complement (see May and Moresi (2008) for an overview). Preconditioners that are expressed through linear algebra operations on submatrices of the saddle point matrix, e.g. $G^T K G \approx \text{diag} \left( G^T \text{diag}(K) G^T \right)$, can be constructed by applying these operations through the petsc4py interface.

Drucker-Prager rheology. In Fraters et al. (2019), a number of solution strategies are explored for this case (amongst others), with the study advocating the use of two modifications to the Jacobian: (i) adding an additional term to Equation (32) that is the transpose of the second term, thus restoring the symmetry of $K$; (ii) to scale those terms associated with $\partial \eta/\partial \mathbf{u}$ by a spatially varying $\alpha_{\text{SPD}}$, calculated at the Gauss points according to:

$$
\alpha_{\text{SPD}} = \begin{cases} 
1 & \text{if } \left[ 1 - \frac{a \cdot b}{\|a\| \|b\|} \right]^2 < c_{\text{safety}} 2\eta(\dot{\mathbf{u}}), \\
\frac{2\eta(\dot{\mathbf{u}})}{c_{\text{safety}} \left[ 1 - \frac{a \cdot b}{\|a\| \|b\|} \right]^2} & \text{otherwise},
\end{cases}
$$

(43)

where $a = \dot{\mathbf{u}}$, and $b = \frac{\partial \eta}{\partial \mathbf{u}}$. This rescaling acts as a stabilisation, ensuring that $K$ remains positive definite. It should be noted that the pressure dependence of the Drucker-Prager rheology also leads to additional terms in the top-right block of the Stokes Jacobian matrix, in addition to $G$ in Equation (28), making the overall system asymmetric, regardless.

In traditional codes, the implementation of such additional terms in the Jacobian and the proposed modifications (stabilisation) require significant development. Analytical expressions for $\partial \eta/\partial \mathbf{u}$ and $\partial \eta/\partial \mathbf{p}$ must be derived for each specific rheological relationship analysed (as is done in the appendices of Fraters et al. (2019)), and the assembly of any additional terms may require a significant overhaul of existing code and data structures as, for example, sparsity structures may change. In Firedrake, the full Jacobian is derived symbolically and the code for its assembly generated automatically, making the entire process automatic, even for highly complex rheologies. We were able to implement the Jacobian modifications proposed in Fraters et al. (2019) in only 7 lines of Python code (the full Python script for this case is available in the repository accompanying this paper) and, as illustrated in Figure 12, we obtain similar results. As indicated in Fraters et al. (2019), the convergence of the problem gets more challenging with increased resolution, and although a reasonably converged result can be obtained for the case shown in Figure 12 at a resolution of $1024 \times 512$, this is insufficient to resolve the details of the unstructured mesh domain used in Spiegelman et al. (2016) who reported non-convergence for this case. Firedrake’s ability to choose from a large variety of discretisation types, including unstructured meshes, and its flexibility to adapt and experiment with the solution strategy, opens up numerous avenues to further investigate the challenges in this, and other, highly nonlinear problems.

It is important to point out that some common components of geodynamical models have not been showcased herein and, to our knowledge, have not yet been explored within the Firedrake framework. These include, for example, a free-surface boundary condition and the ability to model multiple-material flows, often implemented in geodynamical models using the particle-in-cell technique. Our goal for this paper is to provide solid foundations for future work in Firedrake that we, and others in the geodynamical modelling community, can build upon. Nonetheless, we see no fundamental reason why any component