HAL open science

Assessing the robustness and scalability of the accelerated pseudo-transient method

Ludovic Räss, Ivan Utkin, Thibault Duretz, Samuel Omlin, Yuri Y Podladchikov

To cite this version:

Ludovic Räss, Ivan Utkin, Thibault Duretz, Samuel Omlin, Yuri Y Podladchikov. Assessing the robustness and scalability of the accelerated pseudo-transient method. Geoscientific Model Development, 2022, 15 (14), pp.5757 - 5786. 10.5194/gmd-15-5757-2022 . insu-03752043

HAL Id: insu-03752043

https://insu.hal.science/insu-03752043

Submitted on 16 Aug 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Geosci. Model Dev., 15, 5757–5786, 2022
https://doi.org/10.5194/gmd-15-5757-2022
© Author(s) 2022. This work is distributed under
the Creative Commons Attribution 4.0 License.

Geoscientific Model Development
Open Access
EGU

Development and technical paper

Assessing the robustness and scalability of the accelerated pseudo-transient method

Ludovic Räss[1,2,★], **Ivan Utkin**[1,2,3,★], **Thibault Duretz**[4,5], **Samuel Omlin**[6], **and Yuri Y. Podladchikov**[3,7,8]

[1]Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Zurich, Switzerland
[2]Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland
[3]Faculty of Mechanics and Mathematics, Lomonosov Moscow State University, Moscow, Russia
[4]Institut für Geowissenschaften, Goethe-Universität Frankfurt, Frankfurt, Germany
[5]Univ. Rennes, CNRS, Géosciences Rennes UMR 6118, 35000 Rennes, France
[6]Swiss National Supercomputing Centre (CSCS), ETH Zurich, Lugano, Switzerland
[7]Institute of Earth Sciences, University of Lausanne, Lausanne, Switzerland
[8]Swiss Geocomputing Centre, University of Lausanne, Lausanne, Switzerland
★These authors contributed equally to this work.

Correspondence: Ludovic Räss (luraess@ethz.ch)

Received: 10 December 2021 – Discussion started: 13 January 2022
Revised: 28 June 2022 – Accepted: 28 June 2022 – Published: 25 July 2022

Abstract. The development of highly efficient, robust and scalable numerical algorithms lags behind the rapid increase in massive parallelism of modern hardware. We address this challenge with the accelerated pseudo-transient (PT) iterative method and present a physically motivated derivation. We analytically determine optimal iteration parameters for a variety of basic physical processes and confirm the validity of theoretical predictions with numerical experiments. We provide an efficient numerical implementation of PT solvers on graphical processing units (GPUs) using the Julia language. We achieve a parallel efficiency of more than 96 % on 2197 GPUs in distributed-memory parallelisation weak-scaling benchmarks. The 2197 GPUs allow for unprecedented tera-scale solutions of 3D variable viscosity Stokes flow on 4995^3 grid cells involving over 1.2 trillion degrees of freedom (DoFs). We verify the robustness of the method by handling contrasts up to 9 orders of magnitude in material parameters such as viscosity and arbitrary distribution of viscous inclusions for different flow configurations. Moreover, we show that this method is well suited to tackle strongly nonlinear problems such as shear-banding in a visco-elasto-plastic medium. A GPU-based implementation can outperform direct-iterative solvers based on central processing units (CPUs) in terms of wall time, even at relatively low spatial resolution. We additionally motivate the accessibility of the method by its conciseness, flexibility, physically motivated derivation and ease of implementation. This solution strategy thus has a great potential for future high-performance computing (HPC) applications, and for paving the road to exascale in the geosciences and beyond.

1 Introduction

The recent development of multi-core devices has lead to the democratisation of parallel computing. Since the "memory wall" in the early 2000s (Wulf and McKee, 1995), the continuous increase in the ratio between computation speed and memory access speed results in a shift from compute-bound to memory-bound algorithms. Currently, multi-core devices such as graphical processing units (GPUs) feature floating-point arithmetic processing rates that outperform memory access rates by more than 1 order of magnitude. As a result, memory accesses constitute the main performance limiter in a majority of scientific computing applications, with arithmetic efficiency becoming much less performance relevant.

The current computing landscape challenges scientific computing applications looking at solutions to partial differential equations (PDEs) and their legacy implementations that rely on non-local methods, one example being matrix

factorisation-based solvers. The main reasons for these applications not performing optimally on modern hardware are that their prohibitive and nonlinear memory utilisation increase as a function of numbers of degrees of freedom (DoFs), proportional to the global problem size or the spatial numerical resolution. As a result, the usage of sparse direct solvers in high-performance computing (HPC) is only possible for relatively small-scale problems due to excessive memory and computational resource requirements, inherently responsible for limitations in parallel scalability. Even storing the sparse matrix structure and nonzero elements in a compressed form is often not possible due to the limited amount of available memory. This situation naturally increases the attraction of iterative matrix-free algorithms for solving large-scale problems.

Pseudo-transient (PT) or dynamic relaxation (DR) methods have seen a regain in development over the last decades. The PT methods are matrix-free and build on a transient physics analogy to establish a stationary solution. Unlike Krylov-type methods such as the conjugate gradient or gradient or generalised minimal residual (GMRES) methods, PT methods build on a fixed-point iteration, in which the update of each grid point is entirely local and does not require global reductions (and thus global communication) at each step of the algorithm. Given the locality of the algorithm, software implementations can achieve very high per-node performance and near-ideal scaling on distributed-memory systems with accelerators such as GPUs. For Krylov-type methods, some work has been done to limit global communication (Hoemmen, 2010) and reduce the number of global reductions for Gram–Schmidt and GMRES solvers (Swirydowicz et al., 2020). Together with optimal preconditioning and educated guesses for the initial Krylov vector, these approaches could reduce the number of iterations required. Nevertheless, even a few global reductions may still limit scalability of the approach at extreme scales, and limiting communication in the Krylov part may not circumvent the requirement of ideal preconditioners.

The PT methods build on a physical description of a process. It therefore becomes possible to model strongly nonlinear processes and achieve convergence starting from nearly arbitrary initial conditions. Conventional linearisation methods such as the Newton–Raphson method may fail to converge if the initial approximation is not close enough to the solution. Examples include problems of resolving strain localisation owing to plastic yielding (Duretz et al., 2019a) or non-Newtonian power-law rheology, as well as nonlinearities arising from multi-physics coupling such as shear heating (Duretz et al., 2019b; Räss et al., 2020) and two-phase flow localisation (Räss et al., 2018, 2019a).

The implementation conciseness constitutes another advantage of PT methods compared to matrix-based solvers. PT algorithms are concise and short as the explicit pseudo-time integration preserves similarity to the mathematical description of the system of PDEs. Conciseness supports efficient and thus faster development and significantly simplifies the addition of new physics, a crucial step when investigating multi-physics couplings. Also, the similarity between mathematical and discretised code notation makes PT methods an attractive tool for research and education.

The PT method originated as a dynamic-relaxation method in the 1960s, i.e. when it was applied for calculating the stresses and displacements in concrete pressure vessels (Otter, 1965). It builds on pioneering work by Richardson (1911) proposing an iterative solution approach to PDEs related to dam-engineering calculations. In geosciences, the PT method was introduced by Cundall (1976) as the Fast Lagrangian Analysis of Continua (FLAC) algorithm. Subsequently, the FLAC method was successfully applied to simulate the Rayleigh–Taylor instability in visco-elastic flow (Poliakov et al., 1993), as well as the formation of shear bands in rocks (Poliakov et al., 1994). Other applications of the PT method are structural analysis problems including failure (Kilic and Madenci, 2009), buckling (Ramesh and Krishnamoorthy, 1993) and form-finding (Barnes, 1999). The DR terminology is still referenced in the finite-element method (FEM) community (Rezaiee-Pajand et al., 2011).

Interestingly, Richardson developed his iterative approach without being aware of the work by Gauss and Seidel, their method being named the Liebmann method when applied to solving PDEs. Early development of iterative algorithms such as 1D projection methods and Richardson iterations depend on the current iterate only. They were well-suited for early low-memory computers, however lacking in efficient convergence rates. The situation changed in 1950, when Frankel introduced second-order iterations as an extension of the Richardson and Liebmann methods, adding dependency on the previous iterate (Frankel, 1950), resulting in the second-order Richardson and extrapolated Liebmann methods, respectively. These methods feature enhanced convergence rates (Young, 1972), and perform on par, the first being slightly more interesting as fully local (Riley, 1954). By analogy with the explicit solution to time-dependent PDEs, Frankel introduced additional "physically motivated" terms in his iterative scheme. Since the Chebyshev iteration can be recovered for constant parameters, second-order or extrapolated methods are also termed semi-iterative. Note that one challenge related to Chebyshev's semi-iterative methods relies on the need for an accurate estimate of extremal eigenvalues relating to the interval in which the residual is minimised. The review by Saad (2020) provides further interesting developmental insights.

The accelerated PT method for elliptic equations is mathematically equivalent to the second-order Richardson rule (Frankel, 1950; Riley, 1954; Otter et al., 1966). The convergence rate of PT methods is very sensitive to the iteration parameters' choice. For the simplest problems, e.g. the stationary heat conduction in a rectangular domain described by the Laplace's equation, these parameters can be derived analytically based on the analysis of the damped wave equa-

tion (Cox and Zuazua, 1994). In the general case, the values of these parameters are associated with the maximum eigenvalue of the stiffness matrix. The eigenvalue problem is computationally intensive, and for practical purposes the eigenvalues are often approximated based on the Rayleigh's quotient or Gershgorin's theorem (Papadrakakis, 1981). Thus, the effective application of PT methods relies on an efficient method to determine the iteration parameters. In the last decades, several improvements were made to the stability and convergence rate of DR methods (Cassell and Hobbs, 1976; Rezaiee-Pajand et al., 2011; Alamatian, 2012). Determining the general and efficient procedure for estimating the iteration parameters still remains an active area of research.

We identify three important challenges for iterative methods among current ones, namely (1) ensure the iteration count to scale linearly with numerical resolution increase, possibly independent of material parameters' contrasts and nonlinearities, (2) achieve minimal per-device main memory access redundancy at maximal access speed, and (3) achieve a parallel efficiency close to 100 % on multi-device – distributed-memory – systems. In this study, we address (1) by presenting the accelerated PT method and resolving several types of basic physical processes. We consider (2) and (3) as challenges partly related to scientific software design and engineering; we address them using the emerging Julia language (Bezanson et al., 2017), which solves the "two-language problem" and provides the missing tool for making prototype and production code become one and breaking up the technically imposed hard division of the software stack into domain science tasks (higher levels of the stack) and computer science tasks (lower levels of the stack). The Julia applications featured in this study rely on recent Julia package developments undertaken by the authors to empower domain scientists to write architecture-agnostic high-level code for parallel high-performance stencil computations on massively parallel hardware such as latest GPU-accelerated supercomputers.

In this work, we present the results of analytical analysis of the PT equations for (non-)linear diffusion and incompressible visco-elastic Stokes flow problems. We motivate our selection of particular physical processes as a broad range of natural processes categorise mathematically either as diffusive, wave-like or mechanical processes, and thus constitute the main building blocks of multi-physics applications. We derive iteration parameters' approximations from continuous, non-discretised formulations with emphasis on an analogy between these parameters and non-dimensional numbers arising from mathematical modelling of physical processes. Such a physics-inspired numerical optimisation approach has the advantage of providing a framework building on solid classical knowledge and for which various analytical approaches exist to derive or optimise parameters of interest. We assess the algorithmic and implementation performance and scalability of the 2D and 3D numerical Julia (multi-)GPU (non-)linear diffusion and visco-elastic Stokes flow implementations. We report scalability beyond terascale number of DoFs on up to 2197 Nvidia Tesla P100 GPUs on the *Piz Daint* supercomputer at the Swiss National Supercomputing Centre (CSCS). We demonstrate the versatility and the robustness of our approach in handling nonlinear problems by applying the accelerated PT method to resolve spontaneous strain localisation in elasto-viscoplastic (E-VP) media in 2D and 3D, and comparing time to solution with direct-sparse solvers in 2D. We further demonstrate the convergence of the method to be mostly insensitive to arbitrary distributions of viscous inclusions with viscosity contrasts of up to 9 orders of magnitude in the incompressible viscous Stokes flow limit.

The latest versions of the open-source Julia codes used in this study are available from GitHub within the PTsolvers organisation at https://github.com/PTsolvers/PseudoTransientDiffusion.jl (last access: 16 May 2022) and https://github.com/PTsolvers/PseudoTransientStokes.jl (last access: 16 May 2022). Past and future versions of the software are available from a permanent DOI repository (Zenodo) at: https://doi.org/10.5281/zenodo.6553699 (Räss and Utkin, 2022a) and https://doi.org/10.5281/zenodo.6553714 (Räss and Utkin, 2022b). The `README` files provide the instructions to start reproducing majority of the presented results.

2 The pseudo-transient (PT) method

At the core of the PT method lies the idea of considering stationary processes, often described by elliptic PDEs, as the limit of some transient processes described by parabolic or hyperbolic PDEs.

The PT methods were present in literature since the 1950s (Frankel, 1950) and have a long history. However, the equations describing processes under consideration are usually analysed in discretised form with little physical motivation. We here provide examples of PT iterative strategies relying on physical processes as a starting point, both for diffusion and incompressible visco-elastic Stokes problems. We further discuss how the choice of transient physical processes influences the performance of iterative methods and how to select optimal iteration parameters upon analysing the equations in their continuous form.

In the following, we make two assumptions:

1. The computational domain is a cube $x_k \in [0, L], k = 1 \ldots n_\mathrm{d}$, where n_d is the number of spatial dimensions.

2. This domain is discretised with a uniform grid of cells. The number of grid cells is the same in each spatial dimension and is equal to n_x.

However, in practice, this solution strategy is not restricted to cubic meshes with similar resolution in each dimension.

2.1 Diffusion

Let us first consider the diffusion process:

$$\rho \frac{\partial H}{\partial t} = -\nabla_k q_k, \tag{1}$$

$$q_i = -D \nabla_i H, \quad i = 1 \ldots n_\mathrm{d}, \tag{2}$$

where H is some quantity, D is the diffusion coefficient, ρ is a proportionality coefficient and t is the physical time.

By substituting Eq. (2) into Eq. (1) we obtain an equation for H:

$$\rho \frac{\partial H}{\partial t} = \nabla_k (D \nabla_k H), \tag{3}$$

where the case of $D = \mathrm{const}$ is the standard parabolic heat equation. Equation (3) must be supplemented with initial conditions at $t = 0$ and two boundary conditions for each spatial dimension at $x_k = 0$ and $x_k = L$. Here we assume that Dirichlet boundary conditions are specified. The choice of the type of boundary condition affects only the values of the optimal iteration parameters and does not limit the generality of the method.

Firstly, we consider a stationary diffusion process, which is described by Eq. (3) with $\partial H / \partial t \to 0$:

$$\nabla_k (D \nabla_k H) = 0. \tag{4}$$

Solving Eq. (4) numerically using conventional numerical methods would require assembling a coefficient matrix and relying on a direct or iterative sparse solver. Such an approach may be preferred for 1D and some 2D problems, but since our aim is large-scale 3D modelling, we are interested in matrix-free iterative methods. In the following section, we describe two such methods, both of which are based on transient physics.

2.1.1 The first-order PT method

The solution to Eq. (4) is achieved as a limit of the solution to the transient Eq. (3) at $t \to \infty$. Therefore, the natural iteration strategy is to integrate the system numerically in time until convergence, i.e. until changes in H, defined in some metric, are smaller than a predefined tolerance.

The simplest PT method is to replace physical time t in Eq. (3) with numerical pseudo-time τ, and the physical parameter ρ with a numerical parameter $\tilde{\rho}$:

$$\tilde{\rho} \frac{\partial H}{\partial \tau} = \nabla_k (D \nabla_k H). \tag{5}$$

We refer to τ as the "pseudo-time" because we are not interested in the distributions of H at particular values of τ; therefore, τ is relevant only for numerical purposes. The numerical parameter $\tilde{\rho}$ can be chosen arbitrarily.

The number of iterations, i.e. the number of steps in pseudo-time required to reach convergence of the simplest method described by Eq. (5), is proportional to n_x^2 (see Sect. A1 in the Appendix). Quadratic scaling makes the use of the simplest PT method impractical for large problems.

One possible solution to circumvent the poor scaling properties of this first-order method would be to employ an unconditionally stable pseudo-time integration scheme. However, that would require solving systems of linear equations, making the solution cost of one iteration equal to the cost of solving the original steady-state problem. We are thus interested in a method that is not significantly more computationally expensive than the first-order scheme, but that offers an improved scalability.

2.1.2 The accelerated PT method

One of the known extensions to the classical model of diffusion incorporates inertial terms in the flux definition (Chester, 1963). This addition makes it possible to describe wave propagation in otherwise diffusive processes. Those inertial terms are usually neglected because the time of wave propagation and relaxation is small compared to the characteristic time of the process (Maxwell, 1867). The modified definition of the diffusive flux, originally derived by Maxwell from the kinetic theory of ideal gas, takes the following form:

$$\theta_\mathrm{r} \frac{\partial q_i}{\partial \tau} + q_i = -D \nabla_i H, \tag{6}$$

where θ_r is the relaxation time.

A notable difference between the flux definition from Eqs. (6) and (2) is that the resulting system type switches from parabolic to hyperbolic and describes not only diffusion, but wave propagation phenomena as well. Combining Eq. (1), replacing t with τ, and Eq. (6) to eliminate q yields

$$\tilde{\rho} \theta_\mathrm{r} \frac{\partial^2 H}{\partial \tau^2} + \tilde{\rho} \frac{\partial H}{\partial \tau} = \nabla_k (D \nabla_k H), \tag{7}$$

which is a damped wave equation for $D = \mathrm{const}$ that frequently occurs in various branches of physics (Pascal, 1986; Jordan and Puri, 1999). Contrary to the parabolic Eq. (5), the information signal in the damped wave equation propagates at finite speed $V_\mathrm{p} = \sqrt{D / \tilde{\rho} / \theta_\mathrm{r}}$.

Equation (7) includes two numerical parameters, $\tilde{\rho}$ and θ_r. The choice of these parameters significantly influences the performance and the stability of the PT method. Converting Eq. (7) to a non-dimensional form allows the reduction of the number of free parameters to only one non-dimensional quantity ($Re = \tilde{\rho} V_\mathrm{p} L / D$), which can be interpreted as a Reynolds number.

Another restriction on the values of iteration parameters arises from the conditions for the numerical stability of the explicit time integration. The numerical pseudo-time step $\Delta\tau$ is related to the wave speed V_p via the following stability condition:

$$\Delta \tau \leq \frac{C}{V_\mathrm{p}} \Delta x, \tag{8}$$

where $\Delta x = L/n_x$ is the spatial grid step and C is a non-dimensional number determined for the linearised problem using a von Neumann stability analysis procedure. For the damped wave equation, Eq. (7) here considered $C \approx 1/\sqrt{n_\mathrm{d}}$, where n_d is the number of spatial dimensions (Alkhimenkov et al., 2021a).

We choose parameters $\tilde{\rho}$ and θ_r so that the stability condition (Eq. 8) is satisfied for an arbitrary $\Delta\tau$. We introduce the numerical velocity $\widetilde{V} = \widetilde{C}\Delta x/\Delta\tau$, where $\widetilde{C} \leq C$ is an empirically determined parameter. We conclude from numerical experiments that using $\widetilde{C} \approx 0.95C$ is usually sufficient for stable convergence, however, for significantly nonlinear problems, lower values of $\widetilde{C}$ may be specified. Expressions for $\tilde{\rho}$ and θ_r are obtained by taking into account the definition of Re and solving for $V_\mathrm{p} = \widetilde{V}$:

$$\tilde{\rho} = Re\frac{D}{\widetilde{V}L}, \tag{9}$$

$$\theta_\mathrm{r} = \frac{D}{\tilde{\rho}\widetilde{V}^2} = \frac{L}{Re\widetilde{V}}. \tag{10}$$

Depending on the value of the parameter Re, the PT process described by the damped wave equation (Eq. 7) will be more or less diffusive. In case of $Re \to \infty$, diffusion dominates, resulting in the accelerated PT method to be equivalent to the first-order method described in the Sect. 2.1.1, regaining the non-desired quadratic scaling of the convergence rate. If, instead, $Re \to 0$, the system is equivalent to the undamped wave equation, resulting in a never converging method, because waves do not attenuate. An optimal value of Re exists between these two limits, which leads to the fastest convergence.

To estimate the optimal value of Re, we analyse the spectral properties of Eq. (7). The solution to the damped wave equation is decomposed into a superposition of plane waves with particular amplitude, frequency and decay rate. Substituting a plane wave solution into the equation yields the dispersion relation connecting the decay rate of the wave to its frequency and values of Re. Considering the solutions to this dispersion relation, it is possible to determine the optimal value of Re, denoted here as Re_opt. For near-optimal values of Re, the number of iterations required for the method to converge exhibits linear instead of quadratic dependence on the numerical grid resolution n_x, which is a substantial improvement compared to the first-order PT method.

We present detailed explanations and derivations of the dispersion analysis of different problems in the Appendix A, leading to the optimal value of Re:

$$Re_\mathrm{opt} = 2\pi. \tag{11}$$

We quantify the convergence rate by the number of iterations n_iter required to reduce the maximal deviation of the solution to the PT equation from the true solution to the corresponding stationary problem by a factor of e (base of natural logarithm), divided by the number of grid cells n_x. Results of the dispersion analysis for the 1D stationary diffusion problem show $n_\mathrm{iter} \approx 0.3n_x$ given optimal values of Re (Fig. 1a). We estimate that the residual reduction by 14 orders of magnitude requires only $\sim 10n_x$ iterations.

For simplicity we only consider the case $D = \mathrm{const}$ in the dispersion analysis. In this case, both $\tilde{\rho}$ and θ_r are constant. If the physical properties vary in space, i.e. $D = D(x_k)$, the iteration parameters $\tilde{\rho}$ and θ_r are no longer constant and must be locally defined by the value corresponding to each grid point. If the distribution of D is smooth, this approximation works well in practice, and the number of iterations is close to the theoretically predicted value. However, particular care is needed when the distribution of D is discontinuous in space to avoid significantly reduced values of $\widetilde{C}$ to be required for convergence, ultimately leading to a much higher number of PT iterations. We found that taking a local maximum of D between neighbouring grid cells in the definitions of iteration parameters, Eqs. (9) and (10), is sufficient to ensure optimal convergence. The per-grid point local maximum selection thus effectively acts as a preconditioning technique.

For nonlinear and/or complex flow problems, the corresponding optimal values of iteration parameters such as numerical Reynolds number Re may be determined by systematic numerical experiments. In practice, optimal values for iteration parameters do not differ significantly from theoretical predictions derived for the linear case (see Sect. 2.1.3). Most importantly, the linear scaling of the method is still preserved. Also, the accelerated PT method admits explicit numerical integration, and can be implemented with minimal modifications of the simplest PT method.

2.1.3 Diffusion–reaction

The next example addresses stationary diffusion processes coupled with reaction. Here, we assume that reaction is described by the first-order kinetics law:

$$\tilde{\rho}\frac{\partial H}{\partial \tau} = -\nabla_k q_k - \rho\frac{H - H_\mathrm{eq}}{\theta_\mathrm{k}}, \tag{12}$$

where H_eq is the value of H at equilibrium, and θ_k is a characteristic time of reaction. The flux q in Eq. (12) is governed by either Eq. (2) or (6). The addition of a source term does not change the type of PDE involved in the method formulation, and the iteration strategy based on the discretisation of Eq. (2) still exhibits a quadratic scaling. We therefore focus only on the analysis of the accelerated PT method.

Equations (12) and (6) reduce to the following equation governing the evolution of H:

$$\tilde{\rho}\theta_\mathrm{r}\frac{\partial^2 H}{\partial \tau^2} + \left(\rho\frac{\theta_\mathrm{r}}{\theta_\mathrm{k}} + \tilde{\rho}\right)\frac{\partial H}{\partial \tau} = \nabla_k(D\nabla_k H) - \rho\frac{H - H_\mathrm{eq}}{\theta_\mathrm{k}}. \tag{13}$$

The Eq. (13) differs from the damped wave equation (Eq. 7) in that it includes the source term and additional phys-

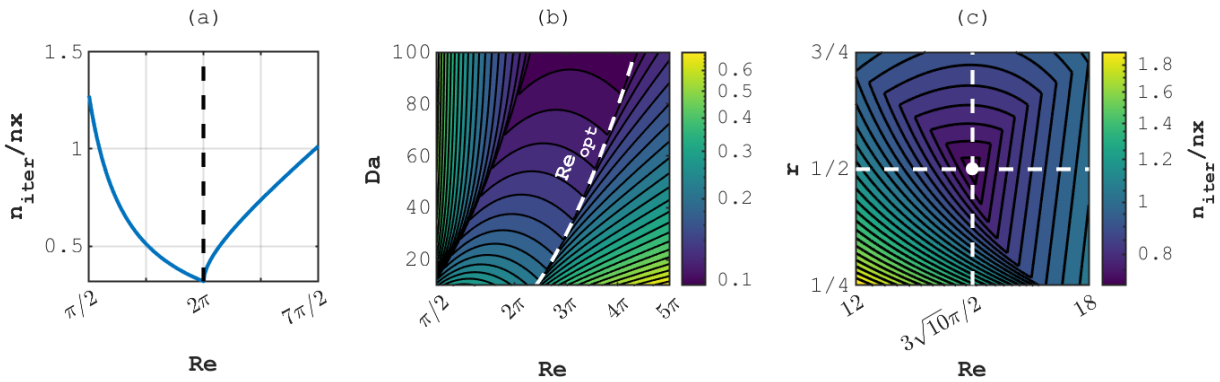

Figure 1. Number of iterations per grid point required for e-fold residual reduction. Panels **(a)**, **(b)** and **(c)** correspond to stationary diffusion, stationary diffusion–reaction and incompressible 2D Stokes problems, respectively.

ical parameters θ_k and ρ. In the non-dimensional form, all parameters of Eq. (13) can be reduced to two non-dimensional numbers: Re, defined equivalently to the stationary diffusion case, and $Da = \rho L^2/D/\theta_k$, a new parameter, which can be interpreted as a Damköhler number characterising the ratio of characteristic diffusion time to reaction timescale. Contrary to the numerical parameter Re, Da depends only on the physical parameters and cannot be arbitrarily specified.

We present the detailed dispersion analysis for the stationary diffusion–reaction problem in Sect. A3 of the Appendix. Parameters $\tilde{\rho}$ and θ_r are defined according to Eqs. (9) and (10), respectively, and by analogy to the stationary diffusion case. Optimal values of Re now depend on the parameter Da:

$$Re_{opt} = \pi + \sqrt{\pi^2 + Da} = \pi + \sqrt{\pi^2 + \frac{\rho L^2}{D\theta_k}}. \tag{14}$$

We report the result of the dispersion analysis for the diffusion–reaction case as the number of iterations required for an e-fold residual reduction n_{iter} per grid point n_x as a function of Re and Da, highlighting Re_{opt} as a function of Da by a dashed line (Fig. 1b). In the limit $Da \to 0$, i.e. when the characteristic time of reaction is infinitely large compared to the characteristic time of diffusion, $Re_{opt} \to 2\pi$, which is the optimal value for the stationary diffusion problem discussed in Sect. 2.1.2. In that limit, the number of iterations required for an e-fold residual reduction n_{iter} is also equivalent to the stationary diffusion problem. However, as Da increases, the process becomes progressively more reaction-dominated and the PT iterations converge accordingly faster.

2.1.4 Transient diffusion

It is possible to apply the PT method not only to the solution of stationary problems, but also to problems including physical transient terms. This method is known in the literature as the "dual-time", or "dual time stepping" method (Gaitonde, 1998; Mandal et al., 2011).

According to the dual-time method, both physical and pseudo-time derivatives are present in the equation:

$$\tilde{\rho}\frac{\partial H}{\partial \tau} = -\nabla_k q_k - \rho\frac{\partial H}{\partial t}. \tag{15}$$

The discretisation of the physical time derivative $\partial H/\partial t$ in Eq. (15) using a first-order backward Euler scheme, leads to

$$\tilde{\rho}\frac{\partial H}{\partial \tau} = -\frac{\partial q}{\partial x} - \rho\frac{H - \widehat{H}}{\Delta t}, \tag{16}$$

where $\widehat{H}$ is the distribution of H at the explicit layer of the integration scheme, and Δt is the physical time step. Comparing Eqs. (16) and (12) shows the two equations to be mathematically identical. Therefore, the optimal iteration parameters given by Eq. (A15) apply to the transient diffusion as well. The Da parameter thus equals $\rho L^2/D/\Delta t$ and characterises the fraction of the domain traversed by a particle transported by a diffusive flux during time Δt. The optimal value of Re is then defined as

$$Re_{opt} = \pi + \sqrt{\pi^2 + \frac{\rho L^2}{D\Delta t}}. \tag{17}$$

Frequently, modelling of certain processes requires relatively small time steps in order to capture important physical features, e.g. shear-heating induced strain localisation (Duretz et al., 2019a) or spontaneous flow localisation in porous media (Räss et al., 2019a). In such cases, values of Da can be very large. Also, every step of numerical simulation serves as a good initial approximation to the next simulation step, thereby reducing error amplitude E_1 in Eq. (A16).

2.2 Incompressible viscous shear-driven Couette flow

Before considering incompressible Stokes equations, we present an illustrative example of shear-driven flow to demonstrate a similarity between already discussed cases addressing generalised diffusion and viscous fluid flow.

Here we consider stationary fluid flow between two parallel plates separated by a distance L. We assume the absence of pressure gradients in directions parallel to the plates. In that case, Stokes equations are reduced to the following system:

$$0 = \nabla_k \tau_{xk}, \tag{18}$$

$$\frac{\tau_{xi}}{\mu_\mathrm{s}} = \nabla_i v_x, \quad i, k \in \{y, z\}, \tag{19}$$

where τ_{xi} is the deviatoric shear stress, v_x is the velocity parallel to the plates, and μ_s is the shear viscosity.

The steady-state process described by Eqs. (18) and (19) can be converted to a PT process similar to the one presented in Sect. 2.1.2, by considering the inertial term in the momentum equation (Eq. 18) and Maxwell visco-elastic rheology as a constitutive relation for the viscous fluid Eq. (19):

$$\tilde{\rho}\frac{\partial v_x}{\partial \tau} = \nabla_k \tau_{xk}, \tag{20}$$

$$\frac{1}{\widetilde{G}}\frac{\partial \tau_{xi}}{\partial \tau} + \frac{\tau_{xi}}{\mu_\mathrm{s}} = \nabla_i v_x. \tag{21}$$

Here, $\tilde{\rho}$ and $\widetilde{G}$ are numerical parameters, interpreted as density and elastic shear modulus, respectively. The system of equations (Eqs. 20 and 21) is mathematically equivalent to the system of equations (Eqs. 1 and 6), describing PT diffusion of the velocity field v_x. The relaxation time θ_r in that case represents the Maxwell relaxation time, and is equal to $\mu_\mathrm{s}/\widetilde{G}$.

2.3 Incompressible viscous Stokes equation

The next example addresses the incompressible creeping flow of a viscous fluid, described by Stokes equations:

$$0 = \nabla_j \left(\tau_{ij} - p\delta_{ij}\right) + f_i, \tag{22}$$

$$0 = \nabla_k v_k, \tag{23}$$

$$\frac{\tau_{ij}}{2\mu_\mathrm{s}} = \frac{1}{2}\left(\nabla_i v_j + \nabla_j v_i\right) = \dot{\varepsilon}_{ij}, \tag{24}$$

where τ_{ij} is the deviatoric stress, p is the pressure, δ_{ij} is the Kronecker delta, f_i is the body forces, v is the velocity, and $\dot{\varepsilon}_{ij}$ is the deviatoric strain rate.

Similar to the shear-driven flow described in Sect. 2.2, a solution to the system (Eqs. 22–24) can be achieved by PT time integration described by

$$\tilde{\rho}\frac{\partial v_i}{\partial \tau} = \nabla_j \left(\tau_{ij} - p\delta_{ij}\right) + f_i, \tag{25}$$

$$\frac{1}{\widetilde{K}}\frac{\partial p}{\partial \tau} = -\nabla_k v_k, \tag{26}$$

$$\frac{1}{2\widetilde{G}}\frac{\partial \tau_{ij}}{\partial \tau} + \frac{\tau_{ij}}{2\mu_\mathrm{s}} = \frac{1}{2}\left(\nabla_i v_j + \nabla_j v_i\right). \tag{27}$$

Equations (25) and (26) now both include pseudo-time derivatives of velocity and pressure, and become an inertial and acoustic approximation to the momentum and mass balance equations, respectively. The additional parameter $\widetilde{K}$ arising in Eq. (26) can be interpreted as a numerical or pseudo-bulk modulus.

We use the primary, or P-wave velocity, as a characteristic velocity scale for the Stokes problem:

$$V_\mathrm{p} = \sqrt{\frac{\widetilde{K} + 2\widetilde{G}}{\tilde{\rho}}}.$$

In addition to the non-dimensional numerical Reynolds number, here defined as $Re = \tilde{\rho} V_p L/\mu_\mathrm{s}$, we introduce the ratio between the bulk and shear elastic modulus $r = \widetilde{K}/\widetilde{G}$.

By analogy to previous cases, substituting $V_\mathrm{p} = \widetilde{V}$ and solving the numerical parameters $\tilde{\rho}$, $\widetilde{G}$ and $\widetilde{K}$ yields

$$\tilde{\rho} = Re\frac{\mu_\mathrm{s}}{\widetilde{V}L}, \tag{28}$$

$$\widetilde{G} = \frac{\tilde{\rho}\,\widetilde{V}^2}{r+2}, \tag{29}$$

$$\widetilde{K} = r\,\widetilde{G}. \tag{30}$$

Similar to the diffusion–reaction problem studied in Sect. 2.1.3, there are two numerical parameters controlling the process. However, in Stokes equations, both parameters are purely numerical and could be tuned to achieve the optimal convergence rate.

The dispersion analysis for 1D linear Stokes equations is detailed in Sect. A4 of the Appendix. We provide the resulting optimal values to converge the 2D Stokes problem:

$$Re_\mathrm{opt} = \frac{3\sqrt{10}}{2}\pi, \tag{31}$$

$$r_\mathrm{opt} = \frac{1}{2}, \tag{32}$$

because they differ from the 1D case values, and because we consider 2D and 3D Stokes formulation in the remaining of this study.

In the numerical experiments, we consistently observe faster convergence with slightly higher values of $r \approx 1$, likely caused by the fact that some of the assumptions made for 1D dispersion analysis do not transfer to the 2D formulation (Fig. 1c). Thus, the values presented in Eqs. (31)–(32) should only be regarded as an estimate of optimal iteration parameters.

2.4 Incompressible visco-elastic Stokes equation

The last example addresses the incompressible Stokes equations accounting for a physical visco-elastic Maxwell rheology:

$$\frac{1}{2G}\frac{\partial \tau_{ij}}{\partial t} + \frac{\tau_{ij}}{2\mu_\mathrm{s}} = \frac{1}{2}\left(\nabla_i v_j + \nabla_j v_i\right), \tag{33}$$

where G is the physical shear modulus.

As in the transient diffusion case presented in Sect. 2.1.4, the problem can be augmented by PT time integration using a dual time stepping approach:

$$\frac{1}{2\widetilde{G}}\frac{\partial \tau_{ij}}{\partial \tau}+\frac{1}{2G}\frac{\tau_{ij}-\hat{\tau}_{ij}}{\Delta t}+\frac{\tau_{ij}}{2\mu_{\mathrm{s}}}=\frac{1}{2}\left(\nabla_i v_j+\nabla_j v_i\right). \tag{34}$$

Collecting terms in front of τ_{ij} and ignoring $\hat{\tau}_{ij}$ because it does not change between successive PT iterations, one can reduce the visco-elastic Stokes problem to the previously discussed viscous Stokes problem by replacing the viscosity in the Eq. (28) with the effective "visco-elastic" viscosity:

$$\mu^{\mathrm{ve}}=\left(\frac{1}{G\Delta t}+\frac{1}{\mu_{\mathrm{s}}}\right)^{-1}. \tag{35}$$

The conclusions and optimal parameters' values presented in Sect. 2.3 thus remain valid for the visco-elastic rheology as well.

3 Performance and scaling

Assessing the performance of iterative stencil-based applications is 2-fold and reported here in terms of algorithmic and implementation efficiency.

The accelerated PT method provides an iterative approach that ensures linear scaling of the iteration count with an increase in numerical grid resolution n_x (Sect. 2) – the algorithmic scalability or performance. The major advantage in the design of such an iterative approach is its concise implementation, extremely similar to explicit time integration schemes. Explicit stencil-based applications, such as elastic wave propagation, can show optimal performance and scaling on multi-GPU configurations because they can keep memory access to the strict minimum, leverage data locality and only require point-to-point communication (Podladtchikov and Podladchikov, 2013). Here we follow a similar strategy.

We introduce two metrics: the effective memory throughput (T_{eff}) and the parallel efficiency (E) (Kumar et al., 1994; Gustafson, 1988; Eager et al., 1989). Early formulations of effective memory throughput analysis are found in Omlin et al. (2015a, b); Omlin (2017). The effective memory throughput permits the assessment of the single-processor (GPU or CPU) performance and allows us to deduce potential room for improvement. The parallel efficiency permits the assessment of distributed-memory scalability which may be hindered by interprocess communication, congestion of shared file systems and other practical considerations from scaling on large supercomputers. We perform single-GPU problem size scaling benchmarks to assess the optimal local problem size based on the T_{eff} metric. We further use the optimal local problem size in weak-scaling benchmarks to assess the parallel efficiency $E(N,P)$.

3.1 The effective memory throughput

Many-core processors such as GPUs are throughput-oriented systems that use their massive parallelism to hide latency. On the scientific application side, most algorithms require fewer floating-point operations per second (FLOPS), compared to the amount of numbers or bytes accessed from main memory, and thus are significantly memory bound. The FLOPS metric, no longer being the most adequate for reporting the application performance (e.g. Fuhrer et al., 2018) in a majority of cases, motivated us to develop a memory throughput-based performance evaluation metric, T_{eff}, to evaluate the performance of iterative stencil-based PDE solvers.

The effective memory access, A_{eff} [GB], is the sum of twice the memory footprint of the unknown fields, D_{u}, (fields that depend on their own history and that need to be read from and written to every iteration) and the known fields, D_{k}, that do not change every iteration. The effective memory access divided by the execution time per iteration, t_{it} [s], defines the effective memory throughput, T_{eff} [GB s^{-1}]:

$$A_{\mathrm{eff}}=2\,D_{\mathrm{u}}+D_{\mathrm{k}}, \tag{36}$$

$$T_{\mathrm{eff}}=\frac{A_{\mathrm{eff}}}{t_{\mathrm{it}}}. \tag{37}$$

The upper bound of T_{eff} is T_{peak} as measured e.g. by McCalpin (1995) for CPUs or a GPU analogue. Defining the T_{eff} metric, we assume that (i) we evaluate an iterative stencil-based solver, (ii) the problem size is much larger than the cache sizes and (iii) the usage of time blocking is not feasible or advantageous (which is a reasonable assumption for real-world applications). An important concept is to not include fields within the effective memory access that do not depend on their own history (e.g. fluxes); such fields can be recomputed on the fly or stored on-chip. Defining a theoretical upper bound for T_{eff} that is closer to the real upper bound is a work in progress (Omlin and Räss, 2021b).

3.2 The parallel efficiency

We employ the parallel efficiency metric to assess the scalability of the iterative solvers when targeting distributed-memory configurations, such as multi-GPU settings. In a weak-scaling configuration, i.e. where the global problem size and computing resources increase proportionally, the parallel efficiency $E(N,P)$ defines the ratio between the execution time of a single process, $T(N,1)$, and the execution time of P processes performing the same number of iterations on a P-fold larger problem, $T(N\cdot P,P)$, where N is the local problem size and P is the number of parallel processes:

$$E(N,P)=\frac{T(N,1)}{T(N\cdot P,P)}. \tag{38}$$

Distributed parallelisation permits overcoming limitations imposed by the available main memory of a GPU or CPU.

It is particularly relevant for GPUs, which have significantly less main memory available than CPUs. Distributing work amongst multiple GPUs, using e.g. the message passing interface (MPI), permits overcoming these limitations and requires parallel computing and supercomputing techniques. Parallel efficiency is a key metric in light of assessing the overall application performance as it ultimately ensures scalability of the PT method.

4 The numerical experiments

We design a suite of numerical experiments to verify the scalability of the accelerated PT method, targeting diffusive processes and mechanics. We consider three distinct diffusion problems in one, two and three dimensions, that exhibit a diffusion coefficient being (i) linear, (ii) a step function with 4 orders of magnitude contrasts and (iii) a cubic power-law relation. We then consider mechanical processes using a velocity–pressure formulation to explore various limits, including variable-viscosity incompressible viscous flow limit, accounting for a Maxwell visco-elastic shear rheology. To demonstrate the versatility of the approach, we tackle the nonlinear mechanical problem of strain localisation in two and three dimensions considering an E-VP rheology (Sect. 8). We verify the robustness of the accelerated PT method by considering two parametric studies featuring different viscous Stokes flow patterns, and demonstrate the convergence of the method for viscosity contrasts up to 9 orders of magnitude. We finally investigate the convergence of visco-elastic Stokes flow for non-similar domain aspect ratio.

4.1 Diffusive processes

We first consider time-dependent (transient) diffusion processes defined by Eqs. (1) and (2), with the proportionality coefficient $\rho = 1$. Practical applications often exhibit at least one diffusive component which can be either linear or nonlinear. Here, we consider linear and nonlinear cases representative of challenging configurations common to a broad variety of forward numerical diffusion-type models:

1. The first case exhibits a linear constant (scalar) diffusion coefficient:

$$D = 1. \tag{39}$$

2. The second case exhibits a spatially variable diffusion coefficient with a contrast of 4 orders of magnitude:

$$D = \begin{cases} 1 & \text{if } L < L_D, \\ 10^{-4} & \text{if } L >= L_D, \end{cases} \tag{40}$$

where L is the domain extent in a specific dimension and L_D the coordinate at which the transition occurs. Large contrasts in material parameters (e.g. permeability or heat conductivity) are common challenges that solvers needs to handle when targeting real-world applications.

3. The third case exhibits a nonlinear power-law diffusion coefficient:

$$D = H^n, \tag{41}$$

where $n = 3$, is a characteristic value in, e.g. soil and poro-mechanical applications to account for the porosity–permeability Carman–Kozeny (Costa, 2006) relation leading to the formation of solitary waves of porosity. Shallow ice approximation or nonlinear viscosity in power-law creep Stokes flow are other applications that exhibit effective diffusion coefficients to be defined as power-law relations.

Practically, we implement the transient diffusion using the accelerated PT method, solving Eqs. (6) and (15) using a dual-time method (Sect. 2.1.4).

4.2 Mechanics

We secondly consider steady-state mechanical problems, defined by Eqs. (22) and (23). In practice, we employ a velocity–pressure formulation, which allows us to also handle the incompressible flow limit. The rheological model builds on an additive decomposition of the deviatoric strain rate tensor (Maxwell's model), given by Eq. (33).

In Sect. 5.2, the mechanical problem is solved in the incompressible limit and assuming a linear visco-elastic deviatoric rheology.

In the subsequent application (Sect. 8), the mechanical problem is solved in the compressible E-VP limit. Hence, the deviatoric rheological model neglects viscous flow and includes viscoplastic flow:

$$\dot{\varepsilon}_{ij} = \dot{\varepsilon}_{ij}^{\mathrm{e}} + \dot{\varepsilon}_{ij}^{\mathrm{vp}} = \frac{1}{2G}\frac{\partial \tau_{ij}}{\partial t} + \dot{\lambda}\frac{\partial Q}{\partial \tau_{ij}}, \tag{42}$$

where $\dot{\lambda}$ and Q stand for the rate of the plastic multiplier and the plastic flow potential, respectively. A similar decomposition is assumed for the divergence of velocity in Eq. (23), which is no longer equal to zero in order to account for elastic and plastic bulk deformation:

$$\nabla_k v_k = \nabla_k v_k^{\mathrm{e}} + \nabla_k v_k^{\mathrm{vp}} = -\frac{1}{K}\frac{\partial p}{\partial t} - \dot{\lambda}\frac{\partial Q}{\partial p}, \tag{43}$$

where K stands for the physical bulk modulus.

In the inclusion parametric study described in Sect. 5.2, we consider the incompressible viscous Stokes flow limit, i.e. $K \to \infty$ and $G \to \infty$.

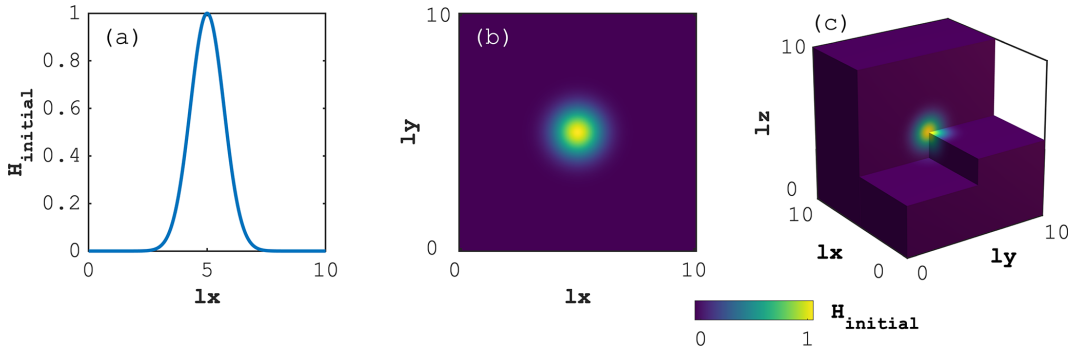

Figure 2. Initial distribution of H for the **(a)** 1D, **(b)** 2D and **(c)** 3D time-dependent diffusion configurations.

5 The model configurations

5.1 The diffusion model

We perform the three different diffusion experiments (see Sect. 4.1) on 1D, 2D and 3D computational domains (Fig. 2a, b and c, respectively). The only difference between the numerical experiments lies in the definition of the diffusion coefficient D. The non-dimensional computational domains are $\Omega_{1D} = [0, L_x]$, $\Omega_{2D} = [0, L_x] \times [0, L_y]$ and $\Omega_{3D} = [0, L_x] \times [0, L_y] \times [0, L_z]$, for 1D, 2D and 3D domains, respectively. The domain extent is $L_x = L_y = L_z = 10$. The initial condition, H_0, consists of a Gaussian distribution of amplitude and standard deviation equal to one located in the domain's centre; in the 1D case:

$$H_0 = \exp\left(-(x_c - 0.5\, L_x)^2\right), \tag{44}$$

where x_c is the vector containing the discrete 1D coordinates of the cell centres. The 2D and 3D cases are done by analogy and contain the respective terms for the y and z directions. We impose Dirichlet boundary conditions such that $H = 0$ on all boundaries. We simulate a total non-dimensional physical time of 1 performing five implicit time steps of $\Delta t = 0.2$.

5.2 The Stokes flow model

We perform the visco-elastic Stokes flow experiments (see Sect. 4.2) on 2D and 3D computational domains (Fig. 3a and b, respectively). The non-dimensional computational domains are $\Omega_{2D} = [0, L_x] \times [0, L_y]$ and $\Omega_{3D} = [0, L_x] \times [0, L_y] \times [0, L_z]$ for 2D and 3D domains, respectively. The domain extend is $L_x = L_y = L_z = 10$. As an initial condition, we define a circular (2D) or spherical (3D) inclusion of radius $r = 1$ centred at $L_x/2$, $L_y/2$ (2D) and $L_x/2$, $L_y/2$, $L_z/2$ (3D), featuring 3 orders of magnitude with lower shear viscosity $\mu_s^{inc} = 10^{-3}$ compared to the background value $\mu_s^0 = 1$ (Fig. 3). We then perform 10 explicit diffusion steps of the viscosity field μ_s to account for smoothing introduced by commonly employed advection schemes (e.g. markers-in-cell, semi-Lagrangian or weighted ENO). We define a uniform and constant elastic shear modulus $G = 1$ and chose the physical time step $\Delta t = \mu_s^0/G/\xi$ to satisfy a visco-elastic Maxwell relaxation time of $\xi = 1$. We impose pure shear boundary conditions; we apply compression in the x direction and extension in the vertical (y in 2D, z in 3D) direction with a background strain rate $\varepsilon_{BG} = 1$. For the 3D case, we apply no inflow/outflow in the y direction (Fig. 3b). All model boundaries are free to slip. We perform a total of five implicit time steps to resolve visco-elastic stress build-up.

We further perform a series of viscous Stokes numerical experiments in 2D (see Sect. 4.2) to analyse the dependence of the optimal iteration parameters on the material viscosity contrast and the volume fraction of the material with lower viscosity. The non-dimensional computational domain is $\Omega_{2D} = [0, L_x] \times [0, L_y]$. As an initial condition, we define a number n_{inc} of circular inclusions that are semi-uniformly distributed in the domain. The viscosity in the inclusions is μ_s^{inc} and the background viscosity is $\mu_s^0 = 1$.

In the parametric study, we vary the number of inclusions n_{inc}, the inclusion viscosity μ_s^{inc}, and the iteration parameter Re. We consider a uniform 3D grid of parameter values, numerically calculating the steady-state distribution of stresses and velocities for each of these combinations.

We consider two different problem setups that correspond to important edge cases. The first setup addresses the shear-driven flow where the strain rates are assumed to be applied externally via boundary conditions. This benchmark might serve as a basis for the calculation of effective-media properties. The second setup addresses the gravity-driven flow with buoyant inclusions. This benchmark is relevant for geophysical applications, e.g. modelling magmatic diapirism or melt segregation, where the volumetric effect of melting leads to the development of either the Rayleigh–Taylor instability or compaction instability, respectively.

In the first setup, we specify pure-shear boundary conditions similar to the singular inclusion case described in Sect. 5.2. The body forces f_i are set to zero in Eq. (22).

In the second setup, we specify the free-slip boundary conditions, which correspond to setting the background strain

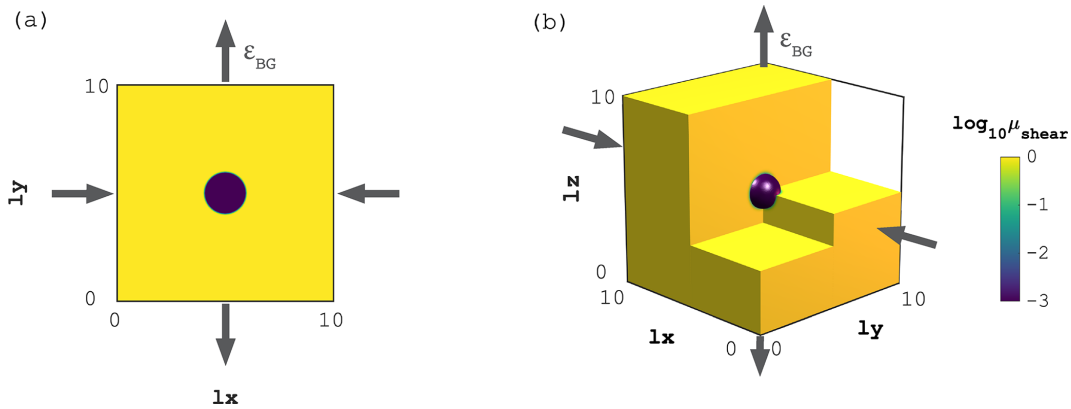

Figure 3. Initial shear viscosity (μ_{shear}) distribution for the **(a)** 2D and **(b)** 3D visco-elastic Stokes flow configuration, respectively.

rate ε_{BG} to 0. We model buoyancy using the Boussinesq approximation: the density differences are accounted for only in the body forces. We set $f_x = 0$, $f_y = -\rho g$. We set $\rho g^0 = 1$, $\rho g^{\text{inc}} = 0.5$.

6 Discretisation

We discretise the systems of partial differential equations (Sect. 4) using the finite-difference method on a regular Cartesian staggered grid. For the diffusion process, the quantity being diffused and the fluxes are located at cell centres and cell interfaces, respectively. For the Stokes flow, pressure, normal stresses and material properties (e.g. viscosity) are located at cell centres, while velocities are located at cell interfaces. Shear stress components are located at cell vertices. The staggering relies on second-order conservative finite differences (Patankar, 1980; Virieux, 1986; McKee et al., 2008), also ensuring that the Stokes flow is inherently devoid of oscillatory pressure modes (Shin and Strikwerda, 1997).

The diffusion process and the visco-elastic Stokes flow include physical time evolution. We implement a backward Euler time integration within the PT solving procedure (see Sect. 2) and do not assess higher-order schemes as such considerations go beyond the scope of this study.

In all simulations we converge the scaled and normalised L2-norm of the residuals, $||R||_{\text{L2}}/\sqrt{n_R}$, where n_R stands for the number of entries of R, for each physical time step to a nonlinear absolute tolerance of $\text{tol}_{\text{nl}} = 10^{-8}$ within the iterative PT procedure (absolute and relative tolerances being comparable, given the non-dimensional form of the example we consider here).

The 46-line code fragment (Fig. 4) provides information about the concise implementation of the accelerated PT algorithm, here for the 1D nonlinear power-law diffusion case ($D = H^3$). Besides the initialisation part (lines 3–22), the core of the algorithm is contained in no more than 20 lines (lines 23–43). The algorithm is implemented as two nested (pseudo-)time loops, referred to as "dual-time"; the outer loop advancing in physical time, the inner loop converging the implicit solution in pseudo-time. The nonlinear term, here the diffusion coefficient D, is explicitly evaluated within the single inner-loop iterative procedure, removing the need for performing nonlinear iterations on top of a linear solver (e.g. Brandt, 1977; Trottenberg et al., 2001; Hackbusch, 1985). This single inner-loop local linearisation shows, in practice, a lower iteration count when compared to global linearisation (nested loops). Note that a relaxation of nonlinearities can be implemented in a straightforward fashion if the nonlinear term hinders convergence (see implementation details in e.g. Räss et al., 2020, 2019a; Duretz et al., 2019b). The iteration parameters are evaluated locally which ensures scalability of the approach and removes the need for performing global reductions, costly in parallel implementation. Note that the numerical iteration parameters $\tilde{\rho}$ and θ_{r}, arising from the finite-difference discretisation of pseudo-time derivatives in Eqs. (16) and (6),

$$\tilde{\rho}\frac{\partial H}{\partial \tau} \approx \tilde{\rho}\frac{H^k - H^{k-1}}{\Delta\tau}, \quad \theta_{\text{r}}\frac{\partial q_i}{\partial \tau} \approx \theta_{\text{r}}\frac{q_i^k - q_i^{k-1}}{\Delta\tau}, \tag{45}$$

where k is the current pseudo-time iteration index, always occur in combination with $\Delta\tau$. Since we are not interested in the evolution of pseudo-time or the particular values of iteration parameters, it is possible to combine them in the implementation. We therefore introduce the variables $\text{d}\tau_\rho = \Delta\tau/\rho$ and $\theta\text{r_d}\tau = \theta\text{r}/\Delta\tau$. Using the new variables helps to avoid specifying the value of $\Delta\tau$, which could otherwise be specified arbitrarily. The two lines of physics, namely the PT updates, are here evaluated in an explicit fashion. Alternatively, one could solve `qHx` and `H` assuming that their values in the residual – the terms contained in the right-most parenthesis – are new instead of current, resulting in an implicit update. Advantages rely on enhanced stability (`CFL` on line 10 could be set to 1) and remove the need for defining a small number (ε in the iteration parameters definition) to prevent division by 0. The implicit approach is implemented

```julia
# [...]
@views function diffusion_1D(; implicit=false, do_viz=true)
    # Physics
    lx     = 10.0       # domain size
    ttot   = 1.0        # total simulation time
    dt     = 0.2        # physical time step
    # Numerics
    nx     = 512        # numerical grid resolution
    tol    = 1e-8       # tolerance
    CFL    = 0.8        # CFL number: 1.0 implicit, 0.8 explicit
    ε      = 1e-2       # small number for explicit stability
    # [...]
    # Derived numerics
    dx     = lx / nx    # grid size
    Vpdτ   = CFL * dx
    xc     = LinRange(dx / 2, lx - dx / 2, nx)
    # Array allocation
    # [...]
    # Initial condition
    H0     = exp.(-(xc .- lx / 2).^2)
    # [...]
    # Physical time loop
    while it < nt
        iter = 0; err = 2 * tol
        # Pseudo-transient iteration
        while err > tol && iter < itMax
            # Diffusion coefficient
            D          .= H.^3
            # Assign iter params
            Re         .= π .+ sqrt.(π^2 .+ lx^2 ./ max.(D,ε) ./ dt)
            θr_dτ      .= lx ./ Vpdτ ./ av(Re)
            dτ_ρ       .= Vpdτ .* lx ./ max.(D[2:end-1],ε) ./ Re[2:end-1]
            # PT updates
            # [...]
            qHx        .= qHx        .+  1.0 ./ θr_dτ .* ( .-qHx .-av(D) .* diff(H) ./ dx )
            H[2:end-1] .= H[2:end-1] .+  dτ_ρ         .* ( .-(H[2:end-1] .- Hold[2:end-1]) ./ dt .- diff(qHx) ./ dx )
            # Check errors
            # [...]
        end
        # Update H
        Hold .= H
        ittot += iter; it += 1; t += dt
    end
    # [...]
    return
end
```

Figure 4. Numerical Julia implementation of the 1D nonlinear diffusion case $D = H^3$. Lines marked with # [...] refer to skipped lines. See diff_1D_nonlin_simple.jl script located in the scripts folder in PseudoTransientDiffusion.jl on GitHub for full code.

as an alternative in the full code available online in the PseudoTransientDiffusion.jl GitHub repository.

6.1 The Julia multi-xPU implementation

We use the Julia language (Bezanson et al., 2017) to implement the suite of numerical experiments. Julia's high-level abstractions, multiple dispatch and meta-programming capabilities make it amenable to portability between back-ends (e.g. multi-core CPUs and Nvidia or AMD GPUs). Also, Julia solves the two-language problem, making it possible to fuse prototype and production applications into a single one that is both high-level and performance oriented – ultimately increasing productivity.

We use the ParallelStencil.jl (Omlin and Räss, 2021b) and ImplicitGlobalGrid.jl (Omlin and Räss, 2021a) Julia packages that we developed as building blocks to implement the diffusion and Stokes numerical experiments. ParallelStencil.jl permits to write architecture–agnostic high-level code for parallel high-performance stencil computations on GPUs and CPUs – here referred to as xPUs. Performance similar to native CUDA C/C++ (Nvidia GPUs) or HIP (AMD GPUs) can be achieved. ParallelStencil.jl seamlessly combines with ImplicitGlobalGrid.jl, which allows for distributed parallelisation of stencil-based xPU applications on a regular staggered grid. In addition, ParallelStencil.jl enables hiding communication behind computation, where the communication package used can, a priori, be any package that allows the user to control when communication is triggered. The overlap approach of communication and computation splits local domain calculations into two regions, i.e. boundary regions and inner region, the latter containing majority of the local domain's grid cells. After successful completion of the boundary region computations, halo update (using e.g. point-to-point MPI) overlaps with inner-point computations. Selecting the appropriate width of the boundary region permits fine-tuning the optimal hiding of MPI communication (Räss et al., 2019c; Alkhimenkov et al., 2021b).

In the present study, we focus on using ParallelStencil.jl with the CUDA.jl back-end to target Nvidia GPUs (Besard

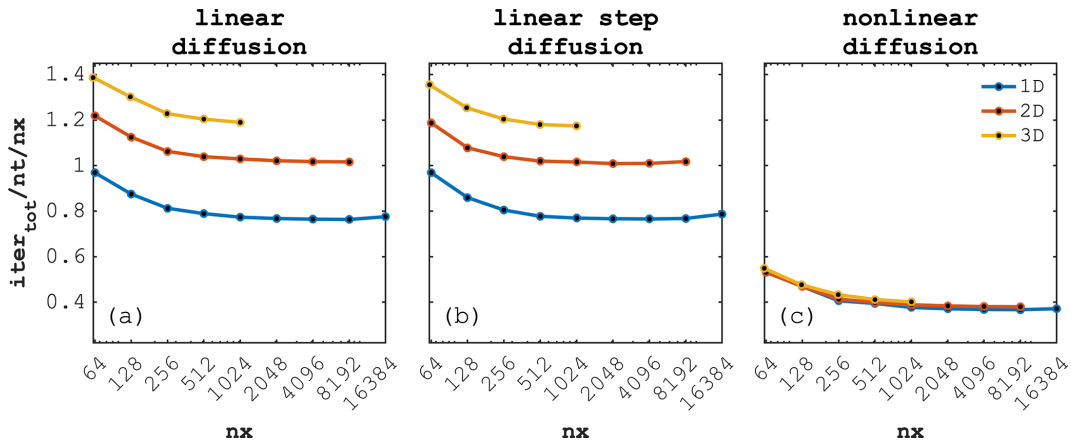

Figure 5. Iteration count scaled by the number of time steps (n_t) and by the number of grid cells in the x direction (n_x) as function of n_x, comparing 1D, 2D and 3D models for the three different diffusion configurations: **(a)** linear diffusion, **(b)** linear step diffusion and **(c)** nonlinear diffusion, respectively.

et al., 2018, 2019), and ImplicitGlobalGrid.jl which relies on MPI.jl (Byrne et al., 2021) and Julia's MPI wrappers to enable distributed-memory parallelisation.

7 Results

We here report the performance of the accelerated PT Julia implementation of the diffusion and the Stokes flow solvers targeting Nvidia GPUs using ParallelStencil.jl's CUDA back-end. For both physical processes, we analyse the iteration count as a function of the number of grid cells (i.e. the algorithmic performance), the effective memory throughput T_{eff} [GB s^{-1}] (performing a single-GPU device problem size scaling), and the parallel efficiency E (multi-GPU weak scaling).

We report the algorithmic performance as the iteration count per number of physical time steps normalised by the number of grid cells in the x direction. We do not normalise by the total number of grid cells in order to report the 1D scaling, even for 2D or 3D implementation. We motivate our choice as it permits a more accurate comparison to analytically derived results and leaves it to the reader to appreciate the actual quadratic and cubic dependence of the normalised iteration count if using the total number of grid cells in 2D and 3D configurations, respectively.

7.1 Solving the diffusion equation

We report a normalised iteration count per total number of physical time steps n_{t} per number of grid cells in the x direction n_x ($\mathrm{iter}_{\mathrm{tot}}/n_t/n_x$), for the 1D, 2D and 3D implementations of the diffusion solver for the linear, step-function and nonlinear case (Fig. 5a, b and c, respectively) relating to the spatial distribution of H after 5 implicit time steps (Fig. 6a–c, d–f and g–i, respectively). All three different configurations exhibit a normalised number of iterations per time step per number of grid cells close to 1 for the lowest resolution of $n_x = 64$ grid cells. The normalised iteration count drops with an increase in numerical resolution (increase in number of grid cells) suggesting a super-linear scaling.

We observe similar behaviour when increasing the number of spatial dimensions while solving the identical problem. For example, in the 3D calculations we actually resolve $n_x \times n_y \times n_z$ (here n_x^3) grid cells, while the reported n_x-normalised iteration count only slightly increases compared to the corresponding 1D case.

It is interesting to note that the diffusion solver with nonlinear (power-law) diffusion coefficient reports the lowest normalised iteration count for all three spatial dimension implementations, reaching the lowest number (> 0.4) of normalised iteration count in the 3D configuration (Fig. 5c). The possible explanation of lower iteration counts for the nonlinear problem is that by the nature of the solution, the distribution of the diffused quantity H at $t = 1$ is much closer to the initial profile than in the linear case. Therefore, at each time step, the values of H are closer to the values of H at the next time step and thus serve as a better initial approximation. Both the diffusion with linear (Fig. 5a) and step function as diffusion coefficient (Fig. 5b) show similar trends in their normalised iteration counts, with values decreasing while increasing the number of spatial dimensions.

7.2 Solving visco-elastic Stokes flow

We further report the normalised iteration count per total number of physical time steps n_{t} per number of grid cells in the x direction n_x ($\mathrm{iter}_{\mathrm{tot}}/n_t/n_x$) for the 2D and 3D implementations of the visco-elastic Stokes solver (Fig. 7) relating to the spatial distribution of vertical velocity (deviation from background) $\Delta V_{\mathrm{vertical}}$, pressure P and shear stress τ_{shear} after 5 implicit time steps (Fig. 8a–b, c–d and e–f, re-

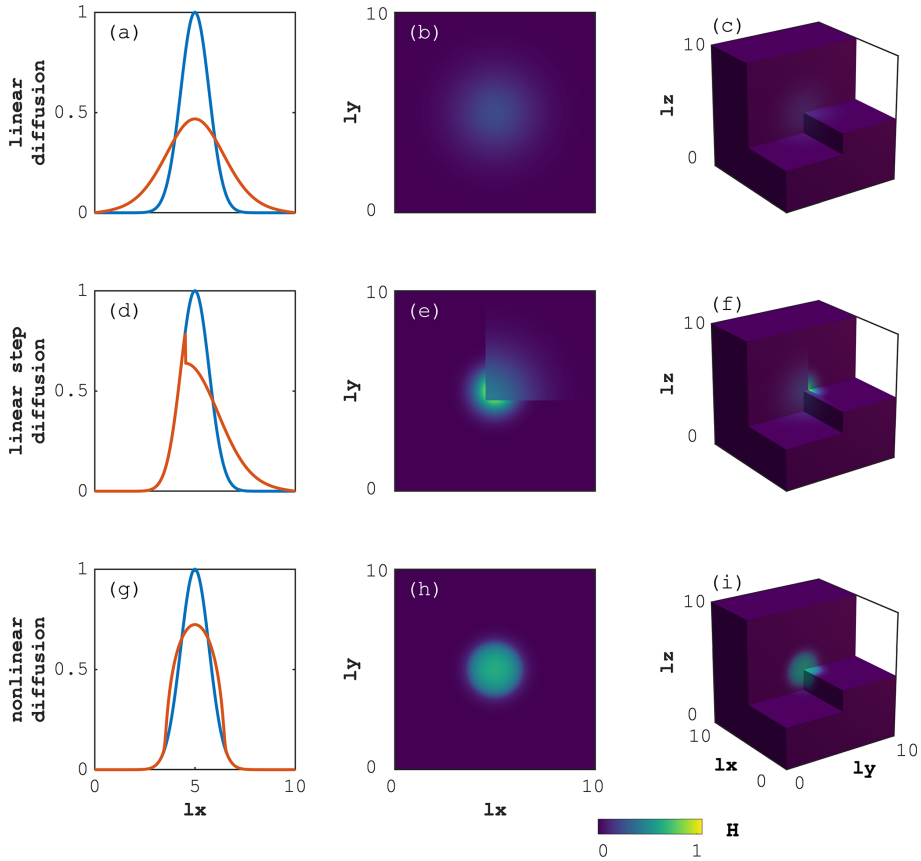

Figure 6. Model output for the 1D **(a, d, g)**, 2D **(b, e, h)** and 3D **(c, f, i)** time-dependent diffusion of quantity H. The upper **(a–c)**, centre **(d–f)** and lower **(g–i)** panels refer to the diffusion with linear, linear step and nonlinear (power-law) diffusion coefficient, respectively. For the 1D case, the blue and red lines represent the initial and final distribution, respectively. The colour map (2D and 3D) relates to the y axis (1D).

spectively). Both 2D and 3D visco-elastic Stokes flow exhibit a normalised number of iterations per time step per number of grid cells close to 10 for the lowest resolution of $n_x = 63$ grid cells. The normalised iteration count drops with an increase in numerical resolution (increase in number of grid cells) suggesting a super-linear scaling. We observe similar behaviour when increasing the number of spatial dimensions from 2D to 3D, while solving the identical problem; 3D calculations are more efficient on a given number of grid cells n_x compared to the corresponding 2D calculations, which is in accordance with results for the various diffusion solver configurations.

The visco-elastic Stokes flow scaling results confirm the trend reporting a decrease of the normalised iteration count with an increase in the numerical resolution (number of grid cells). It is interesting to note that the accelerated PT implementation of the 3D visco-elastic Stokes flow featuring 3 orders of magnitude viscosity contrast ($\mu_s^0/\mu_s^{inc} = 10^3$) only requires less than 17 normalised iterations when targeting resolutions of 1023^3 (Fig. 7).

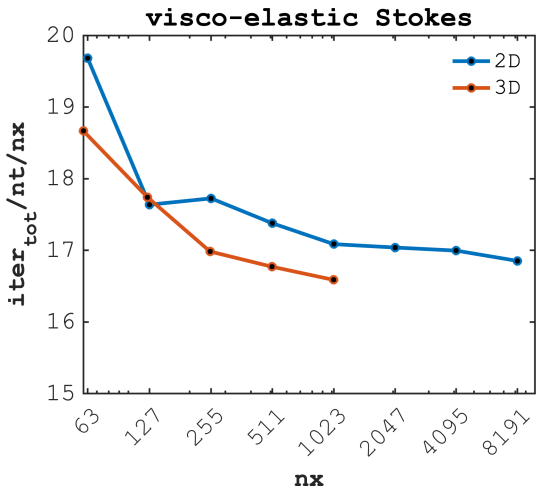

Figure 7. Iteration count scaled by the number of time steps (n_t) and by the number of grid cells in the x direction (n_x) as function of n_x, comparing 2D and 3D visco-elastic Stokes flow containing an inclusion featuring a viscosity contrast ($\mu_0/\mu_i = 10^3$).

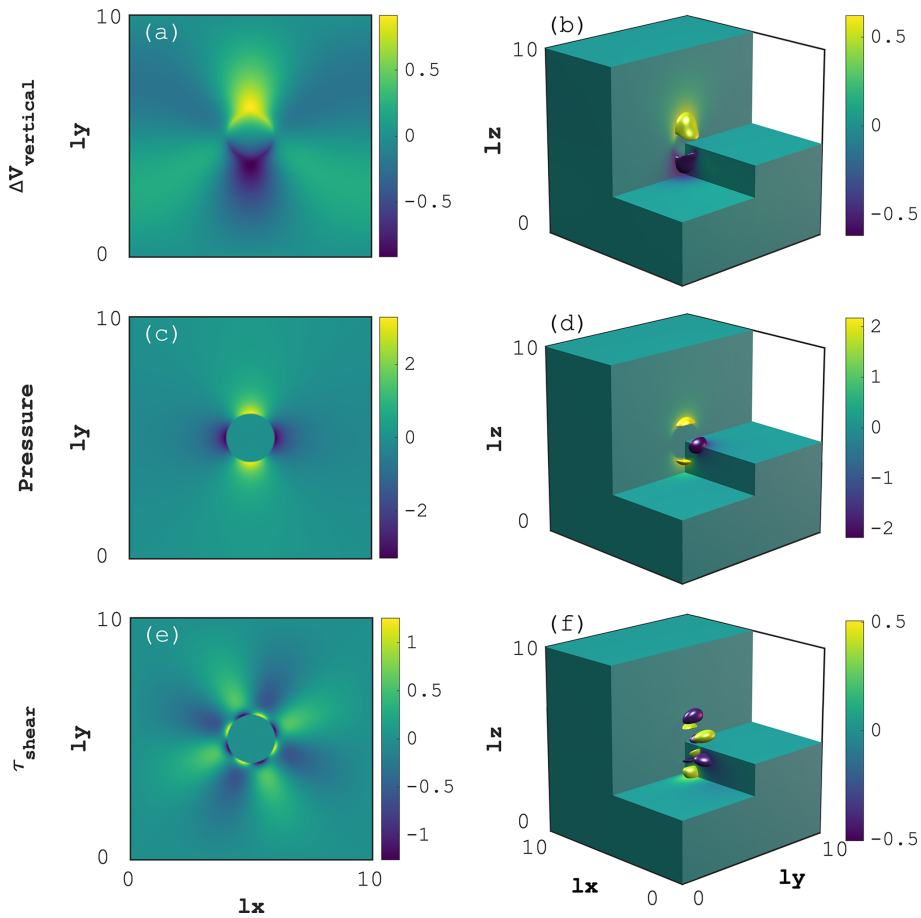

Figure 8. Model output for the 2D **(a, c, e)** and 3D **(b, d, f)** visco-elastic Stokes flow containing an inclusion featuring a viscosity contrast ($\mu_0/\mu_i = 10^3$). The upper **(a–b)**, centre **(c–d)**, and lower **(e–f)** panels depict the deviation from background vertical velocity $\Delta V_{\text{vertical}}$, the dynamic pressure P and deviatoric shear stress τ_{shear}(τ_{xy} in 2D, τ_{xz} in 3D) distribution, respectively.

7.3 Performance

We use the effective memory throughput T_{eff} [GB s^{-1}] and the parallel efficiency $E(N, P)$ to assess the implementation performance of the accelerated PT solvers, as motivated in Sect. 3. We perform the single-GPU problem size scaling and the multi-GPU weak-scaling tests on different Nvidia GPU architectures, namely the "data-centre" GPUs, Tesla P100 (Pascal – PCIe), Tesla V100 (Volta – SXM2) and Tesla A100 (Ampere – SXM4). We run the weak-scaling multi-GPU benchmarks on the *Piz Daint* supercomputer, featuring up to 5704 Nvidia Tesla P100 GPUs, at CSCS, on the *Volta* node of the *Octopus* supercomputer, featuring 8 Nvidia Tesla V100 GPUs with high-throughput (300 GB s^{-1}) SXM2 interconnect, at the Swiss Geocomputing Centre, University of Lausanne, and on the *Superzack* node, featuring 8 Nvidia Tesla A100 GPUs with high-throughput (600 GB s^{-1}) SXM4 interconnect, at the Laboratory of Hydraulics, Hydrology, Glaciology (VAW), ETH Zurich.

We assess the performance of the 2D and 3D implementations of the nonlinear diffusion solver (power-law diffusion coefficient) and the visco-elastic Stokes flow solver, respectively. We perform single-GPU scaling tests for both the 2D and 3D solvers' implementation, and multi-GPU weak-scaling tests for the 3D solvers' implementation only. We report the mean performance out of 5 executions, if applicable.

In order to compute the effective memory throughput T_{eff} in the scaling test, we need to determine the number of unknown and known fields the solvers access during the iterative procedures (see Sect. 3.1). We report the values we use for the nonlinear diffusion solver and the visco-elastic Stokes flow solver in both 2D and 3D, as well as the per grid cell number of DoFs in Table 1.

7.3.1 Single-GPU scaling and effective memory throughput

The 2D and 3D nonlinear diffusion solver single-GPU scaling benchmarks achieve similar effective memory throughput on the targeted GPU architectures relative to their respective peak values T_{peak}; values of T_{eff} for the 2D implementation being slightly higher than the 3D ones for the Volta and

Table 1. Number of unknown (D_u) and known (D_k) fields, as well as per grid cell DoFs used to assess, e.g. the T_{eff} metric in the performance scaling tests.

Application		D_u	D_k	DoFs
Nonlinear diffusion	2D	3	2	1
	3D	4	2	1
Visco-elastic Stokes flow	2D	6	6	6
	3D	10	9	10

Pascal architectures, but similar for the Ampere one. This discrepancy is expected and may be partly explained by an increase in cache-misses when accessing z-direction neighbours which are $n_x \cdot n_y$ grid cells away in main memory. In 2D, we achieve $T_{eff} \approx 920$, 590 and 400 GB s^{-1} on the Tesla A100, Tesla V100 and the Tesla P100, respectively (Fig. 9a). In 3D, we achieve $T_{eff} \approx 920$, 520 and 315 GB s^{-1} on the Tesla A100, Tesla V100 and the Tesla P100, respectively (Fig. 9b).

For the analogous visco-elastic Stokes flow single-GPU scaling tests, we also report higher T_{eff} values for the 2D compared to the 3D implementation for all three targeted architectures. In 2D, we achieve $T_{eff} \approx 930$, 500 and 320 GB s^{-1} on the Tesla A100, Tesla V100 and the Tesla P100, respectively (Fig. 10a). In 3D, we achieve $T_{eff} \approx 730$, 350 and 230 GB s^{-1} on the Tesla A100, Tesla V100 and the Tesla P100, respectively (Fig. 10b). Increased neighbouring access and overall more derivative evaluations may explain the slightly lower effective memory throughput of the visco-elastic Stokes flow solver when compared to the nonlinear diffusion solver.

7.3.2 Weak scaling and parallel efficiency

We assess the parallel efficiency of the 3D nonlinear diffusion and visco-elastic Stokes flow solver multi-GPU implementation performing a weak-scaling benchmark. We use (per GPU) a local problem size of 512^3 for the nonlinear diffusion, and 383^3 and 511^3 for the visco-elastic Stokes flow on the Pascal and Tesla architectures, respectively. Device RAM limitations prevent solving a larger local problem in the latter case. The 3D nonlinear diffusion solver achieves a parallel efficiency E of 97 % on 8 Tesla A100 SXM4 and V100 SXM2, and 98 % on 2197 Tesla P100 GPUs (Fig. 11a). The visco-elastic Stokes flow solver achieves a parallel efficiency E of 99 % on 8 Tesla A100 SXM4, and 96 % on 8 Tesla V100 SXM2 and on 2197 Tesla P100 GPUs (Fig. 11b), respectively. The discrepancy between the 3D nonlinear diffusion and visco-elastic Stokes flow solvers may arise due to the difference in kernel sizes and workload, resulting in different memory and cache usages potentially impacting occupancy. 2197 GPUs represent a 3D Cartesian topology of 13^3, resulting in global problem sizes of 6632^3 and 4995^3 grid cells for the nonlinear diffusion (291 giga DoFs) and visco-elastic Stokes flow (1.2 tera DoFs), respectively. We emphasise that the number of DoFs we report here represents the total number of DoFs, which is the product of the number of grid cells and the per grid cell DoFs (Table 1). In terms of cumulative effective memory throughput T_{eff}, the 3D diffusion and Stokes flow solver achieve 679 and 444 TB s^{-1}, respectively. This near petabyte per second effective throughput reflects the impressive memory bandwidth exposed by GPUs and requires efficient algorithms to leverage it.

We emphasise that we follow a *strict definition of parallel efficiency*, where the runtimes of the multi-xPU implementations are to be compared with the best known single-xPU implementation. As a result, the reported parallel efficiency is also below 100 % for a single GPU, correctly showing that the implementation used for distributed parallelisation performs slightly worse than the best known single-GPU implementation. This small performance loss emerges from the computation splitting in boundary and inner regions required by the hidden communication feature. Parallel efficiency close to 100 % is important to ensure weak scalability of numerical applications when executed on a growing number of distributed-memory processes P, the path to leverage current and future supercomputers' exascale capabilities.

7.4 Multiple-inclusions parametric study

We perform a multiple-inclusions benchmark to assess the robustness of the developed accelerated PT method. We vary the viscosity contrast from 1 to 9 orders of magnitude to demonstrate the successful convergence of iterations, even for extreme cases, that might arise in geophysical applications such as strain localisation. Further, we vary the number of inclusions from 1 to 46 to verify the independence of convergence on the "internal geometry" of the problem. For each combination of viscosity ratio and number of inclusions, we perform a series of simulations varying the iteration parameter Re to assess the influence of the problem configuration on its optimal value, and to verify whether the analytical prediction obtained by the dispersion analysis remains valid over the considered parameter range.

For this parametric study, we considered a computational grid consisting of $n_x \times n_y = 2048^2$ cells. At lower resolutions the convergence deteriorates, resulting in configurations of non-converging large viscosity contrasts. A high-resolution grid is thus necessary for resolving the small-scale details of the flow. We also adjust the nonlinear tolerance for the iterations to 10^{-5} and 10^{-3} for momentum and mass balance, respectively, given our interest in relative dependence of iteration counts on the iteration parameter Re.

Figure 12 depicts the results for the shear-driven flow case. For a single inclusion (Fig. 12a), the optimal value of the iteration parameter Re does not differ significantly from the one reported by Eq. (31). Moreover, the theoretical prediction for Re remains valid for all viscosity contrasts considered in the

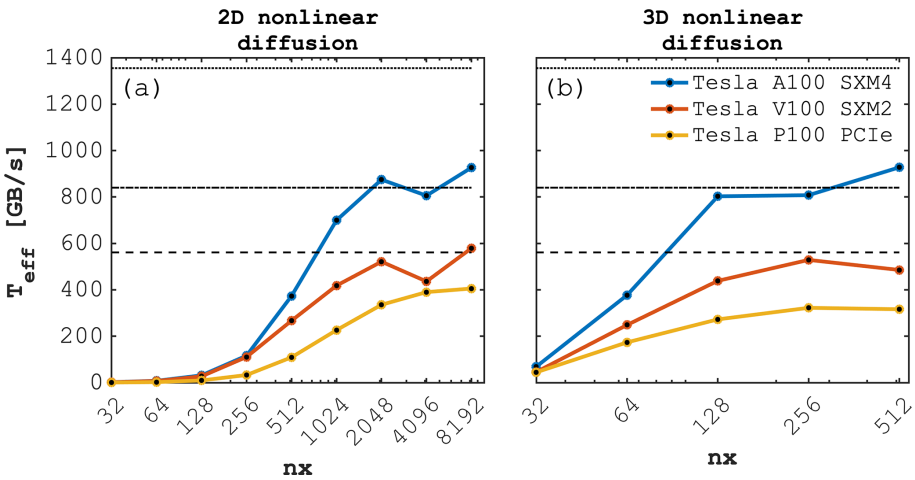

Figure 9. Effective memory throughput T_{eff} [GB s^{-1}] for the **(a)** 2D and **(b)** 3D nonlinear diffusion Julia GPU implementations using ParallelStencil.jl executed on various Nvidia GPUs (Tesla A100 SXM4, Tesla V100 SXM2 and P100 PCIe). The dashed lines report the measured peak memory throughput T_{peak} (1355 GB s^{-1} Tesla A100, 840 GB s^{-1} Tesla V100, 561 GB s^{-1} Tesla P100) one can achieve on a specific GPU architecture, which is a theoretical upper bound of T_{eff}.

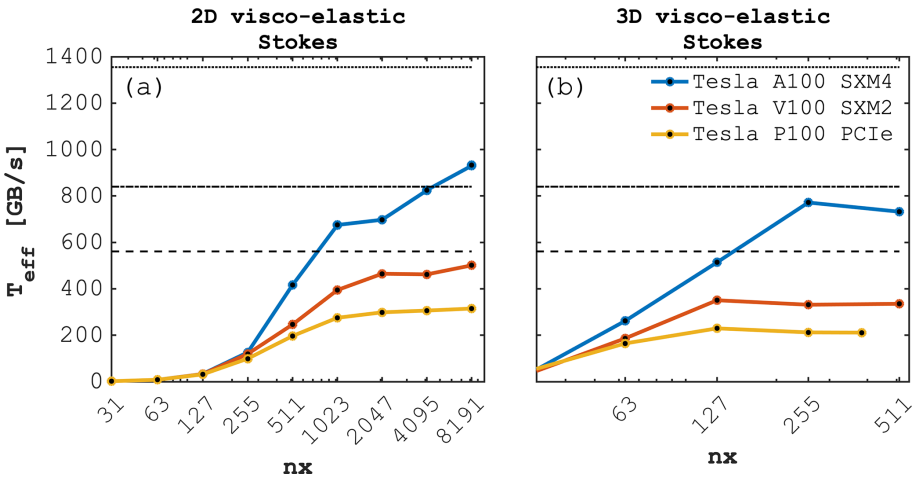

Figure 10. Effective memory throughput T_{eff} [GB s^{-1}] for the **(a)** 2D and **(b)** 3D the visco-elastic Stokes flow Julia GPU implementations using ParallelStencil.jl executed on various Nvidia GPUs (Tesla A100 SXM4, Tesla V100 SXM2 and P100 PCIe). The dashed lines report the measured peak memory throughput T_{peak} (1355 GB s^{-1} Tesla A100, 840 GB s^{-1} Tesla V100, 561 GB s^{-1} Tesla P100) one can achieve on a specific GPU architecture, which is a theoretical upper bound of T_{eff}.

study. For problem configurations involving 14 and 46 inclusions (Fig. 12b and c, respectively), the minimal number of iterations is achieved for values of Re close to the theoretical prediction, only for the viscosity contrast of 1 order of magnitude. For a larger viscosity contrast, the optimal value of Re appears to be lower than theoretically predicted, and the overall iteration count is significantly higher. These iteration counts reach $40n_x$ at the minimum among all values of Re for a given viscosity ratio, and $> 50n_x$ for non-optimal values of Re.

For buoyancy-driven flow (Fig. 13), the convergence of iterations is less sensitive to both the number of inclusions and the viscosity ratio. The observed drift in the optimal value of Re could be partly attributed to the lack of a good preconditioning technique. In this study, we specify the local iteration parameters, $\Delta\tau/\tilde{\rho}$ and $\tilde{G}/\Delta\tau$ (see Sect. 2.3), in each grid cell based on the values of material parameters, which could be regarded as a form of diagonal (Jacobi) preconditioning. This choice is motivated by the parallel scalability requirements of GPU and parallel computing. Even without employing more advanced preconditioners, our method remains stable and successfully converges for viscosity contrasts up to 9 orders of magnitude, though at the cost of increased number of iterations. The physically motivated iteration strategy enables one to control the stability of iterations through the single CFL-like parameter (see Sect. 2.1.2).

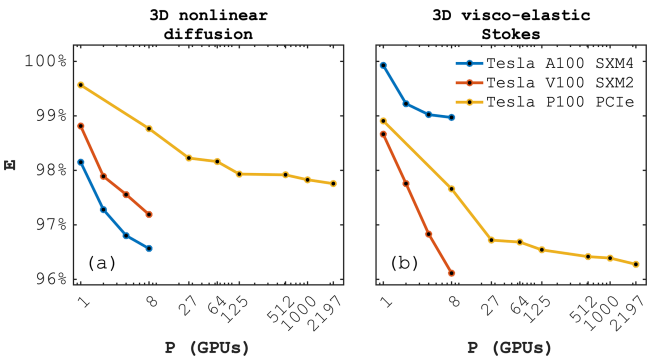

Figure 11. Parallel efficiency E as function of 1–8 Tesla A100 SXM4, 1–8 Tesla V100 SXM2, and 1–2197 Tesla P100 Nvidia GPUs (P). Weak-scaling benchmark for **(a)** the 3D nonlinear diffusion solver and **(b)** the 3D visco-elastic Stokes flow solver based on a 3D Julia implementation using ParallelStencil.jl and ImplicitGlobalGrid.jl.

In both shear-driven and gravity-driven problem setups, the convergence is significantly slower than that of the single-centred inclusion case. This slowdown could be explained by the complicated internal geometry involving non-symmetrical inclusion placement featuring huge viscosity contrasts which results in a stiff system.

7.5 Non-similar domain aspect ratio

In geoscientific models that resolve e.g. flow fields for ice sheets and glaciers evolution, lithospheric deformation, or atmospheric and oceanic circulation, there are usually orders of magnitude differences between horizontal and vertical scales. Such domain configurations feature a large aspect ratio that may challenge the solvers because of the presence of more orders-of-magnitude grid cells in the horizontal than the vertical dimensions. Here we systematically investigate the convergence of the 2D visco-elastic Stokes flow while varying the aspect ratio defined as L_x/L_y from 1 to 8. We repeat the initial condition of the circular inclusion 1 to 8 times in the x direction to investigate the sensitivity of the accelerated PT solver on the domain aspect ratio. We proportionally increase the number of grid cells in order to keep a constant cell aspect ratio of 1. We report the normalised number of iterations to remain constant for all 5 investigated resolutions when the aspect ratio equals 1 (Fig. 14). The number of normalised iteration counts does not vary significantly while increasing the aspect ratio, independently of the employed resolution. Note that we see a slight dependence of the optimal iteration parameter Re on the aspect ratio.

8 Applications to nonlinear mechanical problems with elasto-viscoplastic rheology

To demonstrate the versatility of the approach, we tackle the nonlinear mechanical problem of strain localisation in 2D and 3D. In the following applications we consider an E-VP rheological model, thus the serial viscous damper is deactivated and the flow includes effects of compressibility and plastic dilatancy. We assume a small-strain approximation. Hence, the deviatoric strain rate tensor may be decomposed in an additive manner in Eq. (42). A similar decomposition is assumed for the divergence of velocity in Eq. (43). The plastic model is based on consistent elasto-viscoplasticity and the yield function is defined as

$$F = \tau_{\mathrm{II}} - p\sin\phi - c\cos\phi - \dot{\lambda}\mu^{\mathrm{vp}}, \tag{46}$$

where τ_{II} is the second stress invariant, ϕ is the friction angle, μ^{vp} is the viscoplastic viscosity and c is the cohesion. At the trial state, F^{trial} is evaluated assuming no plastic deformation ($\dot{\lambda} = 0$). Cohesion strain softening is applied and the rate of c is expressed as

$$\frac{\partial c}{\partial t} = \dot{\lambda} h, \tag{47}$$

where h is a hardening/softening modulus. Viscoplastic flow is non-associated and the potential function is expressed as

$$Q = \tau_{\mathrm{II}} - p\sin\psi, \tag{48}$$

where ψ is the dilatancy angle. If $F \geq 0$, viscoplastic flow takes place and the rate of the plastic multiplier is positive and defined as

$$\dot{\lambda} = \frac{F^{\mathrm{trial}}}{\mu^{\mathrm{ve}} + \mu^{\mathrm{vp}} + K\,\Delta t\sin\psi\sin\phi}. \tag{49}$$

The initial model configuration assumes a random initial cohesion field. Pure shear kinematics are imposed at the boundaries of the domain (see Sect. 5.2). The reader is referred to Duretz et al. (2019a) for the complete set of material parameters. We only slightly modify the value of μ^{vp} with respect to Duretz et al. (2019a), such that $\mu^{\mathrm{vp}} = 9\times10^{18}\,\mathrm{Pa\,s}^{-1}$ in the present study.

8.1 Performance benefits for desktop-scale computing

Besides the potential to tackle nonlinear multi-physics problems at supercomputer-scale, the ability to solve smaller-scale nonlinear problems remains an important aspect. Here we investigate wall times for the simulation of the previously-described E-VP shear-band formation in 2D (Fig. 15). We compare a MATLAB-based direct-iterative (DI) and a Julia GPU-based PT solver, respectively. The M2Di MATLAB solver routines (Räss et al., 2017) rely on a Newton linearisation of the nonlinear mechanical problem (Duretz et al., 2019a) and use a performant DI solver to

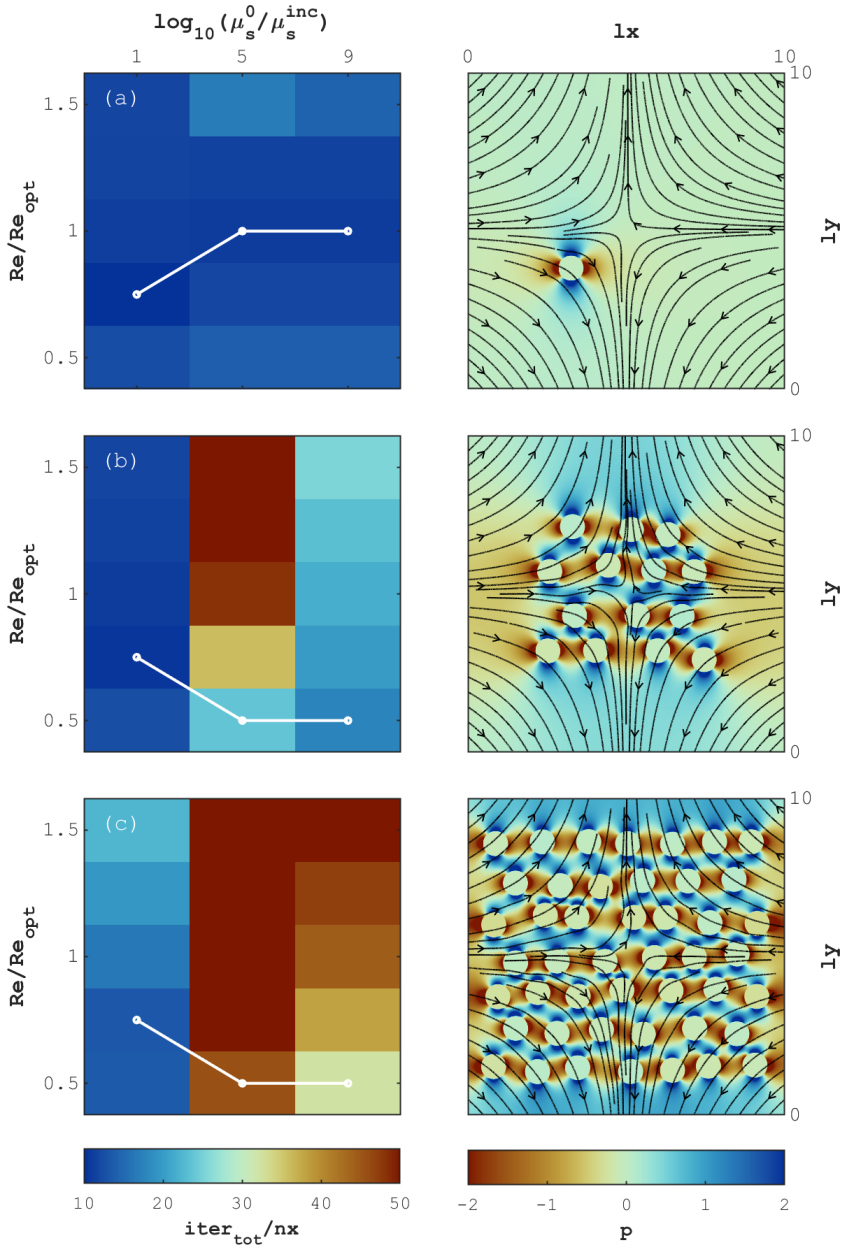

Figure 12. Pure shear-driven flow. The left column reports the number of iterations for different values of viscosity ratio μ_s^0/μ_s^{inc} and ratio of numerical Reynolds number Re to the theoretically predicted value Re_{opt}. Connected white dots indicate the value of Re at which the minimal number of iterations is achieved. The right column depicts the distribution of pressure and velocity streamlines. Panels **(a)**, **(b)** and **(c)** correspond to problem configurations involving 1, 14 and 46 inclusions, respectively.

compute Newton steps. The solver combines outer Powell–Hestenes and inner Krylov iterations (Global Conjugate Residual) that are preconditioned with the Cholesky factorisation of the symmetrised Jacobian (Räss et al., 2017). In the 2D PT solver, written in Julia using the ParallelStencil.jl packages, the evaluation of nonlinearities is embedded in the pseudo-time integration loop. The timings reported for the DI and PT schemes were produced on a 2.9 GHz Intel Core i5 processor and on a single Nvidia Tesla A100 GPU, respectively (Fig. 15a). Each simulation resolves 100 physical (implicit) time steps and we report the total accumulated plastic strain (Fig. 15b) and the associated pressure fields (Fig. 15c). Models were run on resolutions involving 62^2 to 510^2 and up to 1022^2 grid cells for the DI and PT scheme, respectively. As expected for 2D computations, reported wall times are lower using the DI scheme at very low resolutions of $n_x = 62$. However, it is interesting to observe that the GPU-accelerated PT scheme can deliver comparable

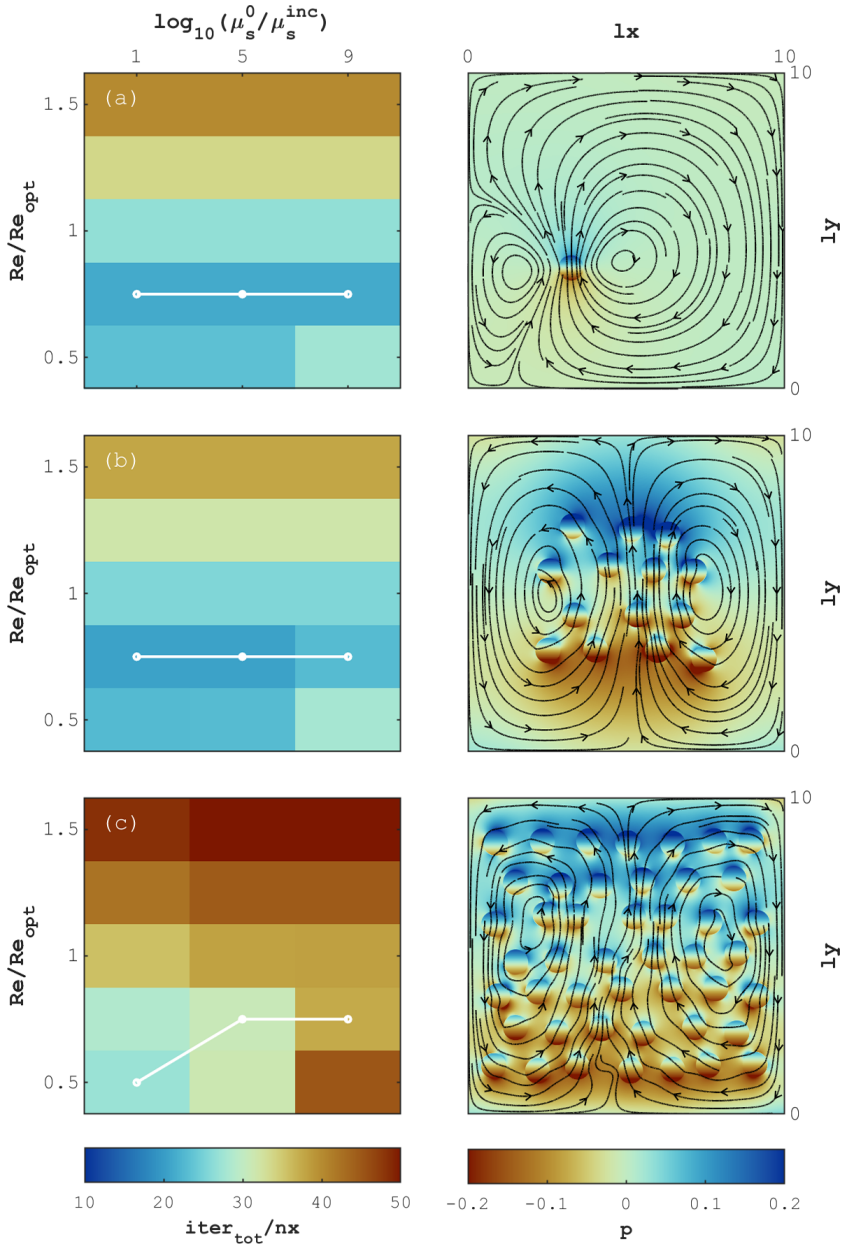

Figure 13. Buoyancy-driven flow. The left column reports the number of iterations for different values of viscosity ratio μ_s^0/μ_s^{inc} and ratio of numerical Reynolds number Re to the theoretically predicted value Re_{opt}. Connected white dots indicate the value of Re at which the minimal number of iterations is achieved. The right column depicts the deviation of pressure from hydrostatic distribution and velocity streamlines. Panels **(a)**, **(b)** and **(c)** correspond to problem configurations involving 1, 14 and 46 inclusions, respectively.

wall times at already relatively low resolutions ($n_x \approx 126$). For numerical resolution of $n_x = 510$, the GPU-based PT solver outperforms the DI solver by a factor ≈ 6 resulting in a wall time of ≈ 20 minutes to resolve the 100 physical time steps (see dashed lines on Fig. 15a). The employed CPU and GPU can be considered as common devices on current scientific desktop machines. We can thus conclude that the use of the GPU-accelerated PT scheme is a viable and practical approach to solve nonlinear mechanical problems on a desktop-scale computer. Moreover the proposed PT scheme has already turned out to be beneficial over common approaches (DI schemes) at relatively low resolutions.

8.2 High-resolution 3D results

We present preliminary 3D results of the spontaneous development of visco-plastic shear bands in pure shear deformation from an initial random cohesion field (Fig. 16). These 3D results further demonstrate the versatility of the PT approach,

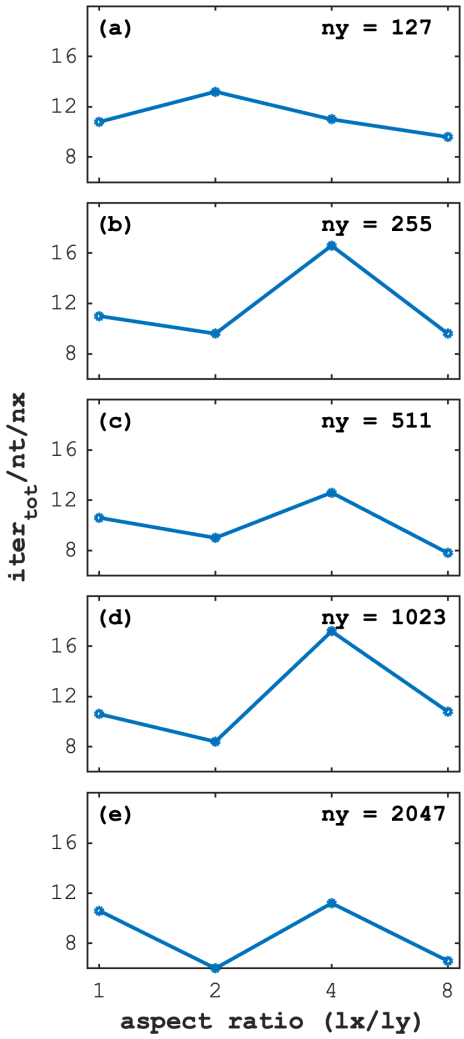

Figure 14. Iteration count scaled by the number of time steps (n_t) and by the number of grid cells in the x direction (n_x) as a function of increasing domain aspect ratios L_x/L_y for 5 different numerical resolutions **(a)**–**(e)**, respectively. The y-direction domain length is kept constant while the x-direction domain length and number of grid cells n_x is increased up to 8 times. We use an identical 2D configuration as in the main study with the only difference being the nonlinear tolerance set here to $\mathrm{tol_{nl}} = 10^{-6}$.

enabling the seamless port of the 2D E-VP algorithm to 3D, extending recent work by Duretz et al. (2019a) to tackle 3D configurations. We generate the 3D initial condition, a Gaussian random field with an exponential co-variance function, following the approach described in Räss et al. (2019b), available through the ParallelRandomFields.jl Julia package (Räss and Omlin, 2021). We perform 100 implicit loading steps using the accelerated PT method and execute the parallel Julia code relying on ParallelStencil.jl and ImplicitGlobalGrid.jl on 8 Tesla V100 on the *Volta* node. The scalability of the 3D E-VP implementation is comparable to the 2D version (Fig. 15a) with similar discrepancies as reported for the visco-elastic Stokes case (Fig. 7).

Both the 2D and 3D E-VP algorithms require only minor modifications of the visco-elastic Stokes solver discussed throughout this paper to account for brittle failure, deactivation of the serial viscous damper and viscoplastic regularisation without significantly affecting the convergence rate provided by the second-order method. These results support the robustness of the approach, predicting elasto-plastic deformation and capturing brittle failure categorised as a rather "stiff" problem which challenges the numerical solvers accordingly.

9 Discussion

The continuous development of many-core devices, with GPUs at the forefront, increasingly shapes the current and future computing landscape. The fact that GPUs and the latest multi-core CPUs turn classical workstations into personal supercomputers is exciting. Tackling previously impossible numerical resolutions or multi-physics solutions becomes feasible as a result of technical progress. However, the current chip design challenges legacy serial and non-local or sparse matrix-based algorithms, seeking solutions to partial differential equations. Naturally, solution strategies designed to specifically target efficient large-scale computations on supercomputers perform most efficiently on GPUs and recent multi-core CPUs, as the algorithms used are typically local and minimise memory accesses. Moreover, efficient strategies will not or only modestly rely on global communication and as a result, exhibit close to optimal scaling.

We introduced the PT method in light of, mostly, iterative type of methods such as dynamic relaxation and semi-iterative algorithms (see, e.g. Saad, 2020, for additional details). These classical methods, as well as the presented accelerated PT method, implement "temporal" damping by considering higher-order derivatives with respect to pseudo-time. This contrasts with multi-grid or multi-level methods, building upon a "spatial" strategy based on space discretisation properties to damp the low-frequency error modes. Multi-grid, or multi-level methods are widely used to achieve numerical solutions in analogous settings as described here (Brandt, 1977; Zheng et al., 2013). Furthermore, multi-grid methods may achieve convergence in $\mathrm{O}(n_x)$ iterations by employing an optimal relaxation algorithm (Bakhvalov, 1966).

Besides their scalable design, most iterative methods are challenged by configurations including heterogeneities and large contrasts in material parameters, motivated by typical applications to a variety of geodynamics problems (e.g. Baumgardner, 1985; Tackley, 1996; May et al., 2015; Kaus et al., 2016). Well-tuned robust multi-grid solvers may overcome these limitations at the cost of more complex implementations. Our systematic investigation results (Sect. 5.2) suggest, however, that the PT method performs promisingly

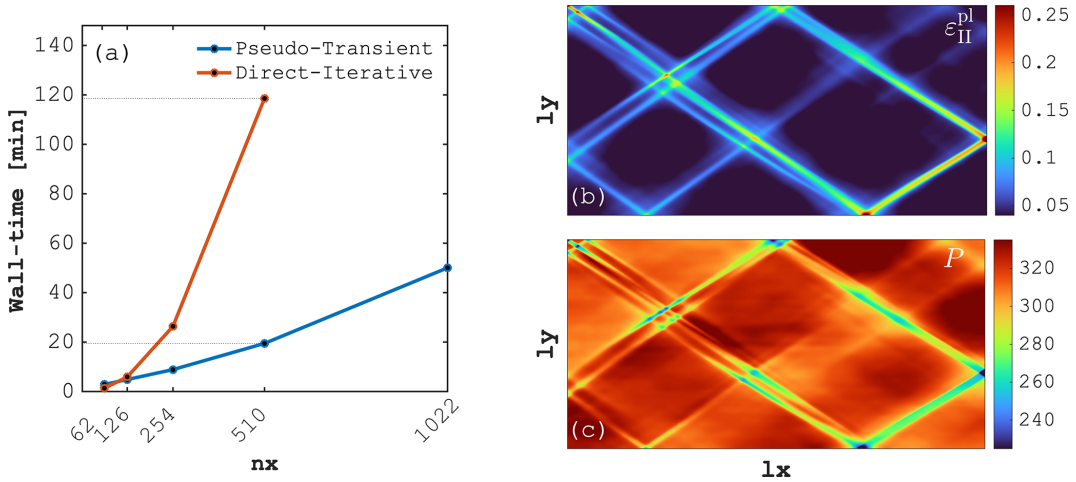

Figure 15. Performance comparison between the pseudo-transient (PT) and direct-iterative (DI) method resolving 2D shear-band formation out of a random noise cohesion field. **(a)** Wall time in minutes as a function of numerical grid resolution in x direction ($\sqrt{n_{\mathrm{dof}}}$). The dashed lines connect wall times for $n_x = 510$ from both solvers. Panels **(b)** and **(c)** depict total accumulated plastic strain $\varepsilon_{\mathrm{II}}^{\mathrm{pl}}$ and pressure [MPa], respectively, from the 1022×1022 resolution 2D PT-GPU simulation.

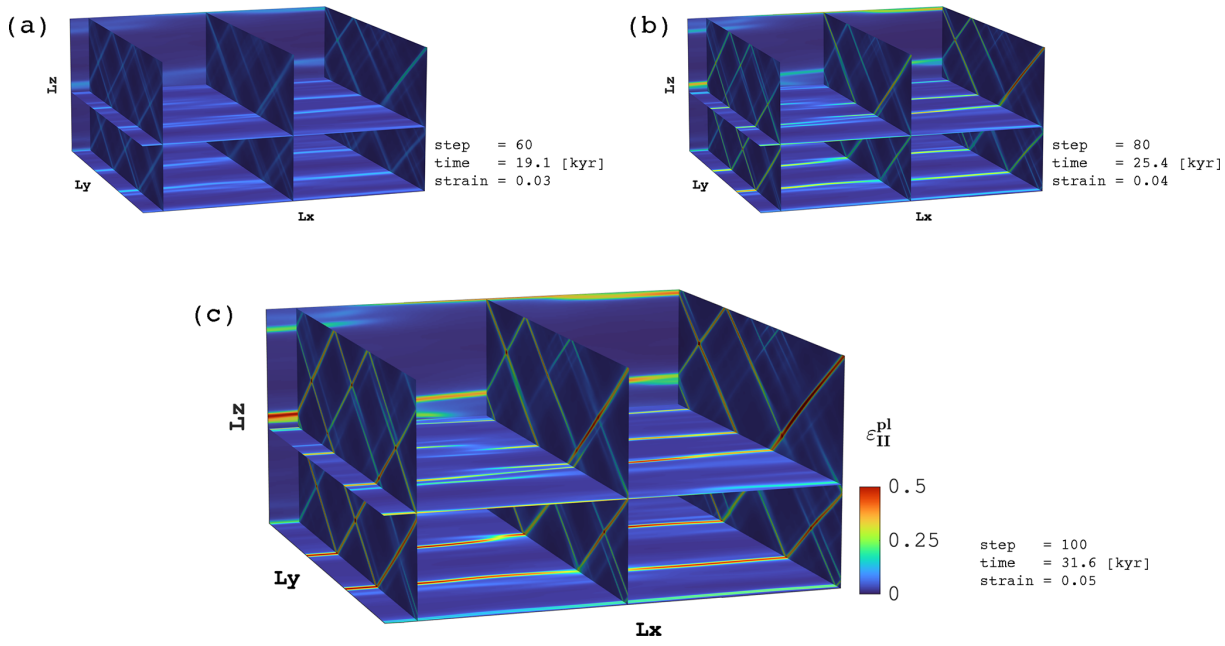

Figure 16. Total accumulated plastic strain $\varepsilon_{\mathrm{II}}^{\mathrm{pl}}$ distribution for a multi-GPU replica of the 2D calculation (Fig. 15b) resolving 3D shear-band formation out of a random noise cohesion field. The numerical resolution includes $378 \times 378 \times 192$ grid cells. Panels depict elasto-plastic shear-band formation after **(a)** 60 steps (corresponding 19.1 kyr and a strain of 0.03), **(b)** 80 steps (corresponding 25.4 kyr and a strain of 0.04) and **(c)** 100 steps (corresponding 31.6 kyr and a strain of 0.05).

well, and at no specific additional design efforts w.r.t. the basic accelerated PT method implementation. Beyond single-phase mechanical considerations, the accelerated PT method delivers accurate solutions in tightly coupled and nonlinear multi-physics problems (e.g. Duretz et al., 2019b; Räss et al., 2019a, 2020; Wang et al., 2021).

The ease of implementation is listed among the main advantages of the accelerated PT method over other more complex ones, e.g. multi-grid. Particularly, all nonlinearities can be relaxed within a unique iteration loop, as reported in the nonlinear diffusion results (Sect. 7.1). Often, due to the properties of the problem, the number of iterations does not exceed or is even less than that in the case of constant coeffi-

cients. Other methods, in which nonlinear iterations are performed separately from the linear solver, cannot take advantage of the details of the physical process to reduce iteration counts. Also, for significantly nonlinear problems, e.g. associated with plastic strain localisation, thermomechanical shear-heating or two-phase fluid focusing, the physical processes occur on multi-spatial scales. Thus, ensuring an accurate description of these multi-scale solutions requires a high-resolution computational grid. It may be challenging for a coarse level multi-grid solver to provide sufficient information in order to accurately resolve small-scale details that are only resolvable on the fine grid. Note that for analogous reasons, multi-grid methods are often used as a preconditioner for other iterative algorithms rather than the solution method.

Besides seeking optimal convergence of the algorithm, the implementation efficiency also favours the accelerated PT method; the approach is simple but efficient, making it possible to further implement advanced optimisations such as explicit shared memory usage and register handling. The choice of a Cartesian regular grid allows static and regular memory access patterns, resulting in access optimisation possibilities and balanced interprocess communications. Additionally, the absence of global reduction in the algorithm avoids severe bottlenecks. Finally, the amount of data transferred in the accelerated PT method is minimal, which allows achieving near-ideal scaling on distributed-memory systems, as reported in Sect. 7.3.2. Previous studies (e.g. Räss et al., 2018; Duretz et al., 2019b; Räss et al., 2019a, 2020) successfully implemented most of the algorithm designs presented here, although relying on a slightly different PT acceleration implementation (see Appendix B).

The main task in the design of PT methods is the estimation of optimal iteration parameters. For that, the spectral radius of the finite-difference operator is often approximated based on the Gershgorin circle theorem (Papadrakakis, 1981). In this paper, we propose an alternative derivation of the PT algorithm which is based entirely on a physical analogy. The analysis of the convergence rate can be carried out within the framework of the spectral analysis of continuous equations, rather than the finite-dimensional space discretisation. The advantage of this approach relies on the availability of a physical interpretation of the iteration parameters, as well as on a clear separation of physics and numerics. For example, we show that for visco-elastic Stokes flow (Sect. 2.4), the PT iteration parameters depend on the Reynolds number and the bulk-to-shear elastic modulus ratio. The stability of the iterative process is ensured by a choice of pseudo-physical material properties that is consistent with the conditions obtained on the basis of a von Neumann stability analysis.

The determination of the optimal iterative parameters is thereby reduced to the search for the optimal values of the dimensionless physical numbers that describe the properties of the underlying physical process. The addition of new physical processes, such as heat conduction, two-phase flow and chemical reactions will lead to the natural emergence of new dimensionless parameters. Since many physical processes have a similar or even identical mathematical description, it is expected that the derivation of the accelerated PT method for such processes can be carried out similarly to those already developed. In this paper, such a derivation is provided for several important processes, namely, the linear and nonlinear diffusion, diffusion–reaction, non-stationary diffusion, and the visco-elastic Stokes problem. The efficiency of the accelerated PT method is demonstrated for essentially nonlinear problems, as well as for the problems with large contrasts in the material properties.

Recently, Wang et al. (2021) studied fluid flow in deformable porous media using an analogous numerical integration scheme. They show that under certain assumptions, the equations governing the dynamics of such two-phase flows reduce to a "double-damped wave equation" system which is mathematically equivalent to the Eqs. (12) and (6) describing the diffusion–reaction process (Sect. 2.1.3). They also report the optimal parameters obtained by dispersion analysis of these equations. These parameters are formulated by Wang et al. (2021) in terms of dimensional physical parameters. Through appropriate rescaling it is possible to recover the non-dimensional form presented in Sect. 2.1.3. We believe that our derivation in terms of non-dimensional variables helps to reveal the analogy between seemingly different physical properties and facilitates reusing the derived iteration parameters for various applications. We provide analysis for the variety of physical processes, including incompressible Stokes flow, in a unified manner, filling some of the gaps missing in previous studies.

The scalability of the accelerated PT method as a function of numerical resolution permits the prediction of the total iteration count, here for the nonlinear diffusion and the visco-elastic Stokes in 3D. The weak-scaling benchmark results provide the time per iteration as a function of the numerical resolution. Combining this information, it is possible to predict the time-to-solution or wall time (Table 2) needed to resolve nonlinear diffusion and visco-elastic Stokes flow on 6632^3 and 4955^3 grid cells, respectively, on 2197 Nvidia Tesla P100 GPUs on the *Piz Daint* supercomputer at CSCS. Single-GPU problem size scaling results for different GPU architectures further permit the extrapolation of these wall times to the Nvidia Ampere or Volta architecture for example, as time per iteration is directly proportional to the effective memory throughput T_{eff}. There is a $3.0\times$ and $1.5\times$ increase in T_{eff} on the A100 and V100 compared to the P100 architecture, respectively, resulting in wall time in the order of 2.5 min and 2.4 h on a A100 powered system and of 5 min and 4.8 h on a V100 powered system, for the nonlinear diffusion and the visco-elastic Stokes solver, respectively.

In practical applications, the patterns of the flow may change drastically throughout the simulation owing to the spontaneous flow localisation or evolution of the interface between immiscible phases with significantly different prop-

Table 2. Wall-time prognostic for resolving the nonlinear diffusion and the visco-elastic Stokes 3D Julia multi-GPU applications on 2197 (13^3) Nvidia Tesla P100 GPUs on the *Piz Daint* supercomputer at CSCS.

Application	GPU	n_x local	n_x global	$n_{\mathrm{dof}}^{\mathrm{tot}}$	t/iter (s)	$n_{\mathrm{iter}}^{\mathrm{tot}}$	wall time
Nonlinear diffusion 3D	Tesla P100	512	6632	0.292×10^{12}	0.0348	12'932	7.5 min
Visco-elastic Stokes 3D	Tesla P100	383	4955	1.23×10^{12}	0.0644	408'788	7.31 h

erties. It is a requirement for the numerical method to be robust with respect to such changes. The iterative algorithm is expected to converge even in extreme cases, e.g. in the presence of sharp gradients across material properties, and the iteration parameters should be insensitive to arbitrary changes in the internal geometry. We present a parametric study to assess the robustness of the accelerated PT method for typical building blocks for geophysical applications. We considered shear- and buoyancy-driven flows with multiple randomly distributed inclusions in a viscous matrix as proxies for more realistic problem formulations. We show that our method is capable of modelling flows with viscosity contrasts up to 9 orders of magnitude. The values of optimal iteration parameters obtained by the means of systematic simulation runs do not change significantly for a wide range of material properties and internal configurations of the computational domain. We observe the significant slowdown in convergence for viscosity contrasts larger than 5 orders of magnitude in some of the considered cases. These results are expected, given the ill-conditioned problem and thus motivate development of a scalable preconditioner suitable for massively parallel GPU workloads. The application of a robust preconditioner, with reference to previous discussions, may help to partly alleviate slow convergence. However, for viscosity contrasts of 6 orders of magnitude and more, a significant increase in the number of iterations may be legitimate (May et al., 2015).

The numerical application to resolve shear-banding in elasto-viscoplastic media in 3D supports the versatility and the robustness of the presented approach putting emphasis on successfully handling complex rheology. These examples complement recent studies employing the accelerated PT method to resolve spontaneous localisation owing to multi-physics coupling (Räss et al., 2018, 2019a, 2020; Duretz et al., 2019b) and entire adjoint-based inversion frameworks (Reuber et al., 2020). The Sect. B of the Appendix provides connections of the presented analysis with previous work.

10 Conclusions

The current HPC landscape redefines the rules governing applications' performance where the multi-core processors' massive parallelism imposes a memory-bound situation. Our work shows that simple dynamic relaxation schemes can be employed efficiently to solve parabolic and nonlinear systems, upon transforming the equations into hyperbolic expressions. The transformation includes the addition of physics-inspired terms which enable the possibility to retrieve optimal parameters that minimise the iteration counts. The optimisation procedure includes von Neumann stability and dispersion analysis for the pseudo-physical parameters. Moreover, the conciseness of the accelerated PT approach permits the applications to execute at effective memory throughput rate the approaching memory copy rates (a theoretical upper bound) of latest GPUs. Further, hiding communication behind computations permits the achievement of parallel efficiency of more than 96 % on various distributed-memory systems and up to 2197 GPUs. The physics we selected for the numerical experiments represent key building blocks to further tackle various multi-physics coupling, usually the combination of mechanical and diffusive processes. Our systematic results on the multi-inclusion setup with huge viscosity contrasts provide some preliminary results assessing the robustness of the accelerated PT method, which we further employ to resolve shear-band formation in 3D as a result of plastic yielding in E-VP materials. Our study paves the way for resolving coupled and nonlinear multi-physics applications in natural sciences and engineering on extremely high resolutions on the next generation of exascale-capable supercomputers, revamping elegant iterative techniques and implementing them with the portable Julia language.

Appendix A: Dispersion analysis

A1 First-order pseudo-transient (PT) diffusion

Let the total pseudo-time required to reach convergence be τ_{tot}. To estimate τ_{tot} and the number of iterations in the numerical implementation of the method, we consider the deviation ϵ of H from the exact solution to Eq. (4). Linearity of the problem makes it possible to reformulate Eq. (5) in terms of ϵ:

$$\frac{\partial \epsilon}{\partial \tau} = \nu \frac{\partial^2 \epsilon}{\partial x^2}, \tag{A1}$$

where $\nu = D/\tilde{\rho}$. Equation (A1) is subject to homogeneous boundary conditions: $\epsilon(0,\tau) = \epsilon(L,\tau) = 0$.

We study the convergence rate of the PT method by performing the dispersion analysis of Eq. (A1). The general solution to the Eq. (A1) with homogeneous Dirichlet boundary

conditions has the following form:

$$\epsilon(x,\tau)=\sum_{k=1}^{\infty}E_k\exp\left(\frac{-\lambda_k V_\mathrm{d}\tau}{L}\right)\sin\left(\frac{\pi k x}{L}\right), \tag{A2}$$

where $V_\mathrm{d}=\nu/L$ is the characteristic velocity scale of diffusion, k is the wavenumber, E_k and λ_k are the initial amplitude and the decay rate of the wave with wavenumber k, respectively. Only values of $k\geq 1$ are accounted for in Eq. (A2), because for other values of k, homogeneous boundary conditions are not satisfied.

We are interested in the exponential decay rate values λ_k, because they indicate the total time it takes for the error component with wavelength πk to vanish. It is sufficient to consider only one typical term from the solution Eq. (A2):

$$\epsilon_k(x,\tau)=E_k\exp\left(\frac{-\lambda_k V_\mathrm{d}\tau}{L}\right)\sin\left(\frac{\pi k x}{L}\right). \tag{A3}$$

Substituting Eq. (A3) into Eq. (A1) gives

$$\lambda_k=\pi^2k^2. \tag{A4}$$

The dispersion relation Eq. (A4) indicates that the harmonics with higher wavenumbers decay faster than the ones with smaller wavenumbers. Therefore, the total time τ_tot to convergence is controlled by the component of ϵ with the smallest wavenumber $k=1$. If the required non-dimensional accuracy, i.e. upper bound on ϵ, is ϵ_tol, then τ_tot can be obtained by solving Eq. (A3) for τ at $k=1$ and $x=L/2$:

$$\tau_\mathrm{tot}=\frac{L^2}{\pi^2\nu}\ln\left(\frac{E_1}{\epsilon_\mathrm{tol}}\right). \tag{A5}$$

The stability analysis of Eq. (5) in discretised form suggests that the maximal time step $\Delta\tau$ allowed by the explicit integration scheme is proportional to the grid spacing Δx squared: $\Delta\tau=\Delta x^2/\nu/2$. If the computational grid is uniform, as it is the case in this paper, $L=n_x\Delta x$, where n_x is the number of grid cells. Assuming $\tau_\mathrm{tot}=n_\mathrm{it}\Delta\tau$, where n_it is the number of PT iterations required for convergence, and substituting expressions for L and τ_tot into the Eq. (A5), we obtain

$$n_\mathrm{it}=2\frac{n_x^2}{\pi^2}\ln\left(\frac{E_1}{\epsilon_\mathrm{tol}}\right). \tag{A6}$$

Thus, the number of pseudo-time iterations required for convergence must be proportional to n_x^2.

A2 Accelerated PT diffusion

Similar to the previous case, we reformulate the problem in terms of deviation from the exact solution and consider the typical term from the Fourier series expansion:

$$\epsilon_k(x,\tau)=E_k\exp\left(\frac{-\lambda_k V_\mathrm{p}\tau}{L}\right)\sin\left(\frac{\pi k x}{L}\right). \tag{A7}$$

Substituting Eq. (A7) into Eq. (7) gives the following dispersion relation:

$$\lambda_k^2-Re\lambda_k+\pi^2k^2=0, \tag{A8}$$

where $Re=LV_\mathrm{p}/\nu=\sqrt{L^2/\nu/\theta_\mathrm{r}}$. The non-dimensional parameter Re can be interpreted as a numerical Reynolds number, or as the inverse of a numerical Deborah number's square root, characterising the ratio between relaxation time θ_r and characteristic time of diffusion L^2/ν.

Solving the Eq. (A8) for λ_k yields

$$\lambda_{k_{1,2}}=\frac{Re\pm\sqrt{Re^2-4\pi^2k^2}}{2}. \tag{A9}$$

Depending on values of Re and k, λ_k can be complex. The imaginary part of λ_k, $\Im(\lambda_k)$, contributes to the oscillatory behaviour of the solution, while the real part $\Re(\lambda_k)$ controls the exponential decay of deviation ϵ. The minimum of $\Re(\lambda_k)$ between the two roots of Eq. (A9) indicates the decay rate of the entire solution. This minimum reaches maximal value when both roots of Eq. (A9) are equal, i.e. when the discriminant of Eq. (A9) $D=Re^2-4\pi^2k^2=0$. Therefore, $Re=\pm 2\pi k$, but negative Re leads to exponential growth of the $\exp(-\lambda_k V_\mathrm{p}t/L)$ term in the solution Eq. (A7), which is non-physical. Finally, since choosing iteration parameters for each wavelength independently is not realistic, we chose Re such that the longest wave with wavenumber $k=0$ is damped most effectively. The optimal value of Re is then given by

$$Re_\mathrm{opt}=2\pi. \tag{A10}$$

Total pseudo-time τ_tot required for convergence with tolerance ϵ_tot is determined in the same way as in the previous case:

$$\tau_\mathrm{tot}=\frac{L}{\pi V_\mathrm{p}}\ln\left(\frac{E_1}{\epsilon_\mathrm{tol}}\right). \tag{A11}$$

The stability analysis of the damped wave equation in discretised form suggests that in a certain range of values of θ_r, the maximal time step $\Delta\tau=\Delta x/V_\mathrm{p}$ (Alkhimenkov et al., 2021a). The number of pseudo-time iterations required for convergence n_it can then be estimated as

$$n_\mathrm{it}=\frac{n_x}{\pi}\ln\left(\frac{E_1}{\epsilon_\mathrm{tol}}\right), \tag{A12}$$

and proportional to n_x in the accelerated scheme.

A3 Diffusion–reaction

Reformulating Eq. (13) in terms of the difference ϵ between the exact steady-state solution and the PT solution and substituting it into Eq. (A7), one obtains the following dispersion relation:

$$\lambda_k^2-\left(\frac{Da}{Re}+Re\right)\lambda_k+Da+\pi^2k^2=0. \tag{A13}$$

Introducing the auxiliary parameter $\widehat{Re} = Da/Re + Re$, it can be seen that the dispersion relation in Eq. (A13) is now similar to the one reported in Eq. (A8). The solution for λ_k is now given by

$$\lambda_{k_{1,2}} = \frac{\widehat{Re} \pm \sqrt{\widehat{Re}^2 - 4\pi^2 k^2 - 4Da}}{2}. \tag{A14}$$

The resulting optimal value is $\widehat{Re} = 2\sqrt{\pi^2 + Da}$, and the optimal value for Re is obtained by solving $\widehat{Re}(Re)$ for Re:

$$Re_{\text{opt}} = \pi + \sqrt{\pi^2 + Da}. \tag{A15}$$

As expected, in the limit of $Da \to 0$, i.e. when the process becomes diffusion-dominated, the optimal value of the parameter Re, determined by Eq. (A15), coincides with the value given by Eq. (A10) in the case of a purely diffusive process.

The number of iterations required to converge to a tolerance ϵ_{tol} is

$$n_{\text{it}} = \frac{n_x}{\sqrt{\pi^2 + Da}} \ln\left(\frac{E_1}{\epsilon_{\text{tol}}}\right). \tag{A16}$$

Interestingly, the iteration count estimate given by Eq. (A16) decreases proportionally to $\sqrt{Da}$. Therefore, the number of iterations for a diffusion–reaction process will always be lower than for a pure diffusion process if $Da > 0$.

A4 Incompressible viscous Stokes

The system of Eqs. (25)–(27) can be reduced to a single equation for velocity v_x:

$$\mu_s \frac{\tilde{\rho}}{\widetilde{G}} \frac{\partial^3 v_x}{\partial \tau^3} - \mu_s \left(\frac{\widetilde{K}}{\widetilde{G}} + 2\right) \frac{\partial^3 v_x}{\partial \tau \partial x^2} + \tilde{\rho} \frac{\partial^2 v_x}{\partial \tau^2} = \widetilde{K} \frac{\partial^2 v_x}{\partial x^2}. \tag{A17}$$

Following the established procedure, we reformulate Eq. (A17) in terms of deviation ϵ from the exact solution, and consider typical terms in the Fourier series expansion of ϵ:

$$\epsilon_k(x, \tau) = E_k \exp\left(\frac{-\lambda_k V_{\text{p}} \tau}{L}\right) \sin\left(\frac{\pi k x}{L}\right). \tag{A18}$$

Substituting Eq. (A18) into Eq. (A17), we obtain the following dispersion relation:

$$\lambda_k^3 - Re\lambda_k^2 + \pi^2 k^2 (r+1)\lambda_k - \pi^2 k^2 r Re = 0. \tag{A19}$$

Depending on the values of the coefficients, the dispersion relation in Eq. (A19) would have either one real root and two complex conjugate roots or three real roots.

For analysis, it is useful to recast the dispersion relation in Eq. (A19) to depressed form $x^3 + px + q$ applying a change of variables $\alpha_k = \lambda_k - Re/(3(r+2))$:

$$\alpha_k^3 + \left[\pi^2 k^2 - \frac{Re^2}{3(r+2)^2}\right]\alpha_k + \frac{2Re}{9(r+2)}\left[\pi^2 k^2 \left(1 + \frac{1-4r}{r+2}\right) - \frac{Re^2}{3(r+2)^2}\right] = 0. \tag{A20}$$

The discriminant of the depressed cubic equation is $x^3 + px + q$ is $-4p^3 - 27q^2$. Setting the discriminant of Eq. (A20) to zero yields

$$r Re^4 - (1 - 2r^2 + 10r)(r+2)\pi^2 k^2 Re^2 + (r+2)^5 \pi^4 k^4 = 0. \tag{A21}$$

Equation (A21) is biquadratic w.r.t. Re. We denote the discriminant of Eq. (A21) by D. For Re to be real, D must be non-negative:

$$D = \pi^4 k^4 (r+2)^2 (1-4r)^3 \geq 0 \Longleftrightarrow r \leq \frac{1}{4}. \tag{A22}$$

By definition, r is the ratio between the bulk and shear modulus, therefore, r must be positive. Thus, $r \in (0; 1/4]$.

By solving Eq. (A21), we obtain the relation between Re and r:

$$Re_{1,2} = \sqrt{\frac{(r+2)\left(1 - 2r^2 + 10r \pm \sqrt{(1-4r)^3}\right)}{2r}} \pi k. \tag{A23}$$

Only the positive branch of the solution is taken in Eq. (A23) because Re must be positive.

When the depressed cubic equation $x^3 + px + q$ has multiple roots, the simple root is $x_1 = 3q/p$, and the double root is $x_2 = x_3 = -3q/(2p)$. Substituting expressions for q and p from the dispersion relation in Eq. (A20), and changing variables back to λ_k from α_k, one obtains:

$$\lambda_{k_1} = \frac{Re}{r+2}\left(1 + \frac{2(r+2)(1-4r)\pi^2 k^2}{3(r+2)^2 \pi^2 k^2 - Re^2}\right), \tag{A24}$$

$$\lambda_{k_{2,3}} = -Re \frac{(1-4r)\pi^2 k^2}{3(r+2)^2 \pi^2 k^2 - Re^2}. \tag{A25}$$

Substituting Eq. (A23) into Eq. (A24), one obtains roots of the dispersion relation in Eq. (A19) as a function of parameter r. The roots depend monotonously on r for all choices of Re (Fig. A1). The minimal root reaches its maximum value at $r_{\text{opt}} = 1/4$. The corresponding value of Re is then $Re_{\text{opt}} = 9\sqrt{3}\pi/4$ for $k = 1$.

Appendix B: Connection to the previous work

It is useful to provide an analogy between the presented analysis and some previous studies, namely, PT continuation model by Räss et al. (2018, 2019a, 2020); Duretz et al.

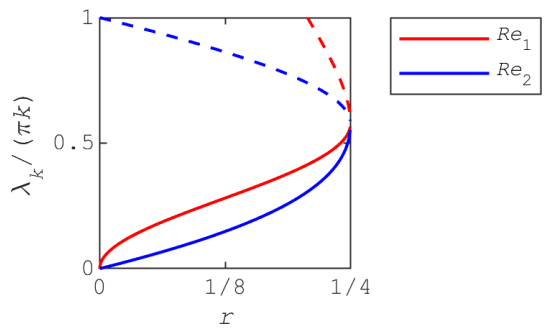

Figure A1. Roots of the dispersion relation λ_k for two branches of the numerical parameter Re. Solid and dashed lines show maximal and minimal roots, respectively.

(2019b), and early work by Frankel (1950). For this comparison, we consider only a stationary diffusion process with $D = \text{const}$.

We reformulate the damped wave equation, Eq. (7), as a first-order system introducing R, the pseudo-time derivative of H:

$$\frac{\partial H}{\partial \tau} = R, \tag{B1}$$

$$\theta_\mathrm{r} \frac{\partial R}{\partial \tau} + R = \frac{1}{\tilde{\rho}} \nabla_k D \nabla_k H. \tag{B2}$$

In all mentioned studies the numerical discretisation of Eqs. (B1) and (B2) were considered. Let f^k be the finite-difference operator approximating the right-hand side of Eq. (B2) at time step k. Using a forward Euler scheme for the time integration, one obtains

$$\frac{H_i^k - H_i^{k-1}}{\Delta \tau} = R_i^k, \tag{B3}$$

$$\theta_\mathrm{r} \frac{R_i^k - R_i^{k-1}}{\Delta \tau} + R_i^{k-1} = f^k. \tag{B4}$$

Let $g_i^k = \theta_\mathrm{r} R_i^k / \Delta \tau$. Rearranging Eqs. (B3) and (B4) to formulate the update-rules for H_i^k and g_i^k:

$$H_i^k = H_i^{k-1} + \frac{\Delta \tau^2}{\theta_\mathrm{r}} g_i^k, \tag{B5}$$

$$g_i^k = f_i^k + \left(1 - \frac{\Delta \tau}{\theta_\mathrm{r}}\right) g_i^{k-1}, \tag{B6}$$

and using the definitions of θ_r and $\Delta \tau$ reported by Eqs. (10) and (8):

$$\frac{\Delta \tau}{\theta_\mathrm{r}} = \tilde{C} \frac{ReL}{\Delta x} = \tilde{C} \frac{Re}{n_x}, \tag{B7}$$

it is evident that, if combined, Eqs. (B5), (B6) and (B7) are equivalent to the formulation of PT continuation method presented in Räss et al. (2018). Substituting Eq. (B6) into Eq. (B5) and expressing g_i^{k-1} in terms of H_i^{k-1} and H_i^{k-2} yields

$$H_i^k = H_i^{k-1} + \frac{\Delta \tau^2}{\theta_\mathrm{r}} f^k + \left(1 - \frac{\Delta \tau}{\theta_\mathrm{r}}\right) \left(H_i^{k-1} - H_i^{k-2}\right), \tag{B8}$$

which is equivalent to the second-order Richardson method proposed in Frankel (1950).

Code availability. The various software developed and used in the scope of this study is licensed under MIT License. The latest versions of the code is available from GitHub at: https://github.com/PTsolvers/PseudoTransientDiffusion.jl (last access: 16 May 2022) and https://github.com/PTsolvers/PseudoTransientStokes.jl (last access: 16 May 2022). Past and future versions of the software are available from a permanent DOI repository (Zenodo) at: https://doi.org/10.5281/zenodo.6553699 (Räss and Utkin, 2022a) and https://doi.org/10.5281/zenodo.6553714 (Räss and Utkin, 2022b). Both the PseudoTransientDiffusion.jl and the PseudoTransientStokes.jl repositories contain most Julia multi-xPU code examples implementing the accelerated PT method and can be readily used to reproduce the figures of the paper. The codes are written using the Julia programming language. Scaling figures can readily be reproduced using the available data, while all other figures require first executing the Julia scripts to produce the expected data. Refer to the repositories' `README` for additional information.

Author contributions. LR designed the original study, developed codes and algorithms, realised the Julia at scale implementation, performed scaling and performance benchmarks, created figures and edited the manuscript. IU developed codes and algorithms, performed the dispersion analysis and the parametric study, created figures and edited the manuscript. TD contributed to the original study design, realised the desktop-scale computation benchmark, performed the E-VP model investigations, created figures and edited the manuscript. SO developed the building blocks for implementing Julia at scale and the effective memory throughput metric, deployed Julia on Piz Daint, provided scalable architecture-agnostic PDE solver implementation and edited the manuscript. YYP provided early work on accelerated PT methods, helped out with the dispersion analysis, contributed to methodological advances to allow PDE solver implementations on GPUs.

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 in published maps and institutional affiliations.

Acknowledgements. We thank Boris Kaus, Stefan Schmalholz and Mauro Werder for stimulating discussions. We thank Lawrence H. Wang, Boris Kaus and an anonymous reviewer for their suggestions to improve the manuscript. We acknowledge the

Swiss Geocomputing Centre, University of Lausanne, for computing resources on the *Octopus* supercomputer and are grateful to Philippe Logean for continuous technical support.

Financial support. This research has been supported by the Swiss University Conference and the Swiss Council of Federal Institutes of Technology through the Platform for Advanced Scientific Computing (PASC) program. This research has been supported by the Swiss National Supercomputing Centre (CSCS) (project ID c23). This research has been supported by the Russian Science Foundation (grant no. 19-47-02015). Publisher's note: the article processing charges for this publication were not paid by a Russian or Belarusian institution.

Review statement. This paper was edited by James Kelly and reviewed by Lawrence Hongliang Wang, Boris Kaus, and one anonymous referee.

References

Alamatian, J.: A new formulation for fictitious mass of the Dynamic Relaxation method with kinetic damping, Comput. Struct., 90–91, 42–54, https://doi.org/10.1016/j.compstruc.2011.10.010, 2012.

Alkhimenkov, Y., Khakimova, L., and Podladchikov, Y.: Stability of discrete schemes of Biot's poroelastic equations, Geophys. J. Int., 225, 354–377, https://doi.org/10.1093/gji/ggaa584, 2021a.

Alkhimenkov, Y., Räss, L., Khakimova, L., Quintal, B., and Podladchikov, Y.: Resolving Wave Propagation in Anisotropic Poroelastic Media Using Graphical Processing Units (GPUs), J. Geophys. Res.-Sol. Ea., 126, 7, https://doi.org/10.1029/2020JB021175, 2021b.

Bakhvalov, N. S.: On the convergence of a relaxation method with natural constraints on the elliptic operator, USSR Comp. Math. Math.+, 6, 101–135, https://doi.org/10.1016/0041-5553(66)90118-2, 1966.

Barnes, M. R.: Form Finding and Analysis of Tension Structures by Dynamic Relaxation, International Journal of Space Structures, 14, 89–104, https://doi.org/10.1260/0266351991494722, 1999.

Baumgardner, J. R.: Three-dimensional treatment of convective flow in the earth's mantle, J. Stat. Phys., 39, 501–511, https://doi.org/10.1007/BF01008348, 1985.

Besard, T., Foket, C., and De Sutter, B.: Effective Extensible Programming: Unleashing Julia on GPUs, IEEE T. Parall. Distr., 30, 827–841, https://doi.org/10.1109/TPDS.2018.2872064, 2018.

Besard, T., Churavy, V., Edelman, A., and De Sutter, B.: Rapid software prototyping for heterogeneous and distributed platforms, Adv. Eng. Softw., 132, 29–46, 2019.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B.: Julia: A fresh approach to numerical computing, SIAM Rev., 59, 65–98, https://doi.org/10.1137/141000671, 2017.

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.

Byrne, S., Wilcox, L. C., and Churavy, V.: MPI.jl: Julia bindings for the Message Passing Interface, Proceedings of the JuliaCon Conferences, 1, 68, https://doi.org/10.21105/jcon.00068, 2021.

Cassell, A. C. and Hobbs, R. E.: Numerical stability of dynamic relaxation analysis of non-linear structures, Int. J. Numer. Meth. Eng., 10, 1407–1410, https://doi.org/10.1002/nme.1620100620, 1976.

Chester, M.: Second sound in solids, Phys. Rev., 131, 2013–2015, https://doi.org/10.1103/PhysRev.131.2013, 1963.

Costa, A.: Permeability-porosity relationship: A reexamination of the Kozeny-Carman equation based on a fractal pore-space geometry assumption, Geophys. Res. Lett., 33, L02318, https://doi.org/10.1029/2005GL025134, 2006.

Cox, S. and Zuazua, E.: The rate at which energy decays in a damped string, Commun. Part. Diff. Eq., 19, 213–243, https://doi.org/10.1080/03605309408821015, 1994.

Cundall, A. P.: Explicit finite differnce method in geomechanics, Second Int. Conf. Numerical Methods in Geomechanics, Blacksburg, 1976, 1, 132–150, 1976.

Duretz, T., de Borst, R., and Le Pourhiet, L.: Finite Thickness of Shear Bands in Frictional Viscoplasticity and Implications for Lithosphere Dynamics, Geochem. Geophy. Geosy., 20, 5598–5616, https://doi.org/10.1029/2019GC008531, 2019a.

Duretz, T., Räss, L., Podladchikov, Y., and Schmalholz, S.: Resolving thermomechanical coupling in two and three dimensions: spontaneous strain localization owing to shear heating, Geophys. J. Int., 216, 365–379, https://doi.org/10.1093/gji/ggy434, 2019b.

Eager, D. L., Zahorjan, J., and Lazowska, E. D.: Speedup versus efficiency in parallel systems, IEEE T. Comput., 38, 408–423, https://doi.org/10.1109/12.21127, 1989.

Frankel, S. P.: Convergence rates of iterative treatments of partial differential equations, Mathematical Tables and Other Aids to Computation, 4, 65–75, 1950.

Fuhrer, O., Chadha, T., Hoefler, T., Kwasniewski, G., Lapillonne, X., Leutwyler, D., Lüthi, D., Osuna, C., Schär, C., Schulthess, T. C., and Vogt, H.: Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0, Geosci. Model Dev., 11, 1665–1681, https://doi.org/10.5194/gmd-11-1665-2018, 2018.

Gaitonde, A. L.: A dual-time method for two-dimensional unsteady incompressible flow calculations, Int. J. Numer. Meth. Eng., 41, 1153–1166, https://doi.org/10.1002/(SICI)1097-0207(19980330)41:6<1153::AID-NME334>3.0.CO;2-9, 1998.

Gustafson, J. L.: Reevaluating Amdahl's law, Commun. ACM, 31, 532–533, https://doi.org/10.1145/42411.42415, 1988.

Hackbusch, W.: Multi-Grid Methods and Applications, vol. 4, Springer Series in Computational Mathematics, Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-662-02427-0, 1985.

Hoemmen, M. F.: Communication-avoiding Krylov subspace methods, PhD thesis, EECS Department, University of California, Berkeley, http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-37.html (last access: 16 May 2022), 2010.

Jordan, P. M. and Puri, A.: Digital signal propagation in dispersive media, J. Appl. Phys., 85, 1273–1282, https://doi.org/10.1063/1.369258, 1999.

Kaus, B. J. P., Popov, A. A., Baumann, T. S., Püsök, A. E., Bauville, A., Fernandez, N., and Collignon, M.: Forward and Inverse Modelling of Lithospheric Deformation on Geological Timescales, in: NIC Symposium 2016, Jülich, Germany, 11–

12 February 2016, edited by: Binder, K., Müller, M., Kremer, A., and Schnurpfeil, A., 48, 299–307, Forschungszentrum Jülich, Jülich, 2016.

Kilic, B. and Madenci, E.: Structural stability and failure analysis using peridynamic theory, Int. J. NonLin. Mech., 44, 845–854, https://doi.org/10.1016/j.ijnonlinmec.2009.05.007, 2009.

Kumar, V., Grama, A., Gupta, A., and Karypis, G.: Introduction to parallel computing, Benjamin/Cummings Redwood City, CA, 110, ISBN 0805331700, 1994.

Mandal, J., Sonawane, C., Iyer, A., and GosaviInamdar, S.: Incompressible flow computations over moving boundary using a novel upwind method, Comput. Fluid., 46, 348–352, 2011.

Maxwell, J. C.: IV. On the dynamical theory of gases, Philos. T. R. Soc. Lond., 157, 49–88, https://doi.org/10.1098/rstl.1867.0004, 1867.

May, D., 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.

McCalpin, J. D.: Memory bandwidth and machine balance in current high performance computers, IEEE computer society technical committee on computer architecture (TCCA) newsletter, 2, 19–25, 1995.

McKee, S., Tomé, M., Ferreira, V., Cuminato, J., Castelo, A., Sousa, F., and Mangiavacchi, N.: The MAC method, Comput. Fluid., 37, 907–930, https://doi.org/10.1016/j.compfluid.2007.10.006, 2008.

Omlin, S.: Development of massively parallel near peak performance solvers for three-dimensional geodynamic modelling, PhD thesis, University of Lausanne, https://doi.org/10.48550/ARXIV.2207.08716, 2017.

Omlin, S. and Räss, L.: ImplicitGlobalGrid.jl, https://github.com/eth-cscs/ImplicitGlobalGrid.jl (last access: 16 May 2022), 2021a.

Omlin, S. and Räss, L.: ParallelStencil.jl, https://github.com/omlins/ParallelStencil.jl (last access: 16 May 2022), 2021b.

Omlin, S., Räss, L., and Podladchikov, Y.: HPC.m – the MATLAB HPC Compiler and its Use for Solving 3D Poromechanics on Supercomputers, in: Platform for Advanced Scientific Computing Conference (PASC15), 1–3 June 2015, Zurich, Switzerland, ETH Zurich, Zurich, Switzerland, 2015a.

Omlin, S., Räss, L., and Podladchikov, Y.: From MATLAB to Supercomputing: the MATLAB HPC-Compiler, in: 2nd Frontiers in Computational Physics: Energy Sciences Conference 2015, 3–5 June 2015, Zurich, Switzerland, ETH Zurich, Zurich, Switzerland, 2015b.

Otter, J.: Computations for prestressed concrete reactor pressure vessels using dynamic relaxation, Nuclear Structural Engineering, 1, 61–75, https://doi.org/10.1016/0369-5816(65)90097-9, 1965.

Otter, J. R. H., Cassell, A. C., and Hobbs, R. E.: Dynamic relaxation, 35, 633–656, https://doi.org/10.1680/IICEP.1966.8604, 1966.

Papadrakakis, M.: A method for the automatic evaluation of the dynamic relaxation parameters, Comput. Method Appl. M., 25, 35–48, https://doi.org/10.1016/0045-7825(81)90066-9, 1981.

Pascal, H.: Pressure wave propagation in a fluid flowing through a porous medium and problems related to interpretation of Stoneley's wave attenuation in acoustical well logging, Int. J. Eng. Sci., 24, 1553–1570, https://doi.org/10.1016/0020-7225(86)90163-1, 1986.

Patankar, S.: Numerical heat transfer and fluid flow, CRC press, https://doi.org/10.1201/9781482234213, 1980.

Podladtchikov, I. and Podladchikov, Y. Y.: Memory Bound Wave Propagation at Hardware Limit, http://on-demand.gputechconf.com/gtc/2013/video/S3176-Memory-Bound-Wave-Propagation-Hardware-Limit.flv (last access: 16 May 2022), 2013.

Poliakov, A. N., Cundall, P. A., Podladchikov, Y. Y., and Lyakhovsky, V. A.: An explicit inertial method for the simulation of viscoelastic flow: an evaluation of elastic effects on diapiric flow in two- and three- layers models, Flow and creep in the solar system: observations, modeling and theory, NATO ASI Series, vol. 391, Springer, Dordrecht, 175–195, https://doi.org/10.1007/978-94-015-8206-3_12, 1993.

Poliakov, A. N. B., Herrmann, H. J., Podladchikov, Y. Y., and Roux, S.: Fractal plastic shear bands, Fractals, 2, 567–581, https://doi.org/10.1142/s0218348x9400079x, 1994.

Ramesh, G. and Krishnamoorthy, C. S.: Post-buckling analysis of structures by dynamic relaxation, Int. J. Numer. Meth. Eng., 36, 1339–1364, https://doi.org/10.1002/nme.1620360806, 1993.

Räss, L. and Omlin, S.: ParallelRandomFields.jl, https://github.com/luraess/ParallelRandomFields.jl (last access: 16 May 2022), 2021.

Räss, L. and Utkin, I.: PTsolvers/PseudoTransientDiffusion.jl: PseudoTransientDiffusion.jl 1.0.0, Zenodo [code], https://doi.org/10.5281/zenodo.6553699, 2022a.

Räss, L. and Utkin, I.: PTsolvers/PseudoTransientStokes.jl: PseudoTransientStokes.jl 1.0.0, Zenodo [code], https://doi.org/10.5281/zenodo.6553714, 2022b.

Räss, L., Duretz, T., Podladchikov, Y. Y., and Schmalholz, S. M.: M2Di: Concise and efficient MATLAB 2-D Stokes solvers using the Finite Difference Method, Geochem. Geophy. Geosy., 18, 755–768, https://doi.org/10.1002/2016GC006727, 2017.

Räss, L., Simon, N. S., and Podladchikov, Y. Y.: Spontaneous formation of fluid escape pipes from subsurface reservoirs, Sci. Rep.-UK, 8, 1–11, 2018.

Räss, L., Duretz, T., and Podladchikov, Y. Y.: Resolving hydro-mechanical coupling in two and three dimensions: Spontaneous channelling of porous fluids owing to decompaction weakening, Geophys. J. Int., 218, 1591–1616, https://doi.org/10.1093/gji/ggz239, 2019a.

Räss, L., Kolyukhin, D., and Minakov, A.: Efficient parallel random field generator for large 3-D geophysical problems, Comput. Geosci., 131, 158–169, https://doi.org/10.1016/j.cageo.2019.06.007, 2019b.

Räss, L., Omlin, S., and Podladchikov, Y. Y.: Resolving Spontaneous Nonlinear Multi-Physics Flow Localization in 3-D: Tackling Hardware Limit, https://developer.nvidia.com/gtc/2019/video/S9368 (last access: 16 May 2022), 2019c.

Räss, L., Licul, A., Herman, F., Podladchikov, Y. Y., and Suckale, J.: Modelling thermomechanical ice deformation using an implicit pseudo-transient method (FastICE v1.0) based on graphical processing units (GPUs), Geosci. Model Dev., 13, 955–976, https://doi.org/10.5194/gmd-13-955-2020, 2020.

Reuber, G. S., Holbach, L., and Räss, L.: Adjoint-based inversion for porosity in shallow reservoirs using pseudo-transient solvers

for non-linear hydro-mechanical processes, J. Comput. Phys., 423, 109797, https://doi.org/10.1016/j.jcp.2020.109797, 2020.

Rezaiee-Pajand, M., Kadkhodayan, M., Alamatian, J., and Zhang, L. C.: A new method of fictitious viscous damping determination for the dynamic relaxation method, Comput. Struct., 89, 783–794, https://doi.org/10.1016/J.COMPSTRUC.2011.02.002, 2011.

Richardson, L. F.: IX. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, Philos. T. R. Soc. A, 210, 307–357, https://doi.org/10.1098/rsta.1911.0009, 1911.

Riley, J. D.: Iteration Procedures for the Dirichlet Difference Problem, Mathematical Tables and Other Aids to Computation, 8, 125–131, https://doi.org/10.2307/2001924, 1954.

Saad, Y.: Iterative methods for linear systems of equations: A brief historical journey, in: Contemporary Mathematics, edited by: Brenner, S., Shparlinski, I., Shu, C.-W., and Szyld, D., American Mathematical Society, Providence, Rhode Island, 754, 197–215, https://doi.org/10.1090/conm/754/15141, 2020.

Shin, D. and Strikwerda, J. C.: Inf-sup conditions for finite-difference approximations of the Stokes equations, J. Aust. Math. Soc. B, 39, 121–134, 1997.

Swirydowicz, K., Langou, J., Ananthan, S., Yang, U., and Thomas, S.: Low synchronization Gram-Schmidt and generalized minimal residual algorithms, Numer. Linear Algebr., 28, e2343, https://doi.org/10.1002/nla.2343, 2020.

Tackley, P. J.: Effects of strongly variable viscosity on three-dimensional compressible convection in planetary mantles, J. Geophys. Res.-Sol. Ea., 101, 3311–3332, https://doi.org/10.1029/95JB03211, 1996.

Trottenberg, U., Oosterlee, C. W. C. W., and Schüller, A.: Multigrid, Academic Press, hardcover ISBN 9780127010700, eBook ISBN 9780080479569, 2001.

Virieux, J.: P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method, Geophysics, 51, 889–901, https://doi.org/10.1190/1.1442147, 1986.

Wang, L. H., Yarushina, V., 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, 2021.

Wulf, W. A. and McKee, S. A.: Hitting the memory wall: implications of the obvious, ACM Comp. Ar., 23, 20–24, https://doi.org/10.1145/216585.216588, 1995.

Young, D. M.: Second-degree iterative methods for the solution of large linear systems, J. Approx. Theory, 5, 137–148, https://doi.org/10.1016/0021-9045(72)90036-6, 1972.

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: Lecture Notes in Earth System Sciences, 0, 321–333, Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-16405-7_21, 2013.