# Penalty / Brinkman Penalization for Immersed Methods in Fluid–Structure Interaction

Annotated bibliography

January 5, 2026

## Overview

The penalty (or Brinkman) approach for immersed methods models a solid (or obstacle) inside a fluid by adding a penalization term in the momentum equations that enforces the solid velocity inside the solid region. A common form of the penalized Navier–Stokes equations is

$$\rho(\partial_t u + u \cdot \nabla u) - \mu \Delta u + \nabla p + \frac{\chi}{\eta}(u - u_s) = f,$$

where $\chi$ is the characteristic function of the solid, $\eta \ll 1$ is the penalty parameter, and $u_s$ is the (prescribed or rigid-body) solid velocity. As $\eta \to 0$ the velocity inside the solid region is driven toward $u_s$. This approach is simple to implement on Cartesian grids and is widely used in immersed and fictitious-domain methods.

## Key references (papers and books)

- C. S. Peskin, "The immersed boundary method," *Acta Numerica*, 2002.
  - Classic, accessible review of the immersed boundary (IB) philosophy and applications to fluid–structure problems (especially biological flows). Good starting point for understanding IB ideas and numerics.
- S. Mittal and G. Iaccarino, "Immersed boundary methods," *Annual Review of Fluid Mechanics*, 2005.

  - A modern survey of immersed boundary/type methods used in CFD, comparing variants (direct forcing, ghost-cell, penalization, etc.). Useful for practitioners choosing an approach.

- D. Angot, F. Bruneau, and P. Fabrie, "A penalization method to take into account obstacles in incompressible viscous flows," (see literature on Brinkman penalization, late 1990s).
  - Introduces and analyzes the idea of penalizing the momentum equations (Brinkman-type term) to represent solid obstacles. This is central to the penalty/Brinkman family of immersed methods.

- T. Richter, "Fluid-Structure Interactions: Models, Analysis and Finite Elements," Springer, 2017.
  - A modern textbook on FSI; contains chapters on different numerical approaches (monolithic, partitioned, immersed/fictitious-domain) and useful pointers to analysis and implementation details.

- H. C. Brinkman, "The viscosity of a fluid in a porous medium," (1947).
  - The penalization term commonly used is often called "Brinkman" penalization because it mimics the Brinkman equation for flow in porous media; understanding this model clarifies the mathematical foundation of penalization.

## Selected applied / numerical papers

- Work applying Brinkman penalization on Cartesian/Eulerian grids for complex geometries and moving bodies (see literature on "Brinkman penalization" and "penalty method" for FSI).
- Variational formulations and immersed finite element/fictitious domain variants: see survey and research articles on "immersed finite element method" (IFEM) and "fictitious domain / distributed Lagrange multiplier" approaches for comparisons with penalization.

## Videos and lectures

- Look for recorded lectures by Charles Peskin (Courant Institute) on the immersed boundary method (many are available online).

- Conference and workshop talks on immersed and fictitious-domain methods: search for "Brinkman penalization", "immersed boundary lecture", and "fictitious domain" on YouTube and conference pages (ICCFD, BIRS workshops, and specialized FSI workshops).

## Notes and practical tips

- Choice of penalty parameter $\eta$: small $\eta$ enforces the constraint more strongly but increases stiffness of the time-stepping; implicit treatment or operator splitting is often used.
- Accuracy near the fluid–solid interface: penalization methods are simple but may smear the interface over a few grid cells. Higher-order treatments and sharp-interface immersed variants (ghost-fluid, cut-cell, or variational immersed methods) improve accuracy there.
- When to use penalization: quick prototyping, complex/moving geometries on Cartesian grids, or coupling to porous-media models. For high-accuracy boundary stresses, consider variational immersed or fitted-mesh approaches.

## Suggested reading order

1. C. S. Peskin (2002) — broad conceptual overview and historical context.
2. S. Mittal and G. Iaccarino (2005) — survey of numerical variants.
3. D. Angot, F. Bruneau, P. Fabrie (late 1990s) — foundational penalization analysis (Brinkman).
4. T. Richter (2017) — textbook treatment of FSI and computational methods.

## Practical definition for an elastic body (linear elasticity)

Consider a computational domain $\Omega$ containing a deformable elastic body occupying $\Omega_s(t) \subset \Omega$ and fluid in the remainder. Using a Brinkman/penalty approach we solve fluid equations on the whole domain and enforce the solid kinematics through a penalization term. A compact monolithic form (Eulerian description) is:

$$\rho_f(\partial_t u + u \cdot \nabla u) - \nabla \cdot \sigma_f + \frac{\chi}{\eta}(u - v_s) = f_f \quad \text{in } \Omega,$$

$$\nabla \cdot u = 0 \quad \text{in } \Omega,$$

$$\rho_s\, \partial_{tt} d - \nabla \cdot \sigma_s - \chi \frac{1}{\eta}(u - v_s) = f_s \quad \text{in } \Omega_s(t),$$

where

- $u$ is the Eulerian fluid velocity,
- $d$ is the solid displacement and $v_s = \partial_t d$ the solid velocity,
- $\chi(x,t)$ is a (possibly smoothed) characteristic function of the instantaneous solid region, $\chi = 1$ in the solid and 0 in the fluid,
- $\eta \ll 1$ is the penalty parameter,
- $\sigma_f = -pI + 2\mu_f \varepsilon(u)$ is the fluid Cauchy stress and
- for small strains $\sigma_s \approx \lambda_s(\nabla \cdot d)I + 2\mu_s \varepsilon(d)$ (linear elasticity).

Remarks and implementation recipe:

- Coupling mechanism: the term $\chi\eta^{-1}(u - v_s)$ enforces $u \approx v_s$ inside the solid; the same term appears with opposite sign in the solid momentum as a reaction force. This yields a monolithic, momentum-conserving coupling.
- Tracking $\chi$: for a deforming body you must update the indicator $\chi(x,t)$ each time step. Options: (i) advect a level-set or a marker field and rebuild a smoothed $\chi$, or (ii) use an immersed (Lagrangian) mesh and spread/interpolate between grids with operators $\mathcal{S}, \mathcal{J}$; then use $\mathcal{S}(u - v_s)$ in the solid and $\mathcal{J}$(reaction) in the fluid.
- Time discretization: treat the stiff penalization implicitly to avoid severe CFL constraints (e.g. backward Euler or BDF for the penalized momentum term). For the solid either use Newmark/implicit schemes or the same implicit time integrator as the fluid for consistency.
- Spatial discretization: popular choices are finite-volume / finite-difference for fluid on Cartesian grids and finite elements for the solid with interpolation/spreading; a fully Eulerian finite-element discretization (both fields on the same grid) is also possible.

- Penalty parameter selection: choose $\eta$ small enough to enforce kinematics with acceptable error but not so small that linear solves become ill-conditioned. In practice $\eta \sim C\Delta t$ (C 0.1–1) or tuned experimentally; see Angot et al. for convergence theory.

- Interface accuracy: Brinkman penalization can smear the interface over a few grid cells. If accurate boundary tractions are required, combine penalization with a sharp-interface correction (ghost-fluid, cut-cell) or use variational immersed methods.

Algorithm sketch (per time step):

1. Given $u^n, d^n, v_s^n$, update/advect level-set or Lagrangian markers and build $\chi^{n+1}$.

2. Predict advection/explicit terms for the fluid; assemble momentum with implicit penalization term $\chi^{n+1}/\eta$.

3. Solve the coupled fluid velocity/pressure and solid elastic update (monolithic solve) or iterate in a partitioned loop where the penalization is treated consistently between substeps.

4. Update solid displacement $d^{n+1}$ from $v_s^{n+1}$ and advance to next step.

References for practical implementations: see Angot et al. (Brinkman penalization analysis), Mittal and Iaccarino (survey of immersed methods), and literature on immersed finite elements and fictitious-domain methods for concrete spreading/interpolation operators.

## Partitioned (staggered) time-stepping

A practical partitioned (staggered) algorithm solves the fluid and solid subproblems separately while exchanging interface/data via the penalization (or reaction) term. Below is a stable, commonly-used pattern with optional acceleration.

Notation: time step $\Delta t$, known states at time $t^n$: $u^n, d^n, v_s^n$. We denote sub-iteration index by $k$.

Algorithm (sub-iterative partitioned):

1. Initialize $d^{n+1,0} = d^n$, $v_s^{n+1,0} = v_s^n$, build $\chi^{n+1,0}$ (advect level-set or map Lagrangian markers).

2. For $k = 0, 1, \ldots$ until convergence:

   (a) Fluid step: given $v_s^{n+1,k}$ and $\chi^{n+1,k}$ solve for $(u^{n+1,k+1}, p^{n+1,k+1})$ on the whole domain by discretizing

   $$\rho_f \frac{u^{n+1} - u^n}{\Delta t} + (\text{convective, viscous, pressure}) + \frac{\chi^{n+1,k}}{\eta}(u^{n+1} - v_s^{n+1,k}) = \text{RHS},$$

   treating the penalization term implicitly in $u^{n+1}$ (i.e. include $\chi/\eta$ on the LHS) to avoid stiff time-step restrictions.

   (b) Reaction computation: form the reaction (force density) applied to the solid from the penalization

   $$f_{\text{react}}^{n+1,k+1} = -\frac{\chi^{n+1,k}}{\eta}\left(u^{n+1,k+1} - v_s^{n+1,k}\right)$$

   (or compute the integrated traction if using stress-based coupling).

   (c) Solid step: solve the (linear) elastodynamics problem on $\Omega_s$ with external force $f_{\text{react}}^{n+1,k+1}$ to obtain $d^{n+1,k+1}$ and $v_s^{n+1,k+1}$ (e.g. implicit Newmark or backward-Euler). Update $\chi^{n+1,k+1}$ if it depends on the updated geometry.

   (d) Convergence check: stop if residuals are below tolerance, e.g.

   $$\|u^{n+1,k+1} - u^{n+1,k}\| + \|v_s^{n+1,k+1} - v_s^{n+1,k}\| < \text{tol}.$$

   Optionally apply relaxation: $v_s^{n+1,k+1} \leftarrow \omega v_s^{n+1,k+1} + (1-\omega) v_s^{n+1,k}$ with Aitken acceleration to speed convergence.

3. Accept $(u^{n+1}, p^{n+1}, d^{n+1}, v_s^{n+1}) = (u^{n+1,k+1}, p^{n+1,k+1}, d^{n+1,k+1}, v_s^{n+1,k+1})$ and advance.

Comments, stability and enhancements:

- Added-mass effect: when the solid density is comparable to or smaller than the displaced fluid, loosely-coupled (no sub-iterations) Dirichlet–Neumann partitioning may be unstable. Sub-iterations (strong coupling) or interface-regularization are recommended.
- Robin (or impedance) coupling: improve stability by replacing strict Dirichlet/Neumann exchange with Robin-type conditions (weighted combination of velocity and traction). This can be implemented by adding an interface impedance term or by modifying the penalization to include a tunable impedance parameter; it often reduces the number of sub-iterations required.
- Quasi-Newton interface solvers (IQN-ILS) and Aitken acceleration are effective at accelerating convergence of the partitioned iterations without fully monolithic solves.
- Implementation notes: use the same time-integration family (implicit) for both subproblems when possible; treat the penalization implicitly in the fluid solver; assemble reaction forces consistently (volume-distributed penalty vs. surface traction) so momentum is conserved.
- Linear solvers/preconditioning: the fluid solve with implicit penalization adds a large diagonal-like term in solid cells. Preconditioners should account for this (e.g., block preconditioners or algebraic multigrid tuned to variable coefficients).

This partitioned approach balances modularity (reuse of existing fluid/solid solvers) with robustness (via sub-iterations, relaxation, or Robin coupling). For challenging added-mass regimes or very tight accuracy on interface stresses, consider monolithic solution strategies.

## Discretization: FV fluid + FEM structure and penalty-term handling

When using lowest-order Finite Volume (FV) for the fluid and Finite Element (FEM) for the structure, the two grids typically differ (FV Cartesian, FEM conformal to the solid). The penalty terms $\chi/\eta(u - v_s)$ must be discretized and exchanged consistently.

## Fluid discretization (lowest-order FV)

On a Cartesian FV grid, the fluid momentum equation is integrated over cell $C_i$ (volume $|C_i|$):

$$\rho_f \frac{u_i^{n+1} - u_i^n}{\Delta t} + (\text{convection, viscosity, pressure}) + \frac{1}{|C_i|} \int_{C_i} \frac{\chi}{\eta}(u - v_s)\, dx = \text{RHS}_i.$$

The integral of the penalty term over the cell is approximated as

$$\frac{1}{|C_i|} \int_{C_i} \frac{\chi}{\eta}(u - v_s)\, dx \approx \frac{\chi_i}{\eta}(u_i - \bar{v}_s^{(i)}),$$

where $\chi_i \in [0, 1]$ is the volume fraction (or indicator) of the solid in cell $C_i$, and $\bar{v}_s^{(i)}$ is the solid velocity *interpolated* to the cell center from the FEM mesh.

## Structure discretization (FEM)

Let $\mathcal{N}_s$ denote the FEM nodes of the solid and $\phi_j$ the associated basis functions. The solid momentum equation is integrated against test functions; on each solid element $E_s$ with nodes $\{j_1, \ldots, j_n\}$, we assemble

$$\int_{E_s} \rho_s \partial_{tt} d \cdot \phi_j\, dx + \int_{E_s} \sigma_s(d) : \varepsilon(\phi_j)\, dx + \int_{E_s} f_{\text{react}} \cdot \phi_j\, dx = 0,$$

where the reaction force density is

$$f_{\text{react}} = -\frac{\chi}{\eta}(u - v_s).$$

To couple to the FV fluid, the reaction must be computed and mapped from the fluid (FV) to the structure (FEM).

## Spreading and interpolation operators

Define two key operators:

- **Restriction/Interpolation operator $\mathcal{J}$:** maps from FV cell values to FEM nodes. For a quantity $q_i$ (cell-centered in FV), we evaluate $q$ at an FEM node $\mathbf{x}_j$ by

$$(\mathcal{J} q)_j = \sum_i W_{ij}\, q_i,$$

where $W_{ij}$ are interpolation weights (e.g., inverse-distance, area-weighted, or linear basis evaluation on the FV mesh). In lowest-order FV this is straightforward: use distance-weighted or volume-averaged interpolation.

- **Spreading operator $\mathcal{S}$:** maps from FEM quantities (nodal values or integrated over elements) back to FV cells. For an FEM vector $\mathbf{f}_j$ (e.g., reaction force at node $j$), we distribute it to overlapping FV cells via
$$F_i = \frac{1}{|C_i|} \sum_j \left( \int_{C_i \cap \mathrm{supp}(\phi_j)} \mathbf{f}_j \phi_j(\mathbf{x})\, d\mathbf{x} \right),$$
or more practically, we compute for each FEM element the reaction-force contribution and accumulate it into the FV cells it overlaps, weighted by the overlap area/volume.

## Consistent penalty-term exchange

At each sub-iteration $k$ within a time step:

1. Given solid velocity $v_s^{n+1,k}$ (from previous iteration), interpolate to FV cells:
$$\bar{v}_s^{(i),n+1,k} = \mathcal{J}(v_s^{n+1,k}),$$
where $\mathcal{J}$ evaluates the FEM shape functions at cell centers (or uses interpolation weights).

2. In the FV solver, assemble the fluid momentum with the penalty term:
$$\rho_f \frac{u_i^{n+1,k+1} - u_i^n}{\Delta t} + \cdots + \frac{\chi_i}{\eta}\big(u_i^{n+1,k+1} - \bar{v}_s^{(i),n+1,k}\big) = \mathrm{RHS}_i.$$
This typically appears as a diagonal-dominant term (since $\chi_i/\eta$ is large in solid cells), so it is included implicitly in the LHS stiffness matrix.

3. Compute cell-centered reaction force:
$$F_i^{n+1,k+1} = -\frac{\chi_i}{\eta}\big(u_i^{n+1,k+1} - \bar{v}_s^{(i),n+1,k}\big).$$

4. Spread (map) the reaction to FEM nodes and assemble into the structure system. For each FEM node $j$, assemble the reaction contribution
$$\mathbf{R}_j = \mathcal{S}(F_i) = \sum_i \left( \int_{\mathrm{supp}(\phi_j) \cap C_i} F_i \phi_j(\mathbf{x})\, d\mathbf{x} \right).$$

This is added to the RHS of the structure equation.

5. Solve the FEM elastic system with the distributed reaction load $\mathbf{R}$ to obtain $d^{n+1,k+1}$ and $v_s^{n+1,k+1}$.

## Practical considerations

- **Volume fraction $\chi_i$:** compute once per time step (or per sub-iteration if geometry changes). For each cell $C_i$, $\chi_i = \frac{\text{volume of cell in solid}}{\text{volume of cell}}$. Use a level-set or robust polygon/polyhedron intersection library.
- **Basis-function support:** when spreading $\mathcal{S}$, integrate $F_i \phi_j$ over overlapping regions. For lowest-order FEM (linear basis), $\phi_j$ is piecewise-linear on the element. Compute the overlap region (cell $C_i \cap$ element $E_s$) and integrate numerically or analytically.
- **Conservation:** ensure that total momentum transferred is conserved. Check that $\sum_i F_i \, |C_i| + \sum_j \mathbf{R}_j = 0$ (or their sum equals prescribed external forces). If $\mathcal{S}$ and $\mathcal{J}$ are adjoint-consistent (i.e., $\mathcal{S}^T = \mathcal{J}$ or similar), momentum is guaranteed to be conserved.
- **Accuracy near the interface:** lowest-order FV and FEM will both smear the interface. Use a small $\eta$ to enforce the constraint tightly, but not so small that linear systems become ill-conditioned. Typically $\eta \sim 0.01$–$0.1$ (or tune relative to $\Delta t$).
- **Quadrature:** when assembling the structure system with the spreading operator, use sufficient quadrature (at least 2-point Gauss per element) to capture the reaction force accurately. If reaction is cell-averaged, use a consistent lumped or element-wise integration.
- **Code organization:** pre-compute and store the nonzero pattern of $\mathcal{J}$ and $\mathcal{S}$ once (at initialization or first time step). Store overlap regions and integration weights. Recompute $\chi_i$ when the solid moves significantly.

(I can now recompile the PDF, add a short MATLAB/Python partitioned-demo, or produce a BibTeX file with full citations.)

## 1 Computation of transfer operators: $C^{-1}B^\top$ and its transpose

When implementing immersed methods for fluid–structure interaction with non-conforming meshes (e.g., finite-volume fluid on a Cartesian grid and finite-element structure on a fitted mesh), we must transfer quantities and forces between the two discretizations. This section explains the construction and use of the transfer operator $C^{-1}B^\top$ and its transpose for bidirectional mapping.

### 1.1 Motivation and setup

In the Lagrange multiplier approach for FSI (or penalty methods), the coupling between fluid and solid is mediated through matrices that relate the two different function spaces. The key matrices are:

$$B_{i,j} := \int_{I(t)} \boldsymbol{\phi}_i \cdot \boldsymbol{\psi}_j \,\mathrm{d}\gamma,$$
$$C_{k,i} := \int_{I(t)} \boldsymbol{\varphi}_k \cdot \boldsymbol{\phi}_i \,\mathrm{d}\gamma,$$

where:

- $\boldsymbol{\phi}_i$ are basis functions for the Lagrange multiplier space (defined on the interaction domain $I(t)$),
- $\boldsymbol{\psi}_j$ are fluid basis functions (e.g., indicator functions for finite-volume cells on the interface),
- $\boldsymbol{\varphi}_k$ are solid basis functions (e.g., finite-element basis functions),
- $I(t)$ is the interaction domain: either $\partial\Omega_s(t)$ (boundary) for immersed-boundary methods or $\Omega_s(t)$ (volume) for immersed-domain methods.

### 1.2 Discretization and mass lumping

The matrix $C$ represents the $L^2(I(t))$ mass matrix of the multiplier space. To avoid solving a global interface system and keep the method computationally efficient, we replace $C$ with its diagonal lumped counterpart:

$$C^{\mathrm{lump}}_{ii} := \sum_k C_{ik} = \int_{I(t)} \boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_i \,\mathrm{d}\gamma, \qquad C^{\mathrm{lump}}_{ij} = 0 \quad \text{for } i \neq j.$$

The inverse is then simply:

$$\big((C^{\mathrm{lump}})^{-1}\big)_{ii} = \frac{1}{C_{ii}^{\mathrm{lump}}}.$$

For the fluid basis, we assume $\boldsymbol{\psi}_j$ are cellwise indicator functions on finite-volume cells $F_j \subset I(t)$. Under this assumption:

$$B_{i,j} = \int_{F_j} \boldsymbol{\phi}_i \,\mathrm{d}\gamma.$$

### 1.3 Formula for the transfer operator

The discrete transfer operator $C^{-1}B^{\top}$ maps fluid quantities (e.g., velocities) defined on the FV mesh to the solid mesh (or multiplier space). Its action is approximated by:

$$\big(C^{-1}B^{\top}\big)_{i,j} \approx \frac{1}{\displaystyle\int_{I(t)} \boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_i \,\mathrm{d}\gamma} \int_{F_j} \boldsymbol{\phi}_i \,\mathrm{d}\gamma.$$

This corresponds to an $L^2$ quasi-projection based on mass lumping, which yields:

- A **local** operator: each solid/multiplier DOF depends only on nearby fluid cells.
- A **conservative** operator: the total mass/momentum transferred is conserved.
- A **stable** operator: no spurious oscillations or energy growth introduced by interpolation.

### 1.4 Bidirectional transfer

At each time step or sub-iteration:

1. **Forward transfer (fluid $\rightarrow$ solid):** given fluid quantities (e.g., velocity $\boldsymbol{u}_f$) at FV cell centers, compute solid quantities:

$$\mathcal{T} := C^{-1}B^{\top}, \qquad \boldsymbol{u}_s = \mathcal{T}\,\boldsymbol{u}_f.$$

   This operator $\mathcal{T}$ performs a weighted interpolation, sampling the fluid velocity at points where the solid basis functions are supported and lumping according to their normalized $L^2$ mass.

2. **Backward transfer (solid $\to$ fluid):** given solid forces (e.g., reaction forces or stresses) $\boldsymbol{f}_s$ at solid nodes/elements, compute the corresponding force density on the fluid mesh:

$$\mathcal{T}^\top := BC^{-\top}, \qquad \boldsymbol{f}_f = \mathcal{T}^\top \boldsymbol{f}_s.$$

   This is the adjoint (transpose) operator, ensuring momentum conservation: the force transferred to the fluid exactly balances (with opposite sign) the force acting on the solid.

## 1.5 Practical implementation

When assembling these operators in code (e.g., in a partitioned FSI solver):

1. **Pre-compute masses:** at initialization, compute and store $C_{ii}^{\text{lump}}$ and its inverse for all multiplier/solid DOFs.
2. **Identify overlaps:** for each fluid cell and each solid basis function, determine which cells overlap with the support of that basis function, and compute the integral $\int_{F_j} \boldsymbol{\phi}_i \, \mathrm{d}\gamma$.
3. **Build sparse matrices:** store $\mathcal{T} = C^{-1}B^\top$ and $\mathcal{T}^\top$ as sparse matrices (CSR or similar) once, or compute them on-the-fly if the geometry is static.
4. **Apply transfers:** use simple matrix–vector products to evaluate $\boldsymbol{u}_s = \mathcal{T}\,\boldsymbol{u}_f$ and $\boldsymbol{f}_f = \mathcal{T}^\top \boldsymbol{f}_s$.
5. **Handle geometry updates:** if the solid deforms, recompute overlaps and the transfer operator at each time step (or when the geometry changes significantly).

## 1.6 Extension to penalty methods

In a penalty-based approach, the constraint $\boldsymbol{u}_f = \boldsymbol{u}_s$ on the interaction domain $I(t)$ is enforced approximately through penalty terms rather than exactly through Lagrange multipliers. The key difference from the Lagrange multiplier method is that we no longer solve an explicit constraint equation; instead, penalty stiffness terms are added directly to the fluid and solid equations.

### 1.6.1 Matrix assembly for the penalty formulation

The coupling matrices $B$ and $C$ are assembled identically to the Lagrange multiplier approach:

$$B_{i,j} := \int_{I(t)} \boldsymbol{\phi}_i \cdot \boldsymbol{\psi}_j \,\mathrm{d}\gamma,$$
$$C_{k,i} := \int_{I(t)} \boldsymbol{\varphi}_k \cdot \boldsymbol{\phi}_i \,\mathrm{d}\gamma,$$

using the same $L^2$-quasi-projection discretization with mass lumping described in Section **??**. The matrix $B$ relates fluid basis functions to multiplier basis functions, and $C$ relates solid basis functions to multiplier basis functions.

For the penalty method, however, we construct two additional composite matrices:

$$M_f^{\mathrm{pen}} := BB^\top \in \mathbb{R}^{N_f \times N_f}, \tag{1}$$
$$M_s^{\mathrm{pen}} := C^\top C \in \mathbb{R}^{N_s \times N_s}. \tag{2}$$

These matrices represent the penalty coupling stiffness contributions.

**Interpretation:**

- $M_f^{\mathrm{pen}} = BB^\top$ acts on the fluid velocity space and penalizes deviations from the constraint on the multiplier/interface space. Physically, it redistributes the penalty force back to fluid DOFs.
- $M_s^{\mathrm{pen}} = C^\top C$ acts on the solid velocity space and couples the solid to the constraint space. It can be computed as a Gramian and is naturally symmetric positive semi-definite (since $C$ has full column rank on typical grids).

### 1.6.2 Derivation: from strong form to the penalty-based system

To understand how $BB^\top$ and $C^\top C$ arise, we start from the strong form of the coupled FSI problem and apply the penalty method systematically.

**Step 1: Strong formulation**

The coupled fluid–structure system without explicit multipliers is:

$$\rho_f \partial_t \boldsymbol{u}_f - \nabla \cdot \boldsymbol{\sigma}_f = \boldsymbol{f}_f \quad \text{in } \Omega_f, \tag{3}$$
$$\rho_s \partial_t \boldsymbol{u}_s - \nabla \cdot \boldsymbol{\sigma}_s = \boldsymbol{f}_s + \boldsymbol{f}_{\mathrm{couple}} \quad \text{in } \Omega_s(t), \tag{4}$$
$$\text{Kinematic constraint:} \quad \boldsymbol{u}_f|_{I(t)} \approx \boldsymbol{u}_s|_{I(t)}. \tag{5}$$

where $\boldsymbol{f}_{\text{couple}}$ is the coupling force acting on the solid due to the fluid–structure interaction.

In the **penalty method**, we enforce the constraint (**??**) by adding a penalty force proportional to the velocity mismatch:

$$\boldsymbol{f}_{\text{couple}} = -\frac{1}{\epsilon}(\boldsymbol{u}_s - \boldsymbol{u}_f|_{I(t)}), \tag{6}$$

where $\epsilon > 0$ is the penalty parameter. By Newton's third law, the fluid experiences an equal and opposite reaction:

$$\boldsymbol{f}_{\text{reaction}} = \frac{1}{\epsilon}(\boldsymbol{u}_s - \boldsymbol{u}_f|_{I(t)}). \tag{7}$$

**Step 2: Variational formulation**

Taking the inner product of (**??**) with a test function $\delta\boldsymbol{u}_f$ and integrating over $\Omega_f$:

$$\int_{\Omega_f} \rho_f \partial_t \boldsymbol{u}_f \cdot \delta\boldsymbol{u}_f \, dx - \int_{\Omega_f} \nabla\cdot\boldsymbol{\sigma}_f \cdot \delta\boldsymbol{u}_f \, dx = \int_{\Omega_f} \boldsymbol{f}_f \cdot \delta\boldsymbol{u}_f \, dx + \int_{I(t)} \boldsymbol{f}_{\text{reaction}} \cdot \delta\boldsymbol{u}_f \, d\gamma. \tag{8}$$

The penalty reaction force term (on the interface $I(t)$) contributes:

$$\int_{I(t)} \frac{1}{\epsilon}(\boldsymbol{u}_s - \boldsymbol{u}_f) \cdot \delta\boldsymbol{u}_f \, d\gamma. \tag{9}$$

Similarly, for the solid:

$$\int_{\Omega_s(t)} \rho_s \partial_t \boldsymbol{u}_s \cdot \delta\boldsymbol{u}_s \, dx - \int_{\Omega_s(t)} \nabla\cdot\boldsymbol{\sigma}_s \cdot \delta\boldsymbol{u}_s \, dx = \int_{\Omega_s(t)} \boldsymbol{f}_s \cdot \delta\boldsymbol{u}_s \, dx - \int_{I(t)} \frac{1}{\epsilon}(\boldsymbol{u}_s - \boldsymbol{u}_f) \cdot \delta\boldsymbol{u}_s \, d\gamma. \tag{10}$$

**Step 3: Discretization with basis functions**

Discretize both fields:

$$\boldsymbol{u}_f^h = \sum_j U_j^f \boldsymbol{\psi}_j, \quad \text{(fluid basis)} \tag{11}$$

$$\boldsymbol{u}_s^h = \sum_i U_i^s \boldsymbol{\varphi}_i, \quad \text{(solid basis)} \tag{12}$$

and also represent the interface constraint using a multiplier-like basis (which will be integrated out):

$$\boldsymbol{\lambda}^h = \sum_k \Lambda_k \boldsymbol{\phi}_k, \quad \text{(multiplier basis on } I(t)) \tag{13}$$

The interface contributions become:

$$\int_{I(t)} (\boldsymbol{u}_s - \boldsymbol{u}_f) \cdot \boldsymbol{\phi}_k \, d\gamma \approx \left( \sum_i U_i^s C_{ki} - \sum_j U_j^f B_{kj} \right), \tag{14}$$

where we use the definitions:

$$B_{k,j} := \int_{I(t)} \boldsymbol{\phi}_k \cdot \boldsymbol{\psi}_j \, d\gamma, \tag{15}$$

$$C_{k,i} := \int_{I(t)} \boldsymbol{\phi}_k \cdot \boldsymbol{\varphi}_i \, d\gamma. \tag{16}$$

**Step 4: Eliminate the multiplier space**

Instead of explicitly solving for $\boldsymbol{\lambda}^h$, we can *project the penalty force back* onto the fluid and solid spaces. The penalty force density on the interface is:

$$\boldsymbol{f}_{\text{pen}} = \frac{1}{\epsilon} (\boldsymbol{u}_s - \boldsymbol{u}_f)|_{I(t)}. \tag{17}$$

When transferred to the fluid DOFs, this becomes a distributed load over the fluid cells:

$$\boldsymbol{F}_f^{\text{pen}} = B \cdot \frac{1}{\epsilon} (C\boldsymbol{u}_s - B^\top \boldsymbol{u}_f). \tag{18}$$

Here, $B$ acts as a spreading operator from the multiplier space to the fluid space. Expanding:

$$\boldsymbol{F}_f^{\text{pen}} = \frac{1}{\epsilon} (BC\boldsymbol{u}_s - BB^\top \boldsymbol{u}_f). \tag{19}$$

Similarly, the force on the solid is:

$$\boldsymbol{F}_s^{\text{pen}} = -C^\top \cdot \frac{1}{\epsilon} (C\boldsymbol{u}_s - B^\top \boldsymbol{u}_f) = -\frac{1}{\epsilon} (C^\top C\boldsymbol{u}_s - C^\top B^\top \boldsymbol{u}_f). \tag{20}$$

**Step 5: Monolithic system with penalty**

Combining the fluid and solid equations with these penalty forces:

$$A_f \boldsymbol{u}_f + \frac{1}{\epsilon} BB^\top \boldsymbol{u}_f - \frac{1}{\epsilon} BC\boldsymbol{u}_s = \boldsymbol{f}^f, \tag{21}$$

$$A_s \boldsymbol{u}_s + \frac{1}{\epsilon} C^\top C\boldsymbol{u}_s - \frac{1}{\epsilon} C^\top B^\top \boldsymbol{u}_f = \boldsymbol{f}^s, \tag{22}$$

which can be written in matrix form as:

$$\begin{bmatrix} A_f + \frac{1}{\epsilon} BB^\top & -\frac{1}{\epsilon} BC \\ -\frac{1}{\epsilon} C^\top B^\top & A_s + \frac{1}{\epsilon} C^\top C \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^f \\ \boldsymbol{f}^s \end{bmatrix}. \tag{23}$$

**Understanding the origin of $BB^\top$ and $C^\top C$:**

- $BB^\top$ **term:** The factor $\frac{1}{\epsilon}BB^\top$ appears in the fluid momentum equation because the penalty force $\boldsymbol{F}_f^{\text{pen}} = \frac{1}{\epsilon}B(C\boldsymbol{u}_s - B^\top\boldsymbol{u}_f)$ includes the product $BB^\top\boldsymbol{u}_f$. This term represents the *local stiffness* added to the fluid from penalizing the velocity mismatch.
- $C^\top C$ **term:** Similarly, $\frac{1}{\epsilon}C^\top C$ arises from the solid penalty force $\boldsymbol{F}_s^{\text{pen}} = -\frac{1}{\epsilon}(C^\top C\boldsymbol{u}_s - C^\top B^\top\boldsymbol{u}_f)$. The Gramian $C^\top C$ couples the solid DOFs through the multiplier space, creating a penalty stiffness that is naturally symmetric positive semi-definite.
- **Off-diagonal coupling:** The cross terms $-\frac{1}{\epsilon}BC$ and $-\frac{1}{\epsilon}C^\top B^\top$ represent the momentum transfer between fluid and solid through the penalty mechanism.

### 1.6.3 Practical assembly of penalty terms

In a partitioned code (e.g., in Python with NumPy/SciPy):

1. **Pre-compute $B$ and $C$:** Assemble both matrices once (or at each time step if geometry changes) using the same quadrature/overlap detection as in the Lagrange multiplier method. Store them as sparse matrices (CSR or COO format).
2. **Compute $BB^\top$:** Form the composite matrix
$$M_f^{\text{pen}} = BB^\top$$
by a sparse matrix product. This operation is efficient if $B$ is sparse (which it is for localized coupling).
3. **Compute $C^\top C$:** Similarly, assemble
$$M_s^{\text{pen}} = C^\top C.$$
If $C$ is already stored, use a transpose and product: `M_s_pen = C.T @ C` (in Python notation).
4. **Scale and add to system matrices:** In the fluid and solid system matrices, add the penalty stiffness:
$$A_f^{\text{pen}} = A_f + \frac{1}{\epsilon}M_f^{\text{pen}}, \qquad A_s^{\text{pen}} = A_s + \frac{1}{\epsilon}M_s^{\text{pen}}.$$
5. **(Optional) Off-diagonal coupling:** If using the fully coupled formulation, also assemble $BC$ and incorporate the term $-\frac{1}{\epsilon}BC$ into the off-diagonal block.

### 1.6.4 Adaptation to a partitioned (sub-iterative) solver

For a partitioned solver that decouples fluid and solid, the penalty method can be reformulated as an *implicit decoupling* with iterations:

**Partitioned Penalty Algorithm (per time step):**

1. Initialize $\boldsymbol{u}_f^{n+1,0} = \boldsymbol{u}_f^n$, $\boldsymbol{u}_s^{n+1,0} = \boldsymbol{u}_s^n$.
2. For $k = 0, 1, \ldots$ until convergence:
   (a) **Fluid sub-solve:** Compute the constraint mismatch force from the previous solid velocity:
   $$\boldsymbol{r}_{\mathrm{f}}^k := \frac{1}{\epsilon} B(\boldsymbol{u}_s^{n+1,k} - \mathcal{T}\boldsymbol{u}_f^{n+1,k}),$$
   where $\mathcal{T} = C^{-1}B^\top$ is the transfer operator (or use direct values). Solve the penalized fluid equation:
   $$\big(A_f + \frac{1}{\epsilon} BB^\top\big)\boldsymbol{u}_f^{n+1,k+1} = \boldsymbol{f}^f + \boldsymbol{r}_{\mathrm{f}}^k.$$
   (b) **Solid sub-solve:** Compute the reaction force from the penalized fluid:
   $$\boldsymbol{r}_{\mathrm{s}}^k := -\frac{1}{\epsilon} C^\top B^\top(\boldsymbol{u}_f^{n+1,k+1} - \boldsymbol{u}_s^{n+1,k}),$$
   or equivalently, from the constraint residual. Solve:
   $$\big(A_s + \frac{1}{\epsilon} C^\top C\big)\boldsymbol{u}_s^{n+1,k+1} = \boldsymbol{f}^s + \boldsymbol{r}_{\mathrm{s}}^k.$$
   (c) **Convergence check:** Stop if $\|\boldsymbol{u}_f^{n+1,k+1} - \boldsymbol{u}_f^{n+1,k}\| + \|\boldsymbol{u}_s^{n+1,k+1} - \boldsymbol{u}_s^{n+1,k}\| < \mathrm{tol}$.
3. Accept the converged solution and advance to the next time step.

**Key observations:**

- The penalty parameter $\frac{1}{\epsilon}$ scales the coupling strength. Smaller $\epsilon$ enforces the constraint more tightly, but the system becomes stiffer and more difficult to solve.
- The sub-iterations naturally couple the fluid and solid. For loose coupling (no sub-iterations), the constraint is satisfied only approximately with error $O(\epsilon)$.

- The matrices $BB^\top$ and $C^\top C$ are computationally expensive to assemble if $B$ and $C$ are large. Pre-compute them once and reuse; if geometry is static, cache them.
- Unlike the Lagrange multiplier method, there are no additional unknown fields (multipliers) to solve for; only the fluid and solid velocities are solved.

### 1.6.5 Comparison: Lagrange vs. Penalty in a partitioned framework

| **Aspect** | **Lagrange Multiplier** | **Penalty** |
|---|---|---|
| Constraint enforcement | Exact (to discretization error) | Approximate ($O(\epsilon)$ error) |
| Additional unknowns | Multiplier $\boldsymbol{\lambda}$ | None |
| System size | Larger (fluid + solid + multiplier) | Smaller (fluid + solid only) |
| Matrix assembly | $B, C, C^{-1}$ | $B, C, BB^\top, C^\top C$ |
| Coupling stiffness | Automatic via constraint | Manual via $\epsilon$ |
| Sub-iteration cost | Low (constraint already solved) | Higher (iterations needed for coupling |
| Stability | Robust; no penalty parameter | Sensitive to $\epsilon$ choice |

### 1.6.6 Numerical stability and penalty parameter selection

The penalty parameter $\epsilon$ controls the trade-off between constraint satisfaction and numerical stability:

- **Too large $\epsilon$:** The coupling is weak, the constraint $\boldsymbol{u}_f \approx \boldsymbol{u}_s$ is poorly satisfied, and solutions may not reflect true FSI physics.
- **Too small $\epsilon$:** The system becomes ill-conditioned. Linear solvers may fail or require many iterations. Round-off errors grow.
- **Sweet spot:** Empirically, $\epsilon \sim 10^{-2}$ to $10^{-4}$ times a reference stiffness (e.g., $\epsilon \sim 0.01 \cdot A_f$ or $\epsilon \sim 0.01 \cdot A_s$ in norm) often works well.

A practical heuristic:

$$\epsilon = \epsilon_0 \cdot \min(\Delta t, h^2, \text{characteristic length scale}),$$

where $\epsilon_0 \sim 0.01$–$0.1$ is a tunable dimensionless factor. Adapt $\epsilon$ per time step based on the condition number of the penalized system or the constraint residual.

In a partitioned explicit scheme (e.g., forward Euler for time), the penalty method can introduce additional time-step restrictions:

$$\Delta t \lesssim \frac{\epsilon \rho_s}{A_s}, \quad \Delta t \lesssim \frac{\epsilon \rho_f}{A_f},$$

so careful time-stepping control is essential.