



Multiphase Flow

MFIX

with Interphase eXchanges

Release 2014-1

October 2014

<https://mfix.netl.doe.gov>

Image credits:

Cover page: Tingwen Li, *URS E&C Inc.*: MFIX-TFM simulation of a reacting transport gasifier colored by CO₂ mass fraction.

Page 8, top: Aytekin Gel, *ALPEMI Consulting, LLC*: MFIX-TFM simulation of coal jet penetration, colored by CO₂ species mass fraction.

Page 8, bottom: Jordan Musser, *DOE NETL*: Reactive MFIX-DEM simulation of a spouted bed, particles colored by temperature.

Page 9, top: Rahul Garg, *URS E&C Inc.*: MFIX-PIC simulation of a cyclone, showing particles and streamlines, colored by velocity.

Page 9, bottom: Jordan Musser, *DOE NETL*: MFIX-Hybrid simulation of a bubbling bed with two solids phases (one continuous phase and one discrete phase). The background is colored by solids bulk density (continuous phase) and spheres represent the second solids phase.

Notice

Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed or represents that its use would not infringe privately owned rights.

- MFIX is provided without any user support for applications in the user's immediate organization. It should not be redistributed in whole or in part.
- The use of MFIX is to be acknowledged in any published paper based on computations using this software by citing the MFIX theory manual. Some of the submodels are being developed by researchers outside of NETL. The use of such submodels is to be acknowledged by citing the appropriate papers of the developers of the submodels.
- The authors would appreciate receiving any reports of bugs or other difficulties with the software, enhancements to the software, and accounts of practical applications of this software.

Disclaimer

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

Table of Contents

1	INTRODUCTION.....	6
2	DEVELOPMENT STATE OF MFIX MODELS	8
3	RELEASE NOTES FOR 2014-1	10
4	MFIX ON A UNIX/LINUX WORKSTATION	12
4.1	EXTRACTING THE MFIX DIRECTORY	12
4.2	BUILDING MFIX.....	12
4.2.1	<i>MFIX Build Script.....</i>	<i>12</i>
4.2.2	<i>Building MFIX: A step-by-step tutorial</i>	<i>14</i>
4.3	RUNNING MFIX.....	16
4.3.1	<i>Serial Execution</i>	<i>16</i>
4.3.2	<i>SMP Execution.....</i>	<i>17</i>
4.3.3	<i>DMP Execution.....</i>	<i>17</i>
4.4	BUILDING POSTMFIX.....	18
4.5	ENABLING NETCDF OUTPUT.....	18
5	MFIX ON WINDOWS	20
5.1	PREBUILT WINDOWS BINARIES FOR MFIX.....	20
5.2	BUILDING MFIX WITH CYGWIN.....	20
5.2.1	<i>Installing Cygwin</i>	<i>20</i>
5.2.2	<i>Building and Running MFIX.....</i>	<i>20</i>
6	MFIX AT RUN TIME	22
6.1	MFIX OUTPUT AND MESSAGES	22
6.2	RESTARTING A RUN	23
6.3	WHEN THE RUN DOES NOT CONVERGE	23
7	POST PROCESSING AND VISUALIZATION	25
7.1	RUNNING POSTMFIX	25
7.2	PARAVIEW.....	27
7.3	VISIT.....	28
8	KEYWORDS IN INPUT DATA FILE (MFIX.DAT).....	31
8.1	RUN CONTROL	31
8.2	PHYSICAL PARAMETERS.....	35
8.3	NUMERICAL PARAMETERS.....	36
8.4	GEOMETRY AND DISCRETIZATION.....	39
8.4.1	<i>Basic Geometry.....</i>	<i>39</i>
8.4.1	<i>Cartesian Grid</i>	<i>43</i>
8.5	GAS PHASE	51
8.6	SOLIDS PHASE	52

8.6.1	<i>Two Fluid Model (TFM)</i>	53
8.6.2	<i>Discrete Element Simulations</i>	57
8.6.2.1	<i>Discrete Element Model (DEM)</i>	59
8.6.2.2	<i>Particle in Cell (PIC)</i>	60
8.7	INITIAL CONDITIONS	61
8.8	BOUNDARY CONDITIONS	65
8.8.1	<i>Wall boundary conditions</i>	67
8.8.1.1	<i>Momentum Equations</i>	67
8.8.1.2	<i>Granular Energy Equation</i>	67
8.8.1.3	<i>Gas and Solids Energy Equations</i>	68
8.8.1.4	<i>Gas and Solids Species Equations</i>	69
8.8.1.5	<i>Scalar Transport Equations</i>	69
8.8.2	<i>Flow Boundary Conditions</i>	70
8.9	INTERNAL SURFACE	73
8.10	POINT SOURCES	74
8.11	OUTPUT CONTROL	76
8.12	USER-DEFINED FUNCTIONS.....	80
8.13	CHEMICAL REACTIONS.....	86
8.13.1	<i>Chemical Reactions Specification</i>	86
8.13.2	<i>Stiff Chemistry Solver</i>	98
8.14	THERMOCHEMICAL PROPERTIES	100
8.15	PARALLELIZATION CONTROL.....	101
8.16	BATCH QUEUE ENVIRONMENT	102
9	MAILING LISTS	104
10	USER CONTRIBUTION	106

1 Introduction

MFIX is an open-source multiphase flow solver and is free to download and use. A one-time no-cost registration is required prior to downloading the source code. To register, go to <https://mfix.netl.doe.gov/registration.php> and complete the form. Once you have read the notice, you can submit your application by clicking on "I Agree." After your application has been reviewed and accepted, you will receive an email notification and instructions on how to download the code. Please allow for 2-3 business days for your registration to be processed.

Potential users may find reviewing the Frequently Asked Questions section of the MFIX website, <https://mfix.netl.doe.gov/faq.php>, useful before downloading the code.

The MFIX distribution contains the following:

Item	Description
model	Source files for MFIX, the multiphase flow solver.
post_mfix	Source files for POSTMFIX, a text-based program to process MFIX simulation data.
tutorials	Example problems, some containing documentation describing the simulation setup which new users may find useful.
tests	Test simulations used to verify the code during development.
tools	Various utilities and development tools.
doc	MFIX documentation (in pdf format)

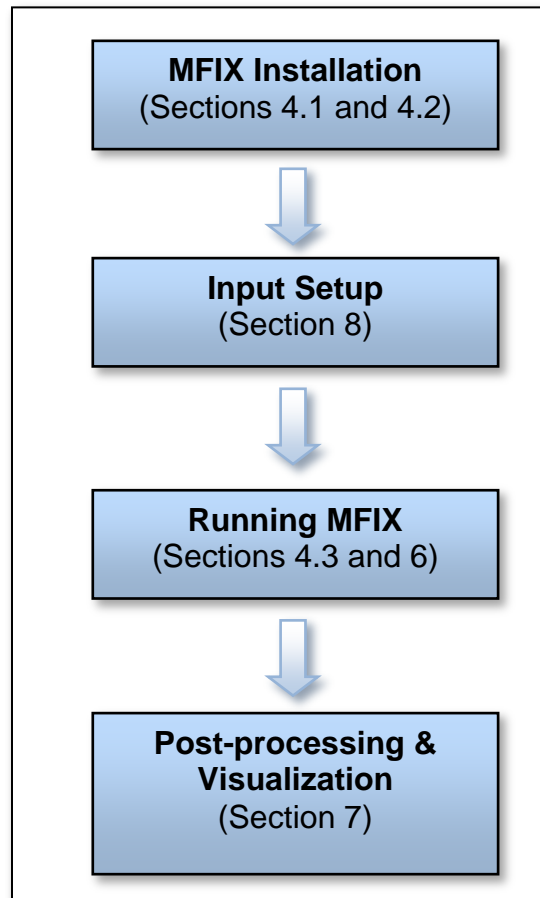
In addition to POSTMFIX, the open-source tools ParaView and VisIt can also be used to both visualize and post-process MFIX results. They are available at:

ParaView: <http://www.paraview.org/>

VisIt: <https://wci.llnl.gov/codes/visit/home.html>

Please follow the instructions on the above websites to install ParaView or VisIt.

The following chart provides a work flow overview with related sections in this document for quick review.



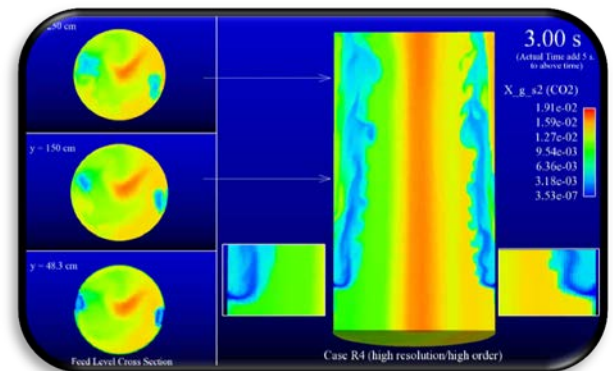
2 Development state of MFX models

MFX provides a suite of models that treat the carrier phase (typically the gas phase¹) and disperse phase (typically the solids phase) differently. Their current state of development is summarized in the tables below.

Two Fluid Model: MFX-TFM (Eulerian-Eulerian)

Both gas and solids phases are treated as interpenetrating continuum phases.

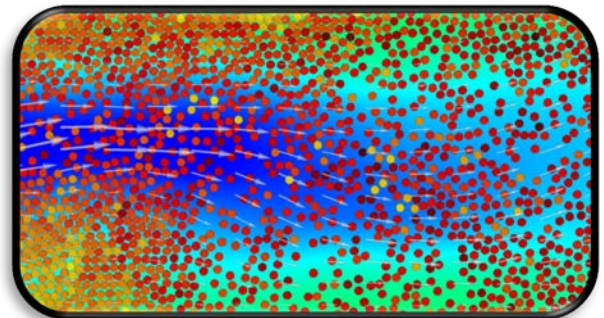
	Serial	[†] DMP	[‡] SMP
Momentum Equations	•	•	•
Energy Equations	•	•	•
Species Equations	•	•	•
Chemical Reactions	•	•	
Cartesian cut-cell	•	•	□



Discrete Element Method: MFX-DEM (Eulerian-Lagrangian)

The gas phase is treated as continuum and particles are tracked individually. Particle collisions are directly resolved.

	Serial	[†] DMP	[‡] SMP
Momentum Equations	•	•	•
Energy Equations	•		
Species Equations	•		
Chemical Reactions	•		
Cartesian cut-cell	○	○	○

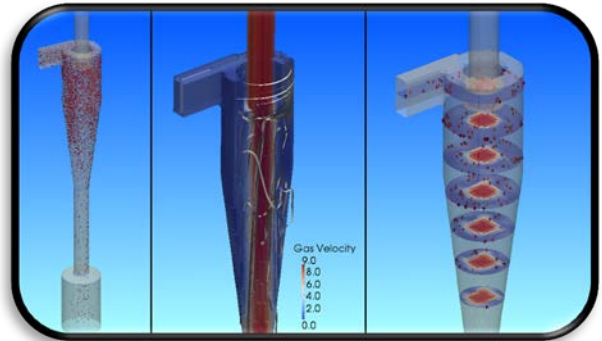


¹ For bubbly gas-liquid flows, the carrier phase would be the liquid phase and the disperse phase would be the gas bubbles.

Particle in Cell: MFX-PIC (Eulerian-Lagrangian)

The gas phase is treated as continuum and particles are represented as parcels. Collisions are not directly resolved. Simulations run faster than DEM, with a trade-off between speed and accuracy.

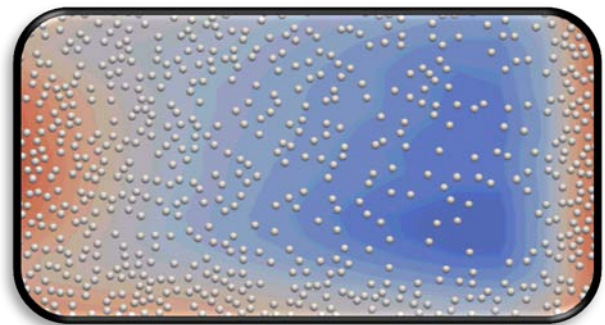
	Serial	[†] DMP	[‡] SMP
Momentum Equations	●		○
Energy Equations			
Species Equations			
Chemical Reactions			
Cartesian cut-cell	○		○



Hybrid model: MFX-Hybrid (Eulerian-Lagrangian-Eulerian)

The gas phase is treated as continuum. Each solid phase can be treated either as continuous (e.g., the dense phase) or discrete (e.g., the sparse phase).

	Serial	[†] DMP	[‡] SMP
Momentum Equations	○	○	○
Energy Equations			
Species Equations			
Chemical Reactions			
Cartesian cut-cell	○	○	○



- – implemented and fully tested
- – implemented with limited testing
- – not tested or status unknown

[†] Models not extended to DMP-parallel are only available for serial runs.

[‡] Models not extended to SMP-parallel are available for SMP runs but do not scale with thread count.

3 Release notes for 2014-1

For MFIX users upgrading to the 2014-1 Release, please note the following changes from the previous (2013-2) release:



Some feature improvements are fairly aggressive and required major changes to the code structure. As a result, **backward compatibility is not ensured for either input or restart files**. Please contact the development team through the mailing list if you have difficulties running a previous model with the new release.



This document is now referred to as the **MFIX User Guide**. It used to be called the **Readme file**, and some other documentation may still mention it, but the two designations are interchangeable.

New features:

- **Variable solids density** - The material density of a solids phase changes as a result of chemical reactions (changing species composition) with the assumption that the solids volume fraction remains constant.
- **DEM Complex boundaries** - The DEM/Cartesian cut-cell implementation was updated such that boundaries (walls) can be defined through STL files. This change allows for greater geometry complexity with better particle-wall collision detection.

Improved Features:

- DEM code refactoring for better performance gave a 4x speedup compared to the Release 2013-2 (for tutorials/FluidBed_DES with 38,400 particles running in serial.)
- DEM particle configuration is based on initial condition settings (e.g., IC_EP_s).
- Cartesian grid cut-cell re-indexing method was implemented to re-order cells when a large number of inactive cells are generated. This provides a more efficient way of looping through computational cells which can substantially speedup the code execution for complex geometries.
- The make_mfix build script was completely overhauled.
 - Build script was split into several files/scripts.
 - Build settings can be passed as build arguments.

- Parallel-make build support.
 - Automatic Makefile generation at build time for dependency detection.
 - Each supported system and compiler has its own settings file allowing for easier modification and maintenance.
- Improved checks for user input
 - Checks on solids model specifications are restricted to the solids model type so that unnecessary data is no longer needed. This improvement applies mainly to DEM and PIC solids models.
 - Error messages are consistently managed based on user input settings. If FULL_LOG is specified, error messages are written the LOG file and sent to std-out. Otherwise, error messages are only written to the LOG file.

Community Involvement:

- Improved center-line treatment for cylindrical coordinate systems.

Bakshi, A., Altantzis, C., Ghoniem, A.F. (2014). "Towards accurate three-dimensional simulation of dense multi-phase flows using cylindrical coordinates," Powder Technology, 264:242-255.

- A shape dependent particle-fluid drag coefficient was used to create a tutorial for the new user defined drag functions (see mfix/tutorials/usr_drag).

Dioguardi, F., Dellino, P., Mele, D. (2014). "Integration of a new shape-dependent particle-fluid drag coefficient law in the multiphase Eulerian-Lagrangian code MFIX-DEM," Powder Technology, 260:68-77.

4 MFIX on a UNIX/LINUX Workstation

MFIX is primarily designed to run on Linux operating systems and this section describes the preferred way of building and running MFIX.



Users should be familiar with basic Linux operating system commands and procedures before attempting to build and run MFIX.

Carefully read and follow these instructions. Check the mailing list archives if you have problems building or running MFIX as your question may have been answered previously. Information on the mailing lists is provided at the end of this document.

4.1 Extracting the MFIX directory

MFIX is distributed as a compressed source tar ball named **mfix.tar.gz**. To decompress and extract the tar file:

```
> tar xzf mfix.tar.gz
```

Here it is assumed that you are in the directory containing the tar ball. Extracting the tar ball creates a directory named **mfix** containing the MFIX and POSTMFIX source codes, tests and tutorials, as well as some additional documentation and utilities.

4.2 Building MFIX

MFIX is built with a shell script called **make_mfix**. The purpose of this script is to gather configuration information from the user, set environment variables, process reaction data, construct a Makefile, and build the executable. Ordinarily the build script does not need to be modified; however, expert users may edit it to customize build operations such as changing compilation flags or adding support for a new compiler.

4.2.1 MFIX Build Script

This section focuses on the build script and the availability of several flags. A step-by-step tutorial is presented in the following section.

Alias creation (optional)

For convenience, an alias can be created to avoid specifying the path to the MFIX build script. Assuming the MFIX source was extracted in the home directory, and you are using the C shell, an alias can be created by executing:

```
> echo "alias mkmfix ~/mfix/model/make_mfix" >> ~/.cshrc
```

This appends the quoted text to the `.cshrc` file located in the home directory. The alias will take effect the on the next login or after sourcing the `.cshrc` file. Afterwards, MFIX can be built from any directory by entering the alias, **mkmfix**. Users familiar with the creation of aliases may choose other various ways to define an alias (e.g., directly editing the `.cshrc` or `.cshrc_aliases` files). Users that use a different shell should create

an alias with the respective shell resource file (e.g. bash shell users ~/.bashrc).

Passing arguments to the build script

Arguments may be passed to the build script to specify configuration settings and invoke expert level options. The build script uses interactive prompts to collect any required information not pass as an argument. Information regarding available arguments is displayed by entering:

```
> ~/mfix/model/make_mfix --help
```

The most common arguments are given in the following table.

Argument	Description
--clean	Remove existing object and module files
--compiler= <i>NAME</i>	Specify the compiler configuration file gcc GCC compiler (gfortran) intel Intel compiler (ifort) portland Portland Group compiler (pgf90)
--debug	Enable debug flags at compile time (e.g., -g)
--default	Build with default settings. Same as passing: --serial --compiler=gcc --opt=O3
--dmp	Enable distributed memory support (MPI)
--exe= <i>NAME</i>	Specify the executable to be named <i>NAME</i>
--force-recompile	Force recompile of modules
--help	Display build script help
--long	Show all build options
--opt= <i>LEVEL</i>	Specify the optimization level O0 No optimization (enables debug flags) O1 Low optimization O2 Moderate optimization O3 Aggressive optimization The first character above is the upper case letter "O" and the second character is a digit (0 to 3)
--repeat	Repeat compilation with last build options
--serial	Build a serial executable (SMP and DMP support disabled)
--smp	Enable shared memory parallel support (OpenMP)
--skip-rxn	Skip the reaction preprocessor

Interactive prompts for build options

The MFIx build script uses interactive prompts to collect required configuration options not passed via command line arguments. A guided example of the interactive script is presented in the next section.

4.2.2 Building MFIX: A step-by-step tutorial

The following example builds the MFIX executable, **mfix.exe**, with configuration settings specified through interactive prompts. The resulting executable is used to run the **fluidbed1** tutorial. This example assumes that the GNU Fortran compiler, **gfortran**, is properly installed.



Before building, verify that a Fortran compiler is installed on your system. Contact your system administrator for assistance if necessary.

To begin, go to the **fluidbed1** tutorial directory and list its contents:

```
> cd ~/mfix/tutorial/fluidbed1
> ls
mfix.dat
```

As shown this folder contains the input file, **mfix.dat**, which specifies the simulation setup. This file is only needed to compile MFIX when user-defined reaction files are present (e.g., **usr_rates.f**). However, the **mfix.dat** file must be present when running MFIX, the last step of this tutorial.

The directory where MFIX will be run is called the run directory. Here the run directory is **fluidbed1**, and it currently contains only the **mfix.dat** file.

The build process is initiated by running the **make_mfix** script:

```
> mkmfix
```

Here, **mkmfix** is an alias pointing to the MFIX build script. If an alias was not defined, the relative path to the build script from the **fluidbed1** tutorial directory can be used:

```
> ../../model/make_mfix
```

Alternatively, the full path to the MFIX build script can be used:

```
> ~/mfix/model/make_mfix
```

Depending on your operating system, you may need to explicitly launch the MFIX build script using this alternative approach:

```
> sh ../../model/make_mfix
```

The first prompt asks the user to specify the mode of operation. Pressing the Enter key selects the default option to build a serial executable.

```
=====
Mode of execution:
=====
[1] Serial
[2] Parallel, Shared Memory (SMP)
[3] Parallel, Distributed Memory (DMP)
[4] Parallel, Hybrid (SMP+DMP)

Select the mode of execution [1] :      <      Press Enter
```

Next the user is asked to specify the optimization level. Typically O0 is reserved for debugging and development whereas O2 and O3 optimizations are used for production runs.

Pressing Enter selects the default option to build with O3 optimization.

```
=====
Compiler optimization levels:
=====
[0] Level 0: No optimization - Debug mode
[1] Level 1: O1 optimization - Low
[2] Level 2: O2 optimization - Moderate
[3] Level 3: O3 optimization - Aggressive

Select the level of optimization [3] :  <      Press Enter
```

Finally, a list of supported compilers is displayed for which the build script is configured. **The list of configured compilers does not indicate that the compilers are installed.** This example assumes that the GNU Fortran compiler, gfortran, is properly installed.

Pressing Enter selects the default GNU Fortran compiler.

```
=====
MFIX Compilation directives for following compilers:
=====
[1] GCC (gfortran) version 4.3 and above
[2] Portland Group (pgf90) version 11.7 and above
[3] Intel (ifort) version 11.1 and above

Select the compiler to compile MFIX? [1]  <      Press Enter
```

The build may take several minutes to complete. If the build is successful, the executable is copied into the run directory and the following message is displayed.

```
*****
Compilation successful: mfix.exe created
To run MFIX type: ./mfix.exe
*****
```

Finally, the simulation is started by entering.

```
> ./mfix.exe
```

Additional details about running MFIX are provided in the subsequent section.

Users that compile MFIX often may find passing an argument list to the build script more practical than answering each question interactively. The above steps could be performed in single command step as follows:

```
> mkmfix --serial --opt=03 --compiler=gcc
```

or

```
> mkmfix --default
```

The compilation flags can even be included in an alias.

4.3 Running MFIX

This section provides basic guidance for running MFIX in serial, SMP, and DMP modes. Contact your system administrator about the policies and procedures for running on your system as you may be required to submit jobs through a queuing system. MFIX requires an mfix.dat file in the directory it is run from. For instance, to run the fluidbed1 tutorial:

```
> cd ~/mfix/tutorial/fluidbed1
> ls
mfix.dat      mfix.exe
```

For your own simulation, you can start with one of the predefined mfix.dat files and customize it. See Section 7 for a full list of keyword options available in mfix.dat.

4.3.1 Serial Execution

MFIX can be launched in different ways in order to manage screen output. The simplest way is to directly run the executable:

```
> ./mfix.exe
```

This approach runs MFIX in the foreground and output messages from MFIX are sent to the terminal.

If you want to run your MFIX job in the background, redirect MFIX output to a log file:

```
> ./mfix.exe >& run.log &
```


You can check the progress of your job by printing the log file to your terminal:

```
> tail -f run.log
```

To stop the tail command type Ctrl+C at any time. This will not interrupt your job.

4.3.2 SMP Execution

Shared memory parallel (SMP) utilizing OpenMP directives, is enabled by either selecting the shared memory option when prompted by the build script or through passing the SMP flag to the build script.

```
> mkmfix --smp
```

The number of threads are set at runtime through the environment variable OMP_NUM_THREADS. For example, four threads can be used during a simulation in a bash shell session by entering:

```
> export OMP_NUM_THREADS=4
```

The same environment variable can be set in csh by entering:

```
> set OMP_NUM_THREADS=4
```

After the OMP_NUM_THREADS variable is specified, an SMP job is launched in the same manner as a serial run. For example, to run your SMP MFIX job in the background and redirect messages to a log file:

```
> ./mfix.exe >& run.log &
```

4.3.3 DMP Execution

Distributed memory parallel (DMP) support through Message Passing Interface (MPI) is enabled by either selecting the distributed memory option when prompted by the build script or through passing the DMP flag to the build script:

```
> mkmfix --dmp
```

The number of MPI ranks is specified at runtime. The actual command to run the job is specific to the compiler, MPI library, and system used to build MFIX. For example, an MFIX executable built with the GNU Fortran compiler and OpenMPI libraries **might be** launched with four MPI processes and run in the background with standard I/O messages redirected to a log file by entering:

```
> mpirun -np 4 ./mfix.exe >& run.log &
```

Again, the actual command can vary from system-to-system.



Verify that your MPI installation works properly if you intend to compile and run the DMP version of MFIX. Contact your system administrator if you are not sure how to proceed.
(see [mfix/tools/mfi/ MPI_verification_for_MFIX.pdf](#))

4.4 Building POSTMFIx

POSTMFIx is built by the script **make_post**. The following example builds the POSTMFIx executable, **post_mfix**, with the configuration information supplied through the interactive prompts. This example assumes that the GNU Fortran compiler (gfortran) is installed.



Before building POSTMFIx, verify that there is a working Fortran compiler on your system. Contact your system administrator if you have questions regarding the availability of Fortran compilers.

Go to the `post_mfix` directory and launch the build script

```
> cd ~/mfix/post_mfix
> ./make_post
```

The build script displays a list of all supported compilers (which may or may not be necessarily installed on your system).

Pressing Enter selects the default GNU Fortran compiler.

```
=====
POSTMFIx Compilation directives for following compilers:
=====
```

- [1] GCC (gfortran) version 4.3 and above
- [2] Portland Group (pgf90) version 11.7 and above
- [3] Intel (ifort) version 11.1 and above

```
Select the compiler to compile POSTMFIx? [1]      ← Press Enter
```

The build may take several minutes to complete. Upon successful compilation, the executable is copied into the current directory and the following message is output.

```
*****
Compilation successful: post_mfix created
To run POSTMFIx type: ./post_mfix
*****
```

Once built, `post_mfix` may be used to extract data from a particular location in the domain or preform some basic averaging of flow field data.

4.5 Enabling NetCDF Output

NetCDF is an open standard format for scientific data that may perform better than RES files for some large simulations. Unlike the other output formats, NetCDF support requires an external library to be linked with MFIx at build and runtime. To build MFIx

with NetCDF support, first set the environment variables for the NetCDF include directory, NETCDF_INCLUDE, and the NetCDF library, NETCDF_LIB:

```
> export NETCDF_INCLUDE=/path/to/netcdf/include/directory
> export NETCDF_LIB=/path/to/netcdf/lib/directory
```

Contact your system administrator for information regarding the availability and usage of NetCDF.

Next, specify the flag to enable NetCDF support when launching the MFIX build script:

```
> make_mfix --enable-netcdf [... usual options ...]
```

At runtime, it may be necessary to set LD_LIBRARY_PATH to include NetCDF if it is not installed in a standard location.

```
> export LD_LIBRARY_PATH=/path/to/netcdf/lib/directory
> ./mfix.exe
```

Again, contact your system administrator for information on how to use external libraries such as NetCDF.

Use the keyword BRWRITE_NETCDF to enable NetCDF output (See section 8.11). MFIX generates NetCDF output files as <RUN_NAME>_###.nc, which can be loaded into ParaView.

5 MFIX on Windows

MFIX is primarily developed and tested on Linux operating systems. Support for building and running MFIX on Windows systems is extremely limited. If you do not currently have access to a Linux system, the Windows version of MFIX may be useful for evaluation. However, the Linux version is recommended for production runs.

5.1 Prebuilt Windows binaries for MFIX

A prebuilt serial MFIX binary for running on Windows systems is available on the download page (<https://mfix.netl.doe.gov/members/download.php>). The prebuilt binary is sufficient unless the MFIX simulation includes user defined files (usr*.f) or chemical reactions.

Additional information is available on the website and in the mailing list archives.
<https://mfix.netl.doe.gov/members/wininst.html>

5.2 Building MFIX with Cygwin

Cygwin can be used to build a native Windows executable. An overview of the steps to build MFIX under Cygwin is given below.

5.2.1 Installing Cygwin

1. Download the Cygwin installer [setup-x86.exe](#) (32-bit installation) or [setup-x86_64.exe](#) (64-bit installation) from <http://www.cygwin.com/>.
2. Run the installer (if you do not have Windows administrator rights, run the installer on the command line with the “--no-admin” option.)
3. Select the default install options
4. At the installation step “Select Packages”, install the packages:
 - gcc-fortran
 - make
 - diffutils
5. You can rerun the Cygwin installer anytime if you want to install additional development tools.

5.2.2 Building and Running MFIX

Once Cygwin is installed, the build process is the same as on Linux. See the section “Building MFIX” for more details.

1. Download MFIX source tarball
2. Extract the tar ball

```
$ tar xzf mfix.tar.gz
```

3. Go to fluidbed1 tutorial

```
$ cd mfix/tutorials/fluidbed1
```

4. Run the MFIX build script

```
$ ../../model/make_mfix
```

5. Run MFIX for the fluidbed1 tutorial

```
$ ./mfix.exe
```

(Optional) To run MFIX in the background

```
$ nohup ./mfix.exe > run.log&
```

While it is running in the background, you can follow the output with:

```
$ tail -f out1
```

If you have problems building or running MFIX in Cygwin, check the mailing list archives to see if your question has already been answered. Information on the mailing lists and mailing list archives are provided at the end of this document.

6 MFIX at Run Time

6.1 MFIX Output and Messages

MFIX generates several output files during a simulation. The following table provides an overview of these files and their purpose.

Extension	Type	Description
.LOG	ASCII	Contains messages about the run. Processor specific log files may be generated for DMP runs.
.OUT	ASCII	Echoes the input file and shows the numerical cell distribution. Field variables are also written if OUT_DT is defined in the mfix.dat file.
.RES	Binary	Double precision file used for restarts.
.SPx	Binary	Multiple single precision files used to store transient simulation data for post processing.

If the data file specifies FULL_LOG = .TRUE., then the progress of the run is displayed in the terminal as shown below. The normalized residuals for various equations are written out for each iteration and every time-step.

Time = 0.92965 Dt = 0.13930E-02 CPU time left = 54.672 s							
Nit	P0	P1	U0	V0	U1	V1	Max Res
1	1.5E-02	2.5E+01	1.3E-03	3.3E-03	3.2E-03	6.3E-03	P0
2	1.	5.0E-02	2.8E-03	6.3E-03	2.5E-03	3.8E-03	P1
3	0.1	2.1E-02	1.2E-03	3.7E-03	1.1E-03	1.8E-03	P0
4	5.2E-02	8.1E-03	6.5E-04	2.1E-03	4.8E-04	8.6E-04	P0
5	3.0E-02	3.9E-03	4.1E-04	1.3E-03	2.5E-04	4.3E-04	P0

The first line shows the simulation time, time-step, and the approximate CPU time needed to complete the run. The remaining CPU time might not be accurate, especially in the beginning of a run. Subsequent lines display the iteration number, the normalized residuals for various equations (e.g., gas continuity, solids continuity, x and y gas momentum, and x and y solids momentum), and the equation with the maximum residual. The residuals P0 and P1 are normalized only when Nit>1. The residuals displayed can be specified with the keyword RESID_STRING.

MFIX uses a variable time step to reduce run time by automatically adjusting the step size within user-defined limits. As a result iterations may not converge at larger time-step sizes. When this happens, the time step size is reduced successively until convergence is obtained. Messages about divergence and recovery are recorded in the .LOG file and displayed in the terminal if FULL_LOG = .TRUE..

MFIX reports errors while reading and processing the mfix.dat file, and during a run. Errors in reading the data file and in opening files are reported to the terminal. All other errors are written to the.LOG. Additionally, runtime errors are displayed to the terminal if

FLULL_LOG=.TRUE..

While reporting errors in reading the data file, MFIX displays the problematic line of input, so that the error can be easily detected. Possible causes of error are (1) incorrect format for the name-list input, (2) unknown (possibly misspelled) keyword, or (3) the dimension of the name-list item is too small. For example, if the dimension of DX is set as 5000 (DIM_I in *param_mod.f*), and if the input data file contains an entry DX(5001), MFIX will report an input processing error.

While processing the input data, MFIX will report errors if the data specified is insufficient or physically unrealistic. MFIX will supply default values only when it is certain that giving a default value is reasonable.

An occasional input-processing error is the inability to determine the flow plane for a boundary condition. The boundary planes defined in the input data file must have a wall-cell on one side and a fluid-cell on the other side. If the initial condition is not specified for the fluid-cell, MFIX will not recognize the cell as a fluid-cell and, hence, MFIX will be unable to determine the flow plane.

Every NLOG number of time steps, MFIX checks whether that

- mass fractions sum to 1.0
- overall interphase mass transfer sums to zero
- viscosities, conductivities, and specific heats are nonnegative
- temperatures are within specified bounds

A message will be printed out if any errors are encountered. The run may be aborted depending upon the severity of the error. Every NLOG time steps, MFIX will print out the number of iterations during the previous time step and the total solids inventory in the reactor.

For the specified mass-outflow condition, after the elapse of time BC_DT_0, MFIX prints out time-averaged mass flow rates. For cyclic boundary conditions, MFIX will print out the volume averaged mass fluxes every NLOG time step.

A message is written to the .LOG file whenever the .RES and .SPx files are written. This message also shows an approximate value of the cumulative disk space usage in megabytes.

6.2 Restarting a Run

A stopped run can be restarted by launching MFIX after specifying RUN_TYPE = 'restart_1' in mfix.dat. The old .OUT file will be overwritten and new messages are appended to the existing .LOG file.

6.3 When the Run does Not Converge

Initial non-convergence:

Ensure that the initial conditions are physically realistic. If in the initial time step, the run displays NaN (Not-a-Number) for any residual, reduce the initial time step. If time step reductions do not help, recheck the problem setup.

Holding the time step constant (`DT_FAC=1`) and ignoring the stalling of iterations (`DETECT_STALL=FALSE.`) may help in overcoming initial nonconvergence. Often a better initial condition will aid convergence. For example, using a hydrostatic rather than a uniform pressure distribution as the initial condition will aid convergence in fluidized-bed simulations.

If there are computational regions where the solids tend to compact (i.e., solids volume fraction less than `EP_star`), convergence may be improved by reducing `UR_FAC(2)` below the default value of 0.5.

Convergence is often difficult with higher order discretization methods. First order upwinding may be used to overcome initial transients and then the higher order method may be turned on. Also, higher-order methods such as van Leer and minmod give faster convergence than methods such as superbee and ULTRA-QUICK.

7 Post Processing and Visualization

MFIx can generate output data in several formats for visualization and analysis. The command line tool POSTMFIx is distributed with MFIx. In addition the open source GUI tools ParaView (www.paraview.org version 4.1 or later recommended) and VisIt (<https://wci.llnl.gov/simulation/computer-codes/visit/>) support some of the MFIx output file formats.

Filename	File Type	Fluid data?	Particle data?	Process with POSTMFIx?	Open in ParaView?	Open in VisIt?
<RUN_NAME>.RES	MFIx Binary	Yes	No	Yes	Yes	Yes
<RUN_NAME>_###_DES.RES	MFIx Binary	No	Yes	No	No	No
<RUN_NAME>_###.nc	NetCDF	Yes	No	No	Yes	Yes
<RUN_NAME>.pvd	VTK	Yes	No	No	Yes	No
<RUN_NAME>_###.vtu	VTK	Yes	No	No	Yes	Yes
<RUN_NAME>_DES.pvd	VTK	No	Yes	No	Yes	No
<RUN_NAME>_DES_###.vtp	VTK	No	Yes	No	Yes	Yes

Note: .pvd files only contain information linking the respective .vtu and .vtp files to time-step information

7.1 Running POSTMFIx

The POSTMFIx tool is used for reading the binary MFIx .SPx output files and outputting data in text files. The following walk through demonstrates how to run POSTMFIx on the results from the FluidBed_DES tutorial with the default settings. The post_mfix executable is assumed to be built or copied into the run directory.

After launching the executable, enter the run name.

```
> ./post_mfix
Enter the RUN_NAME to post_process > DES_FB1    ← DES_FB1<Enter>
```

A simple menu of options is presented. Type '1' to examine/print data and press Enter.

```
*****
0  - Exit POST_MFIx
1  - Examine/print data
2  - Write .RES from data in .SPx files
3  - Write .RES for a new grid, using old data
4  - Calculate miscellaneous quantities
5  - Print out variables
6  - Call user defined subroutine USR_POST
7  - Write a new SPx file with selected records
8  - Write new SPx files with time averaged data
9  - Perform ORNL calculations
10 - run scavenger code
*****
Enter menu selection > 1

Interactive data retrieval program. Type ? any time for help,
or press RETURN to select default values shown in parenthesis.
```

Type F to use the default data precision and press Enter.

Write output using user-supplied precision? (T/F) F ← **F<Enter>**

Enter a time range for data extraction and analysis. The default simulation has a simulation length of one second, so enter a range from 0.1 seconds to 0.9 seconds. The next prompt asks if the data should be time averaged. Press Enter to skip the averaging.

Time: (0.000, 0.000) > 0.1, 0.9
Time average ? (N) > ← **<Enter>**

Enter the variable of interest. The default is the gas phase volume fraction, EP_G. A complete list of possible entries is given by typing '?' and Enter. Press Enter to select the gas phase volume fraction.

Variable: (EP_g) > ← **<Enter>**

Next enter the spatial range to extract the data. This requires an understanding of the I/J/K values for your simulation. Basic geometric information for the simulation is provided in the .OUT file. For this example, we will take a vertical slice from the approximate center of the 2D domain.

I range: (1, 1) >8, 8 ← **8, 8<Enter>**
J range: (1, 1) >2, 40 ← **2, 40<Enter>**
Average or sum over J? (N) > ← **<Enter>**
K range: (1, 1) > ← **<Enter>**

Specify where to output the data. Press Enter to select the default *, printing the data to the terminal. Alternatively, specify a file name to save the data.

File: (*) > ← **<Enter> prints to screen**
(filename<Enter> saves to filename)

```
X = 6.5000
Z = -0.20000
Time = 0.10075
      Y                EP_g
      1.0000          0.43971
      3.0000          0.45994
      ....etc....
```

Return to the original time prompt to continue data extraction or analysis. Type 'q' and Enter to return to the main menu. From the main menu, type '0' and Enter to exit.

Time: (1.000, 1.000) > q ← **q<Enter> to quit**

< main menu is displayed again >


Enter menu selection > 0 ← **0<Enter> to exit**


7.2 ParaView

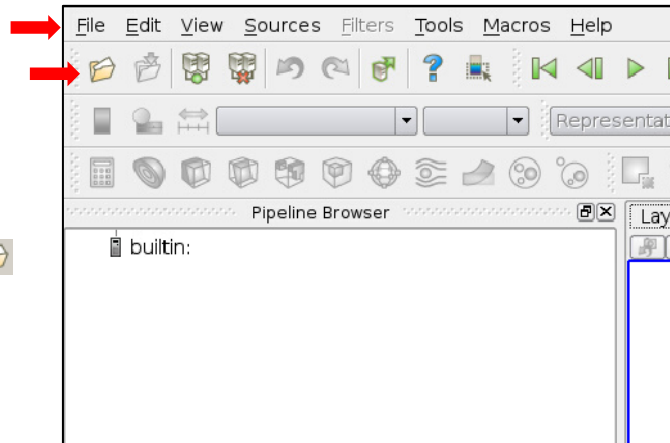
This walk through demonstrates how to use ParaView to visualize the results of the FluidBed_DES tutorial. It is assumed that the FluidBed_DES tutorial was successfully run with default settings and that ParaView is installed on your system. First the tutorial demonstrates how to visualize the fluid field; then the DEM particles are added using spherical glyphs.

Fluid field results for a standard structured mesh are displayed in ParaView by loading the .RES file. To open the files select

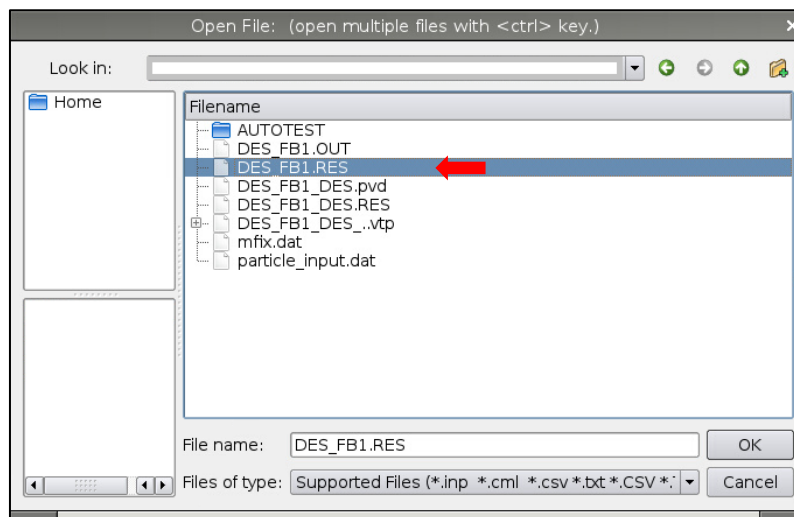
File > Open

Similarly, you can click on the icon 


Once the file is loaded, you need to click on the green “Apply” button  to load all of the variables.




When prompted, select the **DES_FB1.RES** file. **Do not load the DES restart file (DES_FB1_DES.RES)**. This binary file is not supported by any visualization software. It only contains restart data and will likely cause ParaView to crash if loaded.

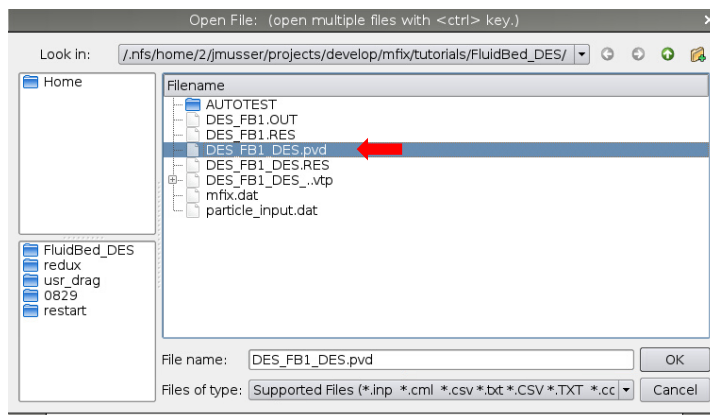


Newer versions of ParaView require that you select the appropriate reader. Choose to open the data with the ‘MFIx Unstructured Grid Files’.

ParaView typically displays the gas phase volume fraction once the data is loaded. It may be necessary to rescale the data range as the initial range may only be suitable for displaying the initial conditions. This can be done by clicking the  button.


DEM and PIC particle simulation data is loaded into ParaView by opening the **.pvd** file.

Again, click on the green “Apply” button  to load all of the variables.



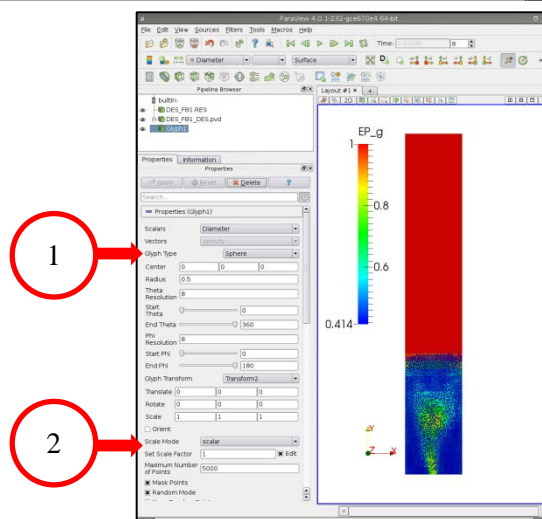
Particles are shown by applying a glyph to the dataset. To apply a glyph filter:

Filters > Alphabetical > Glyph

Similarly you can click the icon: 

Commonly, particle data is represented:

1. ‘Sphere’ glyph type and
2. Scalar scale mode with a scale factor



Note that these screenshots may appear differently depending on your version of ParaView.

7.3 VisIt

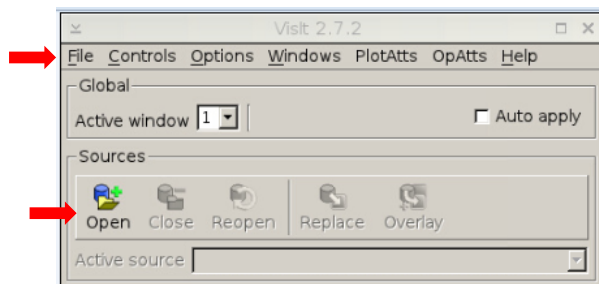
This walk through demonstrates how to use VisIt to visualize the results of the FluidBed_DES tutorial. It is assumed that the FluidBed_DES tutorial was successfully run with default settings and that VisIt is installed on your system. First the tutorial demonstrates how to visualize the fluid field; then the DEM particles are added using spherical glyphs. The screen shots and menu structure were taken from VisIt version 2.7.2 and may be different in different versions.

Fluid field results for a standard structured mesh are displayed in VisIt by loading the **.RES** file.

To open the files select

File > Open

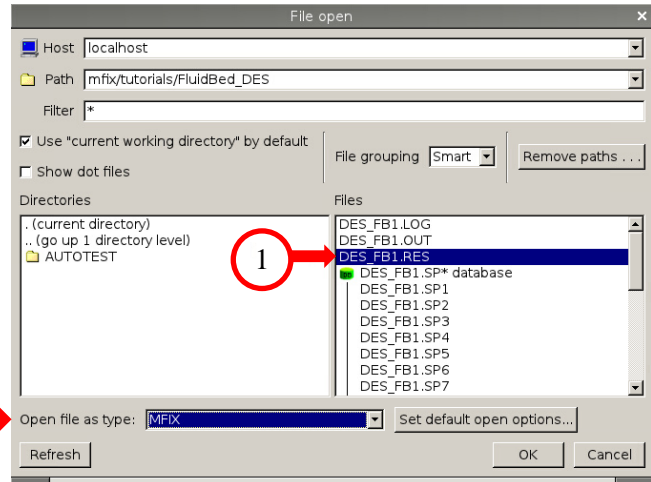
Similarly, you can click on the ‘Open’ icon.



In the 'File Open' dialog:

- (1) Select the **DES_FB1.RES** file. **Do not load the DES restart file (DES_FB1_DES.RES)**. This binary file is not supported by any visualization software. It only contains restart data and will likely cause VisIt to crash if loaded.

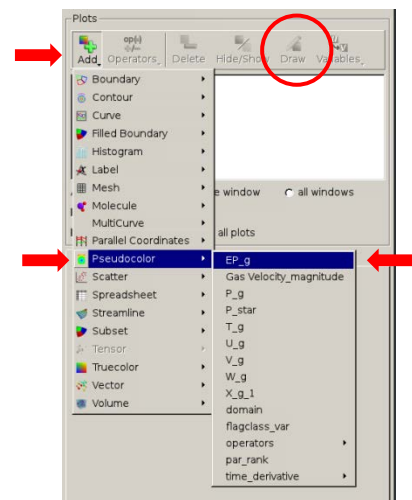
- (2) Specify the 'Open file as type' as MFIx. Failing to select MFIx as the file type typically results in an error.



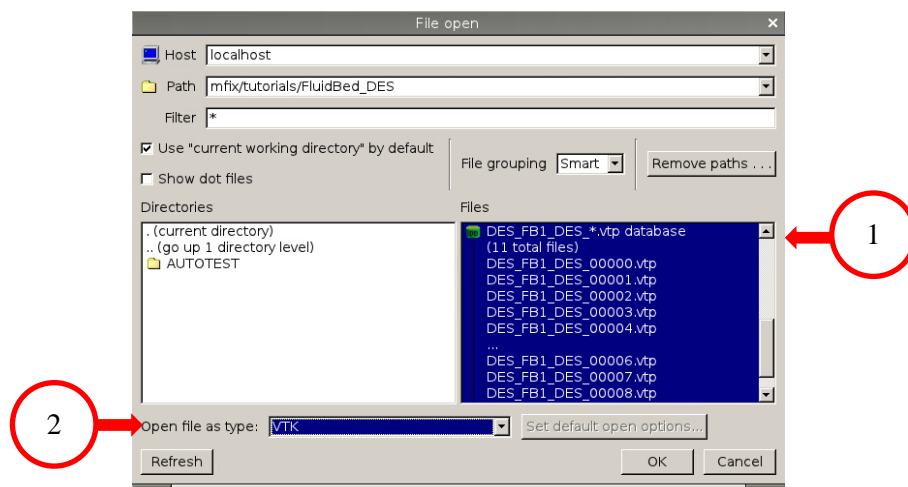
The gas phase volume fraction is plotted by



Once a field variable is selected, it is displayed by clicking the Draw button. This button is unavailable until an item is selected from the Plots menu.



DEM and PIC particle simulation data is loaded into VisIt by opening the (1) ***.vtp database**. (2) Specify the 'Open file as type' as VTK. Failing to select VTK as the file type may result in an error.



The gas phase volume fraction is plotted by

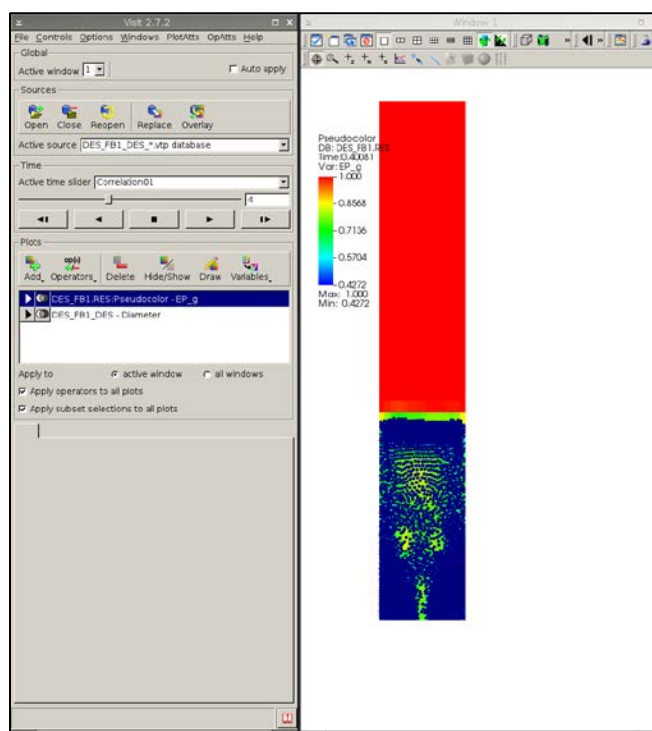
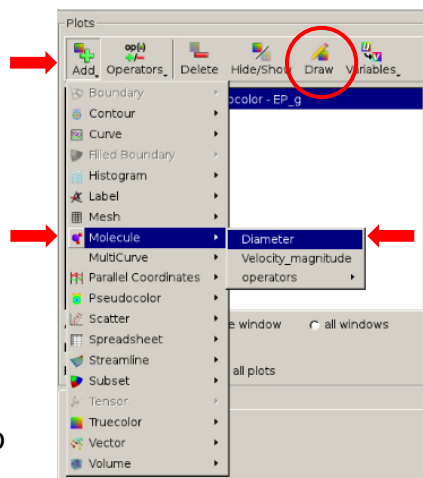


Display the particles by clicking the Draw

A dialog to edit the particle display properties is found by selecting:

PlotAtts > Molecule...

It may be necessary to adjust the “Fixed atom radius” to properly display the particles.



8 Keywords in Input Data File (mfix.dat)

The symbols used in the table are as follows:

Dimension	Description
1	Cell number in x, y, or z direction
M	Solids-phase number
N	Species number
IC	Initial condition number
BC	Boundary condition number
IS	Internal surface number
USR	User-defined output number

[DEFAULT VALUE] indicates a default value.

8.1 Run Control

This section contains basic run time information. The flags for enabling/disabling various governing equations are provided.

RUN_NAME *Required* CHARACTER
Name used to create output files. The name should generate legal file names after appending extensions.
Ex: Given the input, RUN_NAME = "bub01", MFIX will generate the output files: BUB01.LOG, BUB01.OUT, BUB01.RES, etc.

DESCRIPTION CHARACTER
Problem description. Limited to 60 characters.

UNITS *Required* CHARACTER
Simulation input/output units.
CGS -----All input and output in CGS units (g, cm, s, cal).
SI -----All input and output in SI units (kg, m, s, J).

RUN_TYPE *Required* CHARACTER
Type of run.
NEW -----A new run. There should be no .RES, .SPx, .OUT, or .LOG files in the run directory.
RESTART_1 -----Traditional restart. The run continues from the last time the .RES file was updated and new data is added to the SPx files.
RESTART_2 -----Start a new run with initial conditions from a .RES file created from another run. No other data files (SPx) should be in the run directory.

TIME DOUBLE PRECISION
Simulation start time. This is typically zero.

TSTOP

DOUBLE PRECISION

Simulation stop time.

DT

DOUBLE PRECISION

Initial time step size. If left undefined, a steady-state calculation is performed.

DT_MAX

DOUBLE PRECISION

Maximum time step size.

DT_MIN

DOUBLE PRECISION

Minimum time step size.

DT_FAC

DOUBLE PRECISION

Factor for adjusting time step [0.9].

- The value must be less than or equal to 1.0.
- A value of 1.0 keeps the time step constant which may help overcome initial non-convergence.

AUTO_RESTART

LOGICAL

Flag to restart the code when $DT < DT_MIN$.

MOMENTUM_X_EQ(Phase)

LOGICAL

Flag to enable/disable solving the X-momentum equations.

[.TRUE.] ----- Solve X-momentum equations.

[.FALSE.] ----- The X velocity initial conditions persist throughout the entire simulation.

MOMENTUM_Y_EQ(Phase)

LOGICAL

Flag to enable/disable solving the Y-momentum equations.

[.TRUE.] ----- Solve Y-momentum equations.

[.FALSE.] ----- The Y velocity initial conditions persist throughout the entire simulation.

MOMENTUM_Z_EQ(Phase)

LOGICAL

Flag to enable/disable solving the Z-momentum equations.

[.TRUE.] ----- Solve Z-momentum equations.

[.FALSE.] ----- The Z velocity initial conditions persist throughout the entire simulation.

ENERGY_EQ

LOGICAL

Solve energy equations.

[.FALSE.] ----- Do not solve energy equations.

[.TRUE.] ----- Solve energy equations.

SPECIES_EQ(Phase)

LOGICAL

Solve species transport equations.

[.FALSE.] ----- Solve species equations.

[.TRUE.] ----- Do not solve species equations.

GRANULAR_ENERGY

LOGICAL

Granular energy formulation selection.

Applies to solids models: TFM

[.FALSE.] ----- Use algebraic granular energy equation formulation.
 .TRUE. ----- Use granular energy transport equation (PDE) formulation.

K_EPSILON

LOGICAL

The K-Epsilon turbulence model (for single-phase flow).

- Numerical parameters (like under-relaxation) are the same as the ones for SCALAR (index = 9).
- The maximum value of gas turbulent viscosity, MU_GMAX, must be defined.
- There is no default and the code will not run if MU_GMAX is not set. A value MU_GMAX = 1.E+03 is recommended. (see calc_mu_g.f)
- All walls must be defined (NSW, FSW or PSW) in order to use standard wall functions. If a user does not specify a wall type, the simulation will not contain the typical turbulent profile in wall-bounded flows.

.TRUE. ----- Enable the K-epsilon turbulent model (for single-phase flow) using standard wall functions.
 [.FALSE.] ----- Do not use K-epsilon turbulent model

L_SCALE0

DOUBLE PRECISION

Value of turbulent length initialized [0.0]. This may be overwritten in specific regions with the keyword IC_L_SCALE. A zero equation mixing-length turbulence model. Use this to specify a constant value of turbulent length.

MU_GMAX

DOUBLE PRECISION

Maximum value of the turbulent viscosity of the fluid. Must be set if K_EPSILON or L_SCALE0 are used. No default exists, however, a value of 1000 is recommended.

NSCALAR

INTEGER

The number of user-defined scalar transport equations to solve.

PHASE4SCALAR(Scalar Equation)

INTEGER

The phase convecting the indexed scalar transport equation.

DRAG_TYPE

CHARACTER

Available gas-solids drag models. Note: The extension `_PCF` following the specified drag model indicates that the polydisperse correction factor is available. For PCF details see:

- Van der Hoef MA, Beetstra R, Kuipers JAM. (2005) Journal of Fluid Mechanics.528:233-254.
- Beetstra, R., van der Hoef, M. A., Kuipers, J.A.M. (2007). AIChE Journal, 53:489-501.
- Erratum (2007), AIChE Journal, Volume 53:3020

SYAM_OBRIEN ----- Syamlal M, OBrien TJ (1988). International Journal of Multiphase Flow 14:473-481. Two additional parameters may be specified: DRAG_C1, DRAG_D1

GIDASPOW ----- Ding J, Gidaspow D (1990). AIChE Journal 36:523-538

GIDASPOW_BLEND ----- Lathouwers D, Bellan J (2000). Proceedings of the 2000 U.S. DOE Hydrogen Program Review NREL/CP-570-28890.

WEN_YU ----- Wen CY, Yu YH (1966). Chemical Engineering Progress Symposium Series 62:100-111.

KOCH_HILL ----- Hill RJ, Koch DL, Ladd JC (2001). Journal of Fluid Mechanics, 448: 213-241. and 448:243-278.

BVK ----- Beetstra, van der Hoef, Kuipers (2007). Chemical Engineering Science 62:246-255

HYS ----- Yin, X, Sundaresan, S. (2009). AIChE Journal 55:1352-1368 This model has a lubrication cutoff distance, LAM_HYS, that can be specified.

USER_DRAG ----- Invoke user-defined drag law. (usr_drag.f)

GIDASPOW_PCF ----- see GIDASPOW

GIDASPOW_BLEND_PCF ---- see GIDASPOW_BLEND

WEN_YU_PCF ----- see WEN_YU

KOCH_HILL_PCF ----- see KOCH_HILL

DRAG_C1

DOUBLE PRECISION

Quantity for calibrating Syamlal-O'Brien drag correlation using Umf data [0.8]. This is determined using the Umf spreadsheet.

DRAG_D1

DOUBLE PRECISION

Quantity for calibrating Syamlal-O'Brien drag correlation using Umf data [2.65]. This is determined using the Umf spreadsheet.

LAM_HYS

DOUBLE PRECISION

The lubrication cutoff distance for HYS drag model [1 micron]. In practice this number should be on the order of the mean free path of the gas for smooth particles, or the rms roughness of a particle if they are rough (if particle roughness is larger than the mean free path).

SUBGRID_TYPE

CHARACTER

Subgrid models.

Applies to solids models: TFM

IGCI ----- Igci, Y., Pannala, Benyahia, S., and Sundaresan S. (2012). Industrial & Engineering Chemistry Research, 2012, 51(4):2094-2103

MILIOLI ----- Milioli, C.C., Milioli, F. E., Holloway, W., Agrawal, K. and Sundaresan, S. (2013). AIChE Journal, 59:3265-3275.

FILTER_SIZE_RATIO

DOUBLE PRECISION

Ratio of filter size to computational cell size [2.0].

Applies to solids models: TFM

SUBGRID_WALL

LOGICAL

Flag for subgrid wall correction.

Applies to solids models: TFM

[.FALSE.] ----- Do not include wall correction.

.TRUE. ----- Include subgrid wall correction.

MODEL_B

LOGICAL

Shared gas-pressure formulation. See Syamlal, M. and Pannala, S. "Multiphase continuum formulation for gas-solids reacting flows," chapter in Computational Gas-Solids Flows and Reacting Systems: Theory, Methods and Practice, S. Pannala, M. Syamlal and T.J. O'Brien (editors), IGI Global, Hershey, PA, 2011.

[.FALSE.] ----- Use Model A

.TRUE. ----- Use Model B. Bouillard, J.X., Lyczkowski, R.W., Folga, S., Gidaspow, D., Berry, G.F. (1989). Canadian Journal of Chemical Engineering 67:218-229.

8.2 Physical Parameters

P_REF

DOUBLE PRECISION

Reference pressure [0.0].

P_SCALE

DOUBLE PRECISION

Scale factor for pressure [1.0].

GRAVITY

DOUBLE PRECISION

Gravitational acceleration [980.7 in CGS; 9.87 in SI].

GRAVITY_X

DOUBLE PRECISION

X-component of gravitational acceleration vector. By default, the gravity force acts in the negative y-direction.

GRAVITY_Y

DOUBLE PRECISION

Y-component of gravitational acceleration vector. By default, the gravity force acts in the negative y-direction.

GRAVITY_Z

DOUBLE PRECISION

Z-component of gravitational acceleration vector. By default, the gravity force acts in the negative y-direction.

8.3 Numerical Parameters

This section contains keywords for defining numerical parameters. Keywords related to solving the governing equations (e.g., LEQ_IT, DISCRETIZE, UR_FAC, etc.) are dimensioned for the nine types of equations:

Index	Equation Type
1	Gas Pressure
2	Solids Volume Fraction
3	Gas and Solids U Momentum Equation
4	Gas and Solids V Momentum Equation
5	Gas and Solids W Momentum Equation
6	Gas and Solids Energy Equations (Temperature)
7	Gas and Solids Species Mass Fractions
8	Granular Temperature
9	User-Defined Scalar and K-Epsilon Equation

For example, LEQ_IT(3) = 10, specifies to use 10 iterations within the linear equation solver for the U Momentum Equations (of type 3).

MAX_NIT

INTEGER

Maximum number of iterations [500].

NORM_G

DOUBLE PRECISION

Factor to normalize the gas continuity equation residual. The residual from the first iteration is used if NORM_G is left undefined. NORM_G=0 invokes a normalization method based on the dominant term in the continuity equation. This setting may speed up calculations, especially near a steady state and incompressible fluids. But, the number of iterations for the gas phase pressure should be increased, LEQ_IT(1), to ensure mass balance

NORM_S

DOUBLE PRECISION

Factor to normalize the solids continuity equation residual. The residual from the first iteration is used if NORM_S is left undefined. NORM_S = 0 invokes a normalization method based on the dominant term in the continuity equation. This setting may speed up calculations, especially near a steady state and incompressible fluids. But, the number of iterations for the solids volume fraction should be increased, LEQ_IT(2), to ensure mass balance.

TOL_RESID

DOUBLE PRECISION

Maximum residual at convergence (Continuity + Momentum) [1.0d-3].

TOL_RESID_T

DOUBLE PRECISION

Maximum residual at convergence (Energy) [1.0d-4].

TOL_RESID_X

DOUBLE PRECISION

Maximum residual at convergence (Species Balance) [1.0d-4].

TOL_RESID_TH

DOUBLE PRECISION

Maximum residual at convergence (Granular Energy) [1.0d-4].

TOL_RESID_SCALAR

DOUBLE PRECISION

Maximum residual at convergence (Scalar Equations) [1.0d-4].

TOL_RESID_K_EPSILON

DOUBLE PRECISION

Maximum residual at convergence (K_Epsilon Model) [1.0d-4].

TOL_DIVERGE

DOUBLE PRECISION

Minimum residual for declaring divergence [1.0d+4]. This parameter is useful for incompressible fluid simulations because velocity residuals can take large values for the second iteration (e.g., 1e+8) before dropping down to smaller values for the third iteration.

DETECT_STALL

LOGICAL

Reduce the time step if the residuals stop decreasing. Disabling this feature may help overcome initial non-convergence.

.FALSE. ----- Continue iterating if residuals stall.
[.TRUE.] ----- Reduce time step if residuals stall.

LEQ_METHOD(Equation ID Number)

INTEGER

LEQ Solver selection. BiCGSTAB is the default method for all equation types.

1 ----- SOR - Successive over-relaxation
2 ----- BiCGSTAB - Biconjugate gradient stabilized.
3 ----- GMRES - Generalized minimal residual method
5 ----- CG - Conjugate gradient

LEQ_TOL(Equation ID Number)

DOUBLE PRECISION

Linear Equation tolerance [1.0d-4].

LEQ_IT(Equation ID Number)

INTEGER

Number of iterations in the linear equation solver.

- 20 iterations for equation types 1-2
- 5 iterations for equation types 3-5
- 15 iterations for equation types 6-9

LEQ_SWEEP(Equation ID Number)

CHARACTER

Linear equation sweep direction. This applies when using GMRES or when using the LINE preconditioner with BiCGSTAB or CG methods. 'RSRS' is the default for all equation types.

RSRS ----- (Red/Black Sweep, Send Receive) repeated twice
ISIS ----- (Sweep in I, Send Receive) repeated twice
JSJS ----- (Sweep in J, Send Receive) repeated twice
KSKS ----- (Sweep in K, Send Receive) repeated twice
ASAS ----- (All Sweep, Send Receive) repeated twice

LEQ_PC(Equation ID Number)

CHARACTER

Linear precondition used by the BiCGSTAB and CG LEQ solvers. 'LINE' is the default for all equation types.

NONE ----- No preconditioner

LINE ----- Line relaxation

DIAG ----- Diagonal Scaling

UR_FAC(Equation ID Number)

DOUBLE PRECISION

Under relaxation factors. 0.8 for equation types 1,6,9, 0.5 for equation types 2,3,4,5,8, 1.0 for equation types 7

UR_F_GS

DOUBLE PRECISION

The implicitness calculation of the gas-solids drag coefficient may be underrelaxed by changing *ur_f_gs*, which takes values between 0 to 1:

- 0 updates *F_GS* every time step
- 1 updates *F_GS* every iteration

UR_KTH_SML

DOUBLE PRECISION

Under relaxation factor for conductivity coefficient associated with other solids phases for IA Theory [1.0].

DISCRETIZE

INTEGER

Discretization scheme of equations.

[0] ----- First-order upwinding.

1 ----- First-order upwinding (using down-wind factors).

3 ----- Smart.

2 ----- Superbee (recommended method).

5 ----- QUICKEST (does not work).

4 ----- ULTRA-QUICK.

7 ----- van Leer.

6 ----- MUSCL.

8 ----- minmod.

DEF_COR

LOGICAL

Use deferred correction method for implementing higher order discretization.

[.FALSE.] ----- Use down-wind factor method (default).

.TRUE. ----- Use deferred correction method.

CHI_SCHEME

LOGICAL

This scheme guarantees that the set of differenced species mass balance equations maintain the property that the sum of species mass fractions sum to one. This property is not guaranteed when a flux limiter is used with higher order spatial discretization schemes. Note: The chi-scheme is implemented for SMART and MUSCL discretization schemes.

Darwish, M.S., Moukalled, F. (2003). International Journal of Heat and Mass Transfer, 46(4):599-611.

[.FALSE.] ----- Do not use the chi-scheme.

.TRUE. ----- Use the chi-scheme correction.

PPFOI

LOGICAL

Four point fourth order interpolation and is upstream biased. Notes:

- DISCRETIZE(*) defaults to Superbee if this scheme is chosen and DISCRETIZE(*) < 2.
- Set C_FAC between 0 and 1 when using this scheme.

C_FAC

DOUBLE PRECISION

Factor between zero and one used in the universal limiter when using four point, fourth order interpolation (PPFOI).

- Choosing one gives (diffusive) first order upwinding.
- The scheme becomes more compressive as values near zero.

CN_ON

LOGICAL

Temporal discretization scheme.

[.FALSE.] ----- Implicit Euler based temporal discretization scheme employed (first order accurate in time).
 .TRUE. ----- Two-step implicit Runge-Kutta method based temporal discretization scheme employed (second order accurate in time excluding the pressure terms and restart time step which are first order accurate).

MAX_INLET_VEL_FAC

DOUBLE PRECISION

The code declares divergence if the velocity anywhere in the domain exceeds a maximum value. This maximum value is automatically determined from the boundary values. The user may scale the maximum value by adjusting this scale factor [1.0d0].

8.4 Geometry And Discretization

For 2D simulations, the thickness of the third direction specified should be exact if mass or volumetric flow rates, rather than velocities, are specified at the boundaries.

8.4.1 Basic Geometry

This is some lead in text for the basic geometry section.

COORDINATES

CHARACTER

Coordinates used in the simulation.

CARTESIAN ----- Cartesian coordinates.
 CYLINDRICAL ----- Cylindrical coordinates.

IMAX

INTEGER

Number of cells in the x (r) direction.

DX(Cell)

DOUBLE PRECISION

Cell sizes in the x (r) direction. Enter values from DX(0) to DX(IMAX-1). (Use uniform mesh size with higher-order discretization methods. Also in cylindrical coordinates DX should be kept uniform for strict momentum conservation.)

XMIN

DOUBLE PRECISION

The inner radius in the simulation of an annular cylindrical region.

XLENGTH

DOUBLE PRECISION

Reactor length in the x (r) direction.

JMAX

INTEGER

Number of cells in the y-direction.

DY(Cell)

DOUBLE PRECISION

Cell sizes in the y-direction. Enter values from DY(0) to DY(IMAX-1). (Use uniform mesh size with second-order discretization methods.)

YLENGTH

DOUBLE PRECISION

Reactor length in the y-direction.

NO_K

LOGICAL

Flag to disable the third dimension (i.e., 2D simulation).

- Z axis in Cartesian coordinate system
- Theta in Cylindrical coordinate system

[.FALSE.] ----- 3D simulation.

.TRUE. ----- 2D simulation.

KMAX

INTEGER

Number of cells in the z-direction.

DZ(Cell)

DOUBLE PRECISION

Cell sizes in the z (theta) direction. Enter values from DZ(0) to DZ(IMAX-1). (Use uniform mesh size with second-order discretization methods.)

ZLENGTH

DOUBLE PRECISION

Reactor length in the z (theta) direction.

CYCLIC_X

LOGICAL

Flag for making the x-direction cyclic without pressure drop. No other boundary conditions for the x-direction should be specified.

[.FALSE.] ----- no cyclic condition at x-boundary.

.TRUE. ----- cyclic condition at x-boundary.

CYCLIC_X_PD

LOGICAL

Flag for making the x-direction cyclic with pressure drop. if the keyword FLUX_G is given a value this becomes a cyclic boundary condition with specified mass flux. no other boundary conditions for the x-direction should be specified.

[.FALSE.] ----- no cyclic condition at x-boundary.

.TRUE. ----- cyclic condition with pressure drop at x-boundary.

DELP_X

DOUBLE PRECISION

Fluid pressure drop across XLENGTH when a cyclic boundary condition with pressure drop is imposed in the x-direction.

CYCLIC_Y

LOGICAL

Flag for making the y-direction cyclic without pressure drop. No other boundary conditions for the y-direction should be specified.

[.FALSE.] ----- No cyclic condition at y-boundary.

.TRUE. ----- Cyclic condition at x-boundary.

CYCLIC_Y_PD

LOGICAL

Flag for making the y-direction cyclic with pressure drop. If the keyword FLUX_G is given a value this becomes a cyclic boundary condition with specified mass flux. No other boundary conditions for the y-direction should be specified.

[.FALSE.] ----- No cyclic condition at y-boundary.

.TRUE. ----- Cyclic condition with pressure drop at y-boundary.

DELP_Y

DOUBLE PRECISION

Fluid pressure drop across YLENGTH when a cyclic boundary condition with pressure drop is imposed in the y-direction.

CYCLIC_Z

LOGICAL

Flag for making the z-direction cyclic without pressure drop. No other boundary conditions for the z-direction should be specified.

[.FALSE.] ----- no cyclic condition at z-boundary.

.TRUE. ----- cyclic condition at z-boundary.

CYCLIC_Z_PD

LOGICAL

Flag for making the z-direction cyclic with pressure drop. If the keyword FLUX_G is given a value this becomes a cyclic boundary condition with specified mass flux. No other boundary conditions for the z-direction should be specified.

[.FALSE.] ----- No cyclic condition at z-boundary.

.TRUE. ----- Cyclic condition with pressure drop at z-boundary.

DELP_Z

DOUBLE PRECISION

Fluid pressure drop across ZLENGTH when a cyclic boundary condition with pressure drop is imposed in the z-direction.

SHEAR

LOGICAL

Imposes a mean shear on the flow field as a linear function of the x coordinate. This feature should only be used when CYCLIC_X is .TRUE. and the keyword V_SH is set.

V_SH

DOUBLE PRECISION

Specifies the mean y velocity component at the eastern boundary of the domain (V_SH), and the mean Y velocity (-V_SH) at the western boundary of the domain.

FLUX_G

DOUBLE PRECISION

If a value is specified (in units of g/cm².s), the domain-averaged gas flux is held constant at that value in simulations over a periodic domain. A pair of boundaries specified as periodic with fixed pressure drop is then treated as periodic with fixed mass flux. Even for this case a pressure drop must also be specified, which is used as the initial guess in the simulations.

CPX

DOUBLE PRECISION

Location of control points in x-direction.

NCX

INTEGER

Number of cells within a segment (x-direction).

ERX

DOUBLE PRECISION

Expansion ratio (last dx/first dx) in a segment (x-direction).

FIRST_DX

DOUBLE PRECISION

Value of first dx in a segment (x-direction). a negative / value will copy dx from previous segment (if available).

LAST_DX

DOUBLE PRECISION

Value of last dx in a segment (x-direction). a negative value will copy dx from next segment (if available).

CPY

DOUBLE PRECISION

Location of control points in y-direction.

NCY

INTEGER

Number of cells within a segment (y-direction).

ERY

DOUBLE PRECISION

Expansion ratio (last dy/first dy) in a segment (y-direction).

FIRST_DY

DOUBLE PRECISION

Value of first dy in a segment (y-direction). a negative value will copy dy from previous segment (if available).

LAST_DY

DOUBLE PRECISION

Value of last dy in a segment (y-direction). a negative value will copy dy from next segment (if available).

CPZ

DOUBLE PRECISION

Location of control points in z-direction.

NCZ

INTEGER

Number of cells within a segment (z-direction).

ERZ

DOUBLE PRECISION

Expansion ratio (last dz/first dz) in a segment (z-direction).

FIRST_DZ

DOUBLE PRECISION

Value of first dz in a segment (z-direction). a negative value will copy dz from previous segment (if available).

LAST_DZ

DOUBLE PRECISION

Value of last dz in a segment (z-direction). a negative value will copy dz from next segment (if available).

8.4.1 Cartesian Grid

The Cartesian grid cut-cell technique has been implemented in MFIX, which allows the definition of curved or sloping boundaries, instead of the usual stair-step representation. Computational cells are truncated at the wall to conform to the shape of the boundaries. When a face is truncated, the velocity node is moved to the center of the face. The cell truncation introduces an additional face, called the cut face. Face surface areas and cell volumes are updated based on the shape of the cut cell. The contribution of the new cut face is added to the computation. The data can be saved in a vtk file for post-processing purpose.

A detailed user guide named Cartesian_grid_user_guide.pdf is located in mfix/doc directory. It should be read prior to utilizing the Cartesian grid option to get familiar with this technique, and associated keywords. This file is also available online at https://mfix.netl.doe.gov/documentation/Cartesian_grid_user_guide.pdf.

CARTESIAN_GRID

LOGICAL

Activate cartesian grid cut cell technique.

[.FALSE.] -----do not use cartesian grid cut cell technique.

.TRUE. -----use cartesian grid cut cell technique. one of the following methods must be used to define the geometry:

N_QUADRIC

INTEGER

Number of quadric surfaces defining the boundaries (<=100).

USE_STL

LOGICAL

Use stl file to describe geometry.

[.FALSE.] -----do not use stl file.

.TRUE. -----read triangulated geometry (for 3d geometry only) from geometry.stl.

USE_MSH

LOGICAL

Use .msh file to describe geometry.

[.FALSE.] -----do not use .msh file.

.TRUE. -----read geometry (for 3d geometry only) from geometry.msh.

USE_POLYGON

LOGICAL

Use polygons to describe geometry.

[.FALSE.] -----do not use polygons.

.TRUE. -----read polygon data (for 2d geometry only) from poly.dat.

N_USR_DEF

INTEGER

Number of user-defined functions (currently limited to 0 or 1). if set to 1, the geometry is defined in the user subroutine eval_usr_fct.f.

[0] -----Do not use user-defined function

1 -----Use one user-defined function

QUADRIC_FORM(Quadric ID)

CHARACTER

Form of the quadric surface equation.

NORMAL -----use normal form, as defined in equation (1). The lambda

PLANE -----Plane. Needs to define n_x,n_y,n_z (unit normal vector pointing away from fluid cells).

X_CYL_INT -----Cylinder aligned with x-axis, internal flow. Needs to define radius(QID).

X_CYL_EXT -----Cylinder aligned with x-axis, external flow. Needs to define radius(QID).

Y_CYL_INT -----Cylinder aligned with y-axis, internal flow. Needs to define radius(QID).

Y_CYL_EXT -----Cylinder aligned with y-axis, external flow. Needs to define radius(QID).

Z_CYL_INT -----Cylinder aligned with z-axis, internal flow. Needs to define radius(QID).

Z_CYL_EXT -----Cylinder aligned with z-axis, external flow. Needs to define radius(QID).

X_CONE -----Cone aligned with x-axis, internal flow. Needs to define half_angle(QID).

Y_CONE -----Cone aligned with y-axis, internal flow. Needs to define half_angle(QID).

Z_CONE -----Cone aligned with z-axis, internal flow. Needs to define half_angle(QID).

SPHERE_INT -----Sphere, internal flow. Needs to define radius(QID).

SPHERE_EXT -----Sphere, external flow. Needs to define radius(QID).

C2C -----Cylinder-to-cylinder conical junction, internal flow. Needs to be defined between two cylinders.

TORUS_INT -----Torus, internal flow. Needs to define Torus_R1(QID) and Torus_R2(QID).A torus is not a quadric surface but is defined as a basic shape.

TORUS_EXT -----Torus, external flow. Needs to define Torus_R1(QID) and Torus_R2(QID).

QUADRIC_SCALE

DOUBLE PRECISION

Scaling factor, applied to all quadric geometry parameters. Must be a positive number

LAMBDA_X(Quadric ID)

DOUBLE PRECISION

Coefficient lambda_x in equation (1) ('normal' form) or x-component of normal vector defining plane in equation (5) ('degenerate' form).

LAMBDA_Y(Quadric ID)

DOUBLE PRECISION

Coefficient lambda_y in equation (1) ('normal' form) or y-component of normal vector defining plane in equation (5) ('degenerate' form).

LAMBDA_Z(Quadric ID)	DOUBLE PRECISION
Coefficient lambda_z in equation (1) ('normal' form) or z-component of normal vector defining plane in equation (5) ('degenerate' form).	
DQUADRIC(Quadric ID)	DOUBLE PRECISION
Coefficient d in equation (1).	
THETA_X(Quadric ID)	DOUBLE PRECISION
Rotation angle with respect to x-axis (degrees).	
THETA_Y(Quadric ID)	DOUBLE PRECISION
Rotation angle with respect to y-axis (degrees).	
THETA_Z(Quadric ID)	DOUBLE PRECISION
Rotation angle with respect to z-axis (degrees).	
RADIUS(Quadric ID)	DOUBLE PRECISION
Cylinder radius (used when quadric_form = *_cyl_***)	
HALF_ANGLE(Quadric ID)	DOUBLE PRECISION
Cone half angle, expressed in degrees (used when quadric_form = *_cone)	
TORUS_R1(Quadric ID)	DOUBLE PRECISION
Torus Radius 1 (used when quadric_form = Torus_*), R1>R2 for a ring.	
TORUS_R2(Quadric ID)	DOUBLE PRECISION
Torus Radius 2 (used when quadric_form = Torus_*), R1>R2 for a ring.	
N_X(Quadric ID)	DOUBLE PRECISION
X-component of normal vector defining the plane (used when quadric_form = plane)	
N_Y(Quadric ID)	DOUBLE PRECISION
Y-component of normal vector defining the plane (used when quadric_form = plane)	
N_Z(Quadric ID)	DOUBLE PRECISION
Z-component of normal vector defining the plane (used when quadric_form = plane)	
T_X(Quadric ID)	DOUBLE PRECISION
Translation in x-direction.	
T_Y(Quadric ID)	DOUBLE PRECISION
Translation in y-direction.	
T_Z(Quadric ID)	DOUBLE PRECISION
Translation in z-direction.	

CLIP_XMIN(Quadric ID) Lower x-limit where the quadric is defined.	DOUBLE PRECISION
CLIP_XMAX(Quadric ID) Upper x-limit where the quadric is defined.	DOUBLE PRECISION
CLIP_YMIN(Quadric ID) Lower y-limit where the quadric is defined.	DOUBLE PRECISION
CLIP_YMAX(Quadric ID) Upper y-limit where the quadric is defined.	DOUBLE PRECISION
CLIP_ZMIN(Quadric ID) Lower z-limit where the quadric is defined.	DOUBLE PRECISION
CLIP_ZMAX(Quadric ID) Upper z-limit where the quadric is defined.	DOUBLE PRECISION
PIECE_XMIN(Quadric ID) Lower x-limit where the quadric is defined in a piecewise group.	DOUBLE PRECISION
PIECE_XMAX(Quadric ID) Upper z-limit where the quadric is defined in a piecewise group.	DOUBLE PRECISION
PIECE_YMIN(Quadric ID) Lower y-limit where the quadric is defined in a piecewise group.	DOUBLE PRECISION
PIECE_YMAX(Quadric ID) Upper y-limit where the quadric is defined in a piecewise group.	DOUBLE PRECISION
PIECE_ZMIN(Quadric ID) Lower z-limit where the quadric is defined in a piecewise group.	DOUBLE PRECISION
PIECE_ZMAX(Quadric ID) Upper z-limit where the quadric is defined in a piecewise group.	DOUBLE PRECISION
FLUID_IN_CLIPPED_REGION(Quadric ID) Flag defining the type of cells that are outside of the zone defined by [clip_xmin;clip_xmax], [clip_ymin;clip_ymax],[clip_zmin;clip_zmax]. .FALSE. -----remove cells from computational domain. [.TRUE.] -----treat cells as fluid cells.	LOGICAL
BC_ID_Q(Quadric ID) Boundary condition flag	INTEGER

N_GROUP INTEGER
Number of group(s) of quadrics (≤ 50).

GROUP_SIZE INTEGER
Number of quadrics in the group.

GROUP_Q INTEGER
Quadric ID assigned to a group

GROUP_RELATION CHARACTER
Relation among quadrics of a same group.
OR ----- a point belongs to the computational domain if at least one of $f(x,y,z)$ among all quadrics is negative
AND ----- a point belongs to the computational domain if all of $f(x,y,z)$ among all quadrics are negative

RELATION_WITH_PREVIOUS CHARACTER
Relation between current group and combination of all previous groups.
OR ----- a point belongs to the computational domain if f-value for the current group or f-value for the combination of previous groups is negative
AND ----- a point belongs to the computational domain if f-value for the current group and f-value for the combination of previous groups is negative

TOL_SNAP DOUBLE PRECISION
Tolerance used to snap an intersection point onto an existing cell corner (expressed as a fraction of edge length, between 0.0 and 0.5). For stretched grids, three values can be entered in the x, y and z directions.

TOL_DELH DOUBLE PRECISION
Tolerance used to limit acceptable values of normal distance to the wall (expressed as a fraction of cell diagonal, between 0.0 and 1.0).

TOL_SMALL_CELL DOUBLE PRECISION
Tolerance used to detect small cells (expressed as a fraction of cell volume, between 0.0 and 1.0).

TOL_MERGE CHARACTER
Tolerance used to remove duplicate nodes (expressed as a fraction of cell diagonal, between 0.0 and 1.0).

TOL_SMALL_AREA DOUBLE PRECISION
Tolerance used to detect small faces (expressed as a fraction of original face area, between 0.0 and 1.0).

ALPHA_MAX DOUBLE PRECISION
Maximum acceptable value of interpolation correction factor.

TOL_F DOUBLE PRECISION
Tolerance used to find intersection of quadric surfaces or user-defined function with background grid.

TOL_POLY

DOUBLE PRECISION

Tolerance used to find intersection of polygon with background grid.

ITERMAX_INT

INTEGER

Maximum number of iterations used to find intersection points.

TOL_STL

DOUBLE PRECISION

Tolerance used to find intersection of stl triangles with background grid.

STL_SMALL_ANGLE

DOUBLE PRECISION

Smallest angle accepted for valid stl triangles (in degrees). triangles having an angle smaller than this value will be ignored.

TOL_STL_DP

DOUBLE PRECISION

Dot product tolerance when determining if a point lies in a facet.

DIM_FACETS_PER_CELL

INTEGER

Maximum number of STL facets per cell.

MAX_FACETS_PER_CELL_DES

INTEGER

Maximum number of STL facets per cell for des data arrays.

OUT_STL_VALUE

DOUBLE PRECISION

Defines value of f_{stl} outside of the stl geometry. a value of 1.0 means the domain outside of the stl geometry is excluded from computation, i.e., an internal flow is computed.

-1.0 ----- model an external flow

[1.0] ----- model an internal flow

STL_BC_ID

INTEGER

Boundary condition flag for the stl geometry

TX_STL

DOUBLE PRECISION

Translation in x-direction, applied to the stl geometry.

TY_STL

DOUBLE PRECISION

Translation in y-direction, applied to the stl geometry.

TZ_STL

DOUBLE PRECISION

Translation in z-direction, applied to the stl geometry.

SCALE_STL

DOUBLE PRECISION

Scaling factor, applied to the stl geometry. note that translation occurs after scaling.

TOL_MSH

DOUBLE PRECISION

Tolerance used to find intersection of .msh file with background grid.

OUT_MSH_VALUE

DOUBLE PRECISION

Defines value of f outside of the .msh geometry. a value of 1.0 means the domain outside of the .msh geometry is excluded from computation, i.e., an internal flow is computed.

-1.0 ----- model an external flow
[1.0] ----- model an internal flow

TX_MSH

DOUBLE PRECISION

Translation in x-direction, applied to the .msh geometry.

TY_MSH

DOUBLE PRECISION

Translation in y-direction, applied to the .msh geometry.

TZ_MSH

DOUBLE PRECISION

Translation in z-direction, applied to the .msh geometry.

SCALE_MSH

DOUBLE PRECISION

Scaling factor, applied to the .msh geometry. note that translation occurs after scaling.

CAD_PROPAGATE_ORDER

CHARACTER

Ray propagation order used to determine whether any point is located inside or outside of the stl surface. a value of ijk means the propagation occurs in the i , followed by j , and k directions. other available orders are jki and kij

ijk ----- propagation occurs in the i , followed by j , and k directions
 jki ----- propagation occurs in the j , followed by k , and i directions
 kij ----- propagation occurs in the k , followed by i , and j directions

RAY_DIR

CHARACTER

Ray direction when propagating CAD value

SET_CORNER_CELLS

LOGICAL

Flag to detect and treat corner cells the same way as in the original mfix version (i.e. without cut cells). if set to .true., some cut cells may be treated as corner cells.

.TRUE. ----- some cut cells may be treated as corner cells.
[.FALSE.] ----- Na

FAC_DIM_MAX_CUT_CELL

DOUBLE PRECISION

Factor used to allocate some cut cell arrays (expressed as a fraction of dimension_3g)

PG_OPTION

INTEGER

Option for pressure gradient computation in cut cells.

1 ----- use maximum of (east/west), (north/south), and (top/bottom) pairs of velocity cells.
2 ----- use both (east/west), (north/south), and (top/bottom) areas of velocity cells.
[0] ----- use east, north and top areas of pressure cell (same as standard cells).

CG_SAFE_MODE

INTEGER

Run code in safe mode.

- 1 ----- performs initial preprocessing but use all original mfix subroutines during flow solution (using only cell volumes and areas of cut cells).
- [0] ----- runs the code with modified subroutines for cut cell treatment.

PRINT_WARNINGS

LOGICAL

Prints any warning message encountered during pre-processing on the screen.

CG_UR_FAC

DOUBLE PRECISION

Under-relaxation factor used in cut cells (only cg_ur_fac(2) is used).

PRINT_PROGRESS_BAR

LOGICAL

Print a progress bar during each major step of pre-processing stage.

BAR_WIDTH

INTEGER

Width of the progress bar (complete status), expressed in number of characters (between 10 and 80).

BAR_CHAR

CHARACTER

Character used to create the progress bar.

BAR_RESOLUTION

DOUBLE PRECISION

Update frequency of progress bar, expressed in percent of total length (between 1.0 and 100.0).

WRITE_DASHBOARD

LOGICAL

Writes the file dashboard.txt at regular intervals. the file shows a summary of the simulation progress.

F_DASHBOARD

INTEGER

Frequency, expressed in terms of iterations, at which the dashboard is updated.

RE_INDEXING

LOGICAL

Turns on the re-indexing of cells. When true, inactive (dead) cells are removed from computational domain.

ADJUST_PROC_DOMAIN_SIZE

LOGICAL

Attempts to adjust grid partition. Each processor will be assigned its own size to minimize load imbalance

REPORT_BEST_DOMAIN_SIZE

LOGICAL

Attempts to adjust grid partition. Each processor will be assigned its own size to minimize load imbalance

NODESI_REPORT

INTEGER

Temporary setting used in serial run to report best domain size for parallel run

NODESJ_REPORT

INTEGER

Temporary setting used in serial run to report best domain size for parallel run

NODESK_REPORT

INTEGER

Temporary setting used in serial run to report best domain size for parallel run

MINIMIZE_SEND_RECV

LOGICAL

Attempts to minimize the size of the send/receive layers

USE_STL_DES

LOGICAL

Flag to use triangular facet representation for particle/parcel- wall interactions. If PIC model is used, then this flag is forced to true. For DEM model, it will be made default, but currently setting it to false as cohesion model has not been extended to new routines. Expand the size of the particle arrays by an arbitrary factor (multiple of the number of particles). Serves as a knob to allocate more particles than initially specified in the particle configuration file.

8.5 Gas Phase

This section lists keywords for specifying the gas phase.

RO_G0

DOUBLE PRECISION

Specified constant gas density [g/cm³ in CGS]. An equation of state -the ideal gas law by default- is used to calculate the gas density if this parameter is undefined. The value may be set to zero to make the drag zero and to simulate granular flow in a vacuum. For this case, users may turn off solving for gas momentum equations to accelerate convergence.

MU_G0

DOUBLE PRECISION

Specified constant gas viscosity [poise or g/(cm.s) in CGS].

K_G0

DOUBLE PRECISION

Specified constant gas conductivity [cal/(s.cm.K) in CGS].

C_PG0

DOUBLE PRECISION

Specified constant gas specific heat [cal/(g.s.K) in CGS].

DIF_G0

DOUBLE PRECISION

Specified constant gas diffusivity [g/(cm.s) in CGS].

MW_AVG

DOUBLE PRECISION

Average molecular weight of gas. Used in calculating the gas density for non-reacting flows when the gas composition is not defined.

MW_G(Species)

DOUBLE PRECISION

Molecular weight of gas species.

NMAX_G

INTEGER

Number of species comprising the gas phase.

SPECIES_G

CHARACTER

Name of gas phase species as it appears in the materials database.

SPECIES_ALIAS_G

CHARACTER

User defined name for gas phase species. Aliases are used in specifying chemical equations and must be unique.

8.6 Solids Phase

This section contains keywords for specifying solids phases. Keywords under this section are applicable to all solids models (TFM, DEM, and PIC) unless otherwise stated. Keywords specific to one or more solids models are indicated by the subheadings.

SOLIDS_MODEL(Phase)

CHARACTER

Defines the model used for the solids phase. For TFM/DEM hybrid simulations, first define all TFM solids, then define the DEM solids.

TFM ----- Two-fluid Model (continuum).
DEM ----- Discrete Element Model
PIC ----- Multiphase-Particle in Cell

MMAX

INTEGER

Number of solids phases.

Applies to solids models: TFM, DEM, PIC

D_P0(Phase)

DOUBLE PRECISION

Initial particle diameters [cm in CGS].

Applies to solids models: TFM, DEM, PIC

RO_S0(Phase)

DOUBLE PRECISION

Specified constant solids density [g/cm³ in CGS]. Reacting flows may use variable solids density by leaving this parameter undefined and specifying X_S0 and RO_XS0 as well as the index of the inert species.

Applies to solids models: TFM, DEM, PIC

X_S0(Phase, Species)

DOUBLE PRECISION

Baseline species mass fraction. Specifically, the mass fraction of an unreacted sample (e.g., proximate analysis).

Applies to solids models: TFM, DEM

RO_XS0(Phase, Species)

DOUBLE PRECISION

Specified constant solids species density [g/cm³ in CGS].

Applies to solids models: TFM, DEM

INERT_SPECIES(Phase, Species)

INTEGER

Index of inert solids phase species. This species should not be a product or reactant of any chemical reaction.

Applies to solids models: TFM, DEM

K_SO(Phase)

DOUBLE PRECISION

Specified constant solids conductivity [cal/(s.cm.K) in CGS].

Applies to solids models: TFM, DEM

C_PSO(Phase)

DOUBLE PRECISION

Specified constant solids specific heat [cal/(g.s.K) in CGS].

Applies to solids models: TFM, DEM

MW_S(Phase, Species)

DOUBLE PRECISION

Molecular weight of solids phase species.

Applies to solids models: TFM, DEM

NMAX_S(Phase)

INTEGER

Number of species comprising the solids phase.

Applies to solids models: TFM, DEM

SPECIES_S(Phase, Species)

CHARACTER

Name of solids phase M, species N as it appears in the materials database.

Applies to solids models: TFM, DEM

SPECIES_ALIAS_S(Phase, Species)

CHARACTER

User defined name for solids phase species. Aliases are used in specifying chemical equations and must be unique.

Applies to solids models: TFM, DEM

8.6.1 Two Fluid Model (TFM)

Keywords specific to the two fluid model, MFIX-TFM, are defined in this section.

KT_TYPE

CHARACTER

Solids phase stress model. This is only needed when solving the granular energy PDE (GRANULAR_ENERGY = .TRUE.).

AHMADI ----- Cao and Ahmadi (1995). Int. J. Multiphase Flow 21(6), 1203.

GD_99 ----- Garzo and Dufty (1999). Phys. Rev. E 59(5), 5895.

GHD ----- Garzo, Hrenya and Dufty (2007). Phys. Rev. E 76(3), 31304

IA_NONEP ----- Iddir & Arastoopour (2005). AIChE J. 51(6), 1620

LUN_1984 ----- Lun et al (1984). J. Fluid Mech., 140, 223.

SIMONIN ----- Simonin (1996). VKI Lecture Series, 1996-2

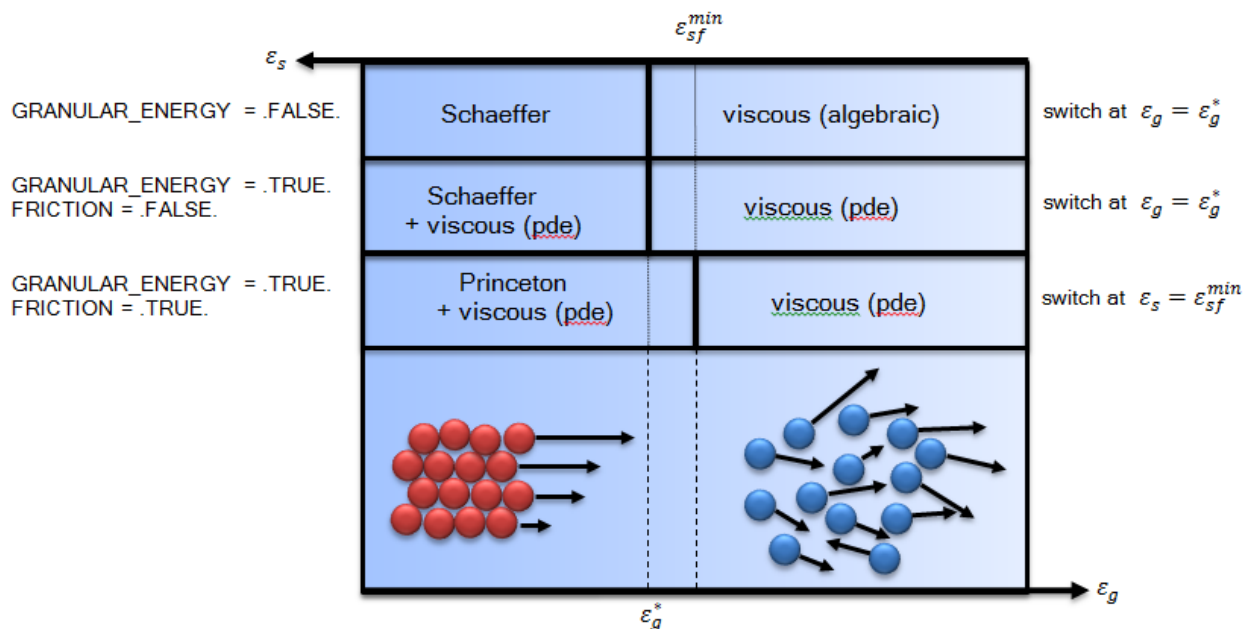
JENKINS

LOGICAL

This flag effects how the momentum and granular energy boundary conditions are implemented. This feature requires that PHI_W is specified.

[.FALSE.] ----- Use standard boundary conditions.
 .TRUE. ----- Use the Jenkins small frictional boundary condition.

The combination of the keywords GRANULAR_ENERGY and FRICTION invokes different solids stress modes as shown below. Note the dependencies on the other keywords EP_STAR, ε_g^* , and EPS_F_MIN, ε_{sf}^{min} .

**FRICTION**

LOGICAL

Solids stress model selection. Invoking fraction requires GRANULAR_ENERGY = .TRUE.

[.FALSE.] ----- Do not use the Princeton solids stress model.
 .TRUE. ----- Use the Princeton solids stress model

SAVAGE

INTEGER

For a term appearing in the frictional stress model invoked with friction = .TRUE.

0 ----- Use S:S in the frictional stress model.
 [1] ----- Use an alternate form suggested by Savage.
 2 ----- An appropriate combination of above.

SCHAEFFER

LOGICAL

Schaeffer frictional stress tensor formulation. Requires Phi be defined.

[.TRUE.] ----- Use the Schaeffer model
 .FALSE. ----- Do not use the Schaeffer model.

BLENDING_STRESS

LOGICAL

Blend the Schaeffer stresses with that of algebraic kinetic theory around EP_STAR.

Applies to solids models: TFM

TANH_BLEND

LOGICAL

Hyperbolic tangent function for blending frictional stress models.

SIGM_BLEND

LOGICAL

A scaled and truncated sigmoidal function for blending frictional stress models.

Applies to solids models: TFM

YU_STANDISH

LOGICAL

Correlation to compute maximum packing for polydisperse systems.

Applies to solids models: TFM

.TRUE. ----- Use the Yu and Standish correlation.

[.FALSE.] ----- Do not use the Yu and Standish correlation.

FEDORS_LANDEL

LOGICAL

Use Fedors and Landel correlation to compute maximum packing for a binary (only) mixture of powders.

Applies to solids models: TFM

RDF_TYPE

CHARACTER

Radial distribution function at contact for polydisperse systems. Do not specify any RDF for monodisperse systems because Carnahan- Starling is the model only available.

Carnahan, N.F. and Starling K.E., (1969). The Journal of Chemical Physics, Vol. 51(2):635-636.

LEBOWITZ ----- Lebowitz, J.L. (1964) The Physical Review, A133, 895-899

MODIFIED_LEBOWITZ ----- Iddir, H. Y., Modeling of the multiphase mixture of particles using the kinetic theory approach. Doctoral Dissertation, Illinois Institute of Technology, Chicago, Illinois, 2004, (chapter 2, equations 2-49 through 2-52.)

MANSOORI ----- Mansoori, GA, Carnahan N.F., Starling, K.E. Leland, T.W. (1971). The Journal of Chemical Physics, Vol. 54:1523-1525.

MODIFIED_MANSOORI ----- van Wachem, B.G.M., Schouten, J.C., van den Bleek, C.M., Krishna, R. and Sinclair, J. L. (2001) AIChE Journal 47:1035-1051.

ADDED_MASS

LOGICAL

Flag to include the added (or virtual) mass force. This force acts to increase the inertia of the dispersed phase, which tends to stabilize simulations of bubbly gas-liquid flows.

Applies to solids models: TFM

M_AM

INTEGER

The disperse phase number where the added mass applies.

Applies to solids models: TFM

C_E DOUBLE PRECISION
 Coefficient of restitution for particle-particle collisions.
 Applies to solids models: TFM

R_P(Phase, Phase) DOUBLE PRECISION
 GHD Theory: Coefficient of restitution for particle-particle collisions.

E_W DOUBLE PRECISION
 Coefficient of restitution for particle-wall collisions [1.0].

PHIP DOUBLE PRECISION
 Specularity coefficient associated with particle-wall collisions [0.6].
 Applies to solids models: TFM

PHIPO DOUBLE PRECISION
 Specify the value of specularity coefficient when the normalized slip velocity goes to zero when BC_JJ_M is .TRUE.. This variable is calculated internally in the code. Do not modify unless an accurate number is known.
 Applies to solids models: TFM

C_F DOUBLE PRECISION
 Coefficient of friction between the particles of two solids phases.
 Applies to solids models: TFM

PHI DOUBLE PRECISION
 Angle of internal friction (in degrees) [0.0]. Set this value to zero to turn off plastic regime stress calculations.
 Applies to solids models: TFM

PHI_W DOUBLE PRECISION
 Angle of internal friction (in degrees) at walls. Set this value to non-zero (PHI_W = 11.31 means TAN_PHI_W = MU = 0.2) when using Jenkins or BC_JJ_M boundary condition.
 Applies to solids models: TFM

EPS_F_MIN DOUBLE PRECISION
 Minimum solids fraction above which friction sets in when friction = .TRUE.. [0.5].
 Applies to solids models: TFM

EP_S_MAX(Phase) DOUBLE PRECISION
 Maximum solids volume fraction at packing for polydisperse systems (more than one solids phase used) [1.0- EP_STAR]. The value of EP_STAR may change during the computation if solids phases with different particle diameters are specified and Yu_Standish or Fedors_Landel correlations are used.
 Applies to solids models: TFM

SEGREGATION_SLOPE_COEFFICIENT

DOUBLE PRECISION

Used in calculating the initial slope of segregation. [0.0].

See Gera et al. (2004) - recommended value 0.3. increasing this coefficient results in decrease in segregation of particles in binary mixtures.

Applies to solids models: TFM

V_EX

DOUBLE PRECISION

Excluded volume in Boyle-Massoudi stress.

Applies to solids models: TFM

0.0 ----- b-m stress is turned off.

MU_S0

DOUBLE PRECISION

Specified constant granular viscosity. If this value is specified, then the kinetic theory calculation is turned off and $P_S = 0$ and $LAMBDA_S = -2/3 MU_{S0}$.

Applies to solids models: TFM

DIF_S0

DOUBLE PRECISION

Specified constant solids diffusivity.

Applies to solids models: TFM

EP_STAR

DOUBLE PRECISION

Packed bed void fraction. Used to calculate plastic stresses (for contribution to viscosity) and when to implement plastic pressure, P_{STAR} . Specifically, if $EP_G < EP_{STAR}$, then plastic pressure is employed in the momentum equations.

Applies to solids models: TFM

CLOSE_PACKED(Phase)

LOGICAL

Flag to enable/disable a phase from forming a packed bed.

Applies to solids models: TFM

[.TRUE.] ----- The phase forms a packed bed with void fraction EP_{STAR} .

.FALSE. ----- The phase can exceed close pack conditions so that it may behave like a liquid.

8.6.2 Discrete Element Simulations

This subsection contains keywords available to MFIX discrete element simulations using either DEM or PIC solids unless stated otherwise.



The keywords **DISCRETE_ELEMENT** and **MPPIC** are no longer used. Each solids phase model is specified through the **SOLIDS_MODEL** keyword.

PARTICLES

INTEGER

Number of particles to be read in from the particle_input.dat file. This value is overwritten when using automatic particle generation. A simulation with a mass inflow BC can start without solids by setting PARTICLES = 0.

MAX_PIS

INTEGER

Maximum number of particles that may exist within a simulation. This quantity is used for calculating the size of arrays for allocation. Required input if PARTICLES is UNDEFINED.

GENER_PART_CONFIG

LOGICAL

Automatically generate the initial particle position and velocity data based on the parameters specified for each initial condition (IC) region.

.TRUE. ----- Generate particle configuration based on the initial condition parameters.
Data provided in the particle_input.dat file, if present, is ignored.
[.FALSE.] ----- Particle position and velocity data are provided in the particle_input.dat
file. A runtime error occurs if this file is not provided.

PARTICLES_FACTOR

DOUBLE PRECISION

Expand the size of the particle arrays by an arbitrary factor (multiple of the number of particles). Serves as a knob to allocate more particles than initially specified in the particle configuration file.

DES_CONTINUUM_COUPLED

LOGICAL

To switch between pure granular or coupled simulations of carried and dispersed phase flows.

.TRUE. ----- Performs coupled simulations.

DES_ONEWAY_COUPLED

LOGICAL

Run one-way coupled simulations. The fluid does not see the particles in terms of drag force. The effect of particle volume is still felt by the fluid through non-unity voidage values.

DES_INTERP_ON

LOGICAL

Use an interpolation suite to calculate the drag force on each particle based on particle location rather than cell averages.

DES_INTERP_MEAN_FIELDS

LOGICAL

Use interpolation to compute dispersed phase average fields, such as, solids volume fraction, solids velocity fields. If false, the average fields are obtained by simple arithmetic averaging.

.TRUE. ----- If drag is interpolated (i.e., DES_INTERP_ON = .TRUE.), then it is forced
to true for backward compatibility. Additionally, if MPPIC or Cut-cells are
used (DEM or MPPIC), then also the mean field interpolation is forced.

DES_INTG_METHOD

CHARACTER

Time stepping scheme.

Applies to solids models: DEM

EULER ----- First-Order Euler Scheme.

ADAMS BASHFORTH ----- Second order ADAMS BASHFORTH scheme (DEM only)

8.6.2.1 Discrete Element Model (DEM)

Keywords specific to the discrete element model, MFX-DEM, are provided in this section.

PARTICLES INTEGER

Number of particles to be read in from the particle_input.dat file. This value is overwritten when using automatic particle generation. A simulation with a mass inflow BC can start without solids by setting PARTICLES = 0.

MAX_PIS INTEGER

Maximum number of particles that may exist within a simulation. This quantity is used for calculating the size of arrays for allocation. Required input if PARTICLES is UNDEFINED.

GENER_PART_CONFIG LOGICAL

Automatically generate the initial particle position and velocity data based on the parameters specified for each initial condition (IC) region.

.TRUE. ----- Generate particle configuration based on the initial condition parameters. Data provided in the particle_input.dat file, if present, is ignored.
[.FALSE.] ----- Particle position and velocity data are provided in the particle_input.dat file. A runtime error occurs if this file is not provided.

PARTICLES_FACTOR DOUBLE PRECISION

Expand the size of the particle arrays by an arbitrary factor (multiple of the number of particles). Serves as a knob to allocate more particles than initially specified in the particle configuration file.

DES_CONTINUUM_COUPLED LOGICAL

To switch between pure granular or coupled simulations of carried and dispersed phase flows.

.TRUE. ----- Performs coupled simulations.

DES_ONEWAY_COUPLED LOGICAL

Run one-way coupled simulations. The fluid does not see the particles in terms of drag force. The effect of particle volume is still felt by the fluid through non-unity voidage values.

DES_INTERP_ON LOGICAL

Use an interpolation suite to calculate the drag force on each particle based on particle location rather than cell averages.

DES_INTERP_MEAN_FIELDS LOGICAL

Use interpolation to compute dispersed phase average fields, such as, solids volume fraction, solids velocity fields. If false, the average fields are obtained by simple arithmetic averaging.

.TRUE. ----- If drag is interpolated (i.e., DES_INTERP_ON = .TRUE.), then it is forced to true for backward compatibility. Additionally, if MPPIC or Cut-cells are used (DEM or MPPIC), then also the mean field interpolation is forced.

DES_INTG_METHOD CHARACTER

Time stepping scheme.

Applies to solids models: DEM

EULER ----- First-Order Euler Scheme.

ADAMS BASHFORTH ----- Second order ADAMS BASHFORTH scheme (DEM only)

8.6.2.2 Particle in Cell (PIC)

Keywords specific to the particle in cell model, MFIX-PIC, are provided in this section. The MFIX-PIC documentation file titled "Documentation of open-source MFIX–PIC software for gas-solids flows" by R. Garg and J. F. Dietiker. This file is located in the mfix/doc directory and is available online and serves as the theory guide for the MPPIC model implementation details in MFIX code.

https://mfix.netl.doe.gov/documentation/mfix_pic_doc.pdf.



Note that MPPIC model does not work with MPI modules. It has only been tested in serial mode and moderate testing with SMP.

The MPPIC model is invoked when one or more solids phases has the SOLIDS_MODEL set to PIC. There are two methods to specify the initial seeding of parcels. In both methods, the user first defines the physical region where the initial solids will be seeded. This is done by the same flags that are used in continuum representation of dispersed phase. In the first method, the user specifies the number of parcels per cell by the setting the flag “CONSTANTNPC” to TRUE and specifying number of parcels per cell for each phase by the array “NPC_PIC”. In this case, the user defined number of parcels per cell is randomly seeded in the initial physical region specified by user. The statistical weight is assigned to parcels such that the solid volume fraction implied by parcels equals the user defined solid volume fraction (see the MPPIC documentation).

In the second method, the statistical weight of parcels is fixed by setting the flag “CONSTANTWT” to TRUE along with specifying the statistical weight of parcels belonging to each phase by the array “STATWT_PIC”. The number of parcels per cell is computed by the code such that the solid volume fraction implied by parcels equals the user defined solid volume fraction.

The current implementation of MPPIC model also has the frictional stress model implementation outlined in Snider’s [1] paper. This is based on our best understanding of the model from the paper. This model does not simulate very stably.

The minimum gas voidage at maximum packing beyond which the MPPIC frictional stress model gets invoked is still defined by the flag EP_STAR that is generally used in continuum representation.

MPPIC_SOLID_STRESS_SNIDER

LOGICAL

Turn on snider’s version of frictional model. Does not run very stably.

MPPIC_COEFF_EN1

DOUBLE PRECISION

First coefficient of restitution for the frictional stress model in the MPPIC model. See the MPPIC documentation for more details.

MPPIC_COEFF_EN2

DOUBLE PRECISION

Second coefficient of restitution for the frictional stress model in the MPPIC model. See the MPPIC documentation for more details.

MPPIC_COEFF_EN_WALL

DOUBLE PRECISION

Normal coefficient of restitution for parcel-wall collisions in the MPPIC model.

MPPIC_COEFF_ET_WALL

DOUBLE PRECISION

Tangential coefficient of restitution for parcel-wall collisions in the MPPIC model. Currently not implemented in the code.

MPPIC_PDRAG_IMPLICIT

LOGICAL

Turn on the implicit treatment for interphase drag force. Valid only for MPPIC model..

MPPIC_GRAV_TREATMENT

LOGICAL

Variable to decide if special treatment is needed or not in the direction of gravity in the frictional stress tensor. See the MPPIC documentation for details.

PIC_REPORT_MIN_EPG

LOGICAL

A run time flag to report minimum value and location of gas voidage. This is useful only for debugging and is not recommended for production runs.

PSFAC_FRIC_PIC

INTEGER

P_s term in the frictional stress model of Snider.

FRIC_EXP_PIC

DOUBLE PRECISION

Beta term in the frictional stress model of Snider.

FRIC_NON_SING_FAC

CHARACTER

Non-singularity term (epsilon) in the frictional stress model of Snider.

CFL_PIC

DOUBLE PRECISION

CFL number used to decide maximum time step size for parcels evolution equations. Relevant to MPPIC model only.

8.7 Initial Conditions

Each initial condition (IC) is specified over a rectangular region (or pie-shaped for cylindrical coordinates) that corresponds to the scalar numerical grid. These are 3D regions: X_w X_e, Y_s Y_n, and Z_t Z_b. The region is defined by the constant coordinates of each of the six faces, which may be specified as the physical coordinates

or the cell indices. The physical coordinates are easier to specify than the cell indices. If cell sizes are not small enough to resolve a region specified using physical coordinates, MFIX will indicate this problem with an error message.

In cylindrical coordinates, when the theta direction crosses the 0 value, split that region into two regions: e.g., Split a region spanning 1.9 pi to 0.1 pi as 1.9 pi to 2 pi and 0 to 0.1 pi.

Two initial condition regions may overlap. When an overlap occurs, MFIX uses the conditions specified for the higher IC number.

IC_X_W(IC)	DOUBLE PRECISION
-------------------	------------------

X coordinate of the west face.

IC_X_E(IC)	DOUBLE PRECISION
-------------------	------------------

X coordinate of the east face.

IC_Y_S(IC)	DOUBLE PRECISION
-------------------	------------------

Y coordinate of the south face.

IC_Y_N(IC)	DOUBLE PRECISION
-------------------	------------------

Y coordinate of the north face.

IC_Z_B(IC)	DOUBLE PRECISION
-------------------	------------------

Z coordinate of the bottom face.

IC_Z_T(IC)	DOUBLE PRECISION
-------------------	------------------

Z coordinate of the top face.

IC_I_W(IC)	INTEGER
-------------------	---------

I index of the west-most wall.

IC_I_E(IC)	INTEGER
-------------------	---------

I index of the east-most wall.

IC_J_S(IC)	INTEGER
-------------------	---------

J index of the south-most wall.

IC_J_N(IC)	INTEGER
-------------------	---------

J index of the north-most wall.

IC_K_B(IC)	INTEGER
-------------------	---------

K index of the bottom-most wall.

IC_K_T(IC)	INTEGER
-------------------	---------

K index of the top-most wall.

IC_TYPE(IC) CHARACTER

Type of initial condition. Mainly used in restart runs to overwrite values read from the .RES file by specifying it as _PATCH_. The user needs to be careful when using the _PATCH_ option, since the values from the .RES file are overwritten and no error checking is done for the patched values.

IC_EP_G(IC) DOUBLE PRECISION

Initial void fraction in the IC region.

IC_P_G(IC) DOUBLE PRECISION

Initial gas pressure in the IC region. If this quantity is not specified, MFIX will set up a hydrostatic pressure profile, which varies only in the y-direction.

IC_P_STAR(IC) DOUBLE PRECISION

Initial solids pressure in the IC region. Usually, this value is specified as zero.

IC_L_SCALE(IC) DOUBLE PRECISION

Turbulence length scale in the IC region.

IC_ROP_S(IC, Phase) DOUBLE PRECISION

Initial bulk density ($rop_s = ro_s \times ep_s$) of solids phase-m in the IC region. Users need to specify this IC only for polydisperse flow ($M_{MAX} > 1$). Users must make sure that summation of $(IC_ROP_s(ic,m) / RO_s(m))$ over all solids phases is equal to $(1.0 - IC_EP_g(ic))$.

IC_EP_S(IC, Phase) DOUBLE PRECISION

Initial solids volume fraction of solids phase-m in the IC region. This may be specified in place of IC_ROP_s .

IC_T_G(IC) DOUBLE PRECISION

Initial gas phase temperature in the IC region.

IC_T_S(IC, Phase) DOUBLE PRECISION

Initial solids phase-m temperature in the IC region.

IC_THETA_M(IC, Phase) DOUBLE PRECISION

Initial solids phase-m granular temperature in the IC region.

IC_GAMA_RG(IC, Phase) DOUBLE PRECISION

Gas phase radiation coefficient in the IC region. Modify file `rdtn2.inc` to change the source term.

IC_T_RG(IC, Phase) DOUBLE PRECISION

Gas phase radiation temperature in the IC region.

IC_GAMA_RS(IC, Phase) DOUBLE PRECISION

Solids phase-m radiation coefficient in the IC region. Modify file `radtn2.inc` to change the source term.

IC_T_RS(IC, Phase)	DOUBLE PRECISION
Solids phase-m radiation temperature in the IC region.	
IC_U_G(IC)	DOUBLE PRECISION
Initial x-component of gas velocity in the IC region.	
IC_U_S(IC, Phase)	DOUBLE PRECISION
Initial x-component of solids-phase velocity in the IC region.	
IC_V_G(IC)	DOUBLE PRECISION
Initial y-component of gas velocity in the IC region.	
IC_V_S(IC, Phase)	DOUBLE PRECISION
Initial y-component of solids-phase velocity in the IC region.	
IC_W_G(IC)	DOUBLE PRECISION
Initial z-component of gas velocity in the IC region.	
IC_W_S(IC, Phase)	DOUBLE PRECISION
Initial z-component of solids-phase velocity in the IC region.	
IC_X_G(IC, Species)	DOUBLE PRECISION
Initial mass fraction of gas species n.	
IC_X_S(IC, Phase, Species)	DOUBLE PRECISION
Initial mass fraction of gas species n.	
IC_SCALAR(IC, Scalar Eq.)	DOUBLE PRECISION
Initial value of Scalar n.	
IC_K_TURB_G(IC)	DOUBLE PRECISION
Initial value of K in K-Epsilon.	
IC_E_TURB_G(IC)	DOUBLE PRECISION
Initial value of Epsilon in K-Epsilon.	
IC_DES_FIT_TO_REGION(IC)	LOGICAL
Flag for inflating initial lattice distribution to the entire IC region.	
IC_PIC_CONST_NPC(IC, Phase)	INTEGER
Flag to specify the initial constant number of particles per cell for the PIC method initialization. Statistical weight of parcels will be calculated by the code.	

IC_PIC_CONST_STATWT(IC, Phase)

DOUBLE PRECISION

Flag to specify the initial constant statistical weight for computational particles/parcels. Actual number of parcels will be automatically computed.

8.8 Boundary Conditions

Boundary conditions (BC) are specified over flow planes or 2D surfaces that are normal to one of the coordinate directions and coincide with a face of the scalar control-volume. The values for one of the three pairs of coordinates are equal. The surface is defined by the constant coordinates of each of the four edges, which can be specified with physical coordinates or cell indices, and the two equal values for the direction normal to the face, which can only be specified with physical coordinates. If cell sizes are not small enough to resolve a surface specified using physical coordinates, MFIX will indicate this problem with an error message.

A flow plane must have a wall cell (or an outside boundary) on one side and a flow cell on the other side. The BC section is also used to specify obstacles in the flow domain. Obstacles are 3D regions, just as for the IC regions: X_w X_e, Y_s Y_n, and Z_t Z_b. By default the outside boundary is initialized as no-slip walls. For cylindrical coordinates the axis is initialized as a free-slip wall.

Two boundary surfaces must not intersect. Two obstacle regions may intersect.

BC_X_W(BC)

DOUBLE PRECISION

X coordinate of the west face or edge.

BC_X_E(BC)

DOUBLE PRECISION

X coordinate of the east face or edge.

BC_Y_S(BC)

DOUBLE PRECISION

Y coordinate of the south face or edge.

BC_Y_N(BC)

DOUBLE PRECISION

Y coordinate of the north face or edge.

BC_Z_B(BC)

DOUBLE PRECISION

Z coordinate of the bottom face or edge.

BC_Z_T(BC)

DOUBLE PRECISION

Z coordinate of the top face or edge.

BC_I_W(BC)

INTEGER

I index of the west-most cell.

BC_I_E(BC) INTEGER
I index of the east-most cell.

BC_J_S(BC) INTEGER
J index of the south-most cell.

BC_J_N(BC) INTEGER
J index of the north-most cell.

BC_K_B(BC) INTEGER
K index of the bottom-most cell.

BC_K_T(BC) INTEGER
K index of the top-most cell.

BC_TYPE(BC) CHARACTER
Type of boundary.

DUMMY ----- The specified boundary condition is ignored. This is useful for turning off some boundary conditions without having to delete them from the file.

MASS_INFLOW or *MI* ----- Mass inflow rates for gas and solids phases are specified at the boundary.

MASS_OUTFLOW or *MO* ----- The specified values of gas and solids mass outflow rates at the boundary are maintained, approximately. This condition should be used sparingly for minor outflows, when the bulk of the outflow is occurring through other constant pressure outflow boundaries.

P_INFLOW or *PI* ----- Inflow from a boundary at a specified constant pressure. To specify as the west, south, or bottom end of the computational region, add a layer of wall cells to the west, south, or bottom of the PI cells. Users need to specify all scalar quantities and velocity components. The specified values of fluid and solids velocities are only used initially as MFIX computes these values at this inlet boundary.

P_OUTFLOW or *PO* ----- Outflow to a boundary at a specified constant pressure. To specify as the west, south, or bottom end of the computational region, add a layer of wall cells to the west, south, or bottom of the PO cells.

FREE_SLIP_WALL or *FSW* --- Velocity gradients at the wall vanish. If BC_JJ_PS is equal to 1, the Johnson-Jackson boundary condition is used for solids. A FSW is equivalent to using a PSW with hw=0.

NO_SLIP_WALL or *NSW* ----- All components of the velocity vanish at the wall. If BC_JJ_PS is equal to 1, the Johnson-Jackson boundary condition is used for solids. A NSW is equivalent to using a PSW with vw=0 and hw undefined.

PAR_SLIP_WALL or *PSW* ----- Partial slip at the wall implemented as $dv/dn + hw(v - vw) = 0$, where n is the normal pointing from the fluid into the wall. The coefficients hw and vw should be specified. For free slip set hw = 0. For no slip leave hw undefined (hw=+inf) and set vw = 0. To set hw = +inf, leave it unspecified. If BC_JJ_PS is equal to 1, the Johnson-Jackson boundary condition is used for solids.

8.8.1 Wall boundary conditions

8.8.1.1 Momentum Equations

Partial slip at the wall implemented as $dv/dn + hw (v - vw) = 0$, where n is the normal pointing from the fluid into the wall. If the Johnson and Jackson partial slip boundary condition is not used (i.e., $BC_JJ_PS(bc) = 0$), the coefficients hw and vw should be specified. For free slip set $hw = 0$. For no slip leave hw undefined ($hw = \infty$) and set $vw = 0$. To set $hw = \infty$, leave it unspecified.

BC_HW_G(BC) DOUBLE PRECISION

Gas phase hw for partial slip boundary.

BC_HW_S(BC, Phase) DOUBLE PRECISION

Solids phase hw for partial slip boundary.

BC_UW_G(BC) DOUBLE PRECISION

Gas phase Uw for partial slip boundary.

BC_UW_S(BC, Phase) DOUBLE PRECISION

Solids phase Uw for partial slip boundary.

BC_VW_G(BC) DOUBLE PRECISION

Gas phase Vw for partial slip boundary.

BC_VW_S(BC, Phase) DOUBLE PRECISION

Solids phase Vw for partial slip boundary.

BC_WW_G(BC) DOUBLE PRECISION

Gas phase Ww for partial slip boundary.

BC_WW_S(BC, Phase) DOUBLE PRECISION

Solids phase Ww for partial slip boundary.

BC_JJ_PS(BC) INTEGER

Johnson and Jackson partial slip BC.

0 ----- Do not use Johnson and Jackson partial slip boundary condition.

1 ----- Use Johnson and Jackson partial slip boundary condition.

8.8.1.2 Granular Energy Equation

The granular energy boundary condition is implemented as $dT/dn + hw (T - Tw) = c$, where n is the normal pointing from the fluid into the wall. If the Johnson and Jackson partial slip boundary condition is not used (i.e., $BC_JJ_PS(bc) = 0$), the coefficients hw and c should be specified. For specified heat flux set $hw=0$ and give a value for c . For specified temperature boundary condition leave hw unspecified ($hw=\infty$ and give a value

for T_w .

BC_THETA_M(BC, Phase)

DOUBLE PRECISION

Specified wall value, $THETA_w_M$, in diffusion boundary condition: $d(Theta_M)/dn + H_w (THETA_M - THETA_w_M) = C$, where n is the fluid-to-wall normal.

BC_HW_THETA_M(BC, Phase)

DOUBLE PRECISION

Transfer coefficient, H_w , in diffusion boundary condition: $d(Theta_M)/dn + H_w (THETA_M - THETA_w_M) = C$, where n is the fluid-to-wall normal.

BC_C_THETA_M(BC, Phase)

DOUBLE PRECISION

Specified constant flux, C , in diffusion boundary condition: $d(Theta_M)/dn + H_w (THETA_M - THETA_w_M) = C$, where n is the fluid-to-wall normal.

8.8.1.3 Gas and Solids Energy Equations

The thermal boundary condition implemented as $dT/dn + h_w (T - T_w) = c$, where n is the normal pointing from the fluid into the wall. The coefficients h_w , T_w , and c should be specified. $H_w = 0 \Rightarrow$ specified heat flux; $h_w = \infty \Rightarrow$ specified temperature boundary condition. To set $h_w = \infty$, leave it unspecified and give a value for T_w .

BC_HW_T_G(BC)

DOUBLE PRECISION

Gas phase heat transfer coefficient, H_w , in diffusion boundary condition: $d(T_g)/dn + H_w (T_g - T_w_g) = C$, where n is the fluid-to-wall normal.

BC_TW_G(BC)

DOUBLE PRECISION

Specified gas phase wall temperature, T_w_g , in diffusion boundary condition: $d(T_g)/dn + H_w (T_g - T_w_g) = C$, where n is the fluid-to-wall normal.

BC_C_T_G(BC)

DOUBLE PRECISION

Specified constant gas phase heat flux, C , in diffusion boundary condition: $d(T_g)/dn + H_w (T_g - T_w_g) = C$, where n is the fluid-to-wall normal.

BC_HW_T_S(BC, Phase)

DOUBLE PRECISION

Solids phase heat transfer coefficient, H_w , in diffusion boundary condition: $d(T_s)/dn + H_w (T_s - T_w_s) = C$, where n is the fluid-to-wall normal.

BC_TW_S(BC, Phase)

DOUBLE PRECISION

Specified solids phase wall temperature, T_w_s , in diffusion boundary condition: $d(T_s)/dn + H_w (T_s - T_w_s) = C$, where n is the fluid-to-wall normal.

BC_C_T_S(BC, Phase)

DOUBLE PRECISION

Specified constant solids phase heat flux, C , in diffusion boundary condition: $d(T_s)/dn + H_w (T_s - T_w_s) = C$, where n is the fluid-to-wall normal.

8.8.1.4 Gas and Solids Species Equations

The species diffusion boundary condition is implemented as $dX/dn + h_w (X - X_w) = c$, where n is the normal pointing from the fluid into the wall. The coefficients h_w , X_w , and c should be specified. $h_w = 0 \Rightarrow$ specified species diffusion flux; $h_w = \infty \Rightarrow$ specified species concentration at the boundary. To set $h_w = \infty$, leave it unspecified and give a value for X_w .

BC_HW_X_G(BC, Species)

DOUBLE PRECISION

Gas phase species mass transfer coefficient, H_w , in diffusion boundary condition: $d(X_g)/dn + H_w (X_g - X_{w_g}) = C$, where n is the fluid-to-wall normal.

BC_XW_G(BC, Species)

DOUBLE PRECISION

Specified wall gas species mass fraction, X_w , in diffusion boundary condition: $d(X_g)/dn + H_w (X_g - X_{w_g}) = C$, where n is the fluid-to-wall normal.

BC_C_X_G(BC, Species)

DOUBLE PRECISION

Specified constant gas species mass flux, C , in diffusion boundary condition: $d(X_g)/dn + H_w (X_g - X_{w_g}) = C$, where n is the fluid-to-wall normal.

BC_HW_X_S(BC, Phase, Species)

DOUBLE PRECISION

Solid phase species mass transfer coefficient, H_w , in diffusion boundary condition: $d(X_s)/dn + H_w (X_s - X_{w_s}) = C$, where n is the fluid-to-wall normal.

BC_XW_S(BC, Phase, Species)

DOUBLE PRECISION

Specified solids species mass fraction at the wall, X_w , in diffusion boundary condition: $d(X_g)/dn + H_w (X_g - X_{w_g}) = C$, where n is the fluid-to-wall normal.

BC_C_X_S(BC, Phase, Species)

DOUBLE PRECISION

Specified constant solids species mass flux, C , in diffusion boundary condition: $d(X_s)/dn + H_w (X_s - X_{w_s}) = C$, where n is the fluid-to-wall normal.

8.8.1.5 Scalar Transport Equations

The scalar boundary condition is implemented as $dS/dn + h_w (S - S_w) = C$, where n is the normal pointing from the fluid into the wall. The coefficients h_w , S_w , and c should be specified. $h_w = 0 \Rightarrow$ specified species diffusion flux; $h_w = \infty \Rightarrow$ specified species concentration at the boundary. To set $h_w = \infty$, leave it unspecified and give a value for S_w .

BC_HW_SCALAR(BC, Scalar Eq.)

DOUBLE PRECISION

Scalar transfer coefficient, H_w , in diffusion boundary condition: $d(Scalar)/dn + H_w (Scalar - ScalarW) = C$, where n is the fluid-to-wall normal.

BC_SCALARW(BC, Scalar Eq.)

DOUBLE PRECISION

Specified scalar value at the wall, $ScalarW$, in diffusion boundary condition: $d(Scalar)/dn + H_w (Scalar - ScalarW) = C$, where n is the fluid-to-wall normal.

BC_C_SCALAR(BC, Scalar Eq.)

DOUBLE PRECISION

Specified constant scalar flux, C , in diffusion boundary condition: $d(\text{Scalar})/dn + H_w (\text{Scalar} - \text{Scalar}_w) = C$, where n is the fluid-to-wall normal.

8.8.2 Flow Boundary Conditions

BC_EP_G(BC)

DOUBLE PRECISION

Void fraction at the BC plane.

BC_P_G(BC)

DOUBLE PRECISION

Gas pressure at the BC plane.

BC_ROP_S(BC, Phase)

DOUBLE PRECISION

Bulk density of solids phase at the BC plane.

BC_EP_S(BC, Phase)

DOUBLE PRECISION

Solids volume fraction at the BC plane.

BC_T_G(BC)

DOUBLE PRECISION

Gas phase temperature at the BC plane.

BC_T_S(BC, Phase)

DOUBLE PRECISION

Solids phase-m temperature at the BC plane.

BC_THETA_M(BC, Phase)

DOUBLE PRECISION

Solids phase-m granular temperature at the BC plane.

BC_X_G(BC, Species)

DOUBLE PRECISION

Mass fraction of gas species n at the BC plane.

BC_X_S(BC, Phase, Species)

DOUBLE PRECISION

Mass fraction of solids phase-m, species n at the BC plane.

BC_U_G(BC)

DOUBLE PRECISION

X-component of gas velocity at the BC plane.

BC_U_S(BC, Phase)

DOUBLE PRECISION

X-component of solids-phase velocity at the BC plane.

BC_V_G(BC)

DOUBLE PRECISION

Y-component of gas velocity at the BC plane.

BC_V_S(BC, Phase)

DOUBLE PRECISION

Y-component of solids-phase velocity at the BC plane.

BC_W_G(BC)

DOUBLE PRECISION

Z-component of gas velocity at the BC plane.

BC_W_S(BC, Phase)

DOUBLE PRECISION

Z-component of solids-phase velocity at the BC plane.

For a mass inflow boundary, instead of specifying the normal velocity at a boundary, the gas and solids flow rates may be specified as the volumetric or mass flow rates. If the volumetric or mass flow rate is specified, MFIX will calculate the velocity normal to the boundary. The velocity calculated by MFIX, however, may differ from the velocity calculated based on the physical dimensions of the port because the simulated dimensions may not be exactly equal to the physical dimensions. Specify positive values for all the flow rates. MFIX will assign the correct sign to the computed velocity values.

If the mass or volumetric flow rate is specified for a mass outflow boundary condition, then at every interval BC_DT_0, MFIX will adjust the normal velocity so that the average computed-outflow rate is equal to the specified value. The user is cautioned, however, that if unrealistic mass flow rates are specified, the computations may become unstable. It is better to specify the velocity at the mass outflow boundary, if some amount of fluctuation in the mass outflow rate is tolerable.

BC_VOLFLOW_G(BC)

DOUBLE PRECISION

Gas volumetric flow rate through the boundary.

BC_VOLFLOW_S(BC, Phase)

DOUBLE PRECISION

Solids volumetric flow rate through the boundary.

BC_MASSFLOW_G(BC)

DOUBLE PRECISION

Gas mass flow rate through the boundary.

BC_MASSFLOW_S(BC, Phase)

DOUBLE PRECISION

Solids mass flow rate through the boundary.

MFIX allows the specification of a transient jet with its velocity fluctuating between two values. The jet conditions will override the steady condition specified for the normal velocity. Therefore, if there is no transient jet, do not specify any of the following, except BC_DT_0, which may be required for mass outflow conditions.

BC_DT_0(BC)

DOUBLE PRECISION

The interval at the beginning when the normal velocity at the boundary is equal to BC_Jet_g0. When restarting, run this value and BC_Jet_g0 should be specified such that the transient jet continues correctly. MFIX does not store the jet conditions. For MASS_OUTFLOW boundary conditions, BC_DT_0 is the time period to average and print the outflow rates. The adjustment of velocities to get a specified mass or volumetric flow rate is based on the average outflow rate.

BC_JET_G0(BC)	DOUBLE PRECISION
Value of normal velocity during the initial interval BC_DT_0.	
BC_DT_H(BC)	DOUBLE PRECISION
The interval when normal velocity is equal to BC_Jet_gh.	
BC_JET_GH(BC)	DOUBLE PRECISION
Value of normal velocity during the interval BC_DT_h.	
BC_DT_L(BC)	DOUBLE PRECISION
The interval when normal velocity is equal to BC_JET_gL.	
BC_JET_GL(BC)	DOUBLE PRECISION
Value of normal velocity during the interval BC_DT_L.	
BC_SCALAR(BC, Scalar Eq.)	DOUBLE PRECISION
Boundary value for user-defined scalar equation.	
BC_K_TURB_G(BC)	DOUBLE PRECISION
Boundary value of K for K-Epsilon Equation.	
BC_E_TURB_G(BC)	DOUBLE PRECISION
Boundary value of Epsilon for K-Epsilon Equation.	
BC_VELMAG_G(BC)	DOUBLE PRECISION
Magnitude of gas velocity in a specified boundary region.	
BC_VELMAG_S(BC, Phase)	DOUBLE PRECISION
Magnitude of gas velocity in a specified boundary region.	
BC_PIC_MI_CONST_NPC(BC, Phase)	INTEGER
Flag to specify the constant number of computational particles per cell for the PIC solids inflow BC. Statistical weight of parcels will be calculated by the code.	
BC_PIC_MI_CONST_STATWT(BC, Phase)	DOUBLE PRECISION
Flag to specify the constant statistical weight for inflowing computational particles/parcels. Actual number of parcels will be automatically computed.	
BC_PO_APPLY_TO_DES(BC)	LOGICAL
Flag to make the PO BC invisible to discrete solids. Set this flag to.FALSE.to remove this BC for discrete solids.	
BC_MI_AS_WALL_FOR_DES(BC)	LOGICAL
Flag to make the inflow plane invisible to discrete solids. Set this flag to.FALSE.to remove to inflow plane.	

BC_JJ_M

LOGICAL

Use the modified Johnson and Jackson partial slip BC with variable specularly coefficient.

8.9 Internal Surface

Internal surfaces (IS) are normal to one of the coordinate directions and coincide with one of the faces of the scalar control volume. One of the three pairs of coordinates is equal. The surface is defined by the constant coordinates of each of the four edges, which can be specified with physical coordinates or cell indices, and the two equal values for the direction normal to the face, which can only be specified with physical coordinates. If cell sizes are not small enough to resolve a surface specified using physical coordinates, MFIX will indicate this problem with an error message.

To specify a large number of internal surfaces in a region, a 3D region may be specified. When IS_Type is specified for such regions, add a prefix (X_, Y_, or Z_) to indicate the direction of the internal surfaces; e.g., X_IMPERMEABLE specifies impermeable internal surfaces parallel to the X coordinate.

Internal surfaces act as free-slip walls in stress computations. This default condition cannot be changed.

IS_X_W(IS)

DOUBLE PRECISION

X coordinate of the west face or edge.

IS_X_E(IS)

DOUBLE PRECISION

X coordinate of the east face or edge.

IS_Y_S(IS)

DOUBLE PRECISION

Y coordinate of the south face or edge

IS_Y_N(IS)

DOUBLE PRECISION

Y coordinate of the north face or edge

IS_Z_B(IS)

DOUBLE PRECISION

Z coordinate of the bottom face or edge

IS_Z_T(IS)

DOUBLE PRECISION

Z coordinate of the top face or edge

IS_I_W(IS)

INTEGER

I index of the west-most cell.

IS_I_E(IS)

INTEGER

I index of the east-most cell

IS_J_S(IS) J index of the south-most cell	INTEGER
---	---------

IS_J_N(IS) J index of the north-most cell	INTEGER
---	---------

IS_K_B(IS) K index of the bottom-most cell	INTEGER
--	---------

IS_K_T(IS) K index of the top-most cell	INTEGER
---	---------

IS_TYPE(IS) Type of internal surface <i>IMPERMEABLE</i> or <i>IP</i> ----- No gas or solids flow through the surface. <i>SEMIPERMEABLE</i> or <i>SP</i> ----- Gas flows through the surface with an additional resistance. Solids velocity through the surface is set to zero or to a user- specified fixed value (i.e., solids momentum equation for this direction is not solved).	CHARACTER
--	-----------

IS_PC(IS) Inertial resistance coefficient.	DOUBLE PRECISION
--	------------------

IS_PC(IS) Inertial resistance coefficient.	DOUBLE PRECISION
--	------------------

IS_VEL_S(IS, Phase) Value of fixed solids velocity through semipermeable surfaces.	DOUBLE PRECISION
--	------------------

8.10 Point Sources

Point sources (PS) are used in place of mass inlets where either the geometry and/or grid resolution prohibit proper boundary condition specification. For example, a point source may be used to model an injector with dimensions smaller than the grid. Point sources may be defined within a single computational cell, along a plane, or as a volume of computational cells.

PS_X_W(PS) X coordinate of the west face or edge.	DOUBLE PRECISION
---	------------------

PS_X_E(PS) X coordinate of the east face or edge.	DOUBLE PRECISION
---	------------------

PS_Y_S(PS) Y coordinate of the south face or edge.	DOUBLE PRECISION
PS_Y_N(PS) Y coordinate of the north face or edge.	DOUBLE PRECISION
PS_Z_B(PS) Z coordinate of the bottom face or edge.	DOUBLE PRECISION
PS_Z_T(PS) Z coordinate of the top face or edge.	DOUBLE PRECISION
PS_I_W(PS) I index of the west-most cell.	INTEGER
PS_I_E(PS) I index of the east-most cell.	INTEGER
PS_J_S(PS) J index of the south-most cell.	INTEGER
PS_J_N(PS) J index of the north-most cell.	INTEGER
PS_K_B(PS) K index of the bottom-most cell.	INTEGER
PS_K_T(PS) K index of the top-most cell.	INTEGER
PS_U_G(PS) X-component of incoming gas velocity.	DOUBLE PRECISION
PS_V_G(PS) Y-component of incoming gas velocity.	DOUBLE PRECISION
PS_W_G(PS) Z-component of incoming gas velocity.	DOUBLE PRECISION
PS_MASSFLOW_G(PS) Gas mass flow rate through the point source.	DOUBLE PRECISION
PS_T_G(PS) Temperature of incoming gas.	DOUBLE PRECISION

PS_X_G(PS) DOUBLE PRECISION
Gas phase incoming species n mass fraction.

PS_U_S(PS, Phase) DOUBLE PRECISION
X-component of incoming solids velocity.

PS_V_S(PS, Phase) DOUBLE PRECISION
Y-component of incoming solids velocity.

PS_W_S(PS, Phase) DOUBLE PRECISION
Z-component of incoming solids velocity.

PS_MASSFLOW_S(PS, Phase) DOUBLE PRECISION
Solids mass flow rate through the point source.

PS_T_S(PS, Phase) DOUBLE PRECISION
Temperature of incoming solids.

PS_X_S(PS, Phase, Species) DOUBLE PRECISION
Solids phase incoming species n mass fraction.

8.11 Output Control

Keywords for controlling the output are provided in this section.

RES_DT DOUBLE PRECISION
Interval at which restart (.res) file is updated.

SPX_DT DOUBLE PRECISION
Interval at which .SPX files are updated.

SP1 -----void fraction (EP_G)
SP2 -----Gas pressure (P_G), and Solids pressure (P_star)
SP3 -----Gas velocity (U_G, V_G, W_G)
SP4 -----Solids velocity (U_S, V_S, W_S)
SP5 -----Solids bulk density (ROP_s)
SP6 -----Gas and solids temperature (T_G, T_S)
SP7 -----Gas and solids mass fractions (X_G, X_S)
SP8 -----Granular temperature (THETA_M)
SP9 -----User defined scalars. (SCALAR)
SPA -----Reaction Rates (ReactionRates)
SPB -----Turbulence quantities (K_TURB_G, E_TURB_G)

NRR INTEGER
The number of user defined chemical reactions stored in the *.SPA file.

OUT_DT

DOUBLE PRECISION

Interval at which standard output (.OUT) file is updated. Only run configuration information is written if left undefined. Otherwise all field variables for the entire domain are written in ASCII format to the .OUT file at OUT_DT intervals.

NLOG

INTEGER

Number of time steps between .LOG file updates.

FULL_LOG

LOGICAL

Display the residuals on the screen and provide messages about convergence on the screen and in the .LOG file.

RESID_STRING(Residual Index)

CHARACTER

Specifies the residuals to display.

P0 ----- Gas pressure
PM ----- Solids phase M pressure
R0 ----- Gas density
RM ----- Solids phase M density
U0 ----- Gas phase U-velocity
V0 ----- Gas phase V-velocity
W0 ----- Gas phase W-velocity
UM ----- Solids phase M U-velocity
VM ----- Solids phase M V-velocity
WM ----- Solids phase M W-velocity
T0 ----- Gas temperature
TM ----- Solids phase M temperature
X0NN ----- Gas phase species NN mass fraction
XMNN ----- Solids phase M species NN mass fraction
K0 ----- K-Epsilon model residuals

GROUP_RESID

LOGICAL

Display residuals by equation.

REPORT_NEG_DENSITY

LOGICAL

Provide detailed logging of negative density errors.

[.FALSE.] ----- Do not log negative density errors.
.TRUE. ----- Log negative density errors.

REPORT_MASS_BALANCE_DT

DOUBLE PRECISION

Frequency to perform an overall species mass balance. Leaving undefined suppresses the mass balance calculations which can slightly extend run time.

PHIP_OUT_JJ

LOGICAL

Output the variable specularity coefficient when BC_JJ_M is .TRUE.. The specularity coefficient will be stored in reactionrates array for post-processing by post-mfix. user needs to set nrr to 1 for this purpose. be careful with this setting when reacting flow is simulated.

BDIST_IO

LOGICAL

Use distributed IO :: Each MPI process generates RES/SPx files.

BSTART_WITH_ONE_RES

LOGICAL

Restart a serial IO run (only one RES file was created) with distributed IO.

BWRITE_NETCDF(NetCDF Variable Reference)

LOGICAL

Flag to write variable in NetCDF output file. NetCDF support is not included in MFIX by default. The executable must be compiled and linked with an appropriate NetCDF library to use this functionality.

Variable Index List:

- 1 Void fraction (EP_G)
- 2 Gas pressure (P_G)
- 3 Solids pressure (P_star)
- 4 Gas velocity (U_G, V_G, W_G)
- 5 Solids velocity (U_S, V_S, W_S)
- 6 Solids bulk density (ROP_s)
- 7 Gas temperature (T_G)
- 8 Gas and solids temperature (T_S)
- 9 Gas mass fractions (X_G)
- 10 Solids mass fractions (X_S)
- 11 Granular temperature (THETA_M)
- 12 User defined scalars. (SCALAR)
- 13 Reaction Rates (ReactionRates)
- 14 Turbulence quantities (K_TURB_G, E_TURB_G)

.TRUE. ----- Write variable in NetCDF output.

[.FALSE.] ----- Do not include variable in NetCDF output.

WRITE_VTK_FILES

LOGICAL

Write vtk files at regular intervals.

[.FALSE.] ----- do not write vtk files. if there are cut cells, they will not be displayed from the usual .res file

.TRUE. ----- valid only if cartesian_grid = .true.

TIME_DEPENDENT_FILENAME

LOGICAL

Use time-dependent vtk file names

.FALSE. ----- the vtk file overwrites the previous file (recommended for steady-state computation).

[.TRUE.] ----- a sequential integer is appended to the vtk filenames as they are written to create a series of files (recommended for transient computation).

VTK_DT

DOUBLE PRECISION

Interval (expressed in seconds of simulation time) at which vtk files are written.

VTK_VAR

INTEGER

List of variables written in the vtk files.

[1]	-----	Void fraction (EP_g)
2	-----	Gas pressure, solids pressure (P_g, P_star)
3	-----	Gas velocity (U_g, V_g, W_g)
4	-----	Solids velocity (U_s, V_s, W_s)
5	-----	Solids density (ROP_s)
6	-----	Gas and solids temperature (T_g, T_s)
7	-----	Gas and solids mass fractions (X_g, X_s)
8	-----	Granular temperature (Theta_m)
9	-----	Scalar
10	-----	Reaction rates
11	-----	K and Epsilon
12	-----	Vorticity magnitude and lambda_2
100	-----	Grid Partition
101	-----	Boundary Condition ID
102	-----	Distance to wall
103	-----	DEM facet count
104	-----	DEM Neighboring facets
999	-----	Cell IJK index
1000	-----	Cut face normal vector

FRAME

INTEGER

Starting Index appended to VTU files

VTU_DIR

CHARACTER

Directory where vtk files are stored (default is run directory)

DES_REPORT_MASS_INTERP

LOGICAL

Reports mass based on Lagrangian particles and continuum representation. Useful to ensure mass conservation between Lagrangian and continuum representations. Recommended use for debugging purposes.

Applies to solids models: DEM, PIC

PRINT_DES_DATA

LOGICAL

Allows writing of discrete particle data to output files. Relevant to both granular and coupled simulations.

Applies to solids models: DEM, PIC

DES_SPX_DT

DOUBLE PRECISION

The frequency at which particle data is written. This only applies to pure granular simulations. For coupled simulation, the output frequency is controlled by SPX_DT(1).

DES_RES_DT

DOUBLE PRECISION

The frequency at which _DES.RES file is written. This only applies to pure granular simulations as the restart frequency is governed by RES_DT for coupled simulations.

DES_OUTPUT_TYPE

CHARACTER

The output file format for DES data.

Applies to solids models: DEM, PIC

PARAVIEW ----- ParaView formatted files (.vtp)

TECPLOT ----- Tecplot formatted files (.dat)

DEBUG_DES

LOGICAL

Runtime flag to generate debugging information. Additional data for FOCUS_PARTICLE is saved.

FOCUS_PARTICLE

INTEGER

Specify particle number for particle level debugging details.

PIC_REPORT_SEEDING_STATS

LOGICAL

Flag to print processor level parcel seeding statistics for inflow BC with PIC model.

Applies to solids models: PIC

PIC_REPORT_DELETION_STATS

LOGICAL

Flag to print processor level parcel deletion statistics for outflow BC with PIC model. Not recommended for production runs.

Applies to solids models: PIC

8.12 User-defined functions

Users may modify any *.f or *.inc file under the MFIX model directory. To modify a file, first copy it from the model directory into the run directory. All modifications should be made to the file in the run directory. The build script will automatically create backup copies of the original files and rebuild MFIX with the edited versions in the run directory.

For example, list contents of the `adiabaticFlame` test case located in the `tests` directory:

```
> cd ~/mfix/tests/adiabaticFlame
> ls
AUTOTEST  mfix.dat  usr3.f  usr_rates.f
```

The AUTOTEST directory contains data from previous runs and is used for regression testing. The `mfix.dat` file contains the simulation setup and `usr3.f` and `usr_rates.f` contain user-defined functions (UDFs) required by this simulation.

Launching the build script and passing the “default” argument to generate a serial executable using the GNU Fortran compiler with O3 optimization gives the output:


```
> mkmfix --default
MFIX source: ~/mfix/model
Specified optimization level: 3
Gfortran on 64 bit machine
Checking for user modified files.
> Creating a backup: ./usr3.0f
> Creating a backup: ./usr_rates.0f
> Creating a backup: ./species.0inc
> Updating: ./species.inc
> Updating: ./usr3.f
> Updating: ./usr_rates.f
Building Makefile.
Updating file list.
```

The fourth line indicates that the build script is checking for modified source files in the run directory. The next three lines indicate that backup copies of the original `usr3.f`, `usr_rates.f`, and `species.inc` files are being created. (Note that `species.inc` is generated by the build script when a reaction rates file is detected.) The following three lines then state that the modified source files in the run directory are being copied into the model directory.

Similarly, by moving to a new run directory that does not contain any modified source files, the original files will be restored. Now go to the `cond_conv` test case and list the contents of the directory.

```
> cd ~/mfix/tests/cond_conv
> ls
mfix.dat
```

Here, only the input file is present.

Again, launching the build script and passing the “default” argument to generate a serial executable using the GNU Fortran compiler with O3 optimization gives the output:

```
> mkmfix --default
MFIX source: ~/mfix/model
Specified optimization level: 3
Gfortran on 64 bit machine
Checking for user modified files.
> Restored: ./species.inc
> Restored: ./usr3.f
> Restored: ./usr_rates.f
Building Makefile.
Updating file list.
```

Now that the modified files are no longer in the run directory, the build script restores the original files.



The build script only recognizes MFIX files with `.f` and `.inc` file extensions. New files can be added by placing an empty copy of the file under the model directory.

The source file structure must be mirrored in the run directory when modifying files that are located below the model directory.

For example, to modify the `usr0_des.f` source file located in the `des` subdirectory,

mfix/model/des, first a **des** folder must be created in the run directory:

```
> mkdir des
```

Then the **usr0_des.f** can be copied in this new directory.

```
> cp ~/mfix/model/des/usr0_des.f des/.
```

Modifications should be made to the local copy of **usr0_des.f**.

The following is a list of MFIX files that are usually modified to include chemical reactions and user defined scalars:

File Name:	Usage Description
physical_prop.f	Physical properties.
scalar_prop.f	Properties and source terms in scalar transport equations.
transport_prop.f	Transport properties.
usr_drag.f	User-defined drag function.
usr_rates.f	Chemical reaction rates.
usr_rates_des.f ^[1]	DES chemical reaction rates.

The following routines are used for writing user-defined output:

File Name:	Usage Description
write_usr0.f	Called once at the start of a run. This file is typically used for opening user-defined output files.
write_usr1.f	Called at intervals defined by USR_DT .

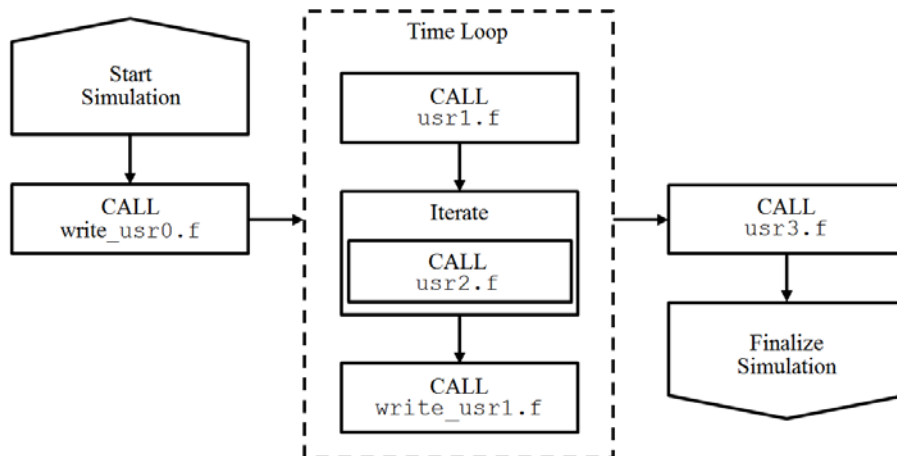
To activate the calls to the following three routines, set **call_usr = .TRUE.** in the input file **mfix.dat**:

File Name:	Usage Description
usr0.f	A subroutine that is called once every run, just before the time-loop begins.
usr1.f	A subroutine that is called once every timestep.
usr2.f	A subroutine that is called once every iteration.
usr3.f	A subroutine that is called once every run, after the time-loop ends.
usrnlst.inc	List of user-defined keywords. These may be used to enter data through the input data file mfix.dat .
usr_init_namelist.f	Initialize user-defined keywords.

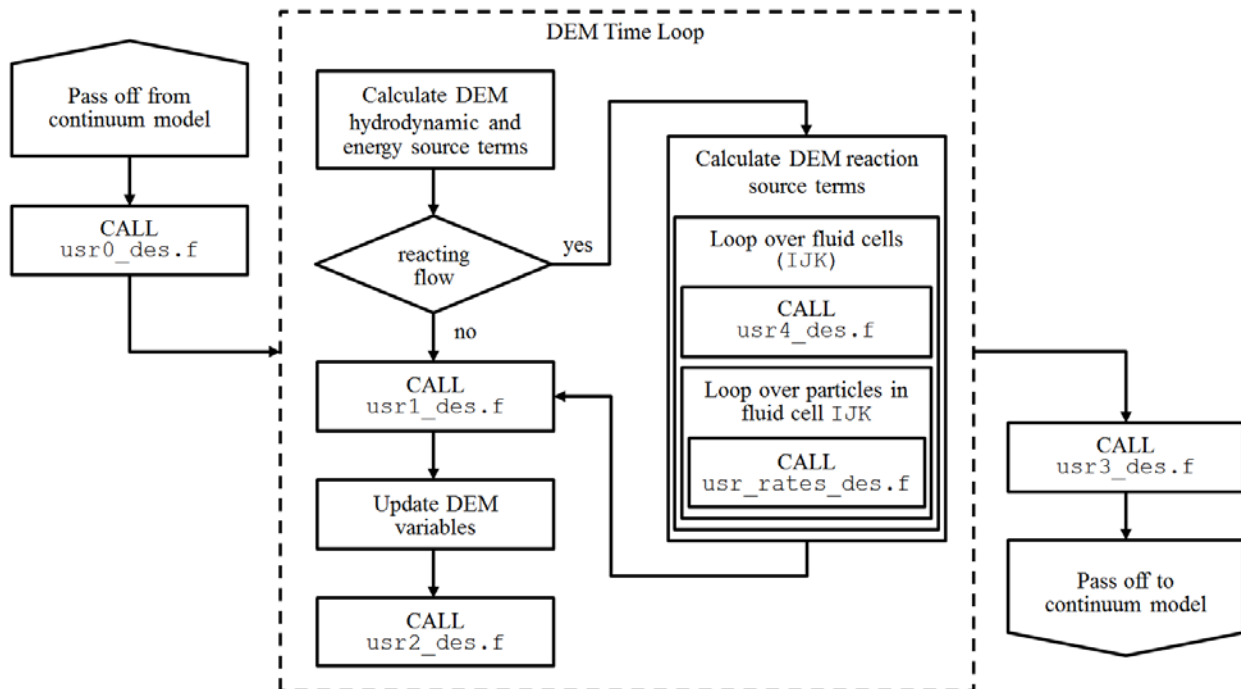
<code>usr_mod.f</code>	User-defined module. Include "Use usr" to use user-defined variables in this module. If allocatable arrays are defined in this module, allocate them in <code>usr0.f</code> .
<code>usr0_des.f</code> ^[1]	A subroutine called before entering the DES time loop.
<code>usr1_des.f</code> ^[1]	A subroutine called every DEM timestep after calculating DES source terms but before source terms are applied to the particles.
<code>usr2_des.f</code> ^[1]	A subroutine called every DES timestep after source terms are applied to the particles.
<code>usr3_des.f</code> ^[1]	A subroutine that is called after completing the DES time loop.
<code>usr4_des.f</code> ^[1]	This subroutine is called from and IJK loop before calculating DEM reaction rates. Used for calculating values that are constant within a fluid cell needed for reaction calculations (e.g., Schmidt Number).

^[1] Denotes files contained in the DES subfolder (mfix/model/des/).

User-defined subroutines call structure:



DES User-defined subroutines call structure:



CALL_USR

LOGICAL

Flag to enable user-defined subroutines: USR0, USR1, USR2, USR3, USR0_DES, USR1_DES, USR2_DES, USR3_DES, USR4_DES.

.TRUE. ----- Call user-defined subroutines.

[.FALSE.] ----- Do NOT call user-defined subroutines.

C

DOUBLE PRECISION

User defined constants.

C_NAME

CHARACTER

Name of user-defined constant. (20 character max)

USR_DT(USR)

DOUBLE PRECISION

Intervals at which subroutine write_usr1 is called.

USR_X_W(USR)

DOUBLE PRECISION

Udf Hook: x coordinate of the west face or edge.

USR_X_E(USR)

DOUBLE PRECISION

Udf Hook: x coordinate of the east face or edge.

USR_Y_S(USR) Udf Hook: y coordinate of the south face or edge.	DOUBLE PRECISION
USR_Y_N(USR) Udf Hook: y coordinate of the north face or edge.	DOUBLE PRECISION
USR_Z_B(USR) Udf Hook: z coordinate of the bottom face or edge.	DOUBLE PRECISION
USR_Z_T(USR) Udf Hook: z coordinate of the top face or edge.	DOUBLE PRECISION
USR_I_W(USR) Udf Hook: i index of the west-most cell.	INTEGER
USR_I_E(USR) Udf Hook: i index of the east-most cell.	INTEGER
USR_J_S(USR) Udf Hook: j index of the south-most cell.	INTEGER
USR_J_N(USR) Udf Hook: j index of the north-most cell.	INTEGER
USR_K_B(USR) Udf Hook: k index of the bottom-most cell.	INTEGER
USR_K_T(USR) Udf Hook: k index of the top-most cell.	INTEGER
USR_TYPE(USR) Udf Hook: Type of user-defined output: Binary or ASCII.	CHARACTER
USR_VAR(USR) Udf Hook: Variables to be written in the user-defined output files.	CHARACTER
USR_FORMAT(USR) Udf Hook: Format for writing user-defined (ASCII) output file.	CHARACTER
USR_EXT(USR) Udf Hook: File extension for the user-defined output.	CHARACTER



The `USR_` keywords do not have any explicit behavior. They are only basic input mechanisms to interact with user-defined functions.

For example, specifying `USR_X_w` does not result in the associated 'I' index, `USR_I_W` being calculated. It is upon the user to ensure that all user-hooks are fully specified.

8.13 Chemical Reactions

Chemical reactions are specified in the data file (`mfix.dat`) by providing species aliases and chemical equations. Rate expressions are specified in one of two user defined subroutines, `usr_rates.f` and `usr_rates_des.f`, respectively. Heats of reaction are automatically calculated. Optionally, users may specify constant heats of reaction in the data file.



An overview of using legacy `rrates.f` files is given at the end of this section. However, **this input method is no longer supported.**

8.13.1 Chemical Reactions Specification

There are five general steps to incorporating chemical reactions into a simulation:

- Provide species names in the data file (`SPECIES_G`, `SPECIES_S`).
- Assign a *unique identifier* (alias) to each species in the data file. (`SPECIES_ALIAS_G` and `SPCIES_ALIAS_S`)
- Define chemical reaction parameters in the data file.
- Define chemical reaction rates in `usr_rates.f` and/or `usr_rates_des.f`.
- Use `make_mfix` to (re)build the MFIX executable.



Species names must appear **exactly** as given in the materials database (see the *Thermochemical Properties* section). Species names are typically 18 characters, and for some species, trailing spaces are needed.

Each species must be assigned a *unique identifier* (alias).

Alias formatting restrictions:

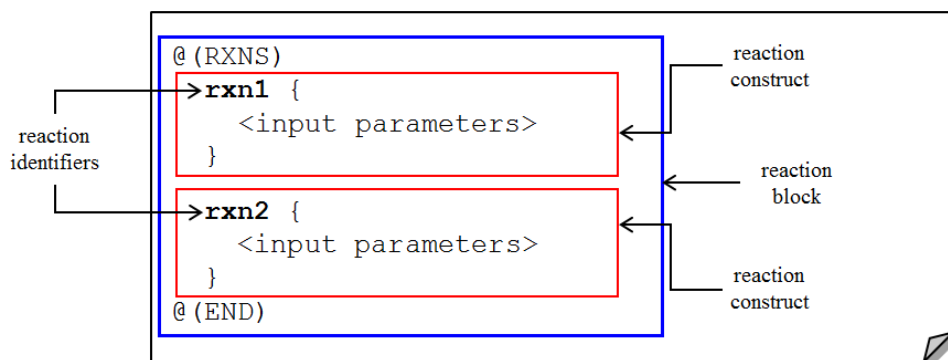
- Aliases must be unique.
- Aliases are limited to 32 characters and must follow FORTRAN variable naming conventions (i.e., alphanumeric combinations with a letter as the first character).

- Aliases are not case sensitive.
- Aliases cannot conflict with existing MFIx variable names (e.g., a species alias of MU_g will cause an error when compiling MFIx).

Define chemical reactions in the data file using species aliases.

Each reaction is identified by a *reaction construct*, and a *reaction block* is used to group reaction constructs in the data file. A reaction construct has the format, rxn_name{...}, where rxn_name is a *unique identifier* for the reaction. Reaction identifiers are limited to 32 characters and must follow FORTRAN variable naming convention.

Reaction input format



MFIx processes chemical reaction data differently than other input in the data file. A *reaction block* indicates the start and end of the reaction input. A *reaction construct* groups a single reaction's input parameters.

There are two reaction block types:

`@ (RXNS) ... @ (END)` – indicates continuum phase chemical reactions (all TFM gas and solids phase reactions and DEM homogeneous gas phase reactions).

`@ (DES_RXNS) ... @ (DES_END)` – indicates heterogeneous DEM chemical reactions (particle/gas).

The following keywords are available within a reaction construct.

CHEM_EQ

CHARACTER

Chemical equation for the reaction constructed from species aliases. Example, Carbon combustion:

`CHEM_EQ = "C + 0.5O2 --> CO"`

* A chemical equation of "NONE" deactivates the reaction during a simulation (e.g., `CHEM_EQ = "NONE"`)

DH

DOUBLE PRECISION

User provided heat of reaction [cal/mole for CGS and J/kmole for SI].

fracDH

DOUBLE PRECISION

The fraction amount of DH supplied to the indexed phase. By default, heats of reaction are automatically calculated and assigned to the appropriate phase(s). However, users may specify a constant heat of reaction, DH, for one or more reactions to override the automated calculations. If DH is given, then fracDH is required. The assigned fractional proportions must sum to one over all phases.

Reaction construct formatting notes:

- Chemical reactions are always specified as irreversible with reactants on the left and products on the right. (CHEM_EQ = "Reactants --> Products")
- An arrow or equals sign can be used to distinguish reactants from products. (Reactants --> Products OR Reactants = Products)
- Reversible reactions are specified as two irreversible reactions. (see below example, Athermal, gas phase, reversible reaction)
- Chemical equations may span several lines by including an ampersand (&) at the end of the line. As the example below illustrates, each line of the chemical equation is bound in quotation marks and the ampersand is located to the right of the second quotation mark.

```
@(RXNS)                                ! Begin reaction block
  CH4_Combustion {                      ! Reaction 1 construct
    chem_eq = "CH4 + 2O2 --> " &        ! Chemical Reaction Line 1
    "CO2 + 2H2O"                        ! Chemical Reaction Line 2
  }                                     ! End reaction 1 construct
@(END)
```

- Chemical equations are limited to 512 characters.
- Chemical equations can be bound within single or double quotes. (CHEM_EQ = 'Reactants = Products' OR "Reactants = Products")
- Catalytic reactions should contain a species from the catalyst phase in the chemical equation with a coefficient of zero. This insures the proper assignment of the heat of reaction. (CHEM_EQ = 'A + 0.Cat -->3.0*R' where Cat is a catalyst phase species)
- Catalyst phase species can be listed as a product, reactant, or both.
-

Several examples illustrating the data file input for several reactions are provided below. Comments are preceded with an exclamation mark (!).

Example: Methane Combustion: $\text{CH}_4(g) + 2\text{O}_2 \rightarrow \text{CO}_2(g) + 2\text{H}_2\text{O}(g)$

Notes: Heat of reaction is automatically calculated (default).

```
NMAX_g = 4                                ! No. of gas phase species
Species_g(1) = "CH4 ANHARMONIC "          ! Methane
Species_g(2) = "O2"                        ! Oxygen
Species_g(3) = "CO2"                       ! Carbon dioxide
Species_g(4) = "H2O"                       ! Water Vapor

Species_Alias_g(1) = "CH4"                 ! Methane
Species_Alias_g(2) = "O2"                  ! Oxygen
Species_Alias_g(3) = "CO2"                 ! Carbon dioxide
Species_Alias_g(4) = "H2O"                 ! Water Vapor

@(RXNS)                                    ! Begin reaction block
  CH4_Combustion {                          ! Reaction 1 construct
    chem_eq = "CH4 + 2O2 --> CO2 + 2H2O"    ! Chemical Reaction Eq
  }                                          ! End reaction 1 construct
@(END)                                     ! End reaction block
```

Example: Athermal, gas phase, reversible reaction: $\text{A}(g) \leftrightarrow \text{R}(g)$

Notes: Species database names and aliases are defined on single lines.

The forward and backward reactions are defined separately.

The heats of reaction are defined as zero (athermal) and explicitly assigned to the gas phase.

```
NMAX_g = 2                                ! No. of gas phase species
Species_g(1) = "A" "R"                    ! Database names
Species_Alias_g(1) = "A" "R"               ! Species aliases

@(RXNS)                                    ! Begin reaction block
  fwd_AtoR {                               ! Reaction 1 construct
    chem_eq = "A --> R"                    ! Chemical Reaction Eq
    DH = 0.0                               ! (cal/moles-reacted)
    fracDH(0) = 1.0                        ! Gas phase HoR
  }                                          ! End reaction 1 construct
  rvs_AtoR {                               ! Reaction 2 construct
    chem_eq = "R --> A"                    ! Chemical Reaction Eq
    DH = 0.0                               ! (cal/moles-reacted)
    fracDH(0) = 1.0                        ! Gas phase HoR
  }                                          ! End reaction 2 construct
@(END)                                     ! End reaction block
```

Example - Char combustion: $C(s) + 0.5O_2(g) \rightarrow CO(g)$

Notes: Species database names and aliases are defined on single lines.

The heat of reaction is defined.

The gas phase receives 20% of the heat of reaction.

Solids phase 1 receives 80% of the heat of reaction.

```
NMAX_g = 2                                ! No. gas phase species

Species_g(1) = "O2" "CO"                  ! Database names
Species_Alias_g(1) = "O2" "CO"            ! Species aliases

NMAX_s(1) = 2                              ! No. solids phase species
Species_s(1,1) = "C(GR) REF ELEMENT"      ! Fixed Carbon (graphite)
Species_s(1,2) = "Coal Ash"               ! Coal Ash

Species_Alias_s(1,1) = "C" "Ash"          ! Fixed Carbon and Coal Ash

@(RXNS)                                   ! Begin reaction block
  Char_Combustion {                       ! Reaction 1 construct
    chem_eq = "C + 0.5O2 --> CO"          ! Chemical Reaction Eq
    DH = -52834.0                         ! (cal/moles-reacted)
    fracDH(0) = 0.2                       ! HoR assigned to gas phase
    fracDH(1) = 0.8                       ! HoR assigned to s. phase 1
  }                                       ! End reaction 1 construct
@(END)                                   ! End reaction block
```

Example – Compound DEM reaction:

CO combustion: $\text{CO}(g) + 0.5\text{O}_2(g) \rightarrow \text{CO}_2(g)$

CO₂ gasification: $\text{C}(s) + \text{CO}_2(g) \rightarrow 2\text{CO}(g)$

Char combustion: $\text{C}(s) + 0.5\text{O}_2(g) \rightarrow \text{CO}(g)$

Notes: Gas phase species names and aliases are defined on the same line.

DES specific keywords are used to defined DES species data.

Heats of reaction for all reactions are calculated automatically.

A TFM reaction block is used for the gas phase homogeneous reaction.

A DEM reaction block is used for gas/solids reactions.

Reaction constructs are given in one line.

```
! Gas phase species data
NMAX_g = 3
Species_g(1) = "O2"      Species_Alias_g(1) = "O2"
Species_g(2) = "CO"      Species_Alias_g(2) = "CO"
Species_g(3) = "CO2"     Species_Alias_g(3) = "CO2"

! DES solids phase species data
NMAX_s(1) = 2
Species_s(1,1) = "C(GR) REF ELEMENT"
Species_s(1,2) = "Coal Ash"

Species_Alias_s(1,1) = "C"
Species_Alias_s(1,2) = "Ash"

! Homogeneous gas phase reactions
@(RXNS)
  CO_Combustion { chem_eq = "CO + 0.5O2 --> CO2" }
@(END)

! DES Reaction block
@(DES_RXNS)
  CO2_Gasification { chem_eq = "2.0C + O2 --> 2CO" }
  Char_Combustion { chem_eq = "C + CO2 --> 2CO" }
@(DES_END)
```

Additional comments:

- Coal Ash is not a species included in the thermochemical database and would require that the properties be given in the data file (see *Section 5.12 Thermochemical properties*).
- One-line reaction constructs are only possible when the heat of reaction is automatically calculated (i.e., the chemical equation is the only input parameter).

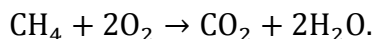
Define chemical reaction rates in UDF files (`usr_rates.f` and `usr_rates_des.f`).

- A reaction rate should be given in either `usr_rates.f` or `usr_rates_des.f` for each reaction listed in the data file.
- All TFM gas and solids phase reactions as well as homogeneous gas phase reactions for DEM simulations are to be included in `usr_rates.f`. Reaction rates defined in `usr_rates.f` must have units of reacted moles per time per volume (i.e., moles/sec/cm³ for CGS units and kmoles/sec/m³ for SI units).
- All discrete phase heterogeneous (particle/gas) reactions are to be included in `usr_rates_des.f` located in the `des` subfolder. Reaction rates defined in `usr_rates_des.f` must have units of reacted moles per time (i.e., moles/sec).



Formation and consumption rates are automatically calculated for each species from the reaction rate and chemical equation.

The rate in terms of reacted moles is related to the rates of formation and consumption through the stoichiometric coefficients. For example, consider homogeneous gas phase reaction of methane combustion:



The rate in terms of reacted moles, *Rate*, is related to the rates of formation and consumption as

$$\begin{aligned} \text{Rate} &= \frac{-r_{\text{CH}_4}}{1} \left(\frac{\text{mol}_{\text{CH}_4}/(\text{s} \cdot \text{cm}^3)}{\text{mol}_{\text{CH}_4}} \right) = \frac{-r_{\text{O}_2}}{2} \left(\frac{\text{mol}_{\text{O}_2}/(\text{s} \cdot \text{cm}^3)}{\text{mol}_{\text{O}_2}} \right) \\ &= \frac{r_{\text{CO}_2}}{1} \left(\frac{\text{mol}_{\text{CO}_2}/(\text{s} \cdot \text{cm}^3)}{\text{mol}_{\text{CO}_2}} \right) = \frac{r_{\text{H}_2\text{O}}}{2} \left(\frac{\text{mol}_{\text{H}_2\text{O}}/(\text{s} \cdot \text{cm}^3)}{\text{mol}_{\text{H}_2\text{O}}} \right), \end{aligned}$$

where $-r_{\text{CH}_4}$ and $-r_{\text{O}_2}$ are the rates of consumption of methane and oxygen, and r_{CO_2} and $r_{\text{H}_2\text{O}}$ are the rates of formation of carbon dioxide and water vapor, respectively. Each reaction rate is assigned to the variable `RATES(rxn_name)`, where `rxn_name` is the reaction identifier used in the reaction construct. To minimize input errors when specifying reaction rates, species aliases (`SPECIES_ALIAS`) defined in the data file should be used in lieu of the associated species index.

For example, if oxygen is defined as gas phase species 2 with an alias of "O2", (e.g., `SPECIES_ALIAS_g(2) = "O2"`), when accessing gas phase species data for oxygen (e.g., molecular weight; `MW_g`), "O2" should be used and not the integer index 2, (e.g., `MW_g(O2)`).

A similar procedure is used for DES reactions with the exception that the reaction rate is assigned to the variable `DES_RATES(rxn_name)`, where `rxn_name` is the reaction identifier used in the reaction construct.

Example: Methane Combustion: $\text{CH}_4(g) + 2\text{O}_2 \rightarrow \text{CO}_2(g) + 2\text{H}_2\text{O}(g)$

Notes: Species database names and alias are defined on the same line.

The fluid cell index (IJK) is passed as a dummy argument.

Global field variables are referenced (RO_g, X_g, T_g, and EP_g)

Species aliases (O2 and CH4) are used instead of the species indices.

Reaction identifier (CH4_Combustion) is used in the rates array.

Reaction rate is stored for post processing (see below).

mfix.dat input:

```
NMAX_g = 4
Species_g(1) = "CH4 ANHARMONIC " Species_Alias_g(1) = "CH4"
Species_g(2) = "O2" Species_Alias_g(2) = "O2"
Species_g(3) = "CO2" Species_Alias_g(3) = "CO2"
Species_g(4) = "H2O" Species_Alias_g(4) = "H2O"

@(RXNS)
  CH4_Combustion { chem_eq = "CH4 + 2O2 --> CO2 + 2H2O" }
@(END)
```

usr_rates.f:

```
SUBROUTINE USR_RATES(IJK, RATES)

  DOUBLE PRECISION, INTENT(IN) :: IJK          ! Fluid Cell Index
  DOUBLE PRECISION, INTENT(OUT) :: RATES(:)    ! Reaction Rates
  DOUBLE PRECISION c_O2 ! Oxygen concentration (mol/cm^3)
  DOUBLE PRECISION c_CH4 ! Methane concentration (mol/cm^3)

  ! Calculate species concentrations:
  c_O2 = (RO_g(IJK) * X_g(IJK,O2))/MW_g(O2)
  c_CH4 = (RO_g(IJK) * X_g(IJK,CH4))/MW_g(CH4)

  ! Methane Combustion
  !-----//
  RATES(CH4_Combustion) = 6.7d12 * exp(-2.4358d4/T_g(IJK)) * &
    EP_g(IJK) * (c_O2**1.3) * (c_CH4**0.2)

  ! Store the reaction rate for output/post processing.
  IF(CH4_Combustion <= NRR) &
    ReactionRates(IJK, CH4_Combustion) = RATES(CH4_Combustion)

END SUBROUTINE USR_RATES
```

Example: Athermal, gas phase, reversible reaction: $A(g) \leftrightarrow R(g)$

Notes: Species database names and alias are defined on the same line.

The fluid cell index (IJK) is passed as a dummy argument.

Global field variables are referenced (RO_g, X_g, T_g, and EP_g)

data file:

```
NMAX_g = 2                                ! No. of gas phase species
Species_g(1) = "A" "R"                    ! Database names
Species_Alias_g(1) = "A" "R"              ! Species Aliases

@(RXNS)                                   ! Begin reaction block
  fwd_AtoR {                               ! Reaction 1 construct
    chem_eq = "A --> R"                   ! Chemical Reaction Eq
    DH = 0.0                               ! (cal/moles-reacted)
    fracDH(0) = 1.0                       ! Gas phase HoR
  }                                         ! End reaction 1 construct
  rvs_AtoR {                               ! Reaction 2 construct
    chem_eq = "R --> A"                   ! Chemical Reaction Eq
    DH = 0.0                               ! (cal/moles-reacted)
    fracDH(0) = 1.0                       ! Gas phase HoR
  }                                         ! End reaction 2 construct
@(END)                                    ! End reaction block
```

usr_rates.f:

```
SUBROUTINE USR_RATES(IJK, RATES)
  DOUBLE PRECISION, INTENT(IN) :: IJK      ! Fluid Cell Index
  DOUBLE PRECISION, INTENT(OUT) :: RATES(:) ! Reaction Rates
  DOUBLE PRECISION c_A ! species A concentration (mol/cm^3)
  DOUBLE PRECISION c_R ! species R concentration (mol/cm^3)

  ! Calculate species concentrations:
  c_A = (RO_g(IJK) * X_g(IJK,A))/MW_g(A)
  c_R = (RO_g(IJK) * X_g(IJK,R))/MW_g(R)

  ! Forward Reaction: A --> R (reacted moles/sec.cm^3)
  !-----//
  RATES(fwd_AtoR) = 1.2d17 * exp(-5.837d3/T_g(IJK)) * &
    EP_g(IJK) * c_A

  ! Reverse Reaction: R --> A (reacted moles/sec.cm^3)
  !-----//
  RATES(rvs_AtoR) = 2.5d41 * exp(-1.4897d4/T_g(IJK)) * &
    EP_g(IJK) * c_R

END SUBROUTINE USR_RATES
```

Example - Char combustion: $C(s) + 0.5O_2(g) \rightarrow CO(g)$

Notes: A representative data file input was presented previously.

The fluid cell index (IJK) is passed as a dummy argument.

Algebraic expressions for the rate limiting steps are omitted for brevity.

```

SUBROUTINE USR_RATES(IJK, RATES)
  DOUBLE PRECISION, INTENT(IN) :: IJK      ! Fluid Cell Index
  DOUBLE PRECISION, INTENT(OUT) :: RATES(:) ! Reaction Rates
  :
! Rate limiting steps:
  DOUBLE PRECISION k_f      ! film diffusion      (cm/sec)
  DOUBLE PRECISION k_a      ! ash layer diffusions (cm/sec)
  DOUBLE PRECISION k_s      ! chemical kinetics   (cm/sec)
  DOUBLE PRECISION k_eff    ! effective rate      (cm/sec)

! Total surface area of solids phase 1 in IJK
  DOUBLE PRECISION Sa      ! (cm^2/cm^3)

! C + 0.5O2 --> CO (reacted moles/sec.cm^3)
!-----//
! Verify that solids are present
  IF(.NOT.COMPARE(EP_g(IJK),ONE)) THEN
! Calculate film diffusion rate
    k_f = < film diffusion rate expression >      ! (cm/sec)
! Calculate ash diffusion rate
    k_a = < ash diffusion rate expression >        ! (cm/sec)
! Calculate kinetic rate rate
    k_s = < kinetic rate expression >              ! (cm/sec)

! Effective rate (cm/sec)
    k_eff = ONE/(ONE/k_a + ONE/k_f + ONE/k_s)

! Calculate total surface area of solids phase 1
    Sa = 6.0 * EP_s(IJK,1) / D_p0(1)
! Calculate the reaction rate.
    RATES(Char_Combustion) = 2.0 * (Sa * k_eff * Conc(O2))
  ELSE
! No solids --> No reaction
    RATES(Char_Combustion) = ZERO
  ENDIF
END SUBROUTINE USR_RATES

```

See `mfix/model/tutorial/SpoutedBedCombustor` for details on a similar simulation setup.

Example – DES droplet evaporation: $H_2O(l) \rightarrow H_2O(g)$

Notes: Various algebraic expressions in the sample UDF are omitted for brevity.
 The global particle index (NP), phase index (pM), and fluid cell index (IJK)
 are passed as dummy arguments.

mfix.dat input:

```
NMAX_g = 2                                ! No. of gas phase species
Species_g(1) = "Air" "H2O"                ! Database names
Species_Alias_g(1) = "Air" "Vapor"        ! Species Aliases

NMAX_s(1) = 1                                ! No. of solids phase species
Species_s(1,1) = "H2O(L)"                 ! Database names
Species_Alias_s(1,1) = "Liquid"           ! Species Aliases

@(DES_RXNS)
  Evap { Liquid --> Vapor }
@(DES_END)
```

usr_rates_des.f input:

```
SUBROUTINE USR_RATES_DES(NP, pM, IJK, DES_RATES)
  DOUBLE PRECISION, INTENT(IN) :: NP      ! Global particle index
  DOUBLE PRECISION, INTENT(IN) :: pM      ! Particle solid phase
  DOUBLE PRECISION, INTENT(IN) :: IJK     ! Fluid Cell Index
  DOUBLE PRECISION, INTENT(OUT) :: DES_RATES(:) ! Reaction Rates

  !-----//
  ! Calculate the concentration gradient (mole/cm^3)
  Cmg_H2O = < expression for calculating gradient >

  IF(Cmg_H2O > ZERO) THEN
    ! Calculate mass transfer coefficient (cm/sec)
    H2O_xfr = < mass transfer coeff calculation >
    ! Calculate droplet surface area (cm^2)
    Sa = Pi * 4.0d0 * (DES_RADIUS(NP)**2)
    ! Calculate the mass transfer rate (moles/sec)
    DES_RATES(Evap) = Sa * H2O_xfr * Cmg_H2O
  ENDIF

  ! Store the reaction rate for post processing.
  IF(Evap <= NRR) ReactionRates(Evap) = &
    ReactionRates(IJK, Evap) + DES_RATES(Evap)

END SUBROUTINE USR_RATES_DES
```

See mfix/tests/dem-tests/evaporation for additional details.

Use `make_mfix` to (re)build the MFIX executable.

Detailed instructions on building the MFIX executable are given in earlier sections. Run `make_mfix` to rebuild `mfix.exe` after making any of the following modifications:

- Changing the number, order, or alias of any species in the data file.
- Changing the number, order, or name of any chemical reaction in the data file .
- Changing the chemical reaction rates in either `usr_rates.f` or `usr_rates_des.f`.



`make_mfix` preprocesses the data file to generate the `species.inc` file which is included within the `usr_rates.f` and `usr_rates_des.f` files as code. Therefore changes in the data file may result in the executable being out of date.

Additional reaction information

To write out reaction rates to SPx file:

1. In the data file, `mfix.dat`, set `NRR` to the desired number of reaction rates to be written out to the file `*.SPA`. This number is typically less than or equal to the total number of reactions.
2. In a reaction UDF (`usr_rates.f` or `usr_rates_des.f`) assign the desired reaction information to the variable `ReactionRates`. `ReactionRates` is a two-dimensional array. The first index references the fluid cell, `IJK`, while the second index ranges from 1 to `NRR`.



If the second index exceeds `NRR`, a run time error can result from over indexing the array. Using logical checks can eliminate potential errors!

Two of the above examples illustrate using the `ReactionRates` variable:

1. *Methane Combustion*: The calculated reaction rate is directly stored, and logical check is used to prevent over indexing the `ReactionRates` array.
2. *DES droplet evaporation*: The calculated reaction rate is added to the storage array. Adding the calculated data to the storage variable is needed in DES since several discrete particles may exist in a single fluid cell. Again, a logical check is performed to prevent over indexing the array.

Using an existing (legacy) `rrates.f` file:

The legacy `rrates.f` file should be copied to the run directory. Additionally, the following keywords should be specified in the data file:

USE_RRATES

LOGICAL

Flag to use legacy chemical reaction UDFs.

SPECIES_NAME

CHARACTER

Names of gas and solids phase species as it appears in the materials database. The first `NMAX(0)` are the names of gas species. The next `NMAX(1)` are the names of solids phase-1 species, etc.

NMAX

INTEGER

Number of species in phase `m`. Note that the gas phase is indicated as `m=0`.



Legacy species keywords `, NMAX(m)` and `SPECIES_NAME(n)`, are **required** when using a legacy `rrates.f` file. Current species keywords `NMAX_g`, `NMAX_s`, `SPECIES_g`, and `SPECIES_s` **cannot** be used.



The only modification needed for a legacy `mfix.dat` and `rrates.f` file combination is the inclusion of `USE_RRATES=.TRUE.` in the data file. An example of legacy file usage: `mfix/tutorials/reactor1b`

Additional remarks:

- `make_mfix` requires that the data file, `mfix.dat`, be present in the run directory as the species aliases and reaction identifiers are needed to construct a `species.inc` file.
- Species aliases and reaction identifiers must be unique. `make_mfix` performs a cursory check on the supplied data and exits if non unique entries are identified.
- If any species alias or reaction identifier conflicts with an existing global variable in MFIX, an error will be reported and the code will fail to compile.

8.13.2 Stiff Chemistry Solver

A stiff chemistry solver has been fully integrated into MFIX. This approach first solves the convection/diffusion equations without chemical reaction source terms. A coupled set of ODEs is then directly integrated to impose chemical reactions effects. This approach may decrease simulation time by permitting larger time steps within the

convection/diffusion model. However, the stiff chemistry solver may increase simulation time, especially if reactions are not stiff.

The stiff chemistry solver is invoked by specifying the following keyword:

STIFF_CHEMISTRY

LOGICAL

Flag to use stiff chemistry solver (Direct Integration).

STIFF_CHEM_MAX_STEPS

INTEGER

Maximum number of internal steps ODEPACK may use to integrate over the time interval. Leaving this value unspecified permits an unlimited number of steps. The stiff solver reports the number of cells that exceed the number of steps as 'incomplete'.

Reactions are specified using the same approach outlined in the chemical reactions section, namely:

1. Provide species names in the data file.
2. Assign a unique identifier (alias) to each species in the data file.
3. Define chemical reaction parameters in the data file.
4. Define chemical reaction rates in `usr_rates.f`
5. Use `make_mfix` to (re)build the MFIX executable.



The stiff chemistry solver does not support legacy rates.f files



The stiff chemistry solver is not available with a DES simulations.

Additional remarks:

- Variables governing ODE convergence criteria are specified as parameters in `stiff_chem_mod.f` found in the `model/chem` directory. Additional information on these parameters and their usage is available in `model/ODEPACK.F`.
- It is recommended to first run your simulation in debug mode. This will catch some common programmatic errors in the `usr_rates.f` file. Additionally, the stiff chemistry solver checks for NaNs calculated in the `usr_rates.f` file.
- The tutorial located in `mfix/tutorials/silane_pyrolysis` shows how to use the stiff chemistry solver.

8.14 Thermochemical Properties

The directory `mfix/model/thermochemical` contains the database of Burcat and Ruscic (2005) and routines for reading the database. With linkage to this database the users need not manually enter data for molecular weight, specific heat, and heats of reactions. Instead the users need to enter the names of the species (keyword `SPECIES_g` and `SPECIES_s`) in the data file. MFIX reads the necessary thermochemical data from files in the following order:

1. `mfix.dat`
2. `BURCAT.THR` file in the run directory
3. `mfix/model/thermochemical/BURCAT.THR`.

The species names are case sensitive and should match the names in `BURCAT.THR` exactly; alternatively aliases can be defined for common species, such as `O2`, in `read_therm.f`. See `mfix/tests/thermo` for a sample case that accesses the database. The format of `BURCAT.THR` file resembles CHEMKIN format, but with several notable differences. Thermochemical data must start below a line that starts with `THERMO DATA`.

Example dataset from `BURCAT.THR` with notations:

74-82-8																	
CH4 METHANE Same as the Anharmonic but calculated Using the RRHO method rather than the NRRHO2. Max Lst Sq Error Cp @ 6000. K 0.62%.																	
CH4 RRHO g 8/99C 1.H 4. 0. 0.G 200.000 6000.000 B 16.04246 1																	
1.91178600E+00 9.60267960E-03 -3.38387841E-06 5.38797240E-10 -3.19306807E-14 2																	
-1.00992136E+04 8.48241861E+00 5.14825732E+00 -1.37002410E-02 4.93749414E-05 3																	
-4.91952339E-08 1.70097299E-11 -1.02453222E+04 -4.63322726E+00 -8.97226656E+03 4																	

Each entry in the database starts with a unique CAS identifier (74-82-8) for the species, followed by several lines of comments highlighted in green. The data section starts with the species name in columns 1-18 (`CH4 RRHO`). Common species names may be followed by strings (`RRHO`) that identify the method used to determine the coefficients. Additional information follows the species name. The numbers toward the end of the line are the temperature limits (200.000 6000.000) in degrees Kelvin where the property calculation is valid and the molecular weight (16.04246). Unlike CHEMKIN the common temperature for the high and low temperature branches are not

recorded; it is always 1000 K. The next three lines give the fourteen coefficients (seven coefficients each for the high and low temperature branches) and the formation enthalpy at 298 K (which is also not included in CHEMKIN format). All the coefficients and the enthalpy of formation are normalized with the gas constant R (cal/mol/K). The low temperature coefficients (a_L) should be used for temperatures in the range T_{low} to 1000K and the high temperature coefficients (a_H) should be used for temperatures in the range 1000K to T_{high} . The coefficients are stored in a fixed format (E15.0) as follows:

where ΔH_f° is the formation enthalpy at 298K.

The normalized specific heat is given by

$$C_p/R = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4.$$

Additional database comments:

- A number of species in the database have a lower temperature limit of 300K which is 2 degrees above the reference temperature (298 K) used for formation enthalpy calculation. For those species MFIx relaxes the lower limit for C_p calculations to 298 K to enable heat of reaction calculation.
see `read_database.f`
- The database reader is set up such that the database is read only if necessary.
- For additional details see the Burcat and Ruscic (2005) report located in the thermo-chemical subdirectory, `mfix/model/thermochemical/intro.pdf`.

8.15 Parallelization Control

The parallel performance depends on several things and one has to evaluate different options before choosing the right strategy for the problem at hand. For e.g. if the J direction is the strongest coupled direction, the preconditioning for the linear solver will be poor if there is decomposition in that direction. However, since decomposing in all the directions reduces the processor grid surface area to the volume, the communication cost will be less for the same computational grid. The preconditioners are chosen with the keyword `LEQ_PC`. In addition to `LINE` relaxation, one can choose the “DIAG” or “NONE” preconditioner that reduces interprocessor communications but would increase the number of linear equation solver iterations. The `DIAG` and `NONE` choices for preconditioners may be appropriate for all equations except the continuity (or pressure and volume fraction correction) equations. The parallel performance is greatly dependent on the choices stated here, and some trial and error may be required to determine the right combination of decomposition direction and the choice of preconditioners to get the best performance in production runs.

`NODESI * NODESJ * NODESK` must be the same as the number of processors

specified using the mpirun (or equivalent command). Otherwise the code will return with an error.

NODESI	INTEGER
Number of grid blocks in x-direction.	

NODESJ	INTEGER
Number of grid blocks in y-direction.	

NODESK	INTEGER
Number of grid blocks in z-direction.	

SOLVER_STATISTICS	LOGICAL
Print out additional statistics for parallel runs	

DEBUG_RESID	LOGICAL
Group residuals to reduce global collectives.	

ENABLE_DMP_LOG	LOGICAL
All ranks write error messages.	

DBGPRN_LAYOUT	LOGICAL
Print the index layout for debugging.	

8.16 Batch Queue Environment

MFIX can be used on systems where code execution is controlled through batch queue submission system instead of interactive or background job type methods shown in the previous section. Usually the user specifies the wall clock time duration of the job and batch queuing system prioritize incoming jobs based on their resource allocation requests. In order for MFIX to avoid abrupt and abnormal termination at the end of the batch job session, several keywords need to be entered in mfix.dat. Controlled and clean termination in environments with batch queue is important as the system may terminate the batch job while MFIX is writing out *.SP files, which may corrupt the files or cause loss of data.

For this purpose, MFIX checks whether the user-specified termination criteria is reached at the beginning of each time step. However, to avoid performance bottlenecks on small systems where the user is running their jobs without a batch queue, this feature is disabled by default. In order to enable this feature the following block of commands need to be entered in mfix.dat.

```

CHK_BATCHQ_END = .TRUE.      ! Enable the controlled termination feature
BATCH_WALLCLOCK = 3600.0    ! Specify the total wall clock duration
                             ! of your job in seconds
TERM_BUFFER = 300.0         ! Specify a buffer time to start
                             ! clean termination of MFIX
    
```

Setting `CHK_BATCHQ_END = .TRUE.` in `mfix.dat` will enable the checking of the termination criteria at the beginning of each time step. In the above example, the user has set the total wall clock time for the duration of the batch session to 1 hour (this is specified in seconds in `mfix.dat`) and a buffer of 300 seconds has been set so that MFIX has sufficient time to terminate cleanly by writing out all `*.SP` and `*.RES` files before the batch session terminates. The duration of the buffer is critical for simulations with large files. MFIX will check if `elapsed time >= (BATCH_WALLCLOCK - TERM_BUFFER)` to start clean termination.

Another way to gracefully terminate MFIX as soon as possible is to create an empty file named `MFIX.STOP` (filename all uppercase) in the working directory where MFIX runs. At the beginning of each time step if `MFIX.STOP` file is detected to exist, then MFIX will terminate gracefully by saving all `*.SP` and `*.RES` files. `CHK_BATCHQ_END` flag must be set to `.TRUE.` in order to activate this feature.

The following command can be used to gracefully terminate MFIX:

```
> touch MFIX.STOP
```

Remember to erase the file once MFIX terminates, otherwise the next time MFIX is run in the same directory it will terminate immediately.

```
> rm -f -r ./MFIX.STOP
```

CHK_BATCHQ_END

LOGICAL

Enables controlled termination feature when running under batch queue system to force MFIX to cleanly terminate before the end of wall clock allocated in the batch session.

BATCH_WALLCLOCK

DOUBLE PRECISION

Total wall-clock duration of the job, in seconds.

TERM_BUFFER

DOUBLE PRECISION

Buffer time specified to allow MFIX to write out the files and cleanly terminate before queue wall clock time limit is reached such that $(BATCH_WALLCLOCK - TERM_BUFFER)$ is less than then batch queue wall clock time limit, in seconds.

9 Mailing lists

Several mailing lists are available to communicate among MFIX users and developers. When your subscription to MFIX is accepted, you are automatically added to the mfix-news mailing list, where important announcements about MFIX are shared with the MFIX community.

The most widely used mailing list is mfix-help, which allows users to post questions and eventually help other users with similar issues.

The mailing list home page is located at <https://mfix.netl.doe.gov/sympa>. Click on the “List of lists” tab to view all available mailing lists. Most of them have a very low bandwidth, and most users only subscribe to the mfix-help list.

Once you subscribe to a list, you can send/receive messages to/from the MFIX community. You can also search archived messages to see if there is already a solution to a common problem.

There are many options to manage your subscription, including subscribing, unsubscribing, and choosing the delivery mode.

Please visit <https://mfix.netl.doe.gov/sympa/help/user> to view the mailing list user guide.

Mailing list etiquette:

- 1) Please allow sufficient time (say 2 to 3 business days) for MFIX developers and users to reply before posting unanswered questions again.
- 2) Unless prior arrangement has been made with a given MFIX developer, do not send requests directly to the developer, but send the request to the appropriate mailing list instead. This ensures proper archiving of the thread and provides better opportunity for everyone to reply. Follow-up questions should also be sent to the mailing list.
- 3) Do not ask for a copy of a reference, e.g., a journal article.
- 4) Prior to submitting help requests regarding MFIX installation or compilation issues, please check the archives of mfix-help and if you are still having a problem, email mfix-help@mfix.netl.doe.gov by providing the following important details in your message after the description of the problem encountered:
 - a. MFIX version you are trying to install or run
 - b. Some details on your operating system environment (for Linux: copy and paste the response of `uname -a` command, Linux distribution name and version also)
 - c. Your compiler name and version number (e.g. `ifort -v` will give the version number for Intel fortran compiler)

- d. Output for your \$PATH environment (in csh type `echo $PATH`)
- e. Your MPI library name and version number (if compilations problem with DMP mode encountered but make sure you can compile and run a simple hello world type MPI program with your current installation) Also please provide hardware details such as number of cores per socket in your system (or send the output for `cat /proc/cpuinfo` and how many cores you are trying to utilize.



Common reasons you may not receive an answer to your request

1. You did not subscribe to the mailing list.
2. You sent the request to an individual and not to the mailing list.
3. Your question has already been answered and is available in the archive.
4. You did not provide sufficient description of your problem (saying “It doesn’t work” is not useful).
5. Your question is outside the scope of the mailing list.

10 User contribution

If you wish to contribute to the development of MFIX, please contact the MFIX team at admin@mfix.netl.doe.gov. We are looking for simulation results (figures, animations, input files, user-defined subroutines), and new models that could benefit the entire MFIX community. If you have written or know any publication that uses MFIX, please let us know and we will post the citation on the website (<https://mfix.netl.doe.gov/citations.php>). Proper credit will be given to all contributors.