Pore-Scale Modeling Framework with Embedded Boundaries and Automatic Jacobian Construction

Sahar Z. Amir[1] and Shuyu Sun[2]

[1] Prince Sultan University, Riyadh, Saudi Arabia
[2] Tongji University, Shanghai, China
szahmed@psu.edu.sa

Abstract. This paper presents a novel framework for pore-scale simulation of Stokes flow in complex porous media, with particular emphasis on the systematic construction of the global stiffness matrix (A matrix). The methodology integrates three key innovations: (1) an embedded boundary formulation on structured Cartesian grids that represents irregular solid grains through a cell-type classification system, eliminating the need for conforming mesh generation; (ii) a unified data structure (AVariable) that simultaneously stores field values and their Jacobians, enabling forward-mode automatic differentiation through operator overloading; and (3) a systematic restriction procedure that reduces the global system from the full grid to only active fluid cells. The framework architecture follows object-oriented design principles with clear component responsibilities, enabling extensibility to additional physics while maintaining computational efficiency through sparse matrix operations.

Keywords: Pore-scale modeling, Stokes flow, Finite difference method, Automatic differentiation, Jacobian matrix, Embedded boundary, Sparse linear systems.

1 Introduction

1.1 Motivation and Background

Understanding fluid flow through porous media is fundamental to numerous scientific and engineering applications, including groundwater hydrology [1], petroleum reservoir engineering [2], carbon sequestration [3], and fuel cell design [4]. The complex interplay between fluid dynamics and pore geometry governs macroscopic transport properties such as permeability, which are essential for predicting system behavior at larger scales.

Traditional approaches to porous media flow often rely on macroscopic Darcy's law [5], which relates flow rate to pressure gradient through a permeability tensor. However, Darcy's law represents an averaged description that cannot capture the detailed physics at the pore scale, including flow recirculation, stagnation zones, and the effects of pore geometry on local velocity fields [6].

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

Pore-scale modeling has emerged as a powerful tool to bridge this gap, resolving flow at the scale of individual pores and providing direct insight into fluid-solid interactions [7-11].

1.2 Literature Review

Pore-scale flow simulation has evolved along several methodological lines:
Lattice Boltzmann methods (LBM) [12,13] handle complex geometries via bounce-back conditions but can suffer instability at low viscosities [14], while *Finite volume methods (FVM)* on unstructured grids [15,16] offer geometric flexibility at the cost of challenging mesh generation [17]. *Immersed boundary methods (IBM)* [18,19,20] and ghost fluid methods [21] combine structured grid simplicity with complex geometry handling by modifying stencils near interfaces [22]. Staggered grid formulations [23] avoid checkerboard pressure oscillations [24] and are adopted in this work. *Automatic differentiation (AD)* has revolutionized scientific computing by enabling systematic computation of exact derivatives without truncation errors [25,26]. Recent work has extended AD to sparse matrix computations [27], making it applicable to large-scale simulations.

1.3 Contributions of This Work

This paper presents a novel framework that synthesizes these advances into a coherent methodology for pore-scale Stokes flow simulation:
Generating a unified data structure that simultaneously stores field values and their Jacobians, enabling automatic differentiation through all computational operations, eliminating manual derivative calculations, and ensuring consistency across the formulation.
Utilizing an embedded boundary formulation on structured grids that handles complex grain geometries without imitating mesh generation, using a cell-type classification system to distinguish between fluid, solid, and boundary regions.
Developing a systematic approach to boundary condition implementation through cell-type classification and modified stencils at fluid-solid interfaces, ensuring accurate representation of both no-slip and set pressure boundaries.
Applying efficient sparse matrix assembly for the coupled Stokes system, leveraging the sparse matrix capabilities and the automatic differentiation framework to construct the global stiffness matrix (A matrix) with minimal manual intervention.

2 Mathematical Formulation

2.1 Governing Equations for Stokes Flow

We consider steady, incompressible Stokes flow in a porous medium domain $\Omega \subset \mathbb{R}^2$. The Stokes equations, which govern creeping flows where inertial effects are negligible compared to viscous forces, are given by:

Mass conservation (incompressibility):

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

$$\nabla \cdot v = 0 \tag{1}$$

Momentum conservation:

$$\mu \nabla^2 v - \nabla p + f = 0 \tag{2}$$

In the equations, $v = (u, v)$ is the velocity vector, p is the pressure, μ is the dynamic viscosity, and f represents body forces (typically neglected in pore-scale studies).
These equations represent a balance between viscous stresses ($\mu \nabla^2 v$), pressure gradients (∇p), and external forces (f). The absence of inertial terms ($v \cdot \nabla v$) is justified when the Reynolds number $Re = \rho V L / \mu \ll 1$ is typical for flow in tight porous media [9].

2.2 Domain Characterization

The computational domain $\Omega_0 = [a_0, a_1] \times [b_0, b_1]$ is a rectangular region containing both fluid and solid phases. It forms a representative elementary volume (REV) of a porous medium, large enough to capture pore-scale heterogeneity but small enough for detailed simulation.

Within this domain, it is possible to distinguish between three fundamental regions: the pore space where fluid can flow, the solid grains that obstruct flow, and the boundaries through which fluid enters or exits. This is systematically handled by a cell-type classification function $\tau(x)$ that characterizes each point in the domain accordingly.

2.3 Boundary Conditions

The boundary $\partial\Omega$ consists of two distinct types that reflect the physical processes occurring at fluid-solid and fluid-fluid interfaces in porous media:

Inflow/outflow boundaries $\Gamma_p \subset \partial\Omega$ where pressure is prescribed $p = p_b$ and tangential velocity vanishes:

$$p = p_b \quad and \quad v.\tau = 0 \quad on\ \Gamma_p \tag{3}$$

These boundaries represent pore throats where fluid enters or exits the domain. The condition $v \cdot \tau = 0$ enforces that the flow is normal to the boundary, consistent with fully developed flow assumptions at inlets and outlets. The set pressure p_b drives the flow through the porous medium.

No-slip boundaries $\Gamma_v \subset \partial\Omega$ (fluid-solid interfaces) where velocity vanishes due to viscous effects:

$$v = 0 \quad on \quad \Gamma_v \tag{4}$$

This condition reflects the physical observation that fluid particles adhere to solid surfaces, a fundamental assumption in continuum fluid mechanics [29]. At the pore scale, this condition captures the frictional resistance provided by grain surfaces.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

The decomposition $\partial\Omega = \Gamma_p \cup \Gamma_v$ with $\Gamma_p \cap \Gamma_v = \emptyset$ naturally arises from the physics of pore-scale flow: fluid enters and exits through pore throats Γ_p while satisfying no-slip conditions on grain surfaces Γ_v. This decomposition is essential for well-posed problems, as it provides sufficient boundary conditions for the elliptic Stokes system.

Mathematical well-posedness: For the Stokes equations in a bounded domain, boundary conditions must be specified on all boundaries to ensure a unique solution [28]. The combination of pressure boundary conditions on Γ_p and velocity boundary conditions on Γ_v satisfies this requirement while remaining physically meaningful for porous media applications.

2.4 Integral Quantities and Flow Rates

Engineering applications are often interested in integral quantities derived from the solution. The most important of these is the flow rate through each boundary segment illustrated in Equation (5), where Γ_k represents the k-th inlet or outlet boundary. In discrete form, this becomes Equation (6), where v_n is the normal velocity component at the boundary edge and Δs is the edge length depending on the orientation.

$$Q_k = \int_{\Gamma_k} v.n \; ds \tag{5}$$

$$Q_k \approx \sum_{edges\ on\ \Gamma_k} v_n \cdot \Delta s \tag{6}$$

The permeability k of the porous medium can then be computed from Darcy's law as illustrated in Equation (7), where Q is the total flow rate through the domain, μ is the fluid viscosity, L is the domain length in the flow direction, A is the cross-sectional area, and Δp is the applied pressure difference. This computed permeability serves as a key upscaled parameter for reservoir-scale simulations

$$k = Q\mu L/A\Delta p \tag{7}$$

2.5 Equivalence to Poisson Equation

For verification purposes, this paper also considers the Poisson Equation, which represents a simplified version of the pressure equation that arises in many physical contexts:

$$-\nabla.(\nabla p) = f \;\; in\ \Omega \tag{8}$$

$$p = p_b \;\; on \;\; \partial\Omega \tag{9}$$

This elliptic equation serves as an essential test case for validating the core numerical methodology before tackling the coupled Stokes system [30]. The Poisson Equation shares key properties with the pressure equation in incompressible flow, including:

- Elliptic nature requiring global coupling
- Sensitivity to boundary conditions
- Second-order accuracy with standard discretization.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

Relationship to Stokes flow: Taking the divergence of the momentum equation (2) and applying the incompressibility condition (1) yields a Poisson equation for pressure (10), with boundary conditions derived from velocity constraints. This connection motivates using the Poisson equation as a building block for verifying numerical methods.

$$\nabla^2 p = \nabla . f \ \ in \ \Omega \tag{10}$$

3 Numerical Methodology

Before delving into the specific discretization schemes and solution algorithms, it is essential to understand how the mathematical formulations from Section 2 translate into a computational framework. Traditional approaches to solving PDEs typically involve: discretizing the continuous equations to obtain algebraic relationships, manually deriving the Jacobian matrix (or assembling the system matrix), and implementing specialized solvers for the resulting linear system.

While straightforward for simple problems, this process becomes increasingly complex in coupled systems like the Stokes equations, where multiple variables interact, and boundary conditions require careful treatment.

Instead of treating physical quantities merely as arrays of numbers, this paper represents them as differentiable objects (AVariable) that know both their current value and their sensitivity (how they change) with respect to every degree of freedom in the system. Mathematically, an AVariable instance encapsulates the values in Equation (11), where u is the vector of field values at grid points, and x represents the vector of all unknowns in the coupled system (pressure and velocity components).

$$u = (values, Jacobian) = (u, \partial u / \partial x) \tag{11}$$

The AVariable system automatically computes the residual values at all grid points, constructs the Jacobian matrix $\partial(mass_{eq}) \ / \ \partial x$ through automatic differentiation, maintains sparsity by storing only non-zero derivatives, and handles the chain rule through operator overloading.

The result is that the linear system $Ax = b$ needed for Newton's method emerges naturally, where r represents the residual of all equations.

$$A = \partial r / \partial x \, , \quad b = \ -r \tag{12}$$

Each continuous operator from Section 2 (gradient, divergence, Laplacian) has a discrete counterpart that, when applied to AVariable instances, produces new AVariable instances with appropriately propagated Jacobians. This creates a direct pipeline from the mathematical equations to the linear system, eliminating manual derivation and reducing coding errors.

3.1 Core Data Structure

The foundation of this paper's numerical framework is the AVariable data structure, which simultaneously stores field values and their derivatives with respect to all degrees

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

of freedom. This enables automatic differentiation through all subsequent operations, eliminating the need for manual Jacobian calculations and ensuring consistency across the entire formulation.

An AVariable object holds an array for the field values on the computational grid and a sparse Jacobian matrix holding the values of Equation (13):

$$A = \partial(field)/\partial(unknowns), \quad b = -r \tag{13}$$

By storing Jacobians alongside field values, the framework automatically computes derivatives of any expression. This forward-mode automatic differentiation provides several advantages:
Exact derivatives: No truncation errors from finite difference approximations.
Consistency: Derivatives are computed through the same code path as values.
Maintainability: Changes to the forward mode automatically update the derivatives.
Sparsity: The Jacobian matrices are naturally sparse due to local stencil operations.

Sparse matrix storage (Compressed Sparse Column format) is essential for large-scale problems, where storing a full Jacobian would be prohibitively expensive. For a typical discretization with second-order stencils, each degree of freedom couples only with its neighbors, yielding Jacobians with $O(N)$ non-zero entries rather than $O(N^2)$.

3.2 Operator Overloading for Automatic Differentiation

The AVariable abstraction derives its power from a carefully designed set of operator overloads that implement forward-mode automatic differentiation (AD). By systematically applying the chain rule [25], these operators enable the seamless propagation of both values and derivatives through every computational operation.

The true power of operator overloading AD emerges in composite expressions. Consider a sequence of operations that computes $w = f(u) + g(v)$, where f and g themselves could be composite functions. Through successive application of the overloaded operators, the Jacobian of w is automatically constructed as illustrated in Equation (14), where J_f and J_g are the Jacobians of the intermediate operations, themselves computed through the same mechanism. This automatic propagation extends to arbitrary computational graphs, making the system extremely flexible and maintainable.

$$J_w = J_f \cdot J_u + J_g \cdot J_v \tag{14}$$

Sparsity preservation — Overloaded operations preserve sparsity via sparse matrix arithmetic, keeping $A = \partial r / \partial x$ sparse for efficient solving. A subtle but critical implementation detail concerns Jacobian sparsity: when adding two AVariable instances, the framework performs sparse matrix addition only for non-zero entries. This seemingly minor choice reduces memory from $O(N^2)$ to $O(N)$ for typical stencil operations. For a 256×256 grid with 40% porosity, the full Jacobian would require approximately 17 GB if stored densely; the sparse representation requires less than 50 MB. Without this optimization, the method would be infeasible for pore-scale resolutions.
Accuracy and maintainability – Unlike numerical differentiation, which introduces truncation errors proportional to the step size, automatic differentiation computes exact

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

derivatives to machine precision. This exactness ensures that the Jacobian used in Newton iterations is consistent with the residual evaluation, leading to quadratic convergence rates for sufficiently good initial guesses. Furthermore, since derivatives are computed through the same code path as the forward-mode AD, updates to the physics automatically propagate to the Jacobian, eliminating a class of manual coding errors and significantly improving maintainability.

Staggered Arrangement – The paper employs a marker-and-cell (MAC) staggered grid [23] with pressures at cell centers and velocities at cell edges. This avoids pressure oscillations and ensures strong velocity-pressure coupling, offering: natural discretization of divergence and gradient; avoidance of odd-even decoupling; and compact stencils using only nearest neighbors, yielding minimal bandwidth

3.3 Domain Restrictions for Irregular Geometries

In porous media simulations, only a fraction of the computational grid contains fluid; solid walls and grains must be excluded from the solution. Restriction operators bridge this gap: equations are assembled on the full grid for simplicity, then restricted to fluid cells for solution. For matrix A, this involves two complementary operations:
Unknown restriction –Defines the column space (which degrees of freedom are active).
Residual restriction – Defines the row space (which equations are active).

Boolean masks derived from cell-type classification identify active degrees of freedom:
Pressure mask – True at cell centers where $\tau = 0$ (fluid).
x-velocity mask – True at x-edges where either adjacent cell is fluid.
y-velocity mask – True at y-edges where either adjacent cell is fluid.

Unknown restriction operates *before* assembly, selecting columns corresponding to active unknowns (pressure at fluid cells, velocity at active edges). Residual restriction operates *after* assembly, selecting rows corresponding to active equations (mass conservation at fluid cells, momentum at active edges).

Impact on Matrix A Structure

The restriction process directly shapes the final system matrix. Before restriction, the conceptual matrix has dimensions $N_{\text{total}} \times N_{\text{total}}$ with a regular block structure, but includes rows/columns for solid regions that would make the system singular. After restriction, the matrix dimensions reduce to $N_{\text{active}} \times N_{\text{active}}$ where $N_{\text{active}} \approx \phi N_{\text{total}}$, producing an irregular sparsity pattern that reflects the pore geometry while ensuring a well-posed system.

Boundary Condition Incorporation

Boundary conditions are implemented by applying modifications to the discrete equations based on a Cell Type classification. This systematic approach ensures consistency across complex geometries and eliminates the need for case-by-case coding.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

Fluid-solid interfaces: Detected where the mask changes between adjacent cells. These locations receive no-slip modifications (replacing momentum equations with $u = 0$). Results in rows with a single diagonal entry (Dirichlet conditions).

Inflow/outflow boundaries: Detected at domain boundaries with $\tau > 0$. These locations receive pressure boundary modifications. Alters gradient stencils to use set pressure values.

No-slip condition on solid boundaries: In both cases, for a boundary adjacent to a solid grain ($\tau < 0$), the velocity is set directly to zero $u_{i,j} = 0$.

These modifications are applied before residual restriction, ensuring they appear in the final A matrix.

Assembly Workflow for Matrix A

The complete process for constructing A is:

1. Initialize solution variables (p, u, v) on the full grid with identity Jacobians.
2. Apply unknown restriction using masks, defining A's column space
3. Assemble equations on a full grid using regular stencils
   - Each equation contributes row values to the matrix A through its Jacobian
   - Boundary conditions adjust specific rows
4. Apply residual restriction using masks, selecting active rows
5. Extract the final A matrix from the restricted residuals' Jacobians

This workflow ensures that the A matrix correctly represents the physics only in fluid regions while maintaining the programming convenience of regular grid assembly.

3.4 Boundary Condition Detection Concept

In a grid where each cell is either fluid or solid (treated as an obstacle), the framework needs to identify boundaries between both regions to enforce the correct physical conditions at these interfaces.

The algorithm illustrated in Fig. 1 and Table 1 could be utilized to identify interfaces between fluid cells (enabled = true) and solid cells (enabled = false) in a 2D grid. It presents a simplified proof-of-concept restricted to x- and y-directions. This controlled setting isolates the core logic of interface detection, providing a foundation that could be extended to curved geometries.

3.5 Management of Nested Unknown Structures

Realistic porous media simulations involve multiple physical fields (pressure, velocity components, additional transport equations) organized in hierarchical structures. To handle these systematically, the framework implements complementary flattening and unflattening operations.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

Flattening – recursively traverses nested collections (vectors, tuples, etc.) containing AVariable instances, extracting all individual objects into a linear array. This transformation is essential for assigning global degree of freedom numbers and concatenating Jacobian blocks into the global system matrix. Only the container hierarchy is dissolved; each AVariable retains its internal structure and identity.

Unflattening – reverses this process after solving the linear system, reconstructing the original nested structure and redistributing solution components to their expected positions. This bidirectional transformation is transparent to the user, who works with natural data structures while the framework handles linear algebra on flattened representations.

Table 1. Horizontal direction detection test with the left cell representing (i, j) and the right cell representing $(i + 1, j)$.

Left Cell (i, j); Right Cell $(i + 1, j)$	Interpretation	Conclusion
False; False	No west-facing boundary; No east-facing boundary; No interior edge.	No edge of any type.
False; True	Solid on left; Fluid on right; WEST-FACING BOUNDARY.	This is a boundary where fluid starts.
True; False	Fluid on left; Solid on right; EAST-FACING BOUNDARY.	This is a boundary where fluid ends.
True; True	Both cells are fluid; INTERIOR EDGE.	This is inside the fluid domain.

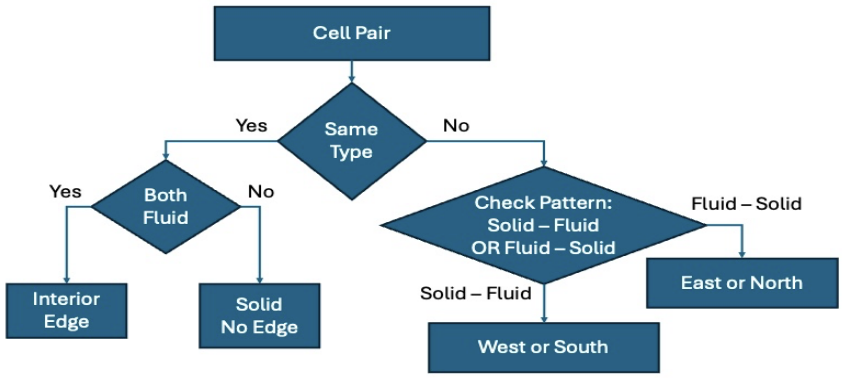

Fig. 1. Boundary Condition Detection Decision Flowchart.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

Global index assignment – establishes a one-to-one mapping between each degree of freedom (pressure at a fluid cell, velocity at an active edge) and its position in the global solution vector. The registration iterates through flattened unknowns, computing cumulative degree-of-freedom counts to determine starting indices for each variable block. It then expands each variable's local Jacobian into a global Jacobian block, transforming sensitivity to its own unknowns into sensitivity to all unknowns in the coupled system.

4 Software Architecture and Implementation Strategy

The numerical framework described in Sections 2 and 3 is realized through a carefully designed object-oriented software architecture that emphasizes modularity, reusability, and clarity. This section presents the class hierarchy, component responsibilities, and data flow that collectively enable efficient pore-scale Stokes flow simulation. The design philosophy centers on representing each physical quantity as a self-contained object that knows both its value and its sensitivity to all unknowns, thereby automating the construction of the coupled linear system.

4.1 Core Class Relationships

The software architecture comprises several key components with well-defined responsibilities and relationships. Fig. 2 presents a simplified Unified Modeling Language (UML) diagram showing the hierarchical structure and interactions among these components.

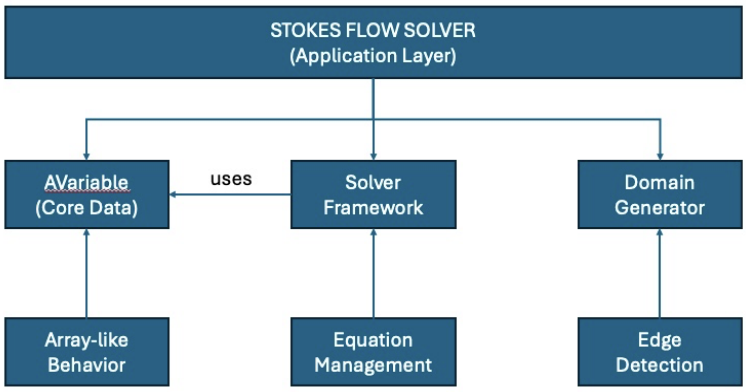

Fig. 2. UML Class Diagram of the Stokes Flow Solver Architecture.

AVariable (Core Data Structure): AVariable is the framework's fundamental building block with dual responsibility: storing field values on the grid while maintaining sensitivity to all degrees of freedom. Each instance contains:

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

- *A multi-dimensional array of floating-point values* (pressure, velocity components, or any other scalar quantity) at grid points.
- *A sparse Jacobian matrix in Compressed Sparse Column (CSC) format* storing $\partial(field)/\partial(unknowns)$, where columns map to global degrees of freedom, and rows map to grid points.

AVariable behaves like an array through overloaded operations:

- *Indexing operations:* Access grid points/subarrays with automatic translation between multi-dimensional and linear indexing.
- *Size queries:* Return grid dimensions, enabling dimension-agnostic code.
- *Axes information:* Provide index ranges for bounds checking.

This array-like design is intentional. Physical fields on structured grids should use intuitive notation, whether they are simple arrays or differentiable quantities, shielding users from Jacobian complexity while enabling automatic differentiation.

Solver Framework (Equation Management): transforms continuous equations into discrete linear systems by orchestrating registration, restriction, and solution phases on AVariable instances.

Domain Generator: produces computational domains ranging from simple test cases to statistically representative porous microstructures, enabling systematic control over porosity and grain size distribution.

4.2 Component Interactions and Data Flow

The interaction among components follows a well-defined pipeline that transforms geometric descriptions into solved flow fields. Fig. 3 illustrates the complete data flow through the system. The data flow proceeds through these stages:

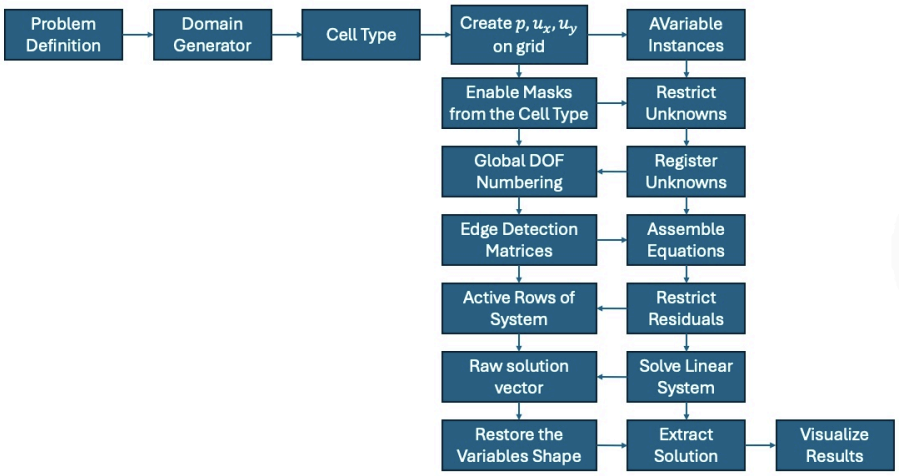

Fig. 3. Data Flow Diagram for Stokes Flow Simulation.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

1. *Domain creation:* The Domain Generator produces a cellType matrix and coordinates two arrays defining the computational grid and material phases.
2. *Variable initialization:* AVariable instances are created for pressure and velocity components on their respective staggered grid locations.
3. *Unknown restriction:* Boolean masks derived from cellType are applied to restrict unknowns to active (fluid) cells.
4. *Unknown registration:* The solver assigns global degree-of-freedom numbers, expanding each variable's Jacobian to span all unknowns.
5. *Equation assembly:* Using edge-detection matrices to identify boundaries, the discrete Stokes equations are assembled through arithmetic on AVariable instances.
6. *Residual restriction:* The assembled equations are restricted to active cells, producing the final system rows.
7. *Linear solution:* The global system is solved using sparse direct methods.
8. Solution extraction: Results are mapped back to the full grid by restoring the variable shape.
9. *Visualization:* Computed fields are displayed using plotting libraries.

5 Framework Results and Discussion

Framework validation uses a representative test case with simplified pore geometry (Fig. 4). This demonstrates the step-by-step model construction—from the initial grid to the final solution—and confirms that the method captures essential porous media flow physics. A single cell at a fluid-solid interface is selected for detailed examination, as boundary conditions require careful enforcement at this location.

The simulation results in Fig. 5 (a) align with the expected physics: higher pressure values close to the pore throat drive flow through the constriction, with the largest pressure gradient where flow accelerates. The velocity fields in Fig. 5 (b) reveal detailed structure within a single cell, demonstrating the method's capability to resolve boundary layers and flow acceleration at sub-cell scales.

While the validation presented above uses a simplified pore geometry, the framework is implemented to support arbitrarily complex porous microstructures. The domain generator (illustrated in Fig.3) could be set to produce random sphere packings via sequential addition with overlap rejection. The restriction operators (Section 3.3) operate on Boolean masks derived from any domain indicator function—whether a single circle, random sphere packing, or digital rock image—with computational complexity.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

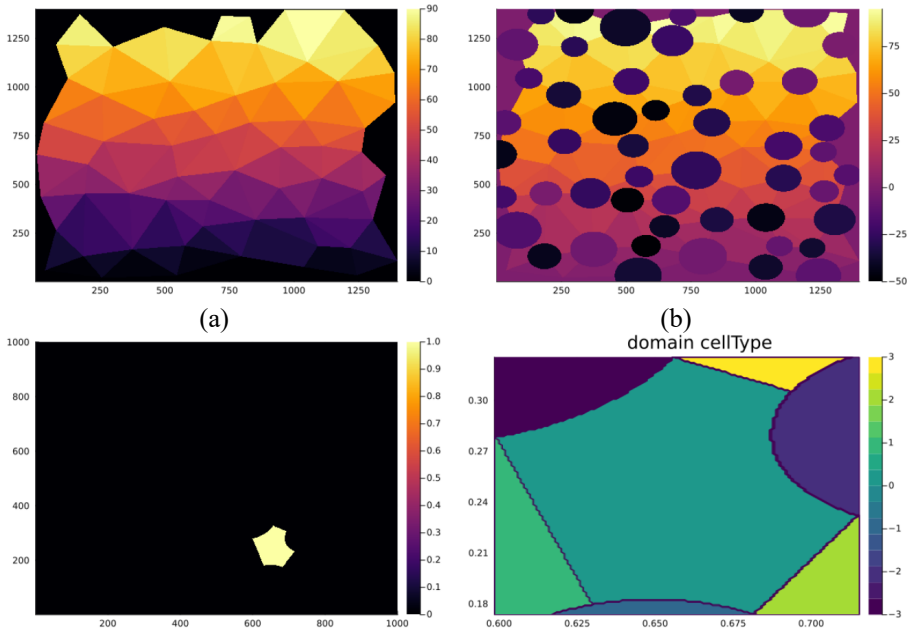

Fig. 4. Step-by-step construction of the computational model: (a) Structured grid before the obstacles introduction; (b) Introduction of solid obstacles to the system; (c) Selection of a single computational cell for detailed analysis; (d) Magnified view of the single cell from (c) showing a fluid region and solid boundaries.

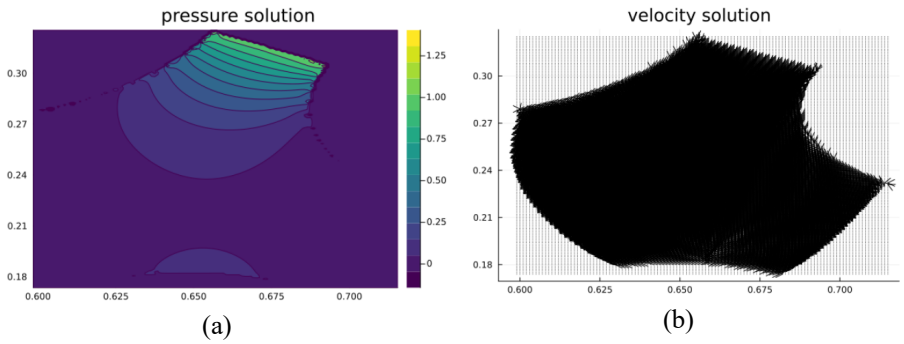

Fig. 5. Results after modeling and simulation: (a) Computed pressure field within a single interface cell; (b) Computed velocity vectors within the selected cell.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

5.1 Convergence Verification Methodology

The convergence properties of the framework are verified using the Poisson equation as a model problem. A manufactured solution approach is employed, prescribing an analytical function $p_{exact}(x, y) = \sin(\pi x) \sin(\pi y)$ on $\Omega = [0,1]^2$ with homogeneous Dirichlet conditions. The corresponding source term $f = -\nabla^2 p_{exact}$ and boundary conditions are derived.

For a sequence of grids with cell counts $N = 10,20,40,80,160$ (grid spacings $h = 1/N$ ranging from 10^{-1} to 6.25×10^{-3}), the discrete L^2 error norm is computed as:

$$\| p_{\text{exact}} - p_h \|_{L^2} \approx \sqrt{\sum_{i,j} (p_{\text{exact},ij} - p_{h,ij})^2 h_x h_y} \tag{15}$$

where the summation runs over all fluid cells. The convergence order is estimated as order $= \log(E_N/E_{2N})/\log(2)$, where E_N denotes the error on a grid with N cells and E_{2N} the error on the next finer grid (halving h).

6 Conclusion

This paper demonstrated that a framework combining embedded boundary methods with automatic differentiation through a unified data structure efficiently simulates Stokes flow in complex porous media. The systematic construction of the global stiffness matrix—from cell-type classification through restriction, registration, and assembly—provides a blueprint for physically realistic and computationally tractable simulation tools. As pore-scale modeling advances, such frameworks help bridge fundamental physics, computational modeling, data science, and engineering applications.

Acknowledgments. The work conducted by Sahar Z. Amir is funded by Prince Sultan University. The work conducted by Shuyu Sun is supported by the National Key Research and Development Project of China (Grant No. 2023YFA1011701), the National Natural Science Foundation of China (Grant No. 12571466), the Fundamental Research Funds for the Central Universities, the Shanghai Magnolia Talent Fund (Innovation Talent Category) of Shanghai Municipal Human Resources and Social Security Bureau, and the Chang Jiang Scholars Program of the Ministry of Education of China.

Disclosure of Interests. The authors declare that they have no competing interests.

References

1. Bear, J. (1972). Dynamics of Fluids in Porous Media. Elsevier.
2. Lake, L. W. (1989). Enhanced Oil Recovery. Prentice Hall.
3. Benson, S. M., & Cole, D. R. (2008). CO2 sequestration in deep sedimentary formations. Elements, 4(5), 325-331.
4. Wang, C. Y. (2004). Fundamental models for fuel cell engineering. Chemical Reviews, 104(10), 4727-4766.
5. Darcy, H. (1856). Les Fontaines Publiques de la Ville de Dijon. Dalmont.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14

6.Adler, P. M. (1992). Porous Media: Geometry and Transports. Butterworth-Heinemann.
7.Blunt, M. J. (2001). Flow in porous media—pore-network models and multiphase flow. Current Opinion in Colloid & Interface Science, 6(3), 197-207.
8.Hazlett, R. D. (1997). Statistical characterization and stochastic modeling of pore networks. Mathematical Geology, 29(6), 801-822.
9.Dullien, F. A. L. (1992). Porous Media: Fluid Transport and Pore Structure (2nd ed.). Academic Press.
10.Andrade, J. S., et al. (1999). Inertial effects on fluid flow through disordered porous media. Physical Review Letters, 82(26), 5249-5252.
11.Blunt, M. J., et al. (2013). Pore-scale imaging and modelling. Advances in Water Resources, 51, 197-216.
12.Chen, S., & Doolen, G. D. (1998). Lattice Boltzmann method for fluid flows. Annual Review of Fluid Mechanics, 30(1), 329-364.
13.Succi, S. (2001). The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press.
14.Aidun, C. K., & Clausen, J. R. (2010). Lattice-Boltzmann method for complex flows. Annual Review of Fluid Mechanics, 42, 439-472.
15.Versteeg, H. K., & Malalasekera, W. (2007). An Introduction to Computational Fluid Dynamics (2nd ed.). Pearson.
16.Moukalled, F., et al. (2016). The Finite Volume Method in CFD. Springer.
17.Thompson, J. F., et al. (1999). Handbook of Grid Generation. CRC Press.
18.Mittal, R., & Iaccarino, G. (2005). Immersed boundary methods. Annual Review of Fluid Mechanics, 37, 239-261.
19.Peskin, C. S. (2002). The immersed boundary method. Acta Numerica, 11, 479-517.
20.Peskin, C. S. (1972). Flow patterns around heart valves. Journal of Computational Physics, 10(2), 252-271.
21.Fedkiw, R. P., et al. (1999). A non-oscillatory Eulerian approach to interfaces in multimaterial flows. Journal of Computational Physics, 152(2), 457-492.
22.Johansen, H., & Colella, P. (1998). A Cartesian grid embedded boundary method for Poisson's equation. Journal of Computational Physics, 147(1), 60-85.
23.Harlow, F. H., & Welch, J. E. (1965). Numerical calculation of time-dependent viscous incompressible flow. The Physics of Fluids, 8(12), 2182-2189.
24.Patankar, S. V. (1980). Numerical Heat Transfer and Fluid Flow. Hemisphere.
25.Griewank, A., & Walther, A. (2008). Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation (2nd ed.). SIAM.
26.Giles, M. B., & Pierce, N. A. (2000). An introduction to the adjoint approach to design. Flow, Turbulence and Combustion, 65(3-4), 393-415.
27.Griewank, A., et al. (2016). Algorithmic differentiation of nonlinear multigrid computations. Optimization Methods and Software, 31(3), 522-540.
28.Temam, R. (1979). Navier-Stokes Equations: Theory and Numerical Analysis (2nd ed.). North-Holland.
29.Batchelor, G. K. (1967). An Introduction to Fluid Dynamics. Cambridge University Press.
30.LeVeque, R. J. (2007). Finite Difference Methods for Ordinary and Partial Differential Equations. SIAM.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_14