https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

High-Performance Geodynamics on GPUs Using the PETSc CUDA Backend (GAUZZ v1.0.0)

Kyeong-Min Lee[1], Deok-Kyu Jang[3], Cedric Thieulot[4], and Byung-Dal So[1,2]

[1]Interdisciplinary Program in Earth Environmental System Science & Engineering, Kangwon National University, Chuncheon, Republic of Korea
[2]Department of Geophysics, Kangwon National University, Chuncheon, Republic of Korea
[3]Research Institute for Mathematical Sciences, Kangwon National University, Chuncheon, Republic of Korea
[4]Department of Earth Sciences, Utrecht University, Princetonlaan 8A, The Netherlands

Correspondence: Byung-Dal So (bdso@kangwon.ac.kr)

Abstract. The GPU is a computing architecture designed for parallelism in vector and matrix operations. To solve the Stokes equations in geodynamics, we apply a preconditioned conjugate-gradient (PCG) method to the pressure Schur complement system. The algorithm is dominated by sparse matrix-vector products, vector inner products, vector updates, and repeated velocity subproblem solves, all of which map onto GPU architectures. We implement this solver on GPUs by coupling CUDA-enabled PETSc with FEniCS. PETSc AIJCUSPARSE matrices with CUDA vectors keep operators and field vectors resident in VRAM, which limits CPU-GPU transfers during iterations. The P_1 mass matrix, the gradient operator, and the divergence operator are reused in the pressure-correction L^2-projection step. HYPRE BoomerAMG was adopted for preconditioning the velocity-only subproblem. Accuracy and performance are evaluated on manufactured solutions, the SolCx benchmark, the sticky-air benchmark, 2D and 3D Rayleigh-Taylor instabilities, 3D thermal convection, and a 2D subduction model with nonlinear viscosity. We confirm that the GPU-based implementation reproduces CPU solutions. We distinguish between the execution time, which includes only the pressure correction step, and the wall time, which includes the workflow from mesh generation to Stokes solves. A single-GPU environment achieves a 5–11$\times$ reduction in execution time relative to a 32-core CPU. For nonlinear viscosity cases with per-step matrix updates, the wall time is reduced by 1.14–3.46$\times$ on a single GPU relative to a 32-core CPU. Using Multi-Process Service to coordinate two-GPU with a 16-core CPU reduces the wall time by 5.83$\times$ compared to a 32-core CPU.

1 Introduction

Numerical geodynamics modelling has advanced the understanding of plate tectonics and planetary interior physics, including subduction (Yoshida et al., 2012; Keum and So, 2021, 2023), rifting (Huismans and Beaumont, 2003; Brune et al., 2012; Wolf et al., 2022), folding (Schmalholz, 2008; Do et al., 2023, 2025), plumes (d'Acremont et al., 2003; Gerya et al., 2015; Wang and Li, 2021), and convection (Davies and Gurnis, 1986; Crameri and Tackley, 2015). Over geological time scales spanning millions of years, solid planetary materials flow at low Reynolds numbers due to high viscosity (Gassmöller et al.,

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

2020; Pysklywec and Shahnas, 2003). Thermal heterogeneity and density contrasts drive buoyant flow, governing the heat and material redistribution (Kronbichler et al., 2012; Adams et al., 2023; Regorda et al., 2023; Schools and Smrekar, 2024).

Geodynamic fluid motion is represented by creeping Stokes flow at an infinite Prandtl number (Tan and Gurnis, 2007; Liao and Gerya, 2015). The velocity field obtained from the Stokes equations drives the advection of heat and mantle material, capturing the evolution of planetary interiors from the surface to the core (Schmeling, 2010; Burstedde et al., 2013). Resolving the coupled, time-dependent Stokes-advection system requires repeated solutions of the Stokes operator. At each time step, the viscosity and density fields are updated, where viscosity is often nonlinearly dependent on temperature, pressure, composition, and strain rate, which substantially increases the computational cost. Strategies for solving the Stokes equations include direct factorization methods such as LU or LDL^T decomposition (Amestoy et al., 2000; Schenk et al., 2001) and iterative Krylov subspace methods that converge within a specified tolerance (Hestenes and Stiefel, 1952; Saad and Schultz, 1986; Muzhinji et al., 2016).

Efficient solvers for the nonlinear Stokes equations remain essential for numerical simulations of mantle flow with temperature- and strain-rate-dependent viscosity. The strength of mantle and lithospheric rock follows a power-law rheology that links stress to strain rate, rendering viscosity a function of the strain-rate invariant derived from velocity gradients. Newton and Picard iterations are used to solve for the velocity field. For each nonlinear iteration, the solver linearizes the strongly nonlinear Stokes system and updates viscosity based on the most recent velocity estimate. The iterative procedure repeats until velocity and pressure converge, constituting a principal source of computational expense in geodynamic models due to the tight coupling between viscosity and velocity.

Three dimensional (3D) geodynamic simulations resolved on a $N \times N \times N$ grid with finite elements $P_2 \times P_1$ yield $O(N^3)$ degrees of freedom (DOFs). A direct solver requires memory that grows as $O(N^{4/3})$ (Saramito, 2016). To overcome memory limitations, Krylov methods such as GMRES or MINRES are commonly used for the full saddle-point Stokes system, while CG is typically applied to symmetric positive-definite sub-blocks (e.g., velocity blocks or Schur complements). The convergence rate of an iterative solver depends critically on the choice of preconditioner. Multigrid preconditioning represents standard practice for simulations at high spatial resolution. Algebraic multigrid (AMG) accelerates convergence by automatically constructing a hierarchy of coarse level representations from the matrix coefficients alone, without requiring an explicit coarse mesh. This property makes AMG particularly suitable for unstructured finite-element discretization.

Many geodynamic problems, such as incompressible viscoelasticity (Zhong et al., 2003; Beuchert and Podladchikov, 2010) and the Stokes equations (Tan and Gurnis, 2007), are formulated as saddle-point problems imposing an incompressibility constraint. Eliminating the velocity unknown from the saddle-point system yields the Schur complement equation for pressure, which can be solved iteratively (Uzawa, 1958; Moresi et al., 1996; Leng and Zhong, 2008). At each iteration, the current pressure p enters the source term, and the algorithm subsequently computes the velocity components (u, v, w) from the velocity-only submatrix. The velocity-only submatrix of the Stokes system is symmetric positive definite. AMG-preconditioned CG or GMRES solves the velocity subproblem efficiently (Clevenger and Heister, 2021). Each pressure-correction iteration requires one such velocity subproblem solve, making it the dominant computational cost. A widely used alternative is to solve the full saddle-point system with block-preconditioned FGMRES, where the Schur complement is approximated by a viscosity-

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

weighted mass matrix or BFBT-type preconditioner (Benzi et al., 2005; May and Moresi, 2008; Rudi et al., 2017). However, even with restart, FGMRES stores substantially more Krylov basis vectors per cycle than the CG-based approach and restart itself may degrade convergence. In contrast, CG applied to the pressure Schur complement system requires only a fixed, small number of pressure- and velocity-space vectors, offering a clear advantage in GPU environments where VRAM is limited. As problem size grows in 3D, where DOFs scale as $O(N^3)$, the velocity subproblem solve increasingly dominates the total computation time, and efficient preconditioning becomes essential. Several CPU-based geodynamic community codes have been developed to handle such complex systems, including ASPECT (Kronbichler et al., 2012), Fluidity (Davies et al., 2011), CitcomS (Zhong et al., 2008), FANTOM (Thieulot, 2011), Underworld2 (Mansour et al., 2020), and pTatin3D (May et al., 2015). Such packages have been widely applied to simulate large-scale geodynamic processes, such as continental rifting (Neuharth et al., 2021), subduction (Suchoy et al., 2021), and mantle plume dynamics (Xie et al., 2024; Roy et al., 2024). Parallel execution is typically achieved through MPI (Message Passing Interface) libraries such as MPICH and Open MPI (Thakur et al., 2005; Graham et al., 2005). The MPI-based architecture delivers strong scalability, yielding near-linear speedup on thousands of CPU cores (Burstedde et al., 2013).

In CPU architecture, individual cores offer high single-thread performance. However, in high-resolution simulations involving repeated access to large sparse matrices and associated vectors, the limited memory bandwidth frequently functions as the primary bottleneck, leading to significant performance degradation (May et al., 2015). In contrast, a graphics processing unit (GPU) provides access to thousands of CUDA cores (e.g., NVIDIA RTX 6000 Ada: 18,176 CUDA cores) concurrently, enabling massively parallel execution of operations such as matrix-vector products, an architectural advantage that clearly distinguishes GPUs from conventional CPUs. Moreover, GPUs sustain memory bandwidth on the order of several hundred gigabytes per second, surpassing those of similarly priced CPUs (RTX 6000 Ada: 960 GB/s; Intel Xeon Platinum 8568Y+: 358 GB/s). Building on these hardware advantages, NVIDIA's Single Instruction Multiple Threads (SIMT) architecture provides efficient thread-level scheduling, further enhancing the performance of GPU-accelerated numerical modelling (Ta et al., 2015).

To date, in geodynamic modelling, GPU-based solutions of the Stokes equations have primarily focused on pseudo-transient (PT) approaches, in which a pseudo-inertial term is added to enable explicit time stepping (Räss et al., 2022; Wang et al., 2022; Alkhimenkov and Podladchikov, 2024). PT methods reduce memory requirements by avoiding full implicit solves, making them attractive for GPU implementations. Subsequent developments led to the accelerated pseudo-transient method, which allows flexible time steps and preconditioning strategies. By exploiting the mass matrix to treat the stiffness matrix explicitly, the PT method achieves a substantial reduction in memory usage. However, explicit pseudo-time stepping can require a large number of iterations to converge, particularly for problems with strong viscosity contrasts (Räss et al., 2022).

PETSc (Portable, Extensible Toolkit for Scientific Computation; Balay et al. 2023) provides a unified interface to Krylov subspace solvers, sparse matrix storage, and external preconditioner packages across both CPU and GPU environments. Among its external interfaces, the HYPRE library and its BoomerAMG algebraic multigrid preconditioner (Falgout et al., 2006) are particularly relevant to geodynamic applications, where viscosity contrasts of 10^3–10^6 render the discretized Stokes system highly ill-conditioned. Since version 3.16 (released in September 2021), PETSc has supported GPU-resident execution of HYPRE preconditioners, enabling the entire Krylov iteration, including matrix–vector products, vector operations, and AMG

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

preconditioning cycles, to remain on the GPU, minimising host–device data transfer. However, in geodynamic modelling, the performance of BoomerAMG on GPUs has not yet been systematically evaluated, and quantitative comparisons between the GPU and CPU environments remain limited.

In this study, we developed a solver for the Stokes system using FEniCS, an open-source finite element platform (Alnæs et al., 2015). The saddle-point system is reduced via the Schur complement, and the resulting pressure equation is solved by a preconditioned conjugate-gradient (PCG) method with the inverse viscosity mass matrix. Both CPU and GPU execution paths are provided. The CPU path links against PETSc 3.20, while the GPU path links against PETSc 3.23.2 compiled with CUDA support, storing all matrices and vectors in GPU VRAM using the AIJCUSPARSE and CUDA vector types. To benchmark CPU and GPU performance, we selected a suite of benchmark problems from computational geodynamics, including manufactured solutions, the SolCx viscosity-discontinuity benchmark, sticky-air free-surface tests, Rayleigh-Taylor instabilities in 2-D and 3-D, 3-D thermal convection, and a subduction scenario with nonlinear mantle viscosity. For problems that involve advection, we also evaluated the performance of the advection equation solver. A comparison with block-preconditioned Stokes solvers widely adopted in geodynamic community codes is presented in Section 5.4. All experiments were performed on three machines with different performance characteristics. The first workstation used an Intel Xeon Gold 5320 processor (2.20 GHz, 39 MB cache). The second workstation used an Intel Xeon w5-2465X processor (3.10 GHz, 33.75 MB cache) and a GeForce RTX 6000 Ada GPU with 48 GB of memory. The third workstation used the same processor but was paired with two GeForce RTX 6000 Ada GPUs, each with 48 GB of memory (96 GB in total). Unlike previous GPU-based geodynamic studies, which primarily rely on pseudo-transient formulations, our study directly evaluates the performance of Krylov solvers with algebraic multigrid preconditioning for mixed finite-element Stokes systems on GPUs.

2 Methods

2.1 Governing equations

Lithospheric deformation and mantle flow associated with geodynamic processes are described by the Stokes equations for incompressible viscous creeping flow (Equations 1 and 2). We incorporated the thermal (Ra) and compositional (Rb) Rayleigh numbers (Equation 3) into the body-force term of the Stokes equations (e.g., van Keken et al. 1997; Tan et al. 2011).

$$\nabla \cdot \boldsymbol{\tau} - \nabla p + [RaT - \sum_i Rb_i C_i]\hat{\mathbf{z}} = \mathbf{0} \quad \text{where} \quad \boldsymbol{\tau} = 2\eta\dot{\boldsymbol{\varepsilon}} \quad \text{and} \quad \dot{\boldsymbol{\varepsilon}} = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

$$Ra = \frac{\alpha\rho_0 g\Delta T h^3}{\kappa\eta_0} \quad \text{and} \quad Rb_i = \frac{\Delta\rho_i g h^3}{\kappa\eta_0} \tag{3}$$

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

p, $\mathbf{u}$, $\boldsymbol{\tau}$, η, $\hat{\mathbf{z}}$, and $\dot{\boldsymbol{\varepsilon}}$ denote pressure, velocity vector, the deviatoric stress tensor, shear viscosity, the unit vector in the direction of gravity, and the deviatoric strain-rate tensor, respectively. T and C refer to the temperature and composition fields, respectively. κ, η_0, h, and g are thermal diffusivity, reference viscosity, the depth of the domain, and gravitational acceleration, respectively. α, ρ_0, ΔT, and $\Delta\rho_i$ represent the thermal expansivity, reference density, temperature contrast between the bottom and top, and the density contrast of the i-th composition relative to ρ_0, respectively. For the Stokes system, we employ two mixed finite-element pairs, the Taylor–Hood element ($P_2 \times P_1$) and the MINI element ($P_1^+ \times P_1$). Both discretizations use the same P_1 pressure space, while the P_1^+ velocity space enriches P_1 with bubble functions. The pair $P_1^+ \times P_1$ is commonly referred to as the MINI element in the geodynamics literature (Thieulot and Bangerth, 2025). The MINI element was adopted to reduce the total number of DOFs in high-resolution and 3D simulations. The thermal evolution of the system is governed by the advection-diffusion equation (Equation 4).

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \nabla \cdot (\nabla T) \tag{4}$$

The temporal evolution of the multi-composition field depends only on advection through the velocity field. We adopted a level-set method to reduce oscillations and numerical diffusion of the compositional field under advection-dominated problem (Hillebrand et al., 2014; Wú et al., 2023). The level-set function ϕ is a continuous function that represents the distance between an arbitrary point and the interface. The different material domains are distinguished by $\phi < 0$ and $\phi \geq 0$. Using the composition field obtained via the level-set functions, density and the logarithm of viscosity were linearly interpolated. To define n compositions, $n-1$ level-set functions are required. We employed one to four level-set functions (one for Rayleigh–Taylor instability and four for the subduction model).

2.2 Saddle-point problem and Schur complement solver

Discretization of the incompressible Stokes equations yields a 2×2 block saddle-point system (Equation 5).

$$\begin{pmatrix} K & G \\ G^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{5}$$

The velocity stiffness matrix K is symmetric and positive definite (Davies et al., 2011). G and G^T represent the gradient and divergence operators for the pressure and velocity fields, respectively (Silvester and Wathen, 1994; Benzi et al., 2005; May and Moresi, 2008). f refers to body force vector. In geodynamic applications, the viscosity contrast is large ranging from 10^3 to 10^6, which leads to an ill-conditioned system and slow convergence of iterative solvers. Furthermore, using a direct solver (e.g., MUMPS) on the full block system causes excessive memory consumption for large-scale problems. Therefore, iterative methods are essential for large-scale problems. One effective approach is to exploit the Schur complement structure of the saddle-point system.

With Schur complement reduction, the Stokes system is transformed into a block upper triangular form (Wathen and Silvester, 1993).

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

$$\begin{pmatrix} K & G \\ 0 & S \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ \hat{h} \end{pmatrix}, \quad S = -G^T K^{-1} G, \quad \hat{h} = -G K^{-1} f \tag{6}$$

To solve the equivalent system $G^T K^{-1} G p = G K^{-1} f$, we apply a PCG iteration with the inverse viscosity mass matrix as preconditioner (Equations 7–12; Jang et al., 2025).

Initialization:

$$p^0 = 0, \quad K u^1 = f - G p^0 \qquad \text{(initial solve)} \tag{7}$$

$$\langle q^1, r\rangle = \langle \nabla \cdot u^1, r\rangle, \quad \langle \tfrac{1}{\eta} w^1, r\rangle = \langle q^1, r\rangle, \quad d^1 = -w^1 \qquad \text{(initial residual \& precond.)} \tag{8}$$

For $k = 1, 2, \cdots$:

$$K h^k = G d^k, \quad \alpha^k = \frac{\langle q^k, w^k\rangle}{\langle \nabla d^k, h^k\rangle} \qquad \text{(inner solve \& step size)} \tag{9}$$

$$p^{k+1} = p^k + \alpha^k d^k, \quad u^{k+1} = u^k - \alpha^k h^k \qquad \text{(vector update)} \tag{10}$$

$$\langle q^{k+1}, r\rangle = \langle \nabla \cdot u^{k+1}, r\rangle, \quad \langle \tfrac{1}{\eta} w^{k+1}, r\rangle = \langle q^{k+1}, r\rangle \qquad \text{(divergence \& precond.)} \tag{11}$$

$$\beta^k = \frac{\langle q^{k+1}, w^{k+1}\rangle}{\langle q^k, w^k\rangle}, \quad d^{k+1} = -w^{k+1} + \beta^k d^k \qquad \text{(CG update)} \tag{12}$$

The velocity and pressure at k-th iteration (u_k, p_k) are updated by the conjugate direction d_k in each iteration. The iteration stops when the divergence error ($\|\nabla \cdot \mathbf{u}\|/\|\mathbf{u}\|$) of velocity is lower than the divergence tolerance.

The algorithm (Equations 7–12) requires only a fixed number of vectors: approximately four in the pressure space (q^k, d^k, p^k, w^k) and two in the velocity space (h^k, u^k). Outside the velocity subproblem solve (Equation 9), the remaining operations-vector inner products, scalar–vector multiplications, and vector additions (Equations 10 and 12)-are memory bandwidth limited and map directly onto GPU architectures. The dominant cost per iteration is the AMG-preconditioned GMRES solve of the velocity subproblem $K h^k = G d^k$ (Equation 9), which involves repeated sparse matrix-vector products and preconditioner applications. The L^2-projection steps that apply the mass matrix and the inverse viscosity mass matrix (Equations 8 and 11) also offer substantial potential for GPU acceleration. Evaluating the GPU performance of these two stages the velocity subproblem solve and the L^2-projection is therefore central to assessing the overall solver efficiency.

3 Numerical implementation

3.1 Implementation algorithms on GPUs and CPUs

To evaluate the computational efficiency (parallel performance) and accuracy (regarding incompressibility) of solving 2D and 3D geodynamic problems, we established independent CPU-based and GPU-based computing environments. In both

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

environments, we utilised FEniCS (ver. 2019.2.0.64), NumPy (ver. 1.23.5), and PETSc (ver. 3.15.5 on CPU and ver. 3.23.2 on GPU; Balay et al. 2019). Although early development of PETSc focused on MPI-based CPU parallel computing, GPU-based vector operations (VecCUDA) were implemented in PETSc 3.12 (2019). Furthermore, PETSc 3.15 (2021) added the GPU-native sparse matrix type AIJCUSPARSE, enabling complete GPU-based matrix-vector multiplication, a fundamental operation in iterative linear solvers.

GPU acceleration is extendable to full Krylov iteration cycles, covering solvers such as CG and GMRES. Within the finite element framework of FEniCS, the PETSc backend provides access to the same Krylov subspace solvers (CG, GMRES, MINRES, and FGMRES) and a broad suite of preconditioners, ranging from incomplete LU to geometric or algebraic multigrid, in both GPU and CPU environments. A previous study (Zheng et al., 2013) reported up to a 4.8-fold speedup by accelerating a finite difference method (FDM) discretized Stokes problem on a single-GPU (NVIDIA Tesla C2070) using a customised GMG solver implemented in MATLAB and CUDA, compared to an Intel i7 workstation (4 cores and 8 threads) running MATLAB with C extensions. Nevertheless, attempts to integrate GPU acceleration into well-established FEM software for high-performance geodynamic simulations remain limited.

In the pressure correction step of the iterative algorithm, an L^2-projection using the mass matrix repeatedly projects the divergence of the velocity field onto the P_1 space. With FEniCS GPU acceleration, matrix assembly is performed on CPU cores tied to MPI ranks mapped to GPUs. Under FEniCS GPU acceleration, sparse matrix assembly runs on the CPU using 1 to 16 cores. A matrix assembled on a single CPU core is solved on 1 GPU. Matrices assembled across multiple CPU cores are solved on 1 or 2 GPUs via NVIDIA Multi-Process Service (MPS). Reassembling the mass matrix at each iteration causes repeated GPU-CPU transfers and synchronization, increasing overhead and overall runtime. To minimise these bottlenecks in the GPU implementation, we adopted a pre-assembly strategy in which the P_1 mass matrix, together with differential operators (gradient and divergence), is assembled only once, then kept resident in GPU memory for reuse throughout subsequent iterations. The memory required for the pre-assembled matrices was 510 MB for 3.3×10^6 DOFs and 1.3 GB for 8.8×10^6 DOFs (Table S3). For a fair comparison between CPU and GPU runs, the same pre-assembly strategy using system memory was also applied to the CPU implementation.

3.2 The BoomerAMG preconditioner in HYPRE

We employed the GMRES Krylov subspace solver to solve the system $Kh^k = Gd^k$ (Equation 9) within the pressure Schur complement iteration. To accelerate GMRES convergence, the BoomerAMG algebraic multigrid (AMG) preconditioner from the HYPRE library was applied, interfaced through PETSc (Falgout et al., 2006). In different computing environments (i.e., CPU and GPU platforms), optimal convergence efficiency and computational performance depend on careful tuning of the BoomerAMG preconditioner hyperparameters, including coarsening schemes (e.g., Falgout, Parallel Modified Independent Set (PMIS), and Hybrid Modified Independent Set (HMIS)), smoothers (e.g., Chebyshev, Gauss-Seidel, Jacobi, symmetric SOR, and symmetric Jacobi), and interpolation options (e.g., L-shaped, Extended, and Direct). We optimised the BoomerAMG preconditioner settings by considering architectural differences between a 32-core CPU and a GPU with thousands of parallel processing cores (NVIDIA RTX 6000 Ada, 18,176 CUDA cores).

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

In the GPU environment, we found configurations that exploit massive parallelism while minimising communication and data dependencies. For coarsening, PMIS and HMIS reduced the computational time for coarse-grid construction (De Sterck et al., 2006; Alber and Olson, 2007). To enhance parallel implementation, we adopted the scaled Jacobi method and Extended+i for smoothing and interpolation, respectively (Offermans et al., 2020; Park et al., 2015). A V-cycle served as the multigrid schedule. In multi-core CPU environments, we configured the solver for sequential algorithms that deliver strong per-iteration error reduction. We used the Falgout algorithm as the default coarsening option, since major linear algebra libraries such as HYPRE, PETSc, and Trilinos confirm its stability as a standard approach (e.g., Falgout et al. 2006; Yang 2002). To ensure fairness relative to GPU runs, the HMIS algorithm was tested as an additional coarsening option. Smoothing employed symmetric SOR and Jacobi smoothers, which provide substantial error reduction within a single iteration (Brandt, 1977; MacLachlan and Oosterlee, 2011). Classical interpolation, highly compatible with Falgout coarsening, served as the interpolation scheme (Brannick et al., 2018). As in the GPU configuration, a V-cycle was used for the multigrid schedule.

We evaluated the accuracy, execution time, and VRAM usage of the preconditioned conjugate-gradient solver on the GPU using manufactured solutions, the SolCx viscosity-contrast benchmark (Zhong, 1996; Thieulot and Bangerth, 2022), the sticky-air free-surface benchmark (Crameri et al., 2012; Hillebrand et al., 2014), 2D and 3D Rayleigh–Taylor instability tests (van Keken et al., 1997; Maljaars et al., 2021), and the thermal convection benchmark of Blankenbach et al. (1989). In 3D runs, stable execution was achieved on a $100 \times 40 \times 100$ grid under $P_1^+ \times P_1$ element discretization on a single 48 GB GPU, despite the limited VRAM capacity of the GPU. Memory distribution efficiency in a two-GPUs system was assessed for memory-demanding cases, including high-resolution 3D runs approaching $100 \times 90 \times 100$ using the $P_1^+ \times P_1$.

4 Numerical results

To clarify how GPU speedup is measured, we distinguish between execution time and wall time. Execution time refers to the time spent in the pressure-correction stage during each iteration, including GMRES iterations, vector updates, and associated CPU and GPU overheads. In contrast, wall time represents the end-to-end runtime for time-dependent viscosity cases (Sections 4.5, 4.6, and 4.7), encompassing velocity subsystem matrix assembly (on CPUs) at each time step and mesh generation. By explicitly distinguishing between execution time and wall time, we can independently assess the GPU performance of the PCG solver itself and the overall computational cost when matrix assembly is included. Each PCG iteration (Equations 7–12) updates the pressure through one cycle of the preconditioned conjugate-gradient method. Within each PCG iteration, the velocity subproblem (Equation 9) is solved by BoomerAMG-preconditioned GMRES (hereafter referred to as the inner GMRES solve).

4.1 Velocity subproblem on GPUs

In the PCG solver for solving Stokes flow, the primary computational cost and memory bottleneck stem from solving the velocity-only submatrix system (Equation 9). Therefore, to assess the GPU acceleration of the entire PCG solver, we first evaluated the performance of the GPU-based solver specifically for this submatrix system. Computational efficiency between CPU

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

and GPU environments was compared using inner GMRES with the BoomerAMG preconditioner. Accuracy and convergence were verified through a manufactured solution prescribed on $\Omega = [0,1]^2$, with the corresponding source term derived from the prescribed solution.

$$\nabla \cdot \boldsymbol{\tau} = \mathbf{f} \quad \text{where} \quad \boldsymbol{\tau} = 2\eta\dot{\boldsymbol{\varepsilon}} \quad \text{and} \quad \dot{\boldsymbol{\varepsilon}} = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right) \tag{13}$$

$$\eta = \exp(cx), \quad c = \ln(10^6) \tag{14}$$

$\mathbf{u}$, η, and $\mathbf{f}$ denote the velocity vector, viscosity, and prescribed source term, respectively. Viscosity increases exponentially along the x-direction (Equation 14), yielding a viscosity contrast of 10^6. The exact solution $\mathbf{u}_{\text{exact}}$ (Figure S1) was prescribed as

$$\mathbf{u}_{\text{exact}} = \begin{pmatrix} \sin(\pi x)\sin(\pi y) \\ \cos(\pi x)\cos(\pi y) \end{pmatrix}, \quad 0 \le x \le 1,\ 0 \le y \le 1 \tag{15}$$

The term $\mathbf{f}$ is derived by substituting the manufactured solution and viscosity into (Equation 13).

$$\mathbf{f} = \begin{pmatrix} 2\pi\exp(cx)\sin(\pi y)\left(c\cos(\pi x) - \pi\sin(\pi x)\right) \\ -2\pi^2\exp(cx)\cos(\pi x)\cos(\pi y) \end{pmatrix} \tag{16}$$

Dirichlet boundary conditions were imposed on $\partial\Omega$ as $\mathbf{u} = \mathbf{u}_{\text{exact}}$.

Performance of the AMG-preconditioned GMRES solver was evaluated for single-GPU and multi-core CPU implementations. The velocity fields obtained in the GPU and CPU runs were identical, with the L^2-error relative to the exact solution remaining on the order of 10^{-10}. Execution time and the GMRES iteration number were monitored. Comparison across various coarsening strategies showed architecture-dependent optimal choices (GPU versus CPU). In the GPU environment (Figure 1a), HMIS (red line) converged in 17 iterations within 0.413 s, whereas PMIS (grey line) required 20 iterations within 0.452 s (inset in Figure 1a). In the CPU environment, PMIS (green line), HMIS (orange line), and Falgout (blue line) required 60 (21.92 s), 25 (9.76 s), and 13 (5.33 s) GMRES iterations, respectively, to reach convergence (Figure 1a). Performance comparisons used the fastest setup on each architecture, with HMIS on GPU and Falgout on CPU. The GPU HMIS setup converged in four more GMRES iterations than the CPU Falgout setup (Figure 1a), while achieving a $13\times$ reduction in the accumulated execution time (Figure 1b). In execution time comparisons between single-GPU and multi-core-CPU runs, the single-GPU HMIS setup ran faster than all CPU Falgout setups, except for the lowest resolution case (33,282 DOFs) (Figure 1c). For the largest DOF case (8,396,802 DOFs), the single-GPU required 0.413 s, representing an approximately $13\times$ reduction relative to the 32-core CPU time of 5.331 s.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

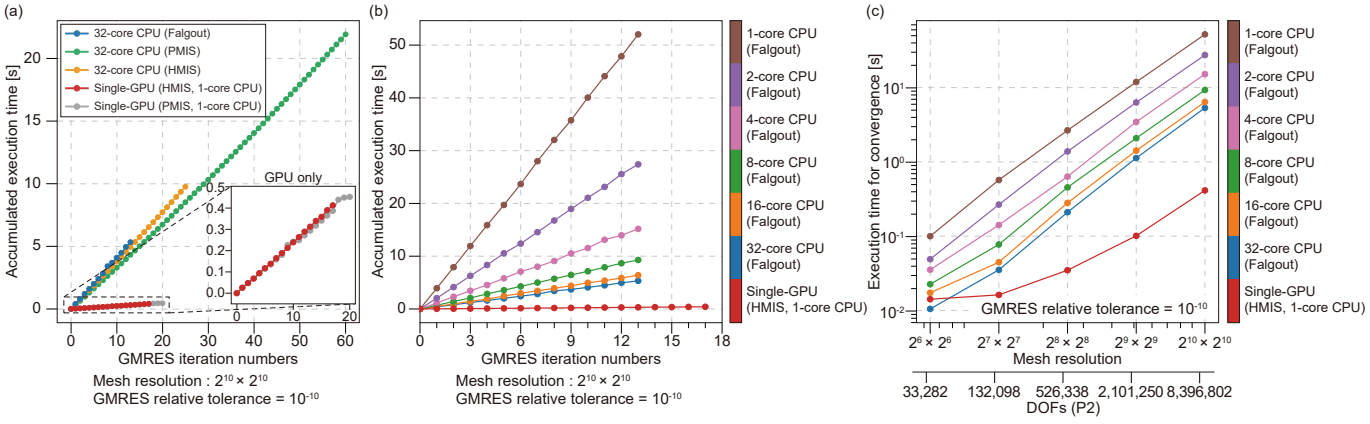

Figure 1. Inner GMRES solver execution time analysis for the velocity subproblem in single-GPU (HMIS) with 1-core CPU and multi-core CPU (Falgout) environments. (a) Accumulated execution times for the GPU and 32-core CPU using different AMG preconditioner coarsening schemes (Falgout, PMIS, and HMIS). (b) Accumulated execution time on the GPU (HMIS) and CPU (Falgout) with the number of cores ranging from 1 to 32. (c) Total execution time to convergence with respect to increasing mesh resolution for the GPU (HMIS) and CPU (Falgout).

4.2 SolCx Problem

We evaluated whether accelerating solving the velocity-only subproblem on single-GPU reduced the overall execution time of the PCG solver. We used the 2D SolCx benchmark, which has been widely employed to assess the stability and convergence of Stokes solvers with strong viscosity discontinuities (Zhong, 1996; Duretz et al., 2011; Kronbichler et al., 2012; Gerya et al., 2013). The computational domain is the unit square $\Omega = [0,1] \times [0,1]$ with free-slip boundary conditions on all four sides. The density field was prescribed to vary smoothly. In contrast, the viscosity exhibited a sharp jump at $x = 0.5$ (Equation 18). We set Ra to 0 and Rb to a spatially varying field to generate buoyancy (Equation 17). A single composition field was adopted with $C_0 = 1$.

$$Rb(x,y) = \sin(\pi y)\cos(\pi x), \quad 0 \le x \le 1, \quad 0 \le y \le 1 \tag{17}$$

$$\eta(x) = \begin{cases} 1, & 0 \le x \le 0.5, \\ 10^6, & 0.5 < x \le 1. \end{cases} \tag{18}$$

We compared the performance of the GPU-accelerated finite-element model with that of multicore CPU environments. The GMRES solver employed a default relative tolerance of 10^{-7}. Performance was tested with varying GMRES relative tolerance and mesh resolution, based on the execution time of the PCG solver and velocity error relative to the analytical solution. The

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

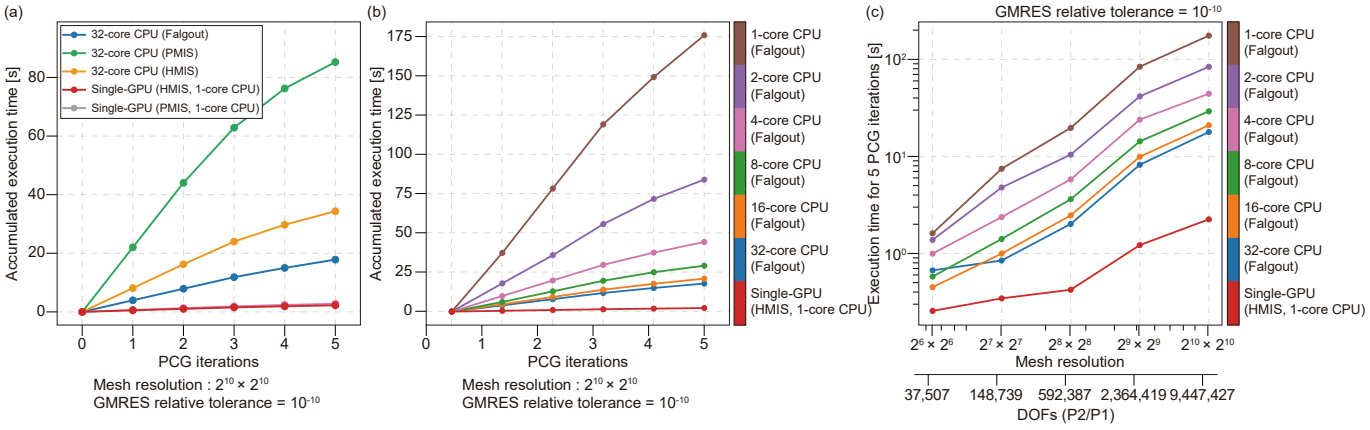

Figure 2. Execution time analysis of the SolCx benchmark in GPU and CPU environments. (a) Accumulated execution times on the GPU and 32-core CPU under different AMG preconditioner coarsening schemes (Falgout, PMIS, and HMIS). (b) Accumulated execution time on single-GPU (HMIS) with 1-core CPU and CPU (Falgout) with the number of cores ranging from 1 to 32. (c) Execution time for five preconditioned conjugate-gradient (PCG) iterations with respect to increasing mesh resolution for the GPU (HMIS) and CPU (Falgout).

number of PCG iterations was fixed at five. After five PCG iterations, the L^2-error relative to the analytical solution was smaller than 10^{-9} for both GPU and multi-core CPU environments.

We found that the performance in solving the velocity-only subproblem was directly translated to the PCG solver performance (Figure 2a). HMIS on the GPU and Falgout on the CPU were the most effective choices. In all cases, PCG solver execution time on the single-GPU was shorter than that in all multi-core CPU environments. The accumulated execution time on the GPU was approximately 11 times shorter than that on the 32-core CPU, which demonstrates the high efficiency of the GPU architecture. Based on the comparison in Figure 2a, the HMIS and Falgout coarseners delivered the best performance on a single-GPU and a 32-core CPU, respectively. Thus, the HMIS and Falgout coarseners were used for single-GPU and multi-core CPU performance comparisons in Figures 2b and 2c. Then, we compared accumulated execution times for five PCG iterations on a single-GPU and on CPUs with the number of cores increasing from 1 to 32 (Figure 2b). The single-GPU recorded the shortest execution time at 1.9 seconds. For CPUs, execution time decreased as the number of cores increases. With a 1-core CPU, execution time was 175 seconds, while a 32-core CPU recorded 14.8 seconds.

The performance evaluation with increasing problem sizes exhibited a similar trend (Figure 2c). As the resolution increased, the gap between the GPU and CPU execution times widened. For all resolutions, the single-GPU delivered shorter execution times than any multi-core CPU environment. This advantage of the GPU was more pronounced for larger problems (Figure 2c). For the highest resolution (1024×1024, 9,447,427 DOFs), the GPU required 2.25 s, while the 32-core CPU required 17.82 s, corresponding to a speedup of approximately $7.9\times$. PCG iterations and GMRES stages are dominated by sparse matrix–vector products. Such operations exhibit high data parallelism, which leads to particularly efficient execution on a GPU equipped with large memory bandwidth.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

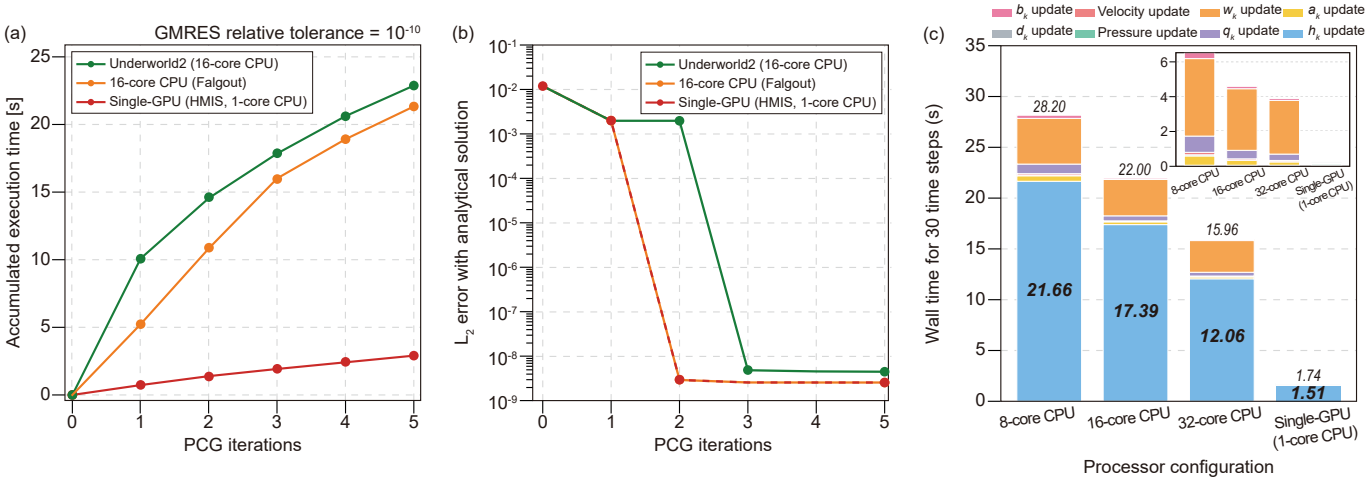

Figure 3. SolCx benchmark. Performance benchmark and detailed execution time analysis of the preconditioned conjugate-gradient (PCG) iterations on GPU architectures. (a) Comparison of accumulated execution time as a function of the number of PCG solver for single-GPU (HMIS) with 1-core CPU, Underworld2 (16 cores), and 16-core CPU (Falgout) environments. (b) Comparison of the L^2-error with respect to the analytical solution as a function of the number of PCG solver for single-GPU (HMIS) with 1-core CPU, Underworld2 (16 cores), and 16-core CPU (Falgout) environments. (c) Breakdown of execution time over five PCG iterations for the GPU and CPUs with 8, 16, and 32 cores.

Figure 3a presents the accumulated execution times for the PCG solver on a single-GPU (red line) and a 16-core CPU environment (orange line). Both environments fixed the number of PCG iterations at five. For comparison, Underworld2 was evaluated on a 16-core CPU (green line) using its default settings. Underworld2 employed Q_1/dQ_0 (quadrilateral) elements. Five pressure-correction iterations were used to solve the Schur-complement system. The velocity-only subsystem was solved using GMG-preconditioned GMRES. The GPU implementation of the PCG solver achieved a cumulative runtime of 2.9 s over five iterations, which is approximately 10 times faster than both Underworld2 and the CPU-based PCG solver. For five iterations, Underworld2 and the PCG solver on the CPU required 30.5 s and 28.07 s, respectively. On the 16-core CPU, our PCG solver implementation achieved a runtime comparable to that of Underworld2. We measured the relative L^2-error against the analytical solution with respect to PCG iteration number (Figure 3b). The PCG solver and the Schur-complement solver reduced the error below 10^{-8} at the second and third iterations, respectively. The final converged error from the PCG solver was lower than that of Underworld2. This difference is attributed to the larger number of DOFs associated with the P_0 element used in FEniCS (2,097,152 DOFs) compared to the dQ_0 element used in Underworld2 (1,048,576 DOFs).

We measured execution times for individual stages of the PCG solver algorithm (assembly, solving velocity-only subsystem, vector update, and projection) to quantify differences between the 8-, 16-, and 32-core CPU runs and the GPU run (Figure 3c). The single-GPU achieved speedups exceeding $14\times$ and $9\times$ relative to the 8-core and 32-core CPUs, respectively. Overall, the velocity-only subsystem stage dominated the total cost. The GPU also demonstrated the shortest runtimes in the assembly and

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

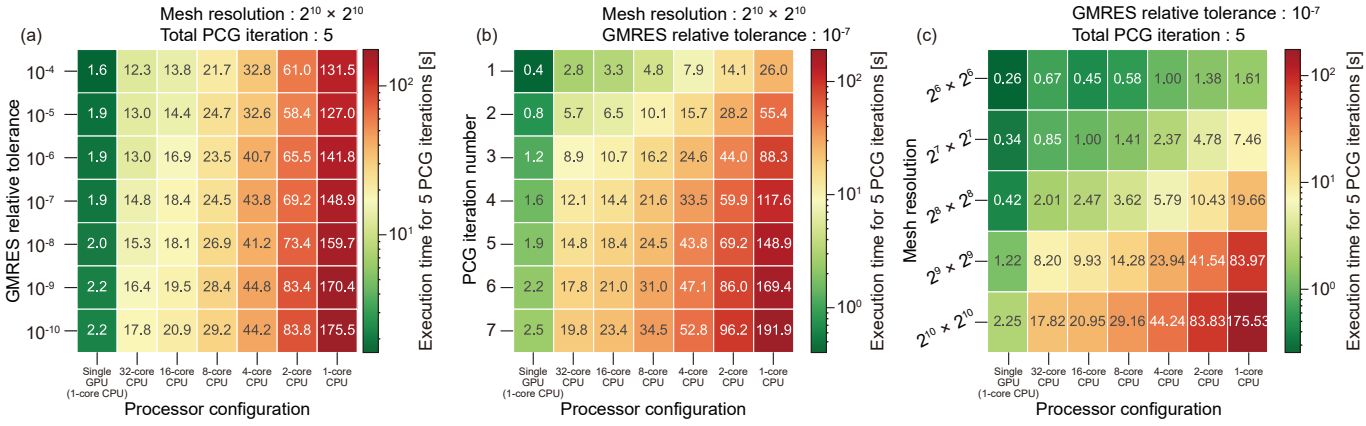

Figure 4. Heatmap analysis of execution times across single-GPU with 1-core CPU and 1-32 core CPU configurations. (a) Execution time for five preconditioned conjugate-gradient (PCG) iterations with varying GMRES relative tolerances (from 10^{-4} to 10^{-10}). (b) Accumulated execution time with increasing PCG iteration numbers (1 to 7) at a fixed mesh resolution of $2^{10} \times 2^{10}$. (c) Execution time for 5 PCG iterations with respect to increasing mesh resolution from $2^6 \times 2^6$ to $2^{10} \times 2^{10}$. The colour bars on the right represent execution time in seconds on a logarithmic scale.

vector update stages. As the vector-update and projection stages rely on memory-bandwidth-limited kernels, the GPU's high memory bandwidth resulted in markedly lower per-iteration execution times.

Figure 4a presents execution times obtained by varying the GMRES relative tolerance from 10^{-4} to 10^{-10}, on a $2^{10} \times 2^{10}$ grid with the number of PCG iterations fixed at five. Smaller tolerances increased runtime in all environments. The GPU maintained the shortest runtime over the entire tolerance range. Figure 4b presents execution times on the same grid, with the GMRES relative tolerance fixed at 10^{-10} and varying numbers of PCG iterations. Execution time increased almost linearly with the iteration number. The GPU achieved the shortest execution time. Figure 4c presents the performance for grid resolutions from 2^6 to 2^{10}. Runtime increased monotonically with resolution. The GPU exhibited a smaller rate of increase than all multi-core CPU configurations. The GPU's performance advantage over CPUs increased as problem size increased.

4.3 3D Thermo-mechanical convection

We extended the classical 2D thermal convection benchmark of Blankenbach et al. (1989) to a 3D non-rotating Boussinesq fluid and adopted it for verification. The thermal and compositional Rayleigh numbers were set to $Ra = 10^4$ and $Rb = 0$ in Equation 1, isolating thermal effects by neglecting compositional buoyancy. A 3D rectangular domain ($1 \times 0.4 \times 1$) was employed to quantitatively compare computational performance and velocity-field accuracy between the GPU and CPU environments. The grid resolution was set to $100 \times 40 \times 100$ using the MINI element ($P_1^+ \times P_1$), yielding approximately 8.8 million DOFs for velocity and pressure. Free-slip boundary conditions were applied to all boundaries. For the temperature boundary conditions, nondimensional temperatures of $T = 0$ and $T = 1$ were prescribed at the top and bottom boundaries, respectively. The initial temperature field T_0 consisted of a linear conductive profile with a superimposed long-wavelength perturbation of amplitude

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

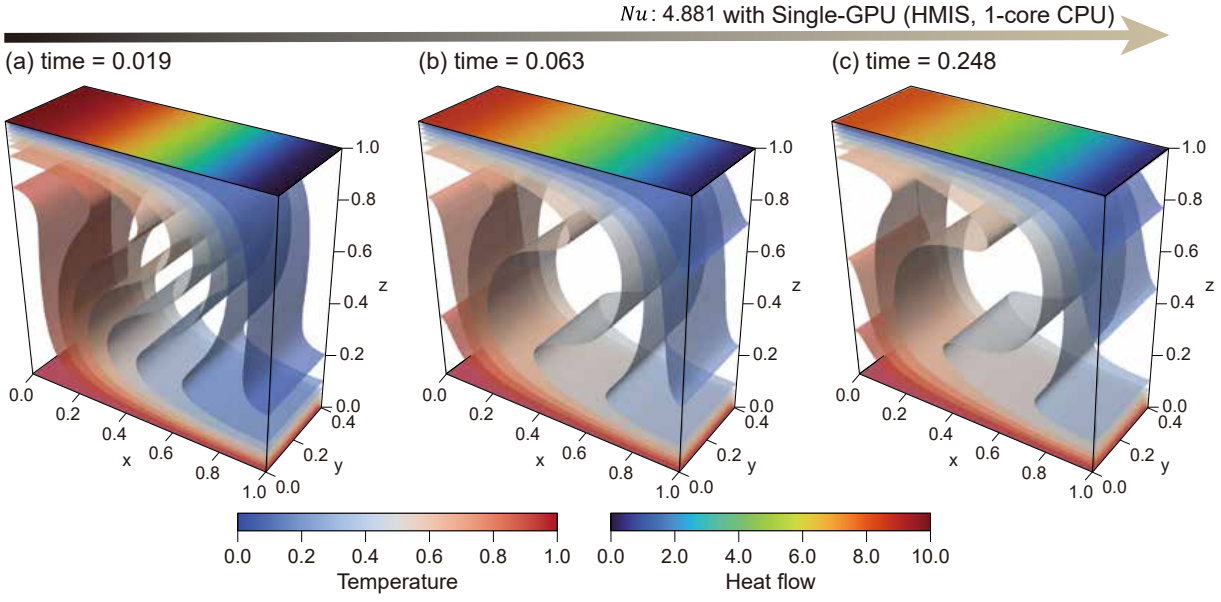

Figure 5. Time evolution of the temperature field from the single-GPU (HMIS) with 1-core CPU. Snapshots of 3D temperature isosurfaces at nondimensional times $t = 0.019$, 0.063, and 0.248 (from left to right).

$\alpha = 0.02$ to initiate natural convection (Equation 19). As time proceeds, thermal plumes rise from the lower boundary, which leads to a progressive reorganization of the internal temperature field and the heat-flow pattern at the boundaries (Figure 5).

$$T_0 = 1 - z + \alpha \cos(\pi x) \sin(\pi z) \tag{19}$$

To evaluate the model accuracy, we measured V_{rms} and the Nusselt number (Nu), comparing the computed values to benchmarks from previous studies (Blankenbach et al., 1989; Tackley and King, 2003; King, 2009). For the 3D thermo-mechanical convection benchmark, we utilised a fixed time step ($\Delta t = 0.001$) with the Crank-Nicolson scheme for time integration. We set the GMRES relative tolerance to 10^{-7} for the velocity-only subsystem on both GPU and CPU, continuing the PCG iterations until the divergence error ($\|\nabla \cdot \mathbf{u}\|/\|\mathbf{u}\|$) converged to divergence tolerance of 10^{-6}. The energy equation (Equation 4) was solved using GMRES with a relative tolerance of 10^{-8}, utilising BoomerAMG as the preconditioner, configured with HMIS for the GPU and Falgout for the CPU, respectively.

For the Stokes equations, the GPU-HMIS achieved the lowest execution time with an average of 1.9 s (Figure 6a; solid red line), outperforming the CPU-Falgout at 28.7 s (8 cores; solid green line), 20.4 s (16 cores; solid orange line), and 17.1 s (32 cores; solid blue line). The speedups against the 8-core and 32-core CPU environments were approximately $15\times$ and $9\times$, respectively. The average number of GMRES iterations per time step was 52 on the GPU and 23 across all CPU environments (Figure 6a, coloured bars). The GPU exhibited shorter execution time per time step despite the iteration number being $2.3\times$ higher than the CPU baseline. Memory usage measurements for the GPU environment included both VRAM and system

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

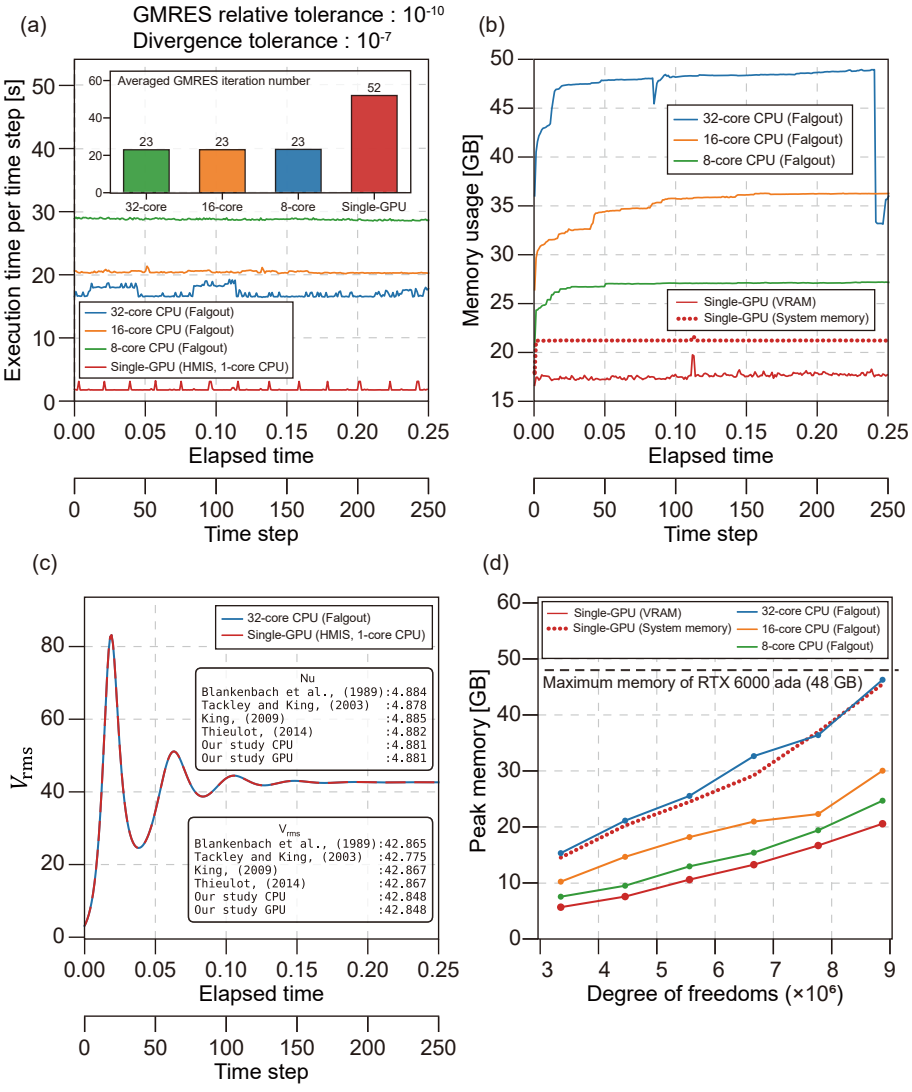

Figure 6. Performance analysis and validation of the 3D thermo-mechanical convection simulation. (a) Execution time per step (solid lines, left axis) and averaged GMRES iteration numbers (inset bar chart) versus elapsed time. (b) Evolution of memory usage over elapsed time, showing GPU VRAM (dashed purple line), GPU system memory (solid red line), and CPU system memory for 8, 16, and 32 cores (solid green, orange, and blue lines, respectively). (c) Time evolution of the root-mean-square velocity (V_{rms}) for the single-GPU (HMIS) with 1-core CPU and 32-core CPU (Falgout), including a table comparing final Nusselt numbers and V_{rms} values with literature benchmarks. (d) Peak memory usage with respect to increasing degrees of freedom (DOFs), with the dashed red line indicating the 48 GB VRAM capacity of the NVIDIA RTX 6000 Ada.

memory (RAM) of the workstation (Figure 6b). The average VRAM and system memory usage per time step were 21.2 GB (dotted red line) and 17.3 GB (solid red line) respectively, for a total of 38.5 GB. In the CPU environment, the average memory

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

usage recorded 26.7 GB, 35.0 GB, and 47.3 GB for 8, 16, and 32 cores, respectively, indicating that the memory usage of the 16-core CPU was comparable to the single-GPU setup.

The temporal evolution of the global root-mean-square velocity (V_{rms}) exhibited complete consistency between the GPU and all CPU environments. The measurements converged to 42.848, consistent with the value of 42.865 reported by Blankenbach et al. (1989). A Nusselt number (Nu) of 4.881 was calculated at the final time step in the GPU and all CPU environments, matching the value of 4.884 presented in benchmark studies (Blankenbach et al., 1989; Tackley and King, 2003; King, 2009). To assess memory constraints associated with the multi-level generation of the AMG preconditioner, we measured the maximum memory usage for each computing environment with increasing mesh resolution (Figure 6d). Measurements for the GPU environment included both VRAM and system memory. At the lowest DOFs (2,226,564), VRAM and system memory usage were 14.5 GB and 5.7 GB, respectively. At the highest DOFs (8,872,964), VRAM usage reached 45.5 GB, while system memory usage reached 20.6 GB. The combined total was 66.1 GB. In the CPU environment, memory consumption increased with the number of cores, peaking at the 32-core setup. The 32-core CPU registered 15.3 GB at the lowest DOFs and 46.3 GB at the highest exhibiting levels comparable to the GPU VRAM usage.

4.4 2D Rayleigh–Taylor instability

The Rayleigh–Taylor instability is a buoyancy-driven flow widely used as a benchmark for evaluating the performance of various advection schemes. Representative approaches include Particle-In-Cell (PIC) methods (e.g., Gerya and Yuen 2003; van Keken et al. 1997) and level-set methods (e.g., Bourgouin et al. 2006). We set $Ra = 0$ and $Rb = 1$, indicating that only compositional buoyancy is activated. The initial interface between the two compositions is defined by a cosine function with an amplitude of 0.02 (Equation 20). The level-set function was initialised from the interface at $t = 0$. The composition field was set to 1 and 0 where the level-set function was positive and negative (Equation 22), respectively. The level-set function was reinitialised every 500 steps.

$$h(x) = 0.02\cos(\pi x/L) + 0.2 \tag{20}$$

$$\phi(x, y, t = 0) = y - h(x) \tag{21}$$

$$C(x, y, t = 0) = \begin{cases} 0, & \phi(x, y, t = 0) \leq 0, \\ 1, & \phi(x, y, t = 0) > 0. \end{cases} \tag{22}$$

For resolving complex velocity and composition structures, we adopted uniform grids with 512×512 resolution. The Taylor-Hood element ($P_2 \times P_1$) was applied to solve the Stokes equations. We compared the V_{rms} values computed by our FEniCS code with those from previous benchmark models (van Keken et al., 1997; Hillebrand et al., 2014). The advection equation

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

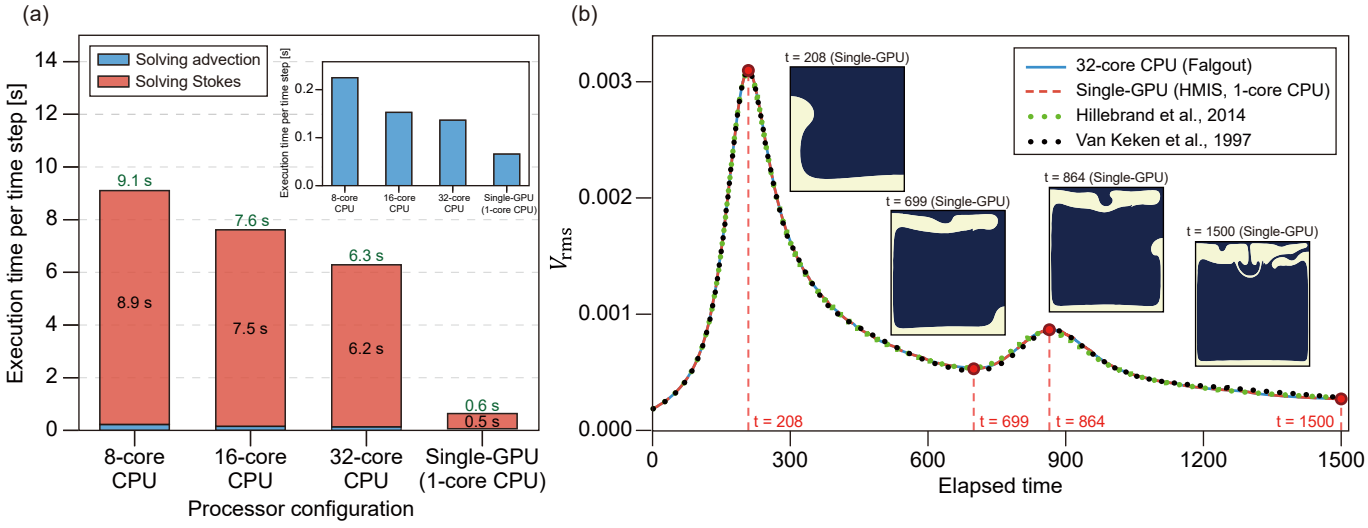

Figure 7. Performance analysis and accuracy validation of single-GPU (HMIS) with 1-core CPU and multi-core CPU (Falgout) configurations for the 2D Rayleigh–Taylor instability benchmark. (a) Execution time per step for solving the Stokes equations (red bars) and the advection equation (blue bars) across GPU and 8 to 32 core CPU configurations. (b) Time evolution of the root-mean-square velocity (V_{rms}) compared with literature benchmarks (van Keken et al., 1997; Hillebrand et al., 2014). The inset snapshots represent the composition fields computed on the GPU at key time steps ($t = 208$, 699, 864, and 1500).

for the level-set function was solved using GMRES preconditioned by HYPRE's BoomerAMG. The average per-step runtime was compared across GPU and CPU environments in the 2D benchmark (Figure 7a). The GPU completed one step in 0.6 s on average (0.56 s for the Stokes equations and 0.04 s for advection), yielding speedups of $\sim 11\times$ relative to the 32-core CPU and $\sim 15\times$ relative to the 8-core CPU. For the CPU environments, increasing the number of cores from 8 to 32 reduced the per-step time from 9.1 s to 6.3 s. Most of the per-step cost was dominated by solving the Stokes equations (8.9 s and 6.2 s for the 8-core and 32-core CPUs, respectively). Despite the substantial improvement in computational efficiency enabled by GPU acceleration, the solution accuracy remains comparable to that of the CPU implementation. This agreement is demonstrated by the time evolution of V_{rms}, which closely matches the CPU simulation (Figure 7b). Snapshot comparisons at key times ($t = 209$, 700, and 864), when V_{rms} attains local maxima or minima, further support the model accuracy. In addition, the GPU-based model reproduces the reference V_{rms} values from Hillebrand et al. (2014) and van Keken et al. (1997).

4.5 3D Rayleigh–Taylor instability

We extended the 2D Rayleigh–Taylor instability model to 3D. The initial composition field was prescribed by imposing a small perturbation on the interface between two fluids with different viscosities and densities. The initial composition (Equation 24) was defined using a level-set function (Equation 23). Ra and Rb were set to 0 and 1, respectively.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

$$\phi(\mathbf{x}, t=0) = z - \left(0.2 + 0.02\cos\left(\frac{\pi x}{L_x}\right)\cos\left(\frac{\pi y}{L_y}\right)\right) \tag{23}$$

$$C(\mathbf{x}, t=0) = \begin{cases} 0, & \phi(\mathbf{x}, t=0) \leq 0, \\ 1, & \phi(\mathbf{x}, t=0) > 0. \end{cases} \tag{24}$$

$L_x = 0.9142$ and $L_y = 0.8142$ denote the domain lengths in the x and y directions, respectively, while the domain length in the z direction was set to 1. The viscosity was prescribed as $\eta = 0.01$ and 1 for $C = 0$ and 1, respectively. To adequately resolve the complex 3D velocity and compositional structures, we used a $65 \times 65 \times 65$ mesh with the MINI element (i.e., $P_1^+ \times P_1$). The advection equation for the level-set function was solved using GMRES with HYPRE BoomerAMG preconditioner. The model evolution shows that the initially imposed cosine-shaped perturbation initiates plume-like upwelling from regions where the interface is locally elevated (Figures 8e–h). By $t = 60$, the plume rises to approximately $z = 0.8$ (Figure 8c). We quantified the normal stress in z direction ($\sigma_{zz} = 2\eta\frac{dw}{dz}$), which controls the surface topography, where η and w denote viscosity and vertical velocity, respectively. Positive dynamic topography developed above the plume upwelling, reaching a maximum value of 6.6. The time evolution of the maximum and minimum dynamic topography values was identical for the GPU and CPU simulations, demonstrating that the GPU implementation also preserves accuracy in the pseudo free-surface calculation. We employed V_{rms} as a metric to assess consistency between GPU and CPU environments (Figure 8i). V_{rms} increases as the plume ascends, peaks near $t \approx 60$, and then decreases. Despite differences in the number of GMRES iterations, the V_{rms} values obtained on the CPU and GPU were identical across all environments.

As time elapses, the density distribution becomes more complex, which can increase the execution time of the Stokes solve. The number of GMRES iterations increases as elapsed time progresses, and the GPU environment requires $\sim 2.5\times$ more GMRES iterations than the CPU environment at all elapsed times (Figure 9b). Nevertheless, the GPU execution time increases more gradually with elapsed time than the CPU execution time. The GPU (HMIS) was approximately $9\times$ faster than the 32-core CPU (Falgout) across the entire time range (Figure 9a). Linear regression of execution time against elapsed time produced slopes of 1.30 for the CPU (Falgout) and 0.27 for the GPU (HMIS). To assess the efficiency of stiffness matrix assembly on the CPU, GMRES iterations on the GPU, and GPU-CPU data transfer, we compared the total simulation wall time across multiple CPU-GPU configurations (Figure 9c). The 32-core CPU-only simulation required 2760 s. The optimal configuration, two-GPU with 16-core CPU, achieved the shortest wall time of 840.1 s, corresponding to a $3.3\times$ speedup over the 32-core CPU-only execution. Using a single-GPU with 16-core CPU increased the wall time to 1016.7 s due to reduced GPU resources. With a single-GPU and single CPU core, wall time further increased to 2415.7 s, as CPU assembly became the dominant bottleneck. The single-GPU with 1-core CPU delivered only a $1.14\times$ speedup relative to the 32-core CPU.

4.6 Sticky air benchmarks

In geodynamic numerical modelling, the sticky-air technique is a widely used approach for approximating a free surface. Crameri et al. (2012) suggested that to correctly represent a free-surface boundary condition, the sticky-air layer requires a

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

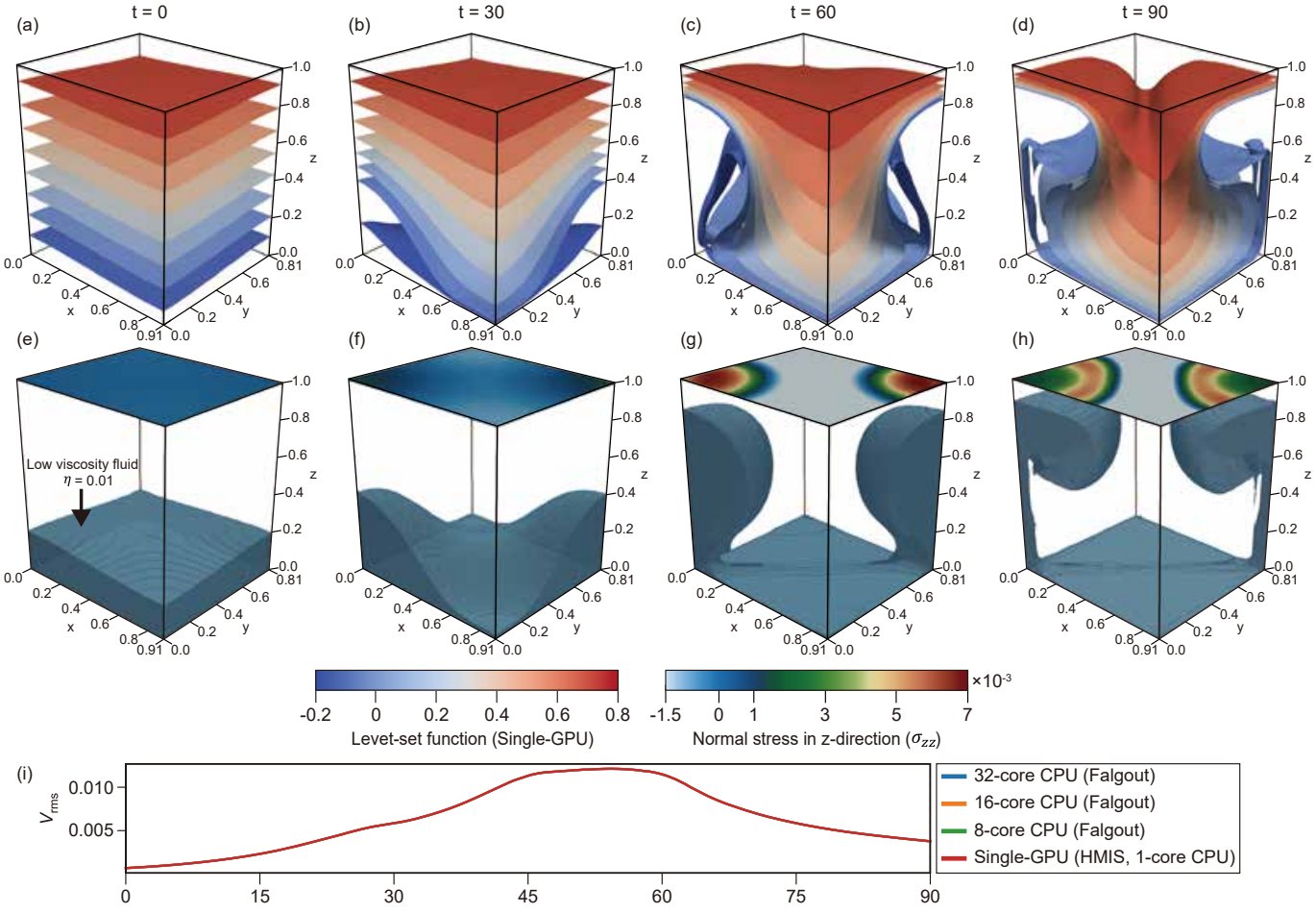

Figure 8. Temporal evolution of the 3D Rayleigh-Taylor instability derived from the GPU (HMIS) with 1-core CPU. (a–d) Snapshots of level-set function isosurfaces at nondimensional times $t = 0$, 30, 60, and 90. (e–h) Evolution of the interface between two compositions and the distribution of normal stress in the z-direction on the top surface. (i) Root-mean-square velocity (V_{rms}) measured in the GPU, 8-core, 16-core, and 32-core CPU environments.

viscosity contrast of 10^{-5} relative to the underlying material and a thickness of 100 km, or alternatively a viscosity contrast of 10^{-4} and a thickness of 200 km. In our study, we set the viscosities of the lithospheric, mantle, and sticky-air layers (with a thickness of 100 km) to $\eta_{\text{litho}} = 10^{23}$ Pa·s, $\eta_{\text{mantle}} = 10^{22}$ Pa·s, and $\eta_{\text{air}} = 10^{18}$ Pa·s, respectively. The nondimensional parameters for the sticky-air benchmark are presented in Table S1. We discretized the Stokes system using $P_2 \times P_1$ finite elements on a 1400×800 mesh. The boundary between the lithosphere and the air was assigned a cosine-shaped perturbation. We compared the maximum topography with the analytical solution from Crameri et al. (2012). The initial composition field (C) is defined as three layers separated by the upper and lower interfaces of the lithosphere (Equation 27). Two level-set functions (ϕ_1 and ϕ_2) were defined as signed distance functions for each interface (Equations 25 and 26).

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

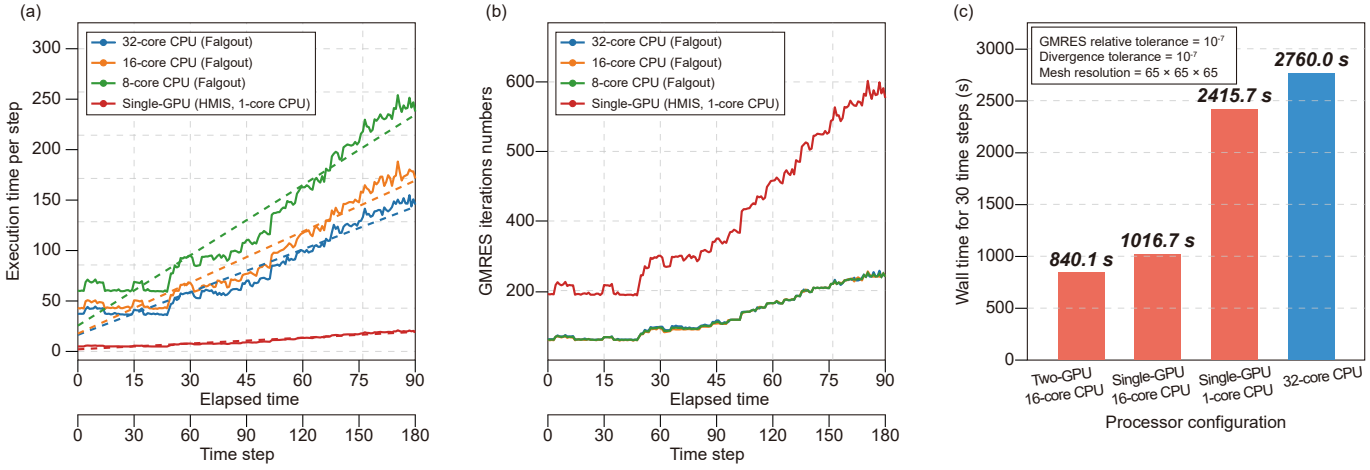

Figure 9. Performance analysis and validation of the 3D Rayleigh-Taylor instability simulation. (a) Execution time per step versus elapsed time for single-GPU (HMIS) with 1-core CPU (solid red line) and CPU (Falgout) with 8, 16, and 32 cores (green, orange, and blue solid lines, respectively). The dashed lines indicate a linearly increasing trend in execution time. (b) Evolution of GMRES iteration numbers per step over elapsed time for the GPU (HMIS) and 8–32 core CPUs (Falgout). (c) Wall time for 30 time steps under four processor configurations: two-GPU with 16-core CPU, single-GPU with 16-core CPU, single-GPU with 1-core CPU, and 32-core CPU (CPU-only).

$$\phi_1(x,y,t=0) = 700 + 7\cos\left(\frac{2\pi x}{2800}\right) - y \tag{25}$$

$$\phi_2(x,y,t=0) = y - 600 \tag{26}$$

$$C = \begin{cases} C_1 & \phi_1(x,y,t=0) \le 0 \\ C_3 & \phi_2(x,y,t=0) < 0 \\ C_2 & \text{else} \end{cases} \tag{27}$$

An initial perturbation imposed on the lithosphere generates a lateral density gradient, which drives upwelling in the centre of the domain and downwelling on both sides, forming two convection cells (Figure 10a). Advective transport progressively flattens the initial topographic perturbation. The surface recovers an almost flat geometry by the final time of 100 kyr (Figure 10c). In contrast, the divergence-free dynamics of mantle and lithosphere flow advect the initially horizontal lithosphere–mantle boundary, producing a secondary perturbation by the final time (Figure 10c). The maximum topography over the simulation time was identical between the GPU and CPU executions (Figure 10). The topographies from the CPU and GPU are consistent with the analytical solution and the reported curve of Hillebrand et al. (2014), indicating that the physical solution on GPU remains consistent despite the larger number of GMRES iterations.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

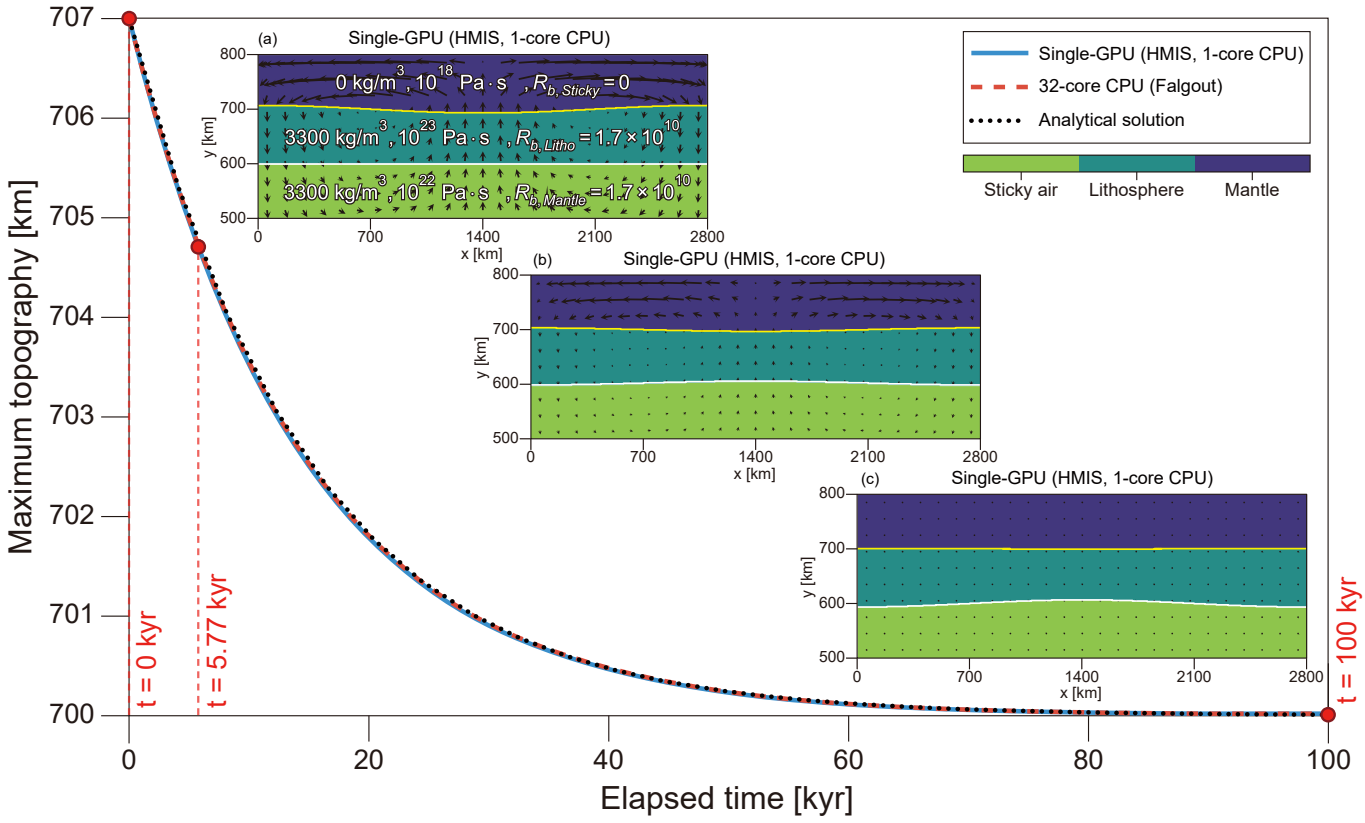

Figure 10. Evolution of maximum topography in the sticky-air benchmark. The solid blue line denotes the single-GPU (HMIS) with 1-core CPU, the dashed red line denotes the 32-core CPU computation, and the dotted black line denotes the analytical solution. Panels (a), (b), and (c) correspond to the compositional fields at $t = 0$, 5.77, and 100 kyr, respectively, obtained from the single-GPU (HMIS) with 1-core CPU. The temporal evolution of maximum topography indicates close agreement between the GPU and CPU computations and with the analytical solution.

Execution time comparison over the total steps shows that the GPU consistently outperformed the 32-core CPU (Figure 11a). The GPU (HMIS) runtime remained stable within 4.7–11.1 s, whereas the CPU (Falgout) execution time increased from 38.1 s to more than 80 s (Figure 11a; solid blue line), corresponding to an overall speedup of roughly 7–8× for the GPU (solid red line). The GPU and CPU environments show an identical pattern in the L^2-norm of the velocity field, which decreased as the initial topographic perturbation decayed (Figure 11a; solid green and dotted yellow lines). The number of PCG iterations varied with simulation time, while the GPU and CPU required the same number of PCG iterations at each time step (Figure 11b). Time steps with a larger number of PCG iterations also exhibited a larger number of GMRES iterations within the iteration loop. The GPU and CPU required 400–600 and 150–200 GMRES iterations, respectively. To evaluate the efficiency of CPU stiffness matrix assembly, GPU GMRES iterations, and GPU-CPU data transfer, we compared the total simulation wall time across different CPU-GPU configurations (Figure 11c). The 32-core CPU-only simulation required 1740 s. The optimal

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

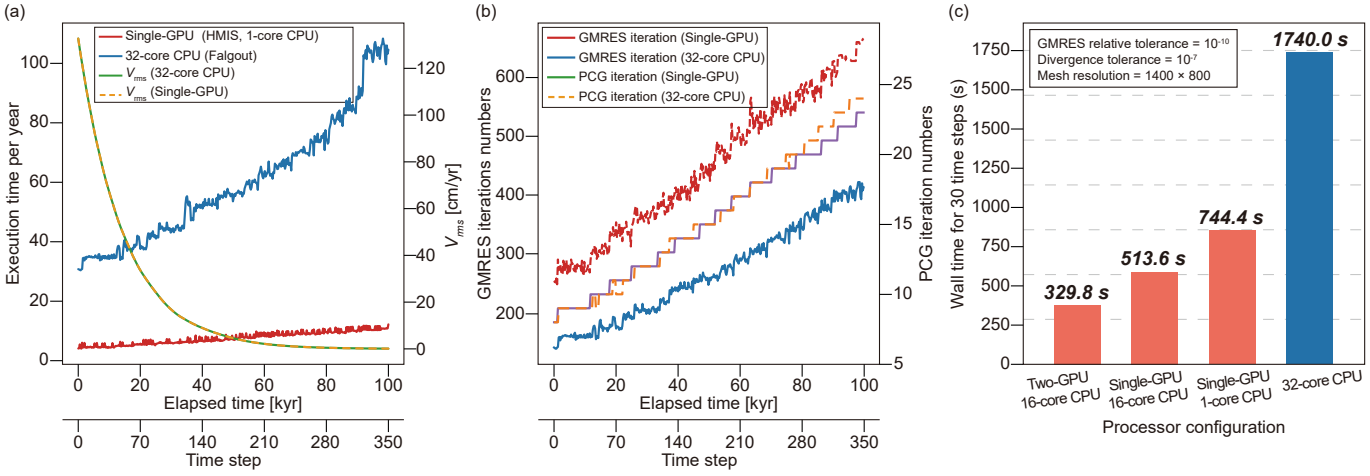

Figure 11. Performance analysis and accuracy validation of the sticky-air benchmark. (a) Execution time per step (left y-axis) and root-mean-square velocity (V_{rms}, right y-axis) over elapsed time derived from single-GPU (HMIS) with 1 core CPU. (b) Evolution of inner GMRES iteration numbers (left axis) and preconditioned conjugate-gradient (PCG) iteration numbers (right axis). (c) Wall time for 30 time steps under four processor configurations: two-GPU with 16-core CPU, single-GPU with 16-core CPU, single-GPU with 1-core CPU, and 32-core CPU (CPU-only).

configuration, two-GPU with 16-core CPU, achieved the shortest wall time of 329.8 s, corresponding to a $5.2\times$ speedup over the 32-core CPU-only execution. Using a single-GPU with 16-core CPU increased the wall time to 513.6 s due to reduced GPU resources. With a single-GPU and single CPU core, wall time further increased to 744.4 s, as CPU assembly became the dominant bottleneck. Nevertheless, even this single-GPU with 1-core CPU configuration achieved a $2.3\times$ speedup relative to the 32-core CPU-only case.

4.7 Subduction modelling

The subduction application targets 2D nonlinear viscous Stokes flow incorporating a plate interface where one plate descends into the mantle. The computational domain is a rectangle with a width of 3960 km and a depth of 660 km, which is then nondimensionalized (Enns et al., 2005; Capitanio et al., 2010; Sharples et al., 2014). Characteristic scales for nondimensionalization are listed in Table S2. The Stokes system was discretized with $P_2 \times P_1$ elements on a 1200 × 200 structured grids, where each rectangular cell was subdivided into two triangular elements along the lower left to upper right diagonal. The compositional structure is partitioned into five regions: mantle, overriding plate, upper slab layer, lower slab layer, and slab core. Four level-set functions, ϕ_1 through ϕ_4, delineate the region interfaces.

The upper mantle serves as the reference fluid with viscosity $\eta_{UM} = 1$ and $\rho = 0$. The overriding plate has the viscosity $\eta_{OP} = 400$, yielding a viscosity contrast of 400 relative to the upper mantle, while its density matches the mantle value. The viscosity of the upper slab layer (η_{US}) follows a Mohr–Coulomb plasticity law (Equation 28).

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

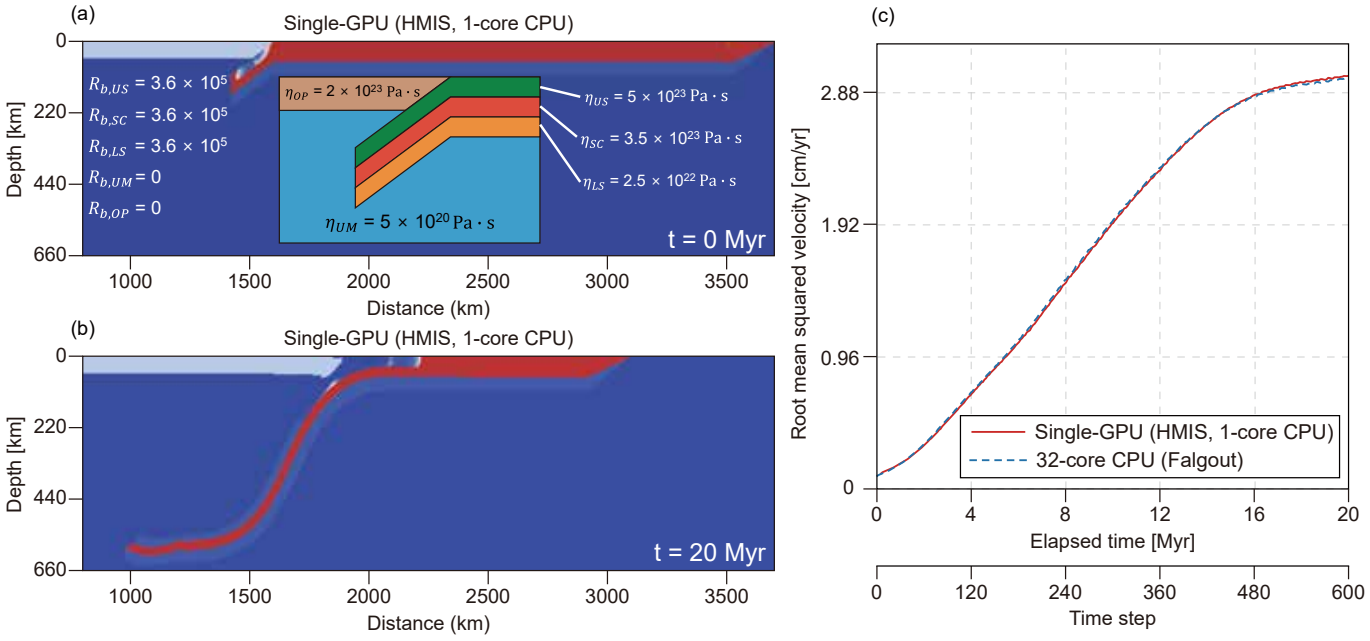

Figure 12. Temporal evolution of subduction modelling derived from the single-GPU (HMIS) with 1-core CPU and 32-core CPU. (a), (b) Viscosity field at 0, 14, and 20 Myr obtained from the single-GPU (HMIS) with 1-core CPU. (c) The root-mean-square velocity (V_{rms}) over the elapsed time.

$$\eta_{US} = \frac{1}{2}\frac{\tau_y}{\dot{\varepsilon}_{II}}, \quad \dot{\varepsilon}_{II} = \sqrt{\frac{1}{2}\dot{\boldsymbol{\varepsilon}} : \dot{\boldsymbol{\varepsilon}}} \tag{28}$$

τ_y, $\dot{\boldsymbol{\varepsilon}}$, and $\dot{\varepsilon}_{II}$ denote the plastic yield stress, strain-rate tensor, and the square root of its second invariant, respectively. τ_y was set to 0.06. η_{US} is bounded between 1 and 1000 to improve numerical stability. Additionally, at nondimensional depth $y < 0.7$, weakening is imposed by prescribing $\eta = 50$. The viscosities of the slab core (η_{SC}) and lower slab (η_{LS}) are 70 and 50, respectively. Mohr–Coulomb plasticity in the upper slab layer introduces nonlinear viscosity into the Stokes system, requiring Picard fixed-point iteration with a relative tolerance of 10^{-3}. As the viscosity contrast increases and compositional interfaces become more complex, the condition number of the discretized Stokes system tends to increase, degrading Krylov solver convergence. We therefore approximated the condition number at each time step using the ratio of the largest to smallest eigenvalues of the stiffness matrix and examined its relationship with the GMRES iteration count and execution time. An initial slab perturbation, pre-subducted to 200 km depth with a 29° dip, triggers subduction (Figures 12a). The slab maintains a steep dip and ultimately descends from the upper to the lower mantle (Figures 12b). The two environments yield nearly identical V_{rms} values, indicating that the GPU and CPU solutions remain essentially consistent even for the nonlinear-viscosity Stokes problem (Figure 12c).

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

The performance gap between the GPU and CPU persists in the subduction model. Over the full time-stepping sequence, the GPU reduces execution time by a factor of 3–4 relative to the 32-core CPU run (Figure 13a). In geodynamic simulations, time evolution increases the spatial complexity of density and viscosity fields, often accompanied by large material contrasts. Such conditions worsen the system conditioning, increasing the condition number (purple line in Figure 13a). Higher condition numbers require more GMRES iterations. In the subduction model, the estimated condition number peaks around the 300th time step (Figure 13b). As the condition number increases, execution time increases in both the GPU and CPU runs (Figure 13b). However, the regression between GMRES iterations per step and condition number shows a steeper slope for the GPU (Figure 13c). Despite this steeper slope, the GPU remains approximately $4\times$ faster in execution time, as its lower per-iteration cost keeps the total execution time shorter. The total simulation wall time was evaluated across multiple CPU-GPU configurations (Figure 13d). The 32-core CPU-only run took 2484.68 s. The fastest setup, two-GPU with 16-core CPU, achieved a wall time of 426.25 s, corresponding to a $5.8\times$ speedup relative to the 32-core CPU-only execution. Switching to single-GPU with 16-core CPU increased the wall time to 791.45 s due to the reduced GPU resources. On single-GPU with 1-core CPU environment, wall time decreased to 717.19 s, which contrasts with the trend observed at higher numbers of cores.

4.8 Multi-GPU Implementation

A multi-GPU configuration is essential to overcome the VRAM limit of a single-GPU. At high resolution, the growth in DOFs substantially increases the memory requirements for the stiffness matrix, the preconditioner, and the global force vector. Multi-GPU execution distributes VRAM usage via MPI-based domain decomposition, which also enables additional speedup through parallel execution. Inter-GPU communication and synchronization, however, introduce extra computational costs. We evaluated how execution time varies with resolution by comparing a single-GPU with two-GPU across seven benchmarks in Sections 4.1 to 4.7. In the velocity subproblem using P_2 elements, the single-GPU achieved shorter execution time than the multi-GPU (2 GPUs) environment at the lowest resolution of 128×128 (1.32×10^5 DOFs). In the low-resolution regime, communication and synchronization overhead exceeded the performance gain from parallelization. Beyond 256×256, however, the two-GPU environment achieved shorter runtimes than the single-GPU. In the high-resolution regime, the increased computational workload sufficiently outweighed the overhead, yielding a clear performance advantage for two-GPU runs. In addition, memory capacity limited the single-GPU to a maximum resolution of 1280×1280 (1.31×10^7 DOFs), whereas the two-GPU configuration ran up to 1536×1536 (1.89×10^7 DOFs). We compared the execution time of single-GPU runs with two-GPU runs across mesh resolutions for the SolCx benchmark using $P_2 \times P_1$ elements (Figure 14b). In the high-resolution regime ($> 640 \times 640$, corresponding to 3.69×10^6 DOFs), the two-GPU configuration outperformed the single-GPU by a factor of 1.1 (for 640×640) to 1.8 (for 1664×1664). The two-GPU configuration executed up to 1664×1664 (2.49×10^7 DOFs). Memory capacity limited the single-GPU to a maximum resolution of 1280×1280 (1.47×10^7 DOFs).

For the 3D thermo-mechanical convection using the MINI element, the grid resolutions in the x and z directions were held fixed at 100 cells (Figure 14c), while the y-direction resolution increased from 10 to 80. Two-GPU delivered shorter execution time at every resolution. The speedup increased from 1.05 to 1.7 as the resolution increased. The 2D Rayleigh–Taylor instability test with $P_2 \times P_1$ elements showed the same crossover behaviour as the 2D SolCx benchmark (Figure 14d).

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

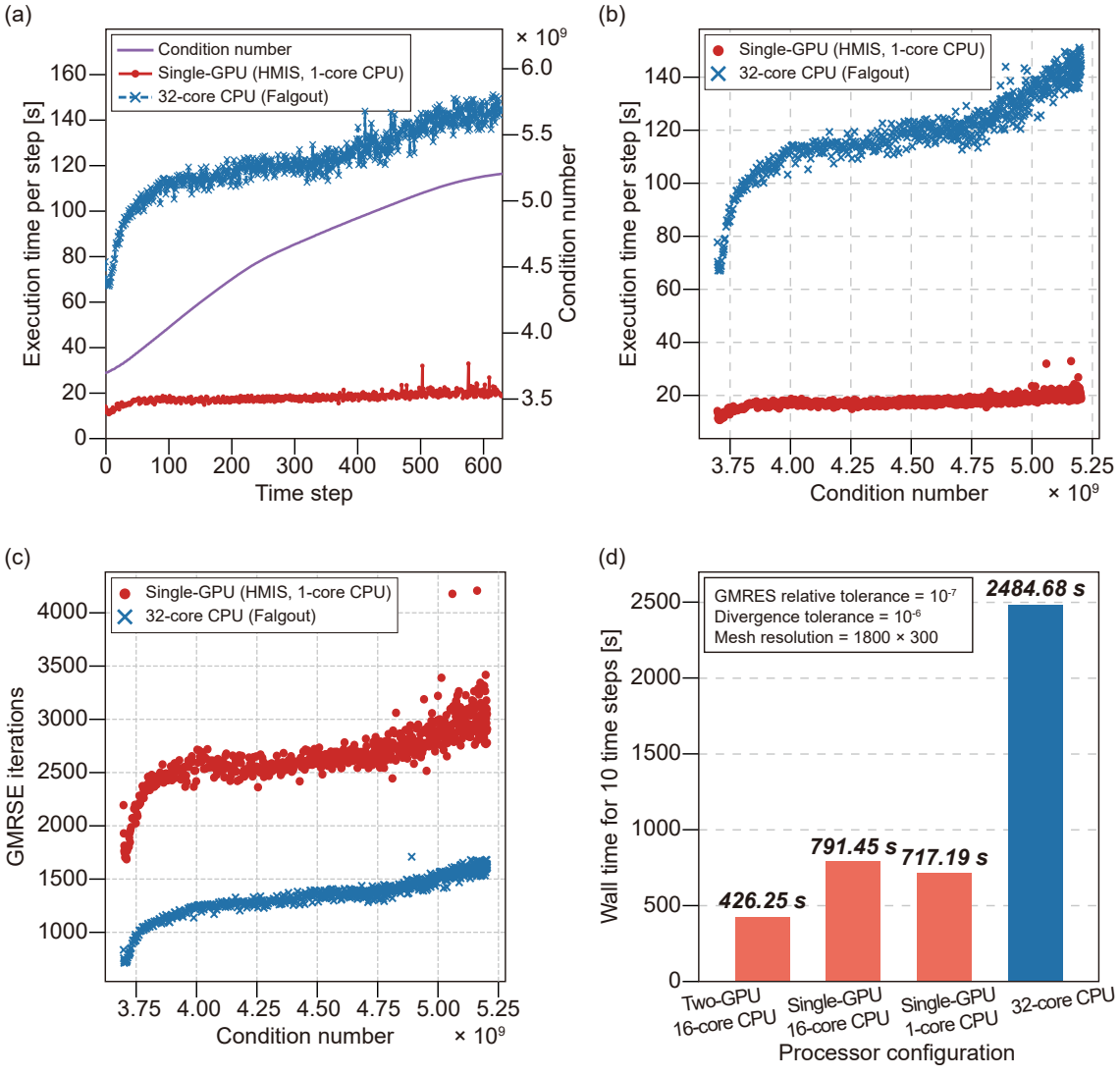

Figure 13. Analysis of GPU performance and matrix condition number for subduction modelling. (a) Evolution of the condition number and corresponding changes in execution time for single-GPU (HMIS) with 1-core CPU and 32-core CPU (Falgout) over elapsed time. (b) Correlation between the matrix condition number and inner GMRES iterations per step for the subduction model. (c) Correlation between the matrix condition number and execution time per step for the subduction model. (d) Wall time for 30 time steps under four processor configurations: two-GPU with 16-core CPU, single-GPU with 16-core CPU, single-GPU with 1-core CPU, and 32-core CPU (CPU-only).

The crossover occurred at 512×512 (2.36×10^6 DOFs), after which the two-GPU execution time remained shorter than the single-GPU execution time. In the 3D Rayleigh–Taylor instability test (the MINI element), the grid counts in the x, y, and z directions were increased at the same rate, producing a cubic increase in DOFs (Figure 14e). Across the full resolution range, the two-GPU runtime remained shorter than the single-GPU runtime. The two-GPU configuration achieved a speedup

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

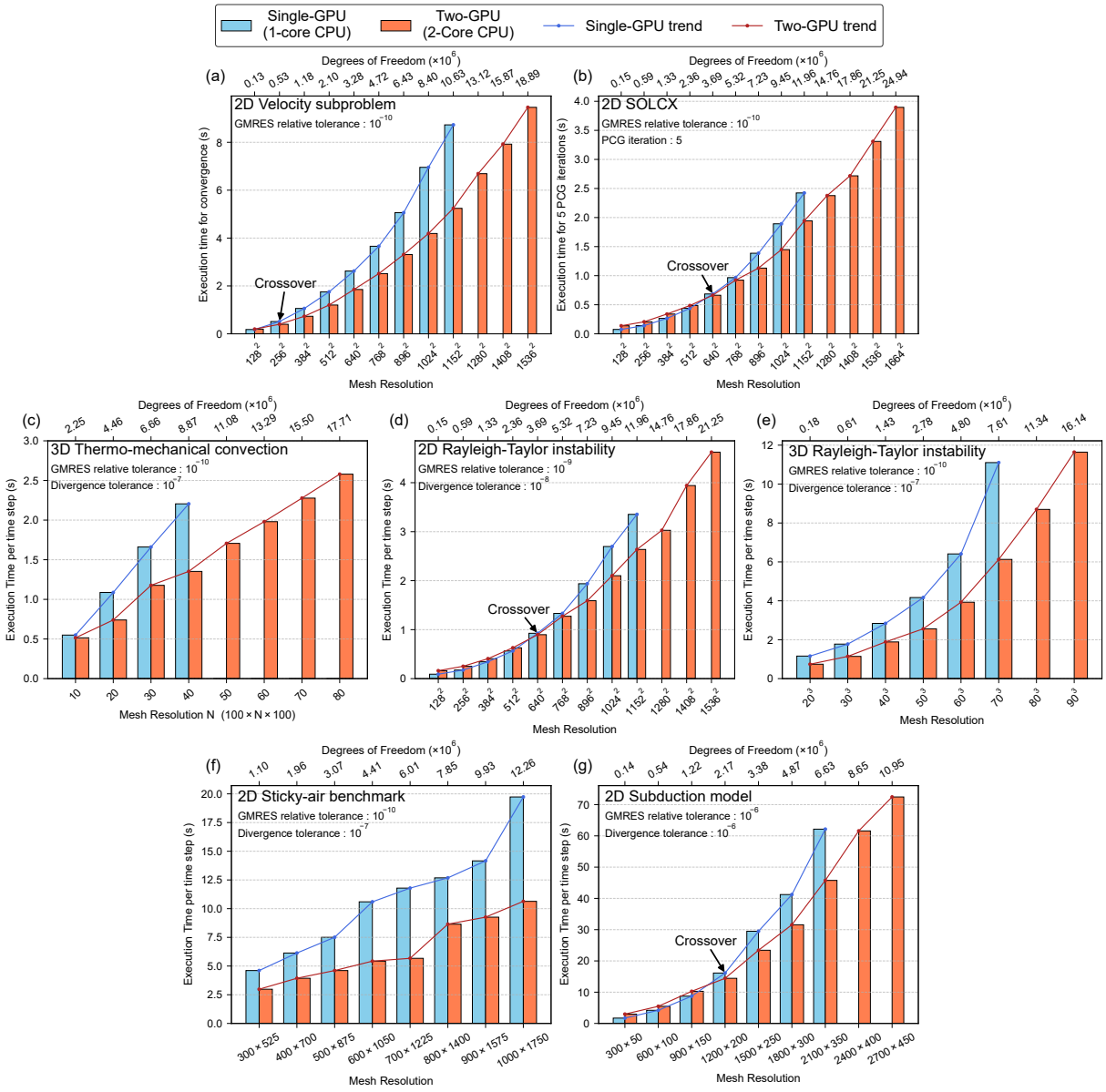

Figure 14. Performance comparison between single-GPU with 1-core CPU and two-GPU configurations in this study. (a)–(g) Comparison of execution times for the velocity subproblem, SolCx, 3D thermo-mechanical convection, 2D/3D Rayleigh-Taylor instability, sticky-air benchmark, and subduction model. Horizontal axes represent mesh resolution and degrees of freedom; vertical axes represent execution time. Black arrows indicate crossover points where the two-GPU configuration begins to outperform the single-GPU setup as problem size increases.

of ~ 1.8 at $70 \times 70 \times 70$, while also extending the maximum resolution to $90 \times 90 \times 90$. In the sticky air benchmark using $P_1^+ \times P_1$ elements, the two-GPU configuration achieved up to $1.9\times$ speedup at high resolution (Figure 14f), compared to the

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

single-GPU configuration. In the subduction model with nonlinear viscosity, the execution time crossover from single-GPU to two-GPU occurred at 792×132. Then, the two-GPU speedup reached $1.9\times$ (Figure 14g). Across all 2D benchmarks, the single-GPU ran faster at low resolution. Once the DOFs exceeded a threshold, the computational workload dominated, reducing the relative impact of overhead, and the two-GPU configuration became more effective. In 3D problems, the computational burden increased more rapidly with DOFs, yielding a consistent advantage for the two-GPU configuration across all resolutions. Two-GPU execution also distributed VRAM usage, enabling higher-resolution runs.

5 Discussion

5.1 Efficiency of pre-assembled operators

In this study, we present a practical strategy to reduce bottlenecks in a single-GPU workflow, where one CPU core assembles matrices and the GPU performs iterative solves. Before performing GPU-based PCG solves, the L^2-projection mass matrix, gradient operator, and divergence operator are assembled once on the CPU and stored in GPU VRAM. This one-time assembly eliminates per-iteration assembly costs, reducing the assembly fraction of the total wall time. The approach becomes increasingly beneficial when many PCG steps are nested inside Picard iterations. The benefits are most pronounced for iteration-heavy workloads, such as the Picard iterations for the nonlinear-viscosity subduction model (Section 4.7). We note that the stiffness matrix must still be reassembled when viscosity changes between time steps. Storing preassembled operators increases VRAM requirements as the number of DOFs increases. For an incompressible Stokes system on a fixed mesh, the gradient and divergence operators for velocity–pressure coupling remain constant across iterations. For example, at a $100 \times 40 \times 100$ resolution with the MINI element for the 3D thermo-mechanical convection benchmark (Section 4.3), storing these operators and associated vectors requires approximately 1.3 GB of VRAM (Table S3), which is a modest fraction of the ~ 47 GB peak memory usage recorded.

Nonlinear Stokes systems with strain-rate-dependent viscosity still require repeated assembly of the stiffness matrix on the CPU, which causes CPU–GPU data-transfer overhead. In multi-GPU runs, stiffness matrix assembly can be distributed across multiple CPU cores via MPI parallelism, offering additional assembly-speed improvements. GPUs are designed to maximise performance by executing large numbers of operations simultaneously. When many CPU cores assemble MPI-distributed matrix data and offload computations to a small number of GPUs (two in our study), technologies such as NVIDIA MPS (Multi-Process Service) become essential. MPS collects computation requests from multiple MPI processes and forwards them to the GPU. The GPU then executes the distributed matrix operations concurrently via MPS. In both 2D and 3D tests, configurations that used MPS to connect 16-core CPUs to single- or two-GPU consistently achieved shorter wall times than the single-GPU with 1-core CPU configuration. For the 3D Rayleigh–Taylor instability with a mesh resolution of $65 \times 65 \times 65$, the wall time in a 16-core CPU and two-GPU configuration was reduced by a factor of three compared with the single-GPU with 1-core CPU configuration. This indicates that for 3D high-resolution problems, an MPS-based multi-core-CPU and multi-GPU setup is most effective.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

A key limitation of the multi-GPU implementation based on FEniCS is the incompatibility between FEniCS's high-level function layer and GPU-aware MPI protocols. When GPU-direct communication is enabled, ghost-node values at MPI partition interfaces are incorrectly set to zero. FEniCS's topology handling and DOF mapping depend on system-memory data structures, while the underlying PETSc backend operates on CUDA vectors. During ghost-value synchronization, FEniCS relies on system memory for scatter/gather operations, whereas GPU-aware MPI relies on device-pointer communication (Mills et al., 2021). This mismatch causes memory incoherence, preventing correct updates of interface values. We addressed this issue by using native PETSc calls for pre-assembly. We constructed the force vector directly via differential and projection operators, bypassing the problematic FEniCS layer. This approach preserves correct boundary synchronization via PETSc-GPU communication protocols while maintaining GPU-aware MPI performance.

Recent work has also explored GPU-accelerated matrix assembly. FEniCSx and Firedrake are adopting CUDA-kernel assembly via projects such as cuda-dolfinx and PyOP2 (Trotter et al., 2023; Rathgeber et al., 2016). Currently, GPU-side assembly remains experimental, with most implementations delivered through extension projects rather than stable support. A longer-term goal is a fully GPU-resident workflow that performs both assembly and linear solves on the GPU, further reducing transfer overhead. Until GPU-side global matrix assembly becomes widely adopted in the geodynamics community, our approach offers a practical alternative by keeping assembly and host–device transfer costs modest while delivering $\sim 5\times$ shorter wall time (16-core CPU and two-GPU) than the 32-core CPU-only configuration.

5.2 Computational efficiency of multi-GPUs

Multi-GPU execution is essential for high-resolution geodynamics, as the approach shortens execution time while extending the maximum mesh resolution and total DOFs depending on VRAM capacity. Our benchmark tests show a clear path towards higher-resolution Stokes modelling in both 2D and 3D through single-GPU speedups and multi-GPU scaling. Recent 2D geodynamics studies increasingly use finer meshes to represent subduction zones, plate boundaries, shear zones, and boundary layers near free surfaces without sacrificing local resolution. For example, a high-resolution rifting setup uses 0.5 km spacing at depths shallower than 200 km (Yu et al., 2025). High-resolution 2D subduction and continental collision studies employed a mesh of 1024×512 elements, extending to 2048×1024 elements for resolution-sensitivity tests (Laik et al., 2023). Under such high-resolution settings, Stokes DOFs and preconditioner memory requirements increase sharply, making GPU VRAM capacity and solver execution time key constraints. In our 2D SolCx benchmarks, two-GPU outperformed a single-GPU beyond 640×640 elements. Execution time speedups reached 4–$11\times$ on a single-GPU compared to the 32-core CPU, while two-GPU delivered an additional 1.1–$1.9\times$ improvement compared to a single-GPU. These speedup factors enable higher-resolution 2D rifting and subduction simulations under tight VRAM limits.

In 3D, DOF scaling becomes much steeper, leading to memory bottlenecks due to the stiffness matrix, preconditioner, and force vector. Building the preconditioner also consumes extra VRAM through coarsening levels, pushing a single-GPU to its memory limit at relatively modest resolutions. Ultra-large 3D Stokes problems therefore require massive multi-GPU execution. Räss et al. (2022) used a pseudo-transient method on 2197 GPUs to solve 3D variable-viscosity Stokes flow at 4995^3 resolution, reaching 1.2 tera-DOFs. The pseudo-transient framework targets ultra-high-resolution finite-difference simulations

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

at extreme multi-GPU scale. We propose that multi-GPU scaling directly supports larger DOF targets within available VRAM. An extrapolation based on our DOF scaling suggests that eight RTX 6000 Ada cards (48 GB VRAM each) would support roughly 7.0×10^7 DOFs. Replacing this configuration with RTX PRO 6000 cards (96 GB VRAM each) would yield a total of 768 GB VRAM, enabling over 1.4×10^8 DOFs under the same extrapolation.

Krylov subspace solvers for the Stokes system (GMRES, CG, and MINRES) depend strongly on FP64 (double-precision) performance. The RTX 6000 Ada delivers an FP64 peak near 1.4 TFLOPS, while HPC-class GPUs such as the H200 provide much higher FP64 performance and larger memory capacity. For example, the H200 delivers 34 TFLOPS, roughly 24 times higher than the RTX 6000 Ada. Using H200-class systems would therefore significantly improve Stokes solver performance at high resolution. Such scaling enables higher-resolution simulations of global mantle convection and complex crustal deformation, demonstrating how GPU system scaling shapes next-generation geodynamics.

5.3 Potential benefits of large-scale geodynamic models

GPU acceleration reduces the execution time of Stokes solves in both single- and multi-GPU configurations. Single-GPU execution reduces the time per solve, while multi-GPU execution also increases the maximum DOFs that can be handled by distributing memory demand across GPUs. This combination enables high-resolution forward modelling and inverse problems that are difficult to run within the execution time and memory limits of conventional CPU clusters. Multi-GPU configurations further support scaling from local domains to global models by combining larger total VRAM with parallel execution.

Accurate simulation of narrow shear zones or thin lower-mantle thermal boundary layers in a 3D spherical-shell domain often requires grid spacing on the order of tens of kilometers (Taiwo et al., 2023; Murphy and King, 2024), which drives the Stokes system to very large DOF. In this setting, the GPU-based PCG solver tested in our study remains effective at large 3D scales and is expected to be important for future global simulations. Li et al. (2023) reconstructed the subduction history of the North American plate using plate-reconstruction-driven data assimilation and emphasised the high computational cost associated with realistic 3D buoyancy and viscosity structures. The study combined seafloor-age constraints and lithosphere-density profiles to build spatially variable buoyancy forcing and viscosity fields. The resulting configuration increased computational cost by more than an order of magnitude relative to simplified models. Such cost limits further extensions, including coupling with surface processes or increasing spatial resolution in global simulations. Faster Stokes solves on GPUs, together with multi-GPU scaling that supports larger DOF cases, are therefore critical for making such high-fidelity data-assimilation models more practical.

GPU acceleration also reduces the cost of adjoint inversion. Adjoint inversion minimises the misfit between an observed mantle state and a model prediction by iteratively updating initial conditions or model parameters. Each iteration computes the misfit through a forward simulation, then solves the adjoint equations backwards in time to obtain sensitivity information used to update the model. Since one iteration requires both forward and adjoint integrations over the full time window, the total cost scales with the number of iterations and the length of the assimilation period. Colli et al. (2020) and Saxena et al. (2023) limited the inversion window to 50 Myr (rather than 100 Myr) to manage the computational cost and excluded an inner loop for calibrating parameters (e.g., viscosity). The GPU speedups demonstrated in our study increase the number of forward-adjoint

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

cycles that fit within a fixed runtime budget. This improvement supports longer assimilation windows. The same improvement also makes repeated parameter-calibration loops, including viscosity updates, more practical. GPU acceleration therefore provides the computational basis for extended inversion workflows that better constrain uncertain material properties and initial conditions in high-resolution geodynamics.

5.4 State-of-the-art Stokes solving scheme using block preconditioner

In computational geodynamics, one of the state-of-the-art Stokes solvers is based on Schur-complement block preconditioners, which are widely used in major community codes such as ASPECT (Kronbichler et al., 2012). A block preconditioner handles the saddle-point problem, with an AMG (or GMG) preconditioner applied to the velocity block and the pressure Schur complement approximated by a viscosity-weighted mass matrix or a BFBT-type scheme. This configuration delivers mesh-independent convergence and excellent scalability on large parallel systems (Benzi et al., 2005; May and Moresi, 2008; Rudi et al., 2017). In contrast, the PCG solver adopted in this study applies a conjugate-gradient iteration to the pressure Schur complement system. Because it stores only a fixed number of vectors, this approach has a smaller memory footprint than block-preconditioned FGMRES, which is advantageous in GPU environments where VRAM capacity is a binding constraint.

To evaluate this solver's effectiveness, we compared a FEniCS-based PCG implementation with ASPECT's block preconditioned solver for the 3D thermo mechanical convection benchmark (Table S4). The FEniCS implementation used two-GPU with 16-core CPU, single-GPU with 16-core, and 32-core CPU, while the ASPECT implementation used 32-core CPU. To enable a fair comparison with ASPECT, which provides Q_2/Q_1 elements, we used a $P_2 \times P_1$ discretization in FEniCS with a resolution of $40 \times 40 \times 40$. The total elapsed time is 0.25 with 250 steps. On 32-core CPU, ASPECT required a wall time of 1473 s to solve the Stokes system over the full elapsed run. Our PCG solver on 32-core CPU required a wall time of 2782 s, indicating that ASPECT's block-preconditioned Stokes solver outperforms our PCG solver on CPUs in this configuration. However, GPU acceleration reverses this result. On a single-GPU with 16-core CPU setup, the same case required approximately 200 s, corresponding to speedups of approximately $13.5\times$ and $7.37\times$ relative to our 32-core PCG run and the 32-core ASPECT run, respectively. On two-GPU with 16-core CPU, the wall time decreased to approximately 140 s, delivering an additional $1.4\times$ speedup over the single-GPU case and overall speedups of $20\times$ and $10.5\times$ relative to our 32-core CPU run and ASPECT, respectively.

A block-preconditioned formulation solves the full Stokes system with block preconditioned FGMRES, and a carefully tuned GPU implementation of this strategy may ultimately be faster than the PCG splitting approach. At the same time, the full Stokes system is larger and each FGMRES iteration must store multiple Krylov vectors, which can lead to higher memory usage, especially in multi-GPU configurations. Identifying the most effective solver in GPU-accelerated settings will therefore require future work that systematically compares PCG splitting and block-preconditioned FGMRES methods in terms of both execution time and memory usage.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

6 Conclusions

In this study, we implemented a preconditioned conjugate-gradient solver to solve finite-element Stokes equations in FEniCS on GPUs using PETSc configured with CUDA and validated both performance and accuracy through quantitative comparisons with CPU runs. We adopted the PCG variant, in which the pressure update follows a preconditioned conjugate-gradient scheme that limits the number of stored vectors, producing an algorithm better suited to GPU memory constraints. By using PETSc AIJCUSPARSE matrices and CUDA vectors, we stored matrices and vectors on the GPU. To reduce the L^2-projection bottleneck that recurs within PCG iterations, we preassembled and reused the mass matrix and the gradient and divergence operators. As a preconditioner, we employed HYPRE BoomerAMG, with optimal configurations varying by hardware architecture. On GPUs, HMIS or PMIS coarsening reduced the cost of coarse-grid construction, while the combination of scaled Jacobi smoothing and Extended+i interpolation improved parallel efficiency. We validated the GPU solver across seven benchmarks: a manufactured solution, SolCx, the sticky-air benchmark, 2D and 3D Rayleigh-Taylor instability, 3D thermo-mechanical convection, and a subduction model with nonlinear viscosity. GPU-based solutions maintained accuracy comparable to analytical solutions and CPU results. Robust convergence was also observed in cases featuring strong viscosity contrasts and sharp compositional boundaries. As problem size increased, GPU advantages became more pronounced, with execution times reduced by factors of 5–11 compared to 32-core CPU execution. Even when GMRES required more iterations on the GPU, the lower per-iteration cost on the GPU often reduced the total execution time. Experiments using single-GPU and two-GPU configurations with RTX 6000 Ada cards showed that single-GPU VRAM capacity limits the achievable resolution, whereas multi-GPU domain decomposition enables larger simulations and provides additional speedup. Moreover, leveraging multi-core CPUs together with NVIDIA MPS reduced the wall time by a factor of $\sim$6 compared to a 32-core CPU run.

Code and data availability. GAUZZ 1.0.0, comprising the FEniCS-based source code, the simulation input scripts used to reproduce all benchmark cases in this study, and a Dockerfile that builds the full computational environment, is archived at Zenodo (https://doi.org/10.5281/zenodo.19936267, Lee et al. 2026) under the GNU General Public License v3.0. Active development of GAUZZ continues at https://github.com/kmlee-geo/gauzz, where issues and contributions are welcomed. The large analytical reference solutions for the SolCx benchmark (`p_a1024.h5` and `u_a1024.h5`) are provided in the Zenodo deposit only, owing to GitHub's per-file size limits; they are required to reproduce the SolCx error analysis.

Simulations were performed with FEniCS 2019.2.0.64, NumPy 1.23.5, PETSc 3.15.5 (CPU) and 3.23.2 (GPU) compiled with CUDA support, and HYPRE BoomerAMG via the PETSc interface. The CPU runs were executed on Intel Xeon Gold 5320 and Intel Xeon w5-2465X processors; the GPU runs were executed on NVIDIA RTX 6000 Ada (48 GB VRAM) workstations.

Author contributions. K.-M. L. designed the algorithms, implemented the GPU-accelerated PCG solver, carried out all numerical experiments, performed the performance analyses, and prepared the first draft of the manuscript. D.-K. J. contributed to the mathematical formulation of the preconditioned conjugate-gradient algorithm and the Schur complement reduction. C. T. contributed substantially to shaping

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

the core ideas of the paper through revisions, provided expertise on the geodynamic benchmark setups, and provided major input during the manuscript editing process. B.-D. S. supervised the project, acquired funding, and revised the manuscript. All authors discussed the results and contributed to the final version of the paper.

Competing interests. The authors declare that they have no competing interests.

Acknowledgements. Kyeong-Min Lee was supported by the Ministry of the Interior and Safety through the Human Resource Development Project in Disaster Management. Model visualisation was performed using ParaView (https://www.paraview.org).

Financial support. This research has been supported by the National Research Foundation of Korea (NRF) grants funded by the Korean Government (MSIT) (grant nos. RS-2025-02293161, NRF-2022R1A2C1009742, and RS-2025-25428518).

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

References

Adams, A. C., Stegman, D. R., Mohammadzadeh, H., Smrekar, S. E., and Tackley, P. J.: Plume-Induced Delamination Initiated at Rift Zones on Venus, Journal of Geophysical Research: Planets, 128, e2023JE007 879, https://doi.org/10.1029/2023JE007879, 2023.

Alber, D. M. and Olson, L. N.: Parallel coarse-grid selection, Numerical Linear Algebra with Applications, 14, 611–643, https://doi.org/10.1002/nla.541, 2007.

Alkhimenkov, Y. and Podladchikov, Y. Y.: Accelerated pseudo-transient method for elastic, viscoelastic, and coupled hydro-mechanical problems with applications, Geoscientific Model Development Discussions, 2024, 1–35, https://doi.org/10.5194/gmd-2024-160, 2024.

Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N.: The FEniCS project version 1.5, Archive of Numerical Software, 3, https://doi.org/10.11588/ans.2015.100.20553, 2015.

Amestoy, P. R., Duff, I. S., L'Excellent, J. Y., and Koster, J.: MUMPS: a general purpose distributed memory sparse solver, in: International Workshop on Applied Parallel Computing, pp. 121–130, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/3-540-70734-4_16, 2000.

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, J.: PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 3.12, Argonne National Laboratory, Lemont, IL, USA, 2019.

Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E. M., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M. G., Kong, F., Kruger, S., May, D. A., McInnes, L. C., Mills, R. T., Mitchell, L., Munson, T., Roman, J. E., Rupp, K., Sanan, P., Sarich, J., Smith, B. F., Zampini, S., Zhang, H., and Zhang, J.: PETSc/TAO Users Manual, Revision 3.20, Tech. Rep. ANL-21/39, Argonne National Laboratory, Argonne, IL, USA, https://doi.org/10.2172/2205494, 2023.

Benzi, M., Golub, G. H., and Liesen, J.: Numerical solution of saddle point problems, Acta Numerica, 14, 1–137, https://doi.org/10.1017/S0962492904000212, 2005.

Beuchert, M. J. and Podladchikov, Y. Y.: Viscoelastic mantle convection and lithospheric stresses, Geophysical Journal International, 183, 35–63, https://doi.org/10.1111/j.1365-246X.2010.04708.x, 2010.

Blankenbach, B., Busse, F., Christensen, U., Cserepes, L., Gunkel, D., Hansen, U., Harder, H., Jarvis, G., Koch, M., Marquart, G., Moore, D., Olson, P., Schmeling, H., and Schnaubelt, T.: A benchmark comparison for mantle convection codes, Geophysical Journal International, 98, 23–38, https://doi.org/10.1111/j.1365-246X.1989.tb05511.x, 1989.

Bourgouin, L., Mühlhaus, H. B., Hale, A. J., and Arsac, A.: Towards realistic simulations of lava dome growth using the level set method, Acta Geotechnica, 1, 225–236, https://doi.org/10.1007/s11440-006-0016-6, 2006.

Brandt, A.: Multi-level adaptive solutions to boundary-value problems, Mathematics of Computation, 31, 333–390, https://doi.org/10.1090/S0025-5718-1977-0431719-X, 1977.

Brannick, J., Cao, F., Kahl, K., Falgout, R. D., and Hu, X.: Optimal interpolation and compatible relaxation in classical algebraic multigrid, SIAM Journal on Scientific Computing, 40, A1473–A1493, https://doi.org/10.1137/17M1123456, 2018.

Brune, S., Popov, A. A., and Sobolev, S. V.: Modeling suggests that oblique extension facilitates rifting and continental break-up, Journal of Geophysical Research: Solid Earth, 117, https://doi.org/10.1029/2012JB009304, 2012.

Burstedde, C., Stadler, G., Alisic, L., Wilcox, L. C., Tan, E., Gurnis, M., and Ghattas, O.: Large-scale adaptive mantle convection simulation, Geophysical Journal International, 192, 889–906, https://doi.org/10.1093/gji/ggs070, 2013.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

Capitanio, F. A., Stegman, D. R., Moresi, L. N., and Sharples, W.: Upper plate controls on deep subduction, trench migrations and deformations at convergent margins, Tectonophysics, 483, 80–92, https://doi.org/10.1016/j.tecto.2009.08.024, 2010.

Clevenger, T. C. and Heister, T.: Comparison between algebraic and matrix-free geometric multigrid for a Stokes problem on adaptive meshes with variable viscosity, Numerical Linear Algebra with Applications, 28, e2375, https://doi.org/10.1002/nla.2375, 2021.

Colli, L., Bunge, H. P., and Oeser, J.: Impact of model inconsistencies on reconstructions of past mantle flow obtained using the adjoint method, Geophysical Journal International, 221, 617–639, https://doi.org/10.1093/gji/ggaa023, 2020.

Crameri, F. and Tackley, P. J.: Parameters controlling dynamically self-consistent plate tectonics and single-sided subduction in global models of mantle convection, Journal of Geophysical Research: Solid Earth, 120, 3680–3706, https://doi.org/10.1002/2014JB011664, 2015.

Crameri, F., Schmeling, H., Golabek, G. J., Duretz, T., Orendt, R., Buiter, S. J. H., May, D. A., Kaus, B. J. P., Gerya, T. V., and Tackley, P. J.: A comparison of numerical surface topography calculations in geodynamic modelling: an evaluation of the 'sticky air' method, Geophysical Journal International, 189, 38–54, https://doi.org/10.1093/gji/ggs053, 2012.

d'Acremont, E., Leroy, S., and Burov, E. B.: Numerical modelling of a mantle plume: the plume head-lithosphere interaction in the formation of an oceanic large igneous province, Earth and Planetary Science Letters, 206, 379–396, https://doi.org/10.1016/S0012-821X(02)01058-0, 2003.

Davies, D. R., Wilson, C. R., and Kramer, S. C.: Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics, Geochemistry, Geophysics, Geosystems, 12, https://doi.org/10.1029/2011GC003551, 2011.

Davies, G. F. and Gurnis, M.: Interaction of mantle dregs with convection: Lateral heterogeneity at the core-mantle boundary, Geophysical Research Letters, 13, 1517–1520, https://doi.org/10.1029/GL013i013p01517, 1986.

De Sterck, H., Yang, U. M., and Heys, J. J.: Reducing complexity in parallel algebraic multigrid preconditioners, SIAM Journal on Matrix Analysis and Applications, 27, 1019–1039, https://doi.org/10.1137/040615729, 2006.

Do, S. H., So, B. D., Kim, Y. G., and Kim, G. B.: Lithospheric strength inferred from modeling of buckling structure: Implications for stress state of the East Sea (Japan Sea), Tectonophysics, 858, 229 859, https://doi.org/10.1016/j.tecto.2023.229859, 2023.

Do, S. H., So, B. D., and Shin, Y. H.: Degree of coupling in 3D multilayer continental lithospheric buckling: Implications for tectonic underpressure, Journal of Geophysical Research: Solid Earth, 130, e2024JB029 465, https://doi.org/10.1029/2024JB029465, 2025.

Duretz, T., May, D. A., Gerya, T. V., and Tackley, P. J.: Discretization errors and free surface stabilization in the finite difference and marker-in-cell method for applied geodynamics: A numerical study, Geochemistry, Geophysics, Geosystems, 12, https://doi.org/10.1029/2011GC003567, 2011.

Enns, A., Becker, T. W., and Schmeling, H.: The dynamics of subduction and trench migration for viscosity stratification, Geophysical Journal International, 160, 761–775, https://doi.org/10.1111/j.1365-246X.2005.02519.x, 2005.

Falgout, R. D., Jones, J. E., and Yang, U. M.: The design and implementation of hypre, a library of parallel high performance preconditioners, in: Numerical Solution of Partial Differential Equations on Parallel Computers, pp. 267–294, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/3-540-31619-1_8, 2006.

Gassmöller, R., Dannberg, J., Bangerth, W., Heister, T., and Myhill, R.: On formulations of compressible mantle convection, Geophysical Journal International, 221, 1264–1280, https://doi.org/10.1093/gji/ggaa078, 2020.

Gerya, T. V. and Yuen, D. A.: Characteristics-based marker-in-cell method with conservative finite-differences schemes for modeling geological flows with strongly variable transport properties, Physics of the Earth and Planetary Interiors, 140, 293–318, https://doi.org/10.1016/j.pepi.2003.09.006, 2003.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

Gerya, T. V., May, D. A., and Duretz, T.: An adaptive staggered grid finite difference method for modeling geodynamic Stokes flows with strongly variable viscosity, Geochemistry, Geophysics, Geosystems, 14, 1200–1225, https://doi.org/10.1029/2012GC004236, 2013.

Gerya, T. V., Stern, R. J., Baes, M., Sobolev, S. V., and Whattam, S. A.: Plate tectonics on the Earth triggered by plume-induced subduction initiation, Nature, 527, 221–225, https://doi.org/10.1038/nature16174, 2015.

Graham, R. L., Woodall, T. S., and Squyres, J. M.: Open MPI: A flexible high performance MPI, in: International Conference on Parallel Processing and Applied Mathematics, pp. 228–239, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/11752578_29, 2005.

Hestenes, M. R. and Stiefel, E.: Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards, 49, 409–436, https://doi.org/10.6028/jres.049.044, 1952.

Hillebrand, B., Thieulot, C., Geenen, T., Van Den Berg, A. P., and Spakman, W.: Using the level set method in geodynamical modeling of multi-material flows and Earth's free surface, Solid Earth, 5, 1087–1098, https://doi.org/10.5194/se-5-1087-2014, 2014.

Huismans, R. S. and Beaumont, C.: Symmetric and asymmetric lithospheric extension: Relative effects of frictional-plastic and viscous strain softening, Journal of Geophysical Research: Solid Earth, 108, https://doi.org/10.1029/2002JB002026, 2003.

Jang, D.-K., Lee, K.-M., Thieulot, C., Choi, W.-H., and So, B.-D.: Efficient Uzawa algorithms with projection strategies for geodynamic Stokes flow, EGUsphere, 2025, 1–22, https://doi.org/10.5194/egusphere-2025-5480, 2025.

Keum, J. Y. and So, B. D.: Effect of buoyant sediment overlying subducting plates on trench geometry: 3D viscoelastic free subduction modeling, Geophysical Research Letters, 48, e2021GL093 498, https://doi.org/10.1029/2021GL093498, 2021.

Keum, J. Y. and So, B. D.: Sediment buoyancy controls the effective slab pull force and deviatoric stress along trenches: Insights from 3D free-subduction model, Tectonophysics, 862, 229 970, https://doi.org/10.1016/j.tecto.2023.229970, 2023.

King, S. D.: On topography and geoid from 2-D stagnant lid convection calculations, Geochemistry, Geophysics, Geosystems, 10, https://doi.org/10.1029/2008GC002250, 2009.

Kronbichler, M., Heister, T., and Bangerth, W.: High accuracy mantle convection simulation through modern numerical methods, Geophysical Journal International, 191, 12–29, https://doi.org/10.1093/gji/ggs070, 2012.

Laik, A., Schellart, W. P., and Strak, V.: Sustained indentation in 2-D models of continental collision involving whole mantle subduction, Geophysical Journal International, 232, 343–365, https://doi.org/10.1093/gji/ggac365, 2023.

Lee, K.-M., Jang, D.-K., Thieulot, C., and So, B.-D.: Codes and Dockerfile for geodynamics simulations using PETSc with GPUs (GAUZZ v1.0.0), https://doi.org/10.5281/zenodo.19936267, [code], 2026.

Leng, W. and Zhong, S.: Viscous heating, adiabatic heating and energetic consistency in compressible mantle convection, Geophysical Journal International, 173, 693–702, https://doi.org/10.1111/j.1365-246X.2008.03745.x, 2008.

Li, Y., Liu, L., Peng, D., Dong, H., and Li, S.: Evaluating tomotectonic plate reconstructions using geodynamic models with data assimilation, the case for North America, Earth-Science Reviews, 244, 104 518, https://doi.org/10.1016/j.earscirev.2023.104518, 2023.

Liao, J. and Gerya, T.: From continental rifting to seafloor spreading: insight from 3D thermo-mechanical modeling, Gondwana Research, 28, 1329–1343, https://doi.org/10.1016/j.gr.2015.01.011, 2015.

MacLachlan, S. P. and Oosterlee, C. W.: Local Fourier analysis for multigrid with overlapping smoothers applied to systems of PDEs, Numerical Linear Algebra with Applications, 18, 751–774, https://doi.org/10.1002/nla.762, 2011.

Maljaars, J. M., Richardson, C. N., and Sime, N.: LEoPart: A particle library for FEniCS, Computers & Mathematics with Applications, 81, 289–315, https://doi.org/10.1016/j.camwa.2020.04.023, 2021.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

Mansour, J., Giordani, J., Moresi, L., Beucher, R., Kaluza, O., Velic, M., Farrington, R., Quenette, S., and Beall, A.: Underworld2: Python Geodynamics Modelling for Desktop, HPC and Cloud, Journal of Open Source Software, 5, 1797, https://doi.org/10.21105/joss.01797, 2020.

May, D. A. and Moresi, L.: Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, Physics of the Earth and Planetary Interiors, 171, 33–47, https://doi.org/10.1016/j.pepi.2008.07.036, 2008.

May, D. A., Brown, J., and Le Pourhiet, L.: A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow, Computer Methods in Applied Mechanics and Engineering, 290, 496–523, https://doi.org/10.1016/j.cma.2015.03.014, 2015.

Mills, R. T., Adams, M. F., Balay, S., Brown, J., Dener, A., Knepley, M., Kruger, S. E., Morgan, H., Munson, T., Rupp, K., Smith, B. F., Zampini, S., Zhang, H., and Zhang, J.: Toward performance-portable PETSc for GPU-based exascale systems, Parallel Computing, 108, 102 831, https://doi.org/10.1016/j.parco.2021.102831, 2021.

Moresi, L., Zhong, S., and Gurnis, M.: The accuracy of finite element solutions of Stokes's flow with strongly varying viscosity, Physics of the Earth and Planetary Interiors, 97, 83–94, https://doi.org/10.1016/0031-9201(96)03163-9, 1996.

Murphy, J. P. and King, S. D.: Reconciling mars insight results, geoid, and melt evolution with 3D spherical models of convection, Journal of Geophysical Research: Planets, 129, e2023JE008 143, https://doi.org/10.1029/2023JE008143, 2024.

Muzhinji, K., Shateyi, S., and Motsa, S. S.: The mixed finite element multigrid preconditioned MINRES method for Stokes equations, Journal of Applied Fluid Mechanics, 9, 1285–1296, https://doi.org/10.18869/acadpub.jafm.68.228.24206, 2016.

Neuharth, D., Brune, S., Glerum, A., Heine, C., and Welford, J. K.: Formation of continental microplates through rift linkage: Numerical modeling and its application to the Flemish Cap and Sao Paulo Plateau, Geochemistry, Geophysics, Geosystems, 22, e2020GC009 615, https://doi.org/10.1029/2020GC009615, 2021.

Offermans, N., Peplinski, A., Marin, O., Merzari, E., and Schlatter, P.: Performance of preconditioners for large-scale simulations using Nek5000, in: Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2018: Selected Papers from the ICOSAHOM Conference, pp. 263–272, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-030-39647-3_21, 2020.

Park, J., Smelyanskiy, M., Yang, U. M., Mudigere, D., and Dubey, P.: High-performance algebraic multigrid solver optimized for multi-core based distributed parallel systems, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–12, https://doi.org/10.1145/2807591.2807603, 2015.

Pysklywec, R. N. and Shahnas, M. H.: Time-dependent surface topography in a coupled crust-mantle convection model, Geophysical Journal International, 154, 268–278, https://doi.org/10.1046/j.1365-246X.2003.01987.x, 2003.

Räss, L., Utkin, I., Duretz, T., Omlin, S., and Podladchikov, Y. Y.: Assessing the robustness and scalability of the accelerated pseudo-transient method towards exascale computing, Geoscientific Model Development Discussions, 2022, 1–46, https://doi.org/10.5194/gmd-15-5757-2022, 2022.

Rathgeber, F., Ham, D. A., Mitchell, L., Lange, M., Luporini, F., McRae, A. T. T., Bercea, G. T., Markall, G. R., and Kelly, P. H. J.: Firedrake: automating the finite element method by composing abstractions, ACM Transactions on Mathematical Software, 43, 1–27, https://doi.org/10.1145/2998441, 2016.

Regorda, A., Thieulot, C., van Zelst, I., Erdős, Z., Maia, J., and Buiter, S.: Rifting Venus: Insights from numerical modeling, Journal of Geophysical Research: Planets, 128, e2022JE007 588, https://doi.org/10.1029/2022JE007588, 2023.

Roy, A., Ghosh, D., and Mandal, N.: Dampening effect of global flows on Rayleigh-Taylor instabilities: implications for deep-mantle plumes vis-à-vis hotspot distributions, Geophysical Journal International, 236, 119–138, https://doi.org/10.1093/gji/ggad382, 2024.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

Rudi, J., Stadler, G., and Ghattas, O.: Weighted BFBT preconditioner for Stokes flow problems with highly heterogeneous viscosity, SIAM Journal on Scientific Computing, 39, S272–S297, https://doi.org/10.1137/16M108078X, 2017.

Saad, Y. and Schultz, M. H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 7, 856–869, https://doi.org/10.1137/0907058, 1986.

Saramito, P.: Complex Fluids, Springer International Publishing, Switzerland, https://doi.org/10.1007/978-3-319-44388-1, 2016.

Saxena, A., Dannberg, J., Gassmöller, R., Fraters, M., Heister, T., and Styron, R.: High-resolution mantle flow models reveal importance of plate boundary geometry and slab pull forces on generating tectonic plate motions, Journal of Geophysical Research: Solid Earth, 128, e2022JB025 877, https://doi.org/10.1029/2022JB025877, 2023.

Schenk, O., Gärtner, K., Fichtner, W., and Stricker, A.: PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation, Future Generation Computer Systems, 18, 69–78, https://doi.org/10.1016/S0167-739X(00)00076-3, 2001.

Schmalholz, S. M.: 3D numerical modeling of forward folding and reverse unfolding of a viscous single-layer: Implications for the formation of folds and fold patterns, Tectonophysics, 446, 31–41, https://doi.org/10.1016/j.tecto.2007.09.005, 2008.

Schmeling, H.: Dynamic models of continental rifting with melt generation, Tectonophysics, 480, 33–47, https://doi.org/10.1016/j.tecto.2009.09.005, 2010.

Schools, J. and Smrekar, S. E.: Formation of coronae topography and fractures via plume buoyancy and melting, Earth and Planetary Science Letters, 633, 118 643, https://doi.org/10.1016/j.epsl.2024.118643, 2024.

Sharples, W., Jadamec, M. A., Moresi, L. N., and Capitanio, F. A.: Overriding plate controls on subduction evolution, Journal of Geophysical Research: Solid Earth, 119, 6684–6704, https://doi.org/10.1002/2014JB011163, 2014.

Silvester, D. and Wathen, A.: Fast iterative solution of stabilised Stokes systems Part II: Using general block preconditioners, SIAM Journal on Numerical Analysis, 31, 1352–1367, https://doi.org/10.1137/0731070, 1994.

Suchoy, L., Goes, S., Maunder, B., Garel, F., and Davies, R.: Effects of basal drag on subduction dynamics from 2D numerical models, Solid Earth, 12, 79–93, https://doi.org/10.5194/se-12-79-2021, 2021.

Ta, T., Choo, K., Tan, E., Jang, B., and Choi, E.: Accelerating DynEarthSol3D on tightly coupled CPU-GPU heterogeneous processors, Computers & Geosciences, 79, 27–37, https://doi.org/10.1016/j.cageo.2015.03.003, 2015.

Tackley, P. J. and King, S. D.: Testing the tracer ratio method for modeling active compositional fields in mantle convection simulations, Geochemistry, Geophysics, Geosystems, 4, https://doi.org/10.1029/2001GC000214, 2003.

Taiwo, A., Bunge, H. P., Schuberth, B. S. A., Colli, L., and Vilacis, B.: Robust global mantle flow trajectories and their validation via dynamic topography histories, Geophysical Journal International, 234, 2160–2179, https://doi.org/10.1093/gji/ggad295, 2023.

Tan, E. and Gurnis, M.: Compressible thermochemical convection and application to lower mantle structures, Journal of Geophysical Research: Solid Earth, 112, https://doi.org/10.1029/2006JB004505, 2007.

Tan, E., Leng, W., Zhong, S., and Gurnis, M.: On the location of plumes and lateral movement of thermochemical structures with high bulk modulus in the 3-D compressible mantle, Geochemistry, Geophysics, Geosystems, 12, https://doi.org/10.1029/2011GC003560, 2011.

Thakur, R., Rabenseifner, R., and Gropp, W.: Optimization of collective communication operations in MPICH, The International Journal of High Performance Computing Applications, 19, 49–66, https://doi.org/10.1177/1094342005051521, 2005.

Thieulot, C.: FANTOM: Two-and three-dimensional numerical modelling of creeping flows for the solution of geological problems, Physics of the Earth and Planetary Interiors, 188, 47–68, https://doi.org/10.1016/j.pepi.2011.06.011, 2011.

Thieulot, C. and Bangerth, W.: On the choice of finite element for applications in geodynamics, Solid Earth, 13, 229–249, https://doi.org/10.5194/se-13-229-2022, 2022.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

Thieulot, C. and Bangerth, W.: On the choice of finite element for applications in geodynamics. Part II: A comparison of simplex and hypercube elements, Solid Earth, 16, 457–476, https://doi.org/10.5194/se-16-457-2025, 2025.

Trotter, J. D., Langguth, J., and Cai, X.: Targeting performance and user-friendliness: GPU-accelerated finite element computation with automated code generation in FEniCS, Parallel Computing, 118, 103 051, https://doi.org/10.1016/j.parco.2023.103051, 2023.

Uzawa, H.: Iterative methods for concave programming, Studies in Linear and Nonlinear Programming, 6, 154–165, 1958.

van Keken, P. V., King, S. D., Schmeling, H., Christensen, U. R., Neumeister, D., and Doin, M. P.: A comparison of methods for the modeling of thermochemical convection, Journal of Geophysical Research: Solid Earth, 102, 22 477–22 495, https://doi.org/10.1029/97JB01353, 1997.

Wang, L. H., Yarushina, V. M., Alkhimenkov, Y., and Podladchikov, Y.: Physics-inspired pseudo-transient method and its application in modelling focused fluid flow with geological complexity, Geophysical Journal International, 229, 1–20, https://doi.org/10.1093/gji/ggab524, 2022.

Wang, Y. and Li, M.: The interaction between mantle plumes and lithosphere and its surface expressions: 3-D numerical modelling, Geophysical Journal International, 225, 906–925, https://doi.org/10.1093/gji/ggab018, 2021.

Wathen, A. and Silvester, D.: Fast iterative solution of stabilised Stokes systems. Part I: Using simple diagonal preconditioners, SIAM Journal on Numerical Analysis, 30, 630–649, https://doi.org/10.1137/0730031, 1993.

Wolf, L., Huismans, R. S., Rouby, D., Gawthorpe, R. L., and Wolf, S. G.: Links between faulting, topography, and sediment production during continental rifting: Insights from coupled surface process, thermomechanical modeling, Journal of Geophysical Research: Solid Earth, 127, e2021JB023 490, https://doi.org/10.1029/2021JB023490, 2022.

Wú, Q., Lin, S., and Unger, A.: Modeling multi-material structural patterns in tectonic flow with a discontinuous Galerkin level set method, Journal of Geophysical Research: Solid Earth, 128, e2023JB026 806, https://doi.org/10.1029/2023JB026806, 2023.

Xie, S., Cao, Z., Liu, L., Yang, D., Liu, M., Li, Y., and Qi, R.: The role of plume-lithosphere interaction in Hawaii-Emperor chain formation, Nature Communications, 15, 6571, https://doi.org/10.1038/s41467-024-50888-2, 2024.

Yang, U. M.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner, Applied Numerical Mathematics, 41, 155–177, https://doi.org/10.1016/S0168-9274(01)00115-5, 2002.

Yoshida, M., Tajima, F., Honda, S., and Morishige, M.: The 3D numerical modeling of subduction dynamics: Plate stagnation and segmentation, and crustal advection in the wet mantle transition zone, Journal of Geophysical Research: Solid Earth, 117, https://doi.org/10.1029/2011JB008989, 2012.

Yu, A. J., Gün, E., McCaffrey, K. J., and Heron, P. J.: A recipe for exotic continental fragment formation: Key constraints from numerical rift models, Journal of Geophysical Research: Solid Earth, 130, e2024JB030 041, https://doi.org/10.1029/2024JB030041, 2025.

Zheng, L., Gerya, T., Knepley, M., Yuen, D. A., Zhang, H., and Shi, Y.: GPU implementation of multigrid solver for stokes equation with strongly variable viscosity, in: GPU Solutions to Multi-scale Problems in Science and Engineering, pp. 321–333, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-16405-7_21, 2013.

Zhong, S.: Analytic solutions for Stokes' flow with lateral variations in viscosity, Geophysical Journal International, 124, 18–28, https://doi.org/10.1111/j.1365-246X.1996.tb06349.x, 1996.

Zhong, S., Paulson, A., and Wahr, J.: Three-dimensional finite-element modelling of Earth's viscoelastic deformation: effects of lateral variations in lithospheric thickness, Geophysical Journal International, 155, 679–695, https://doi.org/10.1046/j.1365-246X.2003.02084.x, 2003.

https://doi.org/10.5194/egusphere-2026-2528
Preprint. Discussion started: 20 May 2026

Zhong, S., McNamara, A., Tan, E., Moresi, L., and Gurnis, M.: A benchmark study on mantle convection in a 3-D spherical shell using CitcomS, Geochemistry, Geophysics, Geosystems, 9, https://doi.org/10.1029/2008GC002048, 2008.