north',+j side of fluid cell;
'south',-j side of fluid cell;
'top',+k side of fluid cell;
'bottom', -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). All of the above 1-dimensional profile plots are limited to 500 separate plots.

9.1.3 2D Contour

It is often useful to plot the contour of a solution quantity on a plane (for example, to identify “hot spots” in certain calculations). Two-dimensional 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 “coplanar” points) are used to define the plane where data are to be taken for the contour plot. Once some (coplanar) 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 9–1, 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(1:4,1) = 3, 1, 1, 1,
pnt(1:4,2) = 3, 12, 10, 1,
pnt(1:4,3) = 2, 1, 7, 1,
pnt(1:4,4) = 15, 12, 7, 1,
```

c2d(1:4,1) = 1, 2, 'pn', 0,
c2d(1:4,2) = 1, 2, 'rsn', 'h2',
c2d(1:4,3) = 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.

9.1.4 Velocity Vector

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. Two-dimensional velocity vector plots show the velocity magnitude and direction on a plane defined by two coplanar points. Three-dimensional 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(1:4,1) = 1, 1, 2, 1,
pnt(1:4,2) = 9, 9, 2, 1,
v2d(1:3,1) = 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.

9.1.5 Graphic and Tabular Particle Data Output

(Warning: Lagrangian particle model is not available in current GASFLOW-MPI 1.0. Parallelization of particle model will be implemented in future release of GASFLOW-MPI.)

Two kinds of graphic output are used to interpret the results of particle computations. One of these displays particles plotted in a perspective view plot of the computational domain. Any combination of particle classes may be selected for each plot. The three-dimensional volume of particles is integrated along each line of sight and plotted onto the two-dimensional plane. A varied selection of eye points from which to view the mesh and particles at selected times in the transport of the particles can give a qualitative interpretation of the particle behavior. Another useful graphic display is the time-history plots of particle number, mass, volume fraction, mass fraction, and mass deposited in selected mesh cells. In addition, the time-history data is written on an output tape, PTHDATA, and is available for examination.

The input parameter definitions are given in Appendix D, which describes NAMELIST group grafic input. However, a summary listing will be given here of the input parameters specifically used in the particle perspective plot displays.

The following input is for perspective particle plots and is in the grafic NAMELIST group:

ippka(np) Selects the package (i.e., particle class, size, or deposition plane) to be plotted. The input corresponds to the array mpac definition, given below. Default = 1.

ipvew(np) Viewpoint for particle plot, i.e., viewpoint 'nv' defined in **viewcrds**. Default = 1. (Input value corresponds with the array pplt(2,100))

nap(np) Number of film frame advances between particle plots 'np'. Default = 1.

npplts The number of particle packages (i.e., classes or deposition planes) to be plotted. Default = 0.

pplt(2,100) Perspective plot viewpoint, i.e., the viewpoint 'nv' defined in **viewcrds**. (Corresponds with **ipvew** and is used for perspective plots in general.)

viewcrds(6,nv) 3D view description, object center and eye coordinates: xco, yco, zco, xeye, yeye, zeye. Default = 1.0.

The following input is for time-history particle plots and is in the grafic NAMELIST group “xput”:

thdt Time increments between time-history data points. Default = 10^{100} s.

pthpt0 Time-history plot initial time (used for runs from restart tapes).

pthp(1,*) i-index of fluid cell.

pthp(2,*) j-index of fluid cell.

pthp(3,*) k-index of fluid cell.

pthp(4,*) Block number.

pthp(5,*) Particle data to be plotted:
'pnc', particle number concentration ;
'pmc', particle mass concentration;
'pmt', total mass in fluid cell;
'pmf', particle mass fraction;
'pvf', particle volume fraction;
'pmd', particle mass deposited;
'pmntr', particle cloud mass detected at each monitor.

pthp(6,*) Particle class (**itpcl(n)**):
= 0, all classes;
> 0, particle class number.

pthp(7,*) Particle size number (**itpsz(n)**):
= 0, all sizes for class **itpcl(n)**;
> 0, particle size number.

pthp(8,*) Particle mass deposited on cell faces (for 'pmd' only):
0, all deposited particles;
>0, cell faces designated in array mpac(n)
11, deposited on east face of cell;
21, deposited on west face of cell;
12, deposited on north face of cell;
22, deposited on south face of cell;
13, deposited on top face of cell;
23, deposited on bottom face of cell.

The procedure to set up perspective view particle plots:

1. Input the selected viewpoints, **viewcrds**. The object center coordinates should be the *x*-, *y*-, *z*-coordinates at the center of the mesh. The eye points selected should position the viewpoint several mesh dimensions from the mesh center.
2. Select the number of particle plots wanted, **npplts**.
3. Select the particle class number and the class size number or the cell faces with deposited particles that are to be plotted, **ippka**. For the particle class and size, the input number is a two-digit integer. The rightmost digit corresponds to the particle class number. The leftmost digit corresponds to the class size number. For example, to plot class 2, size 3 particles the input would be **ippka(1)** = 32. To plot all particles of all classes and sizes set **ippka(1)** = 0. The cell face flags are those defined in the array mpac(n). The input value for **ippka** is the value of mpac(n) + 100. For example, to plot all the particles deposited on the “floor,” i.e., the bottom face of each cell in the k = 2 plane of the mesh, input **ippka(1)** = 123. This would also plot all particles deposited on the top side of an obstacle.
4. Select the perspective view for each plot from the **viewcrds** input. This number is set into both **ipvew** and **pplt**. For the second perspective view being defined, if **viewcrds** 3 is chosen, **ipvew(2)** = 3 and **pplt(1,2)** = 3.

5. The number of film frames advanced, **nap**, is typically 1, which is the default value. However, if the frames are to be cut and mounted for slides, advancing 2 frames between each plot is advisable.

The procedure to set up time-history plots:

1. Choose whether or not the plots will have grid lines, **gline**. The default is 'on'.
2. Select the time interval between data points, **thdt**.
3. Select the time-history plot initial time, **pthpt0**. (Note that the last character in this parameter is a zero.) This has a default value of 0.0, but can be set to the restart time when a steady-state flow field has been established with **solatype** = 0, and a restart with **solatype** = 2 is being used to initialize particle input. For example, if restarting from a tape with a dump time of 1.32 s, the particle time-history plots will start at time 0 if **pthpt0** is set to 1.32.
4. Select the particle time-history data to be plotted, **pthp**, and input using the description of the input parameter given above. For example, to plot the total mass of particle class 2, size 3 in fluid cell i = 5, j = 4, k = 12, block #1, enter

pthp(1:8,1) = 5, 4, 12, 1, 'pmt', 2, 3, 0,

When particle cloud mass is plotted and **pthp(*,5)** is 'pmntr', the other **pthp** input is not required; however it is recommended for plot identification.

Particle data may also be plotted as a 1-dimensional profile plot and 2-dimensional distribution plots. The mechanics of these plots is a combination of Section 9.1.2, Section 9.1.3, and time-history particle plots discussed above. In both **p1d** and **c2d** plotting, the respective solution variable, **p1d(3,*)** and **c2d(3,*)**, is listed as an extension of Table 9-2, listed here as Table 9-2 Particle Plotting.

Table 9-2 Solution variables available for particle plotting

Symbol	Quantity to be plotted
'pnc'	Particle number concentration.
'pmc'	Particle mass concentration.
'pmt'	Total particle mass.
'pmf'	Particle mass fraction.
'pvf'	Particle volume fraction.
'pmd'	Deposited particle mass.
'pup'	Averaged x-direction particle velocity component.
'pvp'	Averaged y-direction particle velocity component.
'pwp'	Averaged z-direction particle velocity component.

It is possible to plot only selected subsets of the total particle data. The user can selectively choose from 3 parameters: (1) particle mass deposition for ‘pmd’ only, (2) particle class, and (3) the particle size. These 3 parameters are packed into the species part of **p1d** and **c2d**, that is, **p1d(4,*)** and **c2d(4,*)**.

The particle size is packed into the least significant digits, 1 through 99, while the particle class is packed into the hundreds through thousands, 100 through 9999, and the mass deposition into the most significant digits 5 and 6. Zero as a parameter means all sizes and classes.

The following input stream illustrates some of the particle plotting capability:

```
$grafic
…
 pnt(1:4,1)     = 3, 4, 1, 1, ; profile point 1
 pnt(1:4,2)     = 3, 4, 10, 1, ; profile point 2
 pnt(1:4,3)     = 2, 1, 7, 1, ; 2D point 1
 pnt(1:4,4)     = 15, 12, 7, 1, ; 2D point 2
 p1d(1:4,1)     = 1, 2, 'pnc', 0,
                  ; particle number concentration: all sizes and classes
 p1d(1:4,2)     = 1, 2, 'pmd', 210302,
                  ; deposited particle mass: size 2, class 3, deposited in west cell face.
 p1d(1:4,3)     = 1, 2, 'pmc', 000103,
                  ; particle mass concentration: size 3, class 1.
 c2d(1:4,1)     = 3, 4, 'pnc', 0200,
                  ; particle number concentration: all sizes and class 2.
 c2d(1:4,2)     = 3, 4, 'pmd', 130103,
                  ; deposited particle mass: size 3, class 1, deposited in top cell faces.
 c2d(1:4,3)     = 3, 4, 'pmc', 0,
                  ; particle mass concentration: all sizes and classes.
…
$end
```

9.1.6 Graphic Display of Criteria of FA and DDT

A methodology has been developed to evaluate the safety of ignitor implementation in complex containment geometries. The method consists of the following steps:

1. determination of bounding H_2/steam sources,
2. high-resolution analysis of the three-dimensional transport calculation,
3. evaluation of the detonation potential at the time of ignition,
4. optimization of the ignitor system such that only early ignition and nonenergetic combustion occurs, and
5. modeling of the continuous deflagration process during H_2 release.

In order to evaluate this new methodology and determine detonation possibilities in complex-shaped rooms for complicated geometries with GASFLOW-MPI, we have implemented a generalized input in the NAMELIST input block xput.

To activate this methodology, the user may define multiple rooms in several ways with the **iroomdef** two-dimensional array. Each room segment volume is defined by eight entries in the **iroomdef** input array.

iroomdef(1,*) Beginning i mesh index (cell face number).
iroomdef(2,*) Ending i mesh index (cell face number).
iroomdef(3,*) Beginning j mesh index (cell face number).
iroomdef(4,*) Ending j mesh index (cell face number).
iroomdef(5,*) Beginning k mesh index (cell face number).
iroomdef(6,*) Ending k mesh index (cell face number).
iroomdef(7,*) Block number (must be 1 for GASFLOW-MPI).
iroomdef(8,*) Actual room number:
> 0 implies positive volume;
< 0 implies negative volume.

The asterisk (*) should be replaced by an integer that identifies the particular **iroomdef** definition. GASFLOW-MPI supports 300 definitions of **iroomdef** and 25 separate and distinct different rooms. The utility of the **iroomdef** input is best demonstrated by an example or two as follows:

Consider a two-dimensional computational mesh that has 9 fluid cells in the *x*-direction and 5 fluid cells in the *y*-direction Figure 9–1. There is a complex-shaped room in this mesh shown by the obstacles.

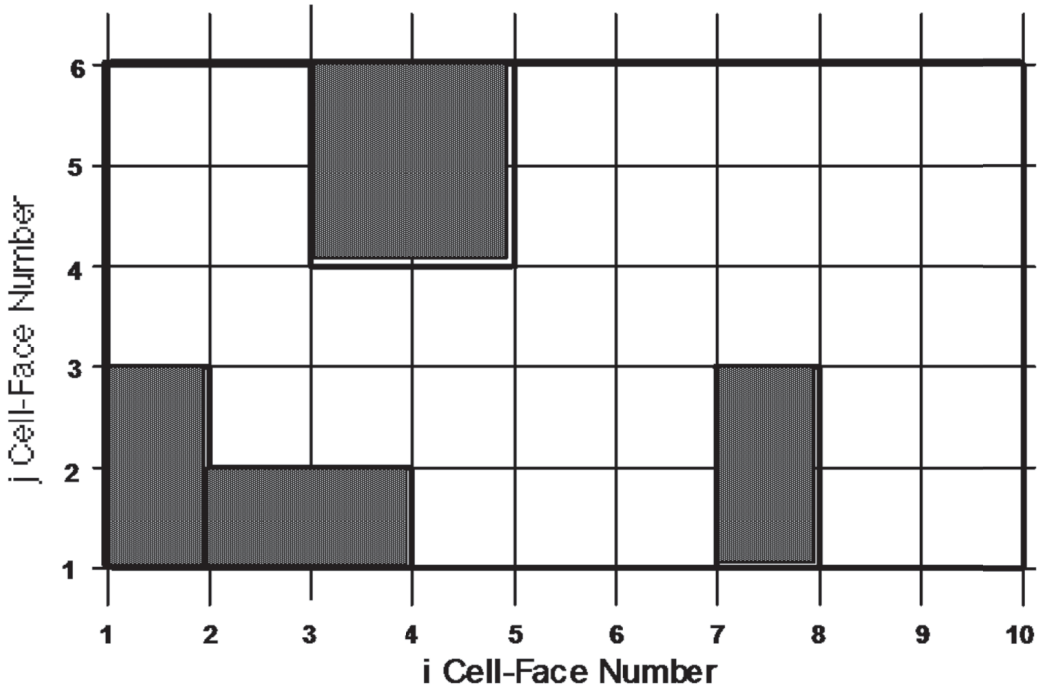


Figure 9-1 Example problem to demonstrate the utility of the iroomdef input option

One way to define this room is

```
$xput
…
 mat        = 'h2', 'n2', 'o2', 'h2o', 'h2ol', ; components -> HEM
 mobs       = 1, 2, 1, 3, 1, 2, 1, 1,
              2, 4, 1, 2, 1, 2, 1, 1,
              3, 5, 4, 6, 1, 2, 1, 1,
              7, 8, 1, 3, 1, 2, 1, 1,
 iroomdef   = 1, 3, 3, 6, 1, 2, 1, +1,
              4, 7, 1, 2, 1, 2, 1, +1,
              2, 7, 2, 3, 1, 2, 1, +1,
              8, 10, 1, 3, 1, 2, 1, +1,
              3, 10, 3, 4, 1, 2, 1, +1,
              5, 10, 4, 6, 1, 2, 1, +1,
…
$end
```

Another way to define this same room is

```
$xput
…
 mat        = 'h2', 'n2', 'o2', 'h2o', 'h2ol', ; components -> HEM
 mobs       = 1, 2, 1, 3, 1, 2, 1, 1,
              2, 4, 1, 2, 1, 2, 1, 1,
              3, 5, 4, 6, 1, 2, 1, 1,
              7, 8, 1, 3, 1, 2, 1, 1,
 iroomdef   = 1, 10, 1, 6, 1, 2, 1, +1,
              1, 2, 1, 3, 1, 2, 1, -1,
              2, 4, 1, 2, 1, 2, 1, -1,
              3, 5, 4, 6, 1, 2, 1, -1,
              7, 8, 1, 3, 1, 2, 1, -1,
…
$end
```

As another example, let’s consider the same computational domain, but make the single room into three rooms by the vertical walls at i = 4 and i = 7 (shown by the heavy thick lines in Figure 9-2).

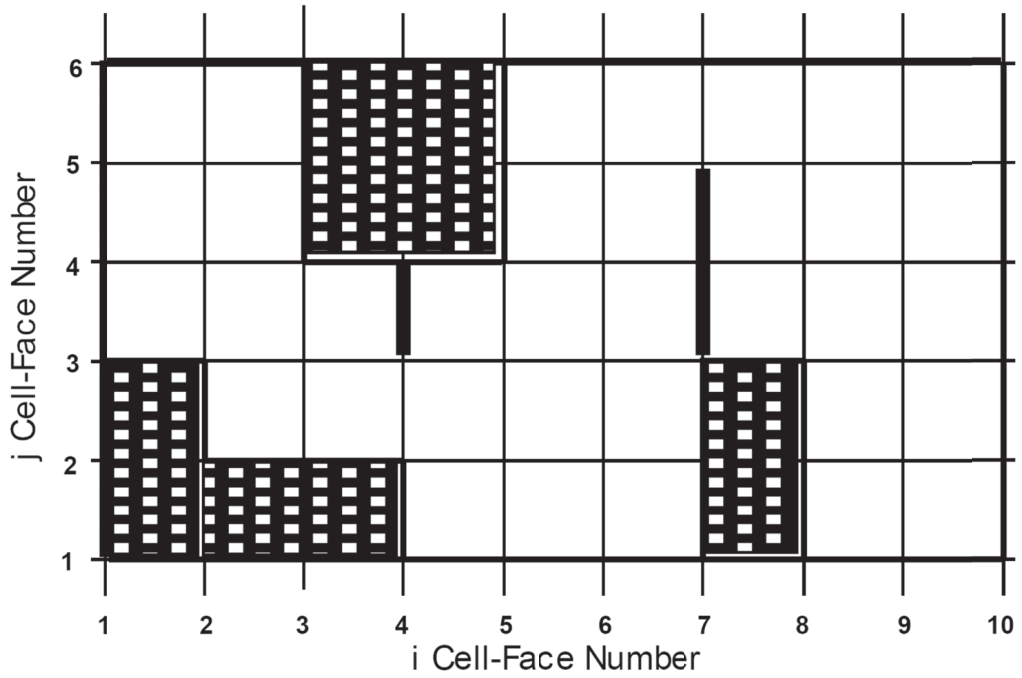


Figure 9-2 Another example problem to demonstrate the utility of the iroomdef input option

Input this geometry as follows:

```
$xput
…
 mat      = 'h2', 'n2', 'o2', 'h2o', 'h2ol', ; components -> HEM
 mobs     = 1, 2, 1, 3, 1, 2, 1, 1,
            2, 4, 1, 2, 1, 2, 1, 1,
            3, 5, 4, 6, 1, 2, 1, 1,
            7, 8, 1, 3, 1, 2, 1, 1,
 walls    = 4, 4, 3, 4, 1, 2, 1, 1,
            7, 7, 3, 5, 1, 2, 1, 1,
 iroomdef = 1, 4, 2, 6, 1, 2, 1, +1,
            1, 2, 2, 3, 1, 2, 1, -1,
            3, 4, 4, 6, 1, 2, 1, -1,
            4, 7, 1, 4, 1, 2, 1, +2,
            5, 7, 4, 6, 1, 2, 1, +2,
            7, 10, 1, 6, 1, 2, 1, +3,
            7, 8, 1, 3, 1, 2, 1, -3,
$end
```

9.1.6.1 DDT Characteristics

The Kurchatov Institute has developed a function that relates the detonation cell size to the concentrations of dry hydrogen and steam in air at temperature, *T*, i.e., $\lambda = \lambda\left([H_2],[H_2O],T\right)$. the exact equation is

$$\log_{10}(\lambda) = \begin{cases} a + \left(b/(A - l/T)^f + i\cdot(A - g\cdot T) + h\cdot(A - g\cdot T)^2\right)\cdot \\ \left(1 + d\cdot C + e\cdot T\cdot C^2\right)\cdot j/T \end{cases} \qquad \text{Equ. 9-1}$$

where *A* is the H_2 volume fraction, *C* is the steam volume fraction and *T* is the temperature. The coefficients a = -1.13331e+00, b = 4.59807e+01, d = 4.65429e-02, e = 3.59620e-07, f = 9.97468e-01, g = -2.66646e-02, h = 8.74995e-04, i = -4.07641e-02, j = 3.31162e+02, *l* = -4.18215e+02.

If there are volumes defined by the **iroomdef** statements and **idetchar** > 0, plots will be produced showing the detonation cell size and the 7λ mixture sensitivity criterion.

The user has some options concerning hydrogen limits in the **iroomdef** statements GASFLOW-MPI will process. The two variables in the NAMELIST group graphic are **h2lowfl** and **h2upfl**. **h2lowfl** is an array dimensioned 10 and **h2upfl** is a constant. These are respectively the lower hydrogen volume fractions and maximum hydrogen volume fraction the user is interested in processing:

h2lowfl(1:10) lower hydrogen volume fraction threshold (default: **h2lowfl(1) =** 0.04).
h2upfl upper hydrogen volume fraction threshold (default: **h2lowfl(1) =** 0.75).

9.1.6.2 Sigma Index Characteristics

Above we describe the implementation of a DDT criterion or index to examine by user defined volumes the time and space sensitivity of hydrogen mixtures to transition from a subsonic flame to a detonation. Another criterion, an index to judge the possibility of a laminar flame becoming turbulent and accelerating, through the so-called sigma criterion or index is also available. KIT has developed a 4 dimensional table, called the sigma criterion table, with the dependent variablesH_2, H_2O, O_2 and temperature T. The range of this table, the **sigma.dat** file, is H_2: from 0% to 70% in 5% increments (i = 1,15,1), H_2O: from 0% to 60% in 10% increments (j = 1,7,1), O_2: from 0% to 25% in 5% increments (k = 1,6,1) and T: from 300K to 1000K in 100K increments (l = 1,8,1).

However, instead of interpolating this 4-dimensional table, an accurate analytic function has been developed so a direct evaluation of the sigma criteria is available. The user can activate plots of the sigma index, defined as

$$\sigma_{index} = \frac{\sigma(h_2, h_2o, o_2, T)}{\sigma_{critical}(h_2, h_2o, T)} \qquad \text{Equ. 9-2}$$

by specifying volumes using the **iroomdef** statement and

$$\textbf{idetchar} = \begin{cases} 0;\ \text{no plots (default)} \\ 1;\ \text{only } 7\lambda \text{ plots} \\ 2;\ 7\lambda \text{ and } \sigma \text{ plots} \end{cases} \qquad \text{Equ. 9-3}$$

in the GASFLOW **xput** input stream. When **idetchar** > 1, the GASFLOW-MPI plothist.nc file will contain sigma index plots for the maximum hydrogen concentration in the cloud, the minimum hydrogen concentration in the cloud, and the average hydrogen concentration in the cloud, where the cloud is defined as all computational volumes in the specified room which are combustible. Cloud combustible limits are judged using the Kumar criterion.

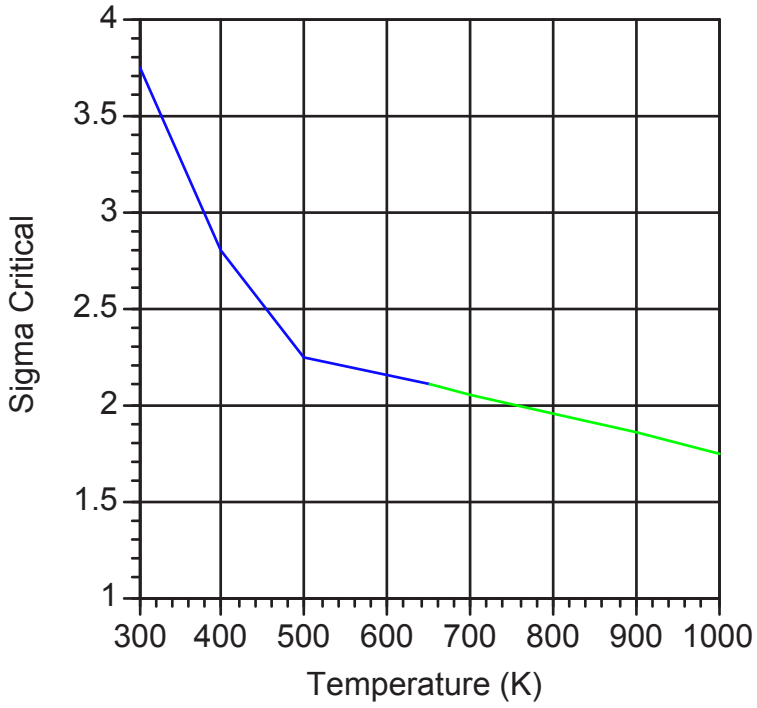


Figure 9-11 Sigma Critical as a function of Temperature for lean hydrogen-oxygen mixtures.

This graph and the associated tabular values listed in the following Table are valid for lean hydrogen-oxygen mixtures, i.e., when $[h_2] < 2[o_2]$, while for rich hydrogen-oxygen mixtures, $[h_2] \geq 2[o_2]$, sigma critical is constant at 3.75.

Table 9-3 List of critical sigma as a function of temperature for hydrogen lean and rich oxygen mixtures.

Temperature	**Critical Sigma** $[h_2] < 2[o_2]$	**Critical Sigma** $[h_2] \geq 2[o_2]$
300	3.75	3.75
400	2.80	3.75
500	2.25	3.75
650	2.10	3.75

The idea is that when the sigma index is less than 1, it is highly unlikely a laminar flame will accelerate, while for sigma index values greater than one, there is the potential for flame acceleration.

9.1.7 Printed Output

In addition to graphical outputs, GASFLOW provides printed outputs for each calculation. A printed output file is cyclinfo, which lists iteration and time-step information at each computational time cycle. This file was discussed in Section 8.1.

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 spacing (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.

GASFLOW does however allow the user to dynamically specify the time interval between output, **prtdt**, by making use of the fact that **prtdt** is an array dimensioned 20. When there is only one entry in **prtdt**, it is treated as a scalar constant, i.e., a single time interval, but when there are multiple entries, they are associated in pairs with the odd elements representing the time interval and the even elements representing the upper time limit for that particular time interval. If the current time exceeds the last time specified in the array **prtdt**, the last maximum time step size will be held for the remaining simulation time up to **twfin**.

An example is shown here for the maximum time step, **prtdt**, as it is read in the NAMELIST XPUT:

```
$xput
…
 prtdt          ;   VALUE    TIME
                =    0.5,   10000.1,
                     0.6,   20000.1,
                     0.7,   30000.1,
                     0.8,   40000.1,
                     0.5,   50000.1,
                     0.6,   60000.1,
                     0.75,  70000.1,
                     1.0,   90000.0,
…
$end
```

For this example, the plotting time interval is held constant at 0.5 s for (0 <= time < 10000), 0.6 s for (0 <= time < 20000), and so on. See Section 8.4 for more details.

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 (gfd*n*, where *n* is an integer) whenever a restart dump file is written. At the end of gfout, the total central processing unit CPU time used is reported, as well as the per-cell, per-cycle CPU time.

9.2 Output to Terminal

Besides graphical and text file outputs, GASFLOW 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 Section 8.1); 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:

cttyfreq Number of cycles between printing of time-step and pressure iteration information to the cyclinfo file. Default = 1.

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.

9.3 Restart

Because GASFLOW-MPI is capable of solving complex, large problems, it may take a large amount of computer time to finish a problem. Therefore, the code provides 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.

GASFLOW-MPI does however allow the user to dynamically specify the time interval between restart dumps, **tddt**, by making use of the fact that **tddt** is an array dimensioned 20. When there is only one entry in **tddt**, it is treated as a scalar constant, i.e., a single time interval, but when there are multiple entries, they are associated in pairs with the odd elements representing the time interval for restart dump output and the even elements representing the upper time limit for that particular time interval. If the current time exceeds the last time specified in the array **tddt**, the last restart dump output twill be **twfin**.

An example is shown here for the maximum time step, **tddt**, as it is read in the NAMELIST XPUT:

```
$xput
…
 tddt          ;  VALUE   TIME
               =  0.5,   10000.1,
                  0.6,   20000.1,
                  0.7,   30000.1,
                  0.8,   40000.1,
                  0.5,   50000.1,
                  0.6,   60000.1,
                  0.75,  70000.1,
                  1.0,   90000.0,
…
$end
```

For this example, the restart dump file interval is held constant at 0.5 s for (0 <= time < 10000), 0.6 s for (0 <= time < 20000), and so on. See Section 8.4 for more details.

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 contains the final solution will have the name gfd*n*, 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.

10 General User Guidance for GASFLOW-MPI

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 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 or unexpected extremely small time step, the user should review the input deck. Most problems arise from incorrectly specifying initial and boundary conditions. The user should ask the following:

1. Have all fluid cells in the entire domain been given the correct initial conditions via defining the **gasdef** array appropriately? If not, please check all the **gasdef** definitions if extremely high pressure difference exists in some cells causing the calculation fail.

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? For problems with shock wave or combustion, always start with small time step 1.0e-6 s or less.

4. Does tightening the pressure iteration convergence criterion (i.e., reducing **epsi0**) give the same unphysical results?

5. Does the turbulence dominant the calculation with very small time step? Try to change the initial conditions of the turbulent variables by using **tkeamb0**, **epsamb0** or **turbdef**.

Finally, the user should note that GASFLOW-MPI 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.

10.1 Approach of GASFLOW parallelization

GASFLOW-MPI is the parallel version of GASFLOW using the paradigms of Message Passing Interface (MPI) and domain decomposition. The data structure, parallel linear solvers and pre-conditioners of Portable Extensible Toolkit for Scientific Computing (PETSc) were employed.

PETSc is one of the most widely used software library for high-performance computational science. It can provide numerical infrastructure for application codes in which the implicit numerical solution of partial differential equations are involved. PETSc features distributed data structures, such as index sets, distributed vectors and distributed matrices in several sparse storage formats, as the fundamental objects. Krylov subspace methods, preconditioners and Newton-like nonlinear methods are implemented in a data structure-neutral manner which provides a uniform interface for application programmers. The portability of PETSc is achieved through MPI, but the detailed message passing required during the coordination of the computations is handled inside the PETSc library.

GASFLOW serial version was written in FORTRAN 90 with more than 120,000 lines and 634 subroutines in version 3.5. The ICE’d-ALE solution methodology incorporated in GASFLOW requires the solution of an elliptic pressure equation for the efficient calculation of flows at all-speeds. The discretization of this elliptic equation results in a large scale symmetrically sparse linear equation system. The GASFLOW serial preconditioning algorithm is dependent upon a recursive numerical methodology that heavily depends upon “indirect addressing” which may reduce the computational efficiency and not be suitable for parallelization. Therefore, all the programs relevant to the linear solver and preconditioner in GASFLOW serial version must be replaced by the parallel linear solvers and preconditioners in the PETSc library. Sparse symmetric system is derived from the discretization of the elliptic pressure equation in GASFLOW-MPI. The combination of linear solver, conjugate gradient (CG) and pre-conditioner, Block Jacobi (BJACOBI), was selected as the default solver for the solution of the elliptic pressure equation in the current version of GASFLOW-MPI.

10.2 To obtain decent parallel efficiency using GASFLOW-MPI

GASFLOW-MPI can run on any kind of parallel systems which supports MPI. In order to achieve the best parallel performance, the users need to have:

1. A fast, low-latency interconnect between computational nodes;

2. High per-core memory performance. Each core needs to have its own memory bandwidth of roughly 2 or more Gigabytes/second. This is because the speed of sparse matrix computations is almost totally determined by the speed of the memory access, not the speed of the CPU. Number of floating point instructions submitted to the CPU is significantly less than number of memory

references which have to be resolved to obtain data, meaning that matrix vector multiply kernel is memory bound;

3. The computational domain must be decomposed in the way that each sub-domain has no less than approximately 10,000~20,000 cells. Workload of each CPU must overweigh the communication time. For example, for small problem with 640,000 cells, using 64 processor can usually obtain good speed-up. The performance may decrease by using more processors because the communication effort increases.

10.3 Running GASFLOW-MPI

We will demonstrate how to run GASFLOW-MPI in parallel on distributed processors. Domain decomposition is used as the method of parallel computing. The geometry and associated field variables are broken into small pieces in sub-domains and allocated to separate processors for solution. The parallel running uses the public domain openMPI implementation of the standard MPI. GASFLOW-MPI has been designed to be compatible to the input and output of the GASFLOW serial version. It means GASFLOW-MPI can read the same input file, ingf, and export the same calculation results in NETCDF format as GASFLOW serial version. Therefore, the users can use GASFLOW-MPI in the same way as they used the GASFLOW serial version without the need to know details of parallelization.

10.3.1 Domain decomposition

In principle, the user should not manually decompose the computational domain in ingf file. By default, the computational domain is automatically decomposed in an optimized way in GASFLOW-MPI. Nevertheless, there is an option available to manually control the domain decomposition for advanced users. For most of the users, they can use the same input deck as they used for GASFLOW serial version.

The input variable is “autodecomp”. By default, autodecomp = 1 which means the domain decomposition is controlled automatically in GASFLOW-MPI. Unless absolutely necessary, such as for debugging purpose, the user can use autodecomp = 0 to manually control the domain decomposition. nxprocs, nyprocs and nzprocs are number of processes in x, y and z axis, respectively. Please note that nxprocs*nyprocs*nzprocs must be equal to the total number of processes allocated to the parallel computating.

Warning: autodecomp=0 means that the users manually control the domain decomposition. With this option, best performance is not guaranteed. It is highly recommended that the users use the default value autodecomp = 1.

10.3.2 Running GASFLOW-MPI in parallel

GASFLOW-MPI can be run on a local multiprocessor machine very simply but when running on machines across a network, a file must be created that contains the host names of the machines. The file can be given any name and located at any path. In the following description we shall refer to such a file by the generic name, including full path, <machines>.

An application is run in parallel using mpirun.

mpirun --hostfile <machines> -np <nProcs> xgfmpi

xgfmpi denotes the executable of GASFLOW-MPI. -np represents number of processes the user needs for the parallel computing. For example, if you have the hostfile, hostpgf, and you want to run xgfmpi with 32 processes:

mpirun --hostfile hostgf -np 32 xgf

For more details of running MPI jobs, please refer to

http://www.open-mpi.de/faq/?category=running#mpirun-specify-hosts

APPENDIX

A. Summary of Variables in NAMELIST Group xput

Variables	Default	Description	Restart
areardef(*,*)	none	Defines fractional mesh areas at openings (plane surface). See Section 3.3.4.	yes
areardef(1,*)	none	Beginning i mesh index (cell face number).	yes
areardef(2,*)	none	Ending i mesh index (cell face number).	yes
areardef(3,*)	none	Beginning j mesh index (cell face number).	yes
areardef(4,*)	none	Ending j mesh index (cell face number).	yes
areardef(5,*)	none	Beginning k mesh index (cell face number).	yes
areardef(6,*)	none	Ending k mesh index (cell face number).	yes
areardef(7,*)	none	Block number.	yes
areardef(8,*)	none	If areardef(12,*) > 0 Fraction of the i, j, k plane open for flow. If areardef(8,*) is less than zero and areardef(9,*) equals zero, then no flow loss is included at these locations. If areardef(8,*) is less than zero and areardef(10,*) equals zero, then no laminar drag loss is included at the locations specified by this areardef definition. If areardef(12,*) < 0 Fraction of defined cell face area open for flow. We normally set this entry < zero with areardef(9,*) and areardef(10,*) either equal to zero or negative to indicate i,j,k,iblock index packing.	yes
areardef(9,*)	0.0	If areardef(12,*) > 0 User input flow loss coefficient. If areardef(9,*) is zero, then orifice loss coefficient calculated by GASFLOW. If areardef(9,*) is zero and areardef(8,*) is less than zero, then no flow loss is calculated for the locations specified by this areardef definition. If areardef(12,*) < 0 Computational index for evaluating Equ. 3-11 = 0.0 for the use of local indexes to compute the mass flow rate. < 0.0 for the use of i,j,k,iblock index packing to compute the mass flow rate.	yes
areardef(10,*)	0.0	If areardef(12,*) > 0 Hydraulic diameter to be used for the laminar drag loss. If areardef(10,*) is zero, then no lamiar drag loss is calculated by GASFLOW for the locations specified by this areardef definition. If areardef(12,*) < 0 Computational index for evaluating Equ. 3-11 = 0.0 for the use of local indexes to compute the mass flow rate.< 0.0 for the use of i,j,k,iblock index packing to compute the mass flow rate	yes

areardef(11,*)	64.0	If areardef(12,*) > 0 Coefficient in the laminar drag loss correlation. Can be used to model non-circular geometries. If areardef(12,*) < 0 Directions of computed mass flow rate allowed < 0.0 allows only negative flows. = 0.0 allows both positive and negative flows. > 0.0 allows only positive flows.	yes
areardef(12,*)	1.0	If areardef(12,*) > 0 Fraction of the cell-center to cell-center distance of the neighboring cells over which the laminar drag loss model should be applied. If areardef(12,*) < 0 Flow loss coefficient, Cd, in Equ. 3-11 for a given flow area, but must be < zero to activate this model; otherwise, all input using areardef is consistent with the original definitions in the 1998 GASFLOW Users Manual. For example, -1.0 is no flow loss, while -0.5 limits the flow by half.	yes
autotemp	800	Auto ignition temperature. See Section **7.4.2**	yes
autot	1.0	Automatic time-step control flag: 1.0 means ON; 0.0 means OFF (fixed time-step size). See Section 8.2.	
c1ke	1.44	Parameter for κ-ε turbulence model. See Section 7.3.2.	
c2ke	1.92	Parameter for κ-ε turbulence model. See Section 7.3.2.	
cbc(*,*)	none	Continuative boundary condition definition array. See Section 5.2.2.	
cbc(1,*)	none	Beginning i mesh index (cell face number).	
cbc(2,*)	none	Ending i mesh index (cell face number).	
cbc(3,*)	none	Beginning j mesh index (cell face number).	
cbc(4,*)	none	Ending j mesh index (cell face number).	
cbc(5,*)	none	Beginning k mesh index (cell face number).	
cbc(6,*)	none	Ending k mesh index (cell face number).	
cbc(7,*)	none	Block number (must be set to 1).	
cbc(8,*)	none	Start time (s).	
cbc(9,*)	none	End time (s).	
cflnum	0.25	Maximum CFL number.	
clength	30.48	Length scale for algebraic turbulence model. See Section 7.3.1.	
cmu	0.05	Constant for algebraic turbulence model. See Section 7.3.1.	
cmug	1.8E-04	Constant value of dynamic viscosity, to be used with muoption = 2. See Section 7.2.	
cmuke	0.09	Parameter for κ-ε turbulence model. See Section 7.3.2.	
continue_plothist		= 0, overwrite the Plotist.nc file. = 1, append new data at the correct location. = 2, append, but do NOT write a new	
cttyfreq	1	Number of cycles between printing of time-step and pressure iteration information to the cyclinfo file. (Default = 1). See Section 9.2	yes
cyl	0	Coordinate system option: 0, Cartesian; 1, cylindrical.	
delt0	0.02	Initial time increment size. See Section 8.2.	

deltmax(*)	1.0E+30	Maximum allowable time increment size. Input as pairs for interval control. See Sections 8.2 and 8.4.	yes
deltmin	1.0E-04	Minimum allowable time increment size. Run will be terminated if this exceeds the time-step size calculated by code. See Sections 8.2 and 8.4.	
epsamb0	1000	Initial turbulent dissipation rate in the ambient.	
epsi0	1.0E-05	Initial pressure iteration error criterion. See Section 8.1.	
epsimax	1.0E-03	Not used in GASFLOW-MPI. Input is still available to provide backward compatibility with old input decks.	
epsimin	1.0E-06	Not used in GASFLOW-MPI. Input is still available to provide backward compatibility with old input decks.	
epsval(*)	none	Array to store values for turbulence dissipation rate. Used with turbdef. See Section 7.3.2.	
esdef(*,*)	none	Volumetric energy source definition array. See Section 7.9	
esdef(1,*)	none	Beginning i mesh index (cell face number).	
esdef(2,*)	none	Ending i mesh index (cell face number).	
esdef(3,*)	none	Beginning j mesh index (cell face number).	
esdef(4,*)	none	Ending j mesh index (cell face number).	
esdef(5,*)	none	Beginning k mesh index (cell face number).	
esdef(6,*)	none	Ending k mesh index (cell face number).	
esdef(7,*)	none	Block number.	
esdef(8,*)	none	Volumetric energy source (ergs/s) added to the cells defined by this esdef definition.	
esdef(9,*)	none	Time (s) at which volumetric energy source begins.	
esdef(10,*)	none	Time (s) at which volumetric energy source ends.	
fandef(*,*)	none	Variable array that allows a generalized fan definition. See Section 7.8.	no
fandef(1,*)	none	Beginning i mesh index (cell face number).	no
fandef(2,*)	none	Ending i mesh index (cell face number).	no
fanded(3,*)	none	Beginning j mesh index (cell face number).	no
fandef(4,*)	none	Ending j mesh index (cell face number).	no
fandef(5,*)	none	Beginning k mesh index (cell face number).	no
fandef(6,*)	none	Ending k mesh index (cell face number).	no
fandef(7,*)	none	Block number.	no
fandef(8,*)	none	ITFAN table number for fan head versus volumetric flow (see fantb below) Maximum number allowed is 20.	no
fandef(9,*)	none	ITSPD table number for fan speed versus time (see fspdtb below) Maximum number allowed is 20.	no
fandef(10,*)	none	SPDR, rated fan blower speed,revolutions/s.	no
fandef(11,*)	none	QR, rated fan volumetric flow rate, cm^3/s. if > 0, then fan is directed in the positive coordinate direction. if < 0, then fan is directed in the negative coordinate direction.	no
fandef(12,*)	none	HR, rated fan head. if > 0, then fan table (see fantb below) is in $dynes/cm^2$. if < 0, then fan table (see fantb below) is in cm H_2O.	no
fandef(13,*)	none	SPD, fan speed at time = 0.0, revolutions/s.	no
fantb(*,*,*)	none	Volumetric flow and Fan head tables for the generalized fan model (see fandef above). See Section 7.8.	no

fantb(1,i,j)	none	Volumetric flow of table pair i for table number j, cm^3/s.	no
fantb(2,i,j)	none	Fan head of table pair i for table j.	no
fractke	0.1	Fraction of mean kinetic energy, used to calculate turbulent kinetic energy in algebraic turbulence model. See Section 7.3.1.	
fspdtb(*,*,*)	none	Fan performance tables (see fandef above). See Section 7.8.	no
fspdtb(1,i,j)	none	Time of table pair i for table number j, s.	no
fspdtb(2,i,j)	none	Fan speed of table pair i for table j, revolutions/s.	no
gasdef(*,*)	none	Gas definition array, used for defining initial and boundary conditions for fluid. See Section 5.1.1.	
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.	
gasdef(8,*)	none	Pressure (dynes/cm2) in defined volume. If gasdef(8,*) is less than zero, then the INT(ABS(gasdef(8,*)) points to the column number in the SORTAM file and the pressure will be obtained from this column in the SORTAM file. If gasdef(8,*) is less than zero and |gasdef(8,*)|is larger than 1,000,000, then it is a packed i, j, k, iblk location and the pressure will be obtained from the pressure in cell i, j, k, iblk.	
gasdef(9,*)	none	Temperature (K) in defined volume. If gasdef(9,*) is less than zero, then the INT(ABS(gasdef(9,*)) points to the column number in the SORTAM file and the temperature will be obtained from this column in the SORTAM file.	
gasdef(10,*)	none	Option flag for specification of gas composition: 1 for mass fraction, 2 for volume fraction, > 9 implies a time-dependent function for the pressure and temperature will be specified.	
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 3-2, e.g., ‘h2’.	
gasdef(14,*)	none	Mass or volume fraction of above gas species in defined volume. If gasdef(14,*) is less than zero, then the INT(ABS(gasdef(14,*)) points to the column number in the SORTAM file and the mass/ volume will be obtained from this column in the SORTAM file.	
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. If gasdef(16,*) is less than zero, then the INT(ABS(gasdef(16,*)) points to the column number in the SORTAM file and the mass/volume will be obtained from this column in the SORTAM file.	
gasdef(40,*)	none	Last index in first dimension is 40. The maximum species, (40 – 12)/2 = 14, can be defined.	

geomodel(*,*)		Geometric modeller definition array, used for defining comples geometries, initial and boundary conditions for fluid. See Section 3.3.3.	
geomodel(1,*)	none	F, the general quadratic equation defined by Equ.3-1. F > 0, obstacles outside. F = 0, walls define surface. F < 0, obstacles inside.	no
geomodel(2,*)	none	Block number that curve is applied.	no
geomodel(3,*)	none	Flag for heat transfer. For obstacles: = mobs(8,*) See Section 3.3.2. For walls: = walls(8,*) Section 3.3.1.	no
geomodel(4,*)	none	xo in Equ.3-1.	no
geomodel(5,*)	none	yo in Equ.3-1.	no
geomodel(6,*)	none	zo in Equ.3-1.	no
geomodel(7,*)	none	a in Equ.3-1.	no
geomodel(8,*)	none	b in Equ.3-1.	no
geomodel(9,*)	none	c in Equ.3-1.	no
geomodel(10,*)	none	d in Equ.3-1.	no
geomodel(11,*)	0	f in Equ.3-1.	no
geomodel(12,*)	0	g in Equ.3-1.	no
geomodel(13,*)	0	h in Equ.3-1.	no
geomodel(14,*)	0	p in Equ.3-1.	no
geomodel(15,*)	0	q in Equ.3-1.	no
geomodel(16,*)	0	r in Equ.3-1.	no
geomodel(17,*)	0	s in Equ.3-1.	no
geomodel(18,*)	0	t in Equ.3-1.	no
geomodel(19,*)	0	u in Equ.3-1.	no
geomodel(20,*)	-1.0e+50	lower x limiter.	no
geomodel(21,*)	+1.0e+50	upper x limiter.	no
geomodel(22,*)	-1.0e+50	lower y limiter.	no
geomodel(23,*)	+1.0e+50	upper y limiter.	no
geomodel(24,*)	-1.0e+50	lower z limiter.	no
geomodel(25,*)	+1.0e+50	upper z limiter.	no
geomodel(26,*)	0	= 0, activates geometric modeler to define obstacles and walls within the computational mesh. > 0, couples the geometric modeler to the gasdef input variable, where the value refers to the gasdef statement number.	no
gx	0.0	Acceleration due to gravity in the i- (x- or r-) direction (cm/s2).	
gy	0.0	Acceleration due to gravity in the j- (y- or θ-) direction (cm/s2).	
gz	0.0	Acceleration due to gravity in the k- (z-) direction (cm/s2).	
holes(*,*)	none	Variable array that puts holes into existing obstacles. See Section 3.3.4.	no
holes(1,*)	none	Beginning i mesh index (cell face number).	no
holes(2,*)	none	Ending i mesh index (cell face number).	no

holes(3,*)	none	Beginning j mesh index (cell face number).	no
holes(4,*)	none	Ending j mesh index (cell face number).	no
holes(5,*)	none	Beginning k mesh index (cell face number).	no
holes(6,*)	none	Ending k mesh index (cell face number).	no
holes(7,*)	none	Block number.	no
holes(8,*)	none	Flag for fluxing on the beginning i mesh face, holes(1,n): = -1, mixed fluxing condition allowed; = 0, implies no fluxing condition allowed; = 1, fluxing condition allowed.	no
holes(9,*)	none	Flag for fluxing on the ending i mesh face, holes(2,n): = -1, mixed fluxing condition allowed; = 0, implies no fluxing condition allowed; = 1, fluxing condition allowed.	no
holes(10,*)	none	Flag for fluxing on the beginning j mesh face, holes(3,n): = -1, mixed fluxing condition allowed; = 0, implies no fluxing condition allowed; = 1, fluxing condition allowed.	no
holes(11,*)	none	Flag for fluxing on the ending j mesh face, holes(4,n): = -1, mixed fluxing condition allowed; = 0, implies no fluxing condition allowed; = 1, fluxing condition allowed.	no
holes(12,*)	none	Flag for fluxing on the beginning k mesh face, holes(5,n): = -1, mixed fluxing condition allowed; = 0, implies no fluxing condition allowed; = 1, fluxing condition allowed.	no
holes(13,*)	none	Flag for fluxing on the ending k mesh face, holes(6,n): = -1, mixed fluxing condition allowed; = 0, implies no fluxing condition allowed; = 1, fluxing condition allowed.	no
ibb	1	Boundary condition indicator for –k (bottom) mesh boundary. See Section 5.2.1. 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: = +1, Hydrogen burn model on (forward reaction only); = 0, Hydrogen burn model off (default); = -1, Hydrogen burn model on (both forward and reverse reactions) See Section 7.4. = 4, Hydrogen combustion models using reaction progress variable.	yes

ibw	1	Boundary condition indicator for –i (west) mesh boundary. See ibb description.	
icopt	0	Method for computing the specific heat cv : = 0, derivative of internal energy function; = 1, constant value (T < 500 K) ; = 2, second-degree polynomial (T < 750 K); = 3, Gordon & McBride approximation.	no
idetchar	0	Flag to control the sigma and 7 lamda criteria plots defined by the iroomdef option. See Section 9.1.6 = 0, no plots (default). = 1, only 7 lamda plots. = 2, 7 lamda and sigma plots.	yes
idiffme	0	Option flag for mass and energy diffusion: 0 means OFF; 1 means ON. See Section 7.2.	
idiffmom	0	Option flag for momentum diffusion: 0 means OFF; 1 means ON. See Section 7.2.	
ieopt	1	Polynomial fit for specific internal energy: = 1, first-degree polynomial; = 2, second-degree polynomial.	no
ifvl	0	Option flag to activate van Leer advection scheme: 0 means donor-cell method; 1 means van Leer method. See Section 8.3.	
ignitaut	0	Flag for activating the spontaneous or auto ignition burn model. See Section **7.4.2**.	yes
ignitdef(*,*)	none	Ignitor definition array. See Section **7.4.2**.	
ignitdef(1,*)	none	i mesh index (cell centered) of ignitor	yes
ignitdef(2,*)	none	j mesh index (cell centered) of ignitor.	yes
ignitdef(3,*)	none	k mesh index (cell centered) of ignitor.	yes
ignitdef(4,*)	none	Block number of ignitor.	yes
ignitdef(5,*)	none	= 0, for continuously operating ignitors; > 0, for periodic sparking ignitors.	yes
ihystat	0	Option flag for imposing hydrostatic pressure gradient in fluid cells according to acceleration components gx, gy, and gz: 0 means OFF; 1 means ON. See Section 7.1.	
impsprayd	0	Spray droplet advection algorithm options: = 0; Explicit droplet advection algorithm (default), = 1; Implicit droplet advection algorithm, See Section 7.10.	yes
iobpl	1	Flag to plot obstacles = 0, obstacles not plotted. = 1, obstacles plotted.	yes
iroomdef(*,*)	none	Defines i, j, k region for evaluation of detonation potential from the d/7λ criterion. See Section 9.1.5. For each iroomdef, the following information is plotted from unit 11: burnable cloud size d, average hydrogen density of burnable cloud, detonation cell size for average hydrogen density in the mixture, d/7λ.	yes
iroomdef(1,*)	none	Beginning i mesh index (cell face number).	no
iroomdef(2,*)	none	Ending i mesh index (cell face number).	no
iroomdef(3,*)	none	Beginning j mesh index (cell face number).	no
iroomdef(4,*)	none	Ending j mesh index (cell face number).	no

iroomdef(5,*)	none	Beginning k mesh index (cell face number).	no
iroomdef(6,*)	none	Ending k mesh index (cell face number).	no
iroomdef(7,*)	none	Block number.	no
iroomdef (8,*)	none	Actual room number: = + implies positive volume; = – implies negative volume (i.e., subtracts obstacles).	no
ispray	0	Flag to activate the Spray model = 0, Spray model not activated = 1, Spray model activated See Section 7.10	no
isourcexi	2	Flag to activate the model for source term when iburn = 4: = 0, source term off; = 1, Arrhenius rate model; = 2, model based on progress variable gradient (default); = 3, under development; = 4, under development; = 5, Eddy dissipation model.	
itdowndt	50	Number of pressure iterations at which time-step size is reduced by 2%. If the number of pressure iterations is larger than itdowndt, then the time-step size is reduced by 2%. See Section 8.2.	
itmax	20	Maximum number of pressure iterations per cycle. See Section 8.2.	
itopt	0	Method for computing the molecular transport properties, dynamic viscosity, thermal conductivity, and mass diffusion coefficient: = 0, nonmechanistic using input data (then muoption applies); = 1, constant value (T < 500 K); = 2, linear (T < 1000 K); = 3, quadratic (T < 3000 K); = 4, cubic (T < 3000 K); = 5, quartic (T < 5000 K); = 6, functions are evaluated from the CHEMKIN library.	no
itpcl	1	Total number of particle classes.	
itpsz(mxpcll)	none	Number of initial particle sizes in each particle class.	
ittyfreq	20	Frequency of printing iteration/cycle information to terminal. See Section 9.2.	
itupdt	100	Number of pressure iterations at which time-step size is increased by 2%. If the number of pressure iterations is larger than itdowndt, then the time-step size is reduced by 2%. See Section 8.2.	
iturbflame	4	= 0, use laminar flame speed; = 1, use the correlation S_T=S_L (1+$\sqrt{\kappa}$/S_L) ; = 2, use Kawanabe correlation; = 3, use Peters correlation; = 4, use Zimont correlation (default); = 5, use Zimont-Mesheriakov correlation; = 6, use Schmidt correlation.	
iwallfunc	0	= 0; no wall functions are active (default value) = 1; no-slip conditions must to active, assumes smooth walls = 2; no-slip conditions must to active, assumes rough walls and krough must be specified See Sections 5.2.1, 0. and 7.5	no

krough	0.1	Wall roughness scale (cm)	no
lpr	1	Control flag for printing to gfout: = 0 results in reduced output sent to gfout file; = 1 results in full output sent to gfout file.	
mat(*)	none	List of gas species symbols (enclosed in single quotes) for specifying gas components. See Section 4.1.	
maxcyc	10	Maximum number of cycles allowed. See Section 8.2. Negative maxcyc is not supported in GASFLOW-MPI anymore.	
mbc(*,*)	none	Mass flow rate boundary condition array. See Section 5.2.2.	
mbc(1,*)	none	Beginning i mesh index (cell face number).	
mbc(2,*)	none	Ending i mesh index (cell face number).	
mbc(3,*)	none	Beginning j mesh index (cell face number).	
mbc(4,*)	none	Ending j mesh index (cell face number).	
mbc(5,*)	none	Beginning k mesh index (cell face number).	
mbc(6,*)	none	Ending k mesh index (cell face number).	
mbc(7,*)	none	Block number.	
mbc(8,*)	none	Element of mvalue that will define the velocity value. Also used as a flag to determine functional dependence of specified velocity.	
mbc(9,*)	none	Start time (s).	
mbc(10,*)	none	End time (s).	
mfunc(*,*)	none	Set of constants for the mass flow rate boundary condition time-dependent functions. See Section 5.2.2.	
mfunc(1,*)	none	The constant b in the following functions: f = a + b * ts f = a + b * ts**2 f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
mfunc(2,*)	none	The constant c in the following functions: f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
mfunc(3,*)	none	The constant d in the following functions: f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
mfunc(4,*)	none	The constant e in the following function: f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
mobs(*,*)	none	Mesh obstacle definition array. See Section 3.3.2.	
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.	

mobs(8,*)	none	Material identification number: 0 disregard heat conduction i from 1 to mpreset use table 6-1. If i > mpreset use table for element-by-element input for thermal conductivity and rho*cp See also wltabwall, rcptabwall (i > mpreset used for layered structures). Only used if ihtflag = 1. See Section 6.2.	
msp	1	Total number of simulation particles allowed in this calculation.	
mtab(*,*,*)	none	Time-dependent mass flow rate boundary condition table. See Section 5.2.2.	
mtab(1,ip,it)	none	Time (s) for the ipth point in the itth table.	
mtab(2,ip,it)	none	Mass flow rate (gm/s) for the ipth point in the itth table.	
muoption	0	Controls how to determine transport properties when itopt is set to 0: = 0, properties are computed from the local density and input kinematic viscosity, nu, and the Prandtl and Schmidt numbers with prandtl = 1.0 and schmidt = 1.0; = 1, properties are computed from the local density and input kinematic viscosity, nu, and constant values for the thermal conductivity and diffusion coefficient; = 2, properties are computed from constant values for the dynamic viscosity, thermal conductivity, and diffusion coefficient.	no
mvalue(*)	none	Array to store values for mass flow rates, used with mbc. See Section 5.2.2.	
nobsgeo	0	Memory allocation variable for Obstacles. See Section 3.3.3	no
nrsdump	0	Number that appears in the name of the restart dump file to be read in. See Section 9.2. Negative or zero values indicate a new run.	
nslipdef(*,*)	none	Array to define no-slip surfaces. See Section 5.2.2.	
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,υ (cm2/s). See Section 7.2.	
nwallsgeo	0	Memory allocation variable for Walls. See Section 3.3.3	no
pamb0	1013250.0	Reference pressure when used with ihystat, xamb0, yamb0, and zxamb0. See Section 7.1. Ambient pressure value for plotting purposes. See Section 9.1.1.	
pbc(*,*)	none	Array for defining pressure boundary conditions. See Section 5.2.2.	
pbc(1,*)	none	Beginning i mesh index (cell face number).	
pbc(2,*)	none	Ending i mesh index (cell face number).	
pbc(3,*)	none	Beginning j mesh index (cell face number).	

pbc(4,*)	none	Ending j mesh index (cell face number).	
pbc(5,*)	none	Beginning k mesh index (cell face number).	
pbc(6,*)	none	Ending k mesh index (cell face number).	
pbc(7,*)	none	Block number (must be set to 1).	
pbc(8,*)	none	Start time (s).	
pbc(9,*)	none	End time (s).	
pfunc(*,*)	none	Set of constants for the pressure time-dependent functions. See Section 5.1.1.	
pfunc(1,*)	none	The constant b in the following functions: f = a + b * ts f = a + b * ts**2 f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
pfunc(2,*)	none	The constant c in the following functions: f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
pfunc(3,*)	none	The constant d in the following functions: f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
pfunc(4,*)	none	The constant e in the following function: f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
pltdt	1.0	Time interval (s) between successive sets of 1D profile, 2D contour, and 2D and 3D velocity vector plots. Input as pairs for interval control. See Sections 8.4 and 9.1.	yes
plt3ddt	0	Time interval (s) between successive sets of 3D plots. Input as pairs for interval control. See Sections 8.4 and 9.1.	
prandtl	1.0	Fluid Prandtl number, used to determine thermal diffusivity. See Section 4.2.	
prtdt	1000	Time interval (s) between printing of the fluid solution field (all velocity components, pressure, and density) to file gfout. Input as pairs for interval control. See Sections 8.4 and 9.2.	
ptab(*,*,*)	none	Time-dependent pressure table. See Section 5.1.1.	
ptab(1,ip,it)	none	Time (s) for the ipth point in the itth table.	
ptab(2,ip,it)	none	Pressure (dynes/cm2) for the ipth point in the itth table.	
rupdkdef(*,*)	none	Defines rupture disk areas on plane surfaces.	no
rupdkdef(1,*)	none	Beginning i mesh index (cell face number).	no
rupdkdef(2,*)	none	Ending i mesh index (cell face number).	no
rupdkdef(3,*)	none	Beginning j mesh index (cell face number).	no
rupdkdef(4,*)	none	Ending j mesh index (cell face number).	no
rupdkdef(5,*)	none	Beginning k mesh index (cell face number).	no
rupdkdef(6,*)	none	Ending k mesh index (cell face number).	no
rupdkdef(7,*)	none	Block number.	no

rupdkdef(8,*)	none	Fraction of geometric flow area available for fluid flow after the rupture disk has failed.	no
rupdkdef(9,*)	none	If rupdkdef(9,*) < 1.e+10: failure criterion provided as a pressure difference in dynes/cm^2. If 1.e+10 <rupdkdef(9,*)< 1.e+20: failure criterion provided as 1.e+10 times absolute failure pressure in dynes/cm^2. If 1.e+20 <rupdkdef(9,*)< 1.e+30: failure criterion provided as 1.e+20 times gas temperature in K.	no
sclamb0	none	Initial turbulent length scale in the ambient.	
sortami	0	Flag for reading the sortam file = 0, no sortam file to read (default). = 1, Adjust time step to coincide exactly with the end points of the piece wise continuous intervals. = 2, No adjustment to time step. See Section 5.2.4.2.	yes
spraydef(*,*)	none	Spray specification statement. See Section 7.10.	yes
spraydef(1,*)	none	Beginning i mesh index (cell face number).	yes
spraydef(2,*)	none	Ending i mesh index (cell face number).	yes
spraydef(3,*)	none	Beginning j mesh index (cell face number).	yes
spraydef(4,*)	none	Ending j mesh index (cell face number).	yes
spraydef(5,*)	none	Beginning k mesh index (cell face number).	yes
spraydef(6,*)	none	Ending k mesh index (cell face number).	yes
spraydef(7,*)	none	Block number.	yes
spraydef(8,*)	none	Spray mass flow rate (g/s).	yes
spraydef(9,*)	none	Spray temperature (K). If value < 0, then the tfunc capabilities may be used. The 1st, or ones digit is ignored, but the 2nd digit, the tens digit, indicates the function type (ifunc), and the 3rd digit, the hundreds digit, indicates the constants, or table number, in ifunc. See Section 5.1.1.	yes
spraydef(10,*)	none	Spray droplet diameter (cm).	yes
spraydef(11,*)	none	Time (s) at which “spray definition” begins.	yes
spraydef(12,*)	none	Time (s) at which “spray definition” ends.	yes
spraygdef(*,*)	none	Spray specification as an extension for the gasdef statement. See Section 7.10.	yes
spraygdef(1,*)	none	Spray temperature (K). See Section 7.10.	yes
spraygdef(2,*)	none	Spray droplet diamater (cm). See Section 7.10.	yes
spxigdef(*,*)	none	Characteristics of ignitor. See Section **7.4.2**.	yes
spxigdef(1,*)	none	Time when ignitor is first turned on.	yes
spxigdef(2,*)	none	Time when ignitor is turned off.	yes
spxigdef(3,*)	none	Time interval between sparks.	yes
spxigdt	0.001	Time of the spark duration. See Section **7.4.2**.	yes
subsodef(*,*)	none	Option to subtract a mass and energy source from the global mass and energy balance. See Section 5.2.3.	yes
subsodef(1,*)	none	Beginning i mesh index (cell face number).	yes
subsodef(2,*)	none	Ending i mesh index (cell face number).	yes

subsodef(3,*)	none	Beginning j mesh index (cell face number).	yes
subsodef(4,*)	none	Ending j mesh index (cell face number).	yes
subsodef(5,*)	none	Beginning k mesh index (cell face number).	yes
subsodef(6,*)	none	Ending k mesh index (cell face number).	yes
subsodef(7,*)	none	Block number.	yes
tddt	10	Time interval (s) at which restart dump file are written. Input as pairs for interval control. See Sections 8.4 and 9.4.	yes
tfunc(*,*)	none	Set of constants for the temperature time-dependent functions. See Section 5.1.1.	
tfunc(1,*)	none	The constant b in the following functions: f = a + b * ts f = a + b * ts**2 f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
tfunc(2,*)	none	The constant c in the following functions: f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
tfunc(3,*)	none	The constant d in the following functions: f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
tfunc(4,*)	none	The constant e in the following function: f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
tshift_sortam	0.0	Time shift for reading the sortam file = 0.0, no time shift (default). > 0.0, time shift that is used when reading the sortam file. See Section 5.2.4.1.	yes
tkeamb0	10.0	Initial turbulent kinetic energy in the ambient.	
tkeval(*)	none	Array to store values for turbulent kinetic energy. Used with turbdef. See Section 0.	
tmodel	'none'	Turbulence model options: 'none' means no turbulence model; 'alg' means algebraic model (see Section 7.3.1); 'ke' means k-ε model (see Section 7.3.2). ‘sstko’ means SST k-ω turbulence model.	
trange	'low'	Range of applicability of fits to internal energy: 'low', for temperatures up to 3000 K; 'high', for temperatures up to 5000 K.	no
ttab(*,*,*)	none	Time-dependent temperature table. See Section 5.1.1.	
ttab(1,ip,it)	none	Time (s) for the ipth point in the itth table.	
ttab(2,ip,it)	none	Temperature (K) for the ipth point in the itth table.	
turbdef(*,*)	none	Array to define turbulence quantities for model. See Section 7.3.2.	
turbdef(1,*)	none	Beginning i mesh index (cell face number).	
turbdef(2,*)	none	Ending i mesh index (cell face number).	
turbdef(3,*)	none	Beginning j mesh index (cell face number).	

turbdef(4,*)	none	Ending j mesh index (cell face number).	
turbdef(5,*)	none	Beginning k mesh index (cell face number).	
turbdef(6,*)	none	Ending k mesh index (cell face number).	
turbdef(7,*)	none	Block number (must be set to 1 for GASFLOW-MPI).	
turbdef(8,*)	none	Integer pointer to location in tkeval array for value of turbulent kinetic energy.	
turbdef(9,*)	none	Integer pointer to location in epsval 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 Section 5.1.2.	
vbc(*,*)	none	Array for defining velocity boundary conditions. See Section 5.2.2.	
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 for GASFLOW-MPI).	
vbc(8,*)	none	Element of vvalue that will define the velocity value. Also used as a flag to determine functional dependence of specified velocity.	
vbc(9,*)	none	Start time (s).	
vbc(10,*)	none	End time (s).	
velmx	2.0	Scaling factor for velocity vector plots (multiplies internally scaled vectors).	
vfunc(*,*)	none	Set of constants for the velocity boundary condition time-dependent functions. See Section 5.2.2.	
vfunc(1,*)	none	The constant b in the following functions: f = a + b * ts f = a + b * ts**2 f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
vfunc(2,*)	none	The constant c in the following functions: f = a + b * ts**2 + c * ts**3 f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b * exp (tr / c) f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
vfunc(3,*)	none	The constant d in the following functions: f = a + b * ts**2 + c * ts**3 +d*t**4 f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	
vfunc(4,*)	none	The constant e in the following function: f = a + b*cos(tr*pi/c) + d*sin(tr*pi/e)	

vi	0.0	Initial velocity in j-direction (cm/s). See Section 5.1.2.	
vtab(*,*,*)	none	Time-dependent velocity boundary condition table. See Section 5.2.2.	
vtab(1,ip,it)	none	Time (s) for the ipth point in the itth table.	
vtab(2,ip,it)	none	Velocity (cm/s) for the ipth point in the itth table.	
vvalue(*)	none	Array to store values for velocity, used with vbc. See Section 5.2.2.	
walls(*,*)	none	Wall definition array. See Section 3.3.1.	
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 for GASFLOW-MPI).	
walls(8,*)	none	Integer to identify the type (material and effective thickness) of wall through the walldef array. Only used if ihtflag = 1. A value of 0 takes this wall out of heat-transfer considerations even when ihtflag is set to 1. See Section 6.1.	
wi	0.0	Initial velocity in k-direction (cm/s). See Section 5.1.2.	
xamb0	0.0	Used with ihystat, pamb0, yamb0, and zamb0 to precompute a hydrostatic pressure gradient. See Section 7.1	no
xecoef	0.0	Xenon decay coefficient [ergs/gm-s]. See Section 7.7.3.	yes
xepowert	0.0	Power (-1 <= xepowert <= 1) for the (t-xet0) term in Equation 7-1. Note that the absolute value is actually used in the evaluation of the equation. See Section 7.7.3.	yes
xetau0	1.0	Xenon decay time constant [s]. See Section 7.7.3.	yes
xet0	0.0	Xenon decay time shift [s]. See Section 7.7.3.	yes
xi_ignitdef(*,*)	none	Ignition model for reaction progress variable.	yes
xi_ignitdef(1,*)	none	beginning i mesh index (cell face number).	yes
xi_ignitdef(2,*)	none	ending i mesh index (cell face number).	yes
xi_ignitdef(3,*)	none	beginning j mesh index (cell face number).	yes
xi_ignitdef(4,*)	none	ending j mesh index (cell face number).	yes
xi_ignitdef(5,*)	none	beginning k mesh index (cell face number).	yes
xi_ignitdef(6,*)	none	ending k mesh index (cell face number).	yes
xi_ignitdef(7,*)	none	block number (must be 1 for GASFLOW-MPI)	yes
xi_ignitdef(8,*)	none	time when ignitors are first turned on (s).	yes
xi_ignitdef(9,*)	none	time of the ignition duration (s).	yes
yamb0	0.0	Used with ihystat, pamb0, xamb0, and zamb0 to precompute a hydrostatic pressure gradient. See Section 7.1.	no
zamb0	0.0	Used with ihystat, pamb0, xamb0, and yamb0 to precompute a hydrostatic pressure gradient. See Section 7.1.	no
zeroddef(*,*)	none	Zero diffusion boundary at source reservoirs (plane surface). See Section 5.2.3	yes
zeroddef(1,*)	none	Beginning i mesh index (cell face number).	yes
zeroddef(2,*)	none	Ending i mesh index (cell face number).	yes

zeroddef(3,*)	none	Beginning j mesh index (cell face number).	yes
zeroddef(4,*)	none	Ending j mesh index (cell face number).	yes
zeroddef(5,*)	none	Beginning k mesh index (cell face number).	yes
zeroddef(6,*)	none	Ending k mesh index (cell face number).	yes
zeroddef(7,*)	none	Block number (must be set to 1 for GASFLOW-MPI).	yes

B. Summary of Variables in NAMELIST Group meshgn

(Refer to Section 3 for detailed explanation of variables.)

Variable	Default	Description
dxmn(*)	none	Array to store minimum cell size (cm) in i-direction for each submesh. Used for automatic mesh generation.
dymn(*)	none	Array to store minimum cell size (cm) in j-direction for each submesh. Used for automatic mesh generation.
dzmn(*)	none	Array to store minimum cell size (cm) in k-direction for each submesh. Used for automatic mesh generation.
iblock	none	Block identification number.
nkx	none	Number of submesh in i-direction.
nky	none	Number of submesh in j-direction.
nkz	none	Number of submesh in k-direction.
nxl(*)	none	Array to store number of cells on the –i side for each submesh. Used for automatic mesh generation.
nxr(*)	none	Array to store number of cells on the +i side for each submesh. Used for automatic mesh generation
nyl(*)	none	Array to store number of cells on the –j side for each submesh. Used for automatic mesh generation.
nyr(*)	none	Array to store number of cells on the +j side for each submesh. Used for automatic mesh generation.
nzl(*)	none	Array to store number of cells on the –k side for each submesh. Used for automatic mesh generation.
nzr(*)	none	Array to store number of cells on the +k side for each submesh. Used for automatic mesh generation.
xc(*)	none	Array to store location (cm) of edge of smallest cell in i-direction. Used for automatic mesh generation.
xgrid(*)	none	Array to store mesh coordinates in i-direction. Used for direct mesh definition.
xl(*)	none	Array to store 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.
yc(*)	none	Array to store location (cm) of edge of smallest cell in j-direction. Used for automatic mesh generation.
ygrid(*)	none	Array to store mesh coordinates in j-direction. Used for direct mesh definition.
yl(*)	none	Array to store j-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.
zc(*)	none	Array to store location (cm) of edge of smallest cell in k-direction. Used for automatic mesh generation.
zgrid(*)	none	Array to store mesh coordinates in k-direction (cm). Used for direct mesh definition.
zl(*)	none	Array to store k-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.

C. Summary of Variables in NAMELIST Group rheat

Variable	Default	Description	Restart
cbulkrlx	0.1	Relaxation factor for bulk condensation (s–1). See Section 7.5.	yes
cfilmdef(*,*)	none	Area that defines the sump surface with constant film thickness. This area must also be defined as a horizontal wall with free space underneath, i.e., define as holes. See Section 7.7.1.	yes
cfilmdef(1,*)	none	Beginning i mesh index (cell face numbers).	yes
cfilmdef(2,*)	none	Ending i mesh index (cell face numbers).	yes
cfilmdef(3,*)	none	Beginning j mesh index (cell face numbers).	yes
cfilmdef(4,*)	none	Ending j mesh index (cell face numbers).	yes
cfilmdef(5,*)	none	Beginning k mesh index (cell face numbers).	yes
cfilmdef(6,*)	none	Ending k mesh index (cell face numbers).	yes
cfilmdef(7,*)	none	Block number (must be set to 1).	yes
cfilmdef(8,*)	none	> 0, Constant film thickness (must be less than 100 cm) < 0, Sump number (must be a negative integer)	yes
clamda	-1.0	radiation mean free path (cm). If clamda > 0, then the user is specifying a constant radiation mean free path; otherwise, the mean free path is computed using the method of Lechner. See Section 7.5	yes
cons1	3.6	Multiplication factor for the heat-transfer coefficient (applies to convection and condensation/evaporation).	yes
crelax	0.01	Relaxation factor for depleting liquid H2O (s–1) . See also iliq.	yes
dxslabc	0.0	> 0. results in variable mesh spacing for all concrete slabs. The first Δx for each concrete slab will be dxslabc; = 0, use equal mesh spacing in concrete slab.	no
dxlabs	0.0	> 0. results in variable mesh spacing for all steel slabs. The first Δx for each steel slab will be dxslabs; = 0, use equal mesh spacing in steel slab.	no
emismat(*)	1.0	material emissivity – see Table 6-1 in Section 6.2. Each material in the heat transfter material data base has a default emismat(*) = 1.	no
epsrad	10-6	Convergence criteria for the solution of the radiation heat transfer equation. See Section 7.5	yes
etainp	1.0	Input PAR efficiency for the standard Areva PARs 5<=type<=9, etainp has no effect on the other PAR types which are defined specifically with the other PAR correlations.	
filmth	0.5	> 0, maximum film thickness from condensation (cm); = 0, no vaporization allowed; < 0, a liquid film is initiated on all slab, wall, and sink surfaces with a thickness of |filmth|.	no
hslablin	1000	Heat-transfer coefficient on slab BC#1, used in evaluating the linear slab cell temperature profile with tslabbc as slab surface temperature on the positive side BC#2 and tn0(m) or tslab0 as fluid temperature on BC#1.	
iareabal	0	Flag to activate heat transfer rebalancing option, must be > 0 to activate. See Section 6.8	no
icond	1	Flag for condensation on structures: 1 means ON; 0 means OFF.	yes

ienh	1	Film-model enhancement from Bird Stewart Lightfoot: 1 means ON; 0 means OFF; so far only for film condensation.	yes
ifcca	1	Flag for Chilton-Colburn analogy: 1 means ON; 0 means OFF. See Section 7.5.	yes
ihtcoef	4	= 0, constant heat transfer coefficient set equal to cons1. (= 1.0; default value in ergs/K-s). = 1, laminar heat transfer coefficient assumes smooth walls. = 2, von Karman analogy heat transfer coefficient. Assumes smooth walls. = 3, Extended von Karman analogy heat transfer coefficient. Assumes smooth walls. = 4, Default model, Reynolds analogy with Colburn correction for the heat transfer coefficient. Assumes smooth walls. = 5, Reynolds analogy with Colburn correction for the heat transfer coefficient. Assumes rough walls and krough must be specified krough =0.1; default value surface roughness depth (cm). = 6, Constant heat flux set equal to cons1. cons1 (= 1.0; default value in ergs/cm^2·s). = 7, Similar velocity-temperature profiles. Temperatures maybe directly scaled using cons1. cons1 (= 1.0; default value). = 8, Heat transfer coefficient = 0, same as ihtcoef = 0 with cons1 = 0. See Section 7.5.	yes
ihtflag	0	Option flag to activate heat transfer and steam condensation: 1 means ON; 0 means OFF. See Section 7.5.	no
iliq	1	Flag for depletion of H2O liquid = 0, Model off. = +1, Parametric or relaxation model on, see crelax. = -1, Mechanistic droplet depletion model. = -2, Both Parametric and mechanistic models. See Section 7.10.	yes
irad	0	= 0; radiation model is not activated (default value). >= 1; radiation transport equation model solved with a preconditioned conjugate gradient method. See Section 7.5.	no
ircomb	0	= 0, no recombiners; = 1, use recombiners.	yes
islablin	0	Flag to set an approximate linear initial temperature profile in each slab boundary cell, based on the temperature of slab BC#2 and fluid cell contiguous with BC#1: = 0 implies no linear profile (i.e., slab initial temperature is constant and equals fluid tn0(m) or tslab0); = 1 implies a linear profile with fluid temperature on negative side being tn0(m) or tslab0. Option applies to all slabs in one block in the same way.	
itmaxr	500	Maximum number of iterations allowed for the solution of the radiation heat transfer equation. See Section 7.5	yes
iwh2o	0	Flag for vaporization of film: 1 means ON; 0 means OFF.	yes
lpanelout	1	Generate panel output.	
lprarea	0	Flag for output of heat structure surfaces into file areas.	yes
matbdy	1	Material type (uniform) for the mesh boundary: = 0 results in no boundary slabs with an adiabatic boundary of the computational mesh.	no

matdef		Resets Material number of selected slabs. See Section 6.9	no
matpanel		=m1,m2,…, materials for which panel output is generated (default panels generated for all materials).	
mpreset	6	Materials 1 to mpreset are automatically loaded from property library. Tabular input must be specified for materials mpreset+1 to ntotmat.	
nhtesink	2	Number of heat-conduction elements in a sink heat structure (< 100).	no
nhteslab	2	Number of heat-conduction elements in a slab heat structure (< 100).	no
nhtewall	2	Number of heat-conduction elements in a wall heat structure (< 100).	no
nsumppts	none	|nsumppts| = number of components in temperature vs. time table for the sump (see also sumptemp and sumptime): > 0, sumptemp is in kelvin; < 0, sumptemp is in degrees centigrade (Celsius).	yes
ntotmat	6	Total number of structure materials (<=20). Read new properties only for materials mpreset + imat to ntotmat because components 1 to mpreset are automatically defined.	
rcombdef(*,*)	none	Array for the definition of recombiner location. See Section 7.7.2.	yes
rcombdef(1,*)	none	Beginning i mesh index of recombiner cells (cell face numbers).	yes
rcombdef(2,*)	none	Ending i mesh index of recombiner cells (cell face numbers).	yes
rcombdef(3,*)	none	Beginning j mesh index of recombiner cells (cell face numbers).	yes
rcombdef(4,*)	none	Ending j mesh index of recombiner cells (cell face numbers).	yes
rcombdef(5,*)	none	Beginning k mesh index of recombiner cells (cell face numbers).	yes
rcombdef(6,*)	none	Ending k mesh index of recombiner cells (cell face numbers).	yes
rcombdef(7,*)	none	Block number (must be set to 1 for GASFLOW-MPI).	yes
rcombdef(8,*)	none	Recombiner area specified for region i, j, k (for recombiner type 1 or 2); projected flow area through the recombiner (for recombiner type 3 or 4). (not used for recombiner type 5 to 9)	yes
rcombdef(9,*)	none	Recombiner type: = 1, NIS granulate recombiner (active ventilation option). = 2, Siemens recombiner (active ventilation option). = 3, Siemens Reco FR90/1-100 with reaction rate from GX tests with buoyancy calculated from reaction rate. = 4, GRS type Reco (not currently in the code). = 5, Siemens Reco type FR90/1-100 with reaction rate from Siemens. = 6, Siemens Reco type FR90/1-150. = 7, Siemens Reco type FR90/1-320. = 8, Siemens Reco type FR90/1-960. = 9, Siemens Reco type FR90/1-1500.	yes
rcombdef(10,*)	-1	Number of levels below recombiner zone where hydrogen is sampled.	yes
rcombdef(11,*)	0.008/ 0.031	Threshold of hydrogen concentration above which the recombiner starts operating (default of 0.008 is for NIS reco, 0.031 for Siemens reco with GX corr. and 0.02 for Siemens reco with Siemens correlations). If negative number is used for rcombdef(11,*) absolute value specifies the starting time of the recombiner operation.	yes
rcombdef(12,*)	1800/ 100	Time constant to approach asymptotic flow rate in recombiner (default 1800 s for NIS reco, 100 s for Siemens reco)	yes

rcptabsink(*,*)		Array that specifies rho*cp in sinks element by element (nhtesink components must be filled).	
rcptabsink(*,imat)		rho*cp table for material imat for all sink elements: imat must be > mpreset; materials 1 to mpreset are automatically filled with constant values from library in table 6-1.	
rcptabslab(*,*)		Array that specifies rho*cp in slabs element by element (nhteslab components must be filled).	
rcptabslab(*,imat)		rho*cp table for material imat for all slab elements: imat must be > mpreset; materials 1 to mpreset are automatically filled with constant values from library in table 6-1.	
rcptabwall(*,*)		Array that specifies rho*cp in walls element by element (nhtewall components must be filled).	
rcptabwall(*,imat)		rho*cp table for material imat for all wall elements: imat must be > mpreset; materials 1 to mpreset are automatically filled with constant values from library in table 6-1.	
refasink(*)	none	Reference area for sinks by material type. See Section 6.8.	no
refaslab(*)	none	Reference area for slabs by material type. See Section 6.8.	no
refawall(*)	none	Reference area for walls by material type. See Section 6.8.	no
rholiqmx	2.e-6	Lower liquid density for liquid depletion (rainout) model (g/cm3).	yes
sinkdef(*,*)	none	Array to define distributed heat sinks. See Section 6.3.	no
sinkdef(1,*)	none	Beginning i mesh index (cell face number).	no
sinkdef(2,*)	none	Ending i mesh index (cell face number).	no
sinkdef(3,*)	none	Beginning j mesh index (cell face number).	no
sinkdef(4,*)	none	Ending j mesh index (cell face number).	no
sinkdef(5,*)	none	Beginning k mesh index (cell face number).	no
sinkdef(6,*)	none	Ending k mesh index (cell face number).	no
sinkdef(7,*)	none	Block number.	no
sinkdef(8,*)	none	Material identification number: i from 1 to mpreset use table 6-1. If i > mpreset use table for element-by-element input for thermal conductivity and rho*cp See also wltabwall, rcptabwall (i > mpreset used for layered structures).	no
sinkdef(9,*)	none	Total material volume (cm3).	no
sinkdef(10,*)	none	Average material thickness (cm).	no
sinkdef(11,*)	0.0	Sets BC for fluid/sink surface: = 0.0 implies sink fluid heat exchange; > 0.0 implies a given sink temperature on the fluid side (i.e., a cooler).	no
sinkdef(12,*)	none	Sets BC at the sink centerline: > 0.0 implies a constant temperature boundary; < 0.0 implies an adiabatic temperature boundary.	
sinkdef(13,*)	0.0	> 0.0 δx for the fluid/surface node of the sink with dynamic mesh expansion for heat-conduction nodes; = 0.0 implies uniform mesh spacing for heat conduction.	

sinkdef(14,*)	0	Flag for further specification of BC on negative side: = 0, no further modification; i > 0 gives table number from surftab that specifies time-dependent surface temperature, the initial temperature at t = 0 is taken from sinkdef(11,*); = –1 applies constant heat flux from sinkdef(16,*) and/or heat transfer with coefficient sinkdef(17,*) and fluid temperature sinkdef(11,*) on negative side (positive heat flux means add energy to wall).	
sinkdef(15,*)	0	Flag for further specification of BC on positive side (centerline) of slab: = 0, no further modification; i > 0 gives table number from surftab that specifies time-dependent surface temperature, the initial temperature at t = 0 is taken from sinkdef(12,*); = –1 applies constant heat flux from sinkdef(16,*) and/or heat transfer with coefficient sinkdef(17,*) and fluid temperature sinkdef(12,*) on positive side (positive heat flux means add energy to wall).	
sinkdef(16,*)	0	Heat flux (ergs/cm2·s) applied as BC by sinkdef(14,*) or sinkdef(15,*).	
sinkdef(17,*)	0	Heat-transfer coefficient (ergs/cm2·s·K) applied as BC by walldef(7,*) or walldef(8,*).	
slabdef(25,*)		Sets tslab0, tslabbc, and slabthk for specific boundaries and blocks. See Section 6.4.	
slabdef(1,*)		Block number (must be set to 1 for GASFLOW-MPI).	
slabdef(2,*)		Thickness of slabs on east boundary.	
slabdef(3,*)		Thickness of slabs on west boundary.	
slabdef(4,*)		Thickness of slabs on north boundary.	
slabdef(5,*)		Thickness of slabs on south boundary.	
slabdef(6,*)		Thickness of slabs on top boundary.	
slabdef(7,*)		Thickness of slabs on bottom boundary.	
slabdef(8,*)		Initial temperature for slabs on east boundary.	
slabdef(9,*)		Initial temperature for slabs on west boundary.	
slabdef(10,*)		Initial temperature for slabs on north boundary.	
slabdef(11,*)		Initial temperature for slabs on south boundary.	
slabdef(12,*)		Initial temperature for slabs on top boundary.	
slabdef(13,*)		Initial temperature for slabs on bottom boundary.	
slabdef(14,*)		Temperature BC for slabs on east boundary.	
slabdef(15,*)		Temperature BC for slabs on west boundary.	
slabdef(16,*)		Temperature BC for slabs on north boundary.	
slabdef(17,*)		Temperature BC for slabs on south boundary.	
slabdef(18,*)		Temperature BC for slabs on top boundary.	
slabdef(19,*)		Temperature BC for slabs on bottom boundary.	
slabdef(20,*)		Material type for slabs on east boundary.	
slabdef(21,*)		Material type for slabs on west boundary.	
slabdef(22,*)		Material type for slabs on north boundary.	
slabdef(23,*)		Material type for slabs on south boundary.	
slabdef(24,*)		Material type for slabs on top boundary.	
slabdef(25,*)		Material type for slabs on bottom boundary.	

slabthk	100.0	Thickness (cm) of a slab heat structure. Also used to determine whether obstacle is a wall or slab heat structure. See Section 6.1. Still determines whether inner obstacles are slabs or walls; slabdef statements can overwrite only the thickness of the boundary slabs.	no
sumpengy(*)	none	Energy source associated with sumpmas, which is the direct energy source in ergs/s or ergs/g-s, depending upon the sign of sumptime. See Section 7.7.1.	yes
sumpmas(*)	none	Mass source associated with sumpengy, which is the direct mass source in g/s. See Section 7.7.1.	yes
sumpprn(*)	none	Energy source associated with the decay of fission products, radio nuclides, in ergs/s. See Section 7.7.1.	yes
sumptemp(*)	none	Sump temperature |nsumppts| components: in Kelvin for nsumppts > 0; in degrees Celsius for nsumppts < 0. See Section 7.7.1.	yes
sumptime(*)	none	cfilmdef(8,*) > 0, Time for sump temperature table in seconds, sumptemp(*). cfilmdef(8,*) < 0, Number of input values for each of the sump source tables in seconds > 0, implies sumpengy is in ergs/s < 0, implies sumpengy is in ergs/g-s See Section 7.7.1.	yes
surftab(2,*,*)		Surface temperature vs. time table referenced by walldef(7,*) or walldef(8,*) and by sinkdef(14,*) or sinkdef(15,*), respectively. See Section 6.2.	
surftab(2,j,i)		Pair of time (s) and temperature (K) at time tj for table i: jmax ≤ 50, imax ≤ 30. The problem time must never exceed the maximum of the table time. See Section 6.2.	
teta	1.0	Heat-conduction integration flag: = 1.0, backward Euler; = 0.5, Crank-Nicholson.	yes
tsink0	300.0	Initial temperature (K) of sink heat structures. Negative values indicate temperature of adjacent fluid cell is to be used.	no
tslab0	300.0	Initial temperature (K) of slab heat structures. Negative values indicate temperature of adjacent fluid cell is to be used throughout the slab. If tslabbc is set, it overwrites the outer boundary temperature irrespective of tslab0.	no
tslabbc	-9.123	Slab temperature boundary condition for all blocks and boundaries away from the fluid node, can be overridden using slabdef: < 0. implies adiabatic BC > 0. implies fixed outer boundary temperature; = 0. (not input) then tslabbc is determined from tslab0 and kept constant on the boundary.	no
twall0	300.0	Initial temperature (K) of wall heat structures. Negative values indicate temperature of adjacent fluid cell is to be used.	no
walldef(*,*)	none	Wall type definition array. See Section 6.2.	
walldef(1,*)	none	Material identification number: I 1 to mpreset use table 6-1. If i > mpreset use table for element-by-element input for thermal conductivity and rho*cp See also wltabwall, rcptabwall (i > mpreset used for layered structures).	

walldef(2,*)	none	Effective thickness of wall (cm).	
walldef(3,*)	0.0	Sets BC#1, i.e., defines conditions on west, south, or bottom side of the wall: = 0.0 implies a fluid-wall heat exchange (default); > 0.0 implies a constant wall temperature T = walldef(3,*); < 0.0 implies an adiabatic wall condition.	
walldef(4,*)	0.0	Sets BC#2, i.e., defines conditions on east, north, or top side of the wall: = 0.0 implies a fluid-wall heat exchange (default); > 0.0 implies a constant wall temperature T = walldef(4,*); < 0.0 implies an adiabatic wall condition.	
walldef(5,*)	0.0	δx for the node size on the two surfaces of the wall with dynamic mesh expansion symmetrical to the midplane of the node for the given wall thickness; = 0.0 implies a uniform mesh spacing.	
walldef(6,*)	1.0	Fraction of wall area from mesh plane that is used for heat transfer.	
walldef(7,*)	0	Flag for further specification of BC on negative side: = 0, no further modification; i > 0 gives table number from surftab that specifies time-dependent surface temperature, the initial temperature at t = 0 is taken from walldef(3,*); = –1 applies constant heat flux from walldef(9,*) and/or heat transfer with coefficient walldef(10,*) and fluid temperature walldef(3,*) on negative side (positive heat flux means add energy to wall). Only apply fuel conditions to one side of the wall; i.e., do not set walldef(8,*) to –1 when wall(7,*) = –1.	
walldef(8,*)	0	Flag for further specification of BC on positive side: = 0, no further modification; i > 0 gives table number from surftab that specifies time-dependent surface temperature, the initial temperature at t = 0 is taken from walldef(4,*); = –1 applies constant heat flux from walldef(9,*) and/or heat transfer with coefficient walldef(10,*) and fluid temperature walldef(4,*) on positive side (positive heat flux means add energy to the wall). For restrictions, see above.	
walldef(9,*)	0	Heat flux (ergs/cm2·s) applied as BC by walldef(7,*) or walldef(8,*), positive heat flux means add energy to wall.	
walldef(10,*)	0	Heat-transfer coefficient (ergs/cm2·s·K) applied as BC by walldef(7,*) or walldef(8,*).	
wltabsink(*,*)		Array that specifies thermal conductivity in sinks element by element (nhtesink components must be filled).	
wltabsink(*,imat)		Thermal conductivity table for material imat for all sink elements: imat must be > mpreset; materials 1 to mpreset are automatically filled with constant values from library in table 6-1.	
wltabslab(*,*)		Array that specifies thermal conductivity in slabs element by element (nhteslab components must be filled).	
wltabslab(*,imat)		Thermal conductivity table for material imat for all slab elements: imat must be > mpreset; materials 1 to mpreset are automatically filled with constant values from library in table 6-1.	
wltabwall(*,*)		Array that specifies thermal conductivity walls element by element (nhtewall components must be filled).	
wltabwall(*,imat)		Thermal conductivity table for material imat for all slab elements: imat must be > mpreset; materials 1 to mpreset are automatically filled with constant values from library in table 6-1.	

D. Summary of Variables in NAMELIST Group grafic

Variable	Default	Description
c2d(*,*)	none	Array for defining 2D contour plots. See Section 9.1.3.
c2d(1,*)	none	Identification number for first point (second index in pnt).
c2d(2,*)	none	Identification number for second point (second index in pnt). When this value is < 0, then volume fractions for the species hydrogen, oxygen, and water vapour are plotted with constant contour values. See Section 9.1.3.
c2d(3,*)	none	Solution variable. Choose from list in Table 9-1.
c2d(4,*)	none	Gas species name or component number. Required only if c2d(3,*) is 'rsn', 'mf', or 'vf'.
condfthp(*,*)	none	Array for defining energy fluxes due to condensation / vaporization (ergs/cm2-s) time-history plots. See Section 9.1.1.
condfthp(1,*)	none	i-index of fluid cell in contact with heat structure.
condfthp(2,*)	none	j-index of fluid cell in contact with heat structure.
condfthp(3,*)	none	k-index of fluid cell in contact with heat structure.
condfthp(4,*)	none	Block number (must be set to 1 in GASFLOW-MPI).
condfthp(5,*)	none	Heat structure type. Choices are 'slab', slab heat structure; 'sink', sink heat structure; 'wall', wall heat structure.
condfthp(6,*)	none	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north', +j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.
convfthp(*,*)	none	Array for defining energy fluxes due to convective heat transfer (ergs/cm^2s) time-history plots. See Section 9.1.1.
convfthp(1,*)	none	i-index of fluid cell in contact with heat structure.
convfthp(2,*)	none	j-index of fluid cell in contact with heat structure.
convfthp(3,*)	none	k-index of fluid cell in contact with heat structure.
convfthp(4,*)	none	Block number (must be set to 1 in GASFLOW-MPI).
convffthp(5,*)	none	Heat structure type. Choices are 'slab', slab heat structure; 'sink', sink heat structure; 'wall', wall heat structure.
convthp(6,*)	none	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north', +j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.

filmthp(*,*)	none	Array for defining surface liquid film thickness (cm) time-history plots. See Section 9.1.1.
filmthp(1,*)	none	i-index of fluid cell in contact with heat structure.
filmthp(2,*)	none	j-index of fluid cell in contact with heat structure.
filmthp(3,*)	none	k-index of fluid cell in contact with heat structure.
filmthp(4,*)	none	Block number (must be set to 1 in GASFLOW-MPI).
filmthp(5,*)	none	Heat structure type. Choices are 'slab', slab heat structure; 'sink', sink heat structure; 'wall', wall heat structure.
filmthp(6,*)	none	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north', +j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.
gfdt(*)	1010	Time interval (s) at which 3rd party data are written. Input as pairs for interval control. See Sections 8.4.
h2lowfl(*)	0.04	Lower hydrogen volume fraction threshold. See Section 9.1.6
h2upfl	0.75	Upper hydrogen volume fraction threshold. See Section 9.1.6
ht1dp(4,*)	none	Block number (must be set to 1 in GASFLOW-MPI).
ht1dp(5,*)	none	Heat structure type. Choices are 'slab', slab heat structure; 'sink', sink heat structure; 'wall', wall heat structure.
ht1dp(6,*)	none	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north',+j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.
htthp(*,*)	none	Array for defining heat structure surface temperature time-history plots. See Section 9.1.1.
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 in GASFLOW-MPI).
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 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north', +j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.
iinc	1	i-direction cell increment between velocity vectors.
ippka(*)	1	Particle type array for particle plots.
ipvew(*)	1	Viewpoint array for particle plots.
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.
npplts	0	Number of particle plots.
p1d(*,*)	none	Array for defining 1D profile plots. See Section 9.1.2.
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.
p1d(4,*)	none	Gas species name or component number. Required only if p1d(3,*) is 'rsn', 'mf', or 'vf'.
p1dsurf(*,*)	none	Array for defining 1D surface profile plots. See Section 9.1.2.
p1dsurf(1,*)	none	Identification number for first point (second index in pnt).
p1dsurf(2,*)	none	Identification number for second point (second index in pnt).
p1dsurf(3,*)	none	Solution variable. Choose from the following list: ‘condf’ - Water vapour energy flux from condensation or evaporation from the given surface, ergs/cm^{2}s (see next input variable). ‘convf’- Convective energy flux for a given surface, ergs/cm^{2}s (see next input variable). ‘filmt’ - Film of water on a given surface, cm (see next input variable). ‘htcoef’ - Heat transfer coefficient for a given surface, ergs/cm^{2}Ks (see next input variable). ‘massf’ - Water vapour mass flux condensing or evaporating from the given surface, g/cm^{2}s (see next input variable). ‘qrecf’ - Energy flux for a given recombiner surface, ergs/cm^{2}s (see next input variable). ‘radf’ - Radiation energy flux for a given surface, ergs/cm^{2}s (see next input variable).
p1dsurf(4,*)	none	Heat structure type. Choices are: 'slab', slab heat structure; 'sink', sink heat structure; 'wall', wall heat structure.
p1dsurf(5,*)	none	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north', +j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.
pnt(*,*)	none	Array to define points for profile, contour, and vector plots. See Section 9.1.2.
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).
pthpt0	0	Time-history plot initial time (used for runs from restart tapes). See Section 9.1.5
pthp(*,*)	none	Array for defining particle time-history plots.
pthp(1,*)	none	i-mesh index (cell number or cell face number).
pthp(2,*)	none	j-mesh index (cell number or cell face number).
pthp(3,*)	none	k-mesh index (cell number or cell face number).
pthp(4,*)	none	Block number (must be set to 1).
pthp(5,*)	none	Particle data to be plotted: 'pnc', particle number concentration ; 'pmc', particle mass concentration; 'pmt', total mass in fluid cell; 'pmf', particle mass fraction; 'pvf', particle volume fraction; 'pmd', particle mass deposited; 'pmntr', particle cloud mass detected at each monitor.
pthp(6,*)	none	Particle class (itpcl(n)): 0, all classes; >0, particle class number.
pthp(7,*)	none	Particle size number (itpsz(n)): 0, all sizes for class itpcl(n); >0, particle size number.
pthp(8,*)	none	Particle mass deposited on cell faces (for 'pmd' only): 0, all deposited particles; >0, cell faces designated in array mpac(n) 11, deposited on east face of cell; 21, deposited on west face of cell; 12, deposited on north face of cell; 22, deposited on south face of cell; 13, deposited on top face of cell; 23, deposited on bottom face of cell
qrecfthp(*,*)	none	Array for defining energy fluxes due to recombination heat transfer (ergs/cm2-s) time-history plots. See Section 9.1.1.
qrecfthp(1,*)	none	i-index of fluid cell in contact with heat structure.
qrecfthp(2,*)	none	j-index of fluid cell in contact with heat structure.
qrecfthp(3,*)	none	k-index of fluid cell in contact with heat structure.
qrecfthp(4,*)	none	Block number (must be set to 1 for GASFLOW-MPI).
qrecfthp(5,*)	none	Heat structure type. Choices are 'slab', slab heat structure; 'sink', sink heat structure; 'wall', wall heat structure.

qrecfthp(6,*)	none	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north', +j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.
radfthp(*,*)	none	Array for defining energy fluxes due to radiative heat transfer (ergs/cm^{2}s) time-history plots. See Section 9.1.1.
radfthp(1,*)	none	i-index of fluid cell in contact with heat structure.
radfthp(2,*)	none	j-index of fluid cell in contact with heat structure.
radfthp(3,*)	none	k-index of fluid cell in contact with heat structure.
radfthp(4,*)	none	Block number (must be set to 1 for GASFLOW-MPI).
radfthp(5,*)	none	Heat structure type. Choices are 'slab', slab heat structure; 'sink', sink heat structure; 'wall', wall heat structure.
radfthp(6,*)	none	Side of fluid cell in contact with heat structure (not needed for sink heat structure). Choices are 'east', +i side of fluid cell; 'west', –i side of fluid cell; 'north', +j side of fluid cell; 'south', –j side of fluid cell; 'top', +k side of fluid cell; 'bottom', –k side of fluid cell.
thdt(*)	10100	Time interval (s) at which time-history data are stored. Input as pairs for interval control. See Sections 8.4 and 9.1.1.
thp(*,*)	none	Array for defining time-history plots. See Section 9.1.1.
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 for GASFLOW-MPI).
thp(5,*)	none	Solution variable.
thp(6,*)	none	Gas species name or component number. Required only if thp(5,*) is 'rsn', 'mf', or 'vf'.
velmx	2	Magnification factor for velocity vector plots. See Section 9.1.4.
velmx(2)	2	Magnification factor for yz plane velocity vector plots. See Section 9.1.4.
velmx(3)	2	Magnification factor for xz plane velocity vector plots. See Section 9.1.4.
velmx(4)	2	Magnification factor for xy plane velocity vector plots. See Section 9.1.4.
v2d(*,*)	none	Array for defining 2D vector plots. See Section 9.1.4.
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 Section 9.1.4.
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).

E. Summary of Variables in NAMELIST Group parts

The parts NAMELIST group is necessary only for problems where particle dynamics are to be modeled or where tracer particles are wanted. It contains all of the particle input parameters, except those for the particle graphics output and **msp**, **itpcl**, and **itpsz**. The particle graphics parameters are input in the NAMELIST group grafic; **msp**, **itpcl**, and **itpsz** are input as part of the xput NAMELIST group. NAMELIST group parts is selected when **solatype** = 1 or 2. It is located in and called from subroutine RPARTS. All of the arrays and variables in NAMELIST group parts are in named common CPARTS. The following input parameters are to be included in the GASFLOW input file ingf. They are to follow $parts and precede $end. In each case, the $ is in column 2.

Variable	Default	Description
A_particle	0.27	Coefficient for particle LaGrangian fluid timescale.
cdrf	10.0	Drag coefficient roughness factor.
core	0.96	Coefficient of restitution of particle material.
dbl	10.0	Boundary layer thickness (cm).
depperc	5.0	Percentage of particles that immediately deposit upon impact.
fluid_velocity	0	Flag for fluid velocity at particle location model.
fpcoupling	1	Flag for fluid and particle momentum coupling
hca	1.0e-12	Hamaker constant (ergs).
icloud	0	Particle cloud model flag: = 0, particle cloud model is off; = 1, cloud model is on.
imarker	0	Marker particle flag: = 1, particles are moved at the local fluid velocity (most of the particle physics is bypassed); = 0, the complete particle dynamics algorithm is executed.
init_random	0	Particle random number generator seed.
inpvol	0	Input parameter to choose actual or fictitious volume pvol = Δx•Δy•Δz for cell in which particle is located; = 0, actual; = 1, fictitious.
intrn	0	Input parameter to choose particle entrainment option: = 0, no particle pickup; = 1, pickup option is on.
ipblkin	1	Mesh block in which particle is initially located.
ipclin	1	Particle class number input for each class.
ipdep	0	Particle deposition flag input for each class: = 0, no adhesion; = 1, adhesion determined by deposition model; = 2, all impacting particles adhere.
itpcl	1	Total number of particle classes.
itpsz	1	Total number of particle sizes in each particle class.
local_rhogas	0	Flag for computing fluid density for particle model.
local_visf	0	Flag for computing fluid kinematic viscosity for particle model.
ndxdp	-1	Number of longitudinal sections in x-direction in which particles deposition is monitored. Default = –1, deposition not monitored or plotted.
niterp	30	Number of iterations for the Newton iteration cycle for determining drag-induced particle velocity.

npinpt	1000	Total number of simulation particles to be input.
ntmntr	0	Total number of particle cloud monitors.
partvel(*,*)	0	Initial particle velocity.
partvel(1,*)	0	Flag to activate initial particle velocity. = 0; No initial particle velocity = 1; Particle velocities from partvel(2:4,*)
partvel(2,*)	0	u-component for initial particle velocity
partvel(3,*)	0	v-component for initial particle velocity
partvel(4,*)	0	w-component for initial particle velocity
pcdc	0	Particle cloud diffusion coefficient (cm2/s).
pdiamin	1.0	Particle diameter input for each class and size (cm).
pinpdt	1.0e+06	Time interval (seconds) between particle inputs for each class.
pmass	1.0	Total real particle mass of each class and size.
poisrt	0.29	Poisson ratio of particle material class.
prhoin	1.0	Material density input for each particle class (g/cm3).
rhogas	0.0012	Constant fluid density (g/cm3) for particle model.
scfacin	1.0	Cunningham slip correction factor for each particle class and size.
Sc_particle	0.7	Particle Schmidt Number.
sdz	4.0e-08	Separation distance of particle and substrate (cm).
sdzrf	100.0	Separation distance roughness factor.
stokes(*,*)	0	Flag for activating particle Stokes flows for each particle class and size.
tdcp	0.0	Particle turbulent diffusion coefficient (cm2/s).
tinjt	0.01	Total injection time for each particle class (s).
twpinp	0.0	Time when to input particles of each class (s).
visf	nu	Coefficient of kinematic viscosity of fluid (= nu) (cm2/s).
wmax	2.0	Maximum argument of error function used in inverse error function table.
xm	0.0	x-coordinate of particle cloud monitor (cm).
xpe	0.0	Maximum x-coordinate of initial particle block for each particle class (cm).
xpw	0.0	Minimum x-coordinate of initial particle block for each particle class (cm).
ym	0.0	y-coordinate of particle cloud monitor (cm).
yngmod	2.10e+12	Young's modulus of particle material (dynes/cm2).
ypn	0.0	Maximum y-coordinate of initial particle block for each particle class (cm).
yps	0.0	Minimum y-coordinate of initial particle block for each particle class (cm).
zm	0.0	z-coordinate of particle cloud monitor (cm).
zpb	0.0	Minimum z-coordinate of initial particle block for each particle class (cm).
zpt	0.0	Maximum z-coordinate of initial particle block for each particle class (cm).

F. Sample Input Deck with Minimum Data Required

To set up a GASFLOW 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 GASFLOW 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 as below, 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 x 10^6 dynes/cm^2 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 x 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 1.0 s. By default, automatic time-step control is chosen. Specification of the grafic 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 x 10^{-6} cm/s.

Fictitious Problem with Minimum Input Data

NOTES: 2-D domain 100 cm x 250 cm with deltax = deltay = deltaz = 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
------------------------------------------------------------------------
 $innet
 $end
 $xput
  twfin = 1.0,
  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 of cycles is 10.
```

```
  mat = 'air',
  ; Initial condition throughout domain:
  gasdef(1,1) = 1, 21, 1, 51, 1, 2, 1, 1.0e6, 300.0, 2, 0.0, 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) = 0.0 , nxl(1) = 0, nxr(1) = 20, dxmn(1) = 9999.,
  xl(2) = 100.0,
  nky = 1,
  yl(1) = 0.0, yc(1) = 0.0, nyl(1) = 0, nyr(1) = 50 , dymn(1) = 9999.,
  yl(2) = 250.0,
  nkz = 1,
  zl(1) = 0.0, zc(1) = 0.0, nzl(1) = 0, nzr(1) = 1 , dzmn(1) = 9999.,
  zl(2) = 5.0,
 $end
---------------------------------------------------------------------
          G R A P H I C S
---------------------------------------------------------------------
 $grafic
  ; 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.
  thdt = 0.1,
  ; Define two points that would cover the entire physical x-y domain.
  pnt(1:4, 1) = 1, 1, 2, 1,
  pnt(1:4, 2) = 21,51, 2, 1,
  ; velocity vector plot on plane defined by points 1 and 2.
  v2d = 1, 2, 1,
  ; temperature contour plot on plane defined by points 1 and 2.
  c2d = 1, 2, 'tk', 0,
 $end
 $rheat
 $end
 $parts
 $end
 $special
 $end
```

G. Binary Output in GASFLOW-MPI

G.1 Introduction

GASFLOW traditionally wrote several binary files containing time dependent information about the calculation for graphical output at the end of the calculation. These files were called PlotHist, tape11, tape12, tape13 and tape14. Sometimes people wanted to access the information in these files. This caused problems because of the following reasons:

- The files are written using Fortran unformatted writes. Therefore these files are only readable on systems with the same binary format.
- There is no (meta) information in these files to help locating the real information. The user had to count columns of data. Frequently errors were made. When the structure of the output files was changed, the analysis programs had to be changed too.
- The data behind the p1d, ht1dp, c2d, and v2d plots, which represent spatial 1-D and 2-D profiles, were not available at all.

To solve these problems the GASFLOW binary output is written in the NetCDF format. For reasons of consistency the contents of the formatted output file ddtchar are also written in the NetCDF format. To make code maintenance easier, the restart files are also written in NetCDF format.

G.2 The NetCDF format

The NetCFD format is a binary, portable format which can be used to store additional information about the data. In depth information about NetCDF is available under http://www.unidata.ucar.edu/packages/netcdf/index.html. Several utilities to read and process NetCDF files exist. Extensions for high level programming languages like Perl, Python, IDL, Matlab and others to directly read and write NetCDF files are readily available. For the most common programming languages C, C++, Fortran, Fortran90 and Java the NetCDF package itself contains the programming interface. The availability of a Fortran90 interface was the reason why the NetCDF format was selected.

Data is stored in NetCDF files in the form of *variables/arrays*. Additional information can be stored using so called *attributes*. Think about using data on a NetCDF file just as if you use an array in your favorite programming language. Every item in a NetCDF file has a *name*. To access a variable you use its *name*, not its position in a file, which might change.

G.3 Utilities for NetCDF file processing

It is not very convenient to write a Fortran or C program every time, you want to access data on the NetCDF file. There are several tools freely available, which make accessing the NetCDF file a lot easier.

G.3.1 ncdump and ncgen

ncdump and ncgen are the basic utilities to convert the content of a NetCDF file to a human readable format. Its output is not designed to be used for further processing in graphics programs. The command ncdump -h file.nc lists the complete header of the NetCDF file, i.e. all variable definitions and attributes. These utilities come with the library itself and should be available on every system where the NetCDF library is available.

G.3.2 fan utilities

The fan utilitiles provide easy to use programs for basic operations on netcdf files. Amongst other things they can be used to convert any variable in a NetCDF file to a readable format. They have many options to select how the data are written. They can be used to extract exactly the needed data from the NetCDF file. Unfortunatly this package is not developed anymore.

G.3.3 nco

The program package nco (short for NetCDF operators) provides sophisticated tools to work with the data on NetCDF files. Amongst other features, it provides a simple way to concatenate NetCDF files. nco is available through http://nco.sf.net. This package provides many features. It is highly recommended to read the documentation carefully. Correct usage of this package can sometimes save a lot of time.

G.4 Gasflow time history plots

All time history information which was written to various files is written to a single file named PlotHist.nc. The spatial profiles are written to the file Profiles.nc (see section 5).

G.4.1 File structure

The variable names in the NetCDF file are (almost) the same as in the Gasflow source. For added accuracy the type of all real*4 variables has been converted to real*8. A partial list of the variable names with a description of their meaning is given in table 1. The items in angle brackets "[ ]" are the units of the variable. The units attribute is set to the same value.

Table 18-1:Time history variables

Variable name	Description
time	Time [sec]
einternl	Total current internal energy in all real cells [erg].
ekinetic	Total kinetic energy in all real cells [erg].
energy_source	Total rate of energy addition to gas phase from heat structures and rainout (includes convective heat transfer, rainout, and phase change of films) [erg/s].

flow	Total time integrated mass flow rate in minus mass flow rate out [gram].
tmassp	Predicted from global mass balance total mass in all real cells [gram].
tmassc	Current total mass in all real cells [gram].
errmas	Mass error [%].
tenergyp	Predicted total internal energy in all real cells [erg].
flowein	In flow rate of energy across all negative boundaries (i.e. west, south, and bottom) [erg/s].
floweout	Out flow rate of energy across all positive boundaries (i.e. east, north, and top) [erg/s].
flowein_tot	Time integrated in flow rate of energy across all negative boundaries (i.e. west, south, and bottom) [erg].
floweout_tot	Time integrated out flow rate of energy across all negative boundaries (i.e. east, north, and top) [erg].
pv_work	Energy loss rate due to PdV work [erg/s].
pv_work_tot	Time integrated energy loss rate due to PdV work [erg].
rsource_tot	Time integrated energy addition rate due to PdV work caused by phase change of films [erg].
energy_strip	Time integrated energy due to removal of film mass, I.e., thickness > filmth [erg].
e_struct	Time integrated energy loading, I.e., from convection, phase-change, and radiation, on all structures, I.e., slabs, walls and sinks [erg].
comb_tot	Time integrated energy addition rate due to combustion [erg].
exenon_tot	Time integrated energy addition rate due to decay heat [erg].
erreng	Energy error [%].
esslabpchy_tot	Time integrated energy addition rate from the film to the gas phase due to phase change of the film on slabs [erg].
eswallpchy_tot	Time integrated energy addition rate from the film to the gas phase due to phase change of the film on walls [erg].
essinkpchy_tot	Time integrated energy addition rate from the film to the gas phase change of the film on sinks [erg].
esslabpcht_tot	Time integrated energy addition rate from the film to all slabs due to phase change of the film [erg].
esswallpcht_tot	Time integrated energy addition rate from the film to all walls due to phase change of the film [erg].
essinckpcht_tot	Time integrated energy addition rate from the film to all sinks due to phase change of the film [erg].
esslabcv_tot	Time integrated energy addition rate due to convection on all slabs [erg].
eswallcv_tot	Time integrated energy addition rate due to convection on all walls [erg].
essinkcv_tot	Time integrated energy addition rate due to convection on all sinks [erg].
esrainout_tot	Time integrated energy addition rate due to rainout [erg].
esource_tot	Time integrated energy addition rate due to convection from heat structures, and phase change of films [erg]. Time integrated of energy_source.
filmslabeng	Total internal energy of all films on all slabs [erg].
filmwalleng	Total internal energy of all films on all walls [erg].
filmsinkeng	Total internal energy of all films on all sinks [erg].
esslabrdt_tot4	Time integrated energy addition rate due to radiation on all slabs [erg].
eswallrdt_tot4	Time integrated energy addition rate due to radiation on all walls [erg].
essinkrdt_tot4	Time integrated energy addition rate due to radiation on all sinks [erg].
totalUrad4	Total radiation intensity of all fluid cells, i.e. radiant energy [erg].
totalEmA4	Total emission/absorption energy, i.e. radiant energy exchange with fluid field of all fluid cells [erg].
mass_strip	Time integrated mass due to removal of film mass, I.e., thickness>filmth [gram].
tfilm	Total mass in films on all heat structures [gram].
tconv	Total change mass due to bulk phase change for all real cells [gram/s].

train	Total rain out for all real cells [gram/s].
sumh2los	Hydrogen recombination rate [gram/s].
slabcond	Total of steam mass condensed on slabs [gram].
slabevap	Total mass of water evaporated on slabs [gram].
wallcond	Total of steam mass condensed on negative sides of walls [gram].
wallevap	Total mass of water evaporated on negative sides of walls [gram].
wal2cond	Total of steam mass condensed on positive sides of walls [gram].
wal2evap	Total mass of water evaporated on positive sides of walls [gram].
sinkcond	Total mass of steam mass condensed on sinks [gram].
sinkevap	Total mass of water evaporated on sinks [gram].
slabcrat	Condensation rate on slabs [gram/s].
slaberat	Evaporation rate on slabs [gram/s].
wallcrat	Condensation rate on negative side of walls [gram/s].
wallerat	Evaporation rate on negative side of walls [gram/s].
wal2crat	Condensation rate on positive side of walls [gram/s].
wal2erat	Evaporation rate on positive side of walls [gram/s].
sinkcrat	Condensation rate on sinks [gram/s].
sinkerat	Evaporation rate on sinks [gram/s].
st_eng_slabs_tot	Time integrated rate of change of stored energy in slabs [erg].
st_eng_walls_tot	Time integrated rate of change of stored energy in walls [erg].
st_eng_sinks_tot	Time integrated rate of change of stored energy in sinks [erg].
erefer	Reference energy at TREFER [erg] (default of TREFER is 300 K).
msrainout_tot	Time integrated mass due to rainout [gram].
tconv_tot	Total energy due to convection on all slabs, walls and sinks [erg].
qrecslab_tot	Rate of energy addition to slabs from recombination on foils [erg/s].
qrecwall_tot	Rate of energy addition to walls from recombination on foils [erg/s].
cycle	cycle number at current output
From here on the presence of the entries depends on the input. The following three dimensions are defined in the output file.	
ntotmat	Total number of materials (default=6)
ncomp	Number of gas components (see mat input)
nblocks	Number of blocks
seslab(ntotmat)	Rate of energy addition to slabs itemized according to material numbers [erg/s].
sewall(ntotmat)	Rate of energy addition to walls itemized according to material numbers [erg/s].
Sesink(ntotmat)	Rate of energy addition to all sinks itemized according to material numbers [erg/s]
tmaxmat(ntotmat)	Maximum temperature per materials [Kelvin].
tmaxmat_ind(ntotmat)	The corresponding 1D index (Value of m).
tmaxslab	Maximum temperature of all slabs [Kelvin].
tmaxslab_ind	The corresponding 1D index.
tmaxwall	Maximum temperature of all walls [Kelvin].
tmaxwall_ind	The corresponding 1D index.
tmaxsink	Maximum temperature of all sinks [Kelvin].
tmaxsink_ind	The corresponding 1D index.
The following variables have a block dimension.	
blkvol(nblocks)	Total gas volume including all real cells in 3D block number 1 [cm3].
blkpave(nblocks)	Average pressure in 3D block number 1 [dyne/[cm^2]].
blktave(3,nblocks)	(1,*): Average fluid temperature in 3D block number 1 [K]. (2,*): Average vapour temperature (spray model must be active) in 3D block number 1 [K]. (3,*): Average droplet temperature (spray model must be active) in 3D block number 1 [K].
spmass(ncomp.nblocks)	Total mass of each species for each block [gram].

ddtchar variables. Dimension: (numrooms, numthresholds)	
ddt_disddt	Characteristic Cloud Dimension
ddt_phi	Average Hydrogen (Dry) Volume Fraction
ddt_sevenlm	Detonation Cell Size x 7,7 lambda
ddt_ratiodol	Ratio of D to 7 lambda
ddt_avexh2	Av. H_2
ddt_avex02	Av. O_2
ddt_avexh2o	Av.H_2O
ddt_avetn	Av. Temp. [K]
ddt_conch2max	Max. H_2
ddt_tnvmax	Max. Temp. [K]
ddt_rainmass	Rainout mass [gram]
ddt_rainenergy	Rainout energy [erg]
ddt_h2mass	Mass of Hydrogen [gram]
ddt_sigmah2max	Sigma Criterion for Maximum Hydrogen
ddt_sigmah2min	Sigma Criterion for Minimum Hydrogen
ddt_sigmaaveh2	Sigma Criterion for Average Hydrogen
ddt_sigmavol	Sigma Cloud Volume [cm3]
ddt_sigmamass	Sigma Cloud Mass [gram]
ddt_h2mass_thres	Mass of Hydrogen above threshold [gram]

In addition to these variables PlotHist.nc also contains the normal user requested time history plots. Because the names have to be unique, the names are generated taking the name of the gasflow input variable and adding a 5 digit number to it. The first thp statement generates a NetCDF variable named thp00001, the second thp00002. Additional information is stored in attributes of these variables. In case you are interested, you can use the command ncdump -h PlotHist.nc to view all variables and attributes in the NetCDF file. The following variable names are used: thp, htthp, filmthp, condfthp, convfthp, radfthp, qrecfthp, pthp, massfthp.

Because of the direct access capabilities of the NetCDF format is was possible to implement a correct continuation of the time history data on restarts. The resulting file will start a time zero and will not contain any overlapping entries. This behaviour is controlled by a new input parameter in namelist group xput:

continue_plothist

0	Overwrite the file.
1	Append new data at the correct location.
2	Append, but do NOT write a new data point immediately after the restart.

Of course this can only work if the file PlotHist.nc from the previous run is in the current execution directory. But beware: the relevant output statements in the input file must not be changed on restart. No check is done. The data in PlotHist.nc are overwritten in place. Therefore in a calculation restarted at 800 s with a PlotHist.nc file containing data up to 1000 s the file will contain old data until the time exceeds 1000 s.

Because the contents of the NetCDF file buffer are flushed to disk every 50 seconds, the programs mentioned in section 3 can be used to monitor a running calculation.

G.4.2 Example

This the output of ncdump -h PlotHist.nc for a small

```
netcdf PlotHist {
dimensions:
	imax = 22 ;
	jmax = 52 ;
	kmax = 3 ;
	two = 2 ;
	three = 3 ;
	four = 4 ;
	six = 6 ;
	eight = 8 ;
	ncomp = 4 ;
	nblocks = 1 ;
	nthpts = UNLIMITED ; // (2 currently)
	ntotmat = 6 ;
variables:
	char mat(ncomp, eight) ;
	double time(nthpts) ;
		time:units = "sec" ;
	double timep(nthpts) ;
		timep:units = "sec" ;
	double thp00001(nthpts) ;
		thp00001:i = 10 ;
		thp00001:j = 25 ;
		thp00001:k = 2 ;
		thp00001:iblk = 1 ;
		thp00001:name1 = "pn" ;
		thp00001:units = "dyne/cm**2" ;
	double thp00002(nthpts) ;
		thp00002:i = 10 ;
		thp00002:j = 25 ;
		thp00002:k = 2 ;
		thp00002:iblk = 1 ;
		thp00002:name1 = "tk" ;
		thp00002:units = "Kelvin" ;
	double thp00003(nthpts) ;
		thp00003:i = 10 ;
		thp00003:j = 5 ;
		thp00003:k = 2 ;
		thp00003:iblk = 1 ;
		thp00003:name1 = "vf" ;
		thp00003:units = "Volume fraction" ;
		thp00003:name2 = "h2" ;
	double thp00004(nthpts) ;
		thp00004:i = 10 ;
		thp00004:j = 5 ;
		thp00004:k = 2 ;
		thp00004:iblk = 1 ;
		thp00004:name1 = "vf" ;
		thp00004:units = "Volume fraction" ;
		thp00004:name2 = "h2o" ;
	double einternl(nthpts) ;
		einternl:units = "erg" ;
```

```
double ekinetic(nthpts) ;
        ekinetic:units = "erg" ;
double esrcsum(nthpts) ;
        esrcsum:units = "erg" ;
double flow(nthpts) ;
        flow:units = "gram/sec" ;
double tmassp(nthpts) ;
        tmassp:units = "gram" ;
double tmassc(nthpts) ;
        tmassc:units = "gram" ;
double errmas(nthpts) ;
        errmas:units = "Percent" ;
double tenergyp(nthpts) ;
        tenergyp:units = "erg" ;
double flowein(nthpts) ;
        flowein:units = "erg/sec" ;
double floweout(nthpts) ;
        floweout:units = "erg/sec" ;
double flowein_tot(nthpts) ;
        flowein_tot:units = "erg" ;
double floweout_tot(nthpts) ;
        floweout_tot:units = "erg" ;
double pv_work(nthpts) ;
        pv_work:units = "erg/sec" ;
double pv_work_tot(nthpts) ;
        pv_work_tot:units = "erg" ;
double rsource_tot(nthpts) ;
        rsource_tot:units = "erg" ;
double mass_strip(nthpts) ;
        mass_strip:units = "gram" ;
double energy_strip(nthpts) ;
        energy_strip:units = "erg" ;
double comb_tot(nthpts) ;
        comb_tot:units = "erg" ;
double exenon_tot(nthpts) ;
        exenon_tot:units = "erg" ;
double erreng(nthpts) ;
        erreng:units = "Percent" ;
double esslabpchy_tot(nthpts) ;
        esslabpchy_tot:units = "erg" ;
double eswallpchy_tot(nthpts) ;
        eswallpchy_tot:units = "erg" ;
double essinkpchy_tot(nthpts) ;
        essinkpchy_tot:units = "erg" ;
double esslabpcht_tot(nthpts) ;
        esslabpcht_tot:units = "erg" ;
double eswallpcht_tot(nthpts) ;
        eswallpcht_tot:units = "erg" ;
double essinkpcht_tot(nthpts) ;
        essinkpcht_tot:units = "erg" ;
double esslabcv_tot(nthpts) ;
        esslabcv_tot:units = "erg" ;
double eswallcv_tot(nthpts) ;
        eswallcv_tot:units = "erg" ;
double essinkcv_tot(nthpts) ;
        essinkcv_tot:units = "erg" ;
```

```
double esrainout_tot(nthpts) ;
        esrainout_tot:units = "erg" ;
double esource_tot(nthpts) ;
        esource_tot:units = "erg" ;
double filmslabeng(nthpts) ;
        filmslabeng:units = "erg" ;
double filmwalleng(nthpts) ;
        filmwalleng:units = "erg" ;
double filmsinkeng(nthpts) ;
        filmsinkeng:units = "erg" ;
double esslabrdht_tot(nthpts) ;
        esslabrdht_tot:units = "erg" ;
double eswallrdht_tot(nthpts) ;
        eswallrdht_tot:units = "erg" ;
double essinkrdht_tot(nthpts) ;
        essinkrdht_tot:units = "erg" ;
double totalUrad(nthpts) ;
        totalUrad:units = "erg" ;
double totalEmA(nthpts) ;
        totalEmA:units = "erg" ;
double e_struct(nthpts) ;
        e_struct:units = "erg" ;
double tfilm(nthpts) ;
        tfilm:units = "gram" ;
double tconv(nthpts) ;
        tconv:units = "gram/sec" ;
double train(nthpts) ;
        train:units = "gram/sec" ;
double sumh2los(nthpts) ;
        sumh2los:units = "gram/sec" ;
double slabcond(nthpts) ;
        slabcond:units = "gram" ;
double slabevap(nthpts) ;
        slabevap:units = "gram" ;
double wal1cond(nthpts) ;
        wal1cond:units = "gram" ;
double wal1evap(nthpts) ;
        wal1evap:units = "gram" ;
double wal2cond(nthpts) ;
        wal2cond:units = "gram" ;
double wal2evap(nthpts) ;
        wal2evap:units = "gram" ;
double sinkcond(nthpts) ;
        sinkcond:units = "gram" ;
double sinkevap(nthpts) ;
        sinkevap:units = "gram" ;
double slabcrat(nthpts) ;
        slabcrat:units = "gram/sec" ;
double slaberat(nthpts) ;
        slaberat:units = "gram/sec" ;
double wal1crat(nthpts) ;
        wal1crat:units = "gram/sec" ;
double wal1erat(nthpts) ;
        wal1erat:units = "gram/sec" ;
double wal2crat(nthpts) ;
        wal2crat:units = "gram/sec" ;
```

```
    double wal2erat(nthpts) ;
        wal2erat:units = "gram/sec" ;
    double sinkcrat(nthpts) ;
        sinkcrat:units = "gram/sec" ;
    double sinkerat(nthpts) ;
        sinkerat:units = "gram/sec" ;
    double st_eng_slabs(nthpts) ;
        st_eng_slabs:units = "erg" ;
    double st_eng_walls(nthpts) ;
        st_eng_walls:units = "erg" ;
    double st_eng_sinks(nthpts) ;
        st_eng_sinks:units = "erg" ;
    double tmass(nthpts) ;
        tmass:units = "erg" ;
    double errfilm_mass(nthpts) ;
        errfilm_mass:units = "Percent" ;
    double tenergy(nthpts) ;
        tenergy:units = "erg" ;
    double errfilm_eng(nthpts) ;
        errfilm_eng:units = "Percent" ;
    double erefer(nthpts) ;
        erefer:units = "erg" ;
    double msrainout_tot(nthpts) ;
        msrainout_tot:units = "gram" ;
    double tconv_tot(nthpts) ;
        tconv_tot:units = "gram" ;
    int cycle(nthpts) ;
    double blkvol(nthpts, nblocks) ;
        blkvol:units = "cm**3" ;
    double blkpave(nthpts, nblocks) ;
        blkpave:units = "dyne/cm**2" ;
    double blktave(nthpts, three, nblocks) ;
        blktave:units = "Kelvin" ;
    double spmass(nthpts, ncomp, nblocks) ;
        spmass:units = "gram" ;

// global attributes:
        :jobtitle = "3-Compartment H2 Burn Test Problem (GF1.4:09/95)" ;
        :code_version = "GASFLOW version 2.2.4.26_fzk $Date: 2003/04/02 12:33:04 $" ;
        :time = "06 Aug 03 11:28:19" ;
```

G.5 Profile plots

All information available through the p1d, ht1dp, c2d, v2d,p1dsurf, plotting requests is written to the file Profiles.nc. The writing frequency is controlled by pltdt. In contrast to the file PlotHist.nc, which is always generated, Profiles.nc is only written, when use_profiles = 1, is set in namelist grafic.

G.5.1 File structure

The dimension nplpts is used to distinguish between the different timesteps. The other dimensions are generated programmatically as required..

The variable mdim contains the mesh dimensions for each block. The variable mflag0 contains the mesh flag array. If you need information about this array, please consult the Gasflow source code.

The xgrid*, ygrid*, zgrid* variables give the location of the cell edges. The xcent_*, ycent_*, zcent_* give the locations of the cell centers. The block number is coded into the variable name.

The various attributes are self explanatory with one exception: mtype. This attribute gives the location of the variable:

0: cell centered variable
1: x-face centered variable
2: y-face centered variable
3: z-face centered variable

The solution variables itself have their index coded into the file name: p1d is coded as p1d0000001, etc. They have the same name as in the input file except p1dsurf, which is truncated to p1ds.

G.5.2 Example

This the output of ncdump -h Profiles.nc for a small testcase:

```
netcdf Profiles {
dimensions:
	npltpts = UNLIMITED ; // (2 currently)
	nblocks = 1 ;
	two = 2 ;
	three = 3 ;
	mft = 21818 ;
	dim0000200 = 200 ;
	dim0000034 = 34 ;
	dim0000004 = 4 ;
	dim0000204 = 204 ;
	dim0000038 = 38 ;
	dim0000005 = 5 ;
	dim0000202 = 202 ;
	dim0000036 = 36 ;
	dim0000003 = 3 ;
variables:
	int mdim(nblocks, three) ;
	int mflag0(mft) ;
	double xgrid001(dim0000204) ;
	double ygrid001(dim0000038) ;
	double zgrid001(dim0000005) ;
	double xcent_01(dim0000202) ;
	double ycent_01(dim0000036) ;
	double zcent_01(dim0000003) ;
	double time(npltpts) ;
	double timep(npltpts) ;
	int cycle(npltpts) ;
	double v2d00001(npltpts, dim0000034, dim0000200, two) ;
		v2d00001:imin = 1 ;
		v2d00001:imax = 201 ;
```

```
        v2d00001:jmin = 1 ;
        v2d00001:jmax = 35 ;
        v2d00001:kmin = 2 ;
        v2d00001:kmax = 2 ;
        v2d00001:iblock = 1 ;
        v2d00001:mtype = 0 ;
        v2d00001:units = "cm/sec" ;
    double c2d00001(npltpts, dim0000034, dim0000200) ;
        c2d00001:Label = "tk" ;
        c2d00001:imin = 1 ;
        c2d00001:imax = 201 ;
        c2d00001:jmin = 1 ;
        c2d00001:jmax = 35 ;
        c2d00001:kmin = 2 ;
        c2d00001:kmax = 2 ;
        c2d00001:iblock = 1 ;
        c2d00001:name = "tk" ;
        c2d00001:mtype = 0 ;
        c2d00001:units = "Kelvin" ;
    double c2d00002(npltpts, dim0000034, dim0000200) ;
        c2d00002:Label = "pn" ;
        c2d00002:imin = 1 ;
        c2d00002:imax = 201 ;
        c2d00002:jmin = 1 ;
        c2d00002:jmax = 35 ;
        c2d00002:kmin = 2 ;
        c2d00002:kmax = 2 ;
        c2d00002:iblock = 1 ;
        c2d00002:name = "pn" ;
        c2d00002:mtype = 0 ;
        c2d00002:units = "dyne/cm**2" ;
    double p1ds0001(npltpts, dim0000004) ;
        p1ds0001:imin = 1 ;
        p1ds0001:imax = 5 ;
        p1ds0001:jmin = 2 ;
        p1ds0001:jmax = 2 ;
        p1ds0001:kmin = 2 ;
        p1ds0001:kmax = 2 ;
        p1ds0001:iblock = 1 ;
        p1ds0001:variable = "massf" ;
        p1ds0001:type = "wall" ;
        p1ds0001:side = "south" ;
        p1ds0001:units = "gram/cm**2/sec" ;
    double p1ds0002(npltpts, dim0000004) ;
        p1ds0002:imin = 1 ;
        p1ds0002:imax = 5 ;
        p1ds0002:jmin = 2 ;
        p1ds0002:jmax = 2 ;
        p1ds0002:kmin = 2 ;
        p1ds0002:kmax = 2 ;
        p1ds0002:iblock = 1 ;
        p1ds0002:variable = "filmt" ;
        p1ds0002:type = "wall" ;
        p1ds0002:side = "south" ;
        p1ds0002:units = "cm" ;
    double p1ds0003(npltpts, dim0000004) ;
```

```
        p1ds0003:imin = 1 ;
        p1ds0003:imax = 5 ;
        p1ds0003:jmin = 2 ;
        p1ds0003:jmax = 2 ;
        p1ds0003:kmin = 2 ;
        p1ds0003:kmax = 2 ;
        p1ds0003:iblock = 1 ;
        p1ds0003:variable = "condf" ;
        p1ds0003:type = "wall" ;
        p1ds0003:side = "south" ;
        p1ds0003:units = "erg/cm**2/sec" ;
    double p1ds0004(npltpts, dim0000004) ;
        p1ds0004:imin = 1 ;
        p1ds0004:imax = 5 ;
        p1ds0004:jmin = 2 ;
        p1ds0004:jmax = 2 ;
        p1ds0004:kmin = 2 ;
        p1ds0004:kmax = 2 ;
        p1ds0004:iblock = 1 ;
        p1ds0004:variable = "convf" ;
        p1ds0004:type = "wall" ;
        p1ds0004:side = "south" ;
        p1ds0004:units = "erg/cm**2/sec" ;
    double p1ds0005(npltpts, dim0000004) ;
        p1ds0005:imin = 1 ;
        p1ds0005:imax = 5 ;
        p1ds0005:jmin = 2 ;
        p1ds0005:jmax = 2 ;
        p1ds0005:kmin = 2 ;
        p1ds0005:kmax = 2 ;
        p1ds0005:iblock = 1 ;
        p1ds0005:variable = "htcoef" ;
        p1ds0005:type = "wall" ;
        p1ds0005:side = "south" ;
        p1ds0005:units = "Unknown" ;

// global attributes:
        :jobtitle = "SARNET Condensation Benchmark: HMT-30-3" ;
        :code_version = "GASFLOW version 2.4.1.6_fzk  $Date: 2006/11/06 14:10:45 $" ;
        :time = "08 Nov 06 14:22:18" ;
        :icyl = 0 ;
```

G.6 Restart files

The NetCDF format is also used for the restart files. This change provides much better diagnostics if a restart fails because of format problems. Each variable in the restart file is checked on restart whether its size is correct. If any check fails, the restart is aborted.

Of course it is also possible to access the data in the restart file without using GASFLOW. This requires a deep understanding of GASFLOW internals and is therefore completely unsupported with one exception: each restart file contains the three global attributes jobtitle, code_version and time.

These contain the title of the job, the code version string and the start time of the job. These can be retrieved for exammple by the ncdump –h command.

G.7 Concluding remarks

The NetCDF format combines several important advantages:

- Portability of the binary files across different platforms.
- Efficient reading/writing.
- Reliable, direct access to the desired information. Drastically reduced danger to access wrong data.
- Additional information about the calculation is stored within the file. More information can be added.
- Using this information, the data can be presented in different units: the pressure can be given in dyne/cm2, bar, pascal.
- If the output is unchanged, the time history information of the previous run can be used. This means, it is easy to get time history plots starting at time = 0 even if the job has to be restarted several times.

ISSN 1869-9669
ISBN 978-3-7315-0448-1 (Vol. 1)
ISBN 978-3-7315-0449-8 (Vol. 2)
ISBN 978-3-7315-0447-4 (Set)