Penguin.jl

A Julia framework for finite-volume PDEs with sharp and moving interfaces

Louis Libat Vincent Le Chenadec Can Selçuk Eric Chénier

Multiscale Modeling and Simulation Laboratory (MSME)
CNRS UMR 8208, Gustave Eiffel University, France

Julia4PDEs 2026

Roadmap

1. Problem: PDEs with jumps at sharp interfaces
2. Method: cut-cell finite-volume discretization on Cartesian grids
3. Result: sharp interface examples
4. Extension: moving interfaces and phase change problems

Why sharp interfaces for multiphase flows?

Key physics is often localized in a **thin interfacial layer** (μm-nm): phase change, surface reactions, species partitioning, ...

Smearing Γ over a few grid cells [1] leads to:

- effective properties yields closure problem
- phase-dependent physics becomes awkward to impose,
- reduced accuracy overall.

Target

✓ Fixed grid ✓ Sharp jumps at Γ
✓ Conservative

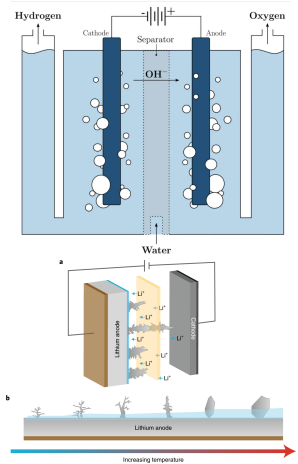


Figure: Electrolysis - Solidification / dendrites

A common sharp-interface template

In each phase $\Omega^{\pm}$:

$$\partial_t \Phi^{\pm} + \nabla \cdot \mathsf{F}^{\pm}(\Phi^{\pm}) = S^{\pm}$$

At the interface Γ:

$$[\![\Phi]\!] = g, \qquad [\![\mathsf{F}\cdot\mathsf{n}]\!] = h$$

Examples in Penguin.jl:

- diffusion - Poisson $\mathsf{F} = -D\nabla\Phi$
- species partitioning $\mathsf{F} = -D\nabla C, \quad [\![C]\!] = (H-1)C^{-}$
- Stokes - Navier-Stokes $\mathsf{F} = -\mu\left(\nabla \mathsf{u} + \nabla \mathsf{u}^{\top}\right) + p\mathsf{I}$
- Stefan - free-boundary $\mathsf{F} = -k\nabla T, \quad h = \rho L\, \mathsf{v}_{\Gamma}\cdot\mathsf{n}$

Numerical question

How do we keep those jumps sharp on a fixed Cartesian grid?

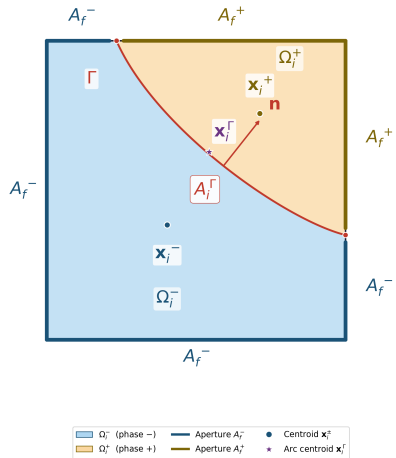
Cut-cell discretization: geometric moments

The interface cuts Cartesian cells. We encode this geometry through **geometric moments**:

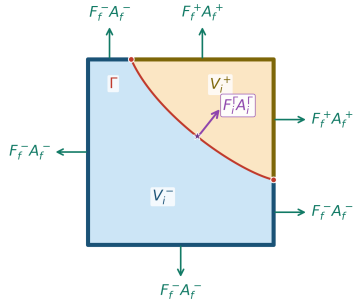
- **phase volumes**: $V_i^{\pm}$
- **open face areas**: $A_f^{\pm}$
- **interface measure**: A_i^{Γ}
- **centroids**: $\mathrm{x}_i^{\pm}$, x_i^{Γ}

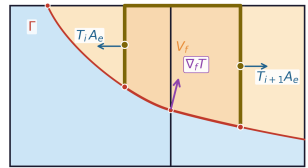
Key point

Only geometric data needed for conservative, compact cut-cell FV operators.



Cut-cell operators + interface DOFs





Finite-volume divergence in phase $\pm$:

$$(\nabla \cdot \mathrm{F})_i \approx \frac{1}{V_i^{\pm}} \left(\sum_f \mathrm{F}_f \cdot \mathrm{n}_f \, A_f^{\pm} + F_i^{\Gamma} A_i^{\Gamma} \right)$$

Gradient on a staggered control volume:

$$\nabla_f T \approx \frac{1}{V_f} \sum_{e \in \partial\Omega_f} T_e \, A_e \, \mathrm{n}_e$$

Introduce interface unknowns $\Phi_i^{\Gamma,\pm}$, one value per side and per cut cell.

One coupled sparse solve

Jump conditions become **additional algebraic equations** for the interface unknowns:

$$\Phi^{\Gamma,+} - \Phi^{\Gamma,-} = g, \qquad F^{\Gamma,+} - F^{\Gamma,-} = A_\Gamma h$$

$$\underbrace{\begin{bmatrix} M_b^- & M_\Gamma^- & & \\ I & & -I & \\ & & M_b^+ & M_\Gamma^+ \\ F_b^- & F_\Gamma^- & F_b^+ & F_\Gamma^+ \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \Phi^- \\ \Phi^{\Gamma,-} \\ \Phi^+ \\ \Phi^{\Gamma,+} \end{bmatrix} = \begin{bmatrix} b^- \\ g \\ b^+ \\ A_\Gamma h \end{bmatrix}$$

Payoff

- No penalty parameter
- No ghost cells / extrapolation
- One sparse coupled solve

In Julia, the assembled sparse systems can be solved with `LinearSolve.jl` (direct, Krylov, preconditioned backends).

Henry Mass Transfer across a bubble (2D)

Bulk diffusion in each phase:

$$\partial_t C^{\pm} = \nabla \cdot (D^{\pm} \nabla C^{\pm}) \qquad \text{in } \Omega^{\pm}$$

Henry's law across Γ:

$$[\![C]\!] = (H-1)C^{-}, \qquad [\![D\nabla C \cdot \mathrm{n}]\!] = 0$$

- Sharp Jump amplitude H is **arbitrary**
- **Flux continuity** enforced at the interface
- Second-order accuracy in the bulk

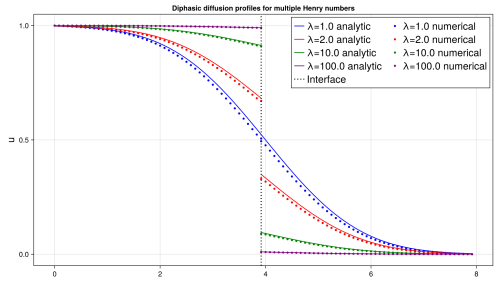
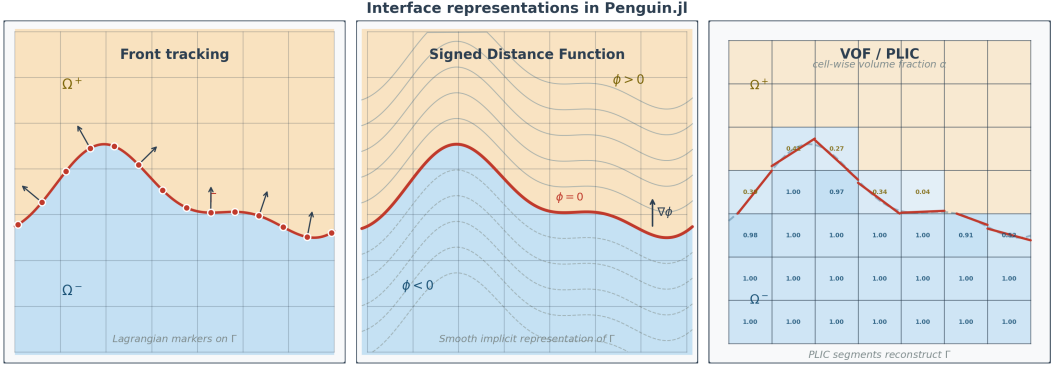


Figure: Radial Concentration profile for various Henry ratios H.

Same cut-cell core, different interface representations

Depending on the physics and topology, Penguin.jl can be coupled to **front tracking**, **signed distance** or **VOF+PLIC** representations.



Moving interfaces: space-time finite volumes

When $\Gamma(t)$ moves, cells can **appear** or **disappear**.

Classical fixes

- Extrapolation: non-conservative
- Cell merging: ad hoc
- Redistribution: only global conservation

Space-time finite volume

Extrude cells over $[t^n, t^{n+1}]$: interface motion becomes a cut in **space-time**.
Fresh / dead cells are simply space-time cut cells.

Local conservation and GCL hold by construction.

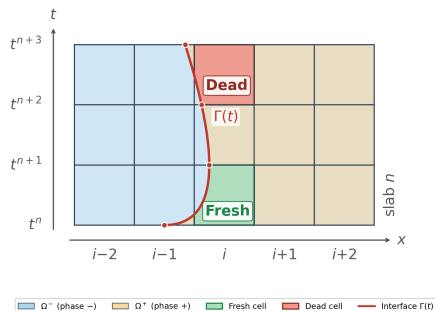


Figure: Space-time control volumes.

Moving interface example: Deforming bubble

Concentration C across a **prescribed moving interface**:

$$\partial_t C^{\pm} = \nabla \cdot (D^{\pm} \nabla C^{\pm}) \qquad \text{in } \Omega^{\pm}(t)$$

- Space-time cut cells handle **fresh/dead cells** automatically
- Jump $[\![C]\!]$ enforced at each time step
- **Local conservation** holds by construction (GCL)
- What if the interface motion depends on the solution?

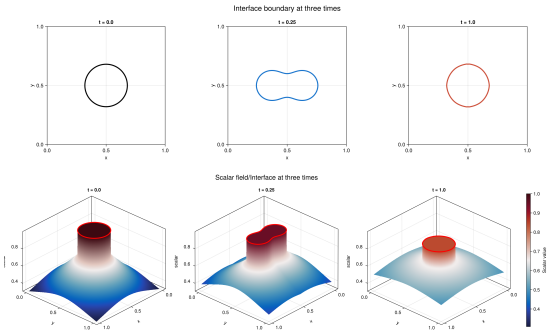
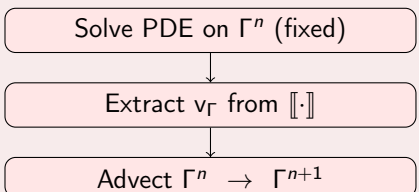


Figure: Concentration field around a moving bubble.

Free boundaries: coupled solve beats splitting

Splitting (classical)



Problem: geometry changes *after* the solve.
Solved domain $\neq$ final domain.

Coupled solve (Penguin.jl)

Find Φ^{n+1} *and* Γ^{n+1} simultaneously:

$$\begin{cases} \mathcal{L}(\Phi^{n+1}, \Gamma^{n+1}) = 0 & \text{(bulk PDE)} \\ \mathcal{R}(\Phi^{n+1}, \Gamma^{n+1}) = 0 & \text{(jump cond.)} \end{cases}$$

Inner loop per time step:

1. Fix $\Gamma^{(k)}$, solve bulk PDE $\rightarrow \Phi^{(k)}$
2. Compute residual $\mathcal{R}(\Phi^{(k)}, \Gamma^{(k)})$
3. Update $\Gamma^{(k+1)}$ to reduce residual
4. Repeat until $|\mathcal{R}| \approx \varepsilon_{\text{mach}}$

Coupled Stefan iterations converge rapidly

- **1st iteration**: classical advection algorithm
- **Next iterations**: inner corrections
- **3-10 iterations** usually enough
- Final residual at **machine precision**

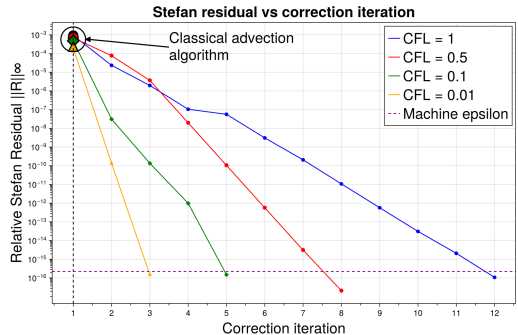


Figure: Fast residual decay of the Stefan correction loop.

Phase change (Stefan) and dendrites

Energy jump drives interface velocity:

$$[\![k\nabla T\cdot \mathrm{n}]\!] = \rho L\, \mathrm{v}_\Gamma \cdot \mathrm{n}$$

Observations:

- Planar fronts converge at **second order** (appendix)
- Unstable modes match Mullins-Sekerka [4]
- Full **dendrite patterns** without ad hoc fixes

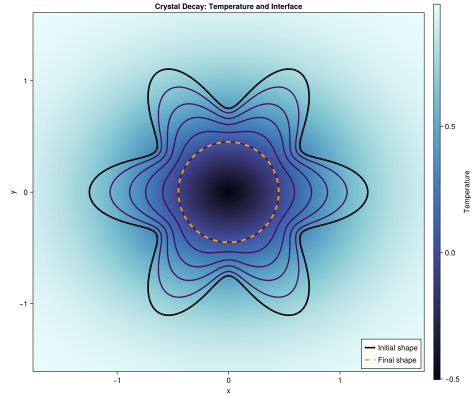
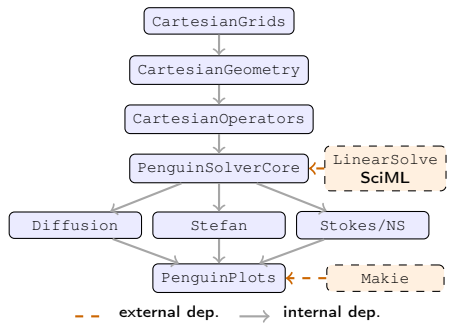


Figure: Dendritic growth: temperature field and tracked interface.

Why Julia for Penguin.jl?

- **Multiple dispatch**: the same `grad/div` work with front tracking, SDF, or VOF
- **Composable packages**: geometry, operators, and physics solvers share one core
- **SciML interoperability**: `LinearSolve.jl` gives direct and iterative backends
- **One-language workflow**: geometry, discretization, and solvers all in Julia



One Julia workflow: coupled Stefan / free boundary

```
using CartesianGeometry, PenguinBCs, PenguinStefan

# geometry from a front-tracking interface
grid  = (0.0:0.1:1.0, 0.0:0.1:1.0)
ft_circle = sample_circle(...)

# outer boundary conditions
bc = BorderConditions(; left=Dirichlet(...), right=Dirichlet(...))

# interface representation + Stefan problem
rep  = FrontTrackingRep(grid, ft_circle; coupling = :ft_lm, max_iter = ...)
# rep = LevelSetRep(...)
prob = StefanDiphProblem(..., rep; bc_border = bc)

# build and march the coupled solver
solver = build_solver(prob; t0 = 0.0, uω0 = uω0, uγ0 = uγ0)
solve!(solver, (0.0, t_end); dt = Δt)
```

Takeaways & next steps

What to remember

- **Two-fluid** + cut cells: sharp jumps on a Cartesian grid
- **Operators** built from geometric moments
- **Interface DOFs**: enforce jumps directly
- **Space-time FV**: moving interfaces with local conservation
- **Strong coupling**: free boundaries without splitting error

Next steps

- 3D space-time (4D cut cells)
- NS + species + Stefan coupling
- **Parallelisation**: via `MPI.jl`: non-linear iterations and implicit solves are the bottleneck
- **GPU port** via `KernelAbstractions.jl`: backend-agnostic kernels for FV operators

Thank you - Questions?

Bibliography

I. Kataoka.
Local instant formulation of two-phase flow.
IJMF, 1986.

L. Libat, V. Le Chenadec, C. Selçuk, and E. Chénier.
A Space-Time Cartesian Cut-Cell Method for Two-Phase Diffusion Problems using a Two-Fluid Approach.
HAL preprint, 2026.

R. A. Remmerswaal, J. M. H. Kuipers, and A. J. van der Ham.
Towards a sharp, structure preserving two-velocity model for two-phase flow: transport of mass and momentum.
2022.

W. W. Mullins and R. F. Sekerka.
Stability of a planar interface during solidification of a dilute binary alloy.
Journal of Applied Physics, 35(2):444-451, 1964.

Two-phase Poisson equation: Julia 3-circle logo

$$-\nabla\cdot\left(\beta^{\pm}\nabla\Phi^{\pm}\right) = f^{\pm} \qquad \text{in } \Omega^{\pm}$$

Sharp interface conditions on Γ:

$$[\![\Phi]\!] = g(\mathrm{x}), \qquad [\![\beta\nabla\Phi\cdot\mathrm{n}]\!] = h(\mathrm{x})$$

- Multiple disconnected interfaces handled in one solve
- Sharp jump captured without smearing
- Clean error localization near Γ

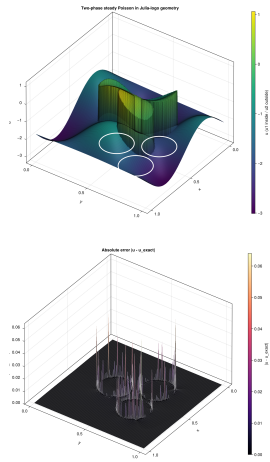


Figure: Top: solution. Bottom: error.

Appendix - Henry jump: gain over one-fluid

n	Cut-cell error	One-fluid error	Improvement
1	3.06×10^{-1}	-	$\times 1.0$
2	6.69×10^{-3}	1.59×10^{-2}	$\times 2.4$
4	1.36×10^{-3}	9.85×10^{-3}	$\times 7.2$
8	3.24×10^{-4}	5.45×10^{-3}	$\times 16.8$
16	5.25×10^{-5}	2.75×10^{-3}	$\times 52.5$
32	6.46×10^{-6}	1.42×10^{-3}	$\times 220$
64	1.26×10^{-6}	7.03×10^{-4}	$\times 560$
128	1.94×10^{-7}	3.32×10^{-4}	$\times 1707$
Observed order	2.56	1.1	-

- Sharp cut-cell is already solvable at $n = 1$
- Only **one cell inside the bubble**
- Gain increases strongly with refinement

Appendix - Static sharp-interface diffusion

```
using CartesianGeometry, CartesianOperators, PenguinBCs, PenguinDiffusion

# geometry in each phase
body(x, y) = ...
moms1 = geometric_moments( body, grid, Float64, nan; method = :vofijul)
moms2 = geometric_moments(-body, grid, Float64, nan; method = :vofijul)

# outer BCs + interface jump conditions
bc = BorderConditions(; left=Dirichlet(...), right=Dirichlet(...))
ic = InterfaceConditions(
    scalar = RobinJump(α, β, g),
    flux   = FluxJump(1.0, 1.0, 0.0))

# cut-cell operators and diphasic model
cap1, cap2 = assembled_capacity(moms1), ...
ops1, ops2 = DiffusionOps(cap1; periodic = periodic_flags(bc, 2)), ...

model = DiffusionModelDiph(cap1, ops1, D1, s1, cap2, ops2, D2, s2;
                           bc_border = bc, ic = ic)
sol = solve_steady!(model)
```

Appendix - Stefan front accuracy

h ($\Delta t = 0.5h$)	$\|T - T_{\text{exact}}\|_2$	$\|s - s_{\text{exact}}\|$	avg iters	last iters
0.2	1.82×10^{-2}	5.69×10^{-3}	24.25	18
0.1	4.84×10^{-3}	1.65×10^{-3}	24.63	14
0.05	1.44×10^{-3}	5.40×10^{-4}	18.27	12
0.025	3.58×10^{-4}	1.47×10^{-4}	14.47	11
0.0125	8.76×10^{-5}	3.42×10^{-5}	11.92	10
0.00625	2.56×10^{-5}	8.70×10^{-6}	10.06	9
0.003125	1.12×10^{-5}	1.92×10^{-6}	8.67	6
Observed order	1.83	1.92	-	-

- Nearly second-order convergence for both temperature and interface position
- Inner correction count decreases with mesh refinement
- Final correction step typically needs only 6-18 iterations

Appendix - FT least-squares correction

Predict a front $\Gamma^{n+1}_{\text{pred}}$, then correct it by normal displacements

$$x_v(d) = x_{v,\text{pred}} + d_v \, n_v^{\text{pred}}.$$

For each candidate front:

$$\Gamma^{n+1}(d) \;\longrightarrow\; \text{space-time diffusion solve} \;\longrightarrow\; r(d) \;\longrightarrow\; \min_d \; \mathcal{J}(d)$$

with

$$r_i(d) = \rho L\big(V_i^{n+1}(d) - V_i^n\big) + \Phi_i(d), \qquad \mathcal{J}(d) = \tfrac{1}{2}\|W\, r(d)\|_2^2 + \tfrac{1}{2} d^\top R d.$$

- near-2nd-order recovered for temperature and interface position
- average inner iterations decrease with refinement
- **1 iteration** typically reduces the residual by a factor 5-20

Appendix - Cut-cell incompressible flow

In each phase:

$$\rho^{\pm}\left(\partial_t \mathsf{u}^{\pm} + \mathsf{u}^{\pm}\cdot\nabla\mathsf{u}^{\pm}\right) = -\nabla p^{\pm} + \nabla\cdot\left(2\mu^{\pm}\mathsf{D}(\mathsf{u}^{\pm})\right) + \mathsf{f}^{\pm}$$

$$\nabla\cdot\mathsf{u}^{\pm} = 0$$

Across the sharp interface Γ:

$$[\![\mathsf{u}]\!] = 0, \qquad [\![\boldsymbol{\sigma}\mathsf{n}]\!] = \mathsf{t}_\Gamma$$

$$\boldsymbol{\sigma} = -p\,\mathsf{I} + 2\mu\mathsf{D}(\mathsf{u}), \qquad \mathsf{D}(\mathsf{u}) = \tfrac{1}{2}\left(\nabla\mathsf{u} + \nabla\mathsf{u}^{\top}\right)$$

- Same cut-cell / staggered-grid philosophy
- Sharp velocity and traction conditions at Γ

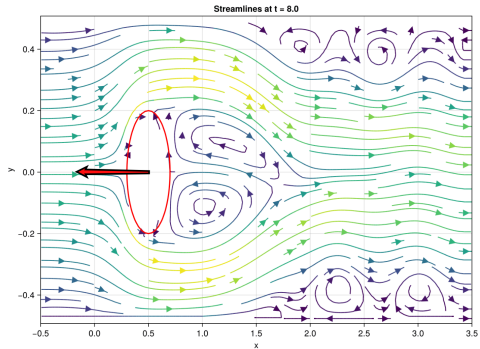


Figure: Representative incompressible-flow example on a Cartesian cut-cell grid.