PFMICE USER GUIDE For 2D Icing Simulation

Wenqiang Zhang
University of Nottingham
wenqiang.zhang@nottingham.ac.uk

September 28, 2021

Contents

1 Overview of PFMICE

PFMICE is a three-dimensional code using Phase-field method for multiphase simulations. It is specificity designed for the flow simulation with phase change, such as the icing process of water droplets on a solid surface. By combining the Cahn-Hilliard model for water-air interface, Allen-Cahn equation for ice and fluid and Navier-Stokes equation for momentum, we solve the evolution of the water-air interface and water-ice interface simultaneously, including the volume expansion associated with solidification and due to the density difference between water and ice. Unlike existing schemes assuming a divergence-free flow field, the proposed continuous formulation allows for density changes while ensuring mass conservation. The solver takes advantage of the consistent and conservative Phase-Field model is developed to study thermogas-liquid-solid flows with liquid-solid phase change. The proposed model is derived with the help of the consistency conditions and exactly reduces to the consistent and conservative Phase-Field method for incompressible two-phase flows, the fictitious domain Brinkman penalization (FD/BP) method for fluidstructure interactions, and the Phase-Field model of solidification of pure material. It honors the mass conservation, defines the volume fractions of individual phases unambiguously, and therefore captures the volume change due to phase change. The momentum is conserved when the solid phase is absent, but it changes when the solid phase appears due to the no-slip condition at the solid boundary. The proposed model also conserves the energy, preserves the temperature equilibrium, and is Galilean invariant. A novel continuous surface tension force to confine its contribution at the gas-liquid interface and a drag force modified from the Carman-Kozeny equation to reduce solid velocity to zero are proposed. The issue of initiating phase change in the original Phase-Field model of solidification is addressed by physically modifying the interpolation function. The corresponding consistent scheme is developed to solve the model, and the numerical results agree well with the analytical solutions and the existing experimental and numerical data. Two challenging problems having a wide range of material properties and complex dynamics are conducted to demonstrate the capability of the proposed model

2 Objective

The test case is a quasi-3D test case. The icing process of a water droplet on a solid surface is simulated. This user guide is used to provide a introduction of how to use the quasi-3D test case. Please notice that this test case is just a sample of PFMICE. However, it contains a few essential subroutines of PFMICE and it is a good example to show the numerical algorithm used in the programme.

3 Phase-field Method

The phase-field method [1] is used to simulate the icing process:

$$\begin{aligned} \frac{\partial \phi}{\partial t} + \bigtriangledown \cdot (u\phi) &= \bigtriangledown \cdot (M_\phi \bigtriangledown \xi_\phi) + \phi \bigtriangledown \cdot u, \\ \xi_\phi &= \lambda_\phi (\frac{1}{\eta_\phi^2} g'(\phi) - \bigtriangledown^2 \phi), \\ \lambda_\phi = 3\sqrt{2}\eta_\phi \sigma, g(\phi) &= \phi^2 (1-\phi)^2, \\ m_\phi &= u\phi - M_\phi \bigtriangledown \xi_\phi. \end{aligned} \tag{1}$$

where ϕ is the phase-field parameter to indicate the volume fraction of liquid in a control volume. u is the velocity, M_ϕ and ξ_ϕ are the mobility and chemical potential of ϕ. λ_ϕ is the mixing energy density of ϕ, η_ϕ is the liquid-gas interface thickness and σ is the surface tension.
The Allen-Cahn equation:

$$\begin{aligned}
\frac{\partial(\phi c)}{\partial t} + \bigtriangledown \cdot (m_\phi c) &= -M_c \xi_c + \phi c \bigtriangledown \cdot u, \\
\xi_c = -\lambda_c \bigtriangledown \cdot (\phi \bigtriangledown c) + \frac{\lambda_c}{\eta_c^2} \phi g'(\phi) &+ \frac{\rho_L L}{T_M} \phi P'(\phi)(T_M - T), \\
\lambda_c = \frac{3\sqrt{2}\rho_L L}{T_M} \Gamma_c \eta_c, M_c &= \frac{\mu_c \Gamma_c}{\lambda_c}, \\
p(c) = c^3(6c^2 - 15c + 10), \tilde{p}'(c) &= \begin{cases} 1, c = 0 \; and \; T \geq T_M \\ 1, c = 1 \; and \; T \leq T_M \\ p'(c), \; else \end{cases} \\
m_{\phi c} &= m_\phi c.
\end{aligned} \tag{2}$$

Here, c is the phase-field parameter to denote the volume fraction of solid in a liquid-solid mixture. M_c is the mobility and ξ_c is the chemical potential. T is the temperature and T_M is the melting temperature of the solid. $m_{\phi c}$ is the phase-field flux of ϕc.
The projection method is used to solve the momentum equation.

$$\begin{aligned}
\frac{\partial(\rho u)}{\partial t} + \bigtriangledown \cdot (m_\rho \otimes u) &= - \bigtriangledown P + \bigtriangledown \cdot [\mu(\bigtriangledown u + \bigtriangledown u^T)] + \rho g + f_s + f_d, \\
f_s = \phi \xi_\phi \bigtriangledown \phi, \; f_d = A_d(u_s - u), \; A_d &= C_d \frac{a_s^2}{(1 - a_s)^3 + e_d}, \\
a_s &= (\phi - \phi c).
\end{aligned} \tag{3}$$

In the momentum equation, P is the pressure and g is the gravitational constant. f_s and f_d are the surface tension at the liquid-gas interface and drag force, respectively. The drag force is to suppress the movement of the solid phase. e_d can be a very small number such as 1.0^{-3}.
The energy equation is as follows:

$$\frac{\partial(\rho C_p)T}{\partial t} + \bigtriangledown \cdot (m_{\rho C_p} T) + \rho_L L(\frac{\partial(\phi c)}{\partial t} + \bigtriangledown \cdot m_{\phi c}) = \bigtriangledown \cdot (\kappa \bigtriangledown T) + Q_T. \tag{4}$$

The physical properties of the mixture are obtained through linear interpolation:

$$\begin{aligned}
\rho &= \rho_g + (\rho_{s-\rho_g})\phi + (\rho_L - \rho_s)(\phi c), \\
\mu &= \mu_g + (\mu_{s-\mu_g})\phi + (\mu_L - \mu_s)(\phi c), \\
\rho C_p &= (\rho C_p)_g + ((\rho C_p)_s - (\rho C_p)_g)\phi + ((\rho C_p)_L - (\rho C_p)_s)(\phi c),
\end{aligned} \tag{5}$$

4 Numerical Scheme

The numerical scheme of the code is shown in the following figure:

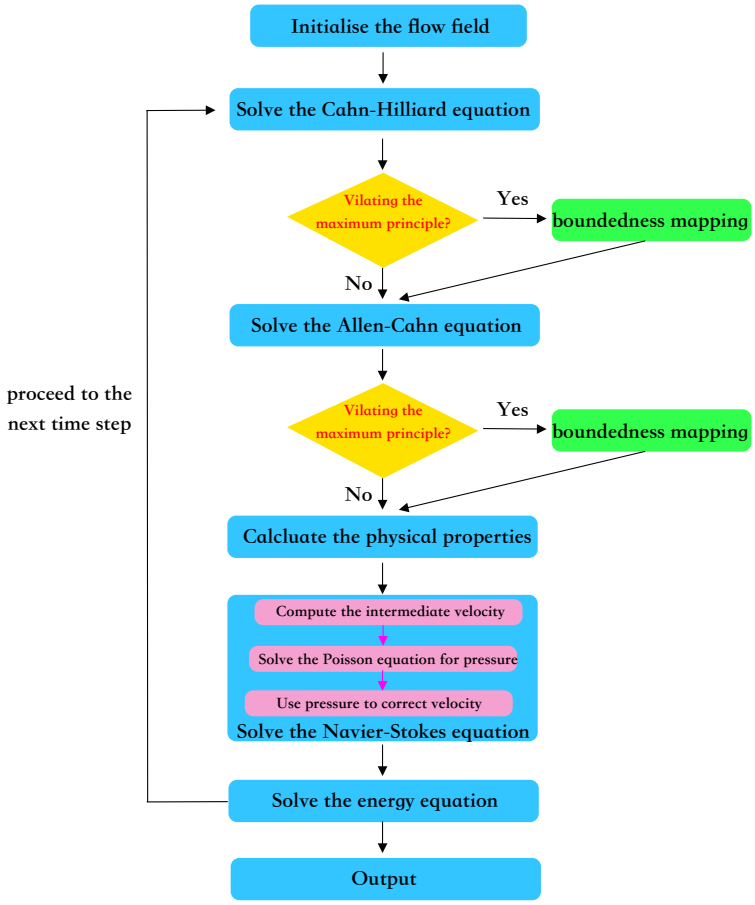

Figure 1: Numerical scheme of PFMICE

A brief introduction of the numerical scheme is as follows:

a. Calculate the value of ϕ and c of time step $n+1$ with Eq. 1 and Eq. 2.

b. Obtain the density, thermal conductivity coefficient and viscosity at time step $n+1$ using Eq.5.

c. Advance the momentum equations to obtain the approximate velocity $\vec{\mathbf{U}}^*$ at time step $n+1$

d. Compute P^{n+1} by solving the Poisson equation introduced below

e. Compute U^{n+1} using P^{n+1} and $\vec{\mathbf{U}}^*$

f. Calculate T^{n+1} using the second-order Crank-Nicolson scheme for the energy equation

g. Correct the velocity of ice and proceed to the next time step

To solve the momentum equation 3, we split it into two equations:

$$\frac{\rho^{n+1}u^* - \rho^n u^n}{\Delta t} = RU^n \tag{6}$$

and

$$\frac{u^{n+1}-u^*}{\Delta t} = -\frac{1}{\rho^{n+1}} \bigtriangledown p^{n+1} \tag{7}$$

thus we have:

$$u^{n+1} = u^* - \frac{\Delta t}{\rho^{n+1}} \bigtriangledown p^{n+1} \tag{8}$$

by taking divergence of 8 we can get:

$$\bigtriangledown \cdot u^{n+1} = \bigtriangledown \cdot u^* - \Delta t \bigtriangledown \cdot (\frac{1}{\rho^{n+1}} \bigtriangledown p^{n+1}), \tag{9}$$

With the split method proposed in 2, the Poisson equation for the pressure is:

$$\bigtriangledown^2 p^{n+1} = \bigtriangledown \cdot (1 - \frac{\rho_g}{\rho^{n+1}} \bigtriangledown \hat{p}) + \frac{\bigtriangledown \cdot u^* - \bigtriangledown \cdot u^{n+1}}{\triangle t}. \tag{10}$$

When there is phase change [1],

$$\bigtriangledown \cdot u^{n+1} = \frac{M_c(\rho_L - \rho_s)}{\rho^{n+1}} \xi_c. \tag{11}$$

The energy equation is solved with Alternating direction methods for three space variables [3].

5 Installation

5.1 Prerequisites

PFMICE is written with Fortran. Before you try to install PFMICE on your local computer or workstation, Fortran compiler (GCC or Intel Fortran) and MPI libraries (e.g., openmpi or Intel MPI) must be installed. OpenMPI is freely available online:
`https://www.open-mpi.org/software/ompi/v4.1/`.
and it can be installed in the following steps:

```
shell$ gunzip -c openmpi-4.1.1.tar.gz | tar xf -
shell$ cd openmpi-4.1.1
shell$ ./configure --prefix=/usr/local
<...lots of output...>
shell$ make all install
```

Intel oneAPI Toolkits can be found in:
`https://software.intel.com/content/www/us/en/develop/tools/oneapi/all-toolkits.html#gs.cpa4px`.

5.2 Compiling PFMICE

A Makefile can be found in the folder. If your current compiling environment is GCC+OpenMPI, in the Makefile, you should use:

```
FC = mpif90
FFLAGS := -O3 -ffixed-line-length-none -mcmodel=large -fdefault-real-8 -cpp -Wall -fcheck=all
```

For Intel compiler+Intel MPI, the setup in the Makefile will be:

```
FC = mpiifort
FFLAGS := -r8 -fpconstant -O3 -132 -cpp
```

To compile the code, you need to compile the libraries at first with:

```
make libraries
```

and then compile the source code with:

```
make
```

you can remove the executables with

```
make clean
```

To run the code:

```
mpirun -np N PFMICE,
```

where N is the number of CPU cores.

6 Libraries

The code takes advantage of the 2DCOMP&FFT libraries developed by N. Li and S. Laizet [4]. The 2D pencil decomposition can split the computational domain in two dimensions. States (a), (b) and (c) are referred to as X-pencil, Y-pencil and Z-pencil arrangements, respectively. This facilitates the parallel computation by applying serial algorithms in local memory with data transposition procedures. This libraries is validated on different platforms and it maintains high parallel performance up to thousands of CPU cores.

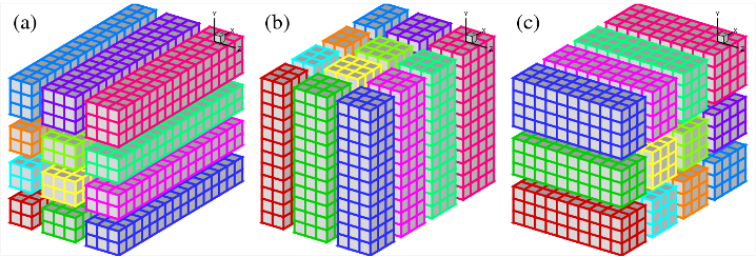

Figure 2: Numerical scheme of PFMICE [4]

The second libraries is the (fast Fourier transform) FFT programme which can be found in `http://www.jjj.de/fft/glassman-fft.f`. The developer has noticed that the library is out of date and will switch to the other FFT libraries in the future.

7 Subroutines

7.1 constants.f90

This subroutine contains the parameters used in the simulation. It can be classified into three categories: computational domain, the physical properties of the flow field (e.g., air, water and ice) and interface properties of the solid.

7.2 vars.f90

The flow variables (e.g., velocity, temperature and phase-field parameters) are defined in this subroutine. The variables used for MPI are also stored in this subroutine.

7.3 parallel.f90

This subroutine is applied to init the parallel computation.

7.4 init.f90

This subroutine is used to define the initial flow field. Typically, the phase field parameters are defined to describe the range of different phases. The initial velocity and temperature of the flow field are given. In this example, the flow field contains a waterdroplet in the air with its bottom attached to the solid wall. The waterdroplet is surround by the air. This is shown in Fig. 4.

7.5 field.f90

This subroutine is used to compute the properties of the multi-phase mixture in the computational domain. As each equation in the whole system was computed separately, the sequence to compute different mixture properties is also different.

7.6 bou.f90

This subroutine is a collection of the boundary conditions used in the computation. Generally, there are four kinds of boundary conditions: Dirichlet boundary condition, Neumann boundary condition, periodic boundary condition and inlet/outlet boundary condition. For this test case, the left, right, front and back surfaces uses periodic boundary condition; the bottom wall is set as the Neumann boundary condition, the top surface is an outlet. The slippery boundary condition can be used for the bottom wall in this case [5].

7.7 loadd.f90

This subroutine is mainly used to write out and read solution files used for restart the computation.

7.8 conserve.f90

This subroutine is used to compute the conservative variables used in the phase-field method. The computation method of these conservative variables can be very different between the explicit scheme and the implicit scheme.

7.9 phi.f90

This subroutine is used to solve the Cahn-Hillard equation to obtain the ϕ value.

7.10 mapc.f90

This subroutine is used to check if boundedness mapping is needed.

7.11 chem.f90

This subroutine computes the chemical potential used in the Cahn-Hilliard equation.

7.12 phi_RHS.f90

This subroutine is used to compute the RHS of the Poisson equation for boundedness mapping. The procedure to carry out the boundedness mapping can be found in [6].

7.13 revise.f90

This subroutine is used with fillphi.f90, after the Q value is computed through boundedness mapping, the flux is corrected to make the computation of ϕ and c consistent.

7.14 c.f90

This subroutine is used to solve the Allen-Cahn equation to obtain the c value.

7.15 chknan.f90

This subroutine is used to check if the solution becomes NaN during the computation. If NaN appears during the simulation, the programme will generate an alarm to show where the NaN is produced and stop the computation.

7.16 uvwstar.f90

This subroutine is used to compute the intermediate velocity with the projection method.

7.17 prefft.f90

As the FFT solver is used to compute the pressure in the Poisson equation, this subroutine is to init the FFT solver. By computing some constants beforehand, this subroutine can avoid some repeated computations and reduce the running time.

7.18 p_RHS.f90

This subroutine is used to compute the RHS of the Poisson equation for the pressure.

7.19 fastsolver.f90

This is the FFT solver used in the programme. It is used for the Poisson equation of pressure as well as in the boundedness mapping.

7.20 correc.f90

After the pressure is obtained, the velocity of time step $n + 1$ is corrected by the pressure in this subroutine.

7.21 temp.f90

This subroutine solves the energy equation explicitly and obtain the temperature of the flow field.

7.22 energy.f90

This subroutine solves the energy equation implicitly and obtain the temperature of the flow field. By adopting the implicit scheme, the time step size can be much larger. Alternating direction implicit (ADI) scheme is used.

7.23 vtk_write.f90

This subroutine is used to output the solution. It can write out .vtk file or .dat file. Paraview can open the .vtk file and .dat file is the format of Tecplot.

7.24 debug.f90

This subroutine can output some important parameters for debug purpose. The user can determine which parameters to be output.

7.25 thomas.f90

This subroutine is used to solve the tridiagonal matrix with Thomas scheme.

7.26 triperiodic.f90

Thomas Algorithm for Periodic Tridiagonal Systems. This subroutine is used when periodic boundary condition is used for the simulation. It transfer a coefficient matrix A (as shown below)

$$\begin{vmatrix} a_1 & c_1 & & & b_1 \\ b_2 & a_2 & c_2 & 0 & \\ & ... & ... & ... & \\ & & ... & ... & c_N \\ c_{N+1} & 0 & & b_{N+1} & a_{N+1} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{vmatrix} = \begin{vmatrix} q \end{vmatrix} \tag{12}$$

into a tridiagonal matrix and solve the matrix with the following steps:
1. condensing the matrxi by eliminating the last row and the last column to get:

$$\begin{vmatrix} a_1 & c_1 & & \\ b_2 & a_2 & c_2 & 0 \\ & \ddots & \ddots & c_{N-1} \\ 0 & & b_N & a_N \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ ... \\ x_N \end{vmatrix} = \begin{vmatrix} q \end{vmatrix} - \begin{vmatrix} b_1 \\ 0 \\ \vdots \\ 0 \\ c_N \end{vmatrix} x_{N+1} \tag{13}$$

2. the solution of the original equation x is a superposition of the form:

$$x = x^1 + x^2 \cdot x_{N+1} \tag{14}$$

where x^1 and x^2 are solutions of the tridiagonal "condensed" system with N unknowns, i.e.,

$$[A^c][x^{(1)}] = [q] \tag{15}$$

$$[A^c][x^{(2)}] = \begin{vmatrix} -b_1 \\ 0 \\ \vdots \\ 0 \\ -c_N \end{vmatrix} \tag{16}$$

3. x_{N+1} is solved with:

$$x_{N+1} = \frac{q_{N+1} - c_{N+1}x_1^1 - b_{N+1}x_N^1}{a_{N+1} + c_{N+1}x_1^2 + b_{N+1}x_N^2} \tag{17}$$

4. the final solution is obtained with 14.

7.27 main.f90

The main subroutine is used to control the simulation.

8 Case Setup

The velocity are set on the surface of the grid while all the other parameters (e.g., pressure, phase-field parameters and physical properties) are set in the cell centre.

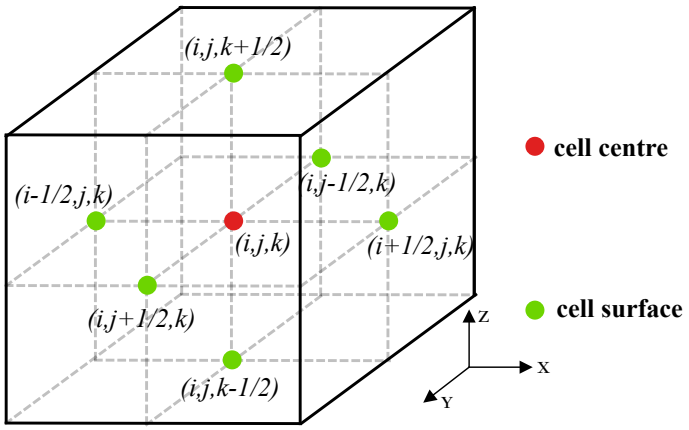

Figure 3: The staggered grid used in the simulation

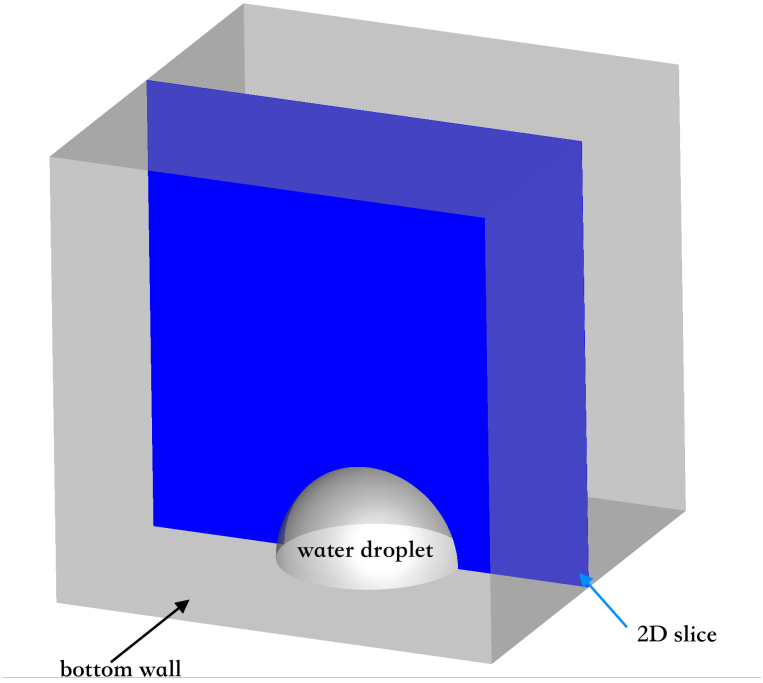

Figure 4: The waterdroplet on a solid surface

9 Acknowledgement

The author would like to thank the developers of 2DECOMP&FFT library (http://www.2decomp.org/) and CaNS (https://github.com/p-costa/CaNS), PFMICE benefits directly or indirectly from these open source code in the early stage development.

References

[1] Ziyang Huang, Guang Lin, and Arezoo M Ardekani. A consistent and conservative phase-field model for thermo-gas-liquid-solid flows including liquid-solid phase change. *arXiv preprint arXiv:2102.06863*, 2021.

[2] Michael S Dodd and Antonino Ferrante. A fast pressure-correction method for incompressible two-fluid flows. *Journal of Computational Physics*, 273:416–434, 2014.

[3] Jim Douglas. Alternating direction methods for three space variables. *Numerische Mathematik*, 4(1):41–63, 1962.

[4] N Li and S Laizet. 2decomp&fft–a highly scalable 2d decomposition library and fft interface, cray user group 2010 conference, edinburgh. *URL http://www. 2decomp. org/pdf/17B-CUG2010-paper-Ning_Li. pdf*, 2010.

[5] Xianmin Xu, Yana Di, and Haijun Yu. Sharp-interface limits of a phase-field model with a generalized navier slip boundary condition for moving contact lines. *Journal of Fluid Mechanics*, 849:805–833, 2018.

[6] Ziyang Huang, Guang Lin, and Arezoo M Ardekani. Consistent and conservative scheme for incompressible two-phase flows using the conservative allen-cahn model. *Journal of Computational Physics*, 420:109718, 2020.