Embedded Boundary Methods for Solving Elliptic Equations And Application in MHD Simulation

Du, Jian

Advisor : James Glimm & Roman Samulyak

Abstract

- MHD system of equations and approximations
- Numerical algorithm and Front tracking for free surface flows
- Former work done
- Embedded Boundary Elliptic Solver for complex domains
- Algorithm and Validation
- Applications and Current Work

1. MHD system of equations

Full system of MHD equations

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{u})$$

$$\rho\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right)\mathbf{u} = -\nabla P + \mu\Delta\mathbf{u} + \frac{1}{c}(\mathbf{J}\times\mathbf{B})$$

$$\rho\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right)e = -P\nabla\cdot\mathbf{u} + \frac{1}{\sigma}\mathbf{J}^2$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla\times(\mathbf{u}\times\mathbf{B}) - \nabla\times\left(\frac{c^2}{4\pi\sigma}\nabla\times\mathbf{B}\right)$$

$$P = P(\rho, e), \qquad \nabla\cdot\mathbf{B} = 0$$

Low magnetic Re approximation & charge neutrality

$$\nabla\cdot J + \frac{\partial \rho_e}{\partial t} = 0$$

$$\mathbf{J} = \sigma\left(-\nabla\phi + \frac{1}{c}\mathbf{u}\times\mathbf{B}\right)$$

$$\Delta\phi = \frac{1}{c}\nabla\cdot(\mathbf{u}\times\mathbf{B}),$$

$$\text{with } \left.\frac{\partial\phi}{\partial\mathbf{n}}\right|_{\Gamma} = \frac{1}{c}(\mathbf{u}\times\mathbf{B})\cdot\mathbf{n}$$

$$\mathbf{B} = \mathbf{B}_{\text{ext}}(x,t), \quad \nabla\cdot\mathbf{B}_{\text{ext}} \equiv 0$$

2. Numerical simulations and Front tracking

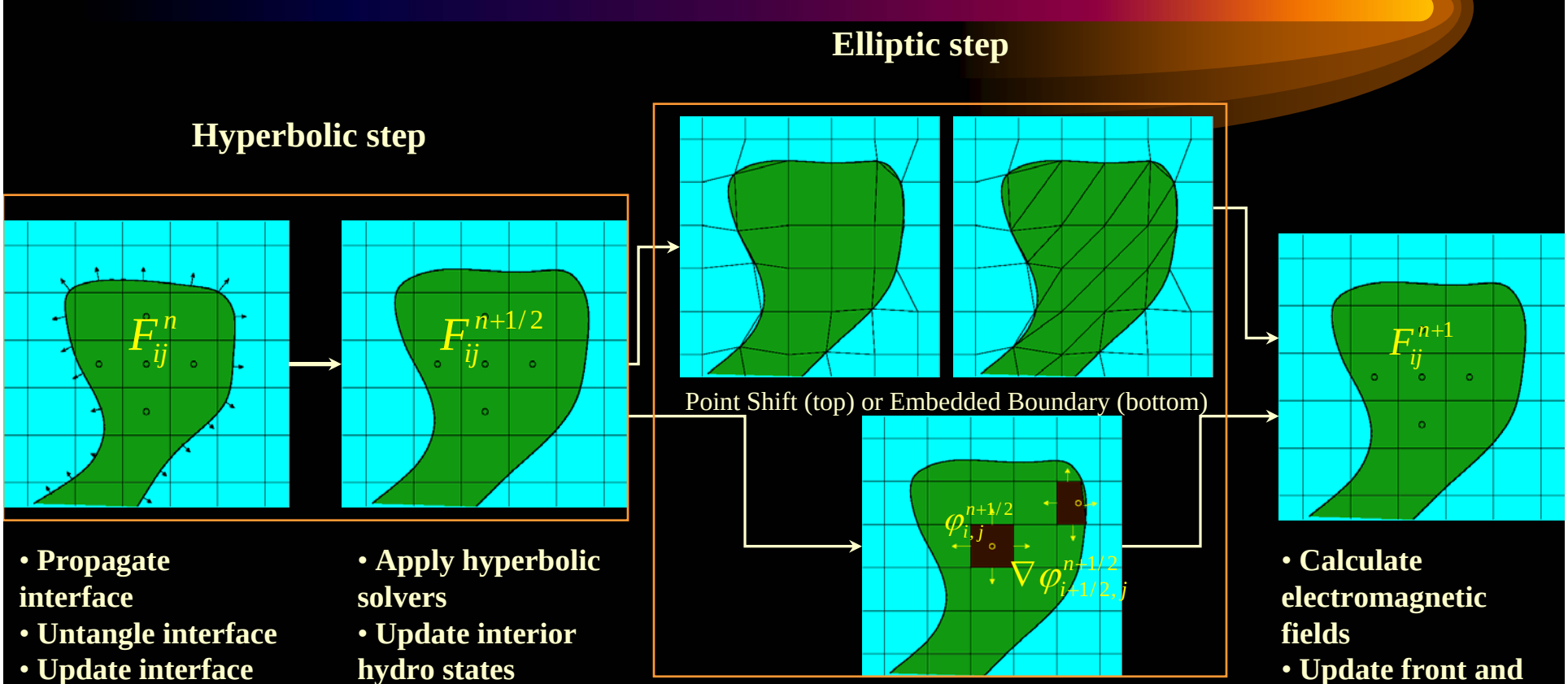
- The low magnetic Re MHD is a coupled hyperbolic/elliptic system. Operator splitting.

- The hyperbolic subsystem is solved on a finite difference grid in both domains separated by the free surface using front tracking numerical techniques.
    - Implemented in FronTier code
    - Riemann problem for interface propagation
    - Complex interfaces with topological changes in 2D and 3D
    - High resolution hyperbolic solvers
    - Realistic EOS models

- The elliptic subsystem is solved in geometrically complex domains

    - Embedded boundary finite volume discretization
    - Fast parallel linear solvers

2. Numerical simulations and Front tracking



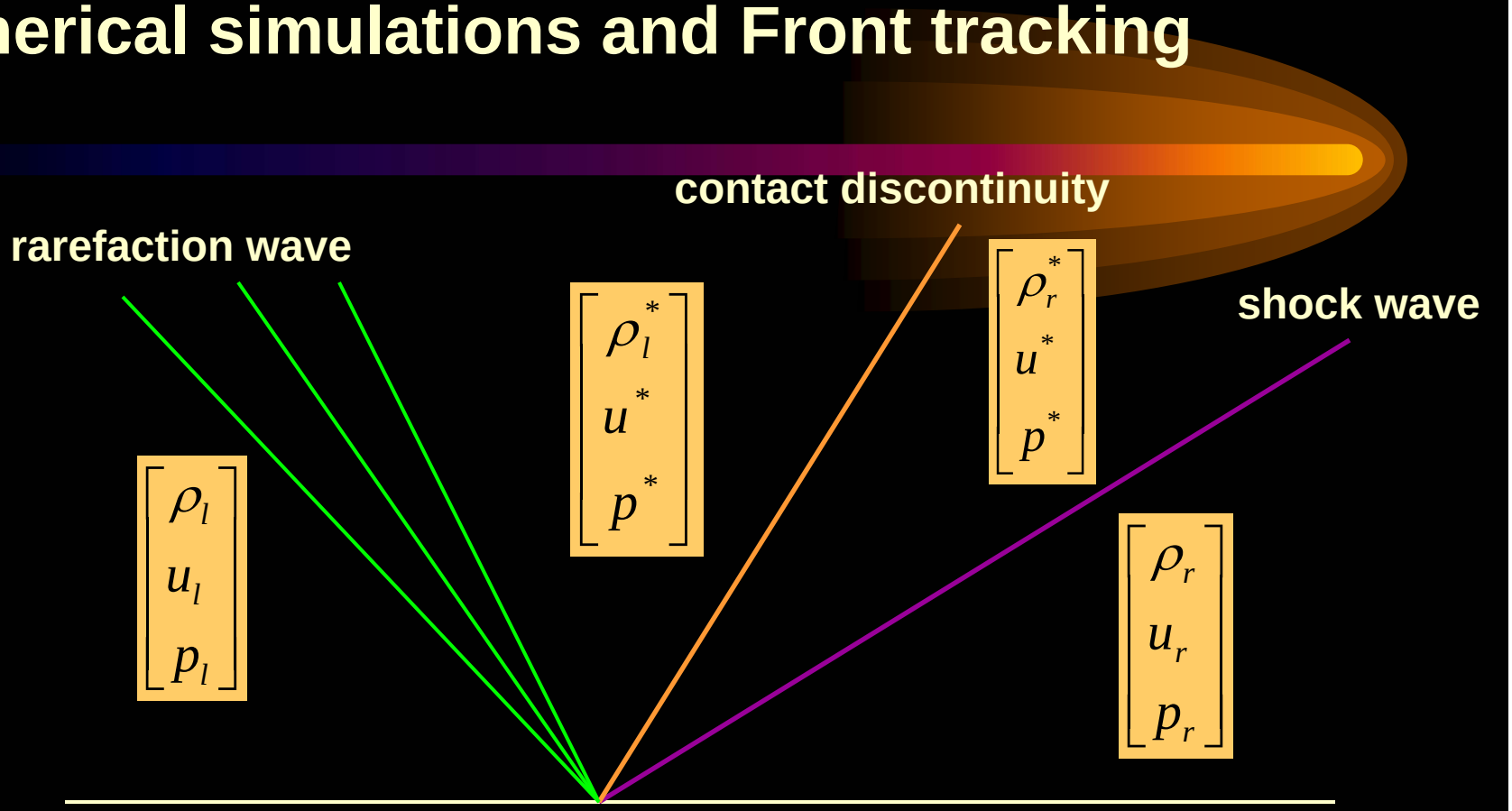
- Propagate interface
- Untangle interface
- Update interface states

- Apply hyperbolic solvers
- Update interior hydro states

- Generate finite element grid
- Perform mixed finite element discretization
  or
- Perform finite volume discretization
- Solve linear system using fast Poisson solvers

- Calculate electromagnetic fields
- Update front and interior states

2. Numerical simulations and Front tracking



1. Front propagation is done by solving generalized Riemann problems locally

2. States on the spatial grid is updated with high order finite difference schemes (MUSCL)

3. Former Work Done

1. **Development of FronTier-MHD code**

   *** treat the hyperbolic-elliptic coupled system in the operator splitting manner**

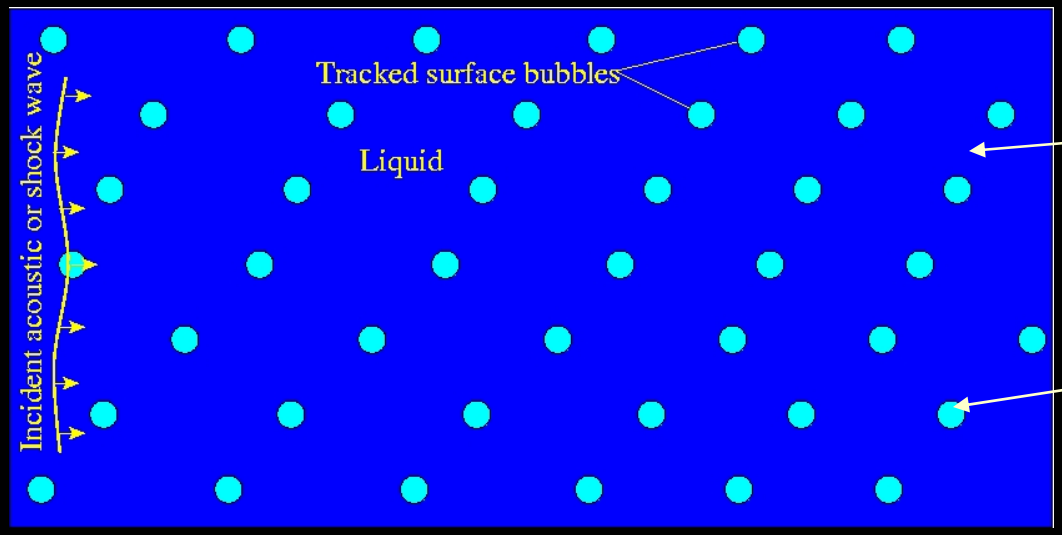
   *** solve the poison's equation with several techniques**

- **Study of the stabilizing effect of the magnetic field on the mercury jet interacting with proton pulses**

3. **Study of the jet distortion in the magnetic field**

3. Former Work Done

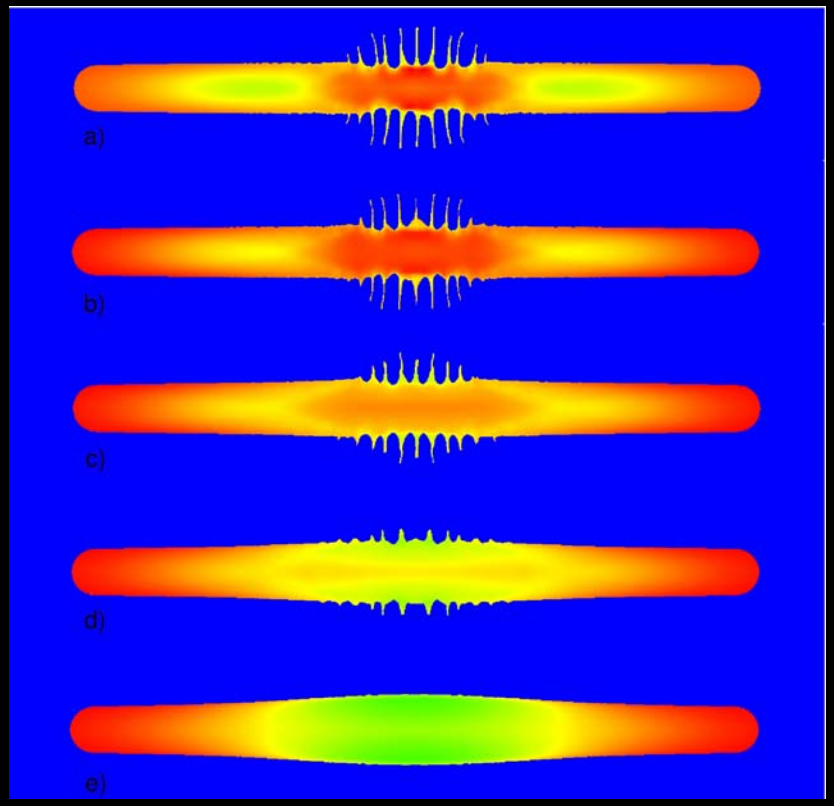
- **Heterogeneous method (Direct Numerical Simulation):** Each individual bubble is explicitly resolved using FronTier interface tracking technique.



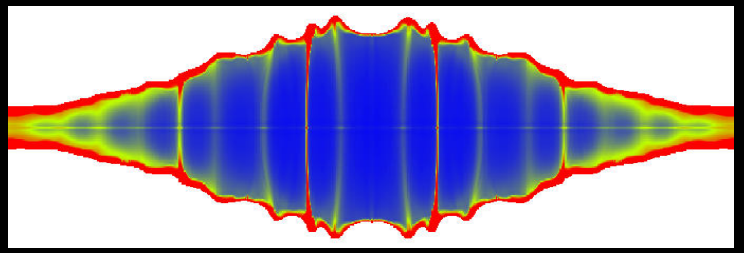
- **Homogeneous EOS model.** Suitable average properties are determined and the mixture is treated as a pseudofluid that obeys an equation of single-component flow.

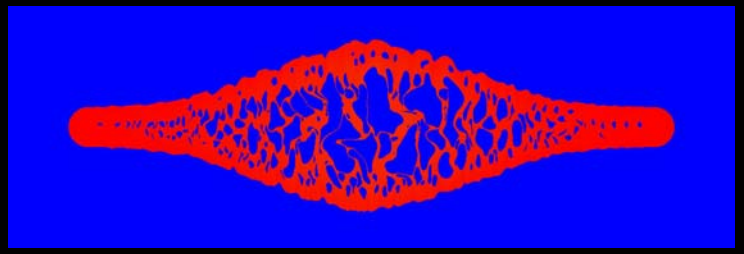
3. Former Work Done

a) B = 0 b) B = 2T c) B = 4T
d)B = 6T e) B = 10T



Homogeneous (upper) and
Heterogeneous EOS (lower)
with dynamic cavitation





400 microseconds

4. Embedded Boundary Elliptic Solver

(1) Why EB

> Point-shift grid generation and finite element discretization method

- **Second order accurate for gradients**
- **Compatible with mixed finite element formulation**
- **Capable of generating grids for vector finite elements**
- **Not robust (especially in 3D)**

> **EB**

- **Advantages of dealing with complex geometric domains**
- **second-order accuracy of solution and robust**
- **Trivial work to implement the algorithm in parallel computing**

4. Embedded Boundary Elliptic Solver

(2) Main Points

- Based on the finite volume discretizations
- Potential is treated as cell centered value, even if the center is outside the computational domain
- Domain boundary is embedded in the rectangular Cartesian grid, and solution is treated as a cell-centered quantity
- Using finite difference for full cell and linear interpolation for cut cell flux calculation

4. Embedded Boundary Elliptic Solver

(3) Principles

$$\begin{cases} \nabla \bullet \nabla \varphi = \rho \\ \dfrac{\partial \varphi}{\partial \vec{n}} \Big|_{\Gamma} = F \end{cases} \Rightarrow \lim_{area \to 0} \frac{\oint \nabla \varphi \cdot ds}{area} = \lim_{area \to 0} \frac{\sum_i f_i \cdot d\vec{l}_i + f^f \cdot d\vec{l}_f}{area}$$

- **Area is corresponding to the computational cell on which to make integration**

- **f_i is the flux across the non-boundary cell edges and f^f is the boundary edge flux given by Neumann Conditions. All fluxes are calculated at the middle point of the edge and dl is the related edge length**

4. Embedded Boundary Elliptic Solver

$$F_j = \frac{\varphi_{i+1,j} - \varphi_{i,j}}{\Delta x}; F_{j+1} = \frac{\varphi_{i+1,j+1} - \varphi_{i,j+1}}{\Delta x}$$

$$F = \frac{1+a}{2} F_j + \frac{1-a}{2} F_{j+1}; a = \frac{ef}{eg}$$

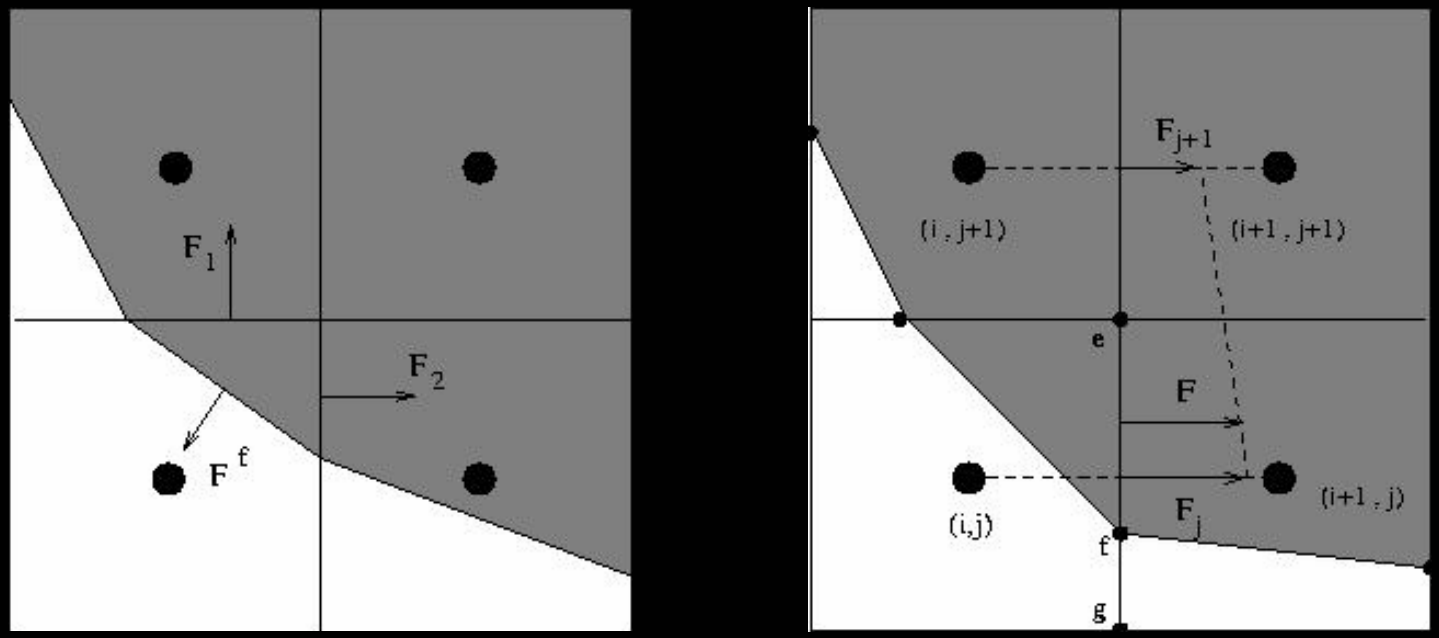


Figure 1. **Flux Discretization**

4. Embedded Boundary Elliptic Solver

(4) Interface Reconstruction and Assumptions

* The number of intersection of each cell with the boundary curve must be either 0 or 2
  ➔ three possible partial cell configurations

* Shift the interface point away from cell boundary
  ➔ remove small volume cells

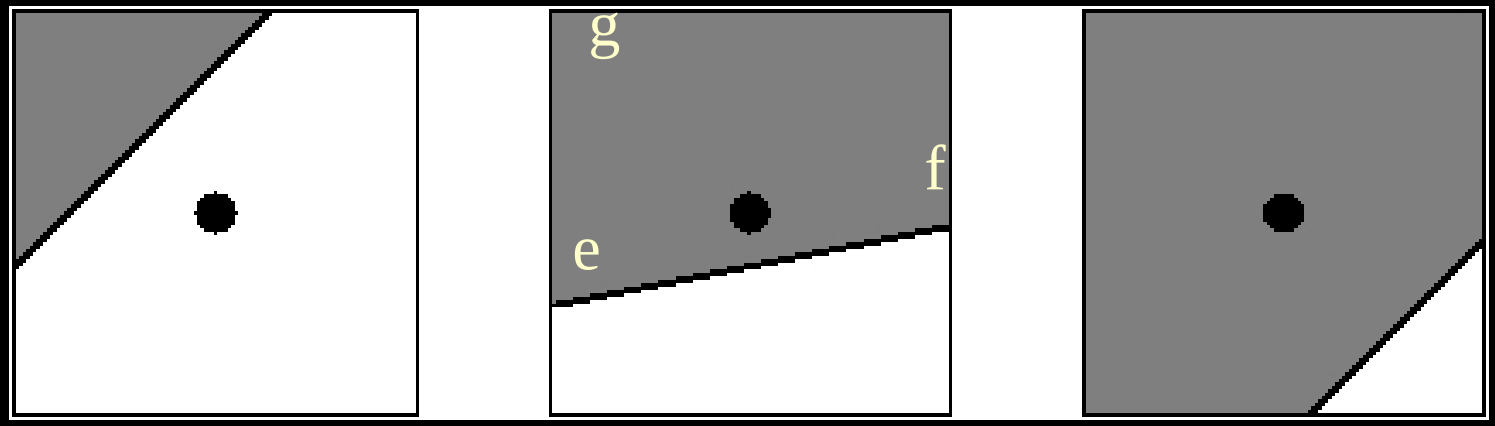


Figure 2. Partial cell configurations

5. Algorithm and Validation

(1) Algorithm

* Reconstruct the interface, record the crossings, and set the component type for grid point

* Count the local number of blocks (both full and partial cells), set the matrix and vector dimension for the linear system solver,set global index for the counted blocks

* For partial cells, the necessary information such as cell area, edge aperture, and boundary normal direction are recorded. Also from the crossing positions and cell corner components, the configuration of the cell can be uniquely determined.

* A nine-point stencil is set for each partial cell. With the above information, related stencil value is set and inserted into the corresponding matrix row.

* Set up the right hand side for the linear system, which is located at the centroid of each cell. Note that for partial cell, RHS also needs to include the boundary flux from left side.

* Solve the linear system Ax = b with some fast parallel linear solver (PETSc or HYPRE)

5. Algorithm and Validation

(2). Stencil Setting

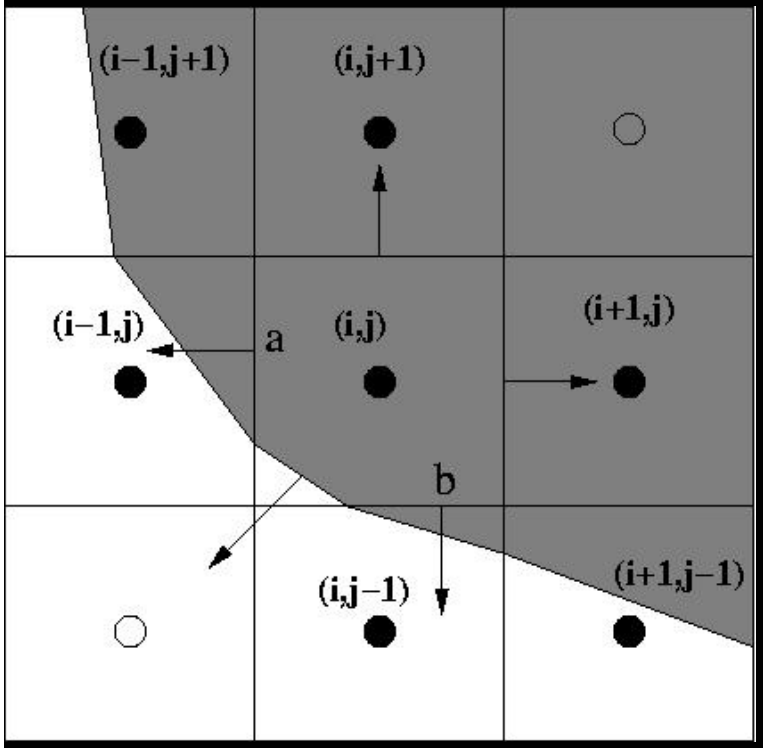


Figure 3. Stencil for partial cells

$$c(i,j) = -\frac{hx}{hy} - \frac{hy}{hx} - \frac{b(1+b)hx}{2hy} - \frac{a(1+a)hy}{2hx}$$

$$c(i,j-1) = \frac{b(1+b)hx}{2hy}; \quad c(i-1,j) = \frac{a(1+a)hy}{2hx}$$

$$c(i+1,j-1) = \frac{b(1-b)hx}{2hy}; \quad c(i-1,j+1) = \frac{a(1-a)hy}{2hx}$$

$$c(i,j+1) = \frac{hx}{hy} - \frac{a(1-a)hy}{2hx}; \quad c(i+1,j) = \frac{hy}{hx} - \frac{b(1-b)hx}{2hy}$$

$$\frac{1}{area}\begin{bmatrix}\cdots & c(i-1,j) & c(i,j) & c(i+1,j) & \cdots\end{bmatrix}$$

$$\begin{pmatrix}\vdots\\ \varphi_{i,j-1}\\ \varphi_{i+1,j-1}\\ \varphi_{i-1,j}\\ \varphi_{i,j}\\ \varphi_{i+1,j}\\ \varphi_{i-1,j+1}\\ \varphi_{i,j+1}\\ \vdots\end{pmatrix}$$

5. Algorithm and Validation

(3) PETSc Introduction

* Stands for the Portable, Extensible Toolkit for Scientific Computation (PETSc)

* Parallel, non-trivial PDE solvers that deliver high performance and provide a distinctly object-oriented interface

* Features an easy-to-use interface to the combination of a Krylov subspace iterative method (Chebychev,GMRES …) and a preconditioner (LU factorization, block Jacobi …)

* By removing the null space of coefficient matrix from the range of the Krylov subspace, thus enables the solving of singular system
(Cond(A) 1e7 => 1e3)

5. Algorithm and Validation

(4) Convergence Test

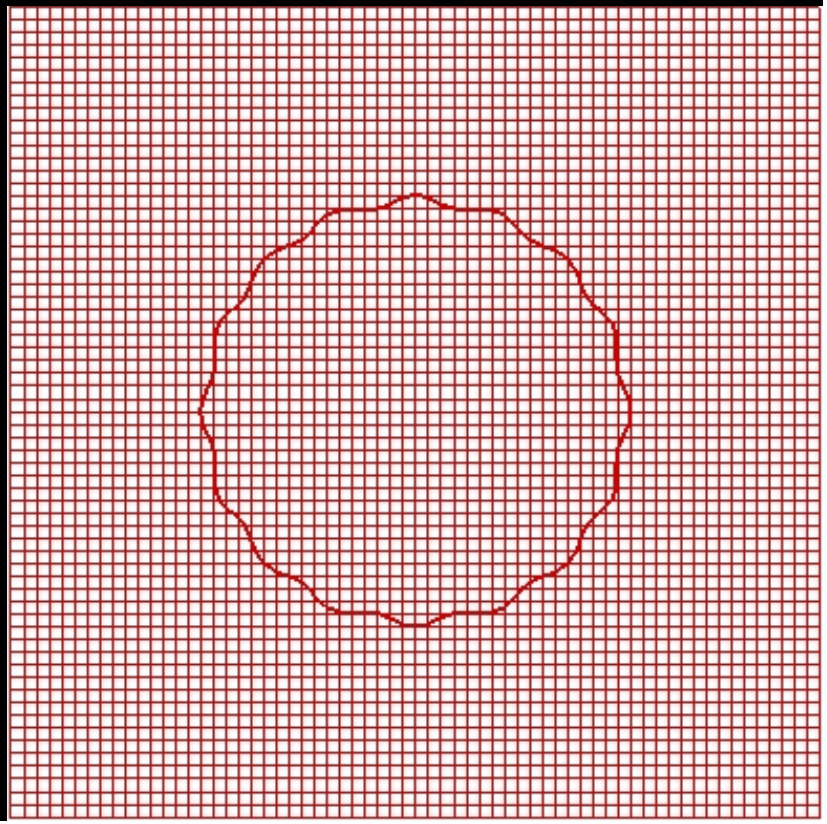
Test Function: $\varphi = e^{k_1 X^2 + k_2 Y^2}$

Domain Boundary: Sin wave perturbed circle

Convergence Rate: $R = \dfrac{\ln(\|e_2\| / \|e_1\|)}{\ln(h_2 / h_1)}$

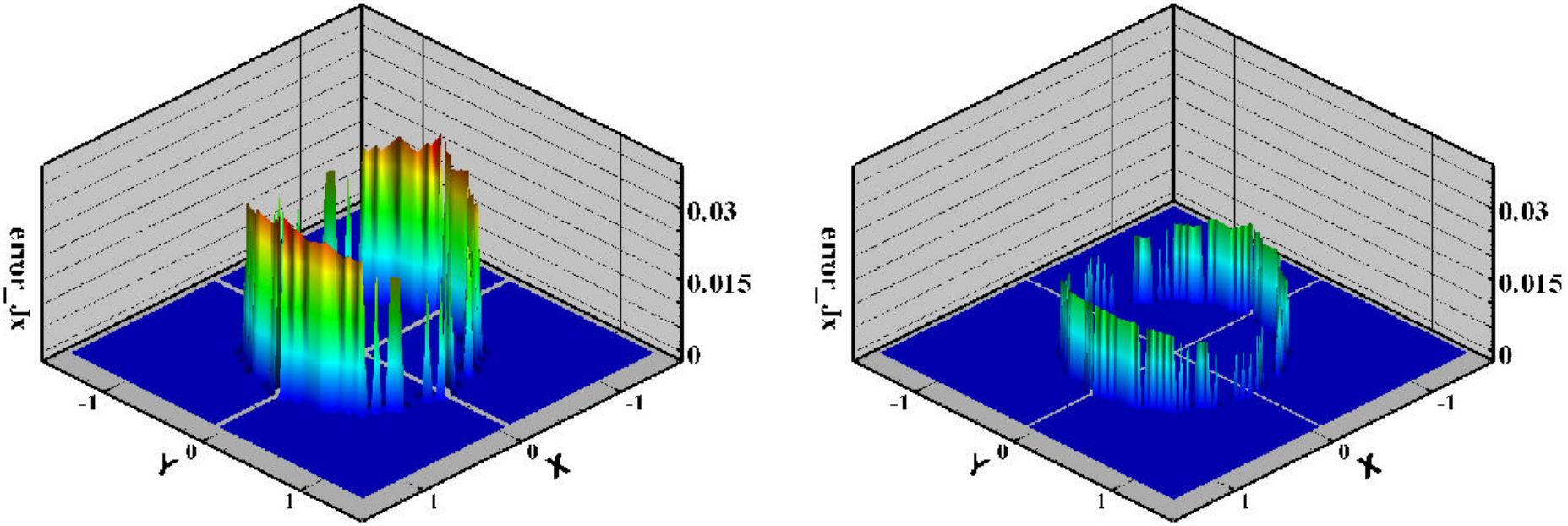
Mesh Size	error (fx)	R
32x32	2.496149e-02	N/A
64x64	8.840943e-03	1.497
128x128	3.093470e-03	1.506
256x256	1.095710e-03	1.503

Figure 4. **Computational Domain**



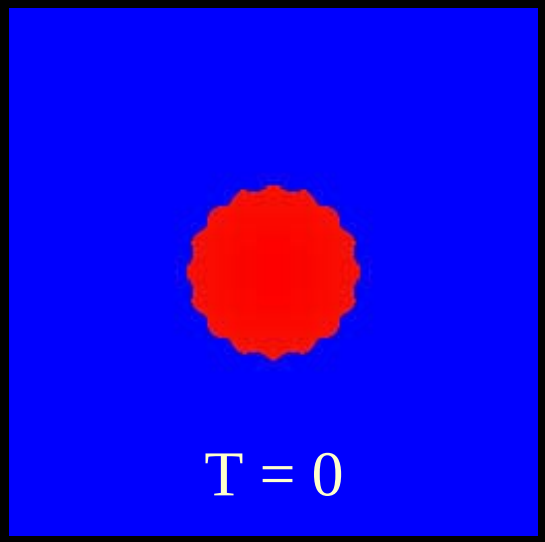
5. Algorithm and Validation

Figure 5. **Illustration of flux error (X direction)**



6. Applications

(1) Mercury Jet simulation with S2phase EOS



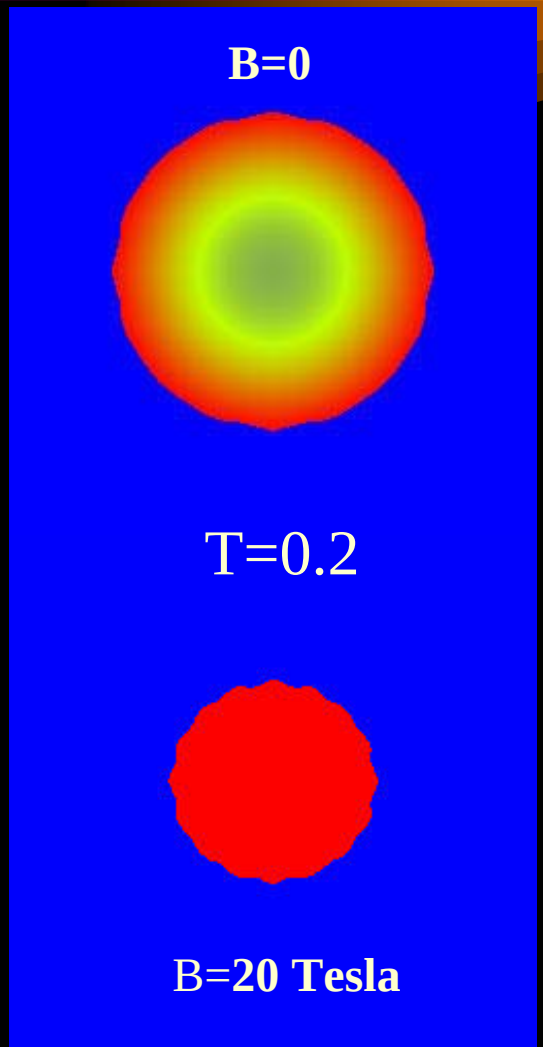
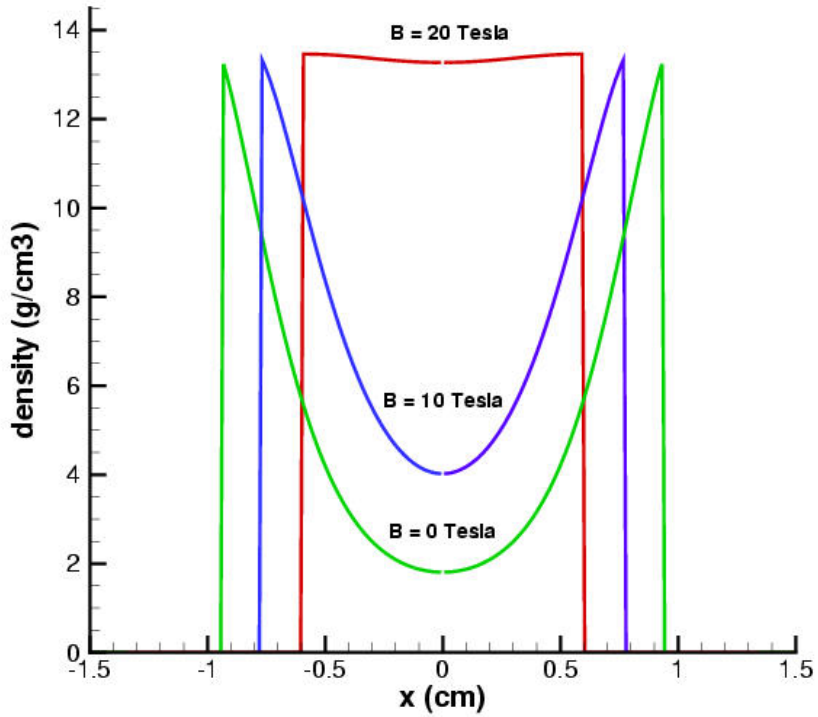
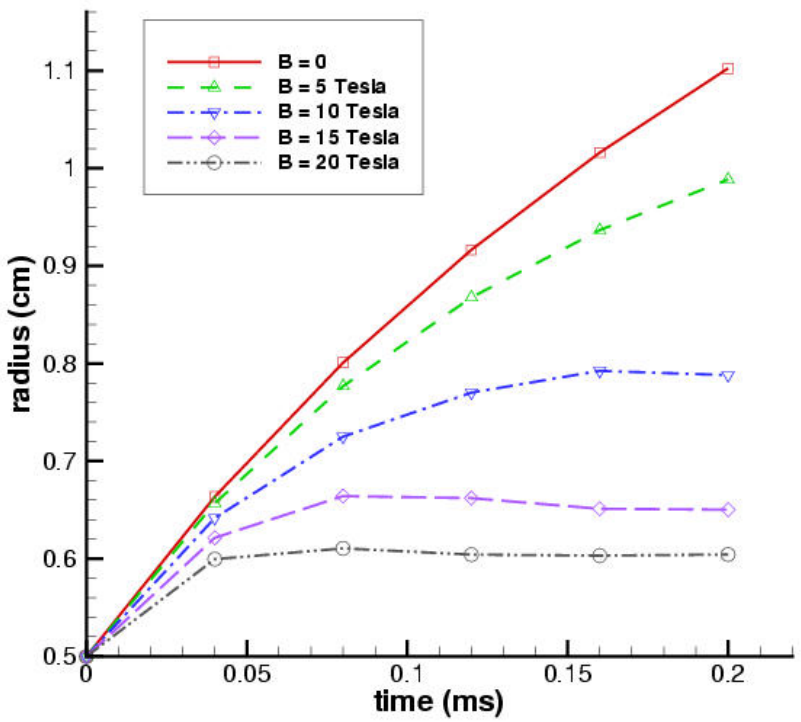


Figure 6. **Simulation of mercury jet expansion**

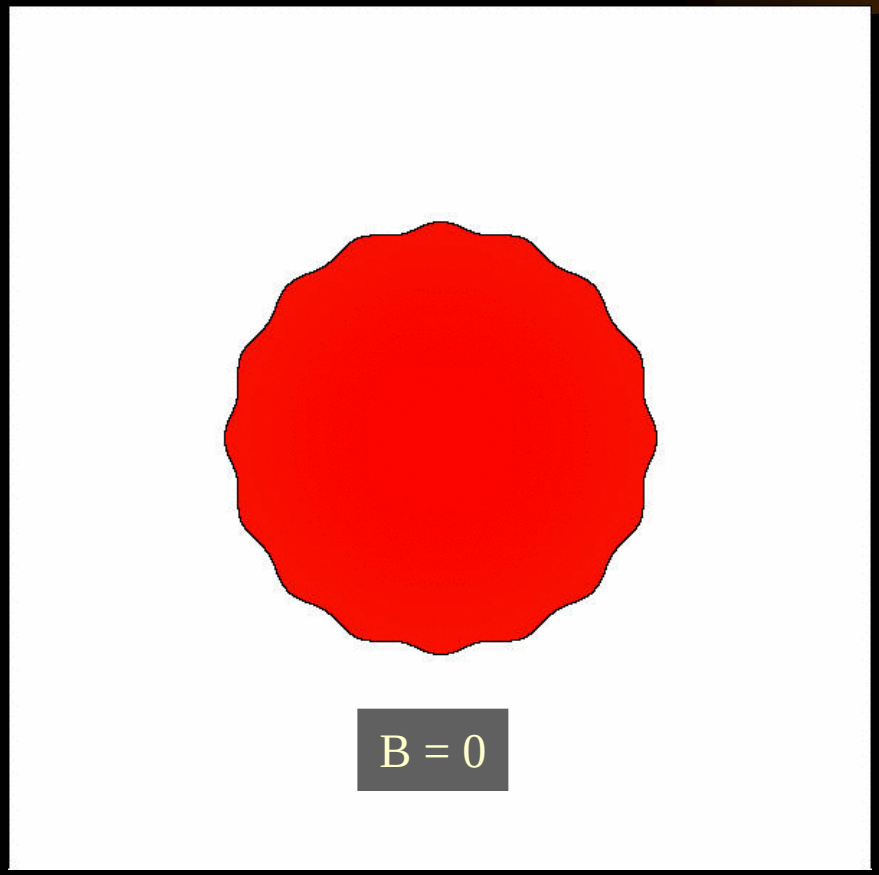
6. Applications





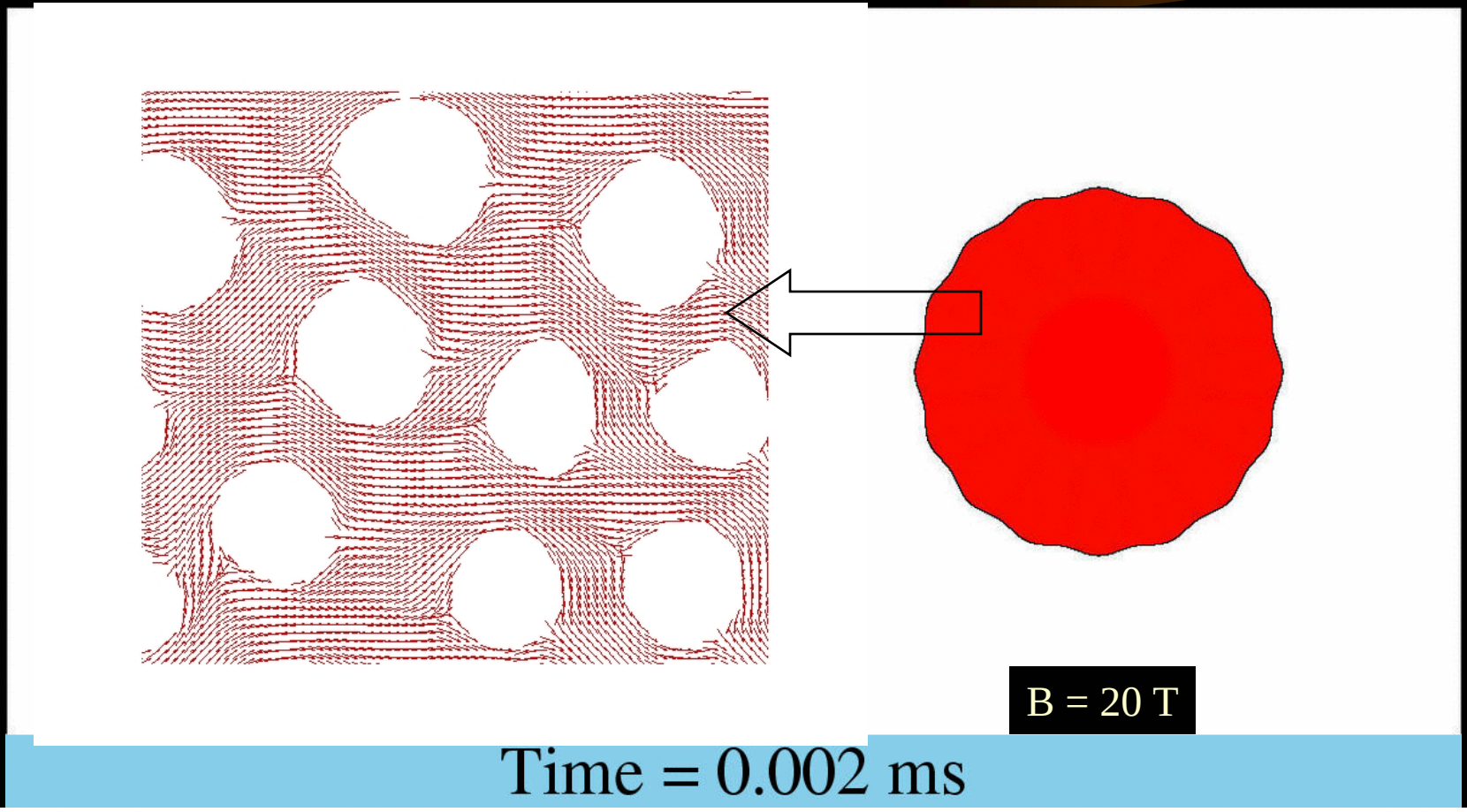
6. Applications

(2) Mercury Jet simulation with dynamic bubble insertion



6. Applications

(2) Mercury Jet simulation



7. Conclusions and current work

(1) Conclusion

* **Embedded Boundary Method for the 2D Neumann boundary elliptic equations are implemented into FronTier code and validated over geometrically complex domain**

* **With application to the MHD system of equations, interface propagation and magnetic flux distribution are as expected**

* **With boundary buffer communication and index mapping ,combined with some high performance linear solvers such as PETSc or HYPRE, above steps can be easily parallelized and the related linear system can also be set and solved with convenience and flexibilities.**

7. Conclusions and current work

(2) 3D implementation

* Same principle as 2D

* Bilinear interpolation of flux