APPENDIX A: A FINITE ELEMENT STOKES SOLVER FOR GLACIER FLOW

ED BUELER

version: June 6, 2022 for McCarthy 2022

This is an appendix to my notes *Numerical modelling of glaciers, ice sheets, and ice shelves*—here called "the notes"—for the International Summer School in Glaciology in McCarthy, Alaska.

First we state the Stokes model for ice flow, with glacier-suitable boundary conditions, for example as in Figure A1, including the slab-on-a-slope case. (The slab exact solution is useful both for verification purposes and as a source of boundary conditions for general cases.) We then derive the *weak form* of the Stokes problem. A brief overview of *finite element* (FE) methods, based on such weak forms, follows. Our particular FE method uses an unstructured mesh of triangular elements, on any planar region, to solve the Glen-Stokes problem by a stable *mixed element* method with distinct approximating spaces for velocity and pressure. We demonstrate a numerical solution for the instantaneous velocity and pressure. Then we describe a moving-mesh scheme to solve the surface kinematical equation, and demonstrate an evolving glacier shape.

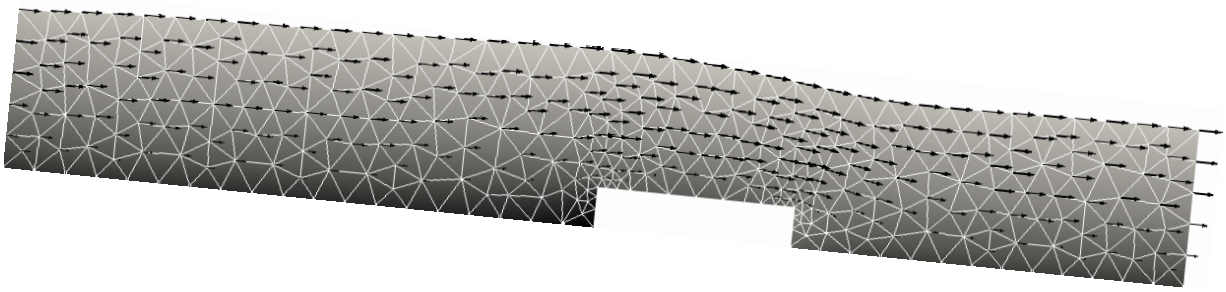

FIGURE A1. A 2D glacier flowing over a bedrock step. Arrows show velocity $\mathbf{u}$ and shading is by pressure p.

Our numerical model is in short Python codes, documented at the end, which exploit four advanced open source tools and libraries:

- Firedrake, an FE library `https://www.firedrakeproject.org/`
  - inside Firedrake is PETSc, a solver library `https://petsc.org/`
- Gmsh, a mesh generator `http://gmsh.info/`
- Paraview, a visualization tool `https://www.paraview.org/`

A.1. GLEN-STOKES MODEL

Recall the Glen-Stokes model in equations (3)–(5) from the notes. This model, also described in [9, 10], applies on a 2D or 3D domain Ω which must have a piecewise smooth

boundary (so that we may apply boundary conditions). Allowing any Glen exponent $n \geq 1$, the equations are:

$$-\nabla \cdot \tau + \nabla p = \rho \mathbf{g} \qquad \textit{stress balance} \tag{A1}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \textit{incompressibility} \tag{A2}$$

$$D\mathbf{u} = A_n |\tau|^{n-1} \tau \qquad \textit{Glen flow law} \tag{A3}$$

The notation here generally follows Table 1 in the notes, including velocity $\mathbf{u}$, pressure p, ice density ρ, acceleration of gravity $\mathbf{g}$, deviatoric stress tensor τ and strain rate tensor $D\mathbf{u}$. Tensors $D\mathbf{u}$ and τ are symmetric and have trace zero. Recall that $D\mathbf{u}$ is the symmetric part of the tensor velocity derivative $\nabla \mathbf{u}$:

$$(D\mathbf{u})_{ij} = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^\top\right) = \frac{1}{2}\left((u_i)_{x_j} + (u_j)_{x_i}\right) \tag{A4}$$

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

$$\sigma = \tau - p\, I, \tag{A5}$$

so equation (A1) simply says $-\nabla \cdot \sigma = \rho \mathbf{g}$. One may derive from (A5) that $p = -\frac{1}{3}\operatorname{tr}(\sigma)$ (in 3D), thus the pressure is the negative of the average normal stress. By definition $\nabla \cdot \tau$ in (A1) is a vector with components which are the divergences of the rows:

$$(\nabla \cdot \tau)_i = (\tau_{i1})_{x_1} + (\tau_{i2})_{x_2} + (\tau_{i3})_{x_3} \tag{A6}$$

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

The viscosity form of (A3) can also be found in the notes:

$$\tau = 2\nu D\mathbf{u} = B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u} \tag{A7}$$

Here $B_n = (A_n)^{-1/n}$ is the n-dependent ice hardness in units $\mathrm{Pa\,s}^{1/n}$. The tensor norm notation used in (A3) and (A7) 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 (A7) we can eliminate τ from equation (A1), thereby rewriting the system in terms of velocity $\mathbf{u}$ and pressure p only:

$$-\nabla \cdot \left(B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u}\right) + \nabla p = \rho \mathbf{g} \tag{A8}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{A9}$$

This system is the preferred Glen-Stokes model. A solution is a pair $(\mathbf{u}, p)$.

From now on we suppose the domain Ω is 2D, so points are denoted (x, z). (Relative to 3D: $y = 0$, $\partial/\partial y = 0$, $v = 0$, and $\mathbf{u} = \langle u, w \rangle$.) Also we will assume the force of gravity is at an angle α with the z-direction so $\mathbf{g} = \langle g \sin\alpha, -g\cos\alpha \rangle$ where $g = |\mathbf{g}|$.

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

$$\mathbf{u} = 0 \qquad \textit{base} \tag{A10}$$

On the top we set a condition of zero applied stress, i.e. $\sigma \hat{\mathbf{n}} = 0$:

$$\left(B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u} - pI\right) \hat{\mathbf{n}} = 0 \qquad \textit{top} \tag{A11}$$

The left-side inflow boundary has outward normal $\hat{\mathbf{n}} = \langle -1, 0\rangle$ in our case, and on this surface we set a nonzero inflow velocity:

$$\mathbf{u} = \langle f(z), 0\rangle \qquad \textit{inflow} \tag{A12}$$

(The function $f(z)$ will satisfy the slab-on-slope equations for a specific thickness H_{in} at the inflow; see below.) On an outflow boundary, where $\hat{\mathbf{n}} = \langle 1, 0\rangle$, we set a nonzero hydrostatic normal stress using the (varying) ice thickness H_{out} at the outflow:

$$\left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}D\mathbf{u} - pI\right)\hat{\mathbf{n}} = C_{\text{out}}\big\langle -\rho g\cos\alpha(H_{\text{out}} - z),\\ \rho g\sin\alpha(H_{\text{out}} - z)\big\rangle \qquad \textit{outflow} \tag{A13}$$

The constant C_{out} is adjusted so that the total applied stress is equal to its value for the input ice thickness H_{in}, thus $C_{\text{out}} = (H_{\text{in}}/H_{\text{out}})^2$.

A.2. Slab-on-slope solutions

Testing a numerical model requires verification tools, namely exact solutions, so here we recapitulate the slab-on-slope construction given in the notes. This time we allow any Glen exponent n. We will construct inflow and outflow boundary conditions, for use when solving for more general flows, from this case.

Using component notation for 2D, equations (A8), (A9) become the following Glen-Stokes model in coordinates (x, z) and velocity components $\mathbf{u} = \langle u, w\rangle$:

$$-\left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}u_x\right)_x - \left(\frac{1}{2}B_n|D\mathbf{u}|^{\frac{1}{n}-1}(u_z + w_x)\right)_z + p_x = \rho g\sin\alpha \tag{A14}$$

$$-\left(\frac{1}{2}B_n|D\mathbf{u}|^{\frac{1}{n}-1}(u_z + w_x)\right)_x - \left(B_n|D\mathbf{u}|^{\frac{1}{n}-1}w_z\right)_z + p_z = -\rho g\cos\alpha \tag{A15}$$

$$u_x + w_z = 0 \tag{A16}$$

The strain-rate norm expands to

$$|D\mathbf{u}|^2 = \frac{1}{2}\left(u_x^2 + \frac{1}{2}(u_z + w_x)^2 + w_z^2\right) \tag{A17}$$

Consider a slab with fixed (x-independent) values of the bed elevation b and the surface elevation h, so the domain is $\Omega = \{(x, z)\,|\,b < z < h\}$. (This describes either an infinitely-long or a periodic slab flow.) 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:

$$-\left(\frac{1}{2}B_n|D\mathbf{u}|^{\frac{1}{n}-1}u_z\right)_z = \rho g\sin\alpha \tag{A18}$$

$$p_z = -\rho g\cos\alpha$$

$$w_z = 0$$

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 surface parallel. Then an integration with respect to z, and assumption of zero pressure at the surface, yields hydrostatic pressure:

$$p(z) = \rho g\cos\alpha(h - z) \tag{A19}$$

Now (A18) 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 $u_z > 0$, with rearrangement the equation is

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

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

$$u_z = 2\left(\frac{\rho g \sin\alpha}{B_n}\right)^n (h-z)^n \tag{A20}$$

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

$$u(z) = \frac{2}{n+1}\left(\frac{\rho g \sin\alpha}{B_n}\right)^n \left((h-b)^{n+1} - (h-z)^{n+1}\right) \tag{A21}$$

Formulas (A19) and (A21) exactly solve the Stokes equations, and they will be used for verifying the numerical solver. Additionally they allow us to set boundary conditions which lead to glaciologically-reasonable solutions for more general shapes like that shown in Figure A1. In our set-up the inflow side will have (A21) applied as a Dirichlet condition—see equation (A12). On the outflow side we apply a normal stress also computed from the slab-on-slope solution. In fact, from (A20) above, and the facts that $w = 0$, $u_x = 0$, and $|D\mathbf{u}| = \frac{1}{2}u_z$, we find that because $\hat{\mathbf{n}} = \langle 1, 0\rangle$ is the outflow outward normal,

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

This justifies formula (A13).

Different Glen exponents n will generate different solutions, but we can determine the ice hardness B_n so that these solutions at least have the same surface velocity. This allows us to create comparable solutions for any $n \geq 1$, with glaciologically-reasonable ice velocities, based on the $n = 3$ case [9]. From the ice softness value $A_3 = 3.1689 \times 10^{-24}\,\mathrm{Pa}^{-3}\,\mathrm{s}^{-1}$ we get $B_3 = (A_3)^{-1/3}$ in $\mathrm{Pa}\,\mathrm{s}^{1/3}$. We now compute B_n so that the surface velocity $u(h)$ from (A21) matches the $n = 3$ value:

$$B_n = \left(\frac{4}{n+1}\right)^{1/n}\left(\rho g \sin\alpha (h-b)\right)^{(n-3)/n} {B_3}^{3/n} \tag{A22}$$

For example, if $h - b = 400$ m and $\alpha = 0.1$ radians (about 5.7 degrees) then B_n from (A22) gives a surface velocity of $u(h) = 906.092\,\mathrm{m\,a^{-1}}$ from (A21), independent of $n \geq 1$. Considering the glaciologically-relevant end-cases $n = 1, 4$ [7], from (A22) we find $B_1 = 4.9663 \times 10^{12}\,\mathrm{Pa\,s}$ and $B_4 = 1.7320 \times 10^7\,\mathrm{Pa\,s^{1/4}}$. According to wikipedia the Newtonian viscosity $\nu = B_1/2$ is about 10^{16} times more viscous than liquid water but about 10^7 times less viscous than granite.

A.3. Weak form

Equations (A8) and (A9) are called the *strong form* of the model. An integral equation form of the same model, 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 so as to define a scalar-valued nonlinear functional F. The weak form then says that F must be zero when acting on test functions.

The significance of the weak form is two-fold:

- It has a larger space of potential solutions and thus it is more flexible with respect to discontinuities in the data. It is well-posed in the sense of always having a unique solution for glaciologically-relevant boundary values [10].
- The FE method creates test functions via local constructions which work on a mesh of triangles or quadrilaterals. Such meshes are more flexible with respect to problem geometry than are finite difference methods based on the strong form.

The latter point is of greater practical importance.

The solution to the weak form is a pair $(\mathbf{u}, p)$ with each function living in a certain function space, thus we write $\mathbf{u} \in V_D$ and $p \in Q$. (The Sobolev spaces V_D and Q are precisely-identified in [10], but we ignore such advanced mathematics in these notes.) Test functions $\mathbf{v} \in V_0$ and $q \in Q$ come from nearly the same spaces. The difference between V_D and V_0 relates to the Dirichlet boundary conditions: $\mathbf{u} \in V_D$ satisfies a nonhomogeneous inflow boundary condition (A12) while $\mathbf{v} \in V_0$ is zero there.

To give an initial definition of F we multiply (A8) by $\mathbf{v} \in V_0$ and (A9) by $q \in Q$, then add and integrate:

$$F(\mathbf{u}, p; \mathbf{v}, q) = \int_\Omega -\left(\nabla \cdot \left(B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u}\right)\right) \cdot \mathbf{v} + \nabla p \cdot \mathbf{v} - \rho \mathbf{g} \cdot \mathbf{v} - (\nabla \cdot \mathbf{u})\, q \qquad \text{(A23)}$$

However, the desired definition of F balances the number of derivatives on $(\mathbf{v}, q)$ versus $(\mathbf{u}, p)$, so we need integration-by-parts. For this step, recall the product rule $\nabla \cdot (f\mathbf{X}) = \nabla f \cdot \mathbf{X} + f \nabla \cdot \mathbf{X}$ and the divergence theorem $\int_\Omega \nabla \cdot \mathbf{X} = \int_{\partial\Omega} \mathbf{X} \cdot \hat{\mathbf{n}}$. Denoting $\tau = B_n |D\mathbf{u}|^{\frac{1}{n}-1} D\mathbf{u}$ for typographical convenience we have

$$\begin{aligned}
\int_\Omega (\nabla \cdot \tau) \cdot \mathbf{v} &= \sum_{j=1}^3 \int_\Omega \nabla \cdot (\tau_{j\circ})\, v_j = \sum_{j=1}^3 \int_\Omega \nabla \cdot (\tau_{j\circ} v_j) - \tau_{j\circ} \nabla v_j \\
&= \sum_{j=1}^3 \int_{\partial\Omega} (\tau_{j\circ} v_j) \cdot \hat{\mathbf{n}} - \int_\Omega \tau_{j\circ} \cdot \nabla v_j = \int_{\partial\Omega} (\tau \hat{\mathbf{n}}) \cdot \mathbf{v} - \int_\Omega \operatorname{tr}(\tau \nabla \mathbf{v})
\end{aligned}$$

where $\circ$ denotes a vector entry index and $\tau_{j\circ}$ denotes the jth row of τ. Here $\nabla \mathbf{v}$ defines a 3×3 matrix,

$$\nabla \mathbf{v} = \left[\begin{array}{c|c|c} & & \\ \nabla v_1 & \nabla v_2 & \nabla v_3 \\ & & \end{array}\right] = \begin{bmatrix} (v_1)_{x_1} & (v_2)_{x_1} & (v_3)_{x_1} \\ (v_1)_{x_2} & (v_2)_{x_2} & (v_3)_{x_2} \\ (v_1)_{x_3} & (v_2)_{x_3} & (v_3)_{x_3} \end{bmatrix}$$

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 $\operatorname{tr}(AB)$ [10].) 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 (A23) with a normal stress boundary integral:

$$F(\mathbf{u},p;\mathbf{v},q) = -\int_{\partial\Omega}(\sigma\hat{\mathbf{n}})\cdot\mathbf{v} + \int_\Omega \operatorname{tr}(\tau D\mathbf{v}) - p(\nabla\cdot\mathbf{v}) - (\nabla\cdot\mathbf{u})\,q - \rho\mathbf{g}\cdot\mathbf{v} \tag{A24}$$

(We have denoted $\sigma = \tau - pI$ for brevity.) Now $\mathbf{u},\mathbf{v}$ appear with at most first derivatives and p,q appear without 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 (A24) are zero. Conditions (A11), (A13) now completely eliminate the unknown solution $\mathbf{u},p$ from the boundary integral.

The above computations yield our final formula for the nonlinear functional:

$$\begin{aligned} F(\mathbf{u},p;\mathbf{v},q) = &\int_\Omega B_n|D\mathbf{u}|^{\frac{1}{n}-1}\operatorname{tr}(D\mathbf{u}D\mathbf{v}) - p(\nabla\cdot\mathbf{v}) - (\nabla\cdot\mathbf{u})\,q \\ &- \int_\Omega \rho\mathbf{g}\cdot\mathbf{v} - \int_{\{\text{outflow}\}} C_{\text{out}}\rho g\cos\alpha(h-z)v \end{aligned} \tag{A25}$$

The last two integrals can be regarded as source terms. For example, if the inflow velocity is zero and if we replace the source terms by zero—no gravity or outflow stress—then the unique solution is $\mathbf{u} = 0$ and $p = 0$.

In conclusion 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 (A25) is zero in all test function directions:

$$F(\mathbf{u},p;\mathbf{v},q) = 0 \qquad \text{for all } \mathbf{v}\in V_0 \text{ and } q\in Q \tag{A26}$$

This weak formulation is proven in [10, Theorem 3.8] to be well-posed under reasonable assumptions about the domain Ω and the boundary data. (These assumptions are satisfied in the cases we consider.) Now our goal is to accurately-approximate $(\mathbf{u},p)$ using an FE method.

A.4. Finite element method

This section gives a very abbreviated summary of our method to solve the Stokes equations. Better coverage of FE methods is in the references [3, 4, 5]. The method here is actually applied through calls to Firedrake [12], so 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 (A26) with finite-dimensional subspaces constructed locally from triangles in the mesh. The approximate solution $(\mathbf{u},p)$ is sought from these subspaces, and functional F in (A25) is computed on test functions $(\mathbf{v},q)$ from these subspaces.

Requiring F to be zero over a basis of the subspaces defines the nonlinear *discrete Glen-Stokes equations*. We solve this nonlinear system of algebraic equations by Newton's method, with preconditioned Krylov iterations to solve the linear equations at each Newton step, using the PETSc solver library [2, 4] living underneath Firedrake.

The triangular mesh $\mathcal{T}_h$ is stored in a `.msh` file generated by Gmsh; see section A.6 below. The triangulation $\mathcal{T}_h$ covers the domain Ω by a finite set of K non-overlapping open triangles $\triangle_k$. The subscript "h" denotes the maximum diameter of the triangles. The N_1 vertices of the triangles are the nodes of the mesh, including nodes on $\partial\Omega$. The nodes are located at coordinates (x_i, z_i).

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, denoted P_1 for the pressure and $(P_2)^2$ for velocity. (Other choices can be made at run-time.) Using such paired spaces is called a *mixed method*; the particular choice here is called P_2-P_1 or 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. Instead of using three degrees of freedom $\{a, b, c\}$ to describe such a function, the three vertices of the triangle are the preferred degrees of freedom (Figure A2). That is, any P_1 function on the triangle is determined by its values at the 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 preferred six degrees of freedom at the vertices and edge midpoints [5].

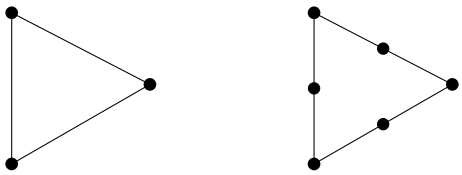

FIGURE A2. P_1 and P_2 elements.

A continuous 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. The space of such functions is a subspace $Q^h \subset Q$ of the pressure function space Q used in the weak form (A26); we say Q^h is a P_1 space.

A continuous function on Ω which is piecewise-quadratic on each triangle, i.e. in P_2 (Figure A2), is determined by its values at the nodes plus the edge midpoints [5]. Let N_2 denote the number of all such nodes at which the solution is sought. (The meshes used in numerical examples have Dirichlet boundaries, and these nodes are not included in N_2.) A small example of this kind of mesh is shown in Figure A3.

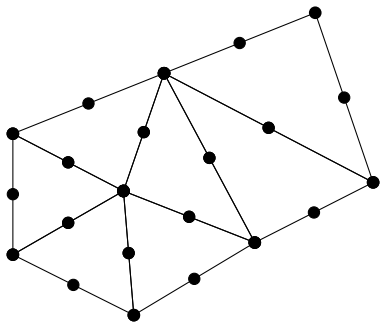

FIGURE A3. A mesh with $K = 7$ elements, $N_1 = 8$ nodes, and $N_2 = 21$ edges.

We 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 boundaries, i.e. where (A10) and (A12) apply. Thus functions from V_0^h are 2D vector fields whose components are piecewise-quadratic scalar functions, i.e. P_2 functions. We define V_D^h to be the nearly the same space except satisfying (A12) on the inflow boundary. Thus we have two P_2 velocity subspaces used in the weak form (A26): $V_0^h \subset V_0$, $V_D^h \subset V_D$.

"Hat" functions $\psi_j(x,z)$ form a convenient pressure test function basis (of Q^h). Such functions are linear on each triangle, continuous on Ω, equal to one at the one node (x_j, z_j), and otherwise zero at the nodes, so that $\psi_j(x_i, z_i) = \delta_{ij}$. (Figure 1.6 in [5] shows such a hat function.) The set $\{\psi_j\}$ is a set of N_1 linearly-independent functions in Q^h. Similarly, for every P_2 node which is not in the base or inflow boundary, whether a triangle vertex or edge midpoint, one defines a pair of basis functions for the velocity test space V_0^h. These are pairs of piecewise-quadratic hat functions, $(\phi_j(x,z), 0)$ or $(0, \phi_j(x,z))$; Figure 1.7 in [5] suggests how such P_2 hat functions would look. These hat functions form a set of $2N_2$ linearly-independent functions in V_0^h.

Using P_2-P_1 elements the dimension of the velocity space is always higher than the pressure space, and often much higher. For example, for the $N_1 = 356$ vertex mesh shown in Figure A1, 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.

The FE method itself, a finite-dimensional approximation of the weak form (A26), can now be stated. It seeks $\mathbf{u}^h \in V_D^h$ and $p^h \in Q^h$ so that

$$F(\mathbf{u}^h, p^h; \mathbf{v}^h, q^h) = 0 \qquad \text{for all } \mathbf{v}^h \in V_0^h \text{ and } q^h \in Q^h \tag{A27}$$

Note that we have merely added "h" superscripts to all continuum quantities! The nonlinear functional F is unchanged, and it is computed concretely, though also approximately, by quadrature [4, 5]. The FEM solution to (A27) can be shown to be well-posed by the same theory that applies to the continuum problem [10, Theorem 4.3]. By linearity of F in the $\mathbf{v}, q$ positions, it suffices to consider only a basis of test functions from V_0^h and Q^h. A system of nonlinear, algebraic equations, the (sparse) discrete Glen-Stokes problem, is thus formed by requiring (A27) to hold for all basis functions identified above. These equations are assembled element-by-element in the sense that for each triangle $\triangle_k$ the contribution from that triangle is added to the correct equation [4, Chapter 10].

Nonlinear system (A27) is solved by Newton's method [4, 11]. At each Newton step we solve a sparse and indefinite linear system. These systems have a well-known block structure (Figure A4), so we apply a Schur complement preconditioner in the GMRES Krylov iteration [5, 8]. This buzzword salad is an indication of the complexity of solver technology, and we abandon further explanation here, but see [4, Chapter 14].

Our Python codes define the nonlinear residual function F in (A25) using the Firedrake/FEniCs domain-specific language for describing such weak forms [1]. Then we call the Firedrake method `solve()` on (A27) [12]. A practical view of our codes, including the sequence in which they are used in practice, and how they are divided into modules, is in the final section of these notes.

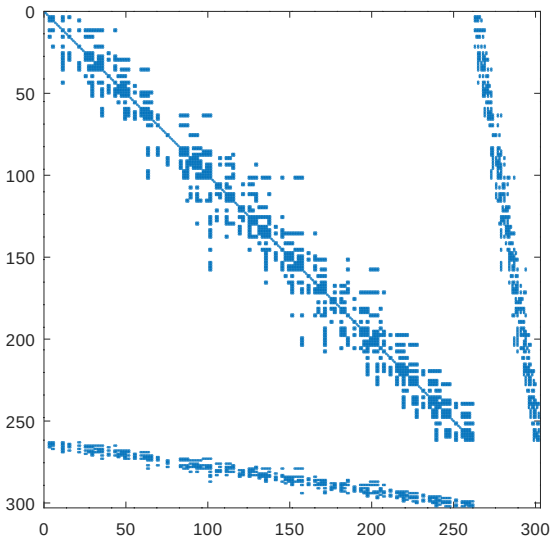

FIGURE A4. Sparsity pattern of the Newton step linear system, when solving the Stokes equations on a coarse mesh ($K = 52, N_1 = 40, N_2 = 91$).

A.5. SURFACE KINEMATICAL EQUATION

A glacier will change shape as it flows, and as its surface simultaneously interacts with the climate (precipitation and melting). We also want a model which includes this action! So now suppose that the domain on which the Stokes equations apply is time-dependent:

$$\Omega^t = \big\{(x,z) \,\big|\, b(x) < z < s(x,t)\big\} \tag{A28}$$

Here $z = b(x)$ is the base elevation and $z = s(x,t)$ is the ice surface elevation.

The time-dependent surface $z = s(x,t)$ evolves according to the surface kinematical equation, (41) in the notes, using ice velocity $\mathbf{u} = \langle u(x,z,t), w(x,z,t)\rangle$:

$$s_t = a(x,t) - u(x,s,t)s_x + w(x,s,t) \tag{A29}$$

Here $a(x,t)$ is the climatic mass balance in units of ice-equivalent $\mathrm{m\,s^{-1}}$. Informally, (A29) determines the change in surface elevation $\Delta s \approx s_t\,\Delta t$ from the climatically added/removed ice $a\,\Delta t$, *plus* the component of the ice motion in the outward (upward) normal direction $\mathbf{n} = \langle -s_x, 1\rangle$. Thus $\Delta s \approx (a + \mathbf{n}\cdot\mathbf{u}|_{z=s})\,\Delta t$.

Equation (A29) applies on the time-dependent ice surface $\Gamma^t = \big\{(x,z) \,\big|\, z = s(x,t)\big\}$. The surface Γ^t is also the zero level surface of the function

$$\Psi(x,z,t) = z - s(x,t)$$

[9, pp. 65–66]; we must regard Ψ as being defined on the closure of Ω^t, including the surface Γ^t. Since the spatial gradient $\nabla\Psi = \langle -s_x, 1\rangle = \mathbf{n}$ is the same outward normal along Γ^t, we have

$$s_t = a + \mathbf{u}|_{\Gamma^t}\cdot(\nabla\Phi)|_{\Gamma^t} \tag{A30}$$

In Firedrake the advantage of using (A30), over (A29), is that it is easier to compute the gradient of the 2D scalar field $\Psi(x,z,t_n)$, and evaluate it along the boundary, than it is to differentiate the surface elevation function $s(x,t_n)$.

In our examples below, because we are studying ice fluid dynamics in isolation, we set $a = 0$. Scientific questions generally require nontrivial models for a. Given such a model,

implementation of nonzero values for $a(x,t)$ is straightforward if applied in an explicit (time-split) manner, which is what we do here.

Regarding the base, in our simplified model the elevation $b(x)$ is time-independent, and the base does not slide, nor does it melt/refreeze liquid water, and thus the base kinematical equation, (42) in the notes, reduces to "0 = 0." Modifying our scheme to apply a nontrivial basal mass balance would not be difficult in our explicit time-stepping paradigm.

We can now describe our explicit time-stepping strategy for moving the surface of a glacier. Namely, we take a time step of the surface kinematical equation (A30) by smoothly displacing the mesh in the vertical direction only. The mesh geometry is given by scalar coordinate fields (x,z), defined on the nodes of the mesh. At time t_n the ice surface elevation $s^n(x) \approx s(x,t_n)$ equals the value of the coordinate field z on the top boundary of the time t_n mesh Ω^n. We determine the boundary value of a Laplace equation problem, for the vertical displacement r^n of every node in the mesh, from (A30).

Given the current (t_n) geometry of the ice and a time step $\Delta t_n > 0$, apply this sequence:

1. Solve the weak-form Stokes equation (A26) on the current mesh Ω^n to compute current velocity and pressure values $(\mathbf{u}^n, p^n)$.
2. From Ω^n also generate the piecewise-linear surface elevation function $s^n(x)$. Then evaluate $\Phi^n(x,z) = z - s^n(x)$ on Ω^n and compute:
$$\Delta s^n(x,z) = (a(x,t_n) + \nabla\Phi^n(x,z) \cdot \mathbf{u}^n(x,z))\,\Delta t_n \tag{A31}$$
Note Δs^n is defined on all of Ω^n, not just the glacier surface.
3. Solve a Laplace equation problem for the vertical displacement field r^n:
$$\begin{aligned} -\nabla^2 r^n &= 0 && \text{on } \Omega^n \\ r^n &= \Delta s^n && \text{(Dirichlet) on the top boundary } \Gamma^n \\ r^n &= 0 && \text{(Dirichlet) on inflow and base of } \partial\Omega^n \\ \nabla r^n \cdot \mathbf{n} &= 0 && \text{(Neumann) on outflow of } \partial\Omega^n \end{aligned} \tag{A32}$$
This linear problem is solved in weak form: $\int_{\Omega^n} \nabla r^n \cdot \nabla q = 0$ for all test functions q with zero values on the Dirichlet part of the boundary $\partial\Omega^n$.
4. Update the mesh coordinates using vertical-only displacement by r^n:
$$x^{n+1} = x^n, \quad z^{n+1} = z^n + r^n \tag{A33}$$

At the end of this sequence the new top boundary Γ^{n+1} is the updated surface $z = s^{n+1}(x)$. By (A31) the sequence has done an explicit step of (A29):
$$s^{n+1}(x) = s^n(x) + \Delta t_n\,(a(x,t_n) - u(x,s^n,t_n)s^n_x + w(x,s^n,t_n))$$

The significance of Dirichlet problem (A32) is that the entire mesh is *smoothly* displaced, in the vertical direction only by (A33), because solutions of the Laplace equation are smooth. Note r^n minimizes $\int |\nabla r^n|^2$ over functions with the given boundary values. This way of displacing the mesh avoids shearing the mesh above sharp discontinuities in base topography, for example.

Our scheme uses fixed time step $\Delta t > 0$ and is explicit. At best it can be conditionally stable, and in practice that *is* what is observed, but no quantiative time step restriction is known; an *a priori* method for setting the time step is not known. For an SIA solver a

sufficient condition like "$D\Delta t/\Delta x^2 < 1$" applies, where Δx is a representative mesh spacing and D is some diffusivity parameter. However, there is no literature which connects the D from the SIA (see the notes) with Stokes time-stepping, or otherwise supplies a stability restriction for surface-evolving Stokes models.

Two methods of addressing this numerical modeling weakness have, I think, barely been started by the glacier modeling community:

(i) Extensively test various configurations to develop an empirical time-step restriction, depending on size and aspect-ratio of elements in some complicated way.

(ii) Solve the Stokes and surface kinematical equations in implicit-step form, as a coupled system.

I am working on the latter!

A.6. Implementation in Python codes

We implement the above numerical solution method in Python codes. These codes use the Firedrake FE [12] and PETSc solver libraries [2, 4]. Here is a simple example solving a time-independent 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
```

In a time-stepping run one supplies the time step in days (`-deltat`) and the number of steps (`-m`). For example, the following run generates the result shown in Figure A1:

```
(firedrake) $ ./flow.py -mesh glacier.msh -deltat 20.0 -m 100
```

The `README.md` file documents usage more thoroughly.

As shown in Figure A5 there are a total of five Python codes:

- `domain.py`: This writes an outline of the initial domain Ω^0 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`. This module also exports a function `getdomaindims()` which dynamically-extracts dimensions from a changing mesh.
- `flow.py`: This driver reads user options, reads the mesh from a Gmsh `.msh` file, uses a `MomentumModel` object to solve the Stokes problem, optionally calls `meshmotion.py` to do the above time-stepping strategy, and writes the solution into a Paraview-readable `.pvd` file.
- `meshmotion.py`: This module assembles and solves linear Dirichlet problem (A32) for vertical mesh displacement using Firedrake's `solve()`, and it updates the mesh coordinates as described. This solver has option prefix `vd_`.
- `momentummodel.py`: Implements `class MomentumModel`. Most actual ice physics is isolated here, including physical constants and the Stokes problem weak form

(A25). The `solve()` method of this class assembles and solves the nonlinear Glen-Stokes problem (A26) using Firedrake's `solve()` command. This solver has option prefix `s_`.

- `surfaceutils.py`: Provides functions which extract boundary values of solution fields and mesh coordinates. Note that the surface values of the solution can be visualized in a separate image file (e.g. `.png`) using option `-osurface`.

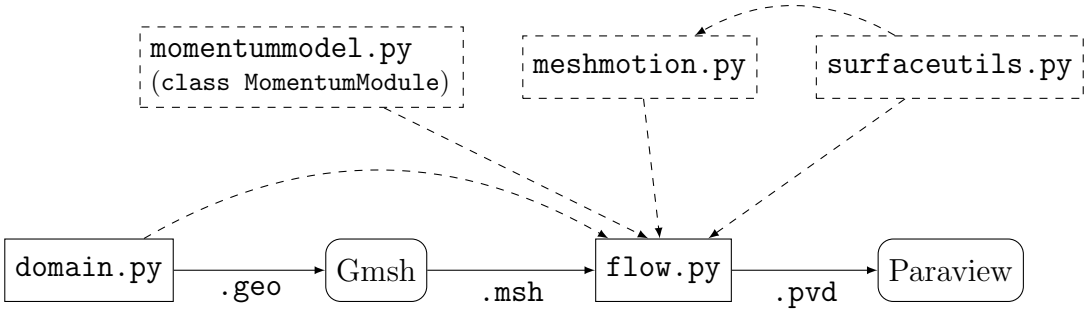

FIGURE A5. Users interact with tools (solid outlines) in the given order.

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] D. L. GOLDSBY AND D. L. KOHLSTEDT, *Superplastic deformation of ice: experimental observations*, J. Geophys. Res., 106 (2001), pp. 11017–11030.

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

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

[10] 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.

[11] C. KELLEY, *Solving Nonlinear Equations with Newton's Method*, SIAM Press, Philadelphia, 2003.

[12] 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.