MFC: User's guide

S. H. Bryngelson, K. Schmidmayer, T. Colonius

Division of Engineering and Applied Science,
California Institute of Technology,
1200 E California Blvd, Pasadena, CA 91125, USA

V. Coralic

Prime Air,
Amazon Inc.,
Seattle, WA 98108, USA

J. C. Meng

Bosch Research and Technology Center,
Sunnyvale, CA 94085, USA

K. Maeda

Department of Mechanical Engineering,
University of Washington,
Seattle, WA 98195, USA

Contents

1 Installation

The documents that describe how to configure and install the MFC are located in the source code as `CONFIGURE` and `INSTALL`. They are also described here.

1.1 Step 1: Configure and ensure dependencies can be located

1.1.1 Main dependencies: MPI and Python

Mac OSX includes Python by default. If you do not have Python, it can be installed via Homebrew[1] on OSX as:

```
# brew install python
```

or compiled via your favorite package manager on Unix systems.

An MPI Fortran compiler is required for all systems. If you do not have one, Homebrew can take care of this on OSX:

```
# brew install open-mp
```

or compiled via another package manager on Unix systems.

1.1.2 Simulation code dependency: FFTW

If you already have FFTW compiled, specify the location of your FFTW library and include files in `Makefile.user` (`fftw_lib_dir` and `fftw_include_dir`)

If you do not have FFTW compiler, the library and installer are included in this package. Just:

```
# cd installers
# ./install_fftw.sh
```

1.1.3 Post process code dependency: Silo/HDF5

Post-processing of parallel data files is not required, but can indeed be handled with the MFC. For this, HDF5 and Silo must be installed

On OSX, a custom Homebrew tap for Silo is included in the `installers/` directory. You can use it via

```
    # cd installers
    # brew install silo.rb
```

This will install silo and its dependences (including HDF5) in their usual locations (`/usr/local/lib` and `/usr/local/include`)

On Unix systems, you can install via a package manager or from source. On CentOS (also Windows 7), HDF5 binaries can be found online.[2] To install them, open their archive in your intended location via

[1]Located at https://brew.sh
[2]For example, https://support.hdfgroup.org/ftp/HDF5/current18/bin/

```
# tar -zxf [your HDF5 archive]
```

Silo should be downloaded[3] and installed via

```
# tar -zxf [your Silo archive]
# cd [your Silo archive]
# ./configure --prefix=[target installation directory] --enable-
    pythonmodule --enable-optimization --disable-hzip --disable-fpzip
--enableportable-binary FC=mpif90 F77=mpif77 -with-hdf5=[your hdf5
    directory]/include,/[your hdf5 directory]/lib --disable-silex
# make
# make install
```

Add the following line to your `~/.bash_profile`:

```
# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/[your silo directory]/
    lib:/[your hdf5 directory]/lib
```

Finally:

```
# source ~/.bash_profile
```

You will then need to modify `silo_lib_dir` and `silo_include_dir` in `Makefile.user` to point to `[your silo directory]`.

1.2 Step 2: Build and test

Once all dependencies have been installed, the MFC can be built via

```
# make
```

from the MFC directory. This will build all MFC components. Individual components can be built via

```
# make [component]
```

where `[component]` is one of `pre_process`, `simulation`, or `post_process`.

Once this is completed, you can ensure that the software is working as intended by

```
# make test
```

2 How to run

The MFC can be run by changing into a case directory and executing the appropriate Python input file. Example Python input files can be found in the /test/ case directories, and they are called `input.py`. Their contents, and a guide to filling them out, are the subject of section 3. The MFC can be executed as

```
# python pre_process
```

[3]Located at: https://wci.llnl.gov/simulation/computer-codes/silo/downloads

which will generate the restart and grid files that will be read by the simulation code. Then

```
# python simulation
```

will execute the flow solver. The last (optional) step is to post treat the data files and output HDF5 databases for the flow variables via

```
# python post_process
```

Note that this requires installation of Silo and HDF5, as described in section 1.1.3.

3 Input parameters

There are several components in the file `input.py`. A description of the variables in this file can be found in section 3. At a minimum, `input.py` must specify

1. Simulation logistics, such as the location of the directory, the number of processors and run time, the spatial grid size and properties, and the time step size (see table 1)
2. Simulation algorithm parameters, such as the physical model selection, WENO order and properties, Riemann solver, and boundary conditions (see table 2)
3. Output parameters, such as the type of output files to write and with what precision, the variables to write to those files, and any probes placed in the domain (see table 3)
4. Patch parameters, such as its geometry, size, and primitive variables (see table 4)
5. Stiffened-gas equation of state parameters for each component (see table 5)

Additional options are available and have their own set of required parameters if enabled, including

1. Acoustic source options, such as the number of sources, their amplitude, size, location, and direction (see table 6)
2. Bubble model parameters, such as the single-bubble model used, the compression model, and reference bubble size, viscosity, and surface tension coefficient (see table 7)

Parameter	Type	Description
`case_dir`	String	Case script directory
`run_time_info`	Logical	Output run-time information
`nodes`	Integer	Number of nodes
`ppn`	Integer	Number of cores
`queue`	String	Queue name
`walltime`	Time	Maximum run time
`mail_list`	String	Information sent to this email
`x[y,z]_domain%beg[end]`	Real	Beginning [ending] of the x[y,z]-direction domain
`stretch_x[y,z]`	Logical	Stretching of the mesh in the x[y,z]-direction
`a_x[y,z]`	Real	Rate at which the grid is stretched in the x[y,z]-direction
`x[y,z]_a`	Real	Beginning of the stretching in the negative x[y,z]-direction
`x[y,z]_b`	Real	Beginning of the stretching in the positive x[y,z]-direction
`cyl_coord`	Logical	Cylindrical coordinates (2D: Axisymmetric, 3D: Cylindrical)
`m`	Integer	Number of grid cells in the x-coordinate direction
`n`	Integer	Number of grid cells in the y-coordinate direction
`p`	Integer	Number of grid cells in the z-coordinate direction
`dt`	Real	Time step size
`t_step_start`	Integer	Simulation starting time step
`t_step_stop`	Integer	Simulation stopping time step
`t_step_save`	Integer	How often to output data

Table 1: Logistics and computational domain parameters

Parameter	Type	Description
`num_patches`	Integer	Number of initial condition geometric patches
`model_eqns`	Integer	Multicomponent model: [1] Γ/Π_∞; [2] 5-equation; [3] 6-equation
`alt_soundspeed`	Logical	Alternate sound speed and $K\nabla\cdot\boldsymbol{u}$ for 5-equation model
`num_fluids`	Integer	Number of fluids/components present in the flow
`adv_alphan`	Logical	Equations for all N volume fractions (instead of $N-1$)
`mpp_lim`	Logical	Mixture physical parameters limits
`mixture_err`	Logical	Mixture properties correction
`time_stepper`	Integer	Runge–Kutta order [1–5]
`weno_vars`	Integer	WENO reconstruction on [1] Conservative; [2] Primitive variables
`weno_order`	Integer	WENO order [1,3,5]
`weno_eps`	Real	WENO perturbation (avoid division by zero)
`char_decomp`	Logical	Characteristic decomposition
`avg_state`	Integer	Averaged state evaluation method: [1] Roe average; [2] Arithmetic mean
`mapped_weno`	Logical	WENO with mapping of nonlinear weights
`null_weights`	Logical	Null undesired WENO weights
`mp_weno`	Logical	Monotonicity preserving WENO
`riemann_solver`	Integer	Riemann solver algorithm: [1] HLL; [2] HLLC; [3] Exact
`wave_speeds`	Integer	Wave-speed estimation: [1] Direct (Batten et al. 1997); [2] Pressure-velocity (Toro 1999)
`commute_err`	Logical	Commutative error correction via cell-interior quadrature
`split_err`	Logical	Dimensional splitting error correction via cell-boundary
`bc_x[y,z]%beg[end]`	Integer	Beginning [ending] boundary condition in the x[y,z]-direction (negative integer, see table 10)

Table 2: Simulation algorithm parameters

Parameter	Type	Description
`format`	Integer	Output format. [1]: Silo-HDF5; [2] Binary
`precision`	Integer	[1] Single; [2] Double
`parallel_io`	Logical	Parallel I/O
`cons_vars_wrt`	Logical	Write conservative variables
`prim_vars_wrt`	Logical	Write primitive variables
`fourier_decomp`	Logical	Apply a Fourier decomposition to the output variables
`alpha_rho_wrt(i)`	Logical	Add the partial density of the fluid `i` to the database
`rho_wrt`	Logical	Add the mixture density to the database
`mom_wrt(i)`	Logical	Add the `i`-direction momentum to the database
`vel_wrt(i)`	Logical	Add the `i`-direction velocity to the database
`E_wrt`	Logical	Add the total energy to the database
`pres_wrt`	Logical	Add the pressure to the database
`alpha_wrt(i)`	Logical	Add the volume fraction of fluid `i` to the database
`gamma_wrt`	Logical	Add the specific heat ratio function to the database
`heat_ratio_wrt`	Logical	Add the specific heat ratio to the database
`pi_inf_wrt`	Logical	Add the liquid stiffness function to the database
`pres_inf_wrt`	Logical	Add the liquid stiffness to the formatted database
`c_wrt`	Logical	Add the sound speed to the database
`omega_wrt(i)`	Logical	Add the `i`-direction vorticity to the database
`schlieren_wrt`	Logical	Write numerical schlieren
`fd_order`	Integer	Order [1,2,4] finite differences for the numerical Schlieren function
`schlieren_alpha(i)`	Real	Numerical Schlieren computed via `alpha(i)`
`probe_wrt`	Logical	Write the flow chosen probes data files for each time step
`num_probes`	Integer	Number of probes
`probe(i)%x[y,z]`	Real	Coordinates of probe `i`
`com_wrt(i)`	Logical	Add the center of mass of fluid `i` to the database
`cb_wrt(i)`	Logical	Add coherent body data of fluid `i` to the database

Table 3: Formatted database output parameters

Parameter	Type	Description
`alter_patch(i)`	Logical	Alter the `i`-th patch
`geometry`	Integer	Geometry configuration of the patch (see table 11)
`x[y,z]_centroid`	Real	Centroid of the applied geometry in the $x[y,z]$-direction
`length_x[y,z]`	Real	Length, if applicable, in the $x[y,z]$-direction
`radius`	Real	Radius, if applicable, of the applied geometry
`smoothen`	Logical	Smoothen the applied patch
`smooth_patch_id`	Integer	Smoothen of the applied patch with another patch
`smooth_coeff`	Real	Smoothen coefficient
`alpha(i)`	Real	Volume fraction of fluid `i`
`alpha_rho(i)`	Real	Partial density of fluid `i`
`pres`	Real	Pressure
`vel(i)`	Real	Velocity in direction `i`

Table 4: Patch parameters. All parameters should be prepended with `patch_icpp(j)%` where `j` is the patch index.

Parameter	Type	Description
`gamma`	Real	Stiffened-gas parameter Γ of fluid `i`: Specific heat ratio
`pi_inf`	Real	Stiffened-gas parameter Π_∞ of fluid `i`: Liquid stiffness
	Properties used only for non-polytropic bubble compression model	
`mu_l0`	Real	Liquid viscosity (only specify in liquid phase)
`ss`	Real	Surface tension (only specify in liquid phase)
`pv`	Real	Vapor pressure (only specify in liquid phase)
`gamma_v[n]`	Real	Water [air] compression model property (see Ando 2010)
`M_v[n]`	Real	Water [air] compression model property (see Ando 2010)
`mu_v[n]`	Real	Water [air] compression model property (see Ando 2010)
`k_v[n]`	Real	Water [air] compression model property (see Ando 2010)

Table 5: Fluid properties. All parameters should be prepended with `fluid_pp(i)%` where `i` is the fluid index.

Parameter	Type	Description
`Monopole`	Logical	Acoustic source terms
`num_mono`	Integer	Number of acoustic sources
	Properties of acoustic source `i`	
`Mono(i)%pulse`	Integer	Type of pulse. [1] Sine [2] Gaussian [3] Square
`Mono(i)%npulse`	Integer	Number of pulse cycles
`Mono(i)%support`	Integer	Spatial support [1] Delta function [2] Finite width (2D) [3] Support for finite line/patch
`Mono(i)%loc(j)`	Real	Location of source in coordinate direction `j`
`Mono(i)%dir`	Real	Direction of propagation
`Mono(i)%mag`	Real	Pulse magnitude
`Mono(i)%length`	Real	Spatial pulse length

Table 6: Acoustic source terms.

Parameter	Type	Description
`bubbles`	Logical	Ensemble-averaged bubble modeling
`bubble_model`	Integer	[1] Gilmore; [2] Keller–Miksis
`polytropic`	Logical	Polytropic gas compression
`thermal`	Integer	Thermal model: [1] Adiabatic; [2] Isothermal; [3] Transfer
`R0ref`	Real	Reference bubble radius
`nb`	Integer	Number of bins: [1] Monodisperse; [>1] Polydisperse
`Ca`	Real	Cavitation number
`Web`	Real	Weber number
`Re_inv`	Real	Inverse Reynolds number

Table 7: Ensemble-averaged bubble model parameters.

Parameter	Type	Description
`weno_avg`	Logical	Averaged left/right cell-boundary states (for Re and We)
`weno_Re_flux`	Logical	WENO reconstruct velocity gradients for viscous stress tensor
`regularization`	Logical	Regularization algorithm of Tiwari et al. (2013)
`reg_eps`	Real	Interface thickness parameter for regularization terms
`tvd_riemann_flux`	Logical	Apply TVD flux limiter to cell edges inside Riemann solver
`tvd_rhs_flux`	Logical	Apply TVD flux limiter to intercell fluxes outside Riemann solver
`tvd_wave_speeds`	Logical	TVD wave-speeds for flux computation inside Riemann solver
`flux_lim`	Integer	Choice of flux limiter: [1] Minmod; [2] MC; [3] Ospre; [4] Superbee; [5] Sweby; [6] van Albada; [7] van Leer.
`We_riemann_flux`	Logical	Capillary effects in the Riemann solver
`We_rhs_flux`	Logical	Capillary effects using a conservative formulation
`We_src`	Logical	Capillary effects using a non-conservative formulation
`We_wave_speeds`	Logical	Capillary effects when computing the contact wave speed
`lsq_deriv`	Logical	Use linear least squares to calculate normals and curvatures
`alt_crv`	Logical	Alternate curvature definition

Table 8: Experimental features

4 Source code

4.1 Documentation

The source code, located in the `src/` directory, contains three components: `pre_process/`, `simulation`, and `post_process`. These codes are all documented via Doxygen, which can be located at `https://mfc-caltech.github.io`.

4.2 Naming conventions

The Fortran files `*.f90` in the source code directories utilize the naming conventions found in table 9.

Variable	Description
*_sf	Scalar field
*_vf	Vector field
*_pp	Physical parameters
*[K,L,R]	WENO-reconstructed cell averages
*_avg	Roe/arithmetic average
*_cb	Cell boundary
*_cc	Cell center
*_cbc	Characteristic boundary conditions
*_cons_*	Conservative
*_prim_*	Primitive
gm_*	Gradient magnitude
*_ndqp	Normal direction Gaussian quadrature points
*_qp	Cell-interior Gaussian quadrature points
un_*	Unit-normal
dgm_*	Curvature (derived gradient magnitude)
*_icpp	Initial condition patch parameters
*_idx	Indices of first and last (object)
cont_*	Continuity equations
mom_*	Momentum equations
E_*	Total energy equation
adv_*	Volume fraction equations
*_id	Identifier
dflt_*	Default value
*_fp	???
orig_*	Original variable
q_*	Cell-average conservative or primitive variables
q[L,R]_*	Left[right] WENO-reconstructed cell-boundary values
dq_*	First-order spatial derivatives
*_rs	Riemann solver variables
*_src	Source terms
*_gsrc	Geometric source terms
[lo,hi]_*	Related to TVD options
*_IC	Inter-cell
*_ts	Time-stage (for time-stepper algorithm)
wa_*	WENO average
crv_*	Geometrical curvature of the material interfaces

Table 9: Code variables

Acknowledgements

This work was supported in part by multiple past grants from the US National Institute of Health (NIH), the US Office of Naval Research (ONR), and the US National Science Foundation (NSF), as well as current NIH grant number 2P01-DK043881 and ONR grant numbers N0014-17-1-2676 and N0014-18-1-2625. The computations presented here utilized the Extreme Science and Engineering Discovery Environment, which is supported under NSF grant number CTS120005. K.M. acknowledges support from the Funai Foundation for Information Technology via the Overseas Scholarship.

References

Thompson, K. W. (1987). Time dependent boundary conditions for hyperbolic systems, I. *J. Comp. Phys.*, 68:1–24.

Thompson, K. W. (1990). Time dependent boundary conditions for hyperbolic systems, II. *J. Comp. Phys.*, 89:439–461.

A Boundary conditions

	#	Description
Normal	−1	Periodic
	−2	Reflective
	−3	Ghost cell extrapolation
	−4	Riemann extrapolation
	−5	Slip wall
Thompson char.	−6	Non-reflecting subsonic buffer
	−7	Non-reflecting subsonic inflow
	−8	Non-reflecting subsonic outflow
	−9	Force-free subsonic outflow
	−10	Constant pressure subsonic outflow
	−11	Supersonic inflow
	−12	Supersonic outflow

Table 10: Boundary conditions.

The boundary condition supported by the MFC are listed in table 10. Their number (#) corresponds to the input value in `input.py` labeled `bc_x[y,z]%beg[end]` (see table 2). The boundary conditions labeled "Thompson char." are characteristic based and follow Thompson (1987) and Thompson (1990).

B Patch types

#	Name	Dim.	Smooth	Description and required parameters
1	Line segment	1	N	Requires `x_centroid` and `x_length`.
2	Circle	2	Y	Requires `x[y]_centroid` and `radius`.
3	Rectangle	2	N	Coordinate-aligned. Requires `x[y]_centroid` and `x[y]_length`.
4	Sweep line	2	Y	Not coordinate aligned. Requires `x[y]_centroid` and `normal(i)`.
5	Ellipse	2	Y	Requires `x[y]_centroid` and `radii(i)`.
6	Vortex	2	N	Isentropic flow disturbance. Requires `x[y]_centroid` and `radius`.
7	2D analytical	2	N	Assigns the primitive variables as analytical functions.
8	Sphere	3	Y	Requires `x[y,z]_centroid` and `radius`.
9	Cuboid	3	N	Coordinate-aligned. Requires `x[y,z]_centroid` and `x[y,z]_length`.
10	Cylinder	3	Y	Requires `x[y,z]_centroid`, `radius`, and `x[y,z]_length`.
11	Sweep plane	3	Y	Not coordinate-aligned. Requires `x[y,z]_centroid` and `normal(i)`.
12	Ellipsoid	3	Y	Requires `x[y,z]_centroid` and `radii(i)`.
13	3D analytical	3	N	Assigns the primitive variables as analytical functions.
14	Sph. harmonic	3	N	Generates spherical harmonic perturbations to a sphere.

Table 11: Patch geometries

The patch types supported by the MFC are listed in table 11. This includes types exclusive to one-, two-, and three-dimensional problems. The patch type number (#) corresponds to the input value in `input.py` labeled `patch_icpp(j)%geometry` where `j` is the patch index. Each patch requires a different set of parameters, which are also listed in this table.