

Release 2014-1

Fall 2014

***Image credits:***

Cover page: Aaron Morris, *University of Colorado at Boulder*: MFIX-DEM simulation of solids falling through an array of hexagonal tubes.

Page 7, top: Aytekin Gel, *ALPEMI Consulting, LLC*: MFIX-TFM simulation of coal jet penetration, colored by CO2 species mass fraction.

Page 7, bottom: Jordan Musser, *DOE NETL*: Reactive MFIX-DEM simulation of a spouted bed, particles colored by temperature.

Page 8, top: Rahul Garg, *URS E&C Inc.*: MFIX-PIC simulation of a cyclone, showing particles and streamlines, colored by velocity.

Page 8, 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](#_Toc397681726)

[2 2014-1 Release notes 7](#_Toc397681727)

[3 Development state of MFIX models 9](#_Toc397681728)

[4 Building and Running MFIX on a UNIX/LINUX Workstation 11](#_Toc397681729)

[4.1 Creating MFIX Directory 11](#_Toc397681730)

[4.2 Alias creation (optional) 12](#_Toc397681731)

[4.3 Building MFIX 12](#_Toc397681732)

[4.3.1 MFIX Build Script 13](#_Toc397681733)

[4.3.2 Building MFIX for the first time: A step-by-step example 15](#_Toc397681734)

[4.3.3 Compiling POSTMFIX 17](#_Toc397681735)

[4.4 Starting a New Run 19](#_Toc397681736)

[4.5 Modifying the Post-Processing Codes 23](#_Toc397681737)

[5 Setting Up and Running MFIX on Windows 23](#_Toc397681738)

[5.1 Prebuilt Windows binaries for MFIX 23](#_Toc397681739)

[5.2 Building MFIX with Cygwin and gfortran 23](#_Toc397681740)

[6 MFIX at Run Time 25](#_Toc397681741)

[6.1 MFIX Output and Messages 25](#_Toc397681742)

[6.2 Restarting a Run 27](#_Toc397681743)

[7 When the Run Does Not Converge 27](#_Toc397681744)

[8 Data Visualization with ParaView 27](#_Toc397681745)

[9 User-Defined Subroutines 29](#_Toc397681746)

[10 Keywords in Input Data File (mfix.dat) 32](#_Toc397681747)

[10.1 The symbols used in the table are as follows: 32](#_Toc397681748)

[10.2 The data type symbols are as follows: 32](#_Toc397681749)

[10.3 Category Name 32](#_Toc397681750)

[10.4 Run Control 32](#_Toc397681751)

[10.5 Physical Parameters 40](#_Toc397681752)

[10.6 Numerical Parameters 43](#_Toc397681753)

[10.7 Geometry And Discretization 47](#_Toc397681754)

[10.8 Cartesian Grid 51](#_Toc397681755)

[10.9 Gas Phase 64](#_Toc397681756)

[10.10 Solids Phase 65](#_Toc397681757)

[10.11 Discrete Element Model (DEM) 67](#_Toc397681758)

[10.12 Cohesion Model in DEM 81](#_Toc397681759)

[10.13 MPPIC model: 85](#_Toc397681760)

[10.14 Discrete Element 88](#_Toc397681761)

[10.15 Initial Condition 99](#_Toc397681762)

[10.16 Boundary Condition 103](#_Toc397681763)

[10.17 Internal Surface 110](#_Toc397681764)

[10.18 Point Source 112](#_Toc397681765)

[10.19 Output Control 114](#_Toc397681766)

[10.20 Chemical Reactions 117](#_Toc397681767)

[10.21 Chemical Reactions – basic options 118](#_Toc397681768)

[10.22 Chemical reactions – advanced options 134](#_Toc397681769)

[10.23 Thermochemical Properties 136](#_Toc397681770)

[10.24 Parallelization Control 139](#_Toc397681771)

[10.25 Batch Queue Environment 140](#_Toc397681772)

[10.26 Direct Quadrature Method Of Moments (Dqmom) 141](#_Toc397681773)

[11 Mailing lists 143](#_Toc397681774)

[12 User contribution 144](#_Toc397681775)

# Introduction

MFIX is an open-source multiphase flow solver and is therefore free to download and use. A one-time free registration is required prior to downloading the source code. To register, go to the MFIX website at <https://mfix.netl.doe.gov>, click on "Register" at the bottom of the home page or go directly to <https://mfix.netl.doe.gov/registration.php>. Complete the form, read the notice, and click on "I Agree" to submit your application. Once your application has been reviewed and accepted, you will receive an email notification, and instructions to download the code.

**MFIX**

**Installation**

(Sections 4.1 to 4.3)

**MFIX**

**Input setup**

(Sections 7)

**Run MFIX**

(Sections 4.5 & 6)

**Post-processing &**

**Visualization**

(Sections 4.3.3 & 4.4)

The above flow chart provides an overview of the work flow and related sections in this document for quick review.

Some Frequently Asked Questions located at <https://mfix.netl.doe.gov/faq.php> may be useful to review before downloading MFIX.

# 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:

**New features:**

* Variable solids density - The material density of a solids phase changes  
    as a result of chemical reactions (changing species composition) while  
    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.  (Jeff/Rahul, edit as needed.)

**Improved Features:**

* DEM serial code improvements.
  + Code profiling was used to identify hot spots within DEM algorithms.

Refactoring the code for better performance gave a 4x speedup compared to the Release 2013-2 (for tutorials/FluidBed\_DES with 38,400 particles.)

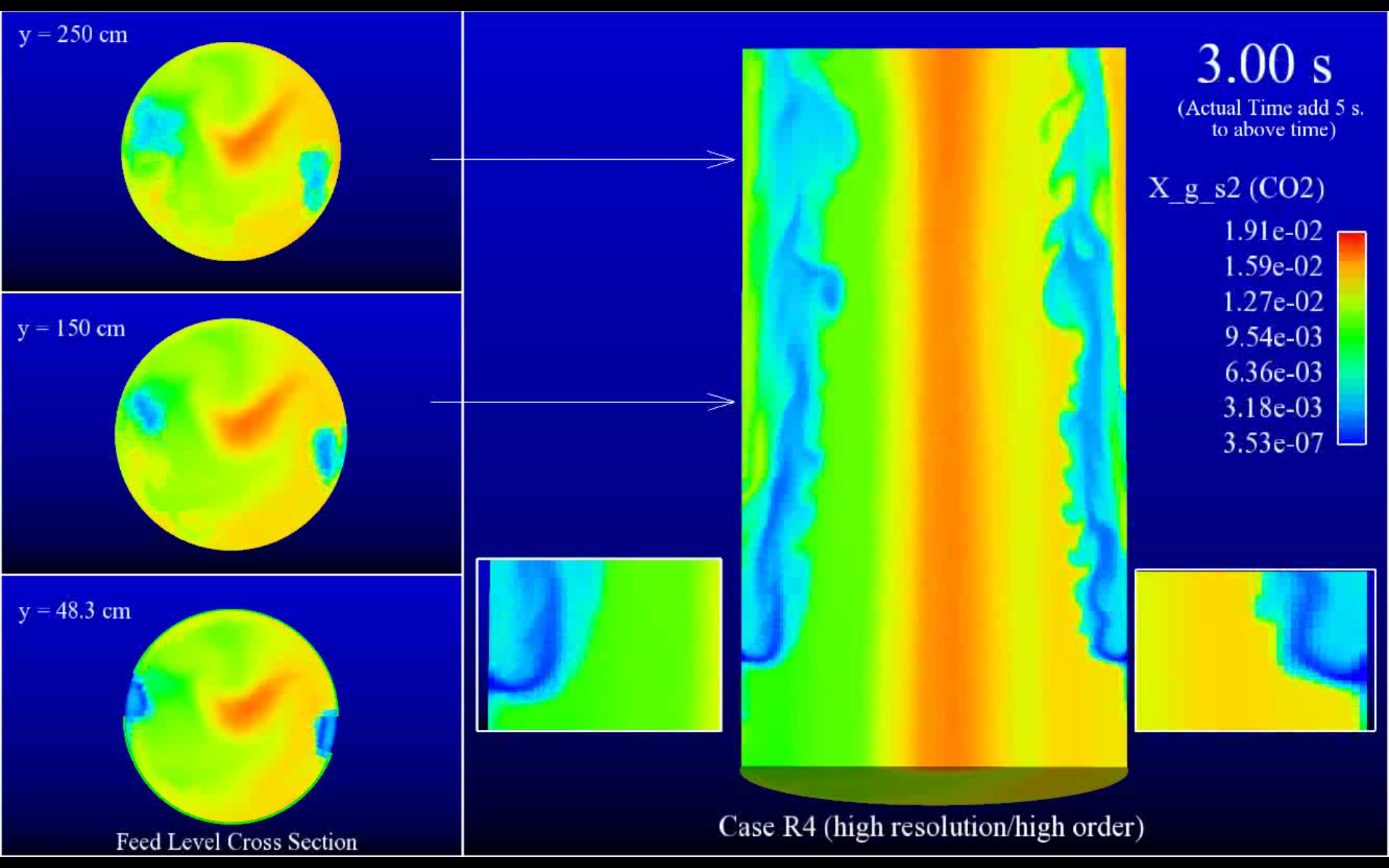
* DEM particle configuration -  
    o Rahul to complete.
* Cartesian grid cut-cell reindexing -  
    o Jeff to complete.
* The make\_mfix build script was completely overhauled.  
    o Build script was split into several files/scripts for simplification.  
    o Build settings can be passed as build arguments.  
    o Parallel-make build support.  
    o Automatic Makefile generation at build time for dependency detection.  
    o Each supported system and compiler has its own settings file allowing  
          for easier modification and maintenance.
* Improved run-time user input checks.  
    o Checks on solids model specifications are restricted to the solids  
           model type so that unnecessary data is no longer needed. This  
           improvement applies to DEM and PIC solids models.  
    o Error messages are consistently managed based on user input settings.  
           If FULL\_LOG is specified, error messages are written the the LOG file  
           and sent to std-out. Otherwise, error messages are only written to  
           the LOG file.

# Development state of MFIX models

MFIX provides a suite of models that treat the carrier phase (typically the gas phase[[1]](#footnote-1)) and disperse phase (typically the solids phase) differently. Their current state of development is summarized in the tables below.

* **Two Fluid Model: MFIX-TFM (Eulerian-Eulerian)**

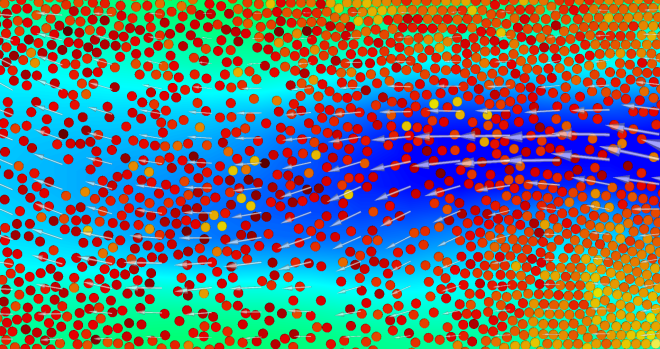
Both gas and solids phases are treated as interpenetrating continuum phases.



|  |  |  |  |
| --- | --- | --- | --- |
|  | Serial | †DMP | ‡SMP |
| Continuity Equations | ● | ● | ● |
| Momentum Equations | ● | ● | ● |
| Energy Equations | ● | ● | ● |
| Species Equations | ● | ● | ● |
| Chemical Reactions | ● | ● |  |
| Cartesian cut-cell | ● | ● | **□** |

* **Discrete Element Method: MFIX-DEM (Eulerian-Lagrangian)**

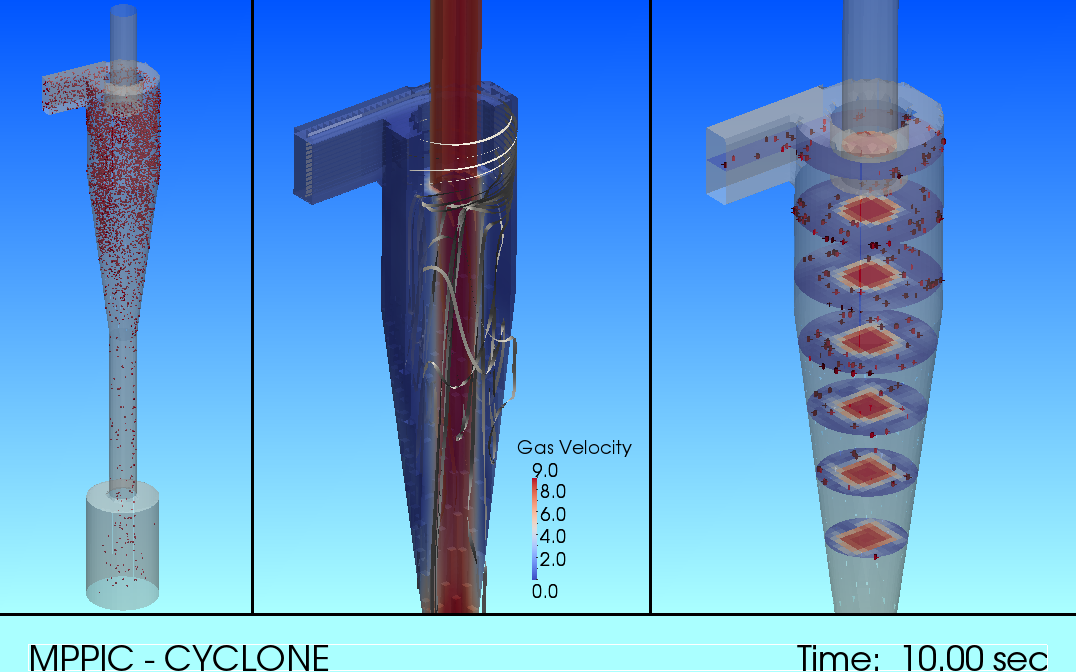
The gas phase is treated as continuum and particles positions are tracked individually. Particle collisions are directly resolved.

****

|  |  |  |  |
| --- | --- | --- | --- |
|  | Serial | †DMP | ‡SMP |
| Momentum Equations | ● | ● | ● |
| Energy Equations | ● |  |  |
| Species Equations | ● |  |  |
| Chemical Reactions | ● |  |  |
| Cartesian cut-cell | ○ | **□** |  |

* **Particle in Cell: MFIX-PIC (Eulerian-Lagrangian)**

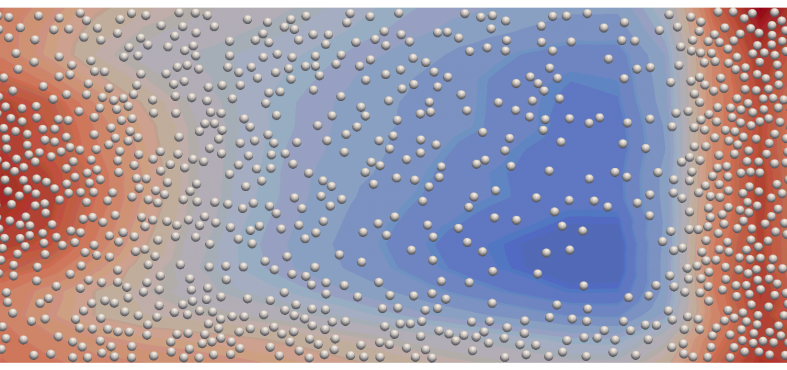
The gas phase is treated as continuum and particles are represented as parcels. Collisions are not directly resolved. Simulations run faster than DEM, with a trade-off between speed and accuracy.



|  |  |  |  |
| --- | --- | --- | --- |
|  | Serial | †DMP | ‡SMP |
| Momentum Equations | ● | ○ | ○ |
| Energy Equations |  |  |  |
| Species Equations |  |  |  |
| Chemical Reactions |  |  |  |
| Cartesian cut-cell | ○ | **□** |  |

* **Hybrid model: MFIX-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.

# Building and Running MFIX on a UNIX/LINUX Workstation

Please read this file carefully and if you still have problems with installation or use, send an email to mfix-help@mfix.netl.doe.gov. Before sending a question, subscribe to mfix-help mailing list at <https://mfix.netl.doe.gov/sympa/info/mfix-help>. Once subscribed, please search the archives at https://mfix.netl.doe.gov/sympa/arc/mfix-help as your question might be already answered.

MFIX is primarily designed to run on Linux operating system. The setup described in this section is the preferred way of installing and running MFIX. You should be familiar with basic Linux operating system commands and procedures before attempting to use MFIX.



## Creating MFIX Directory

It is assumed that you have registered and downloaded MFIX. MFIX is distributed as a source tar ball named **mfix.tar.gz**. To extract MFIX (version 2014-1) from the tar file:

> tar xzf mfix.tar.gz

Extracting the tar ball will create the directory mfix, which contains the following subdirectories:

|  |  |
| --- | --- |
| **Subdirectory** | **Contents** |
| model | MFIX source files |
| post\_mfix | Post mfix source files |
| tutorials | Several example problems and corresponding pdf files describing some of the tutorials exists, which are useful for a new user |
| tests | Test simulations used to verify the code during development |
| Tools | Development tools |
| Doc | MFIX documentation (in pdf format) |

## Alias creation (optional)

For convenience, create aliases by adding the following lines to the *.login* or the .cshrc file (assuming MFIX was installed in the home directory, and you are using the C shell):

> echo “alias make\_mfix ~/mfix/model/make\_mfix” >> ~/.cshrc

> echo “alias make\_post ~/mfix/post\_mfix/make\_post” >> ~/.cshrc

After adding the aliases in the .login or .cshrc file, either logout and login again, or source your .cshrc file for the aliases to take effect. Then MFIX can be built from any directory by with **make\_mfix**, and the post-processor can be built with **make\_post**.

## Building MFIX

A Fortran compiler is needed to build MFIX. Before continuing, check that you have a working Fortran compiler installed on your system. Please contact your system administrator if you do not have access to a compiler.

The distribution of MFIX provides two sets of source codes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Purpose** | **Location of source code** | **Name of executable** |
| **MFIX** | Multiphase flow solver | ~/mfix/model | **mfix.exe** |
| **POSTMFIX** | Simple post processing of the data. This is a text-based program. Data is extracted based on (I,J,K) location in the computational domain. | ~/mfix/post\_mfix | **post\_mfix** |

The open-source visualization tools ParaView and VisIt can be used to visualize and post-process MFIX results. They can be downloaded from:

**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.

### MFIX Build Script

Both MFIX and POSTMFIX are built with a build script, called make\_mfix for MFIX and make\_post for POSTMFIX. Ordinarily these files do not need to be modified, but expert users could edit them to customize build options specific. The next section focuses on make\_mfix, due to the availability of several flags. Invoking make\_post is best described by following the example in section 2.3.3.

**Building with interactive prompts for options**

Run make\_mfix without arguments to be prompted for different build options. (See tutorial in section 4.3.2)

> ~/mfix/model/make\_mfix

or if you created an alias (section 4.2):

> make\_mfix

The following examples assume the alias was created.

**Building with default options**

Use the -default flag to compile with the default options:

- Serial, optimized code

- GNU Fortran compiler

> make\_mfix -default

**Repeating a build**

Use the -repeat flag to rebuild with the same options as the last successful build.

> make\_mfix –repeat

**Showing all compiler options**

Less commonly used compilers are hidden by default. Use the -long flag to display all compiler options.

> make\_mfix -long

**Clean and rebuild everything**

Use the -clean flag to remove all .o, .a and .mod files from the object directory (last compilation) and then do a build

> make\_mfix -clean

**Showing all build options**

Use the -help flag to display all build options.

> make\_mfix -help

**Shortcuts**

Each flag described above has an equivalent short version for convenience:

|  |  |  |
| --- | --- | --- |
| **Description** | **Flag** | **Shortcut** |
| Clean and Rebuild | -clean | -c |
| Default options (serial, optimized, GNU) | -default | -d |
| Display help | -help | -h |
| Show all compiler options | -long | -l |
| Repeat compilation with same options as previous | -repeat | -r |

For example, to compile with default options, the following two commands are equivalent:

> make\_mfix -default

> make\_mfix -d

### Building MFIX: A step-by-step tutorial

Before building MFIX, you must be sure to have a working Fortran compiler installed on your computer. In the following example, it is assumed that the gfortran compiler is installed on a 64-bit computer. MFIX will be compiled and run for a simple fluidbed tutorial case. At this point, it is not important to understand the simulation setup. The executable file mfix.exe will be created for serial execution.

1. Go to the tutorial directory:

> cd ~/mfix/tutorial/fluidbed1

> ls

mfix.dat

1. This folder contains the input file mfix.dat that contains the simulation setup. This file is not required to compile MFIX, but will be used at the end of this example to run the simulation. MFIX can be compiled from any directory except the /mfix/model directory. The directory where MFIX will be run is called the run directory. Here the run directory is **fluidbed1**, and currently only contains the mfix.dat file.
2. Build MFIX

Run the build script.

> make\_mfix

You will be prompted to answer 3 questions. Press Enter to keep the default answers to all 3 questions. This will compile MFIX in optimized serial mode.

==============================================================

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

==============================================================

Level of Optimization:

==============================================================

[0] None (Debug mode)

[x] Level 1 (not available)

[x] Level 2 (not available)

[3] Level 3 (most aggressive)

Select the level of optimization [3] : 🡨 Press Enter

==============================================================

Option to re-compile source files in run directory:

==============================================================

[1] Do not force re-compilation

[2] Force re-compilation

Select Option to re-compile source files in run directory [1] : 🡨 Press Enter

Next, a list of compilers for which the build script is configured is displayed. It does not mean they are actually all installed on your machine. Choose a compiler that you know is properly installed on your machine. We will select gfortran as an example:

64 bit Linux system detected, please select compiler.

==============================================================

MFIX Compilation directives available for following compilers:

==============================================================

[1] GNU (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 will begin and may take several minutes to complete. Once the compilation is successful, the executable file mfix.exe is created and ready to use.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Compilation successful: mfix.2014-1 created

To run MFIX type: mfix.exe

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4) Run MFIX:

./mfix.exe

As the solution proceeds, a lot of information is displayed on the screen. Please see section 4 for a description of the output. The run should complete in a few minutes.

### Building POSTMFIX

You can use POSTMFIX to extract data at a particular location in the flow field. To build POSTMFIX, follow the following procedure. Again, it is assumed that gfortran is installed.

1) Go to the post\_mfix folder:

cd ~/mfix/post\_mfix

2) Run the build script:

> ./make\_post

3) A list of compilers will be displayed, based on your operating system. Again, this list displays the compilers supported by the build script, which are not necessarily installed on your system. In our case, we will select the gfortran compiler. Answering no (n) to the first three questions will select gfortran.

Linux system with 64 bit processor detected, please select compiler

MFIX Compilation directives available for following compilers:

- Intel Fortran Compiler (IFORT - FCE for 64 bit)

- Portland Group Linux Fortran Compiler (pgf90)

- PathScale compiler (pathf90)

- gfortran

Do you want to compile with Intel Compiler? (y/n) [yes] n 🡨 Type n and press Enter

Do you want to compile with Portland Group Compiler? (y/n) [yes] n 🡨 Type n and press Enter

Do you want to compile with PathScale Compiler? (y/n) [yes] n 🡨 Type n and press Enter

gfortran selected

4) Before continuing, delete object files:

Object files (\*.o, \*.mod, \*.a) in mfix/model will be deleted. Continue? (y/n) [no] y 🡨 Type y and press Enter

5) The compilation will start, and upon successful compilation, the following message will be displayed:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Compilation successful: post\_mfix created\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Once post\_mfix is successfully compiled, the alias created in Section 2.2 is operational.

## Starting a New Run

If you successfully built MFIX in Section 2.3.1, you are ready to explore running MFIX with other options for your own simulation.

Create a separate subdirectory for each run, referred to as the run directory.

If you want to modify MFIX subroutines, do the following. Copy the subroutine from the *mfix/model* directory into the run directory. Modify the file copied to the run directory. Do not modify the files in the *mfix/model* directory.



Create an MFIX executable file by typing **make\_mfix**. The MFIX executable file *mfix.exe* will be created and copied into the run-directory.

First, select the type of executable (currently, only serial and DMP options are available). Regardless of the option chosen here, the executable is always named **mfix.exe**

==============================================================

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] :

Type 1 for serial code or 3 for DMP code and press Enter.

Note: To keep the default answer (1 = serial), just press Enter.

If you intend to compile the DMP version of MFIX, please verify that your MPI installation works properly **(see MPI\_verification\_for\_MFIX.pdf document in /mfix/tools/mpi).** Contact your system administrator if you are not sure how to proceed.



Next, select the level of optimization:

==============================================================

Level of Optimization:

==============================================================

[0] None (Debug mode)

[x] Level 1 (not available)

[x] Level 2 (not available)

[3] Level 3 (most aggressive)

Select the level of optimization [3] :

Type 0 and press Enter for executable in debug mode. This option is useful if you modified some source files and want to test the code for possible bugs before production run. Execution time will be slower in debug mode than in optimized mode.

Press Enter to keep the default answer (3). This will generate an optimized executable, which will run faster for production run.

==============================================================

Option to re-compile source files in run directory:

==============================================================

[1] Do not force re-compilation

[2] Force re-compilation

Select Option to re-compile source files in run directory [1] :

Type 2 and press Enter to make sure source files in the run directory are compiled. This option is needed only in the rare instances when the run directory contains a module file (e.g., *run\_mod.f*) and another file (e.g., *usr0.f*), in which the corresponding use statement has been inserted (e.g., use run). The build script will not be aware of this dependency (*usr0* depends upon *run*). So, in this example, changes made in *run\_mod.f* will not cause the (necessary) recompilation of *usr0.f*, unless this option is selected.

Press Enter to keep the default answer (1).

If you keep all the above default options by pressing Enter for each question, an optimized serial version executable will be produced for that particular platform.

Finally, a list of compilers is proposed for your system:

==============================================================

MFIX Compilation directives available for following compilers:

==============================================================

[1] GNU (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]

Type the number corresponding to the compiler you want to use, and press Enter. If you want to use the default option (1 = GNU gfortran), just press Enter.

After selecting the compiler, the compilation process will start. At the end of the compilation process, you should have the executable **mfix.exe** in this directory.

If you have compiled MFIX once, you can run make\_mfix with the same options by invoking the build script with the flag **–repeat** or **–r**. For example

make\_mfix –r

will repeat the compilation process with the last known configuration. This is useful if you compile MFIX many times during your own development phase, and do not wish to repeatedly entering the same build options.



Write an MFIX data file and name it *mfix.dat*. The key words are listed in section 5. Examples are available in the *mfix/tutorials* directory.

You should run several tutorials and carefully study the corresponding mfix.dat before attempting to create your own input file. It is strongly recommended to start with an existing mfix.dat from a tutorial that closely matches your setup, and modify this file.



**Serial Execution**

When MFIX runs, it displays some output on the screen. There are several options to run MFIX with different ways to manage the screen output:

|  |  |
| --- | --- |
| **Command** | **Purpose** |
| ./mfix.exe | Standard way to run MFIX. The output is displayed directly on the screen. |
| ./mfix.exe & | Same as above, with MFIX running in the background. |
| ./mfix.exe > screen.log & | Run in batch mode with the screen output redirected to the file screen.log. To follow the screen output at runtime, type tail –f screen.log**.** Press Ctrl+C to stop following the screen output. (Note that the file screen.log can become very large.). |
| ./mfix.exe |& tee screen.log | Displays the output on the screen and saves it into the screen.log file. **This option is available only for csh or tcshrc shell users.** |

**Note:** you may be able to use **mfix.exe** instead of **./mfix.exe** (i.e. without the dot and slash) on your computer if you have your $PATH environment variable defined properly for current working directory.

**Parallel Execution**

Run the DMP version using **mpirun -np <Number of Processors> mfix.exe** . This command may be different on other machines (e.g., **mpprun -n <Number of Processors> mfix.exe**). Recent versions of MPI library (i.e., MPICH2 on Linux clusters) require mpi daemons running on the compute nodes prior to the launch of any mpi executable. Please make sure the appropriate MPI initialization procedures are followed and the simple MPI examples run successfully to verify MPI setup prior to the launch of MFIX executable in DMP mode.

Note that MFIX results can be retrieved, even while the run is in progress.

To retrieve and manipulate data and to create special restart files, run the post\_processor **post\_mfix**. **post\_mfix** will prompt the user for the run name. Enter the run name (e.g*., BUB01* ).

## Modifying the Post-Processing Codes

To make *post\_mfix*, first change the directory to the *mfix/post\_mfix* directory. If user-defined post-processing is required, modify the *usr\_post.f* file in the directory. Then type **sh make\_post**. Note that the *mfix/model* is required for creating a *post\_mfix* executable.

# Running MFIX on Windows

## Prebuilt Windows binaries for MFIX

This is not very well supported; please see our download page (<https://mfix.netl.doe.gov/members/download.php>) for special remarks (<https://mfix.netl.doe.gov/members/wininst.html>) and also earlier postings on mfix-help mailing list.

## Building MFIX with Cygwin and gfortran

1. Install Cygwin from <http://www.cygwin.com/>.
2. Check that Devel and Editors are installed (To the right of the words will be "default"; Left button click on the word "default" until it says "install")
3. Make sure that make and ex are installed

$ which ex  
/usr/bin/ex  
  
$ which make  
/usr/bin/make

1. Now install gfortran (this is now part of gcc)
   * Check that gfortran is available (by typing gfortran on console, you should get 'gfortran: no input files')
2. Download MFIX from our website, and go to the directory where the MFIX tar ball was saved.
3. issue this command "tar -xzvf mfix.tar.gz"
4. go to tutorials/fluidbed1 by typing "cd mfix/tutorials/fluidbed1"
5. Issue the command "sh ../../model/make\_mfix"
   * This should start the compilation process and depending on the computer this might take a long time.
   * At the end you should have the executable mfix.exe in this directory. To list the files you can use the 'ls' command
6. Now issue the command "nohup ./mfix.exe > out1&" - this will launch the program in the background.
7. You can see the output from the out file by issuing command "tail -f out1". This would stop as soon as the program finished executing.
8. You can visualize the output using ParaView (http://www.paraview.org).
9. Any questions 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:
   1. MFIX version you are trying to install or run
   2. Some details on your operating system environment (for Linux: copy and paste the response of uname –a command, Linux distribution name and version also)
   3. Your compiler name and version number (e.g. ifort –v will give the version number for Intel fortran compiler)
   4. Output for your $PATH environment (in csh type echo $PATH)
   5. 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.

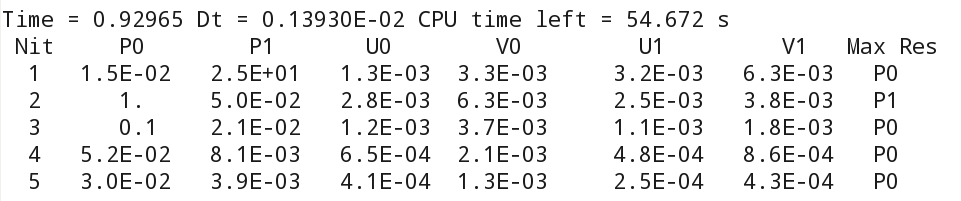
# MFIX at Run Time

## MFIX Output and Messages

MFIX output is stored in nine *\*.SPx* binary files. The restart info is periodically written to the *\*.RES* binary file. The text file *\*.OUT* echoes the input, shows the numerical cell distribution, and, if OUT\_DT is defined, prints the field variables at the specified intervals.

The text file *\*.LOG* contains run information. For the DMP version, a LOG file is created for each of the processors and they are numbered as *Name###.LOG*.

Messages about the run are written to the *.LOG* file. The progress of the run will be displayed at the terminal as shown below if the data file specifies FULL\_LOG = .TRUE.



The first line shows the time, the time-step, and the CPU time remaining to complete the run. The CPU time remaining is not accurate, especially in the beginning of the run. For each time-step, the normalized residuals for various equations are written out every iteration.

MFIX uses a variable time step, which is automatically adjusted within user-defined limits to reduce the run time. At large values of DT, the iterations may not converge. When this happens, the time step size is successively reduced until convergence is obtained. Messages about divergence and recovery are displayed on the terminal and in the *.LOG* file, if FULL\_LOG = .TRUE.

The 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 selected with the keyword RESID\_STRING.

MFIX reports errors while reading the data file, while processing input data, and at run time. Errors in reading the data file and in opening files are reported to the terminal. All other errors are reported in the *.LOG* file.

While reporting errors in reading the data file, MFIX displays the problematic line of input, so that the error can be easily detected. The possible causes of error are (1) incorrect format for the name-list input, (2) unknown (possibly misspelled) variable name, 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 the mass fractions add up to 1.0, whether the overall reaction rates add up to zero, whether the viscosities, conductivities, and specific heats are greater than zero, whether and the temperatures are within the specified limits. 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.

## Restarting a Run

A run is restarted by rerunning MFIX after typing RUN\_TYPE = 'restart\_1' in *mfix.dat*. The old *.OUT* file will be overwritten. The *.LOG* messages will be appended to the old *.LOG* file.

# 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), the convergence can 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.

# Data Visualization with ParaView

It is assumed you have downloaded and installed a suitable binary version of ParaView. To visualize MFIX results, Launch ParaView, and go to File > Open, (or click on the  icon) and select the .RES file in the run directory. For example, to visualize the results from the fluidbed1 tutorial, open the BUB01.RES file. Next, click on the green “Apply” button  on the left pane to load all variables. This step may be optional on your version of ParaView. You will see a contour plot of the void fraction (EP\_g). To show an animation of void fraction, press the play button in the animation toolbar . You will see bubbles forming along the left boundary and moving upward. This tutorial is using a 2D cylindrical coordinate system, and the left boundary is the axis of symmetry of the system. Only one side was simulated due to symmetry. To visualize the entire system, go to Filter >Alphabetical>Reflect. In the Object inspector, select Plane X, and Center 0, and click Apply. You should see the complete symmetric system now, with bubbles forming in the center.

# User-Defined Subroutines

The user may modify any \*.f or \*.inc file in MFIX. To modify a file, first copy it from the mfix/model directory into the run directory. Modify only this copy in the run directory; do **NOT** modify the original files in mfix/model. Then, invoke the 'sh make\_mfix' command from the run directory. The make\_mfix messages will identify the files from the run directory used to create the MFIX executable. All the (MFIX and non-MFIX) \*.inc files from the run directory will be used to create the MFIX executable. Only MFIX \*.f files from the run directory will be used, however. Non-MFIX \*.f files in the run directory will be ignored. To use new Fortran files, include them in one of the MFIX \*.f files.

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

|  |  |
| --- | --- |
| usr\_rates.f | Chemical reaction rates. |
| transport\_prop.f | Transport properties. |
| physical\_prop.f | Physical properties. |
| scalar\_prop.f | Properties and source terms in scalar transport equations. |
| usr\_rates\_des.f[1] | DES chemical reaction rates. |

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

|  |  |
| --- | --- |
| write\_usr0.f | Called once during the run. Can be used for opening user-defined 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 data file:

|  |  |
| --- | --- |
| 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. |
| usr\_mod.f | 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 usr0.f. |
| usr0\_des.f[1] | A subroutine called before entering the DES time loop. |
| usr1\_des.f[1] | A subroutine called every DEM timestep after calculating DES source terms but before source terms are applied to the particles. |
| usr2\_des.f[1] | A subroutine called every DES timestep after source terms are applied to the particles. |
| usr3\_des.f[1] | A subroutine that is called after completing the DES time loop. |
| usr4\_des.f[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/).

DES User-defined subroutine call structure:

calculate DES  
hydrodynamic and energy source terms

call to user1\_des.f

update DES  
variables

call to user2\_des.f

reacting flow

calculate DES  
reaction source terms

Loop over  
fluid cells (IJK)

call to  
user4\_des.f

loop over particles  
in fluid cell IJK

call to  
user\_rates\_des.f

no

yes

DES Time Loop

Pass off from continuum phase model

call to user0\_des.f

Pass off to continuum phase model

call to user3\_des.f

# 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 |

The data type symbols are as follows:

|  |  |
| --- | --- |
| Type | Description |
| C | Character |
| DP | Double Precision |
| I | Integer |
| L | Logical |

[DEFAULT VALUE ] indicates a default value.

## Run Control

|  |  |  |
| --- | --- | --- |
| **RUN\_NAME** |  | C |
| 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, etcs. | | |
| *Required:* | true | |
| **DESCRIPTION** |  | C |
| Problem description. Limited to 60 characters. | | |
| **UNITS** |  | C |
| Simulation input/output units. | | |
| *Required:* | true | |
| *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** |  | C |
| Type of run. | | |
| *Required:* | true | |
| *NEW* | New run. | |
| *RESTART\_1* | Traditional restart. | |
| *RESTART\_2* | Start a new run with initial conditions from a .res file created from another run. | |
| **TIME** |  | DP |
| Start-time of the run. | | |
| *MAX* | +Inf | |
| *MIN* | 0.0 | |
| **TSTOP** |  | DP |
| Stop-time of the run. | | |
| *MAX* | +Inf | |
| *MIN* | 0.0 | |
| **DT** |  | DP |
| Starting time step. If left undefined, a steady-state calculation is performed. | | |
| *MAX* | +Inf | |
| *MIN* | 0.0 | |
| *Dependent: TIME* | Defined | |
| *Dependent: TSTOP* | Defined | |
| **DT\_MAX** | *[ ONE ]* | DP |
| Maximum time step. | | |
| *MAX* | +Inf | |
| *MIN* | 0.0 | |
| *Dependent: TIME* | Defined | |
| *Dependent: TSTOP* | Defined | |
| **DT\_MIN** | *[ 1.0D-6 ]* | DP |
| Minimum time step. | | |
| *MAX* | +Inf | |
| *MIN* | 0.0 | |
| *Dependent: TIME* | Defined | |
| *Dependent: TSTOP* | Defined | |
| **DT\_FAC** | *[ 0.9D0 ]* | DP |
| Factor for adjusting time step. Must be less than 1. | | |
| *MAX* | 1 | |
| *MIN* | 0.0 | |
| *Dependent: TIME* | Defined | |
| *Dependent: TSTOP* | Defined | |
| **DETECT\_STALL** | *[ .TRUE. ]* | L |
| Na | | |
| *.FALSE.* | Do not reduce time step for stalled iterations. | |
| *.TRUE.* | Reduce time step if the residuals sum does not decrease. | |
| **AUTO\_RESTART** | *[ .FALSE. ]* | L |
| Flag to restart the code when dt < dt\_min. | | |
| **MODEL\_B** | *[ .FALSE. ]* | L |
| Momentum equations. | | |
| *.FALSE.* | Model A | |
| *.TRUE.* | Model B | |
| **MOMENTUM\_X\_EQ**(Phase) | *[ .TRUE. ]* | L |
| Solve x-momentum equations. | | |
| *.TRUE.* | Solve x-momentum equations. | |
| *.FALSE.* | Do not solve x-momentum equations. | |
| **MOMENTUM\_Y\_EQ**(Phase) | *[ .TRUE. ]* | L |
| Solve y-momentum equations. | | |
| *.TRUE.* | Solve y-momentum equations. | |
| *.FALSE.* | Do not solve y-momentum equations. | |
| **MOMENTUM\_Z\_EQ**(Phase) | *[ .TRUE. ]* | L |
| Solve z-momentum equations. | | |
| *.TRUE.* | Solve z-momentum equations. | |
| *.FALSE.* | Do not solve z-momentum equations. | |
| **ENERGY\_EQ** | *[ .TRUE. ]* | L |
| Solve energy equations. | | |
| *.FALSE.* | Do not solve energy equations. | |
| *.TRUE.* | Solve energy equations. | |
| **SPECIES\_EQ**(Phase) | *[ .TRUE. ]* | L |
| Solve species transport equations. | | |
| *.FALSE.* | Solve species equations. | |
| *.TRUE.* | Do not solve species equations. | |
| **GRANULAR\_ENERGY** | *[ .FALSE. ]* | L |
| Granular energy formulation selection. | | |
| *.FALSE.* | Use algebraic granular energy equation formulation. | |
| *.TRUE.* | Use granular energy transport equation (pde) formulation. | |
| **KT\_TYPE** | *[ "LUN\_1984" ]* | C |
| Solids phase stress model. | | |
| *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 | |
| *Dependent: GRANULAR\_ENERGY* | .true. | |
| **JENKINS** | *[ .FALSE. ]* | L |
| Jenkins small frictional boundary condition. | | |
| *.FALSE.* | Na | |
| *.TRUE.* | Use the jenkins small fricational boundary condition. | |
| *Dependent: GRANULAR\_ENERGY* | .true. | |
| *Dependent: PHI\_W* | Defined | |
| **FRICTION** | *[ .FALSE. ]* | L |
| Solids stress model selection. | | |
| *.FALSE.* | Use the schaeffer solids stress model. | |
| *.TRUE.* | Use The Princeton Solids Stress Model | |
| *Dependent: GRANULAR\_ENERGY* | .true. | |
| *Dependent: PHI* | Defined | |
| *Dependent: PHI\_W* | Defined | |

The combination of the keywords GRANULAR\_ENERGY and FRICTION invokes different solids stress models as shown below:

GRANULAR\_ENERGY = .FALSE.

EP\_g < EP\_star > Schaeffer

EP\_g >= EP\_star > viscous (algebraic)

GRANULAR\_ENERGY = .TRUE.

FRICTION = .TRUE.

EP\_s(IJK,M) > EPS\_f\_min > Princeton + viscous (pde)

EP\_s(IJK,M) < EPS\_f\_min > viscous (pde)

FRICTION = .FALSE.

EP\_g < EP\_star > Schaeffer + viscous (pde)

EP\_g >= EP\_star > viscous (pde)

|  |  |  |
| --- | --- | --- |
| **SAVAGE**(Phase) | *[ 1 ]* | I |
| For a term appearing in the frictional stress model invoked with friction = .true. <arg index="1" id="phase" min="0" max="dim\_m"/> | | |
| *0* | Use s:s in the frictional stress model. | |
| *2* | An appropriate combination of the above two forms. | |
| *1* | Use an alternate form suggested by savage. | |
| *Dependent: FRICTION* | .true. | |
| **SCHAEFFER** | *[ .TRUE. ]* | L |
| Schaeffer frictional stress tensor formulation. | | |
| *.TRUE.* | Use The Schaeffer Model | |
| *.FALSE.* | Do not use the schaeffer model. | |
| *Dependent: PHI* | Defined | |
| **BLENDING\_STRESS** | *[ .FALSE. ]* | L |
| Blend the schaeffer stresses with that of kinetic theory around ep\*. | | |
| **TANH\_BLEND** | *[ .TRUE. ]* | L |
| Hyperbolic tangent function for blending frictional stress models. | | |
| *Dependent: BLENDING\_STRESS* | .true. | |
| *Conflict: SIGM\_BLEND* | .true. | |
| **SIGM\_BLEND** | *[ .FALSE. ]* | L |
| A scaled and truncated sigmoidal function for blending frictional stress models. | | |
| *Dependent: BLENDING\_STRESS* | .true. | |
| *Conflict: TANH\_BLEND* | .true. | |
| **YU\_STANDISH** | *[ .FALSE. ]* | L |
| Correlation to compute maximum packing for polydisperse systems. | | |
| *.TRUE.* | Use the yu and standish correlation. | |
| *.FALSE.* | Do not use the yu and standish correlation. | |
| **FEDORS\_LANDEL** | *[ .FALSE. ]* | L |
| Use fedors and landel correlation to compute maximum packing for a binary (only) mixture of powders. | | |
| **CALL\_USR** | *[ .FALSE. ]* | L |
| Call user-defined subroutines. | | |
| *.TRUE.* | Call user-defined subroutines. | |
| *.FALSE.* | Do not call user-defined subroutines. | |
| **NRR** | *[ 0 ]* | I |
| The number of user defined chemical reactions stored in the \*.spa file. See section 4.10 chemical reactions. | | |
| **K\_EPSILON** | *[ .FALSE. ]* | L |
| When activated the k-epsilon turbulence model (for single -phase flow) is solved using standard wall functions. | | |
| *Conflict: L\_SCALE0* | Defined | |
| **DRAG\_TYPE** | *[ 'SYAM\_OBRIEN' ]* | C |
| Drag Model | | |
| *SYAM\_OBRIEN* | Na | |
| *GIDASPOW* | Na | |
| *GIDASPW\_PCF* | Na | |
| *GIDASPOW\_BLEND* | Na | |
| *GIDASPOW\_BLEND\_PCF* | Na | |
| *WEN\_YU* | Na | |
| *WEN\_YU\_PCF* | Na | |
| *KOCH\_HILL* | Na | |
| *KOCH\_HILL\_PCF* | Na | |
| *BVK* | Na | |
| *HYS* | Na | |
| *USER\_DRAG* | Na | |
| **RDF\_TYPE** | *[ 'LEBOWITZ' ]* | C |
| Radial distribution function at contact for polydisperse systems. | | |
| *LEBOWITZ* | Na | |
| *MODIFIED\_LEBOWITZ* | Na | |
| *MANSOORI* | Na | |
| *MODIFIED\_MANSOORI* | Na | |
| **ADDED\_MASS** | *[ .FALSE. ]* | L |
| When activated the added (or virtual) mass force effectively acts to increase the inertia of the dispersed phase, which tends to stabilize simulations of bubbly gas-liquid flows. | | |
| **SUBGRID\_TYPE** |  | C |
| Subgrid Models Include: Igci And Milioli | | |
| *IGCI* | Na | |
| *MILIOLI* | Na | |
| **FILTER\_SIZE\_RATIO** | *[ 2.0D0 ]* | DP |
| Ratio of filter size to computational cell size. | | |
| **SUBGRID\_WALL** | *[ .FALSE. ]* | L |
| Flag for subgrid wall effects. | | |
| *.FALSE.* | Do not include wall effects. | |
| *.TRUE.* | Include subgrid wall effects. | |
| **REPORT\_NEG\_DENSITY** | *[ .FALSE. ]* | L |
| Provide detailed logging of negative density errors. | | |
| *.FALSE.* | Do not log negative density errors. | |
| *.TRUE.* | Log negative density errors. | |
| **NSCALAR** | *[ 0 ]* | I |
| Number of user-defined scalar transport equations to solve. | | |
| *MAX* | Dim\_Scalar | |
| *MIN* | 0 | |
| **PHASE4SCALAR**(Scalar Equation) |  | I |
| The phase convecting the indexed scalar transport equation. | | |
| *MAX* | Dim\_M | |
| *MIN* | 0 | |
| **DEF\_COR** | *[ .FALSE. ]* | L |
| Use deferred correction method for implementing higher order discretization. | | |
| *.FALSE.* | Use down-wind factor method (default). | |
| *.TRUE.* | Use deferred correction method for implementing higher order discretization. | |

### Restart

|  |  |  |
| --- | --- | --- |
| **AUTOMATIC\_RESTART** | *[ .FALSE. ]* | L |
| Variable which triggers an automatic restart. | | |
| **ITER\_RESTART** | *[ 1 ]* | I |
| Atuo\_restart counter. | | |

## Physical Parameters

|  |  |  |
| --- | --- | --- |
| **C** |  | DP |
| User defined constants. | | |
| **C\_NAME** | *[ '....................' ]* | C |
| Name of user-defined constant. (20 character max) | | |
| **C\_E** |  | DP |
| Coefficient of restitution for particle-particle collisions. | | |
| **R\_P**(Phase, Phase) |  | DP |
| Ghd theory: coefficient of restitution for particle-particle collisions. | | |
| **E\_W** | *[ 1.D0 ]* | DP |
| Coefficient of restitution for particle-wall collisions. | | |
| **PHIP** | *[ 0.6D0 ]* | DP |
| Specularity coefficient associated with particle-wall collisions. | | |
| **PHIP\_OUT\_JJ** | *[ .FALSE. ]* | L |
| 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. | | |
| **PHIP0** |  | DP |
| 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. | | |
| **C\_F** |  | DP |
| Coefficient of friction between the particles of two solids phases. | | |
| **PHI** |  | DP |
| Angle of internal friction (in degrees). Set this value to zero to turn off plastic regime stress calculations. | | |
| **PHI\_W** |  | DP |
| 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. | | |
| **EPS\_F\_MIN** | *[ 0.5D0 ]* | DP |
| Minimum solids fraction above which friction sets in. (when friction = .true.) | | |
| **EP\_S\_MAX**(Phase) |  | DP |
| Maximum solids volume fraction at packing for polydisperse systems (more than one solids phase used). 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. | | |
| *MAX* | 1-Ep\_Star | |
| *MIN* | 0 | |
| **SEGREGATION\_SLOPE\_COEFFICIENT** | *[ 0.D0 ]* | DP |
| Used in calculating the initial slope of segregation: see gera et al. (2004) - recommended value 0.3. Increasing this coefficient results in decrease in segregation of particles in binary mixtures. | | |
| **L\_SCALE0** | *[ ZERO ]* | DP |
| Value of turbulent length initialized. This may be overwritten in specific regions with the keyword ic\_l\_scale. | | |
| *Conflict: K\_EPSILON* | .true. | |
| **MU\_GMAX** |  | DP |
| Maximum value of the turbulent viscosity of the fluid. | | |
| **V\_EX** | *[ ZERO ]* | DP |
| Excluded volume in boyle-massoudi stress. | | |
| *0.0* | B-m stress is turned off. | |
| **P\_REF** | *[ ZERO ]* | DP |
| Reference pressure. | | |
| **P\_SCALE** | *[ ONE ]* | DP |
| Scale factor for pressure. | | |
| **GRAVITY** |  | DP |
| Gravitational acceleration. | | |
| **GRAVITY\_X** | *[ ZERO ]* | DP |
| X-component of gravitational acceleration vector. By default, the gravity force acts in the negative y-direction. | | |
| **GRAVITY\_Y** | *[ ZERO ]* | DP |
| Y-component of gravitational acceleration vector. By default, the gravity force acts in the negative y-direction. | | |
| **GRAVITY\_Z** | *[ ZERO ]* | DP |
| Z-component of gravitational acceleration vector. By default, the gravity force acts in the negative y-direction. | | |
| **DRAG\_C1** | *[ 0.8D0 ]* | DP |
| Quantity for calibrating syamlal-o'brien drag correlation using umf data. This are determined using the umf spreadsheet. | | |
| **DRAG\_D1** | *[ 2.65D0 ]* | DP |
| Quantity for calibrating syamlal-o'brien drag correlation using umf data. This are determined using the umf spreadsheet. | | |
| **LAM\_HYS** |  | DP |
| If use\_def\_lam\_hys is set to .false. The user is able to specify a value for the lubrication cutoff distance (lam\_hys). 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). | | |
| **M\_AM** |  | I |
| Disperse phase number where the added mass applies. | | |

## Numerical Parameters

|  |  |  |
| --- | --- | --- |
| **MAX\_NIT** | *[ 500 ]* | I |
| Maximum number of iterations. | | |
| **NORM\_G** |  | DP |
| Factor to normalize the gas continuity equation residual. | | |
| **NORM\_S** |  | DP |
| Factor to normalize the solids continuity equation residual. | | |
| **TOL\_RESID** | *[ 1.0D-3 ]* | DP |
| Maximum residual at convergence (continuity+momentum). | | |
| **TOL\_RESID\_TH** | *[ 1.0D-4 ]* | DP |
| Maximum residual at convergence (granular energy). | | |
| **TOL\_RESID\_T** | *[ 1.0D-4 ]* | DP |
| Maximum residual at convergence (energy). | | |
| **TOL\_RESID\_X** | *[ 1.0D-4 ]* | DP |
| Maximum residual at convergence (species balance). | | |
| **TOL\_RESID\_SCALAR** | *[ 1.0D-4 ]* | DP |
| Maximum residual at convergence (scalar balances.) | | |
| **TOL\_RESID\_K\_EPSILON** | *[ 1.0D-4 ]* | DP |
| Maximum Residual At Convergence (K\_Epsilon Model) | | |
| **TOL\_DIVERGE** | *[ 1.0D+4 ]* | DP |
| Minimum residual for declaring divergence. This parameter is useful for incompressible fluid simulations because velocity residuals can take large values for the second iteration (e.g., 1e+8) before drop down to smaller values for third third iteration (e.g., 0.1). | | |
| **MAX\_INLET\_VEL\_FAC** | *[ ONE ]* | DP |
| 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. | | |

The next keywords LEQ\_IT, LEQ\_METHOD, LEQ\_SWEEP, LEQ\_TOL, UR\_FAC, and DISCRETIZE are dimensioned for the nine types of equations:

|  |  |
| --- | --- |
| Index | Equation Type |
| 1 | gas pressure |
| 2 | solids volume fraction |
| 3 | gas and solids u-momentum |
| 4 | gas and solids v-momentum |
| 5 | gas and solids w-momentum |
| 6 | Temperature |
| 7 | species mass fractions |
| 8 | granular temperature |
| 9 | user-defined scalar |

For example, LEQ\_IT(3) = 10 will make MFIX use 10 linear equation iterations while solving the gas and solids u-momentum equation (Equation Type =3).

|  |  |  |
| --- | --- | --- |
| **LEQ\_IT**(Equation ID Number) | *[ 20 ]* | I |
| Number of iterations in the linear equation solver. | | |
| **LEQ\_METHOD**(Equation ID Number) | *[ 2 ]* | I |
| Leq solver selection. | | |
| *1* | Sor | |
| *2* | Bicgstab | |
| *3* | Gmres | |
| *5* | Cg | |
| **LEQ\_SWEEP**(Equation ID Number) | *[ 'RSRS' ]* | C |
| Linear equation sweep direction. | | |
| *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\_TOL**(Equation ID Number) | *[ 1.0D-4 ]* | DP |
| Linear equation tolerance. | | |
| *Dependent: LEQ\_METHOD* | 3 | |
| **LEQ\_PC**(Equation ID Number) | *[ 'LINE' ]* | C |
| Linear precondition used for leq solver sweeps. | | |
| *NONE* | No Preconditioner | |
| *LINE* | Line Relaxation | |
| *DIAG* | Diagonal Scaling | |
| **UR\_FAC**(Equation ID Number) | *[ 0.8D0 ]* | DP |
| Under relaxation factors. | | |
| **DISCRETIZE** | *[ 0 ]* | I |
| Discretization scheme of equations. | | |
| *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. | |
| *0* | First-order upwinding. | |
| **FPFOI** | *[ .FALSE. ]* | L |
| Four point fourth order interpolation and is upstream biased. If this scheme is chosen and discretize(\*) < 2, discretize(\*) is defaulted to 2. If you chose this scheme, set the c\_fac value between 0 and 1. | | |
| *Dependent: C\_FAC* | Defined | |
| **C\_FAC** |  | DP |
| Factor used in the universal limiter (when fpfoi is set .true.) and can be any value in the set (0,1). The choice of 1 will give (diffusion) first order upwinding and as this value becomes closer to 0 the scheme becomes more compressive. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| *Dependent: FPFOI* | .true. | |
| **CN\_ON** | *[ .FALSE. ]* | L |
| Temporal discretization scheme. | | |
| *.FALSE.* | Implicit euler based temporal discretization scheme employed (first order accurate in time). | |
| *.TRUE.* | Crank-nicholson based temporal discretization scheme employed (second order accurate in time excluding the restart timestep which is first order). | |
| **CHI\_SCHEME** | *[ .FALSE. ]* | L |
| Chi-scheme, proposed by darwish and moukalled (2003), is activated. This scheme guarantees that the set of differenced species mass balance equations has the property that the sum of mass fractions add up to one. When a flux limiter is used with (higher order) spatial discretization schemes it is not guaranteed that the mass fractions add up to one. This problem may be rectified by activating the chi-scheme. | | |
| **UR\_F\_GS** | *[ 1.0D0 ]* | DP |
| The Implicitness Calculation Of The Gas-Solids Drag Coefficient May Be Underrelaxed By Changing Ur\_F\_Gs, Which Takes Values Between 0 To 1: | | |
| *MAX* | 1 | |
| *MIN* | 0 | |
| **UR\_KTH\_SML** | *[ 1.0D0 ]* | DP |
| Unknown | | |
| **DO\_TRANSPOSE** | *[ .FALSE. ]* | L |
| Solve transpose of linear system. (bicgstab only). | | |
| *Dependent: LEQ\_METHOD* | 2 | |
| **ICHECK\_BICGS** | *[ 1 ]* | I |
| Frequency to check for convergence. (bicgstab only) | | |
| *Dependent: LEQ\_METHOD* | 2 | |
| **OPT\_PARALLEL** | *[ .FALSE. ]* | L |
| Sets optimal leq flags for parallel runs. | | |
| **USE\_DOLOOP** | *[ .FALSE. ]* | L |
| Use do-loop assignment over direct vector assignment. | | |
| **IS\_SERIAL** | *[ .TRUE. ]* | L |
| Calculate dot-products more efficiently (serial runs only.) | | |

## 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.

|  |  |  |
| --- | --- | --- |
| **COORDINATES** |  | C |
| Coordinates used in the simulation. | | |
| *CARTESIAN* | Cartesian coordinates. | |
| *CYLINDRICAL* | Cylindrical coordinates. | |
| **NO\_I** | *[ .FALSE. ]* | L |
| (do not use.) | | |
| *.FALSE.* | X (r) direction is considered. | |
| *.TRUE.* | X (r) direction is not considered. | |
| **IMAX** |  | I |
| Number of cells in the x (r) direction. | | |
| **DX**(Cell) |  | DP |
| 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** | *[ ZERO ]* | DP |
| The inner radius in the simulation of an annular cylindrical region. | | |
| **XLENGTH** |  | DP |
| Reactor length in the x (r) direction. | | |
| **NO\_J** | *[ .FALSE. ]* | L |
| (do not use.) | | |
| *.FALSE* | Y direction is considered. | |
| *.TRUE.* | Y direction is not considered. | |
| **JMAX** |  | I |
| Number of cells in the y direction. | | |
| **DY**(Cell) |  | DP |
| 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** |  | DP |
| Reactor length in the y direction. | | |
| **NO\_K** | *[ .FALSE. ]* | L |
| Na | | |
| *.FALSE.* | Z(theta) direction is considered. | |
| *.TRUE.* | Z(theta) direction is not considered. | |
| **KMAX** |  | I |
| Number of cells in the z (() direction. | | |
| **DZ**(Cell) |  | DP |
| 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** |  | DP |
| Reactor length in the z (theta) direction. | | |
| **CYCLIC\_X** | *[ .FALSE. ]* | L |
| 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** | *[ .FALSE. ]* | L |
| 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** |  | DP |
| Fluid pressure drop across xlength when a cyclic boundary condition with pressure drop is imposed in the x-direction. | | |
| **CYCLIC\_Y** | *[ .FALSE. ]* | L |
| 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** | *[ .FALSE. ]* | L |
| 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** |  | DP |
| Fluid pressure drop across ylength when a cyclic boundary condition with pressure drop is imposed in the y-direction. | | |
| **CYCLIC\_Z** | *[ .FALSE. ]* | L |
| 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** | *[ .FALSE. ]* | L |
| 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** |  | DP |
| Fluid pressure drop across zlength when a cyclic boundary condition with pressure drop is imposed in the z-direction. | | |
| **SHEAR** | *[ .FALSE. ]* | L |
| If .true. Imposes a mean shear on the flow field as a linear function of \_x\_ coordinate. This feature should only be used when cyclic\_x=.true. Also, the keyword v-sh needs to be set. | | |
| **V\_SH** | *[ 0D0 ]* | DP |
| 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** |  | DP |
| 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. | | |

### Cartesian Grid

A new capability, called 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** | *[ .FALSE. ]* | L |
| 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: | |
| *Dependent: COORDINATES* | Cartesian | |
| *Conflict: COORDINATES* | Cylindrical | |
| **N\_QUADRIC** | *[ 0 ]* | I |
| Number of quadric surfaces defining the boundaries (<=100). | | |
| *MAX* | 100 | |
| *MIN* | 0 | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **USE\_STL** | *[ .FALSE. ]* | L |
| Use stl file to describe geometry. | | |
| *.FALSE.* | Do not use stl file. | |
| *.TRUE.* | Read triangulated geometry (for 3d geometry only) from geometry.stl. | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **USE\_MSH** | *[ .FALSE. ]* | L |
| Use .msh file to describe geometry. | | |
| *.FALSE.* | Do not use .msh file. | |
| *.TRUE.* | Read geometry (for 3d geometry only) from geometry.msh. | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **USE\_POLYGON** | *[ .FALSE. ]* | L |
| Use polygons to describe geometry. | | |
| *.FALSE.* | Do not use polygons. | |
| *.TRUE.* | Read polygon data (for 2d geometry only) from poly.dat. | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **N\_USR\_DEF** | *[ 0 ]* | I |
| 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 | |
| *MAX* | 1 | |
| *MIN* | 0 | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **QUADRIC\_FORM**(Quadric ID) | *[ 'NORMAL' ]* | C |
| Form of the quadric surface equation. | | |
| *NORMAL* | Use normal form, as defined in equation (1). The lamdba | |
| *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. | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **QUADRIC\_SCALE** | *[ ONE ]* | DP |
| Scaling factor, applied to all quadric geometry parameters. Must be a positive number | | |
| *MIN* | 0.0 | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **LAMBDA\_X**(Quadric ID) | *[ ZERO ]* | DP |
| Coefficient lambda\_x in equation (1) ('normal' form) or x-component of normal vector defining plane in equation (5) ('degenerate' form). | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **LAMBDA\_Y**(Quadric ID) | *[ ZERO ]* | DP |
| Coefficient lambda\_y in equation (1) ('normal' form) or y-component of normal vector defining plane in equation (5) ('degenerate' form). | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **LAMBDA\_Z**(Quadric ID) | *[ ZERO ]* | DP |
| Coefficient lambda\_z in equation (1) ('normal' form) or z-component of normal vector defining plane in equation (5) ('degenerate' form). | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **DQUADRIC**(Quadric ID) | *[ ZERO ]* | DP |
| Coefficient d in equation (1). | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **THETA\_X**(Quadric ID) | *[ ZERO ]* | DP |
| Rotation angle with respect to x-axis (degrees). | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **THETA\_Y**(Quadric ID) | *[ ZERO ]* | DP |
| Rotation angle with respect to y-axis (degrees). | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **THETA\_Z**(Quadric ID) | *[ ZERO ]* | DP |
| Rotation angle with respect to z-axis (degrees). | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **RADIUS**(Quadric ID) | *[ ZERO ]* | DP |
| Cylinder Radius (Used When Quadric\_Form = \*\_Cyl\_\*\*\*) | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **HALF\_ANGLE**(Quadric ID) | *[ ZERO ]* | DP |
| Cone Half Angle, Expressed In Degrees (Used When Quadric\_Form = \*\_Cone) | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **N\_X**(Quadric ID) | *[ ZERO ]* | DP |
| X-Component Of Normal Vector Defining The Plane (Used When Quadric\_Form = Plane) | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **N\_Y**(Quadric ID) | *[ ZERO ]* | DP |
| Y-Component Of Normal Vector Defining The Plane (Used When Quadric\_Form = Plane) | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **N\_Z**(Quadric ID) | *[ ZERO ]* | DP |
| Z-Component Of Normal Vector Defining The Plane (Used When Quadric\_Form = Plane) | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **T\_X**(Quadric ID) | *[ ZERO ]* | DP |
| Translation in x-direction. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **T\_Y**(Quadric ID) | *[ ZERO ]* | DP |
| Translation in y-direction. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **T\_Z**(Quadric ID) | *[ ZERO ]* | DP |
| Translation in z-direction. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **CLIP\_XMIN**(Quadric ID) | *[ -LARGE\_NUMBER ]* | DP |
| Lower x-limit where the quadric is defined. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **CLIP\_XMAX**(Quadric ID) | *[ LARGE\_NUMBER ]* | DP |
| Upper x-limit where the quadric is defined. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **CLIP\_YMIN**(Quadric ID) | *[ -LARGE\_NUMBER ]* | DP |
| Lower y-limit where the quadric is defined. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **CLIP\_YMAX**(Quadric ID) | *[ LARGE\_NUMBER ]* | DP |
| Upper y-limit where the quadric is defined. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **CLIP\_ZMIN**(Quadric ID) | *[ -LARGE\_NUMBER ]* | DP |
| Lower z-limit where the quadric is defined. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **CLIP\_ZMAX**(Quadric ID) | *[ LARGE\_NUMBER ]* | DP |
| Upper z-limit where the quadric is defined. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **PIECE\_XMIN**(Quadric ID) | *[ -LARGE\_NUMBER ]* | DP |
| Lower x-limit where the quadric is defined in a piecewise group. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **PIECE\_XMAX**(Quadric ID) | *[ LARGE\_NUMBER ]* | DP |
| Upper z-limit where the quadric is defined in a piecewise group. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **PIECE\_YMIN**(Quadric ID) | *[ -LARGE\_NUMBER ]* | DP |
| Lower y-limit where the quadric is defined in a piecewise group. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **PIECE\_YMAX**(Quadric ID) | *[ LARGE\_NUMBER ]* | DP |
| Upper y-limit where the quadric is defined in a piecewise group. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **PIECE\_ZMIN**(Quadric ID) | *[ -LARGE\_NUMBER ]* | DP |
| Lower z-limit where the quadric is defined in a piecewise group. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **PIECE\_ZMAX**(Quadric ID) | *[ LARGE\_NUMBER ]* | DP |
| Upper z-limit where the quadric is defined in a piecewise group. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **FLUID\_IN\_CLIPPED\_REGION**(Quadric ID) | *[ .TRUE. ]* | L |
| 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. | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **BC\_ID\_Q**(Quadric ID) |  | I |
| Boundary Condition Flag | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **N\_GROUP** | *[ 1 ]* | I |
| Number of group(s) of quadrics (<=50). | | |
| **GROUP\_SIZE** | *[ 1 ]* | I |
| Number of quadrics in the group. | | |
| **GROUP\_Q** | *[ 0 ]* | I |
| Na | | |
| **GROUP\_RELATION** | *[ 'OR' ]* | C |
| 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** | *[ 'OR' ]* | C |
| 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** | *[ 0.00D0 ]* | DP |
| 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. | | |
| *MAX* | 0.5 | |
| *MIN* | 0.0 | |
| **TOL\_DELH** | *[ 0.00D0 ]* | DP |
| 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** | *[ 0.01D0 ]* | DP |
| Tolerance used to detect small cells (expressed as a fraction of cell volume, between 0.0 and 1.0). | | |
| **TOL\_MERGE** | *[ 1.0D-12 ]* | C |
| Tolerance used to remove duplicate nodes (expressed as a fraction of cell diagonal, between 0.0 and 1.0). | | |
| **TOL\_SMALL\_AREA** | *[ 0.01D0 ]* | DP |
| Tolerance used to detect small faces (expressed as a fraction of original face area, between 0.0 and 1.0). | | |
| **ALPHA\_MAX** | *[ ONE ]* | DP |
| Maximum acceptable value of interpolation correction factor. | | |
| **TOL\_F** | *[ 1.0D-9 ]* | DP |
| Tolerance used to find intersection of quadric surfaces or user-defined function with background grid. | | |
| **TOL\_POLY** | *[ 1.0D-9 ]* | DP |
| Tolerance used to find intersection of polygon with background grid. | | |
| **ITERMAX\_INT** | *[ 10000 ]* | I |
| Maximum number of iterations used to find intersection points. | | |
| **TOL\_STL** | *[ 1.0D-6 ]* | DP |
| Tolerance used to find intersection of stl triangles with background grid. | | |
| **STL\_SMALL\_ANGLE** | *[ 5.0 ]* | DP |
| Smallest angle accepted for valid stl triangles (in degrees). Triangles having an angle smaller that this value will be ignored. | | |
| **TOL\_STL\_DP** | *[ 1.0D-3 ]* | DP |
| Dot product tolerance when determining if a point lies in a facet. | | |
| **DIM\_FACETS\_PER\_CELL** | *[ 10 ]* | I |
| Maximum number of stl facets per cell. | | |
| **MAX\_FACETS\_PER\_CELL\_DES** | *[ 24 ]* | I |
| Maximum number of stl facets per cell for des data arrays. | | |
| **OUT\_STL\_VALUE** | *[ 1.0 ]* | DP |
| 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** |  | I |
| Boundary Condition Flag For The Stl Geometry | | |
| **TX\_STL** | *[ ZERO ]* | DP |
| Translation in x-direction, applied to the stl geometry. | | |
| **TY\_STL** | *[ ZERO ]* | DP |
| Translation in y-direction, applied to the stl geometry. | | |
| **TZ\_STL** | *[ ZERO ]* | DP |
| Translation in z-direction, applied to the stl geometry. | | |
| **SCALE\_STL** | *[ ONE ]* | DP |
| Scaling factor, applied to the stl geometry. Note that translation occurs after scaling. | | |
| **TOL\_MSH** | *[ 1.0D-6 ]* | DP |
| Tolerance used to find intersection of .msh file with background grid. | | |
| **OUT\_MSH\_VALUE** | *[ 1.0 ]* | DP |
| 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** | *[ ZERO ]* | DP |
| Translation in x-direction, applied to the .msh geometry. | | |
| **TY\_MSH** | *[ ZERO ]* | DP |
| Translation in y-direction, applied to the .msh geometry. | | |
| **TZ\_MSH** | *[ ZERO ]* | DP |
| Translation in z-direction, applied to the .msh geometry. | | |
| **SCALE\_MSH** | *[ ONE ]* | DP |
| Scaling factor, applied to the .msh geometry. Note that translation occurs after scaling. | | |
| **CAD\_PROPAGATE\_ORDER** | *[ '' ]* | C |
| 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** | *[ 'X-' ]* | C |
| Na | | |
| **SET\_CORNER\_CELLS** | *[ .FALSE. ]* | L |
| 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** | *[ 0.25 ]* | DP |
| Factor Used To Allocate Some Cut Cell Arrays (Expressed As A Fraction Of Dimension\_3G) | | |
| **WRITE\_VTK\_FILES** | *[ .FALSE. ]* | L |
| 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. | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **TIME\_DEPENDENT\_FILENAME** | *[ .TRUE. ]* | L |
| 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** |  | DP |
| Interval (expressed in seconds of simulation time) at which vtk files are written. | | |
| **FRAME** | *[ -1 ]* | I |
| Na | | |
| **VTU\_DIR** | *[ '.' ]* | C |
| Directory Where Vtk Files Are Stored (Default Is Run Directory) | | |
| **PG\_OPTION** | *[ 0 ]* | I |
| 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** | *[ 0 ]* | I |
| 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** | *[ .FALSE. ]* | L |
| Prints any warning message encountered during pre-processing on the screen. | | |
| **CG\_UR\_FAC** | *[ 1.0 ]* | DP |
| Under-relaxation factor used in cut cells (only cg\_ur\_fac(2) is used). | | |
| **PRINT\_PROGRESS\_BAR** | *[ .FALSE. ]* | L |
| Print a progress bar during each major step of pre-processing stage. | | |
| **BAR\_WIDTH** | *[ 50 ]* | I |
| Width of the progress bar (complete status), expressed in number of characters (between 10 and 80). | | |
| *MAX* | 80 | |
| *MIN* | 10 | |
| **BAR\_CHAR** | *[ ' ]* | C |
| Character used to create the progress bar. | | |
| **BAR\_RESOLUTION** | *[ 5.0 ]* | DP |
| Update frequency of progress bar, expressed in percent of total length (between 1.0 and 100.0). | | |
| *MAX* | 100.0 | |
| *MIN* | 1.0 | |
| **WRITE\_DASHBOARD** | *[ .FALSE. ]* | L |
| Writes the file dashboard.txt at regular intervals. The file shows a summary of the simulation progress. | | |
| **F\_DASHBOARD** | *[ 1 ]* | I |
| Frequency, expressed in terms of iterations, at which the dashboard is updated. | | |
| **CPX** | *[ ZERO ]* | DP |
| Location of control points in x-direction. | | |
| **NCX** | *[ 0 ]* | I |
| Number of cells within a segment (x-direction). | | |
| **ERX** | *[ ONE ]* | DP |
| Expansion ratio (last dx/first dx) in a segment (x-direction). | | |
| **FIRST\_DX** | *[ ZERO ]* | DP |
| Value of first dx in a segment (x-direction). A negative / ro\_s(m) ) over all solids phases is equal to ( 1.0 - ic\_ep\_g(ic)). E value will copy dx from previous segment (if available). | | |
| **LAST\_DX** | *[ ZERO ]* | DP |
| Value of last dx in a segment (x-direction). A negative value will copy dx from next segment (if available). | | |
| **CPY** | *[ ZERO ]* | DP |
| Location of control points in y-direction. | | |
| **NCY** | *[ 0 ]* | I |
| Number of cells within a segment (y-direction). | | |
| **ERY** | *[ ONE ]* | DP |
| Expansion ratio (last dy/first dy) in a segment (y-direction). | | |
| **FIRST\_DY** | *[ ZERO ]* | DP |
| Value of first dy in a segment (y-direction). A negative value will copy dy from previous segment (if available). | | |
| **LAST\_DY** | *[ ZERO ]* | DP |
| Value of last dy in a segment (y-direction). A negative value will copy dy from next segment (if available). | | |
| **CPZ** | *[ ZERO ]* | DP |
| Location of control points in z-direction. | | |
| **NCZ** | *[ 0 ]* | I |
| Number of cells within a segment (z-direction). | | |
| **ERZ** | *[ ONE ]* | DP |
| Expansion ratio (last dz/first dz) in a segment (z-direction). | | |
| **FIRST\_DZ** | *[ ZERO ]* | DP |
| Value of first dz in a segment (z-direction). A negative value will copy dz from previous segment (if available). | | |
| **LAST\_DZ** | *[ ZERO ]* | DP |
| Value of last dz in a segment (z-direction). A negative value will copy dz from next segment (if available). | | |
| **RE\_INDEXING** | *[ .FALSE. ]* | L |
| Turns on the re-indexing of cells. When true, inactive (dead) cells are removed from computational domain. | | |
| **ADJUST\_PROC\_DOMAIN\_SIZE** | *[ .FALSE. ]* | L |
| Attemps to adjust grid partition. Each processor will be assigned its own size to minimize load imbalance | | |
| **REPORT\_BEST\_DOMAIN\_SIZE** | *[ .FALSE. ]* | L |
| Attemps to adjust grid partition. Each processor will be assigned its own size to minimize load imbalance | | |
| **NODESI\_REPORT** | *[ 1 ]* | I |
| Temporary Setting Used In Serial Run To Report Best Domain Size For Parallel Run | | |
| **NODESJ\_REPORT** | *[ 1 ]* | I |
| Temporary Setting Used In Serial Run To Report Best Domain Size For Parallel Run | | |
| **NODESK\_REPORT** | *[ 1 ]* | I |
| Temporary Setting Used In Serial Run To Report Best Domain Size For Parallel Run | | |
| **MINIMIZE\_SEND\_RECV** | *[ .TRUE. ]* | L |
| Attemps To Minimize The Size Of The Send/Receive Layers | | |

## Gas Phase

|  |  |  |
| --- | --- | --- |
| **RO\_G0** |  | DP |
| Specified constant gas density. This 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** |  | DP |
| Specified constant gas viscosity. | | |
| **K\_G0** |  | DP |
| Specified constant gas conductivity. | | |
| **C\_PG0** |  | DP |
| Specified constant gas specific heat. | | |
| **DIF\_G0** |  | DP |
| Specified constant gas diffusivity. | | |
| **MW\_AVG** |  | DP |
| Average molecular weight of gas. | | |
| **MW\_G**(Species) |  | DP |
| Molecular weight of gas species n. | | |
| **NMAX\_G** |  | I |
| Number of species comprising the gas phase. | | |
| **SPECIES\_G** |  | C |
| Name of gas phase species n as it appears in the materials database. | | |
| **SPECIES\_ALIAS\_G** |  | C |
| User defined name for gas phase species n. | | |

## Solids Phase

|  |  |  |
| --- | --- | --- |
| **MMAX** | *[ 1 ]* | I |
| Number of solids phases. | | |
| **D\_P0**(Phase) |  | DP |
| Initial particle diameters. | | |
| **RO\_S0**(Phase) |  | DP |
| Specified constant solids density. | | |
| **X\_S0**(Phase, Species) |  | DP |
| Baseline species mass fraction. | | |
| *Dependent: SPECIES\_EQ* | .true. | |
| *Dependent: RO\_XS0* | Defined | |
| *Dependent: INERT\_SPECIES* | Defined | |
| *Conflict: RO\_S0* | Defined | |
| **RO\_XS0**(Phase, Species) |  | DP |
| Specified constant solids species density. | | |
| *Dependent: SPECIES\_EQ* | .true. | |
| *Dependent: X\_S0* | Defined | |
| *Dependent: INERT\_SPECIES* | Defined | |
| *Conflict: RO\_S0* | Defined | |
| **INERT\_SPECIES**(Phase, Species) |  | I |
| Index of inert solids phase species. | | |
| *Dependent: SPECIES\_EQ* | .true. | |
| *Dependent: X\_S0* | Defined | |
| *Dependent: RO\_XS0* | Defined | |
| *Conflict: RO\_S0* | Defined | |
| **MU\_S0** |  | DP |
| 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. | | |
| **K\_S0**(Phase) |  | DP |
| Specified constant solids conductivity. | | |
| **C\_PS0**(Phase) |  | DP |
| Specified constant solids specific heat. | | |
| **DIF\_S0** |  | DP |
| Specified constant solids diffusivity. | | |
| **MW\_S**(Phase, Species) |  | DP |
| Molecular weight of solids phase-m, species n. | | |
| **NMAX\_S**(Phase) |  | I |
| Number of species comprising solids phase m. | | |
| **SPECIES\_S**(Phase, Species) |  | C |
| Name of solids phase m, species n as it appears in the materials database. | | |
| **SPECIES\_ALIAS\_S**(Phase, Species) |  | C |
| User Defined Name For Solids Phase M, Species N | | |
| **EP\_STAR** |  | DP |
| Packed bed void fraction. | | |
| **CLOSE\_PACKED**(Phase) | *[ .TRUE. ]* | L |
| Indicates that the solids phase forms a packed bed with a void fraction ep\_star. | | |
| **SOLIDS\_MODEL**(Phase) | *[ 'TFM' ]* | C |
| 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. | | |
| *TFM* | Na | |
| *DEM* | Na | |
| *PIC* | Na | |

### Discrete Element Model (DEM)

The MFIX-DEM main documentation file is titled "Documentation of open-source MFIX–DEM software for gas-solids flows" by R. Garg, J. Galvin, T. Li, and S. Pannala. It is available online at <https://mfix.netl.doe.gov/documentation/dem_doc_2012-1.pdf>, or in /mfix/doc directory.

The indicated document serves as the user guide for MFIX-DEM and it includes pointers to the DEM code and a discussion on the associated theory.

|  |  |  |
| --- | --- | --- |
| **PARTICLES** |  | I |
| Total number of particles to be read in from the user provided particle configuration file. Only valid when gener\_part\_config is set to false. When gener\_part\_config is true, then particle count is automatically calculated by mfix. For an inflow case beginning with no solids inventory, particles can be specified as 0. | | |
| *MAX* | +Inf | |
| *MIN* | 0 | |
| **MN** | *[ 10 ]* | I |
| Maximum number of neighbors per particle. Relevant only for dem model. | | |
| *MAX* | +Inf | |
| *MIN* | 0 | |
| **NFACTOR** | *[ 10 ]* | I |
| Only needed if des\_continuum\_coupled is true and dem model is used. Number of times a pure granular simulation is run before the coupled dem simulation is started. It is used to obtain an initial settled configuration. | | |
| *MAX* | +Inf | |
| *MIN* | 0 | |
| **DES\_ONEWAY\_COUPLED** | *[ .FALSE. ]* | L |
| 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. | | |
| **PARTICLES\_FACTOR** | *[ 1.2D0 ]* | DP |
| 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. | | |
| *MAX* | +Inf | |
| *MIN* | 1.0 | |
| *Dependent: PARTICLES* | Defined | |
| **USE\_STL\_DES** | *[ .FALSE. ]* | L |
| 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. | | |
| **DISCRETE\_ELEMENT** | *[ .FALSE. ]* | L |
| Use discrete particle model for solids. Must be true to invoke dem, mppic, hybrid models. Default is dem model, to turn on mppic model, set mppic to true as well. | | |
| **DES\_CONTINUUM\_COUPLED** | *[ .FALSE. ]* | L |
| To switch between pure granular or coupled simulations of carried and dispersed phase flows. | | |
| *.TRUE.* | Performs coupled simulations. | |
| **DES\_INTERP\_ON** | *[ .FALSE. ]* | L |
| Use an interpolation suite to calculate the drag force on each particle based on particle location rather than cell averages. | | |
| **DES\_INTERP\_MEAN\_FIELDS** | *[ .FALSE. ]* | L |
| 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\_REPORT\_MASS\_INTERP** | *[ .FALSE. ]* | L |
| Reports mass based on lagrangian particles and continuum representation. Useful to ensure mass conservation between lagrangian and continuum representations. Recommended use for debugging purposes. | | |
| *Dependent: DES\_INTERP\_MEAN\_FIELDS* | .true. | |
| **DES\_INTG\_METHOD** | *[ 'EULER' ]* | C |
| Time stepping scheme (relevant to dem model only). Mppic is only limited to euler scheme. | | |
| *EULER* | First-order euler scheme. | |
| *ADAMS BASHFORTH* | Second Order Adams Bashforth Scheme | |
| **USE\_COHESION** | *[ .FALSE. ]* | L |
| Switch to turn cohesion model (limited to dem model only) on and off. | | |
| *Conflict: MPPIC* | .true. | |
| **VAN\_DER\_WAALS** | *[ .FALSE. ]* | L |
| Flag to turn on the use hamaker van der waals forces. | | |
| *Dependent: USE\_COHESION* | .true. | |
| **DES\_CONTINUUM\_HYBRID** | *[ .FALSE. ]* | L |
| Invoke the hybrid scheme. | | |
| **DES\_NEIGHBOR\_SEARCH** | *[ 4 ]* | I |
| Flag to set the neighbor search algorithm relevant to dem model only. | | |
| *1* | N-Square Search Algorithm (Most Expensive) | |
| *2-4* | Grid-Based Neighbor Search (Recommended) | |
| **USE\_VDH\_DEM\_MODEL** | *[ .FALSE. ]* | L |
| Flag to use van der hoef et al. (2006) model for adjusting the rotation of the contact plane. See the mfix-dem documentation. | | |
| **NEIGHBOR\_SEARCH\_N** | *[ 25 ]* | I |
| Maximum number of steps through a dem loop before a neighbor search will be performed. (search may be called earlier based on other logic). | | |
| *MAX* | +Inf | |
| *MIN* | 0.0 | |
| **NEIGHBOR\_SEARCH\_RAD\_RATIO** | *[ 1.0D0 ]* | DP |
| Ratio of the distance (imaginary sphere radius) to particle radius that is allowed before a neighbor search is performed. This works in conjunction with the logic imposed by neighbor\_search\_n in deciding calls to the neighbor search algorithm. | | |
| **FACTOR\_RLM** | *[ 1.2 ]* | DP |
| Effectively increase the radius of a particle (multiple of the sum of particle radii) during the building of particle neighbor list. Relevant to dem model only. | | |
| **DESGRIDSEARCH\_IMAX** |  | I |
| Number of des grid cells in the i- direction. If left undefined, then it is set by mfix such that its size equals three times the maximum particle diameter with a minimum of 1 cell. | | |
| **DESGRIDSEARCH\_JMAX** |  | I |
| Number of des grid cells in the j- direction. If left undefined, then it is set by mfix such that its size equals three times the maximum particle diameter with a minimum of 1 cell. | | |
| **DESGRIDSEARCH\_KMAX** |  | I |
| Number of des grid cells in the k- direction. If left undefined, then it is set by mfix such that its size equals three times the maximum particle diameter with a minimum of 1 cell. | | |
| **DES\_COLL\_MODEL** | *[ 'LSD' ]* | C |
| Collision model for the soft-sphere approach used in dem model. All models require specifying the following parameters: des\_en\_input, des\_en\_wall\_input, mew, and mew\_w. | | |
| *LSD* | The linear spring-dashpot model. Requires additional initialization for kn, kn\_w, kt\_fac, kt w\_fac, des\_etat\_fac, & des\_etat\_w\_fac. | |
| *HERTZIAN* | The hertzian model. Requires additional initialization for des\_et\_input, des\_et\_wall\_input, e\_young, ew\_young, v\_poisson, & vw\_poisson. | |
| **KN** |  | DP |
| Normal spring constant for inter-particle collisions needed when using the default (linear spring-dashpot) collision model. | | |
| **KT\_FAC** | *[ 2.D0/7.D0 ]* | C |
| Ratio of the tangential spring constant to normal spring constant for inter-particle collisions. Use it to specify the tangential spring constant for particle-particle collisions as kt\_fac\*kn. Required when using the default (linear spring-dashpot) collision model. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| *Dependent: DES\_COLL\_MODEL* | Lsd | |
| **KN\_W** |  | DP |
| Normal spring constant for particle-wall collisions. Needed when using the default (linear spring-dashpot) collision model. | | |
| **KT\_W\_FAC** | *[ 2.D0/7.D0 ]* | C |
| Ratio of the tangential spring constant to normal spring constant for particle-wall collisions. Use it to specify the tangential spring constant for particle-wall collisions as kt\_w\_fac\*kn\_w. Needed when using the default (linear spring-dashpot) collision model. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| *Dependent: DES\_COLL\_MODEL* | Lsd | |
| **MEW** |  | DP |
| Inter-particle columb friction coefficient required for dem model. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| **MEW\_W** |  | DP |
| Particle-wall friction coefficient required for dem model. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| **DES\_EN\_INPUT** |  | DP |
| The normal restitution coefficient for interparticle collisions that is used to determine the inter-particle normal damping factor. Values are stored as a one dimensional array (see mfix-dem doc). So if max=3, then 6 values are needed, which are defined as follows: en11 en12 en13 en22 en23 en33. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| **DES\_EN\_WALL\_INPUT** |  | DP |
| Normal restitution coefficient for particle wall collisions that is used to determine the particle-wall normal damping factor (see cfassign.f for details). Values are stored as a one dimensional array. So, if mmax=3, then 3 values are needed, which are defined as follows: enw1 enw2 enw3. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| **DES\_ET\_INPUT** |  | DP |
| Tangential restitution coefficient for interparticle collisions. Values are stored as a one dimensional array. Only needed when using the hertzian collision model. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| *Dependent: DES\_COLL\_MODEL* | Hertzian | |
| **DES\_ET\_WALL\_INPUT** |  | DP |
| Tangential restitution coefficient for particle wall collisions. Values are stored as a one dimensional array. Only needed when using the hertzian collision model. | | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| *Dependent: DES\_COLL\_MODEL* | Hertzian | |
| **DES\_ETAT\_FAC** |  | DP |
| Ratio of the tangential damping factor to the normal damping factor for inter-particle collisions. Required for the linear spring-dashpot model for soft-spring collision modelling under dem. For the hertzian model, the tangential damping coefficients have to be explicity specified and this variable is not required. | | |
| *UNDEFINED* | For lsd model, if left undefined, mfix will will revert to default value of 0.5 | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| *Dependent: DES\_COLL\_MODEL* | Lsd | |
| **DES\_ETAT\_W\_FAC** |  | DP |
| Ratio of the tangential damping factor to the normal damping factor for particle-wall collisions. Required for the linear spring-dashpot model for soft-spring collision modelling under dem. For the hertzian model, the tangential damping coefficients have to be explicity specified and specification of this variable is not required. | | |
| *UNDEFINED* | For lsd model, if left undefined, mfix will will revert to default value of 0.5 | |
| *MAX* | 1.0 | |
| *MIN* | 0.0 | |
| *Dependent: DES\_COLL\_MODEL* | Lsd | |
| **EW\_YOUNG** |  | DP |
| Youngs modulus for the wall. Needed when using the hertzian spring-dashpot model for soft-spring collision modelling under dem. | | |
| *Dependent: DES\_COLL\_MODEL* | Hertzian | |
| **VW\_POISSON** |  | DP |
| Poisson ratio for the wall. Needed when using the hertzian spring-dashpot model for soft-spring collision modelling under dem. | | |
| *Dependent: DES\_COLL\_MODEL* | Hertzian | |
| **E\_YOUNG**(Phase) |  | DP |
| Youngs modulus for the particle. Needed when using the hertzian spring-dashpot model for soft-spring collision modelling under dem. | | |
| *Dependent: DES\_COLL\_MODEL* | Hertzian | |
| **V\_POISSON**(Phase) |  | DP |
| Poissons ratio for the particle. Needed when using the hertzian spring-dashpot model for soft-spring collision modelling under dem. | | |
| *Dependent: DES\_COLL\_MODEL* | Hertzian | |
| **DES\_D\_P0**(Phase) |  | DP |
| Particle Diameter Associated With Discrete Solids Phase M | | |
| **DES\_RO\_S**(Phase) |  | DP |
| Particle Density Associated With Discrete Solids Phase M | | |
| **DES\_MMAX** |  | I |
| Number of solid phases with discrete representation. When des\_continuum\_hybrid is false, no need to set this variable as it is set equal to mmax. Only relevant when des\_continuum\_hybrid is true so as to differentiate between discrete and continuum representation of solids phase | | |
| *Dependent: DES\_CONTINUUM\_HYBRID* | .true. | |
| **DES\_LE\_BC** | *[ .FALSE. ]* | L |
| Lees-edwards boundary condition to simulate homogeneous shear problem with periodic boundary conditions. Not supported in this version. | | |
| **DES\_LE\_REL\_VEL** |  | DP |
| Relative velocity needed for lees-edwards bc. Not supported in this version. | | |
| *Dependent: DES\_LE\_BC* | .true. | |
| **DES\_LE\_SHEAR\_DIR** |  | C |
| Direction of shear for lees-edwards bc. Not supported in this version. | | |
| *Dependent: DES\_LE\_BC* | .true. | |
| **MAX\_PIS** |  | I |
| Maximum number of particles that may exist within a simulation. This quantity is used for calculating the size of arrays for allocation. | | |
| **HAMAKER\_CONSTANT** |  | DP |
| Hamaker Constant Used For Particle-Particle Interactions | | |
| *Dependent: USE\_COHESION* | .true. | |
| **WALL\_HAMAKER\_CONSTANT** |  | DP |
| Hamaker constant used in particle-wall interactions. | | |
| *Dependent: USE\_COHESION* | .true. | |
| **VDW\_OUTER\_CUTOFF** |  | DP |
| Maximum separation distance above which van der waals forces are not implemented. | | |
| *Dependent: USE\_COHESION* | .true. | |
| **VDW\_INNER\_CUTOFF** |  | DP |
| Minimum separation distance below which van der waals forces are calculated using a surface adhesion model. | | |
| *Dependent: USE\_COHESION* | .true. | |
| **WALL\_VDW\_OUTER\_CUTOFF** | *[ ZERO ]* | DP |
| Maximum separation distance above which van der waals forces are not implemented (particle-wall interactions). | | |
| *Dependent: USE\_COHESION* | .true. | |
| **WALL\_VDW\_INNER\_CUTOFF** |  | DP |
| Minimum separation distance below which van der waals forces are calculated using a surface adhesion model (particle-wall interactions). | | |
| *Dependent: USE\_COHESION* | .true. | |
| **ASPERITIES** | *[ ZERO ]* | DP |
| Mean radius of surface asperities that influence the cohesive force following a model by rumpf (1990). | | |
| *Dependent: USE\_COHESION* | .true. | |
| **PRINT\_DES\_DATA** | *[ .FALSE. ]* | L |
| Allows writing of discrete particle data to output files. Relevant to both granular and coupled simulations. | | |
| **DES\_SPX\_DT** | *[ LARGE\_NUMBER ]* | DP |
| If print\_des\_data, this is the frequency at which particle data is written out. This only applies to pure granular simulations. For coupled simulation, the output frequency is controlled by spx\_dt(1). | | |
| *Dependent: PRINT\_DES\_DATA* | .true. | |
| *Dependent: DES\_CONTINUUM\_COUPLED* | .false. | |
| *Conflict: DES\_CONTINUUM\_COUPLED* | .true. | |
| *Conflict: MPPIC* | .true. | |
| **DES\_RES\_DT** | *[ LARGE\_NUMBER ]* | DP |
| This is the frequency at which des.res and .res files will be written. This only applies to pure granular simulations. For coupled simulation the restart frequency is controlled by res\_dt. | | |
| *Dependent: DES\_CONTINUUM\_COUPLED* | .false. | |
| *Conflict: DES\_CONTINUUM\_COUPLED* | .true. | |
| *Conflict: MPPIC* | .true. | |
| **DES\_OUTPUT\_TYPE** |  | C |
| If undefined the default vtp files are written. | | |
| *TECPLOT* | If tecplot is specified then several .dat files are written including des\_data.dat & avg\_eps.dat. See write\_des\_data.f for details. | |
| **DEBUG\_DES** | *[ .FALSE. ]* | L |
| Print out additional information from dem model. | | |
| **FOCUS\_PARTICLE** | *[ 0 ]* | I |
| Specify particle number for particle level debugging details. | | |
| **DES\_CALC\_CLUSTER** | *[ .FALSE. ]* | L |
| Flag to turn on on-the-fly cluster data calculations. | | |
| *Conflict: MPPIC* | .true. | |
| **CLUSTER\_LENGTH\_CUTOFF** |  | DP |
| The physical distance relative to a particle for peforming cluster statistics calculations. | | |
| *Dependent: DES\_CALC\_CLUSTER* | .true. | |
| *Conflict: MPPIC* | .true. | |
| **GENER\_PART\_CONFIG** | *[ .FALSE. ]* | L |
| Automatically generate an initial particle configuration (position), otherwise use particle input.dat now uses the conventional flag used by continuum solver. Once defined, this feature will determine the total number of particles in the system and their initial placement. The particle\_input.dat file (if present in the run directory) is ignored. | | |
| **MPPIC\_SOLID\_STRESS\_SNIDER** | *[ .FALSE. ]* | L |
| Turn on snider's version of frictional model. Does not run very stably. | | |
| **MPPIC\_COEFF\_EN1** |  | DP |
| First coefficient of restitution for the frictional stress model in the mppic model. See the mppic documentation for more details. | | |
| *Dependent: MPPIC* | .true. | |
| **MPPIC\_COEFF\_EN2** |  | DP |
| Second coefficient of restitution for the frictional stress model in the mppic model. See the mppic documentation for more details. | | |
| *Dependent: MPPIC* | .true. | |
| **MPPIC\_COEFF\_EN\_WALL** |  | DP |
| Normal coefficient of restitution for parcel-wall collisions in the mppic model. | | |
| *Dependent: MPPIC* | .true. | |
| **MPPIC\_COEFF\_ET\_WALL** | *[ 1.0 ]* | DP |
| Tangential coefficient of restitution for parcel-wall collisions in the mppic model. Currently not implemented in the code. | | |
| *Dependent: MPPIC* | .true. | |
| **MPPIC\_PDRAG\_IMPLICIT** | *[ .FALSE. ]* | L |
| Turn on the implicit treatment for interphase drag force. Valid only for mppic model.. | | |
| *Dependent: MPPIC* | .true. | |
| **MPPIC\_GRAV\_TREATMENT** | *[ .TRUE. ]* | L |
| Variable to decide if special treatment is needed or not in the direction of gravity in the frictional stress tensor valid only for mppic model. See the mppic documenation. | | |
| *Dependent: MPPIC* | .true. | |
| **PIC\_REPORT\_SEEDING\_STATS** | *[ .FALSE. ]* | L |
| Flag to print processor level parcel seeding statistics for inflow bc with pic model. | | |
| *Dependent: MPPIC* | .true. | |
| **PIC\_REPORT\_DELETION\_STATS** | *[ .FALSE. ]* | L |
| Flag to print processor level parcel deletion statistics for outflow bc with pic model. Not recommended for production runs. | | |
| *Dependent: MPPIC* | .true. | |
| **PIC\_REPORT\_MIN\_EPG** | *[ .FALSE. ]* | L |
| 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. | | |
| *Dependent: MPPIC* | .true. | |
| **PSFAC\_FRIC\_PIC** | *[ 100 ]* | I |
| P\_S Term In The Frictional Stress Model Of Snider | | |
| *Dependent: MPPIC* | .true. | |
| **FRIC\_EXP\_PIC** | *[ 2.5 ]* | DP |
| Beta Term In The Frictional Stress Model Of Snider | | |
| *Dependent: MPPIC* | .true. | |
| **FRIC\_NON\_SING\_FAC** | *[ 1E-07 ]* | C |
| Non-Singularity Term (Epsilon) In The Frictional Stress Model Of Snider | | |
| *Dependent: MPPIC* | .true. | |
| **CFL\_PIC** | *[ 0.1 ]* | DP |
| Cfl number used to decide maximum time step size for parcels evolution equations. Relevant to mppic model only. | | |
| *Dependent: MPPIC* | .true. | |
| **DES\_CONV\_CORR** | *[ 'RANZ\_1952' ]* | C |
| Specify the nusselt number correlation used for particle-gas convection. (only ranz\_1952 is presently included.) | | |
| **DES\_MIN\_COND\_DIST** |  | DP |
| Minimum separation distance between the surfaces of two contacting particles. | | |
| **FLPC** | *[ 1.0D0/5.0D0 ]* | C |
| Fluid lens proportion constant used to calculate the radius of the fluid lens that surrounds a particle. Used in the particle-fluid-particle conduction model. | | |
| **DES\_EM**(Phase) |  | DP |
| Emissivity of solids phase m. | | |

### Cohesion Model in DEM

Cohesive interparticle forces represent an addition to the discrete-particle simulation. The raw code for this addition was located in the *mfix/model/cohesion* directory. These forces can be implemented using both a square-well model and a Hamaker van der Waals model. Cohesive forces are turned-on by setting the logical variable “USE\_COHESION” equal to true in the mfix.dat file.

**Square-Well model**

The nature of the square-well model is determined by the well width (*router*) and the well depth (*D*). Both of these parameters are specified in the *mfix.dat* file. Furthermore, the simulation is set up to use a separate set of square-well parameters with particle-particle interactions and particle-wall interactions. Generally, the well depth is set to be twice as large for particle-wall interactions to be consistent with the increased surface contact that would exist in such interactions.

Within the square well model, cohesive interactions are implemented as instantaneous momentum impulses (*J\**) resulting in a change in particle velocity as shown below:

 (1)

where *m* is the particle mass, is the pre-interaction velocity, is the post-interaction velocity and is the unit vector connecting the particle centers. This treatment leads to three types of cohesive interactions in the square-well model: approaching cohesive interactions, escaping cohesive interactions and capture cohesive interactions. The momentum impulse during each of these interactions can be calculated by solving the momentum balance along with the stipulation that the total kinetic energy of the two-particle system changes by the depth of the square well (*D*). This treatment leads to the following expressions for the momentum impulse in each type of interaction:

 (2)

 (3)

 (4)

Because cohesive interactions are implemented only once on approach and once as the particles depart, the identification of a square-well interaction cannot be done simply using the separation distance. Special book-keeping is employed to ensure that square-well cohesive interactions are implemented only during the time step when the well-widths first become overlapped for approaching interactions and when they first become un-overlapped for departing interactions. This book-keeping involves recording a “list of links” for each particle. Two particles are considered “linked” is their separation distance is less than the width of their square-wells (the square-wells overlap). The list of links is updated at every cohesive interaction. Cohesive square-well interactions are recognized using both the list of links and the particle separation distance at each. For example, an approaching cohesive interaction is only implemented if two particles overlap their square-wells, yet they are not on their respective list of links. The particles will be added to each list during the interaction and at the next time step, they may still have overlapped square-wells, but an interaction will not be implemented because they particles are already on their respective list of links. A similar algorithm is used to identify departing interactions.

**Hamaker model**

The Hamaker model for van der Waals forces predicts the cohesive force between equal-sized, spherical particles according to the following equation [1]:

 (5)

where  is the particle radius; *A* is the Hamaker constant, which is specific to a given material and has typical values on the order of 10-20 J; andis the minimum surface-to-surface separation distance between two particles *i* and *j*:

 (6)

The corresponding cohesive force between a spherical particle and a flat wall is [1]:

 (7)

For both of these expressions, the cohesive force approaches infinity as the separation distance approaches zero. This singularity incurred at particle contact is avoided by introducing a “cutoff” distance,. For separation distances below this cutoff distance, the interparticle cohesive force is given by a surface adhesion force () model [2].

 (8)

where  is a constant surface energy per unit area. The contact surface energy is calculated at the beginning of each simulation based on the specified Hamaker constant and cutoff distance to ensure that the force is continuous at the cutoff distance. Thus, the cohesive force is maintained at a constant value for any separation distances (based on equation 6) below the minimum cutoff distance, which includes any “negative” separation distances that occur during actual particle contact.

**Cohesive Interaction Search**

Cohesive interactions are identified using a method separate from the functions used to identify particle contacts in the discrete-particle model. The simulation domain is divided up into search grids in order to reduce the number of particles that must be checked for interactions by each particle. The search grid is made up of several uniform boxes that are defined to be at least as wide as the largest length-scale in the cohesive forces. In this way, particles are only checked against other particles in their box and adjacent boxes. The size of the search boxes ensures that a particle could not have an interaction with particles that are located in boxes outside the adjacent boxes. The arrays recording the indices of the search box for each particle are updated at each time step (*update\_search\_grids.f*). Note that the van der Waals model uses the same search algorithm as for DEM collisions and is fully parallelized unlike the square-well model.

### Particle in Cell (MFIX-PIC)

The MFIX-DEM main documentation file is titled "Documentation of open-source MFIX–PIC software for gas-solids flows" by R. Garg and J. F. Dietiker. It is available online at <https://mfix.netl.doe.gov/documentation/mfix_pic_doc.pdf>, or in /mfix/doc directory.

The indicated document serves as the theory guide for the MPPIC model implementation details in MFIX code. The MFIX code with MPPIC model is referred to as MFIX-PIC in the indicated document and hereinafter. A brief discussion on setting up of input file for MPPIC model assumes reader’s familiarity with the MPPIC documentation.

Note that MPPIC model has not been tested with MPI modules. It has only been tested in serial compilation mode. It will work with cut-cell modules in MFIX, but there needs to be work done to ensure conservative coupling between the two phases.



The MPPIC model is invoked by setting MPPIC flag to TRUE in conjunction with DISCRETE\_ELEMENT set to TRUE. The initial conditions for the MPPIC model are specified in the same way as continuum model. It is noted that this is a departure from the DEM model in MFIX where the initial condition for solid phase was specified by special DEM related flags. Although the physical region where the parcels are seeded is specified by the same flags as those used in continuum model setup, there is still need to specify the number of parcels per cell and their statistical weights. 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. As discussed earlier, 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 are 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.

## 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) |  | DP |
| X coordinate of the west face. | | |
| **IC\_X\_E**(IC) |  | DP |
| X coordinate of the east face. | | |
| **IC\_Y\_S**(IC) |  | DP |
| Y coordinate of the south face. | | |
| **IC\_Y\_N**(IC) |  | DP |
| Y coordinate of the north face. | | |
| **IC\_Z\_B**(IC) |  | DP |
| Z coordinate of the bottom face. | | |
| **IC\_Z\_T**(IC) |  | DP |
| Z coordinate of the top face. | | |
| **IC\_I\_W**(IC) |  | I |
| I index of the west-most wall. | | |
| **IC\_I\_E**(IC) |  | I |
| I index of the east-most wall. | | |
| **IC\_J\_S**(IC) |  | I |
| J index of the south-most wall. | | |
| **IC\_J\_N**(IC) |  | I |
| J index of the north-most wall. | | |
| **IC\_K\_B**(IC) |  | I |
| K index of the bottom-most wall. | | |
| **IC\_K\_T**(IC) |  | I |
| K index of the top-most wall. | | |
| **IC\_TYPE**(IC) |  | C |
| 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) |  | DP |
| Initial void fraction in the ic region. | | |
| **IC\_P\_G**(IC) |  | DP |
| 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) |  | DP |
| Initial solids pressure in the ic region. Usually, this value is specified as zero. | | |
| **IC\_L\_SCALE**(IC) |  | DP |
| Turbulence length scale in the ic region. | | |
| **IC\_ROP\_S**(IC, Phase) |  | DP |
| Initial bulk density (rop\_s = ro\_s x ep\_s) of solids phase-m in the ic region. Users need to specify this ic only for polydisperse flow (mmax > 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) |  | DP |
| Initial solids volume fraction of solids phase-m in the ic region. | | |
| **IC\_T\_G**(IC) |  | DP |
| Initial gas phase temperature in the ic region. | | |
| **IC\_T\_S**(IC, Phase) |  | DP |
| Initial solids phase-m temperature in the ic region. | | |
| **IC\_THETA\_M**(IC, Phase) |  | DP |
| Initial solids phase-m granular temperature in the ic region. | | |
| **IC\_GAMA\_RG**(IC, Phase) | *[ ZERO ]* | DP |
| Gas phase radiation coefficient in the ic region. Modify file rdtn2.inc to change the source term. | | |
| **IC\_T\_RG**(IC, Phase) |  | DP |
| Gas phase radiation temperature in the ic region. | | |
| **IC\_GAMA\_RS**(IC, Phase) | *[ ZERO ]* | DP |
| Solids phase-m radiation coefficient in the ic region. Modify file radtn2.inc to change the source term. | | |
| **IC\_T\_RS**(IC, Phase) |  | DP |
| Solids phase-m radiation temperature in the ic region. | | |
| **IC\_U\_G**(IC) |  | DP |
| Initial x-component of gas velocity in the ic region. | | |
| **IC\_U\_S**(IC, Phase) |  | DP |
| Initial x-component of solids-phase velocity in the ic region. | | |
| **IC\_V\_G**(IC) |  | DP |
| Initial y-component of gas velocity in the ic region. | | |
| **IC\_V\_S**(IC, Phase) |  | DP |
| Initial y-component of solids-phase velocity in the ic region. | | |
| **IC\_W\_G**(IC) |  | DP |
| Initial z-component of gas velocity in the ic region. | | |
| **IC\_W\_S**(IC, Phase) |  | DP |
| Initial z-component of solids-phase velocity in the ic region. | | |
| **IC\_X\_G**(IC, Species) |  | DP |
| Initial mass fraction of gas species n. | | |
| **IC\_X\_S**(IC, Phase, Species) |  | DP |
| Initial mass fraction of gas species n. | | |
| **IC\_SCALAR**(IC, Scalar Eq.) |  | DP |
| Initial value of scalar n. | | |
| **IC\_K\_TURB\_G**(IC) |  | DP |
| Initial value of k in k-epsilon. | | |
| **IC\_E\_TURB\_G**(IC) |  | DP |
| Initial value of epsilon in k-epsilon. | | |
| **IC\_DES\_FIT\_TO\_REGION**(IC) | *[ .FALSE. ]* | L |
| Flag for inflating initial lattice distribution to the entire ic region. | | |
| **IC\_PIC\_CONST\_NPC**(IC, Phase) | *[ 0 ]* | I |
| 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. | | |
| *Dependent: SOLIDS\_MODEL* | Pic | |
| *Conflict: IC\_PIC\_CONST\_STATWT* | Defined | |
| **IC\_PIC\_CONST\_STATWT**(IC, Phase) | *[ ZERO ]* | DP |
| Flag to specify the initial constant statistical weight for computational particles/parcels. Actual number of parcels will be automatically computed. | | |
| *Dependent: SOLIDS\_MODEL* | Pic | |
| *Conflict: IC\_PIC\_CONST\_NPC* | Defined | |

## 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) |  | DP |
| X coordinate of the west face or edge. | | |
| **BC\_X\_E**(BC) |  | DP |
| X coordinate of the east face or edge. | | |
| **BC\_Y\_S**(BC) |  | DP |
| Y coordinate of the south face or edge. | | |
| **BC\_Y\_N**(BC) |  | DP |
| Y coordinate of the north face or edge. | | |
| **BC\_Z\_B**(BC) |  | DP |
| Z coordinate of the bottom face or edge. | | |
| **BC\_Z\_T**(BC) |  | DP |
| Z coordinate of the top face or edge. | | |
| **BC\_I\_W**(BC) |  | I |
| I index of the west-most cell. | | |
| **BC\_I\_E**(BC) |  | I |
| I index of the east-most cell. | | |
| **BC\_J\_S**(BC) |  | I |
| J index of the south-most cell. | | |
| **BC\_J\_N**(BC) |  | I |
| J index of the north-most cell. | | |
| **BC\_K\_B**(BC) |  | I |
| K index of the bottom-most cell. | | |
| **BC\_K\_T**(BC) |  | I |
| K index of the top-most cell. | | |
| **BC\_TYPE**(BC) |  | C |
| Type of boundary. | | |
| *DUMMY* | Na | |
| *MASS\_INFLOW* or *MI* | Na | |
| *MASS\_OUTFLOW* or *MO* | Na | |
| *P\_INFLOW* or *PI* | Na | |
| *P\_OUTFLOW* or *PO* | Na | |
| *FREE\_SLIP\_WALL* or *FSW* | Na | |
| *NO\_SLIP\_WALL* or *NSW* | Na | |
| *PAR\_SLIP\_WALL* or *PSW* | Na | |
| **BC\_HW\_G**(BC) |  | DP |
| Gas phase hw for partial slip boundary. | | |
| **BC\_HW\_S**(BC, Phase) |  | DP |
| Solids phase hw for partial slip boundary. | | |
| **BC\_UW\_G**(BC) |  | DP |
| Gas phase uw for partial slip boundary. | | |
| **BC\_UW\_S**(BC, Phase) |  | DP |
| Solids phase uw for partial slip boundary. | | |
| **BC\_VW\_G**(BC) |  | DP |
| Gas phase vw for partial slip boundary. | | |
| **BC\_VW\_S**(BC, Phase) |  | DP |
| Solids phase vw for partial slip boundary. | | |
| **BC\_WW\_G**(BC) |  | DP |
| Gas phase ww for partial slip boundary. | | |
| **BC\_WW\_S**(BC, Phase) |  | DP |
| Solids phase ww for partial slip boundary. | | |
| **BC\_JJ\_PS**(BC) |  | I |
| Johnson and jackson partial slip bc. | | |
| *0* | Na | |
| *1* | Na | |
| **BC\_THETAW\_M**(BC, Phase) |  | DP |
| Specified wall value, thetaw\_m, in diffusion boundary condition: d(theta\_m)/dn + hw (theta\_m - thetaw\_m) = c, where n is the fluid-to-wall normal. | | |
| **BC\_HW\_THETA\_M**(BC, Phase) |  | DP |
| Transfer coefficient, hw, in diffusion boundary condition: d(theta\_m)/dn + hw (theta\_m - thetaw\_m) = c, where n is the fluid-to-wall normal. | | |
| **BC\_C\_THETA\_M**(BC, Phase) |  | DP |
| Specified constant flux, c, in diffusion boundary condition: d(theta\_m)/dn + hw (theta\_m - thetaw\_m) = c, where n is the fluid-to-wall normal. | | |
| **BC\_HW\_T\_G**(BC) |  | DP |
| Gas phase heat transfer coefficient, hw, in diffusion boundary condition: d(t\_g)/dn + hw (t\_g - tw\_g) = c, where n is the fluid-to-wall normal. | | |
| **BC\_TW\_G**(BC) |  | DP |
| Specified gas phase wall temperature, tw\_g, in diffusion boundary condition: d(t\_g)/dn + hw (t\_g - tw\_g) = c, where n is the fluid-to-wall normal. | | |
| **BC\_C\_T\_G**(BC) |  | DP |
| Specified constant gas phase heat flux, c, in diffusion boundary condition: d(t\_g)/dn + hw (t\_g - tw\_g) = c, where n is the fluid-to-wall normal. | | |
| **BC\_HW\_T\_S**(BC, Phase) |  | DP |
| Solids phase heat transfer coefficient, hw, in diffusion boundary condition: d(t\_s)/dn + hw (t\_s - tw\_s) = c, where n is the fluid-to-wall normal. | | |
| **BC\_TW\_S**(BC, Phase) |  | DP |
| Specified solids phase wall temperature, tw\_s, in diffusion boundary condition: d(t\_s)/dn + hw (t\_s - tw\_s) = c, where n is the fluid-to-wall normal. | | |
| **BC\_C\_T\_S**(BC, Phase) |  | DP |
| Specified constant solids phase heat flux, c, in diffusion boundary condition: d(t\_s)/dn + hw (t\_s - tw\_s) = c, where n is the fluid-to-wall normal. | | |
| **BC\_HW\_X\_G**(BC, Species) |  | DP |
| Gas phase species mass transfer coefficient, hw, in diffusion boundary condition: d(x\_g)/dn + hw (x\_g - xw\_g) = c, where n is the fluid-to-wall normal. | | |
| **BC\_XW\_G**(BC, Species) |  | DP |
| Specified wall gas species mass fraction, xw, in diffusion boundary condition: d(x\_g)/dn + hw (x\_g - xw\_g) = c, where n is the fluid-to-wall normal. | | |
| **BC\_C\_X\_G**(BC, Species) |  | DP |
| Specified constant gas species mass flux, c, in diffusion boundary condition: d(x\_g)/dn + hw (x\_g - xw\_g) = c, where n is the fluid-to-wall normal. | | |
| **BC\_HW\_X\_S**(BC, Phase, Species) |  | DP |
| Solid phase species mass transfer coefficient, hw, in diffusion boundary condition: d(x\_s)/dn + hw (x\_s - xw\_s) = c, where n is the fluid-to-wall normal. | | |
| **BC\_XW\_S**(BC, Phase, Species) |  | DP |
| Specified solids species mass fraction at the wall, xw, in diffusion boundary condition: d(x\_g)/dn + hw (x\_g - xw\_g) = c, where n is the fluid-to-wall normal. | | |
| **BC\_C\_X\_S**(BC, Phase, Species) |  | DP |
| Specified constant solids species mass flux, c, in diffusion boundary condition: d(x\_s)/dn + hw (x\_s - xw\_s) = c, where n is the fluid-to-wall normal. | | |
| **BC\_HW\_SCALAR**(BC, Scalar Eq.) |  | DP |
| Scalar transfer coefficient, hw, in diffusion boundary condition: d(scalar)/dn + hw (scalar - scalarw) = c, where n is the fluid-to-wall normal. | | |
| **BC\_SCALARW**(BC, Scalar Eq.) |  | DP |
| Specified scalar value at the wall, scalarw, in diffusion boundary condition: d(scalar)/dn + hw (scalar - scalarw) = c, where n is the fluid-to-wall normal. | | |
| **BC\_C\_SCALAR**(BC, Scalar Eq.) |  | DP |
| Specified constant scalar flux, c, in diffusion boundary condition: d(scalar)/dn + hw (scalar - scalarw) = c, where n is the fluid-to-wall normal. | | |
| **BC\_EP\_G**(BC) |  | DP |
| Void fraction at the bc plane. | | |
| **BC\_P\_G**(BC) |  | DP |
| Gas pressure at the bc plane. | | |
| **BC\_ROP\_S**(BC, Phase) |  | DP |
| Bulk density of solids phase at the bc plane. | | |
| **BC\_EP\_S**(BC, Phase) |  | DP |
| Solids volume fraction at the bc plane. | | |
| **BC\_T\_G**(BC) |  | DP |
| Gas phase temperature at the bc plane. | | |
| **BC\_T\_S**(BC, Phase) |  | DP |
| Solids phase-m temperature at the bc plane. | | |
| **BC\_THETA\_M**(BC, Phase) |  | DP |
| Solids phase-m granular temperature at the bc plane. | | |
| **BC\_X\_G**(BC, Species) |  | DP |
| Mass fraction of gas species n at the bc plane. | | |
| **BC\_X\_S**(BC, Phase, Species) |  | DP |
| Mass fraction of solids phase-m, species n at the bc plane. | | |
| **BC\_U\_G**(BC) |  | DP |
| X-component of gas velocity at the bc plane. | | |
| **BC\_U\_S**(BC, Phase) |  | DP |
| X-component of solids-phase velocity at the bc plane. | | |
| **BC\_V\_G**(BC) |  | DP |
| Y-component of gas velocity at the bc plane. | | |
| **BC\_V\_S**(BC, Phase) |  | DP |
| Y-component of solids-phase velocity at the bc plane. | | |
| **BC\_W\_G**(BC) |  | DP |
| Z-component of gas velocity at the bc plane. | | |
| **BC\_W\_S**(BC, Phase) |  | DP |
| 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) |  | DP |
| Gas volumetric flow rate through the boundary. | | |
| **BC\_VOLFLOW\_S**(BC, Phase) |  | DP |
| Solids volumetric flow rate through the boundary. | | |
| **BC\_MASSFLOW\_G**(BC) |  | DP |
| Gas mass flow rate through the boundary. | | |
| **BC\_MASSFLOW\_S**(BC, Phase) |  | DP |
| 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) |  | DP |
| 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) |  | DP |
| Value of normal velocity during the initial interval bc\_dt\_0. | | |
| **BC\_DT\_H**(BC) |  | DP |
| The interval when normal velocity is equal to bc\_jet\_gh. | | |
| **BC\_JET\_GH**(BC) |  | DP |
| Value of normal velocity during the interval bc\_dt\_h. | | |
| **BC\_DT\_L**(BC) |  | DP |
| The interval when normal velocity is equal to bc\_jet\_gl. | | |
| **BC\_JET\_GL**(BC) |  | DP |
| Value of normal velocity during the interval bc\_dt\_l. | | |
| **BC\_SCALAR**(BC, Scalar Eq.) |  | DP |
| Boundary value for user-defined scalar equation. | | |
| **BC\_K\_TURB\_G**(BC) |  | DP |
| Boundary value of k for k-epsilon equation. | | |
| **BC\_E\_TURB\_G**(BC) |  | DP |
| Boundary value of epsilon for k-epsilon equation. | | |
| **BC\_VELMAG\_G**(BC) |  | DP |
| Magnitude of gas velocity in a specifed boundary region. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **BC\_VELMAG\_S**(BC, Phase) |  | DP |
| Magnitude of gas velocity in a specifed boundary region. | | |
| *Dependent: CARTESIAN\_GRID* | .true. | |
| **BC\_PIC\_MI\_CONST\_NPC**(BC, Phase) | *[ 0 ]* | I |
| 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. | | |
| *Dependent: SOLIDS\_MODEL* | Pic | |
| *Conflict: BC\_PIC\_CONST\_STATWT* | Defined | |
| **BC\_PIC\_MI\_CONST\_STATWT**(BC, Phase) | *[ ZERO ]* | DP |
| Flag to specify the constant statistical weight for inflowing computational particles/parcels. Actual number of parcels will be automatically computed. | | |
| *Conflict: IC\_PIC\_CONST\_NPC* | Defined | |
| **BC\_PO\_APPLY\_TO\_DES**(BC) | *[ .TRUE. ]* | L |
| 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) | *[ .TRUE. ]* | L |
| Flag to make the inflow plane invisible to discrete solids. Set this flag to false to remove to inflow plane. | | |
| **BC\_JJ\_M** | *[ .FALSE. ]* | L |
| Use the modified johnson and jackson partial slip bc with variable specularity coefficient. | | |
| *Dependent: E\_W* | Defined | |
| *Dependent: PHI\_W* | Defined | |
| **FORCE\_ORD\_BC** | *[ .FALSE. ]* | L |
| Logical to force the inlet to operate with an ordered boundary condition. This may be useful during long simulations or if the inlet appears to be taking a long time to randomly place particles. | | |

## 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) |  | DP |
| X coordinate of the west face or edge. | | |
| **IS\_X\_E**(IS) |  | DP |
| X coordinate of the east face or edge. | | |
| **IS\_Y\_S**(IS) |  | DP |
| Y Coordinate Of The South Face Or Edge | | |
| **IS\_Y\_N**(IS) |  | DP |
| Y Coordinate Of The North Face Or Edge | | |
| **IS\_Z\_B**(IS) |  | DP |
| Z Coordinate Of The Bottom Face Or Edge | | |
| **IS\_Z\_T**(IS) |  | DP |
| Z Coordinate Of The Top Face Or Edge | | |
| **IS\_I\_W**(IS) |  | I |
| I index of the west-most cell. | | |
| **IS\_I\_E**(IS) |  | I |
| I Index Of The East-Most Cell | | |
| **IS\_J\_S**(IS) |  | I |
| J Index Of The South-Most Cell | | |
| **IS\_J\_N**(IS) |  | I |
| J Index Of The North-Most Cell | | |
| **IS\_K\_B**(IS) |  | I |
| K Index Of The Bottom-Most Cell | | |
| **IS\_K\_T**(IS) |  | I |
| K Index Of The Top-Most Cell | | |
| **IS\_TYPE**(IS) |  | C |
| Type Of Internal Surface | | |
| *IMPERMEABLE* or *IP* | No gas or solids flow through the surface. | |
| *SEMIPERMEABLE* | 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). | |
| **IS\_PC**(IS) | *[ ZERO ]* | DP |
| Inertial resistance coefficient. | | |
| **IS\_PC**(IS) | *[ ZERO ]* | DP |
| Inertial resistance coefficient. | | |
| **IS\_VEL\_S**(IS, Phase) | *[ ZERO ]* | DP |
| Value of fixed solids velocity through semipermeable surfaces. | | |

## Point Source

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) |  | DP |
| X coordinate of the west face or edge. | | |
| **PS\_X\_E**(PS) |  | DP |
| X coordinate of the east face or edge. | | |
| **PS\_Y\_S**(PS) |  | DP |
| Y coordinate of the south face or edge. | | |
| **PS\_Y\_N**(PS) |  | DP |
| Y coordinate of the north face or edge. | | |
| **PS\_Z\_B**(PS) |  | DP |
| Z coordinate of the bottom face or edge. | | |
| **PS\_Z\_T**(PS) |  | DP |
| Z coordinate of the top face or edge. | | |
| **PS\_I\_W**(PS) |  | I |
| I index of the west-most cell. | | |
| **PS\_I\_E**(PS) |  | I |
| I index of the east-most cell. | | |
| **PS\_J\_S**(PS) |  | I |
| J index of the south-most cell. | | |
| **PS\_J\_N**(PS) |  | I |
| J index of the north-most cell. | | |
| **PS\_K\_B**(PS) |  | I |
| K index of the bottom-most cell. | | |
| **PS\_K\_T**(PS) |  | I |
| K index of the top-most cell. | | |

Point sources introduce mass directly into a computational cell unlike a boundary condition which specifies flow along a cell face. One consequence of this implementation is that point sources are subjected to convection/diffusion forces and may not travel parallel to the specified directional preference. Directional preference is specified with a velocity vector, however, directional preference is not required.

|  |  |  |
| --- | --- | --- |
| **PS\_U\_G**(PS) |  | DP |
| X-component of incoming gas velocity. | | |
| **PS\_V\_G**(PS) |  | DP |
| Y-component of incoming gas velocity. | | |
| **PS\_W\_G**(PS) |  | DP |
| Z-component of incoming gas velocity. | | |
| **PS\_MASSFLOW\_G**(PS) |  | DP |
| Gas mass flow rate through the point source. | | |
| **PS\_T\_G**(PS) |  | DP |
| Temperature of incoming gas. | | |
| **PS\_X\_G**(PS) |  | DP |
| Gas phase incoming species n mass fraction. | | |
| **PS\_U\_S**(PS, Phase) |  | DP |
| X-component of incoming solids velocity. | | |
| **PS\_V\_S**(PS, Phase) |  | DP |
| Y-component of incoming solids velocity. | | |
| **PS\_W\_S**(PS, Phase) |  | DP |
| Z-component of incoming solids velocity. | | |

Examples showing how to setup point sources can be found in /mfix/tutorials/point\_source\_spiral

|  |  |  |
| --- | --- | --- |
| **PS\_MASSFLOW\_S**(PS, Phase) |  | DP |
| Solids mass flow rate through the point source. | | |
| **PS\_T\_S**(PS, Phase) |  | DP |
| Temperature of incoming solids. | | |
| **PS\_X\_S**(PS, Phase, Species) |  | DP |
| Solids phase incoming species n mass fraction. | | |

## Output Control

|  |  |  |
| --- | --- | --- |
| **RES\_DT** |  | DP |
| Interval at which restart (.res) file is updated. | | |
| **SPX\_DT** |  | DP |
| Interval at which .spx files are updated. Sp1: void fraction (ep\_g) sp2: gas pressure (p\_g), and solids pressure (p\_star) sp3: gas veloicty (u\_g, v\_g, w\_g) sp4: solids veloicty (u\_s, v\_s, w\_s) sp5: solids builk 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) | | |
| **OUT\_DT** |  | DP |
| 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** | *[ 25 ]* | I |
| Number of time steps between .log file updates. | | |
| **FULL\_LOG** | *[ .FALSE. ]* | L |
| Display the residuals on the screen and provide messages about convergence on the screen and in the .log file. | | |
| **RESID\_STRING**(Residual Index) |  | C |
| 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** | *[ .FALSE. ]* | L |
| Display residuals by equation. | | |
| **USR\_DT**(USR) |  | DP |
| Intervals at which subroutine write\_usr1 is called. | | |
| **USR\_X\_W**(USR) |  | DP |
| Udf hook: x coordinate of the west face or edge. | | |
| **USR\_X\_E**(USR) |  | DP |
| Udf hook: x coordinate of the east face or edge. | | |
| **USR\_Y\_S**(USR) |  | DP |
| Udf hook: y coordinate of the south face or edge. | | |
| **USR\_Y\_N**(USR) |  | DP |
| Udf hook: y coordinate of the north face or edge. | | |
| **USR\_Z\_B**(USR) |  | DP |
| Udf hook: z coordinate of the bottom face or edge. | | |
| **USR\_Z\_T**(USR) |  | DP |
| Udf hook: z coordinate of the top face or edge. | | |
| **USR\_I\_W**(USR) |  | I |
| Udf hook: i index of the west-most cell. | | |
| **USR\_I\_E**(USR) |  | I |
| Udf hook: i index of the east-most cell. | | |
| **USR\_J\_S**(USR) |  | I |
| Udf hook: j index of the south-most cell. | | |
| **USR\_J\_N**(USR) |  | I |
| Udf hook: j index of the north-most cell. | | |
| **USR\_K\_B**(USR) |  | I |
| Udf hook: k index of the bottom-most cell. | | |
| **USR\_K\_T**(USR) |  | I |
| Udf hook: k index of the top-most cell. | | |
| **USR\_TYPE**(USR) |  | C |
| Udf hook: type of user-defined ouput: binary of ascii. | | |
| **USR\_VAR**(USR) |  | C |
| Udf hook: variables to be written in the user-defined output files. | | |
| **USR\_FORMAT**(USR) |  | C |
| Udf hook: format for writing user-defined (ascii) output file. | | |
| **USR\_EXT**(USR) |  | C |
| Udf hook: file extension for the user-defined output. | | |
| **REPORT\_MASS\_BALANCE\_DT** |  | DP |
| Frequency to perform an overall species mass balance. Leaving undefined suppresses the mass balance calculations which can slightly extend run time. | | |
| **BDIST\_IO** | *[ .FALSE. ]* | L |
| Use distributed io :: each rank generates res/spx files. | | |
| *Dependent: SPECIES\_EQ* | .true. | |
| **BSTART\_WITH\_ONE\_RES** | *[ .FALSE. ]* | L |
| Restart a unified io run as distributed io. | | |
| *Dependent: RUN\_TYPE* | Restart\_2 | |
| *Dependent: BDIST\_IO* | .true. | |
| **BWRITE\_NETCDF**(NetCDF Variable Reference) | *[ .FALSE. ]* | L |
| 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 veloicty (u\_g, v\_g, w\_g) 5: solids veloicty (u\_s, v\_s, w\_s) 6: solids builk 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. | |

## Chemical Reactions

|  |  |  |
| --- | --- | --- |
| **STIFF\_CHEMISTRY** | *[ .FALSE. ]* | L |
| Flag to use stiff chemistry solver (direct integration). | | |
| *Conflict: USE\_RRATES* | .true. | |
| **STIFF\_CHEM\_MAX\_STEPS** |  | I |
| Maximum number of internal steps odepack may use to integrate over the time interval.leaving this value unspecifed permits an unlimited number of steps. Thee stiff solver reports the number of cells that exceed the number of steps as 'incomplete'. | | |
| *Dependent: STIFF\_CHEMISTRY* | .true. | |
| *Conflict: USE\_RRATES* | .true. | |
| **USE\_RRATES** | *[ .FALSE. ]* | L |
| Flag to use legacy chemcial reaction udfs. | | |
| **SPECIES\_NAME** |  | C |
| 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. | | |
| *Legacy:* | true | |
| *Dependent: USE\_RRATES* | .true. | |
| **NMAX** |  | I |
| Number of species in phase m. Note that the gas phase is indicated as m=0. | | |
| *Dependent: USE\_RRATES* | .true. | |
| **CALL\_DI** | *[ .FALSE. ]* | L |
| Flag for previous stiff solver. | | |
| *Legacy:* | true | |
| **CALL\_GROW** | *[ .FALSE. ]* | L |
| Flag to specify variable solids diameter in original stiff chem solver. (non-functional, removed in 2013-2 release) | | |
| *Legacy:* | true | |
| **CALL\_ISAT** | *[ .FALSE. ]* | L |
| Flag to use isat tables with original di solver. (non-functional, removed in 2013-2 release) | | |
| *Legacy:* | true | |
| **ISATDT** |  | DP |
| Specified constant call to isat functions. (non-functional, removed in 2013-2 release) | | |
| *Legacy:* | true | |

## Chemical Reactions – basic options

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 directly supported.



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

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 and/or usr\_rates\_des.f.
5. Use make\_mfix to (re)build the MFiX executable.

### Provide species names in the data file with the following keywords.

|  |  |  |
| --- | --- | --- |
| **Keyword (dimension)** | **Type** | **Description** |
| NMAX\_g  [UNDEFINED\_I] | I | Number of species comprising the gas phase |
| SPECIES\_g(n)  [UNDEFINED\_C] | C | Name of gas phase species n as it appears in the materials database |
| NMAX\_s(m)  [UNDEFINED\_I] | I | Number of species comprising solids phase m |
| SPECIES\_s(m,n)  [UNDEFINED\_C] | C | Name of solids phase m, species n as it appears in the materials database |

Species names must appear **exactly** as given in the materials database (see *Section 5.12,* *Thermochemical Properties*). Species names are typically 18 characters, and for some species, trailing spaces are needed.



For reacting discrete element simulations (DES), gas phase species are defined using the above keywords (NMAX\_g, Species\_g). However, DES specific keywords are used to identify the number of discrete solids phase species and material database names.

|  |  |  |
| --- | --- | --- |
| **Keyword (dimension)** | **Type** | **Description** |
| DES\_NMAX\_s(m)  [UNDEFINED\_I] | I | Number of species comprising discrete solids phase m |
| DES\_SPECIES\_s(m,n)  [UNDEFINED\_C] | C | Name of discrete solids phase m, species n as it appears in the materials database |

### Assign a *unique identifier* (alias) to each species with the following keyworks.

|  |  |  |
| --- | --- | --- |
| **Keyword (dimension)** | **Type** | **Description** |
| SPECIES\_ALIAS\_g(n)  [UNDEFINED\_C] | C | User defined name for gas phase species n |
| SPECIES\_ALIAS\_s(m,n)  [UNDEFINED\_C] | C | User defined name for solids phase m, species n |
| DES\_SPECIES\_ALIAS\_s (m,n)  [UNDEFINED\_C] | C | User defined name for discrete solids phase m, species n |

**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.

@(RXNS)

rxn1 {

<input parameters>

}

rxn2 {

<input parameters>

}

@(END)

reaction construct

reaction block

reaction construct

reaction identifiers

mfix.dat 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 DES homogeneous gas phase reactions).

@(DES\_RXNS)…@(DES\_END) – indicates heterogeneous DES chemical reactions (particle/gas).

A data file can only contain **one reaction block of each type**, whereas a reaction block must contain **one or more reaction constructs**.



The following keywords are available within a reaction construct.

|  |  |  |
| --- | --- | --- |
| **Keyword (dimension)** | **Type** | **Description** |
| CHEM\_EQ  [UNDEFINED\_C] | C | Chemical equation for the reaction constructed from species aliases.  Ex: Char combustion  CHEM\_EQ = “C + 0.5O2 --> CO” |
| DH **(†Optional)**  [UNDEFINED] | DP | User provided heat of reaction  (cal/ mole for CGS and J/kmole for SI). |
| fracDH(m) **(†Optional)**  [ZERO] | DP | The fractional amount of DH supplied to phase m. |

\* A chemical reaction equation of “NONE” deactivates the reaction during a simulation (e.g., CHEM\_EQ = “NONE”).

† 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 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.   
  hidden text

@(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) ! *End reaction block*

* 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 make (!).

**Example: Methane Combustion:**

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:**

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:**

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:

CO2 gasification:

Char combustion:

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

DES\_NMAX\_s(1) = 2

DES\_Species\_s(1,1) = “C(GR) REF ELEMENT”

DES\_Species\_s(1,2) = “Coal Ash”

DES\_Species\_Alias\_s(1,1) = “C”

DES\_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.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

where and are the rates of consumption of methane and oxygen, and and 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:**

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).

data file:

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\_02 ! *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*

! *CH4 + 2O2 --> CO2 + 2H2O (reacted moles/sec.cm^3)*

!*-----------------------------------------------------------------//*

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:**

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:**

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:**

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.

data file input:

NMAX\_g = 2 ! *No. of gas phase species*

Species\_g(1) = “Air” “H2O” ! Database names

Species\_Alias\_g(1) = “Air” “Vapor” ! Species Aliases

DES\_NMAX\_s(1) = 1 ! *No. of DEM solids phase species*

DES\_Species\_s(1,1) = “H2O(L)” ! Database names

DES\_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*

⋮

! *Liquid --> Vapor (reacted moles/sec)*

!*-------------------------------------------------------//*

! *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^3)*

Sa = Pi \* 4.0d0 \* (DES\_RADIUS(NP)\*\*2)

! *Calculate the mass transfer rate (moles/sec)*

DES\_RATES(Evap) = Sa \* H2O\_xfr \* Cmg\_H2O

ELSE

DES\_RATES(Evap) = ZERO

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 Section 2.3.1. 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 generated the species.inc file which is included within the user\_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 preformed 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:

|  |  |  |
| --- | --- | --- |
| **Keyword (dimension)** | **Type** | **Description** |
| USE\_RRATES  [.FALSE] | L | Use the automated reaction rate UDFS; usr\_rates.f and usr\_rates\_des.f. |
| .TRUE. |  | Access legacy rrates.f file for reaction rate information. Rates of formation/consumption as well as heats of reaction are provided by the user. |
| NMAX(m)  [UNDEFINED\_I] | I | Number of species in phase m. Note that the gas phase is indicated as m=0. |
| SPECIES\_NAME(n)  [UNDEFINED\_C] | C | 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. |

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\_RRATS=.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.

## Chemical reactions – advanced options

**Stiff Chemistry Solver (Fractional-Step Method)**

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:

|  |  |  |
| --- | --- | --- |
| **Keyword (dimension)** | **Type** | **Description** |
| STIFF\_CHEMISTRY  [.FALSE.] | L | Runtime logical to invoke stiff chemistry solver. |

Reactions are specified using the same approached 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 rrates.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 there usage is available in model/ODEPACK.F.
* It is recommend 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.

**ISAT and Direct Integration for Chemical Reactions**

MFIX no longer supports implementing reactive chemistry via ISAT (In Situ Adaptive Tabulation). The following keywords are disabled and result in runtime errors when included:

|  |  |  |
| --- | --- | --- |
| **Keyword (dimension)** | **Type** | **Description** |
| CALL\_DI | L | **DISABLED**  ~~Variable to decide if chemical reactions are time-split and solved using direct integration (DI) with ODE solver~~. |
| CALL\_ISAT | L | **DISABLED**  ~~Variable to decide if chemical reactions are time-split and solved using ISAT~~. |
| CALL\_GROW | L | **DISABLED**  ~~Variable to decide if particle growth due to chemical reactions is calculated~~. |
| ISATdt | DP | **DISABLED**  ~~Time step for ISAT simulations~~. |

## 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 thermo-chemical data from files in the following order:

1. mfix.dat

2. BURCAT.THR file in the run directory

3. mfix/model/thermochemical/BURCAT.THR.

**Option 3. Above is only available on Linux systems. If you are compiling under Windows with MinGW or Cygwin you need to place a copy of the BURCAT.THR in your run directory.  Otherwise, Windows will not be able to locate it at runtime.**



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:

1.91178600E+00

9.60267960E-03

-3.38387841E-06

5.38797240E-10

-3.19306807E-14

-1.00992136E+04

8.48241861E+00

5.14825732E+00

-1.37002410E-02

4.93749414E-05

-4.91952339E-08

1.70097299E-11

-1.02453222E+04

-4.63322726E+00

-8.97226656E+03

CH4 RRHO

g 8/99C 1.H 4. 0. 0.G

200.000 6000.000

B

16.04246

1

than the NRRAO2. Max Lst Sq Error Cp @ 6000. K 0.62%.

3

4

CH4 METHANE Same as the Anharmonic but calculated Using the RRHO method rather

74-82-8

2

CAS  
identifier

valid temperature range

molecular weight

comments

formation enthalpy at 298K

species  
name

high temperature coefficients

low temperature coefficients

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 () should be used for temperatures in the range Tlow to 1000K and the high temperature coefficients () should be used for temperatures in the range 1000K to Thigh. The coefficients are stored in a fixed format (E15.0) as follows:

where is the formation enthalpy at 298K.

The normalized specific heat is given by

**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 Cp 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.

**If you include thermochemical properties in mfix.dat, all keywords defined below the line that starts with THERMO DATA will be ignored.**



## Parallelization Control

|  |  |  |
| --- | --- | --- |
| **NODESI** |  | I |
| Number of grid blocks in x-direction. | | |
| **NODESJ** |  | I |
| Number of grid blocks in y-direction. | | |
| **NODESK** |  | I |
| Number of grid blocks in z-direction. | | |
| **SOLVER\_STATISTICS** | *[ .FALSE. ]* | L |
| Print Out Additional Statistics For Parallel Runs | | |
| **DEBUG\_RESID** | *[ .TRUE. ]* | L |
| Group residuals to reduce global collectives. | | |
| **ENABLE\_DMP\_LOG** | *[ .FALSE. ]* | L |
| All ranks write error messages. | | |
| **DBGPRN\_LAYOUT** | *[ .FALSE. ]* | L |
| Print the index layout for debugging. | | |

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.

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” preconditioners 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 an 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.

## 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 terminate cleanly at the end of the batch job session, several keywords need to be entered in mfix.dat. 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 clean 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

“CO2 + 2H2O” ! Chemical Reaction Line 2

} ! End reaction 1 construct

@(END) ! End reaction block

Setting CHK\_BATCH\_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** | *[ .FALSE. ]* | L |
| Enables clean termination feature. | | |
| **BATCH\_WALLCLOCK** | *[ 9000.0 ]* | DP |
| Total wall-clock duration of the job, in seconds. | | |
| **TERM\_BUFFER** | *[ 180.0 ]* | DP |
| Buffer time when initiating clean termination, in seconds. | | |

## Direct Quadrature Method Of Moments (Dqmom)

|  |  |  |
| --- | --- | --- |
| **CALL\_DQMOM** | *[ .FALSE. ]* | L |
| Variable to decide if the population balance equations are solved. | | |
| **AGGREGATION\_EFF** | *[ 0.D0 ]* | DP |
| Success-factor for aggregation. | | |
| **BREAKAGE\_EFF** | *[ 0.D0 ]* | DP |
| Success-factor for breakage. | | |

**Example using DQMOM**

The example case is a fluidized bed simulation with two solid phases; each has its own particle size. In the code, the population balance equation is turned on. If aggregation dominates, the average particle size will increase. If breakage dominates, the average particle size will decrease. The user can turn off DQMOM by set “Call\_ DQMOM =.FALSE.”. The ODE solver for the scalar in the MFIX code solves the population balance equation, so “Nscalar” has to be set as the number of solid phases. The initial values for the scalar are set as the initial particle diameter. The aggregation and breakage kernel from kinetic theory is used. The success factor of aggregation and breakage can be changed by setting different values for “Aggregation\_eff” and “Breakage\_eff”. An example mfix.dat file is included.

Please refer to the following document for a brief description of the implementation and utilization of the quadrature method of moments (QMOM): A. Passalacqua, and R. O. Fox, “Documentation of open-source MFIX-QMOM software for gas-solids flows”, available from https://mfix.netl.doe.gov/documentation/qmomk doc\_2012-1.pdf

# 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:
   1. MFIX version you are trying to install or run
   2. Some details on your operating system environment (for Linux: copy and paste the response of uname –a command, Linux distribution name and version also)
   3. Your compiler name and version number (e.g. ifort –v will give the version number for Intel fortran compiler)
   4. Output for your $PATH environment (in csh type echo $PATH)
   5. 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.



# User contribution

If you wish to contribute to the development of MFIX, please contact the MFIX team at [admin@mfix.netl.doe.gov](emailto: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. Proper credit will be given to all contributors.

1. For bubbly gas-liquid flows, the carrier phase would be the liquid phase and the disperse phase would be the gas bubbles. [↑](#footnote-ref-1)