



High-performance geodynamics on GPUs using the PETSc CUDA backend (GAUZZ v1.0.0)

Kyeong-Min Lee¹, Deok-Kyu Jang³, Cedric Thieulot⁴, and Byung-Dal So^{1,2}

¹Interdisciplinary Program in Earth Environmental System Science & Engineering,
Kangwon National University, Chuncheon, Republic of Korea

²Department of Geophysics, Kangwon National University, Chuncheon, Republic of Korea

³Research Institute for Mathematical Sciences, Kangwon National University, Chuncheon, Republic of Korea

⁴Department of Earth Sciences, Utrecht University, Princetonlaan 8A, Utrecht, the Netherlands

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

Received: 1 May 2026 – Discussion started: 20 May 2026

Revised: 25 September 2026 – Accepted: 28 September 2026 – Published: 13 October 2026

Abstract. GPUs provide much higher memory bandwidth than CPUs. Data-parallel computations with regular memory access patterns exploit the GPU architecture efficiently. To solve the Stokes equations, we apply a preconditioned conjugate-gradient (PCG) method to the 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 the solver in GAUZZ v1.0.0 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. 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 a manufactured solution, the SolCx benchmark, the sticky-air benchmark, 2D Rayleigh–Taylor instabilities, 3D thermo-mechanical convection, and a 2D subduction model with nonlinear viscosity. We confirm that the GPU-based implementation reproduces CPU solutions. Performance is quantified by the execution time of the pressure-correction step and by throughput, defined as the number of Stokes degrees of freedom divided by the execution time of one Stokes solve. A single GPU delivers 6–11× the throughput of a 32-core CPU across the Stokes benchmarks. Two GPUs increase the throughput at sufficiently large problem sizes, reaching up to 2.1× the single-GPU throughput. The two-GPU configuration also ex-

tends the maximum problem size to 2.5×10^7 DOFs, beyond the 48 GB VRAM limit of a single GPU.

1 Introduction

Numerical geodynamic modelling has advanced the understanding of plate tectonics and planetary interior physics, including subduction (Yoshida et al., 2012; Keum and So, 2021, 2023; Seong et al., 2026), rifting (Huisman 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., 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; Lee et al., 2024). Resolving the cou-

pled, time-dependent Stokes-advection system requires repeated solutions of the Stokes operator. At each time step, the viscosity and density fields are updated. Viscosity often depends nonlinearly 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., 2001; 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 $O(N^4)$ (Chen et al., 2018). 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, 1989; 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 GM-

RES 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-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. Additional memory is required to store the Krylov basis vectors generated during each FGMRES restart cycle. In contrast, short-recurrence Krylov methods such as BiCGStab and IDR(s) maintain a fixed storage requirement independent of the iteration number. Clevenger and Heister (2021) used BiCGStab and IDR(s) to solve the full Stokes system in 3D mantle convection simulations. Applying CG to the pressure Schur complement system forms a nested iteration structure. The outer CG iteration updates the pressure, while each Schur-complement evaluation requires an inner linear solve with the velocity stiffness matrix. Despite the nested structure, the outer iteration requires only a fixed, small number of pressure- and velocity-space vectors. The inner velocity solve adds separate Krylov storage. 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; Chang and So, 2025; Jang and So, 2026), subduction (Suchov 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., 2006). The MPI-based architecture delivers strong scalability, yielding near-linear speedup on thousands of CPU cores (Burstedde et al., 2013).

A GPU integrates thousands of arithmetic units designed for massively parallel computation (Ta et al., 2015). A CPU uses a few complex cores designed for fast sequential computation. The GPU also accesses memory at a higher bandwidth than the CPU, with theoretical peaks of 960 and 153.6 GB s^{-1} for the NVIDIA RTX 6000 Ada Generation and the Xeon Gold 6338 workstation used in this study, respectively. Sparse matrix–vector products and vector updates dominate the cost of Krylov solvers (May et al., 2015). Both operations perform approximately one multiplication and one addition per value read from memory (Williams et al., 2009). The low ratio of arithmetic to memory traf-

fic, termed arithmetic intensity, makes memory bandwidth, rather than the number of arithmetic units or cores, a strong control on the cost of the two operations. Preconditioning and other solver steps also contribute to the time per Krylov iteration.

To date, GPU-based solutions of the Stokes equations in geodynamic modelling have often used pseudo-transient (PT) methods (Räss et al., 2022), which have also been extended to fluid flow and to elastic, viscoelastic, and hydromechanical problems (Wang et al., 2022; Alkhimenkov and Podladchikov, 2025). PT methods augment the governing equations with pseudo-time derivatives and integrate them explicitly until the residuals are reduced below a prescribed tolerance. In the cited Stokes implementations, the explicit update applies local finite-difference stencils on regular staggered grids, requiring no assembly of global matrices. The matrix-free formulation keeps the memory footprint small, which suits GPUs with limited device memory. The accelerated PT method adds pseudo-time terms that give rise to damped-wave dynamics (Räss et al., 2022). With near-optimal iteration parameters, the iteration count grows approximately linearly, rather than quadratically, with the number of grid cells along one dimension. Räss et al. (2022) demonstrated convergence for viscosity contrasts of up to nine orders of magnitude. Convergence slowed in some of the high-contrast cases. In contrast, GPU implementations of Krylov solvers with algebraic multigrid preconditioning, the standard approach in finite-element geodynamic codes, have received less attention.

PETSc (Portable, Extensible Toolkit for Scientific Computation; Balay et al., 2025) 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 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 3.15.5, 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, 2D Rayleigh–Taylor instabilities, 3D thermo-mechanical 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 Sect. 5.4. All experiments were performed on three machines with different performance characteristics. The first workstation used an Intel Xeon Gold 6338 processor (2.00 GHz, 48 MB cache, 32 physical cores and 64 threads). The second workstation used an Intel Xeon w5-2465X processor (3.10 GHz, 33.75 MB cache, 16 physical cores and 32 threads) and an NVIDIA RTX 6000 Ada Generation GPU with 48 GB of memory. The third workstation used the same processor but was paired with two NVIDIA RTX 6000 Ada Generation GPUs, each with 48 GB of memory (96 GB in total). The Xeon Gold 6338 supports eight DDR4-3200 memory channels, six of which are populated in the CPU workstation used here, giving a theoretical memory bandwidth of 153.6 GB s^{-1} . The theoretical bandwidth per NVIDIA RTX 6000 Ada Generation is 960 GB s^{-1} . We measured the sustained bandwidth with a memory benchmark based on the vector update $a = a + s \cdot b$, following the approach of Deakin et al. (2018). The benchmark sustained 824.0 GB s^{-1} on a single GPU (86 % of the peak) and 113.2 GB s^{-1} on the 32-core CPU (74 % of the peak) (Table S1 in the Supplement). All CPU execution times reported in this study were measured on the first workstation, with one MPI rank assigned to each physical core. The second and third workstations were used for the single-GPU and two-GPU calculations, respectively. In both GPU systems, the Xeon w5-2465X processor handled matrix assembly, mesh-related operations, and CPU–GPU data management, while the linear solves were executed on the GPU. Unlike previous GPU-based geodynamic studies, which primarily rely on PT 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 (Eqs. 1 and 2). We incorporated the thermal (Ra) and compositional (Rb)

Rayleigh numbers (Eq. 3) into the buoyancy term (e.g., van Keken et al., 1997; Tan et al., 2011). Equations (1) and (4) assume a general form, with each benchmark activating only the terms required by the physical setting. Thermal-buoyancy cases set $Rb = 0$, while compositional-buoyancy cases set $Ra = 0$ in the absence of a temperature field. The presence of a temperature field determines whether Eq. (4) enters the solution. Each calculation adopts one of the two settings alone, keeping the thermal and compositional drivers separate.

$$\nabla \cdot \boldsymbol{\tau} - \nabla p + \left[RaT - \sum_i Rb_i C_i \right] \hat{\mathbf{z}} = \mathbf{0} \quad (1)$$

where $\boldsymbol{\tau} = 2\eta \dot{\boldsymbol{\epsilon}}$ and $\dot{\boldsymbol{\epsilon}} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$

$$\nabla \cdot \mathbf{u} = 0 \quad (2)$$

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

p , $\mathbf{u}$, $\boldsymbol{\tau}$, η , $\hat{\mathbf{z}}$, and $\dot{\boldsymbol{\epsilon}}$ denote pressure, velocity vector, the deviatoric stress tensor, shear viscosity, the upward unit vector, 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 (Eq. 4).

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

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 conditions (Hillebrand et al., 2014; Wú et al., 2023). The advection of the level-set function ϕ governs the evolution of the composition field (Eq. 5). The level-set function is reinitialized to a signed-distance function satisfying $|\nabla \phi| = 1$ by solving the reinitialization equation (Eq. 6) using second-order ENO spatial discretization, a Godunov Hamiltonian, and RK2 pseudo-time integration.

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \quad (5)$$

$$\frac{\partial \phi}{\partial \tau} + S(\phi_0)(|\nabla \phi| - 1) = 0, \quad \text{where } S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + \varepsilon^2}} \quad (6)$$

S , ϕ_0 , and ε denote the smoothed sign function, the level-set function before reinitialization, and the smoothing parameter, respectively, with $\varepsilon = 1.5 \min(\frac{1}{N_x}, \frac{1}{N_y})$. N_x and N_y are the numbers of grid cells in the x and y directions, respectively. The level-set function ϕ is a continuous scalar field with the interface defined by $\phi = 0$. 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 (Eq. 7).

$$\begin{pmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{0} \end{pmatrix} \quad (7)$$

The velocity stiffness matrix $\mathbf{K}$ is symmetric and positive definite (Davies et al., 2011). $\mathbf{G}$ and $\mathbf{G}^T$ 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). $\mathbf{f}$ denotes the body force vector. In geodynamic applications, the viscosity contrast ranges 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; Eq. 8).

$$\begin{pmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \hat{\mathbf{h}} \end{pmatrix}, \quad \mathbf{S} = \mathbf{G}^T \mathbf{K}^{-1} \mathbf{G}, \quad \hat{\mathbf{h}} = \mathbf{G}^T \mathbf{K}^{-1} \mathbf{f} \quad (8)$$

To solve the equivalent system $\mathbf{G}^T \mathbf{K}^{-1} \mathbf{G} p = \mathbf{G}^T \mathbf{K}^{-1} \mathbf{f}$, we apply a PCG iteration with the inverse viscosity mass matrix as preconditioner (Eqs. 9–14; Jang et al., 2025).

Initialization:

$$\mathbf{p}^0 = \mathbf{0}, \quad \mathbf{K}\mathbf{u}^0 = \mathbf{f} - \mathbf{G}\mathbf{p}^0 \text{ (initial solve)} \quad (9)$$

$$\langle q^0, r \rangle = \langle \nabla \cdot \mathbf{u}^0, r \rangle, \quad \left\langle \frac{1}{\eta} \mathbf{w}^0, r \right\rangle = \langle q^0, r \rangle, \quad \mathbf{d}^0 = -\mathbf{w}^0 \quad (10)$$

(initial residual and precondition.)

For $k = 0, 1, 2, \dots$:

$$\mathbf{K}\mathbf{h}^k = \mathbf{G}\mathbf{d}^k, \quad \alpha^k = \frac{\langle q^k, \mathbf{w}^k \rangle}{\langle \nabla \mathbf{d}^k, \mathbf{h}^k \rangle} \text{ (inner solve and step size)} \quad (11)$$

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \alpha^k \mathbf{d}^k, \quad \mathbf{u}^{k+1} = \mathbf{u}^k - \alpha^k \mathbf{h}^k \text{ (vector update)} \quad (12)$$

$$\langle q^{k+1}, r \rangle = \langle \nabla \cdot \mathbf{u}^{k+1}, r \rangle, \quad \left\langle \frac{1}{\eta} \mathbf{w}^{k+1}, r \right\rangle = \langle q^{k+1}, r \rangle \quad (13)$$

(divergence and precondition.)

$$\beta^k = \frac{\langle q^{k+1}, \mathbf{w}^{k+1} \rangle}{\langle q^k, \mathbf{w}^k \rangle}, \quad \mathbf{d}^{k+1} = -\mathbf{w}^{k+1} + \beta^k \mathbf{d}^k \text{ (CG update)} \quad (14)$$

The pressure is updated along the conjugate direction $\mathbf{d}^k$, while the velocity is updated using the corresponding velocity-space vector $\mathbf{h}^k$. The iteration stops when the divergence error ($\|\nabla \cdot \mathbf{u}\| / \|\mathbf{u}\|$) of velocity is lower than the divergence tolerance. The algorithm (Eqs. 9–14) requires only a fixed number of vectors in the outer PCG iteration: approximately four in the pressure space ($q^k, \mathbf{d}^k, \mathbf{p}^k, \mathbf{w}^k$) and two in the velocity space ($\mathbf{h}^k, \mathbf{u}^k$). The inner velocity subproblem is solved with GMRES, requiring a separate Krylov basis. However, the symmetric positive definiteness of the velocity stiffness matrix $\mathbf{K}$ also permits CG as an alternative inner solver. CG uses a short recurrence with only a few stored vectors, offering a potential reduction in Krylov-vector memory. A further option applies a fixed number of BoomerAMG V-cycles directly as a spectrally equivalent approximation of $\mathbf{K}^{-1}$ within the outer PCG iteration, removing the inner Krylov solver. Outside the velocity subproblem solve (Eq. 11), the remaining operations include vector inner products, scalar–vector multiplications, and vector additions (Eqs. 12 and 14). Memory bandwidth limits the speed of the three operations, which map directly onto GPU architectures. The dominant cost per iteration is the AMG-preconditioned GMRES solve of the velocity subproblem $\mathbf{K}\mathbf{h}^k = \mathbf{G}\mathbf{d}^k$ (Eq. 11), 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 (Eqs. 10 and 13) also offer substantial potential for GPU acceleration. Evaluating the GPU performance of the velocity subproblem solve and the L_2 -projection is therefore essential for 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 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., 2025). Although early development of PETSc focused on MPI-based CPU parallel computing, CUDA-based GPU support was available in PETSc 3.4 (2013), whose documentation included the AIJCUSPARSE matrix type for sparse-matrix operations on NVIDIA GPUs using the cuSPARSE library. PETSc 3.9 (2018) made CUDA-based GPU backends available in the release version, rather than only in the development branch. PETSc 3.12 (2019) further refined the Vec-CUDA interface and introduced in-place vector-array access that avoids unnecessary data transfers from GPU to CPU. PETSc therefore provides GPU-resident vector and sparse-matrix operations required for efficient 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 for a finite-difference Stokes problem on a single GPU (NVIDIA Tesla C2070) with a customised GMG solver implemented in MATLAB and CUDA, compared with 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. Although GPU-side matrix assembly has already been developed (Cecka et al., 2011; Markall et al., 2013), performing CPU assembly remains a limitation of our code implemented in FEniCS 2019. 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 execution time. To minimise these bot-

tlenecks 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 3.4×10^6 DOFs and 1.3 GB for 8.9×10^6 DOFs (Table S2). 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 $\mathbf{K}\mathbf{h}^k = \mathbf{G}\mathbf{d}^k$ (Eq. 11) 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 arithmetic units.

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 selected solver configurations that provide strong per-iteration error reduction. We adopted the Falgout algorithm, a standard HYPRE BoomerAMG coarsening strategy, as the default option on the CPU (Falgout et al., 2006; Henson and 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 (Cramer et al., 2012; Hillebrand et al., 2014), 2D Rayleigh–Taylor instability tests (van Keken et al., 1997; Maljaars et al., 2021), and the thermo-mechanical 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-GPU system was assessed for memory-demanding cases, including high-resolution 3D runs approaching $100 \times 80 \times 100$ using the $P_1^+ \times P_1$.

4 Numerical results

To quantify GPU performance, we report execution time and throughput. Execution time refers to the time spent in the pressure-correction stage of one Stokes solve, including the inner GMRES iterations, vector updates, L_2 -projections, and the associated CPU and GPU overheads. Throughput is defined as the total number of Stokes degrees of freedom (velocity and pressure) divided by the execution time of one Stokes solve, in units of DOFs s^{-1} . Execution time compares hardware configurations at a fixed problem size (Figs. 1–11). Throughput normalizes the cost by problem size. Throughput therefore allows comparison across mesh resolutions, element types, and single- and multi-GPU configurations (Fig. 12). For time-dependent cases (Sects. 4.3–4.6), execution time and throughput are evaluated per time step. For the nonlinear subduction model (Sect. 4.6), the execution time of one time step includes all Picard iterations within the step. Each PCG iteration (Eqs. 11–14) updates the pressure through one cycle of the preconditioned conjugate-gradient method. Within each PCG iteration, the velocity subproblem (Eq. 11) 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 Stokes flow, the primary computational cost and memory bottleneck stem from solving the velocity-only submatrix system (Eq. 11). 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 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.

$$\nabla \cdot \boldsymbol{\tau} = \mathbf{f} \text{ where } \boldsymbol{\tau} = 2\eta \dot{\boldsymbol{\epsilon}} \text{ and } \dot{\boldsymbol{\epsilon}} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \quad (15)$$

$$\eta = \exp(cx), \quad c = \ln(10^6) \quad (16)$$

Table 1. Comparison of AMG coarsening strategies on the single-GPU and 32-core CPU.

Configuration	Coarsening	Iterations	Time to solution (s)
32-core CPU	Falgout	13	5.331
32-core CPU	HMIS	25	9.756
32-core CPU	PMIS	60	21.915
Single-GPU (1-core CPU)	HMIS	17	0.413
Single-GPU (1-core CPU)	PMIS	20	0.452

$\mathbf{u}$, η , and $\mathbf{f}$ denote the velocity vector, viscosity, and prescribed source term, respectively. Viscosity increases exponentially along the x direction (Eq. 16), yielding a viscosity contrast of 10^6 . The exact solution $\mathbf{u}_{\text{exact}}$ (Fig. S1 in the Supplement) was prescribed as

$$\mathbf{u}_{\text{exact}} = \begin{pmatrix} \sin(\pi x) \sin(\pi y) \\ \cos(\pi x) \cos(\pi y) \end{pmatrix}, \quad 0 \leq x \leq 1, \quad 0 \leq y \leq 1. \quad (17)$$

The term $\mathbf{f}$ is derived by substituting the manufactured solution and viscosity into Eq. (15).

$$\mathbf{f} = \begin{pmatrix} 2\pi \exp(cx) \sin(\pi y) (c \cos(\pi x) - \pi \sin(\pi x)) \\ -2\pi^2 \exp(cx) \cos(\pi x) \cos(\pi y) \end{pmatrix} \quad (18)$$

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, HMIS converged in 17 iterations within 0.413 s, whereas PMIS required 20 iterations within 0.452 s (Table 1). In the CPU environment, PMIS, HMIS, and Falgout required 60 (21.92 s), 25 (9.76 s), and 13 (5.33 s) GMRES iterations, respectively, to reach convergence. 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, while achieving a $13\times$ speedup in accumulated execution time (Table 2). 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) (Fig. 1). For the largest DOF case (8 396 802 DOFs), the single-GPU required 0.413 s, corresponding to an approximately $13\times$ speedup over the 32-core CPU time of 5.331 s.

4.2 SolCx problem

We used the 2D SolCx benchmark, which has been widely employed to assess the stability and convergence of GPU

Table 2. Single-GPU performance and CPU scaling.

Configuration	Iterations	Time to solution (s)
1-core CPU (Falgout)	13	52.032
2-core CPU (Falgout)	13	27.400
4-core CPU (Falgout)	13	15.175
8-core CPU (Falgout)	13	9.281
16-core CPU (Falgout)	13	6.391
32-core CPU (Falgout)	13	5.331
Single-GPU (HMIS, 1-core CPU)	17	0.413

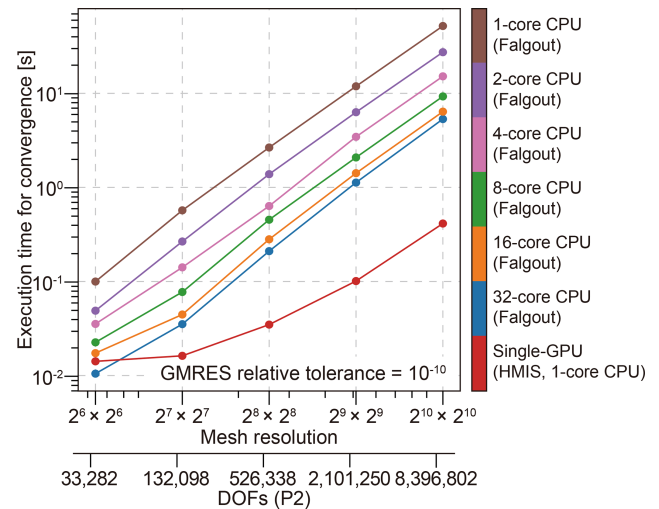


Figure 1. Inner GMRES solver execution time analysis for the velocity subproblem in single-GPU (HMIS) with 1-core CPU and multi-core CPU (Falgout) environments. Total execution time to convergence with respect to increasing mesh resolution for the GPU and CPU.

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 $x = 0.5$ (Eq. 19). We set Ra to 0 and Rb to a spatially varying field to generate buoyancy (Eq. 20). A single composition field was adopted with $C_0 = 1$.

$$\eta(x) = \begin{cases} 1, & 0 \leq x \leq 0.5, \\ 10^6, & 0.5 < x \leq 1. \end{cases} \quad (19)$$

$$Rb(x, y) = -\sin(\pi y) \cos(\pi x), \quad 0 \leq x \leq 1, \quad 0 \leq y \leq 1 \quad (20)$$

The viscosity jump of 10^6 restricts the flow to the low-viscosity half, with the pressure discontinuous across the interface (Fig. S2). We compared the performance of the GPU-accelerated finite-element model with that of the 32-core

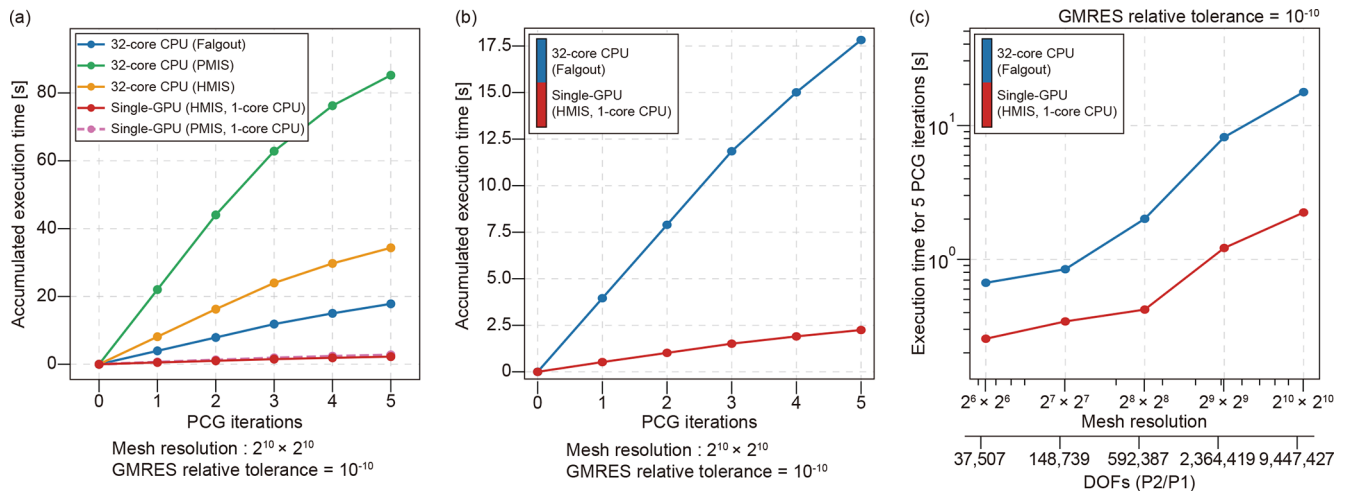


Figure 2. Execution time analysis of the SolCx benchmark in GPU and CPU environments. (a) Accumulated execution times on the GPU and 32-core CPU under different AMG preconditioner coarsening schemes (Falgout, PMIS, and HMIS). (b) Accumulated execution time on the single-GPU (HMIS) and 32-core CPU (Falgout).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

CPU. The GMRES solver employed a default relative tolerance of 10^{-10} . 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 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 32-core CPU configurations.

We found that the performance of the velocity-only subproblem translated directly into the PCG solver performance (Fig. 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 on the 32-core CPU (Fig. 2a). The GPU achieved an approximately $8\times$ speedup over the 32-core CPU, demonstrating the high efficiency of the GPU architecture. Based on the comparison in Fig. 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 32-core CPU performance comparisons in Fig. 2b and c. Then, we compared accumulated execution times for five PCG iterations on a single-GPU and 32-core CPU (Fig. 2b). The single-GPU recorded the shortest execution time at 2.25 s.

The performance evaluation with increasing problem sizes exhibited a similar trend (Fig. 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 the 32-core CPU environment. This advantage of the GPU was more pronounced for larger problems (Fig. 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 approx-

imately $7.9\times$. The single GPU reached a throughput of 4.2×10^6 DOFs s^{-1} , compared with 5.3×10^5 DOFs s^{-1} on the 32-core CPU. In the memory bandwidth benchmark (Table S1), the GPU sustains 824.0 GB s^{-1} and the 32-core CPU sustains 113.2 GB s^{-1} , a ratio of $7.3\times$. The sustained ratio exceeds the theoretical peak ratio of $6.25\times$ (960 and 153.6 GB s^{-1}). The speedups measured across the Stokes benchmarks span $6\text{--}11\times$ and cluster around the sustained ratio. The cost of PCG and GMRES is dominated by sparse matrix-vector products, data-parallel operations that map onto the GPU, where high memory bandwidth sustains the streaming access pattern. The unequal fractions of peak bandwidth attained by the GPU and CPU (86 % and 74 %, respectively) widen the sustained ratio beyond the peak ratio. The CPU and GPU versions of BoomerAMG further adopt different coarsening strategies that shift the number of iterations. The measured speedup depends on the cost per iteration and on the number of iterations. The sustained ratio serves as an approximate reference rather than an upper bound.

Figure 3a presents the accumulated execution times for the PCG solver on a single-GPU (red line) and a 32-core CPU environment (orange line). Both environments fixed the number of PCG iterations at three. Underworld2 (Mansour et al., 2020) was evaluated on a 32-core CPU using the default settings, including Q_1/dQ_0 quadrilateral elements, at a 2048×2048 resolution. Three pressure-correction iterations were used to solve the Schur-complement system. The velocity-only subsystem was solved using GMG-preconditioned GMRES. In Underworld2, the cumulative execution time over the three iterations was 7.7 s. For comparison with Underworld2, our FEniCS calculations used triangular P_1/P_0 elements at the same 2048×2048 resolution. Over three PCG iterations, the single GPU required 1.2 s, whereas the 32-core

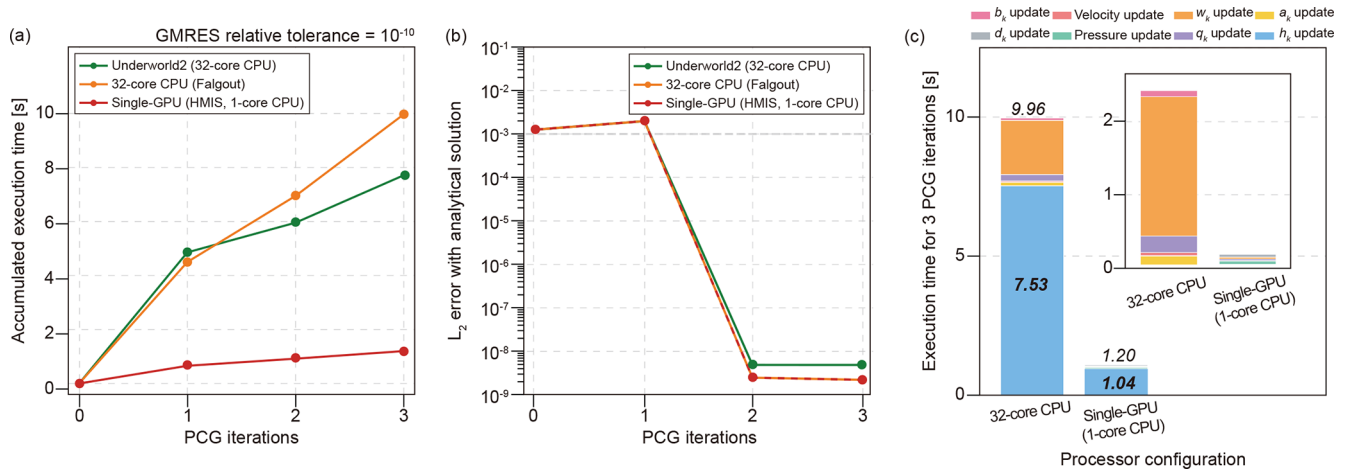


Figure 3. Performance benchmark and detailed execution time analysis of the preconditioned conjugate-gradient (PCG) iterations for the SolCx benchmark. (a) Comparison of accumulated execution time as a function of the number of PCG iterations for single-GPU (HMIS) with 1-core CPU, Underworld2 (Mansour et al., 2020; 32-core CPU), and 32-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 iterations for single-GPU (HMIS) with 1-core CPU, Underworld2 (32-core CPU), and 32-core CPU (Falgout) environments. (c) Breakdown of execution time over 3 PCG iterations for the single-GPU with 1-core CPU and 32-core CPU.

CPU required 9.96 s. The single GPU was therefore $6.42 \times$ faster than Underworld2 and $8.3 \times$ faster than the 32-core CPU implementation.

We measured the L_2 -error against the analytical solution with respect to PCG iteration number (Fig. 3b). All three configurations reduced the L_2 -error below 10^{-8} after the second pressure-correction iteration. 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 (8 388 608 DOFs) compared to the dQ_0 element used in Underworld2 (4 194 304 DOFs).

We measured execution times for individual stages of the PCG solver algorithm (solving velocity-only subsystem, vector update, and projection) to quantify differences between the 32-core CPU run and the GPU run (Fig. 3c). The single-GPU achieved an overall speedup of $8.3 \times$ relative to the 32-core CPU. The velocity-only subsystem stage dominated the total cost, with 7.53 s of the 9.96 s on the CPU and 1.04 s of the 1.20 s on the GPU, a speedup of $7.2 \times$. The speed of the vector update and projection stages is limited by memory bandwidth. The high memory bandwidth of the GPU reduced the time of the two stages from approximately 2.4 s on the CPU to 0.16 s.

We measured the execution time to convergence while decreasing the GMRES relative tolerance from 10^{-4} to 10^{-10} (Fig. 4a). The single-GPU and 32-core CPU configurations attain the minimum execution times at 10^{-5} , with execution time increasing at smaller tolerances. The single-GPU maintains the lowest execution time across the tested range, requiring 0.6–1.7 s compared with 5.4–15.2 s on the 32-core CPU. The number of outer PCG iterations decreases from

seven at 10^{-4} to two at 10^{-5} and remains at two for smaller tolerances. The 7 PCG iterations at 10^{-4} cost more execution time than the 2 iterations at 10^{-5} on both configurations. A GMRES relative tolerance smaller than 10^{-4} delivers both the lower outer PCG iteration number and the shorter execution time. The single-GPU and 32-core CPU require an identical number of outer PCG iterations at each tolerance. For GMRES relative tolerances of 10^{-5} and smaller, the outer PCG iteration count remains fixed at two. Changes in total execution time therefore reflect the computational cost per outer iteration. We also calculated the L_2 errors relative to the analytical solution over the tested tolerance range (Fig. 4b). The velocity error generally decreases as the GMRES relative tolerance is tightened, reaching the order of 10^{-11} at 10^{-8} and remaining at a similar level for tighter tolerances. Below 10^{-8} , the execution time continues to increase without further improvement in velocity accuracy. The pressure error plateaus near 10^{-3} for GMRES relative tolerances below 10^{-8} . A GMRES relative tolerance of 10^{-8} therefore provides converged accuracy without the additional computational cost associated with smaller tolerances. The outer PCG solver converged in two iterations across resolutions from 64^2 to 1024^2 on the single-GPU (Table 4) and 32-core CPU (Table 3), indicating mesh-independent convergence of the Schur-complement iteration. The lower convergence rate of the pressure L_2 error with mesh refinement reflects the pressure discontinuity in the SolCx benchmark and the use of a continuous P_1 pressure approximation.

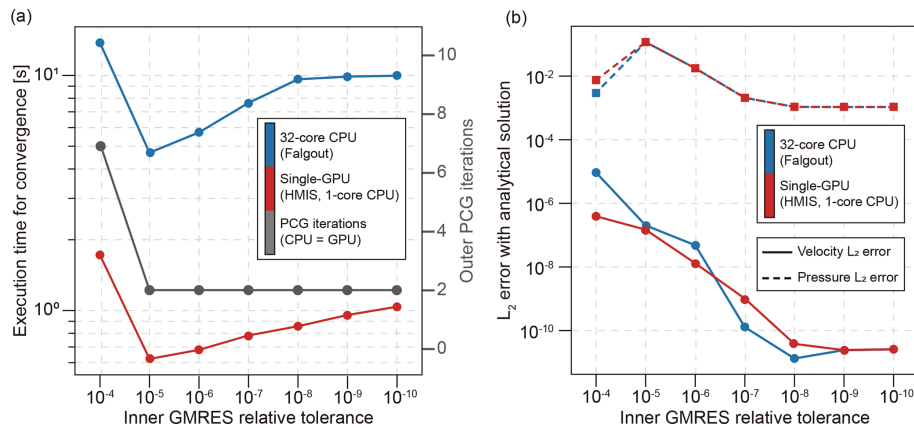


Figure 4. PCG solver performance across inner GMRES relative tolerances for the SolCx benchmark at a $2^{10} \times 2^{10}$ resolution. **(a)** Execution time to convergence on the single-GPU (HMIS, red line) and 32-core CPU (Falgout, blue line), with the number of outer PCG iterations (grey line). The number of outer PCG iterations for convergence is identical for both configurations. **(b)** L_2 errors of the velocity (solid) and pressure (dashed) solutions relative to the analytical solution for the single-GPU and 32-core CPU configurations.

Table 3. Execution time, outer PCG iteration number, and L_2 errors with increasing mesh resolution for the SolCx benchmark on a 32-core CPU. The GMRES relative tolerance and divergence tolerance are 10^{-10} and 10^{-6} , respectively.

Resolution	Stokes DOFs	Execution time (s)	PCG iteration	Velocity L_2 -error	Pressure L_2 -error
64^2	37 507	0.0679	2	3.2005×10^{-8}	4.3786×10^{-3}
128^2	148 739	0.1506	2	4.0030×10^{-9}	3.0590×10^{-3}
256^2	592 387	0.8070	2	5.0923×10^{-10}	2.1497×10^{-3}
512^2	2 364 419	4.0943	2	7.9689×10^{-11}	1.5152×10^{-3}
1024^2	9 447 427	11.0808	2	2.5796×10^{-11}	1.0696×10^{-3}

Table 4. Execution time, outer PCG iteration number, and L_2 errors with increasing mesh resolution for the SolCx benchmark on a single-GPU. The GMRES relative tolerance and divergence tolerance are 10^{-10} and 10^{-6} , respectively.

Resolution	Stokes DOFs	Execution time (s)	PCG iteration	Velocity L_2 -error	Pressure L_2 -error
64^2	37 507	0.0367	2	3.2005×10^{-8}	4.3787×10^{-3}
128^2	148 739	0.0439	2	4.0030×10^{-9}	3.0590×10^{-3}
256^2	592 387	0.1278	2	5.0928×10^{-10}	2.1498×10^{-3}
512^2	2 364 419	0.3383	2	7.9782×10^{-11}	1.5152×10^{-3}
1024^2	9 447 427	1.2556	2	2.5918×10^{-11}	1.0696×10^{-3}

4.3 3D Thermo-mechanical convection

We extended the classical 2D thermo-mechanical convection benchmark of Blankenbach et al. (1989) to a 3D non-rotating Boussinesq fluid to evaluate the performance and memory usage of the GPU solver in three dimensions. The thermal and compositional Rayleigh numbers were set to $Ra = 10^4$ and $Rb = 0$ in Eq. (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 8.9 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 $\alpha = 0.02$ to initiate natural convection (Eq. 21). 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 (Fig. 5).

$$T_0 = 1 - z + \alpha \cos(\pi x) \sin(\pi z) \quad (21)$$

We utilised a fixed time step ($\Delta t = 0.00025$) with the Crank–Nicolson scheme for time integration. We set the GMRES relative tolerance to 10^{-10} for the velocity-only subsystem on both GPU and CPU, continuing the PCG iterations until the divergence error ($\|\nabla \cdot \mathbf{u}\| / \|\mathbf{u}\|$) fell below 10^{-7} . The energy equation (Eq. 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 single-GPU (HMIS) achieved the lowest execution time with an average of 1.9 s (Fig. 6a; solid red line), compared with 17.1 s for the 32-core CPU (Falgout) (solid blue line). The speedup relative to the 32-core CPU was approximately $9\times$. At 8.87×10^6 DOFs, the average execution time per step corresponds to a throughput of 4.7×10^6 DOFs s^{-1} on the single GPU and 5.2×10^5 DOFs s^{-1} on the 32-core CPU. The average number of GMRES iterations per time step was 52 on the GPU and 23 on the 32-core CPU (Fig. 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 memory (RAM) of the workstation (Fig. 6b). The average VRAM and system memory usage per time step were 21.2 GB (dotted red line) and 17.4 GB (solid red line), respectively, for a total of 38.6 GB. In the CPU environment, the average memory usage was 46.3 GB on the 32-core CPU, approximately $1.20\times$ the total memory usage of the single-GPU configuration.

The temporal evolution of the global root-mean-square velocity (V_{rms}) exhibited complete consistency between the GPU and 32-core CPU environments (see Fig. S3). 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 both configurations, 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 (Fig. 6c). Measurements for the GPU environment included both VRAM and system memory. At the lowest DOFs (3 352 864), VRAM and system memory usage were 14.5 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 32-core CPU registered 15.3 GB at the lowest DOFs and 47.3 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., Bourgoin 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 (Eq. 22). The level-set function was initialized (Eq. 23) from the interface at $t = 0$. The composition field was set to 1 and 0 where the level-set function was positive and negative (Eq. 24), respectively. The level-set function was reinitialised every 500 steps.

$$h(x) = 0.02 \cos(\pi x / 0.9142) + 0.2 \quad (22)$$

$$\phi(x, y, t = 0) = y - h(x) \quad (23)$$

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

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 for the level-set function was solved using GMRES preconditioned by HYPRE’s BoomerAMG. The average per-step execution time was compared across GPU and CPU environments in the 2D benchmark (Fig. 7a). The GPU completed one step in 0.6 s on average (0.53 s for the Stokes equations and 0.07 s for advection), yielding speedups of $\sim 11\times$ relative to the 32-core CPU, which required 6.3 s per step. The per-step cost was dominated by solving the Stokes equations (6.2 s for the 32-core CPU). The Stokes solve at 512×512 (2.36×10^6 DOFs) reached a throughput of 4.5×10^6 DOFs s^{-1} on the single GPU and 3.8×10^5 DOFs s^{-1} on the 32-core CPU. 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 (Fig. 7b). Snapshot comparisons at key times ($t = 208, 699, 864$, and 1500), 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). We also tested the Rayleigh–Taylor instability in 3D on a $65 \times 65 \times 65$ mesh with the MINI element. The single GPU delivered a speedup of $\sim 9\times$ relative to the 32-core CPU across the entire time range, while the two configurations produced identical V_{rms} values (Fig. S4).

4.5 Sticky-air benchmarks

In geodynamic numerical modelling, the sticky-air technique is a widely used approach for approximating a free surface.

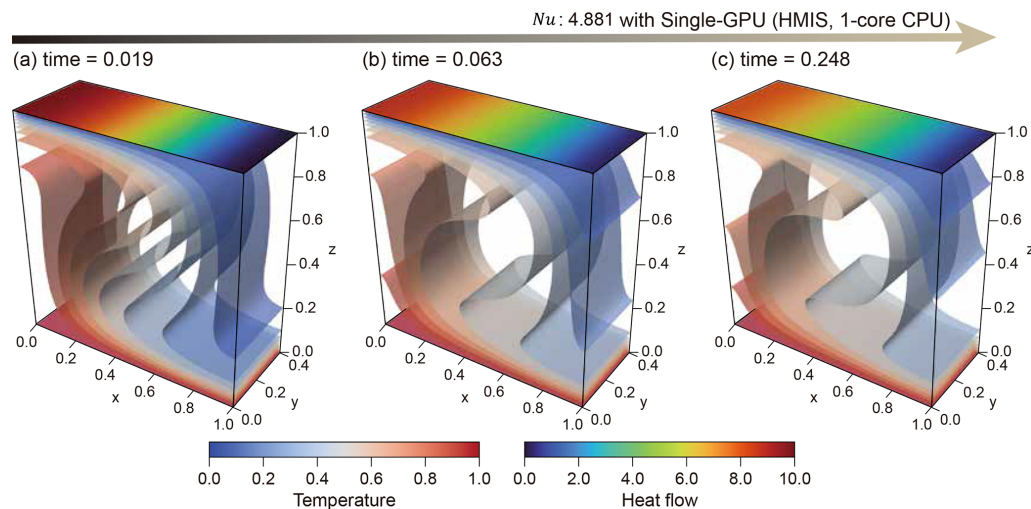


Figure 5. Time evolution of the temperature field from the single-GPU (HMIS) with 1-core CPU. Snapshots of 3D temperature isosurfaces at nondimensional times $t = 0.019, 0.063$, and 0.248 (from left to right).

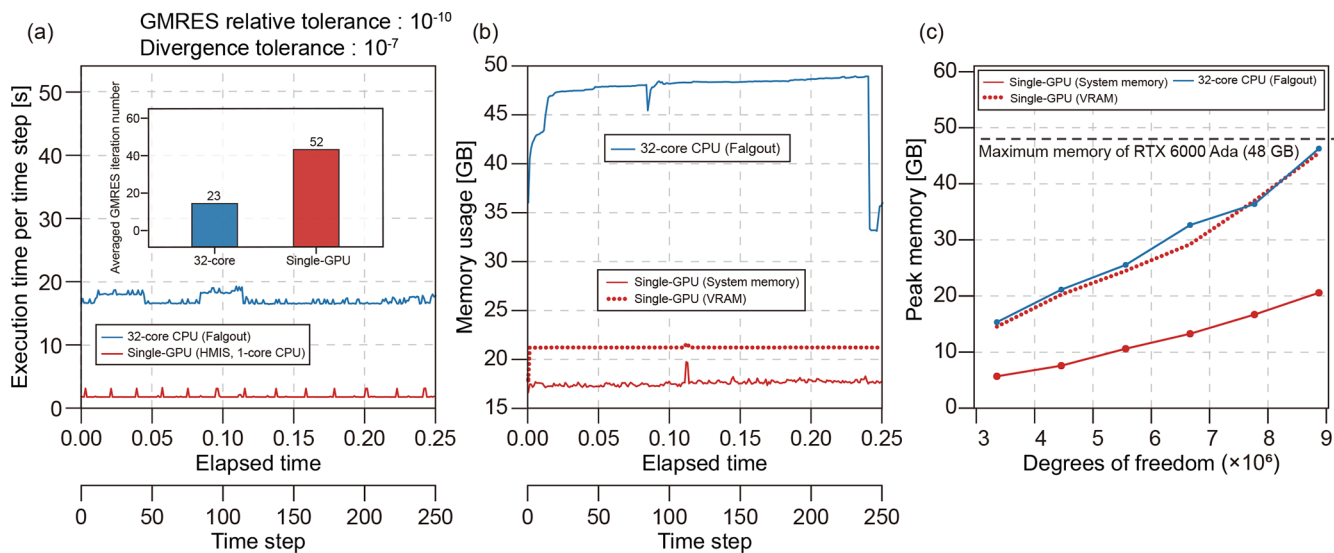


Figure 6. Performance analysis and validation of the 3D thermo-mechanical convection simulation. **(a)** Execution time per step (solid lines) and averaged GMRES iteration numbers (inset bar chart) versus elapsed time. **(b)** Evolution of memory usage over elapsed time, showing GPU VRAM (dotted red line), system memory of the GPU workstation (solid red line), and system memory of the CPU workstation (solid blue line). **(c)** Peak memory usage with respect to increasing degrees of freedom (DOFs), with the dashed black line indicating the 48 GB VRAM capacity of the NVIDIA RTX 6000 Ada.

Cramer et al. (2012) suggested that to correctly represent a free-surface boundary condition, the sticky-air layer requires a 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. Buoyancy in the sticky-air benchmark is purely compositional. The isothermal configuration of Cramer et

al. (2012) fixes $Ra = 0$ in the absence of a temperature field, leaving Eq. (4) outside the solution. The benchmark constrains the free-surface response alone, with the thermo-mechanical benchmark held separate. The dimensional parameters for the sticky-air benchmark are presented in Table S3. 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 sticky-air was assigned a cosine-shaped perturbation. We compared the maximum topography with the analytical solution from Cramer et al. (2012). The ini-

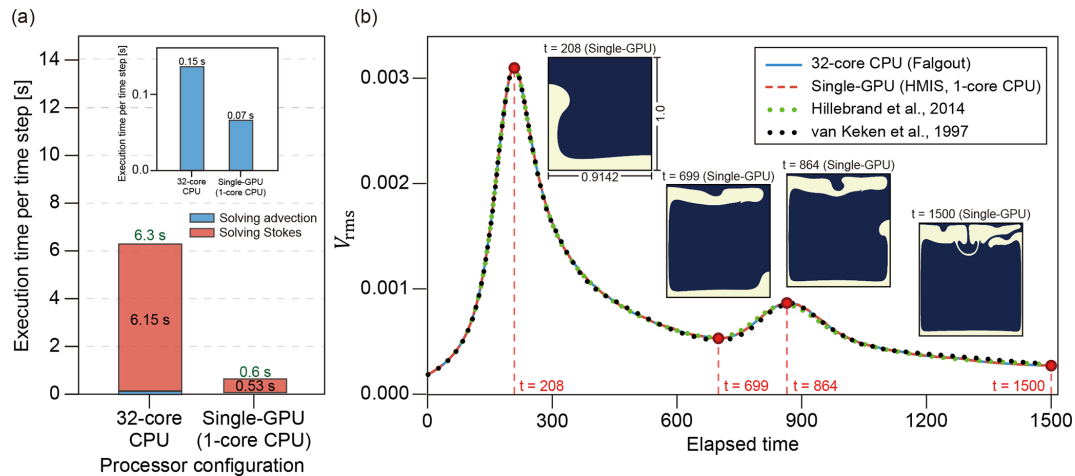


Figure 7. Performance analysis and accuracy validation of single-GPU (HMIS) with 1-core CPU and 32-core CPU (Falgout) configurations for the 2D Rayleigh–Taylor instability benchmark. **(a)** Execution time per step for solving the Stokes equations (red bars) and the advection equation (blue bars) across GPU and 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).

tial composition field (C) is defined as three layers separated by the upper and lower interfaces of the lithosphere (Eq. 25). C_1 , C_2 , and C_3 denote sticky-air, lithosphere, and mantle, respectively. Two level-set functions (ϕ_1 and ϕ_2) were defined for the two interfaces (Eqs. 26 and 27).

$$C = \begin{cases} C_1, & \phi_1(x, y, t = 0) \leq 0 \\ C_3, & \phi_2(x, y, t = 0) < 0 \\ C_2, & \text{else} \end{cases} \quad (25)$$

$$\phi_1(x, y, t = 0) = 700 + 7 \cos\left(\frac{2\pi x}{2800}\right) - y \quad (26)$$

$$\phi_2(x, y, t = 0) = y - 600 \quad (27)$$

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 (Fig. 8a). Advective transport progressively flattens the initial topographic perturbation. At 5.77 kyr, the decaying perturbation weakens the flow in the two convection cells (Fig. 8b). The surface recovers an almost flat geometry by the final time of 100 kyr (Fig. 8c). 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 (Fig. 8c). The maximum topography over the simulation time was identical between the GPU and CPU executions (Fig. 8). 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.

Execution time comparison over the total steps shows that the GPU consistently outperformed the 32-core CPU (Fig. 9a). The GPU (HMIS) execution time remained stable within 4.7–11.1 s, whereas the CPU (Falgout) execution time increased from 30 s to more than 100 s (Fig. 9a; solid blue line), corresponding to an overall speedup of roughly 7–8× for the GPU (solid red line). At 1.01×10^7 DOFs, the per-step execution time corresponds to a throughput of $0.9\text{--}2.1 \times 10^6$ DOFs s^{-1} on the single GPU and $1.0\text{--}3.4 \times 10^5$ DOFs s^{-1} on the 32-core CPU. The number of PCG iterations varied with simulation time, while the GPU and CPU generally required similar numbers of PCG iterations at each time step (Fig. 9b). 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 250–650 and 150–410 GMRES iterations, respectively.

4.6 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 S4. The Stokes system was discretized with $P_2 \times P_1$ elements on a 1200×200 structured grid, 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.

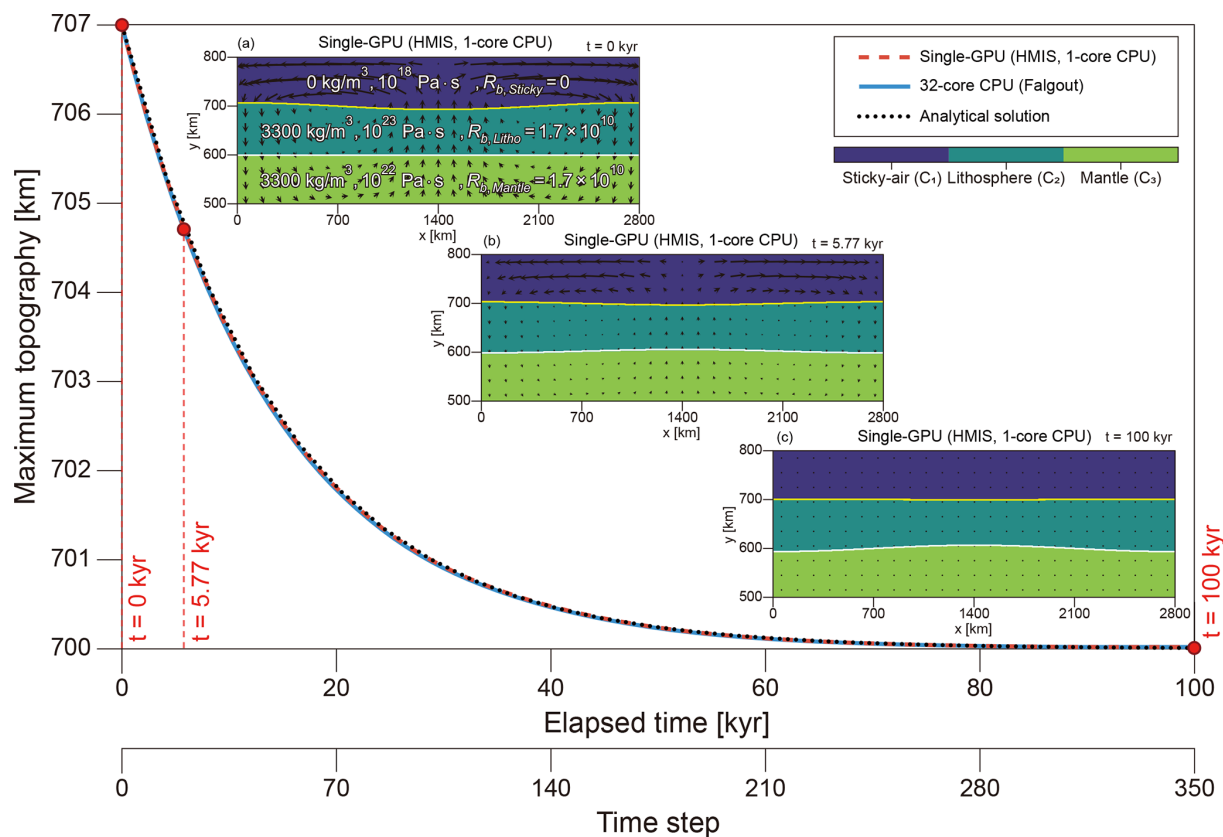


Figure 8. Evolution of maximum topography in the sticky-air benchmark. The dashed red line denotes the single-GPU (HMIS) with 1-core CPU, the solid blue line denotes the 32-core CPU computation, and the dotted black line denotes the analytical solution. (a–c) Temporal evolution of the compositional field at $t = 0$, 5.77, and 100 kyr, respectively, from the single-GPU (HMIS, 1-core CPU). The temporal evolution of maximum topography indicates close agreement between the GPU and CPU computations and with the analytical solution.

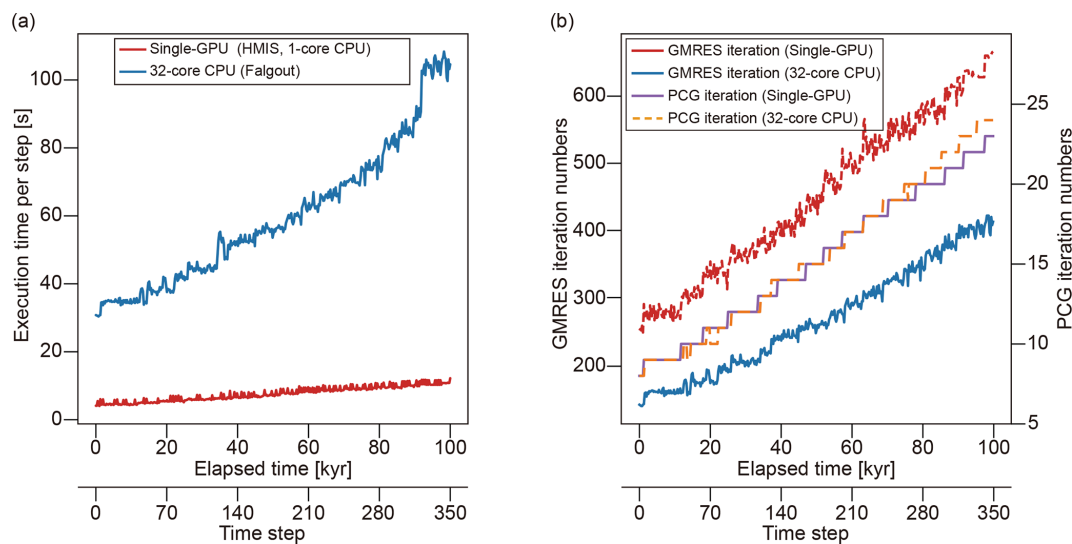


Figure 9. Performance analysis and accuracy validation of the sticky-air benchmark. (a) Execution time per step over elapsed time for the single-GPU (HMIS) with 1-core CPU and 32-core CPU (Falgout). (b) Evolution of inner GMRES iteration numbers (left axis) and preconditioned conjugate-gradient (PCG) iteration numbers (right axis).

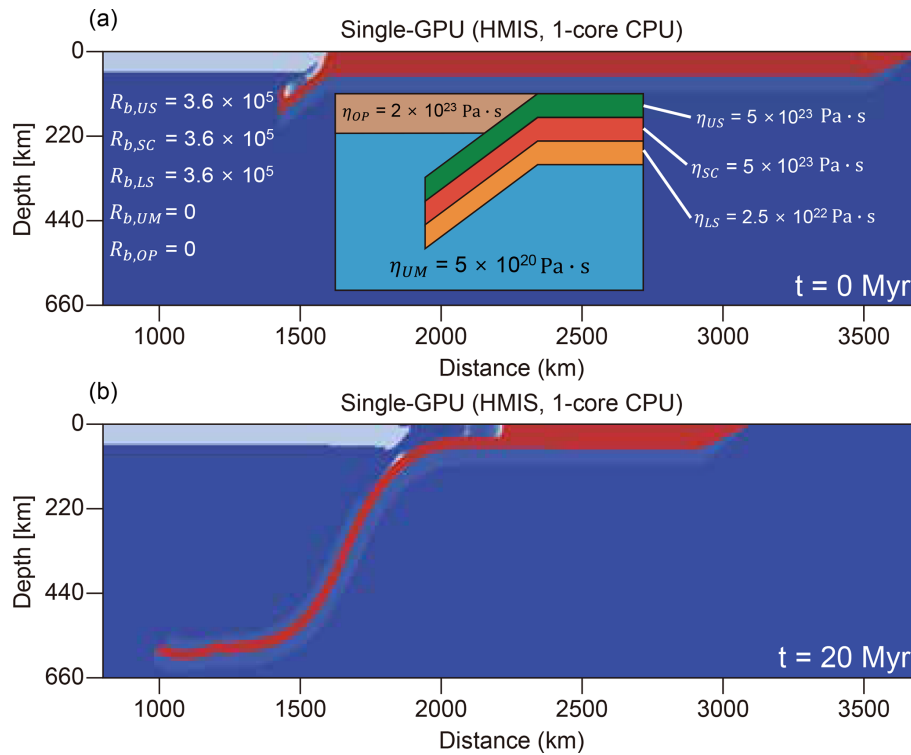


Figure 10. Temporal evolution of subduction modelling derived from the single-GPU (HMIS). **(a)** Initial model configuration at 0 Myr with the prescribed viscosity structure and buoyancy parameters. **(b)** Subduction geometry at 20 Myr after the slab descends into the upper mantle.

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 (Eq. 28).

$$\eta_{US} = \frac{1}{2} \frac{\tau_y}{\dot{\epsilon}_{II}}, \quad \dot{\epsilon}_{II} = \sqrt{\frac{1}{2} \dot{\epsilon} : \dot{\epsilon}} \quad (28)$$

τ_y , $\dot{\epsilon}$, and $\dot{\epsilon}_{II}$ 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 1000 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 number and execution time. An initial slab perturbation, pre-subducted to 200 km depth with a 29° dip, triggers

subduction (Fig. 10a). The slab maintains a steep dip and ultimately reaches the base of the upper mantle (Fig. 10b).

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 6–7 relative to the 32-core CPU run (Fig. 11a). 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 Fig. 11a). Higher condition numbers require more GMRES iterations. In the subduction model, the estimated condition number increases with time step (Fig. 11a). As the condition number increases, execution time increases in both the GPU and CPU runs (Fig. 11b). However, the regression between GMRES iterations per step and condition number shows a steeper slope for the GPU (Fig. 11c). Despite this steeper slope, the GPU remains approximately $6\times$ faster in execution time, as its lower per-iteration cost keeps the total execution time shorter. At 2.17×10^6 DOFs, the throughput ranged from 1.1×10^5 to 1.8×10^5 DOFs s^{-1} on the single GPU and from 1.4×10^4 to 3.1×10^4 DOFs s^{-1} on the 32-core CPU. The Picard iterations within each step lower the throughput by an order of magnitude relative to the linear benchmarks.

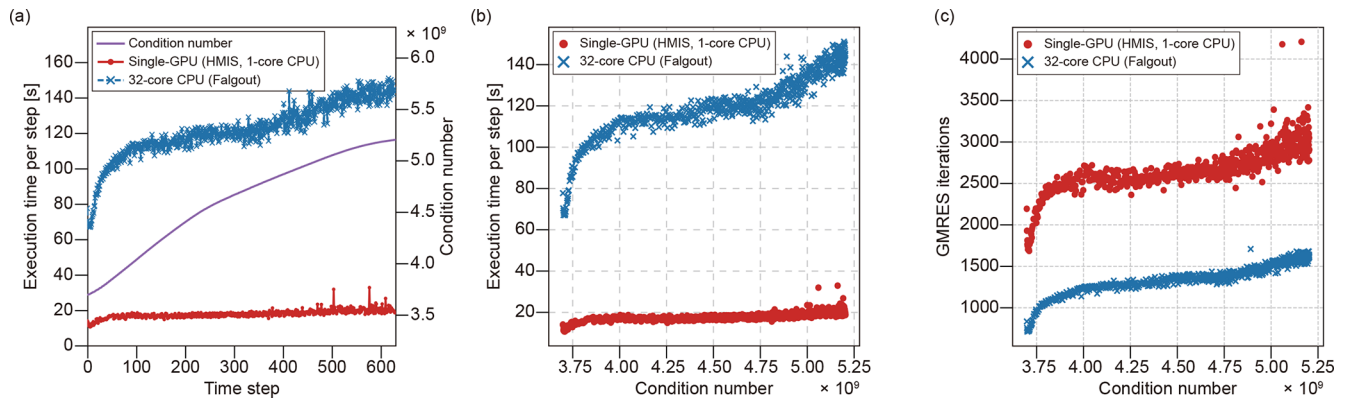


Figure 11. Analysis of GPU performance and matrix condition number for subduction modelling. **(a)** Evolution of the condition number and corresponding changes in execution time for single-GPU (HMIS) with 1-core CPU and 32-core CPU (Falgout) over time steps. **(b)** Correlation between the matrix condition number and execution time per step for the subduction model. **(c)** Correlation between the matrix condition number and inner GMRES iterations per step for the subduction model.

4.7 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 throughput varies with resolution by comparing single-GPU and two-GPU configurations across six benchmarks in Sects. 4.1–4.6. In the velocity subproblem using P_2 elements, the single-GPU delivered higher throughput than the two-GPU configuration at the lowest resolution of 128×128 (1.32×10^5 DOFs) (Fig. 12a). In the low-resolution regime, communication and synchronization overhead exceeded the performance gain from parallelization. At resolutions of 256×256 and higher, however, the two-GPU configuration delivered higher throughput 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 1152×1152 (1.06×10^7 DOFs), whereas the two-GPU configuration ran up to 1536×1536 (1.89×10^7 DOFs). We compared the throughput of single-GPU runs with two-GPU runs across mesh resolutions for the SolCx benchmark using $P_2 \times P_1$ elements (Fig. 12b). In the high-resolution regime ($> 512 \times 512$, corresponding to 2.36×10^6 DOFs), the two-GPU configuration outperformed the single-GPU by a factor of 1.03 (for 640×640) to 1.31 (for 1024×1024). The two-GPU configuration supported a maximum resolution of 1664×1664 (24 936 707 DOFs), whereas the single GPU

was limited by memory capacity to 1152×1152 (11 955 459 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 (Fig. 12c), while the y direction resolution increased from 10 to 80. The two-GPU configuration delivered higher throughput at every resolution. The throughput ratio increased from 1.07 to 1.64 as the resolution increased. The 2D Rayleigh–Taylor instability test with $P_2 \times P_1$ elements followed the same pattern as the 2D SolCx benchmark (Fig. 12d). The single-GPU delivered higher throughput at low resolution, whereas the two-GPU throughput exceeded the single-GPU throughput beyond 512×512 . In the sticky-air benchmark, the two-GPU configuration delivered higher throughput than the single-GPU at all resolutions (Fig. 12e). The throughput ratio ranged from 1.5 to 2.1. In the subduction model with nonlinear viscosity, the two-GPU throughput exceeded the single-GPU throughput beyond 900×150 . Then, the two-GPU throughput reached 1.4 times the single-GPU throughput (Fig. 12f). The single-GPU delivered higher throughput at low resolution in the SolCx, 2D Rayleigh–Taylor, and subduction benchmarks. 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 number of DOFs increases more rapidly with mesh resolution, yielding a consistent advantage for the two-GPU configuration across the tested resolutions.

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 as-

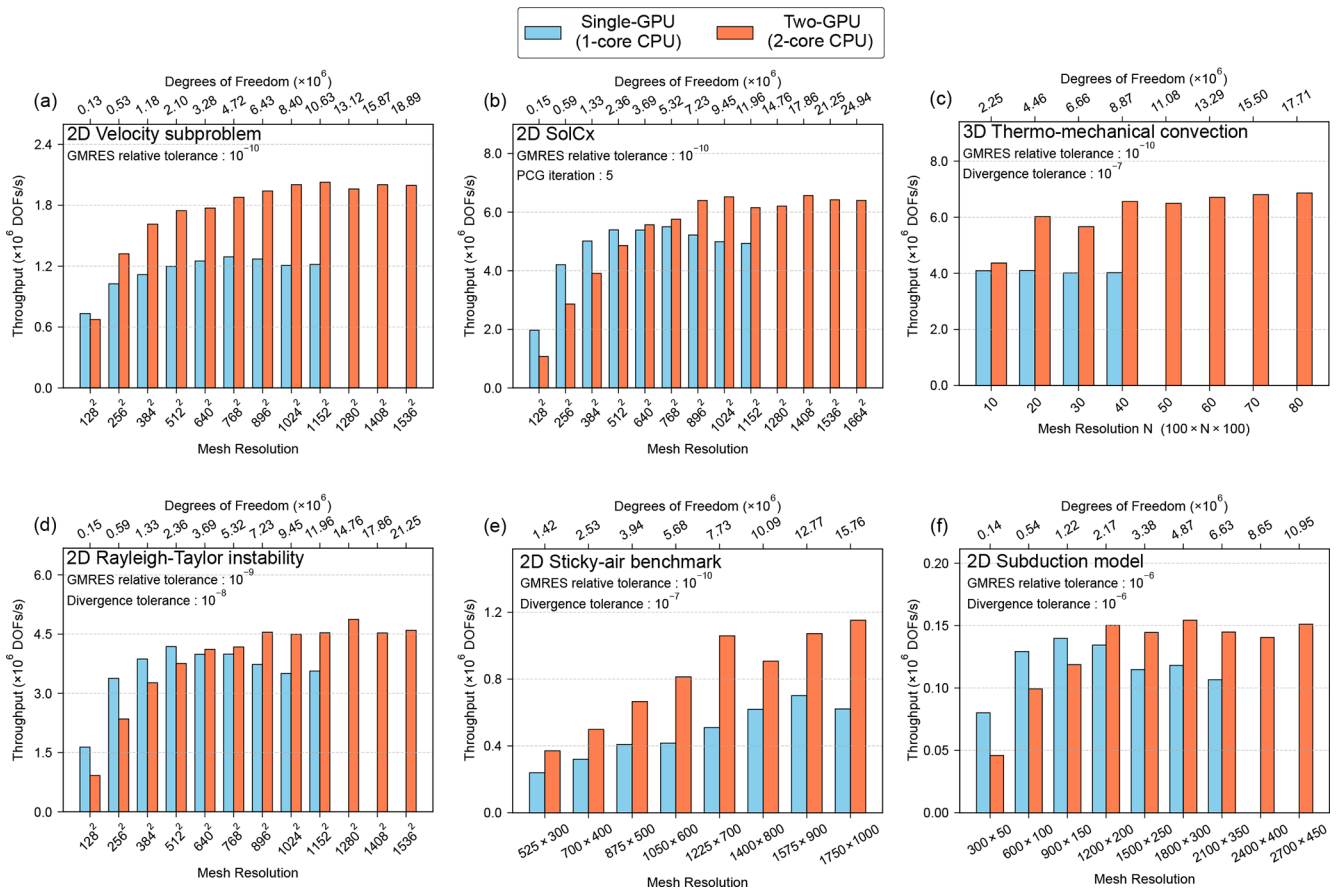


Figure 12. Performance comparison between single-GPU (1-core CPU) and two-GPU (2-core CPU) configurations in this study. (a–f) Comparison of throughput for the velocity subproblem, SolCx, 3D thermo-mechanical convection, 2D Rayleigh–Taylor instability, sticky-air benchmark, and subduction model, respectively. Horizontal axes represent mesh resolution and degrees of freedom; vertical axes represent throughput (DOFs s^{-1}).

sembles 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. Pre-assembly of the three operators removes the assembly cost from every PCG iteration. The end-to-end throughput of each time step therefore increases. 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 (Sect. 4.6). 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 (Sect. 4.3), storing these operators and associated

vectors requires approximately 1.3 GB of VRAM (Table S2), which is a modest fraction of the ~ 46 GB peak VRAM usage recorded.

Nonlinear Stokes systems with strain-rate-dependent viscosity require repeated assembly of the stiffness matrix. FEniCS 2019 performs each assembly on the CPU, adding a CPU–GPU data transfer at every nonlinear iteration. 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, the 16-core CPU with one or two GPUs achieved higher end-to-end throughput than the

1-core CPU with a single GPU. For the sticky-air benchmark at a 1400×800 resolution, the end-to-end throughput of the 16-core CPU with two GPUs was twice that of the single GPU with a 1-core CPU (Fig. S5). Across the time-dependent benchmarks, the single GPU with a 1-core CPU delivered $1.14\text{--}3.46\times$ the end-to-end throughput of the 32-core CPU. The 16-core CPU with two GPUs delivered $5.19\times$ the end-to-end throughput of the 32-core CPU (Table S5). This indicates that for 3D high-resolution problems, an MPS-based multi-core CPU and multi-GPU setup is most effective. 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.

Recently, finite element codes have incorporated GPU acceleration through global sparse matrix assembly and matrix-free evaluation. GPU-accelerated finite-element assembly has been demonstrated in FEniCS (Trotter et al., 2023). Matrix-free GPU evaluation is available in deal.II and MFEM (Kronbichler and Ljungkvist, 2019; Anderson et al., 2021). MFEM additionally assembles global sparse matrices on the GPU. 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 $\sim 5\times$ the end-to-end throughput of the 32-core CPU-only configuration with a 16-core CPU and two GPUs.

5.2 Computational efficiency of multi-GPU configurations

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, the two-GPU configuration outperformed a single-GPU configuration above 512×512 elements. Between 640×640 and 1024×1024 , the two-GPU configuration delivered $1.03\text{--}1.31\times$ the single-GPU throughput. These speedup factors enable higher resolution 2D rifting and subduction simulations under tight VRAM limits.

In 3D, the number of DOFs increases more rapidly with mesh resolution, 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 at extreme multi-GPU scale. We propose that multi-GPU scaling directly supports larger DOF targets within available VRAM. An extrapolation based on our 3D 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 solvers for the Stokes system benefit mainly from high memory bandwidth and large device memory. HPC-class GPUs such as the NVIDIA H200 provide 4.8 TB s^{-1} and 141 GB for the two quantities, approximately $5\times$ and $2.9\times$ the values of the RTX 6000 Ada, respectively. The higher bandwidth of the H200 is expected to lower the cost of operations limited by memory bandwidth. The overall gain for the Stokes solver requires direct measurement on the H200. The larger device memory increases the maximum problem size per GPU. H200-class GPUs offer a path toward higher-resolution simulations of global mantle convection and complex crustal deformation.

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) 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 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 solver using a 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. The outer iteration stores only a fixed number of vectors, requiring less memory than block-preconditioned FGMRES, which accumulates Krylov vectors as the iteration proceeds. However, the inner velocity subproblem is solved with GMRES and requires additional Krylov-vector storage. As a result, the overall memory saving is smaller than that suggested by the outer iteration alone.

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 S5). The FEniCS implementation was tested on a two-GPU configuration with a 16-core CPU, a single-GPU configuration with a 16-core CPU, and a 32-core CPU, while ASPECT was run on the 32-core CPU. The ASPECT calculation used the single-Advection, single-Stokes nonlinear solver scheme with the block-AMG Stokes solver. The Stokes linear solver tolerance was set to 10^{-2} , with a GMRES restart length of 100, while the temperature solver tolerance was 10^{-6} . To enable a fair comparison with ASPECT, which provides Q_2/Q_1 elements, we used a P_2/P_1 discretization in FEniCS with a resolution of $40 \times 40 \times 40$. The simulation covered a total nondimensional time of 0.25 over 1000 steps. On the 32-core CPU, ASPECT required an execution time of 471 s to solve the Stokes system over the full elapsed run. Our PCG solver on the 32-core CPU required an execution time of 961 s, indicating that ASPECT's block-preconditioned Stokes solver outperforms our PCG solver on CPUs in this configuration. However, with a single GPU and a 16-core CPU, the same case required 247 s, corresponding to speedups of approximately $3.89\times$ and $1.91\times$ relative to our 32-core PCG run and the 32-core ASPECT run, respectively. With two GPUs and a 16-core CPU, the Stokes solving time decreased to approximately 126 s, delivering an additional $2.0\times$ speedup over the single-GPU case and overall speedups of $7.63\times$ and $3.74\times$ 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. The velocity-only submatrix is symmetric positive definite, allowing CG to replace GMRES for the velocity subproblem. CG stores fewer Krylov vectors than GMRES, whereas the dominance of the BoomerAMG hierarchy in solver memory limits the overall memory reduction. Selecting the most effective solver requires a systematic comparison of PCG splitting and block-preconditioned FGMRES in execution time and memory usage.

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 six benchmarks: a manufactured solution, SolCx, the sticky-air benchmark, 2D Rayleigh–Taylor instability, 3D thermo-mechanical convection, and a subduction model with non-linear 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, the GPU advantage became more pronounced. The single GPU delivered 6–11× the throughput of the 32-core CPU across the Stokes benchmarks. Even when GMRES required more iterations on the GPU, the lower cost per iteration kept the GPU throughput higher. Experiments with single-GPU and two-GPU configurations of RTX 6000 Ada cards showed that the VRAM capacity of a single GPU limits the achievable resolution. Multi-GPU domain decomposition extended the problem size to 2.5×10^7 DOFs. The two-GPU configuration further increased the throughput at large problem sizes, reaching up to $2.1 \times$ the single-GPU throughput. Coordinating a 16-core CPU with two GPUs through NVIDIA MPS delivered $5.19 \times$ the end-

to-end throughput of the 32-core CPU in the 3D thermo-mechanical convection benchmark.

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 an Intel Xeon Gold 6338 processor, while the Intel Xeon w5-2465X served as the host CPU of the GPU workstations; the GPU runs were executed on NVIDIA RTX 6000 Ada (48 GB VRAM) workstations.

Supplement. The supplement related to this article is available online at <https://doi.org/10.5194/gmd-19-9781-2026-supplement>.

Author contributions. KML 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. DKJ contributed to the mathematical formulation of the preconditioned conjugate-gradient algorithm and the Schur complement reduction. CT contributed substantially to shaping the core ideas of the paper through revisions, provided expertise on the geodynamic benchmark setups, and provided major input during the manuscript editing process. BDS 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 contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims made in the text, published maps, institutional affiliations, or any other geographical representation in this paper. The authors bear the ultimate responsibility for providing appropriate place names. Views expressed in the text are those of the authors and do not necessarily reflect the views of the publisher.

Acknowledgements. We thank Topic Editor Ludovic Räss, Christian Hüttig, and an anonymous reviewer for their constructive com-

ments, which improved the manuscript. 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>, last access: 8 October 2026). Byung-Dal So was supported by the “Ocean Research Facility and Equipment sharing Program” of the Korea Institute of Marine Science & Technology Promotion (KIMST), with funding from the Ministry of Oceans and Fisheries, Republic of Korea.

Financial support. This research has been supported by the National Research Foundation of Korea (NRF) grants funded by the Korean Government (grant nos. RS-2025-02293161, RS-2022-NR069527, RS-2019-NR040076, and RS-2025-25428518). Deok-Kyu Jang was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (grant no. RS-2025-00519120).

Review statement. This paper was edited by Ludovic Räss and reviewed by Christian Hüttig and one anonymous referee.

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, *J. Geophys. Res.-Planet.*, 128, e2023JE007879, <https://doi.org/10.1029/2023JE007879>, 2023.
- Alber, D. M. and Olson, L. N.: Parallel coarse-grid selection, *Numer. Linear Algebr.*, 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 hydromechanical problems with applications, *Geosci. Model Dev.*, 18, 563–583, <https://doi.org/10.5194/gmd-18-563-2025>, 2025.
- 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: *Applied Parallel Computing. New Paradigms for HPC in Industry and Academia*, edited by: Sørveik, T., Manne, F., Gebremedhin, A. H., and Moe, R., Springer, Berlin, Heidelberg, 121–130, https://doi.org/10.1007/3-540-70734-4_16, 2001.
- Anderson, R., Andrej, J., Barker, A., Bramwell, J., Camier, J.-S., Cervený, J., Dobrev, V., Dudouit, Y., Fisher, A., Kolev, T., Pazner, W., Stowell, M., Tomov, V., Akkerman, I., Dahm, J., Medina, D., and Zampini, S.: MFEM: A modular finite element methods library, *Comput. Math. Appl.*, 81, 42–74, <https://doi.org/10.1016/j.camwa.2020.06.009>, 2021.
- Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E. M., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., 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., Suh, H., Zampini, S., Zhang, H., Zhang, H., and Zhang, J.: PETSc/TAO Users Manual, Tech. Rep. ANL-21/39 – Revision 3.23, Argonne National Laboratory, Argonne, IL, USA, <https://doi.org/10.2172/2565610>, 2025.
- Benzi, M., Golub, G. H., and Liesen, J.: Numerical solution of saddle point problems, *Acta Numer.*, 14, 1–137, <https://doi.org/10.1017/S0962492904000212>, 2005.
- Beuchert, M. J. and Podladchikov, Y. Y.: Viscoelastic mantle convection and lithospheric stresses, *Geophys. J. Int.*, 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, *Geophys. J. Int.*, 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 Geotech.*, 1, 225–236, <https://doi.org/10.1007/s11440-006-0016-6>, 2006.
- Brandt, A.: Multi-level adaptive solutions to boundary-value problems, *Math. Comput.*, 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 J. Sci. Comput.*, 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, *J. Geophys. Res.-Sol. Ea.*, 117, <https://doi.org/10.1029/2011JB008860>, 2012.
- Burstedde, C., Stadler, G., Alisic, L., Wilcox, L. C., Tan, E., Gurnis, M., and Ghattas, O.: Large-scale adaptive mantle convection simulation, *Geophys. J. Int.*, 192, 889–906, <https://doi.org/10.1093/gji/ggs070>, 2013.
- 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.2009.08.020>, 2010.
- Cecka, C., Lew, A. J., and Darve, E.: Assembly of finite element methods on graphics processors, *Int. J. Numer. Meth. Eng.*, 85, 640–669, <https://doi.org/10.1002/nme.2989>, 2011.
- Chang, J. and So, B.-D.: Partial Nonlinear Iteration During Early Continental Rifting for Symmetric Surface Topography and Heat Flux, *Ocean Sci. J.*, 60, 38, <https://doi.org/10.1007/s12601-025-00232-y>, 2025.
- Chen, C., Pouransari, H., Rajamanickam, S., Boman, E. G., and Darve, E.: A distributed-memory hierarchical solver for general sparse linear systems, *Parallel Comput.*, 74, 49–64, <https://doi.org/10.1016/j.parco.2017.12.004>, 2018.
- Clevenger, T. C. and Heister, T.: Comparison between algebraic and matrix-free geometric multigrid for a Stokes problem on adaptive meshes with variable viscosity, *Numer. Linear Algebr.*, 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, *Geophys. J. Int.*, 221, 617–639, <https://doi.org/10.1093/gji/ggaa023>, 2020.

- Cramer, F. and Tackley, P. J.: Parameters controlling dynamically self-consistent plate tectonics and single-sided subduction in global models of mantle convection, *J. Geophys. Res.-Sol. Ea.*, 120, 3680–3706, <https://doi.org/10.1002/2014JB011664>, 2015.
- Cramer, 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, *Geophys. J. Int.*, 189, 38–54, <https://doi.org/10.1111/j.1365-246X.2012.05388.x>, 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 Planet. Sc. Lett.*, 206, 379–396, [https://doi.org/10.1016/S0012-821X\(02\)01058-0](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 modelling framework for geodynamics, *Geochem. Geophys. Geosy.*, 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, *Geophys. Res. Lett.*, 13, 1517–1520, <https://doi.org/10.1029/GL013i013p01517>, 1986.
- Deakin, T., Price, J., Martineau, M., and McIntosh-Smith, S.: Evaluating attainable memory bandwidth of parallel programming models via BabelStream, *International Journal of Computational Science and Engineering*, 17, 247–262, <https://doi.org/10.1504/IJCSE.2018.095847>, 2018.
- De Sterck, H., Yang, U. M., and Heys, J. J.: Reducing complexity in parallel algebraic multigrid preconditioners, *SIAM J. Matrix Anal. A.*, 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, 229859, <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, *J. Geophys. Res.-Sol. Ea.*, 130, e2024JB029465, <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, *Geochem. Geophys. Geosy.*, 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, *Geophys. J. Int.*, 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*, edited by: Bruaset, A. M. and Tveito, A., *Lecture Notes in Computational Science and Engineering*, 51, Springer, Berlin, Heidelberg, 267–294, https://doi.org/10.1007/3-540-31619-1_8, 2006.
- Gassmüller, R., Dannberg, J., Bangerth, W., Heister, T., and Myhill, R.: On formulations of compressible mantle convection, *Geophys. J. Int.*, 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, *Phys. Earth Planet. In.*, 140, 293–318, <https://doi.org/10.1016/j.pepi.2003.09.006>, 2003.
- 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, *Geochem. Geophys. Geosy.*, 14, 1200–1225, <https://doi.org/10.1002/ggge.20078>, 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/nature15752>, 2015.
- Graham, R. L., Woodall, T. S., and Squyres, J. M.: Open MPI: A Flexible High Performance MPI, in: *Parallel Processing and Applied Mathematics*, edited by: Wyrzykowski, R., Dongarra, J., Meyer, N., and Waśniewski, J., Springer Berlin Heidelberg, Berlin, Heidelberg, 228–239, https://doi.org/10.1007/11752578_29, 2006.
- Henson, V. E. and Yang, U. M.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner, *Appl. Numer. Math.*, 41, 155–177, [https://doi.org/10.1016/S0168-9274\(01\)00115-5](https://doi.org/10.1016/S0168-9274(01)00115-5), 2002.
- Hestenes, M. R. and Stiefel, E.: Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Stand.*, 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.
- Huisman, R. S. and Beaumont, C.: Symmetric and asymmetric lithospheric extension: Relative effects of frictional-plastic and viscous strain softening, *J. Geophys. Res.-Sol. Ea.*, 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, *EGU sphere [preprint]*, <https://doi.org/10.5194/egusphere-2025-5480>, 2025.
- Jang, M.-S. and So, B.-D.: Generation of Channelized Mantle Flow Beyond the Rift Zone Accelerated by Thermal Heterogeneity and Rift Propagation, *Geochem. Geophys. Geosy.*, 27, e2025GC012743, <https://doi.org/10.1029/2025GC012743>, 2026.
- Keum, J. Y. and So, B. D.: Effect of buoyant sediment overlying subducting plates on trench geometry: 3D viscoelastic free subduction modeling, *Geophys. Res. Lett.*, 48, e2021GL093498, <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, 229970, <https://doi.org/10.1016/j.tecto.2023.229970>, 2023.
- King, S. D.: On topography and geoid from 2-D stagnant lid convection calculations, *Geochem. Geophys. Geosy.*, 10, <https://doi.org/10.1029/2008GC002250>, 2009.
- Kronbichler, M. and Ljungkvist, K.: Multigrid for Matrix-Free High-Order Finite Element Computations on Graphics Process-

- sors, *ACM Transactions on Parallel Computing*, 6, 2:1–2:32, <https://doi.org/10.1145/3322813>, 2019.
- Kronbichler, M., Heister, T., and Bangerth, W.: High accuracy mantle convection simulation through modern numerical methods, *Geophys. J. Int.*, 191, 12–29, <https://doi.org/10.1111/j.1365-246X.2012.05609.x>, 2012.
- Laik, A., Schellart, W. P., and Strak, V.: Sustained indentation in 2-D models of continental collision involving whole mantle subduction, *Geophys. J. Int.*, 232, 343–365, <https://doi.org/10.1093/gji/ggac339>, 2023.
- Lee, K.-M., Kim, H.-I., Jang, D.-K., and So, B.-D.: A Comparative Study of 2D Numerical Simulations Using Vector and Potential Methods for Extending 3D Modeling of Planetary Evolution, *Geosci. J.*, 28, 465–481, <https://doi.org/10.1007/s12303-024-0013-5>, 2024.
- Lee, K.-M., Jang, D.-K., Thieulot, C., and So, B.-D.: Codes and Dockerfile for geodynamics simulations using PETSc with GPUs (GAUZZ 1.0.0), Zenodo [code], <https://doi.org/10.5281/zenodo.19936267>, 2026 (code also available at: <https://github.com/kmlee-geo/gauzz>, last access: 8 October 2026).
- Leng, W. and Zhong, S.: Viscous heating, adiabatic heating and energetic consistency in compressible mantle convection, *Geophys. J. Int.*, 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-Sci. Rev.*, 244, 104518, <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 Res.*, 28, 1329–1343, <https://doi.org/10.1016/j.gr.2014.11.004>, 2015.
- MacLachlan, S. P. and Oosterlee, C. W.: Local Fourier analysis for multigrid with overlapping smoothers applied to systems of PDEs, *Numer. Linear Algebr.*, 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, *Comput. Math. Appl.*, 81, 289–315, <https://doi.org/10.1016/j.camwa.2020.04.023>, 2021.
- 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.
- Markall, G. R., Slemmer, A., Ham, D. A., Kelly, P. H. J., Cantwell, C. D., and Sherwin, S. J.: Finite Element Assembly Strategies on MultiCore and Many-Core Architectures, *Int. J. Numer. Meth. Fl.*, 71, 80–97, <https://doi.org/10.1002/fld.3648>, 2013.
- May, D. A. and Moresi, L.: Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, *Phys. Earth Planet. In.*, 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, *Comput. Method. Appl. M.*, 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 Comput.*, 108, 102831, <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, *Phys. Earth Planet. In.*, 97, 83–94, [https://doi.org/10.1016/0031-9201\(96\)03163-9](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, *J. Geophys. Res.-Planet.*, 129, e2023JE008143, <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, *J. Appl. Fluid Mech.*, 9, 1285–1296, <https://doi.org/10.18869/acadpub.jafm.68.228.22805>, 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, *Geochem. Geophys. Geosy.*, 22, e2020GC009615, <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*, edited by: Sherwin, S. J., Moxey, D., Peiró, J., Vincent, P. E., and Schwab, C., Lecture Notes in Computational Science and Engineering, 134, Springer, Cham, 263–272, https://doi.org/10.1007/978-3-030-39647-3_20, 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*, 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, *Geophys. J. Int.*, 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, *Geosci. Model Dev.*, 15, 5757–5786, <https://doi.org/10.5194/gmd-15-5757-2022>, 2022.
- Regorda, A., Thieulot, C., van Zelst, I., Erdős, Z., Maia, J., and Buiter, S.: Rifting Venus: Insights from numerical modeling, *J. Geophys. Res.-Planet.*, 128, e2022JE007588, <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, *Geophys. J. Int.*, 236, 119–138, <https://doi.org/10.1093/gji/ggad414>, 2024.
- Rudi, J., Stadler, G., and Ghattas, O.: Weighted BFBT preconditioner for Stokes flow problems with highly heterogeneous viscosity, *SIAM J. Sci. Comput.*, 39, S272–S297, <https://doi.org/10.1137/16M108450X>, 2017.
- Saad, Y. and Schultz, M. H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.*, 7, 856–869, <https://doi.org/10.1137/0907058>, 1986.

- Schenk, O., Gärtner, K., Fichtner, W., and Stricker, A.: PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation, *Future Gener. Comp. Sy.*, 18, 69–78, [https://doi.org/10.1016/S0167-739X\(00\)00076-5](https://doi.org/10.1016/S0167-739X(00)00076-5), 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 Planet. Sc. Lett.*, 633, 118643, <https://doi.org/10.1016/j.epsl.2024.118643>, 2024.
- Seong, M., Kim, H.-S., Lee, K.-M., Kim, J., and So, B.-D.: Numerical Modeling of Subduction Zone Thermal Structure with Firedrake and Gmsh, *Ocean Sci. J.*, 61, 43, <https://doi.org/10.1007/s12601-026-00292-8>, 2026.
- Sharples, W., Jadamec, M. A., Moresi, L. N., and Capitanio, F. A.: Overriding plate controls on subduction evolution, *J. Geophys. Res.-Sol. Ea.*, 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 J. Numer. Anal.*, 31, 1352–1367, <https://doi.org/10.1137/0731070>, 1994.
- Suchow, 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, *Comput. Geosci.*, 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, *Geochem. Geophys. Geosy.*, 4, <https://doi.org/10.1029/2001GC000214>, 2003.
- Taiwo, A., Bunge, H. P., Schubert, B. S. A., Colli, L., and Vilacis, B.: Robust global mantle flow trajectories and their validation via dynamic topography histories, *Geophys. J. Int.*, 234, 2160–2179, <https://doi.org/10.1093/gji/ggad188>, 2023.
- Tan, E. and Gurnis, M.: Compressible thermochemical convection and application to lower mantle structures, *J. Geophys. Res.-Sol. Ea.*, 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, *Geochem. Geophys. Geosy.*, 12, <https://doi.org/10.1029/2011GC003665>, 2011.
- Thakur, R., Rabenseifner, R., and Gropp, W.: Optimization of collective communication operations in MPICH, *Int. J. High Perform. C.*, 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, *Phys. Earth Planet. In.*, 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.
- Thieulot, C. and Bangerth, W.: On the choice of finite element for applications in geodynamics – Part 2: 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 Comput.*, 118, 103051, <https://doi.org/10.1016/j.parco.2023.103051>, 2023.
- Uzawa, H.: Iterative methods for concave programming, in: Preference, Production and Capital: Selected Papers of Hirofumi Uzawa, Cambridge University Press, Cambridge, 135–148, <https://doi.org/10.1017/CBO9780511664496.011>, 1989.
- van Keken, P. E., King, S. D., Schmeling, H., Christensen, U. R., Neumeister, D., and Doin, M. P.: A comparison of methods for the modeling of thermochemical convection, *J. Geophys. Res.-Sol. Ea.*, 102, 22477–22495, <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, *Geophys. J. Int.*, 229, 1–20, <https://doi.org/10.1093/gji/ggab426>, 2022.
- Wang, Y. and Li, M.: The interaction between mantle plumes and lithosphere and its surface expressions: 3-D numerical modelling, *Geophys. J. Int.*, 225, 906–925, <https://doi.org/10.1093/gji/ggab014>, 2021.
- Wathen, A. and Silvester, D.: Fast iterative solution of stabilised Stokes systems. Part I: Using simple diagonal preconditioners, *SIAM J. Numer. Anal.*, 30, 630–649, <https://doi.org/10.1137/0730031>, 1993.
- Williams, S., Waterman, A., and Patterson, D.: Roofline: an insightful visual performance model for multicore architectures, *Commun. ACM*, 52, 65–76, <https://doi.org/10.1145/1498765.1498785>, 2009.
- 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, *J. Geophys. Res.-Sol. Ea.*, 127, e2021JB023490, <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, *J. Geophys. Res.-Sol. Ea.*, 128, e2023JB026806, <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, *Nat. Commun.*, 15, 6571, <https://doi.org/10.1038/s41467-024-51055-9>, 2024.
- 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, *J. Geophys. Res.-Sol. Ea.*, 117, <https://doi.org/10.1029/2011JB008989>, 2012.
- Yu, A. J., Gün, E., McCaffrey, K. J. W., and Heron, P. J.: A recipe for exotic continental fragment formation: Key constraints

- from numerical rift models, *J. Geophys. Res.-Sol. Ea.*, 130, e2024JB030041, <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*, edited by: Yuen, D. A., Wang, L., Chi, X., Johnsson, L., Ge, W., and Shi, Y., *Lecture Notes in Earth System Sciences*, Springer, Berlin, Heidelberg, 321–333, https://doi.org/10.1007/978-3-642-16405-7_21, 2013.
- Zhong, S.: Analytic solutions for Stokes' flow with lateral variations in viscosity, *Geophys. J. Int.*, 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, *Geophys. J. Int.*, 155, 679–695, <https://doi.org/10.1046/j.1365-246X.2003.02084.x>, 2003.
- 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, *Geochem. Geophys. Geosy.*, 9, <https://doi.org/10.1029/2008GC002048>, 2008.