C PSEUDOCODE

We give pseudocode for surface domains, expressed via a halfedge mesh data structure encoding a triangle mesh $M = (V, E, F)$. We use $\overrightarrow{ij}$ to denote the halfedge from i to j. We assume only that meshes have been specified via intrinsic quantities including edge lengths and corner angles, which we denote using the notation defined in §5.1. Subroutines not defined here are described in the list below. For simplicity, we assume here that M is oriented.

- $\textsc{Orientation}(\overrightarrow{ij})$ — returns +1 if the orientation of halfedge $\overrightarrow{ij}$ matches the canonical orientation of its edge ij, and −1 otherwise.
- $\textsc{Face}(p)$ — returns a face ijk that the barycentric point p lies within.
- $\textsc{SharedHalfedge}(A, B)$ — returns the halfedge going from element A to B, which may be vertices or barycentric points, if any.
- $\textsc{SharedFace}(A, B)$ — returns a face shared by mesh elements A and B, if any. The elements A and B may be vertices, edges, faces, or barycentric points.
- $\textsc{BarycentricVector}(p_A, p_B)$ — returns a barycentric vector defined by the barycentric points p_A and p_B as its endpoints. If p_A and p_B coincide with vertices, the barycentric vector lies on an edge; otherwise, it lies in a face.
- $\textsc{BarycentricVectorInFace}(\overrightarrow{ij}, ijk)$ — returns the barycentric vector defined by the endpoints of halfedge $\overrightarrow{ij}$, with coordinates expressed with respect to face ijk.
- $\textsc{BarycentricCoordsInFace}(p, ijk)$ — returns the barycentric coordinates of the barycentric point p with respect to face ijk.
- $\textsc{BarycentricCoordsInSomeFace}(p)$ — returns the barycentric coordinates of the barycentric point p with respect to one its containing faces, along with the face itself.
- $\textsc{BarycentricCoordsInFace}(v, ijk)$ — returns the barycentric coordinates of the barycentric vector v with respect to face ijk.
- $\textsc{Norm}(M, v)$ — returns the norm of the barycentric vector v defined on triangle mesh M.
- $\textsc{Dot}(M, v_A, v_B)$ — returns the inner product $\langle v_A, v_B\rangle \in \mathbb{R}$ between two barycentric vectors v_A, v_B defined on triangle mesh M.
- $\textsc{Rotated90}(M, v)$ — returns the barycentric vector v, rotated counterclockwise 90° in its local tangent plane on mesh M.
- $\textsc{SolveSparseSquare}(\mathsf{A}, \mathsf{b})$ — solves the sparse square linear system $\mathsf{Ax} = \mathsf{b}$, returning x.
- $\textsc{SolveSparsePositiveSemidefinite}(\mathsf{A}, \mathsf{b})$ — solves the sparse positive semidefinite linear system $\mathsf{Ax} = \mathsf{b}$, returning x (and picking an arbitrary shift if A has constants in its null space).

Algorithm 1 $\textsc{SolveGeneralizedSignedDistance}(M, \Omega, t, C)$

Input: Points and/or curves Ω on a triangle mesh M, diffusion time t, and constraints C.

Output: The generalized signed distance function ϕ to Ω.

```
1: $X_t \leftarrow \textsc{IntegrateVectorHeatFlow}(M, \Omega, t)$
2: $Y_t \leftarrow \textsc{Normalize}(X_t)$
3: $\phi \leftarrow \textsc{IntegrateVectorField}(M, Y_t, C)$
4: return $\phi$
```

Algorithm 2 $\textsc{IntegrateVectorHeatFlow}(M, \Omega, t)$

Input: Integrate the vector heat flow in Equation 12 for time t on the triangle mesh $M = (V, E, F)$, with initial conditions defined by the geometry Ω.

Output: The diffused vector field $X_t \in \mathbb{C}^{|E|}$.

```
1: $L^\nabla \leftarrow \textsc{CrouzeixRaviartConnectionLaplacian}(M)$
2: $M \leftarrow \textsc{CrouzeixRaviartMassMatrix}(M)$
3: $X_0 \leftarrow \textsc{BuildSource}(M, \Omega)$
4: $X_t \leftarrow \textsc{SolveSparsePositiveSemidefinite}(M + tL^\nabla, X_0)$
5: return $X_t$
```

Algorithm 3 $\textsc{Normalize}(M, X)$

Input: A vector field $X \in \mathbb{C}^{|E|}$ expressed in the edge basis defined in §5.2, defined on triangle mesh $M = (V, E, F)$.

Output: The normalized vector field $Y \in \mathbb{R}^{|F|\times 3}$, sampled onto face barycenters and encoded via barycentric vectors.

```
1: $Y \leftarrow 0^{|F|\times 3}$
2: for $pqr \in F$ do
3:     $y \leftarrow 0^3$
4:     for $ijk \in C(pqr)$ do                ▷$C$: circular shifts
5:         $s_{\overrightarrow{ij}} \leftarrow \textsc{Orientation}(\overrightarrow{ij})$
6:         $\tau \leftarrow \textsc{BarycentricVectorInFace}(\overrightarrow{ij}, pqr) \cdot s_{\overrightarrow{ij}}$
7:         $v \leftarrow \textsc{Rotated90}(M, \tau)$
8:         $\tau$ /= $\textsc{Norm}(M, \tau)$
9:         $v$ /= $\textsc{Norm}(M, v)$
10:        $\lambda_\tau \leftarrow \textsc{BarycentricCoordsInFace}(\tau, pqr)$
11:        $\lambda_v \leftarrow \textsc{BarycentricCoordsInFace}(v, pqr)$
12:        $y$ += $\mathrm{Re}(X_{ij}) \cdot \lambda_\tau$
13:        $y$ += $\mathrm{Im}(X_{ij}) \cdot \lambda_v$
14:    $Y_{pqr} \leftarrow y$
15: return $Y$
```

Algorithm 4 $\textsc{IntegrateVectorField}(M, X, C)$

Input: A vector field $X \in \mathbb{R}^{|F|\times 3}$ defined on a triangle mesh $M = (V, E, F)$, and constraints C.

Output: The solution $\phi \in \mathbb{R}^{|E|}$ to the Poisson problem in Equation 13 satisfying the constraints C (§7).

```
1: $L \leftarrow \textsc{CotanLaplacian}(M)$
2: $b \leftarrow \textsc{Divergence}(M, X)$
3: if $C = \varnothing$ then
4:     $\phi \leftarrow -\textsc{SolveSparsePositiveSemidefinite}(L, b)$
5:     $\phi \leftarrow \textsc{Shift}(\phi, \Omega)$
6:     return $\phi$
7: if $C = \textsc{PreserveZeroLevelset}$ then
8:     $A \leftarrow \textsc{ConstraintMatrix}(\Omega)$
9:     $u \leftarrow -\textsc{SolveSparseSquare}\left(\begin{bmatrix} L & A^T \\ A & 0 \end{bmatrix}, \begin{bmatrix} b \\ 0 \end{bmatrix}\right)$
10:    $\phi \leftarrow \textsc{Shift}(u_{:|E|}, \Omega)$
11:    return $\phi$
```

Algorithm 5 $\textsc{CrouzeixRaviartConnectionLaplacian}(M)$

Input: A triangle mesh $M = (V, E, F)$.

Output: The Crouzeix-Raviart connection Laplacian $\mathsf{L}^\nabla \in \mathbb{C}^{|E|\times|E|}$ (§5.4).

```
1:  $\mathsf{L}^\nabla \leftarrow 0^{|E|\times|E|}$      ▷initialize empty sparse complex matrix
2:  for $pqr \in F$ do
3:      for $ijk \in C(pqr)$ do      ▷C: circular shifts
4:          $w \leftarrow 2\cot\theta_j^{ki}$
5:          $r_{ij,jk} \leftarrow$ EdgeRotation$(ij, jk)$
6:          $\mathsf{L}^\nabla_{ij,ij}$ += $w$
7:          $\mathsf{L}^\nabla_{jk,jk}$ += $w$
8:          $\mathsf{L}^\nabla_{ij,jk}$ −= $w \cdot \bar{r}_{ij\to jk}$
9:          $\mathsf{L}^\nabla_{jk,ij}$ −= $w \cdot r_{ij\to jk}$
10: return $\mathsf{L}^\nabla$
```

Algorithm 6 CrouzeixRaviartMassMatrix(M)

Input: A triangle mesh $M = (V, E, F)$.

Output: The Crouzeix-Raviart mass matrix $\mathsf{M} \in \mathbb{C}^{|E|\times|E|}$ (§B.3).

```
1:  $\mathsf{M} \leftarrow 0^{|E|\times|E|}$      ▷initialize empty sparse complex matrix
2:  for $pqr \in F$ do
3:      for $ij < pqr$ do $\mathsf{M}_{ij,ij}$ += $\frac{|ijk|}{3}$
4:  return $\mathsf{M}$
```

Algorithm 7 CotanLaplacian(M)

Input: A triangle mesh $M = (V, E, F)$.

Output: The positive definite cotan Laplacian $\mathsf{L} \in \mathbb{R}^{|V|\times|V|}$.

```
1:  $\mathsf{L} \leftarrow 0^{|V|\times|V|}$      ▷initialize empty sparse matrix
2:  for $pqr \in F$ do
3:      for $ijk \in C(pqr)$ do      ▷C: circular shifts
4:          $w \leftarrow \frac{1}{2}\cot\theta_k^{ij}$
5:          $\mathsf{L}_{i,i}$ += $w$
6:          $\mathsf{L}_{j,j}$ += $w$
7:          $\mathsf{L}_{i,j}$ −= $w$
8:          $\mathsf{L}_{j,i}$ −= $w$
9:  return $\mathsf{L}$
```

Algorithm 8 Divergence(M, X)

Input: A triangle mesh $M = (V, E, F)$, and vector field $\mathsf{X} \in \mathbb{C}^{|F|}$.

Output: The finite-element divergence $\mathsf{b} := \nabla \cdot \mathsf{X} \in \mathbb{R}^{|V|}$, defined per vertex.

```
1:  $\mathsf{b} \leftarrow 0^{|V|}$
2:  for $i \in V$ do
3:      for $ijk > i$ do
4:          $v_A \leftarrow$ BarycentricVectorInFace$(\overrightarrow{ij}, ijk)$
5:          $v_B \leftarrow$ BarycentricVectorInFace$(\overrightarrow{ki}, ijk)$
6:          $d_A \leftarrow$ Dot$(M, v_A, \mathsf{X}_{ijk})$
7:          $d_B \leftarrow$ Dot$(M, v_B, \mathsf{X}_{ijk})$
8:          $\mathsf{b}_i$ += $\frac{1}{2}\cot\theta_k^{ij} \cdot d_A + \frac{1}{2}\cot\theta_j^{ki} \cdot d_B$
9:  return $\mathsf{b}$
```

Algorithm 9 ConstraintMatrix(Ω)

Input: Source geometry Ω considered as a set of barycentric points $\{p_i\}$ on triangle mesh $M = (V, E, F)$.

Output: The constraint matrix $\mathsf{A} \in \mathbb{R}^{m\times|V|}$ defined in Equation 14, where m is the number of constraints.

```
1:  $m \leftarrow 0$
2:  $\lambda_0, abc \leftarrow$ BarycentricCoordsInSomeFace$(p_0)$
3:  for $p \in \Omega$ do
4:      $ijk \leftarrow$ Face$(p)$
5:      $\lambda \leftarrow$ BarycentricCoordsInFace$(p, ijk)$
6:      for $l < ijk$ do $\mathsf{C}_{m,l}$ += $\lambda_l$
7:      for $l < abc$ do $\mathsf{C}_{m,l}$ −= $(\lambda_0)_l$
8:      $m$ += 1
9:  return $\mathsf{C}$
```

Algorithm 10 BuildSource(M, Ω)

Input: Source geometry $\Omega = \{\Gamma, P\}$ consisting of a collection of curves Γ and points P, defined on triangle mesh $M = (V, E, F)$ (§5.6).

Output: The r.h.s. $\mathsf{X}_0 \in \mathbb{C}^{|E|}$ to Equation 12.

```
1:  $\mathsf{X}_0 \leftarrow 0^{|E|}$      ▷initialize empty complex vector
2:  $\mathsf{X}_0$ += BuildOrientedCurveSources$(M, \Gamma)$
3:  $\mathsf{X}_0$ += BuildUnorientedPointSources$(M, P)$
4:  return $\mathsf{X}_0$
```

Algorithm 11 BuildOrientedCurveSources(M, Γ)

Input: A collection of oriented curves $\Gamma = \{\gamma_i\}$ on triangle mesh $M = (V, E, F)$ consisting of linear segments γ_i, each defined by barycentric points sharing a face (§5.6).

Output: A source term $\mathsf{X}_0 \in \mathbb{C}^{|E|}$ encoding Γ.

```
1:  $\mathsf{X}_0 \leftarrow 0^{|E|}$      ▷initialize empty complex vector
2:  for $\gamma = (p_A, p_B) \in \Gamma$ do
3:      $\ell \leftarrow$ Length$(\gamma)$
4:      $\overrightarrow{ij} \leftarrow$ SharedHalfedge$(p_A, p_B)$
5:      if $ij$ = Null then
6:          $ijk \leftarrow$ SharedFace$(p_A, p_B)$
7:          for $ij < ijk$ do
8:              $(\mathsf{X}_0)_{ij}$ += $\ell \cdot$ CurveNormal$(M, \gamma, ij)$
9:      else
10:         $n \leftarrow \imath \cdot$ Orientation$(\overrightarrow{ij})$
11:         $(\mathsf{X}_0)_{ij}$ += $\ell \cdot n$
12: return $\mathsf{X}_0$
```

Algorithm 12 BuildUnorientedPointSources(M, P)

Input: A collection of vertices P on triangle mesh $M = (V, E, F)$.

Output: A source term $\mathsf{X}_0 \in \mathbb{C}^{|E|}$ encoding P.

```
1:  $\mathsf{X}_0 \leftarrow 0^{|E|}$      ▷initialize empty complex vector
2:  for $i \in P$ do
3:      ▷Compute angle sum.
4:      $\Theta \leftarrow 0$
5:      for $\frac{jk}{i} < i$ do $\Theta$ += $\theta_i^{jk}$
6:      ▷Add contributions per-face.
7:      for $\frac{jk}{i} < i$ do
8:          $s_{\overrightarrow{ij}} \leftarrow$ Orientation$(\overrightarrow{ij})$
9:          $s_{\overrightarrow{jk}} \leftarrow$ Orientation$(\overrightarrow{jk})$
```

```
10:         $s_{\overrightarrow{ki}} \leftarrow \textsc{Orientation}(\overrightarrow{ki})$
11:         $r_{\overrightarrow{ij}\to\overrightarrow{jk}} \leftarrow \textsc{HalfedgeRotation}(\overrightarrow{ij}, \overrightarrow{jk})$
12:         $r_{\overrightarrow{ki}\to\overrightarrow{ij}} \leftarrow \textsc{HalfedgeRotation}(\overrightarrow{ki}, \overrightarrow{ij})$
13:         $n \leftarrow \frac{\imath(1-e^{\imath\theta_i^{jk}})}{\Theta}$
14:         $(X_0)_{ij}$ += $s_{\overrightarrow{ij}} \cdot n$
15:         $(X_0)_{jk}$ += $s_{\overrightarrow{jk}} \cdot \overline{r}_{\overrightarrow{ij}\to\overrightarrow{jk}} \cdot n$
16:         $(X_0)_{ki}$ += $s_{\overrightarrow{ki}} \cdot r_{\overrightarrow{ki}\to\overrightarrow{ij}} \cdot n$
17: return $X_0$
```

Algorithm 13 Shift(M, f, Ω)

Input: A function $\mathrm{f} \in \mathbb{R}^{|V|}$ and source geometry $\Omega = \{\Gamma, P\}$, defined on triangle mesh $M = (V, E, F)$.
Output: The function $\mathrm{g} \in \mathbb{R}^{|V|}$ shifted to average zero along Ω.

```
1: $c \leftarrow 0$
2: $L \leftarrow 0$
3: for $\gamma \in \Gamma$ do
4:     $\ell \leftarrow \textsc{Length}(M, \gamma)$
5:     $ijk, \lambda \leftarrow \textsc{Midpoint}(\gamma)$
6:     for $l < ijk$ do  $c \leftarrow \ell \cdot \lambda_l \cdot \mathrm{f}_l$
7:     $L$ += $\ell$
8: for $p \in P$ do
9:     $ijk \leftarrow \textsc{Face}(p)$
10:    $\lambda \leftarrow \textsc{BarycentricCoordsInFace}(p, ijk)$
11:    for $l < ijk$ do $c$ += $\mathrm{f}_l \cdot \lambda_l$
12:    $L$ += 1
13: $c$ /= $L$
14: $\mathrm{g} \leftarrow \mathrm{f} - c \cdot \mathbf{1}^{|V|}$
15: return g
```

Algorithm 14 EdgeRotation(ij, jk)

Input: Two edges ij and jk in face ijk.
Output: The complex number encoding the smallest rotation from the local coordinate basis at edge ij to that of edge jk. (§5.4).

```
1: $r_{\overrightarrow{ij}\to\overrightarrow{jk}} \leftarrow \textsc{HalfedgeRotation}(\overrightarrow{ij}, \overrightarrow{jk})$
2: $s_{ij\to jk} \leftarrow \textsc{Orientation}(\overrightarrow{ij}) \cdot \textsc{Orientation}(\overrightarrow{jk})$
3: $r_{ij\to jk} \leftarrow s_{ij\to jk} \cdot \overline{r}_{\overrightarrow{ij}\to\overrightarrow{jk}}$
4: return $r_{ij\to jk}$
```

Algorithm 15 HalfedgeRotation$(\overrightarrow{ij}, \overrightarrow{jk})$

Input: Two halfedges $\overrightarrow{ij}$ and $\overrightarrow{jk}$ in face ijk.
Output: The complex number encoding the smallest rotation from $e_{\overrightarrow{ij}}$ to $e_{\overrightarrow{jk}}$.

```
1: $r_{\overrightarrow{ij}\to\overrightarrow{jk}} \leftarrow -e^{-\imath\theta_j^{ki}}$
2: return $r_{\overrightarrow{ij}\to\overrightarrow{jk}}$
```

Algorithm 16 CurveNormal(M, ij)

Input: A curve segment $\gamma = (p_A, p_B)$ specified by two barycentric points p_A and p_B, and edge ij defined on triangle mesh M.
Output: The complex number $n \in \mathbb{C}$ encoding the unit normal to γ, expressed w.r.t. the local basis of ij (§5.4).

```
1: $\beta \leftarrow \textsc{BarycentricVector}(i, j)$
2: $\tau \leftarrow \textsc{BarycentricVector}(p_A, p_B)$
3: $\nu \leftarrow \textsc{Rotated90}(M, \tau)$
4: $\tau$ /= $\textsc{Norm}(M, \tau)$
5: $\nu$ /= $\textsc{Norm}(M, \nu)$
6: $n \leftarrow \textsc{Dot}(M, \nu, \beta) + \imath \cdot \textsc{Dot}(M, \tau, \beta)$
7: return $n$
```

Algorithm 17 Length(M, γ)

Input: A curve segment $\gamma = (p_A, p_B)$ specified by two barycentric points p_A and p_B, defined on the triangle mesh M.
Output: The length of γ.

```
1: $\nu \leftarrow \textsc{BarycentricVector}(p_A, p_B)$
2: $\ell \leftarrow \textsc{Norm}(M, \nu)$
3: return $\ell$
```

Algorithm 18 Midpoint(γ)

Input: A curve segment $\gamma = (p_A, p_B)$ specified by two barycentric points p_A and p_B.
Output: The barycentric point at the midpoint of γ, expressed via its containing face ijk and barycentric coordinates w.r.t. ijk.

```
1: $ijk \leftarrow \textsc{SharedFace}(p_A, p_B)$
2: $\lambda_A \leftarrow \textsc{BarycentricCoordsInFace}(p_A, ijk)$
3: $\lambda_B \leftarrow \textsc{BarycentricCoordsInFace}(p_B, ijk)$
4: return $ijk, \frac{1}{2}(\lambda_A + \lambda_B)$
```