A FINITE ELEMENT STOKES SOLVER FOR GLACIER FLOW

ED BUELER

version: May 24, 2024 for McCarthy 2024

This document is an appendix to my notes for the International Summer School in Glaciology in McCarthy, Alaska. It documents a Stokes numerical model, written in short Python codes, exploiting these open source tools and libraries:

- Firedrake, a finite element library `https://www.firedrakeproject.org/`
- Gmsh, a mesh generator `http://gmsh.info/`
- Paraview, a visualization tool `https://www.paraview.org/`

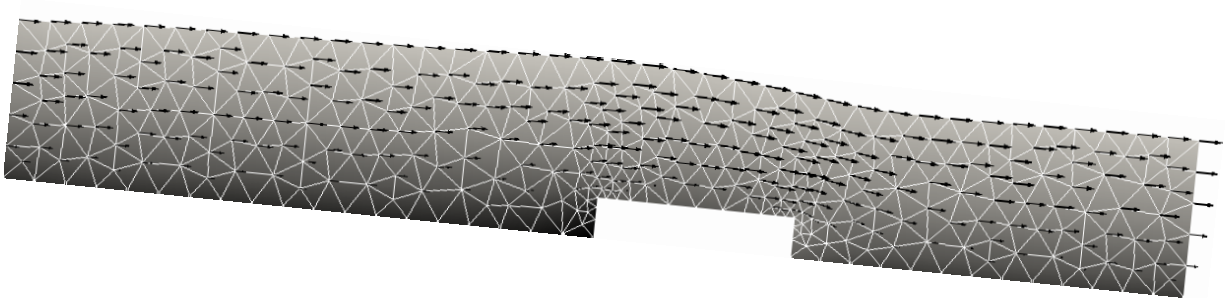

FIGURE 1. (FIXME: PUT NEW CODE RESULT) A 2D glacier flowing over a bedrock step. Arrows show velocity $\mathbf{u}$ and shading is by pressure p.

To start the presentation we state the Stokes model for 2D ice flow, with glacier-suitable boundary conditions, for example as in Figure 1. The solver works for any reasonable ice geometry, for example with a bedrock step (rise) as shown. However, because the slab-on-a-slope case has an exact solution, which is useful for understanding, verification purposes, and as a source of boundary conditions for other cases, we address that special case first. We then derive the weak form of the Stokes problem and give a brief overview of finite element (FE) methods. Our particular FE method uses an unstructured mesh of triangular or quadrilateral elements, optionally using an extruded mesh, to solve the Glen-Stokes problem by a stable "mixed" method with distinct approximating spaces for velocity and pressure. Finally, we demonstrate numerical solutions and give a brief guide to usage.

1. GLEN-STOKES MODEL

Let us start from the Glen-Stokes model stated in the notes. This model, also described in [8, 9], applies on any 2D or 3D domain Ω which has a piecewise smooth boundary.

Allowing any Glen exponent $n \geq 1$, the equations are:

$$-\nabla \cdot \tau + \nabla p = \rho \mathbf{g} \qquad \textit{stress balance} \tag{1}$$

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

$$D\mathbf{u} = A_n |\tau|^{n-1} \tau \qquad \textit{Glen flow law} \tag{3}$$

The notation here generally follows the notes, including velocity $\mathbf{u}$, pressure p, ice density ρ, acceleration of gravity $\mathbf{g}$, deviatoric stress tensor τ, strain rate tensor $D\mathbf{u}$, and ice softness A_n. A key idea is that tensors $D\mathbf{u}$ and τ are symmetric (conservation of angular momentum) and have trace zero (incompressibility). Note that $D\mathbf{u}$ is the symmetric part of the tensor velocity derivative $\nabla \mathbf{u}$:

$$(D\mathbf{u})_{ij} = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^\top\right) = \frac{1}{2}\left((u_i)_{x_j} + (u_j)_{x_i}\right) \tag{4}$$

The full (Cauchy) stress tensor σ is the deviatoric stress tensor τ minus the pressure,

$$\sigma = \tau - p\, I, \tag{5}$$

so equation (1) simply says $-\nabla \cdot \sigma = \rho \mathbf{g}$. One may derive from (5) that $p = -\frac{1}{2}\operatorname{tr}(\sigma)$, in 2D anyway, thus the pressure is the negative of the average normal stress. By definition $\nabla \cdot \tau$ in (1) is a vector with components which are the divergences of the columns:

$$(\nabla \cdot \tau)_i = (\tau_{1i})_{x_1} + (\tau_{2i})_{x_2} \tag{6}$$

Note $\nabla \cdot \tau$, ∇p, and $\mathbf{g}$ are regarded as column vectors.

The viscosity form of (3) can also be found in the notes:

$$\tau = 2\nu D\mathbf{u} = B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u} \tag{7}$$

Here $B_n = (A_n)^{-1/n}$ is the ice hardness in units $\mathrm{Pa\,s}^{1/n}$. The tensor norm notation used in (3) and (7) is defined as follows, with the summation convention:

$$|\tau|^2 = \frac{1}{2}\operatorname{tr}\left(\tau^2\right) = \frac{1}{2}\tau_{ij}\tau_{ij}, \qquad |D\mathbf{u}|^2 = \frac{1}{2}\operatorname{tr}\left((D\mathbf{u})^2\right) = \frac{1}{2}(D\mathbf{u})_{ij}(D\mathbf{u})_{ij}$$

Using (7) we can eliminate τ from equation (1), thereby rewriting the system in terms of velocity $\mathbf{u}$ and pressure p only:

$$-\nabla \cdot \left(B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u}\right) + \nabla p = \rho \mathbf{g} \tag{8}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{9}$$

Also, from now on we denote the points of the 2D region Ω by (x, z), and we add an angle parameter: gravity is at an angle α with the z-direction, so that $\mathbf{g} = \langle g \sin\alpha, -g\cos\alpha \rangle$ where $g > 0$. The numerical model in these notes solves system (8), (9) with a body force given by this $\rho\mathbf{g}$. A solution of the problem is a pair $(\mathbf{u}, p)$.

Certain glacier-suitable velocity and stress boundary conditions will be used here. We assume that base, top, inflow, and outflow boundaries can all be identified. On the base we require no slip:

$$\mathbf{u} = 0 \qquad \textit{base} \tag{10}$$

(although this can be modified if the user desires). On the top we set a condition of zero applied stress, $\sigma\hat{\mathbf{n}} = 0$:

$$\left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}D\mathbf{u} - pI\right)\hat{\mathbf{n}} = 0 \qquad \textit{top} \tag{11}$$

The inflow boundary is the left side, which has outward normal $\hat{\mathbf{n}} = \langle -1, 0\rangle$, and here we set a nonzero inflow velocity:

$$\mathbf{u} = \langle f(z), 0\rangle \qquad \textit{inflow} \tag{12}$$

The implemented function $f(z)$ satisfies the slab-on-slope equations for a given thickness H_{in}; see below. On an outflow boundary, which is the right side and has $\hat{\mathbf{n}} = \langle 1, 0\rangle$ in the implemented form, we set a nonzero hydrostatic normal stress using the ice thickness H_{out} at that location:

$$\begin{aligned}\left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}D\mathbf{u} - pI\right)\hat{\mathbf{n}} = C_{\text{out}}\big\langle &-\rho g\cos\alpha(H_{\text{out}} - z),\\ &\rho g\sin\alpha(H_{\text{out}} - z)\big\rangle \qquad \textit{outflow}\end{aligned} \tag{13}$$

The implemented constant C_{out} is set so that the applied stress on the outflow equals what the applied stress on the inflow would be, thus $C_{\text{out}} = (H_{\text{in}}/H_{\text{out}})^2$.

2. Slab-on-slope solutions

Testing a numerical model requires verification tools, namely exact solutions. Such solutions are in short supply for the Glen-Stokes model, so now we recapitulate the slab-on-slope construction given in the notes, allowing any Glen exponent n. This also generates inflow and outflow boundary conditions which can be used for more general flows.

Using component notation for 2D, equations (8), (9) become the following Glen-Stokes model in coordinates (x, z) and velocity components $\mathbf{u} = \langle u, w\rangle$:

$$-\left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}u_x\right)_x - \left(\frac{1}{2}B_n|D\mathbf{u}|^{\frac{1}{n}-1}(u_z + w_x)\right)_z + p_x = \rho g\sin\alpha \tag{14}$$

$$-\left(\frac{1}{2}B_n|D\mathbf{u}|^{\frac{1}{n}-1}(u_z + w_x)\right)_x - \left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}w_z\right)_z + p_z = -\rho g\cos\alpha \tag{15}$$

$$u_x + w_z = 0 \tag{16}$$

The strain-rate norm expands to

$$|D\mathbf{u}|^2 = \frac{1}{2}\left(u_x^2 + \frac{1}{2}(u_z + w_x)^2 + w_z^2\right) \tag{17}$$

Consider either an infinitely-long or a periodic slab flow. It has fixed (x-independent) values of the bed elevation b and the surface elevation h, so the domain is $\Omega = \{(x, z)\,|\,b < z < h\}$. Assume that the boundary stresses are also x-independent. Then there is no variation in x, i.e. $\partial/\partial x = 0$, so the system simplifies:

$$\begin{aligned}-\left(\frac{1}{2}B_n|D\mathbf{u}|^{\frac{1}{n}-1}u_z\right)_z &= \rho g\sin\alpha\\ p_z &= -\rho g\cos\alpha\\ w_z &= 0\end{aligned} \tag{18}$$

Also, the strain-rate norm simplifies to $|D\mathbf{u}| = \frac{1}{2}|u_z|$. If we further assume that there is no melt or freeze at the base then $w = 0$ identically; the ice velocity is bed and surface parallel. Then an integration with respect to z, and assumption of zero pressure at the surface, yields hydrostatic pressure:

$$p(z) = \rho g \cos\alpha (h - z) \tag{19}$$

Now (18) yields a single nontrivial equation for the horizontal velocity:

$$-\left(\frac{B_n}{2^{1/n}}|u_z|^{\frac{1}{n}-1}u_z\right)_z = \rho g \sin\alpha$$

As we expect $u_z > 0$, rearrangement gives

$$\left((u_z)^{1/n}\right)_z = -\frac{2^{1/n}\rho g \sin\alpha}{B_n}$$

This can be vertically-integrated downward from the surface $z = h$, using the no-stress condition, which simplifies to $u_z = 0$, to give

$$u_z = 2\left(\frac{\rho g \sin\alpha}{B_n}\right)^n (h - z)^n \tag{20}$$

Integrating vertically again, upward from the base $z = b$ where $u = 0$, gives

$$u(z) = \frac{2}{n+1}\left(\frac{\rho g \sin\alpha}{B_n}\right)^n \left((h-b)^{n+1} - (h-z)^{n+1}\right) \tag{21}$$

Formulas (19) and (21) exactly solve the Stokes equations and the described boundary conditions. They will be used for verifying the numerical solver.

Additionally these slab-on-slope formulas allow us to set boundary conditions which lead to glaciologically-reasonable solutions for more general shapes like that shown in Figure 1. In the implemented cases, the inflow side has (21) applied as a Dirichlet condition, as in (12). On the outflow side we apply a normal stress derived from (20). That is, using the facts that $w = 0$, $u_x = 0$, $|D\mathbf{u}| = \frac{1}{2}u_z$, and $\hat{\mathbf{n}} = \langle 1, 0\rangle$, we have

$$\begin{aligned}\sigma\hat{\mathbf{n}} &= \left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}D\mathbf{u} - pI\right)\hat{\mathbf{n}} = \left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}\begin{pmatrix} u_x & \frac{1}{2}(u_z+w_x) \\ \frac{1}{2}(u_z+w_x) & w_z \end{pmatrix} - pI\right)\hat{\mathbf{n}} \\ &= B_n\left(\frac{1}{2}u_z\right)^{\frac{1}{n}-1}\begin{pmatrix} 0 \\ \frac{1}{2}u_z \end{pmatrix} - \begin{pmatrix} p \\ 0 \end{pmatrix} = \begin{pmatrix} -p \\ \frac{B_n}{2^{1/n}}(u_z)^{1/n} \end{pmatrix} = \begin{pmatrix} -\rho g\cos\alpha(h-z) \\ \rho g \sin\alpha (h-z) \end{pmatrix}\end{aligned}$$

This justifies formula (13).

3. Weak form

Equations (8) and (9) are called the *strong form* of the model. An integral equation form of the same equations, the *weak form*, is needed when building a numerical finite element (FE) method [5]. This new form is derived by multiplying the strong form equations by test functions and then integrating over Ω. This defines a nonlinear functional F, that is, a function which takes functions as input and returns scalars. The weak form says that F is zero when acting on test functions. The significance of the weak form can be stated in several ways:

- It is well-posed in the sense of always having a unique solution for glaciologically-relevant boundary values [9].

- It has a larger space of potential solutions than the strong form, for instance it is more flexible with respect to discontinuities in the data.
- An FE method can create test functions, via local constructions which work on a mesh of triangles or quadrilaterals, and these go into the weak form. This framework is more flexible with respect to problem geometry than are, for example, finite difference methods based on the strong form.

A solution to the weak form stated below is a pair $(\mathbf{u}, p)$ wherein each function lives in a certain function space. We write $\mathbf{u} \in V_D$ and $p \in Q$ here, but the Sobolev spaces V_D and Q are precisely-identified in [9]. Test functions $\mathbf{v} \in V_0$ and $q \in Q$ come from nearly the same spaces, with V_D and V_0 differing only by the value on the Dirichlet boundary: $\mathbf{u} \in V_D$ satisfies a nonhomogeneous inflow condition (12) while $\mathbf{v} \in V_0$ is zero there.

To give an initial definition of F we multiply (8) by $\mathbf{v} \in V_0$ and (9) by $q \in Q$, then add and integrate:

$$F(\mathbf{u}, p; \mathbf{v}, q) = \int_\Omega -\left(\nabla \cdot \left(B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u}\right)\right) \cdot \mathbf{v} + \nabla p \cdot \mathbf{v} - \rho \mathbf{g} \cdot \mathbf{v} - (\nabla \cdot \mathbf{u})\, q \qquad (22)$$

However, the desired definition of F balances the number of derivatives on $(\mathbf{v}, q)$ versus $(\mathbf{u}, p)$, that is, it has only first derivatives on $\mathbf{u}, \mathbf{v}$, and no derivative on p, q. To make this change we need integration-by-parts.

Recall the product rule $\nabla \cdot (f\mathbf{X}) = \nabla f \cdot \mathbf{X} + f \nabla \cdot \mathbf{X}$ and the divergence theorem $\int_\Omega \nabla \cdot (f\mathbf{X}) = \int_{\partial\Omega} f\mathbf{X} \cdot \hat{\mathbf{n}}$. Denoting $\tau = B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u}$ for convenience we have

$$\begin{aligned}\int_\Omega (\nabla \cdot \tau) \cdot \mathbf{v} &= \sum_{j=1}^{2} \int_\Omega \nabla \cdot (\tau_{j\circ})\, v_j = \sum_{j=1}^{2} \int_\Omega \nabla \cdot (\tau_{j\circ} v_j) - \tau_{j\circ} \nabla v_j \\ &= \sum_{j=1}^{2} \int_{\partial\Omega} (\tau_{j\circ} v_j) \cdot \hat{\mathbf{n}} - \int_\Omega \tau_{j\circ} \cdot \nabla v_j = \int_{\partial\Omega} (\tau \hat{\mathbf{n}}) \cdot \mathbf{v} - \int_\Omega \operatorname{tr}(\tau \nabla \mathbf{v})\end{aligned}$$

where $\circ$ denotes a vector entry index and $\tau_{j\circ}$ denotes the jth row of τ. Here $\nabla \mathbf{v}$ defines a 2×2 matrix,

$$\nabla \mathbf{v} = \left[\begin{array}{c|c} \nabla v_1 & \nabla v_2 \end{array}\right] = \begin{bmatrix} (v_1)_{x_1} & (v_2)_{x_1} \\ (v_1)_{x_2} & (v_2)_{x_2} \end{bmatrix}$$

and so

$$\operatorname{tr}(\tau \nabla \mathbf{v}) = \sum_{j=1}^{3} \tau_{j\circ} \cdot \nabla v_j = \sum_{i,j=1}^{3} \tau_{ji} (v_j)_{x_i}$$

(Some sources write $A : B$ for $\operatorname{tr}(AB)$ [9].) Note $\operatorname{tr}(\tau \nabla \mathbf{v}) = \operatorname{tr}(\tau D\mathbf{v})$ because $\operatorname{tr}(AB) = 0$ if A is symmetric and B is antisymmetric. (To show this take $A = \tau$ and $B = \nabla \mathbf{v} - D\mathbf{v}$.) Finally we do a straightforward integration-by-parts on the pressure part of F:

$$\int_\Omega \nabla p \cdot \mathbf{v} = \int_\Omega \nabla \cdot (p\, \mathbf{v}) - p(\nabla \cdot \mathbf{v}) = \int_{\partial\Omega} p \hat{\mathbf{n}} \cdot \mathbf{v} - \int_\Omega p(\nabla \cdot \mathbf{v})$$

The above facts allow us to rewrite (22) with a normal stress boundary integral:

$$F(\mathbf{u}, p; \mathbf{v}, q) = \int_\Omega \left[\operatorname{tr}(\tau D\mathbf{v}) - p(\nabla \cdot \mathbf{v}) - (\nabla \cdot \mathbf{u})\, q - \rho \mathbf{g} \cdot \mathbf{v}\right] - \int_{\partial\Omega} (\sigma \hat{\mathbf{n}}) \cdot \mathbf{v} \qquad (23)$$

(We have denoted $\sigma = \tau - pI$ for brevity.) Note $\mathbf{u}, \mathbf{v}$ appear with first derivatives and p, q have no derivatives.

Next recall that $\mathbf{v} \in V_0$ satisfies $\mathbf{v} = 0$ along the base and inflow surfaces. Thus these parts of the integral over $\partial\Omega$ in (23) are zero. Conditions (11), (13) now completely eliminate the unknown solution $\mathbf{u}, p$ from the boundary integral. The above computations yield our final formula for the nonlinear functional:

$$F(\mathbf{u}, p; \mathbf{v}, q) = \int_\Omega B_n |D\mathbf{u}|^{\frac{1}{n}-1} \operatorname{tr}(D\mathbf{u} D\mathbf{v}) - p(\nabla \cdot \mathbf{v}) - (\nabla \cdot \mathbf{u})\, q \qquad (24)$$
$$- \int_\Omega \rho \mathbf{g} \cdot \mathbf{v} - \int_{\{\text{outflow}\}} C_{\text{out}} \rho g \cos\alpha (h - z) v$$

The weak form of the Glen-Stokes model is the statement that, at the solution $\mathbf{u} \in V_D$ and $p \in Q$, functional F in (24) is zero in all test function directions:

$$F(\mathbf{u}, p; \mathbf{v}, q) = 0 \qquad \text{for all } \mathbf{v} \in V_0 \text{ and } q \in Q \qquad (25)$$

This weak formulation is proven in [9, Theorem 3.8] to be well-posed, so that a unique solution exists. The well-posedness theorem makes reasonable assumptions about the domain Ω and the boundary data, and these are satisfied in the cases we consider. Noting that the last two integrals in (24) can be regarded as source terms for stress, if the inflow velocity is zero and if we replace the source terms by zero—we remove gravity and outflow stresses—then the unique solution to (25) is $\mathbf{u} = 0$ and $p = 0$.

Now our goal is to accurately approximate the solution $(\mathbf{u}, p)$ of (25) using an FE method.

4. Finite element method

This section gives a very abbreviated summary of our finite element (FE) method to solve the glaciological Glen-Stokes equations, but better coverage of FE methods, including for the linear Stokes equations, is in references [3, 4, 5]. The method here uses Firedrake [11]; we do *not* implement most of the following techniques ourselves.

The fundamental FE idea is to replace the infinite-dimensional function spaces appearing in the weak form (25) with finite-dimensional subspaces constructed locally from triangles (or quadrilaterals) in a mesh. The approximate solution $(\mathbf{u}^h, p^h)$ lives in these subspaces, and the functional F in (24) is computed on test functions from these subspaces. Requiring F to be zero over a basis of the test-functions, as stated in equation (26) below, defines the (nonlinear) discrete equations. We solve this nonlinear system of algebraic equations using Newton's method [4].

The mesh itself is denoted $\mathcal{T}_h$. It covers the domain Ω by a finite set of K non-overlapping open elements[1] $\triangle_k$, is stored in a `.msh` file generated by Gmsh; see section 5 below. The subscript "h" on $\mathcal{T}_h$ also denotes the maximum diameter of the triangles. Let N_1 be the number of element nodes (vertices) in the mesh, including nodes on $\partial\Omega$, and denote the coordinates of the ith node by (x_i, z_i) for $i = 0, 1, \dots, N_1 - 1$.

For a triangle $\triangle_k$ there are various choices of a *finite element space*, a space of low-degree polynomials defined on that triangle. By default we use two particular FE spaces,

[1] Here elements are either triangles or quadrilaterals, but we stick with notation $\mathcal{T}_h$ and $\triangle_k$.

denoted P_1 for the pressure and $(P_2)^2$ for velocity. Using such paired spaces is called a *mixed method*, and P_2-P_1 chosen here are the lowest-order *Taylor-Hood* elements [5].

Consider just one triangle. The name P_1 refers to the space of linear functions $a+bx+cz$ on the triangle. The three vertices of the triangle are the preferred degrees of freedom (Figure 2 left) in the sense that any P_1 function on the triangle is determined by its values at these vertices. The P_2 space, used for the scalar components of velocity, is the space of quadratic functions $a + bx + cy + dx^2 + exy + fy^2$, with six degrees of freedom at the vertices and edge midpoints (Figure 2 right). The function spaces for quadrilaterals are comparable [5], but we omit the details.

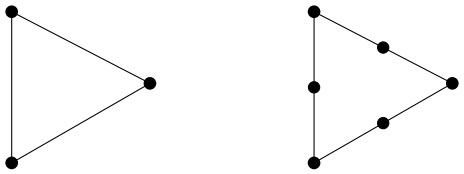

FIGURE 2. P_1 and P_2 triangular elements.

A continuous scalar function on Ω which is piecewise-linear on each triangle, i.e. in P_1 on each triangle, is determined by its values at the N_1 nodes of the mesh. The space of such functions is a finite-dimensional subspace $Q^h \subset Q$ of the pressure space Q used in the weak form (25). Next, a continuous scalar function on Ω which is piecewise-quadratic (P_2) on each triangle is determined by its values at the nodes *and* edge midpoints in the mesh [5]. Let N_2 denote the number of all such P_2 degrees of freedom, not including Dirichlet boundary nodes. A small example of this kind of mesh is shown in Figure 3.

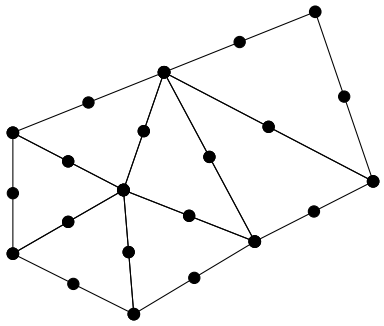

FIGURE 3. A mesh with $K = 7$ triangular elements, $N_1 = 8$ P_1 degrees of freedom (nodes), and $N_2 = 22$ degrees of freedom for P_2.

However, velocity fields are vector-valued. Let us define V_0^h to be the space of *pairs* of scalar piecewise-quadratic functions on Ω, additionally requiring them to be zero on the base and inflow (i.e. Dirichlet) boundaries, where (10) and (12) apply. The functions in V_0^h are 2D vector fields with components which are piecewise-quadratic scalar functions in P_2. We define V_D^h to be the nearly the same space except satisfying (12) on the inflow boundary. There are two P_2 subspaces used in the FE weak form (26) below: $V_0^h \subset V_0$, $V_D^h \subset V_D$.

Finally we think about the test functions, specifically the basis functions called *hat functions*,[2] used in the FE weak form (26). For pressure, let $\psi_j(x,z)$ be a P_1 hat function from Q^h which is linear on each triangle, continuous on Ω, equal to one at node (x_j,z_j), and otherwise zero at all other nodes. Thus $\psi_j(x_i,z_i) = \delta_{ij}$. The set $\{\psi_j\}$ is a set of N_1 linearly-independent functions in Q^h, and thus a basis of Q^h. Similarly, for every P_2 node which is not in the base or inflow boundary, whether a triangle vertex or edge midpoint, there is a P_2 hat function. Pairs of such piecewise-quadratic hat functions, $(\phi_j(x,z),0)$ or $(0,\phi_j(x,z))$, form a basis of $2N_2$ functions for the velocity test space V_0^h.

When using P_2 elements for velocity and P_1 elements for pressure we see that the dimension of the velocity space is much higher than the pressure space. For example, for the $N_1 = 356$ vertex mesh shown in Figure 1, the velocity dimension $2N_2 = 2646$ is seven times greater than N_1. Such a dimension imbalance turns out to be desirable! As explained in the FE literature under the obscure name "inf-sup condition" [3, 4, 5], Stokes equations mixed methods are only stable if the velocity space is sufficiently-large relative to the pressure space.

Finally we can state the FE method itself. It is a finite-dimensional approximation of weak form (25), namely it finds $\mathbf{u}^h \in V_D^h$ and $p^h \in Q^h$ so that

$$F(\mathbf{u}^h, p^h; \mathbf{v}^h, q^h) = 0 \qquad \text{for all } \mathbf{v}^h \in V_0^h \text{ and } q^h \in Q^h \tag{26}$$

Note that relative to (25) the nonlinear functional F is unchanged and we have merely added "h" superscripts to all continuum quantities! Problem (26) can be shown to be well-posed by the same theory that applies to the continuum problem [9, Theorem 4.3].

By linearity of F in the $\mathbf{v}, q$ positions, it suffices to consider only a basis of test functions, i.e. hat functions, from V_0^h and Q^h. However, (26) is a nonlinear system which should be solved by Newton's method [4, 10]. In fact, suppose we have some current iterate $\mathbf{u}^h, p^h$ which does not exactly solve (26). Then for each test function $\mathbf{v}^h$ or q^h, i.e. for each hat function, the value of the "residual" $F(\mathbf{u}^h, p^h; \mathbf{v}^h, q^h)$ can be computed concretely, though also approximately, by quadrature [4, 5]. Also we can use the current iterate, and the derivatives of F, to evaluate the Jacobian at the current iterate. This "assembly" process generates a system of sparse, linear algebraic equations, namely the Newton step equations of the discrete Glen-Stokes problem. In actuality the equations are assembled element-by-element, not node-by-node, in the sense that for each triangle $\triangle_k$ the contribution from that triangle is added to the correct equation, but this represents an unneeded detail. The point of using an FE library like Firedrake is that it hides the assembly process and it allows us to think about the problem at a higher level [11].

At each Newton step we solve a sparse and indefinite linear system. These systems have a well-known block structure (Figure 4), so we may apply a Schur complement preconditioner in a GMRES Krylov iteration [5, 7]. This buzzword salad suggests the complexity of solver technology, and we abandon further explanation, but see Chapter 14 of [4].

[2] Figure 1.6 in [5] and Figure 10.4 in [4] show P_1 hat functions, while Figure 1.7 in [5] suggests how P_2 hat functions would look.

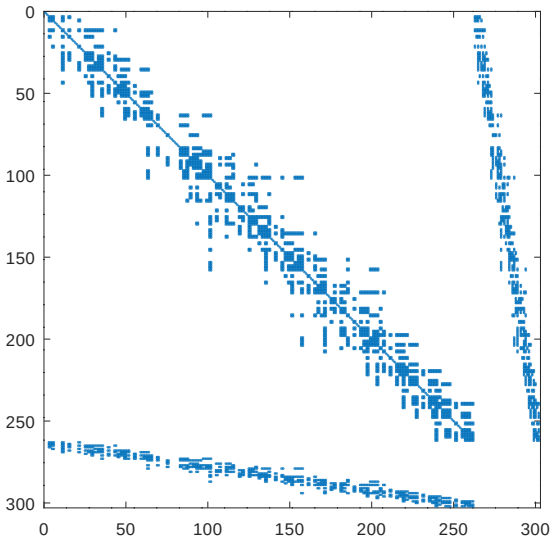

FIGURE 4. Sparsity pattern of the Newton step linear system, when solving the Stokes equations on a coarse mesh ($K = 52, N_1 = 40, N_2 = 131$).

5. Implementation in Python

This section gives a practical view of our codes, including how they are divided into modules and the sequence in which they are used in practice. In overview, we implement problem (26), and make the stated discretization choices, in Python using Firedrake [11]. The PETSc solver library [2, 4] is called via Firedrake on the discretized nonlinear equations.

To define the nonlinear residual function F in (24), the UFL domain-specific language, designed for describing such weak forms [1], is applied. This language comes rather close to the mathematical form. Compare (4), (7), and (24) to the code as written:

```
def D(U):
    return 0.5 * (grad(U) + grad(U).T)
Du2 = 0.5 * inner(D(u), D(u)) + (eps * Dtyp)**2.0
rr = 1.0/3.0 - 1.0
F = ( inner(B3 * Du2**(rr/2.0) * D(u), D(v)) - p * div(v) - div(u) * q ) * dx \
    - inner(f_body, v) ) * dx - inner(outflow_sigma, v) * ds(bdryids['outflow'])
```

Firedrake knows how to take this code, plus a choice of mesh and function spaces, and run the assemble process to create the Newton step equations for this problem. Then Firedrake's `solve()` command solves (26) [11].

As shown in Figure 5 there are three major Python components:

- `domain.py`: This writes an outline of the domain Ω into an ASCII file (`.geo` extension) using the Gmsh [6] geometry description language. Note Gmsh can be used to examine and mesh this domain interactively, or at the command-line as above. Portions of the boundary are marked with integer tags; see the Python dictionary `bdryids`.
- `momentummodel.py`: Implements `class MomentumModel`. Most actual ice physics is isolated here, including physical constants and the Stokes problem weak form

(24). The `solve()` method assembles and solves the nonlinear Glen-Stokes problem (25) using Firedrake's `solve()` command. This solver has option prefix `s_`.

- `flow.py`: This driver reads user options, loads the mesh from a Gmsh `.msh` file, uses a `MomentumModel` object to solve the Stokes problem, and writes the solution into a Paraview-readable `.pvd` file.

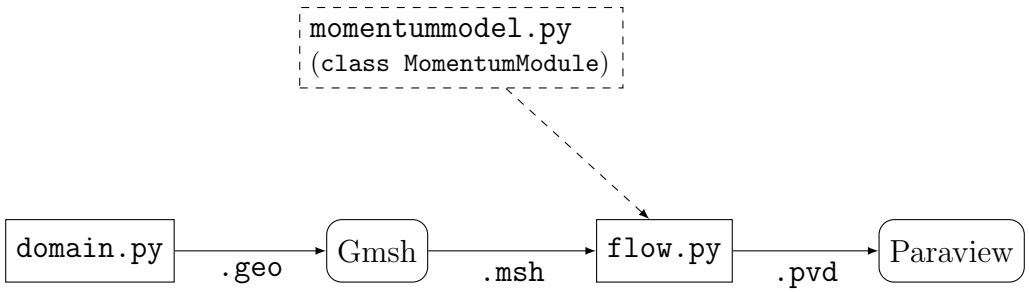

FIGURE 5. Users interact with the solid-outline tools in the given order.

Here is a simple example solving the default Stokes problem:

```
$ ./domain.py -o glacier.geo                    # create domain outline
$ gmsh -2 glacier.geo                           # mesh domain
$ source ~/firedrake/bin/activate               # start Firedrake
(firedrake) $ ./flow.py -mesh glacier.msh       # solve Stokes problem
(firedrake) $ paraview glacier.pvd              # visualize results
```

The `README.md` file documents usage and regression testing more thoroughly.

REFERENCES

[1] M. S. ALNÆS, A. LOGG, K. B. OLGAARD, M. E. ROGNES, AND G. N. WELLS, *Unified Form Language: A domain-specific language for weak formulations of partial differential equations*, ACM Trans. Math. Softw., 40 (2014), pp. 9:1–9:37.

[2] S. BALAY, S. ABHYANKAR, M. F. ADAMS, J. BROWN, P. BRUNE, K. BUSCHELMAN, L. DALCIN, V. EIJKHOUT, W. D. GROPP, D. KAUSHIK, M. G. KNEPLEY, L. C. MCINNES, K. RUPP, B. F. SMITH, S. ZAMPINI, H. ZHANG, AND H. ZHANG, *PETSc Users Manual*, Tech. Rep. ANL-95/11 - Revision 3.8, Argonne National Laboratory, 2017.

[3] D. BRAESS, *Finite Elements: Theory, Fast Solvers, and Applications in Elasticity Theory*, Cambridge University Press, 3rd ed., 2007.

[4] E. BUELER, *PETSc for Partial Differential Equations: Numerical Solutions in C and Python*, SIAM Press, Philadelphia, 2021.

[5] H. C. ELMAN, D. J. SILVESTER, AND A. J. WATHEN, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2nd ed., 2014.

[6] C. GEUZAINE AND J.-F. REMACLE, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, Int. J. Numer. Meth. Eng., 79 (2009), pp. 1309–1331.

[7] G. GOLUB AND C. V. LOAN, *Matrix Computations*, Johns Hopkins University Press, 4th ed., 2013.

[8] R. GREVE AND H. BLATTER, *Dynamics of Ice Sheets and Glaciers*, Advances in Geophysical and Environmental Mechanics and Mathematics, Springer, 2009.

[9] G. JOUVET AND J. RAPPAZ, *Analysis and finite element approximation of a nonlinear stationary Stokes problem arising in glaciology*, Advances in Numerical Analysis, 2011 (2011), p. 24 pages.

[10] C. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM Press, Philadelphia, 2003.
[11] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. McRae, G.-T. Bercea, G. R. Markall, and P. H. J. Kelly, *Firedrake: automating the finite element method by composing abstractions*, ACM Trans. Math. Softw., 43 (2016), pp. 24:1–24:27.