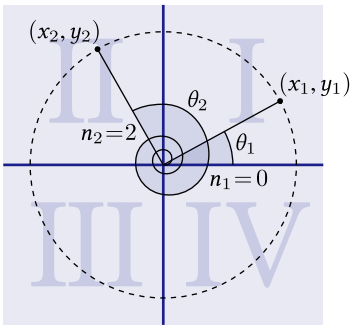
Navigating Intrinsic Triangulations (Supplemental Material)

Nicholas Sharp, Yousuf Soliman, Keenan Crane

1 Angle Representation

In this section we describe an optional numerical strategy that can be used to improve the run time performance of the signpost data structure. We stress that this strategy is *not* essential to achieving numerical robustness—nor for getting decent speed. It does however provide a moderate boost in performance, and was used for the timings presented in the main paper.



Two frequently used operations are particularly expensive when working with angles represented by floating point values: computing interior angles θ_i^{jk} from edge lengths ℓ_{ij}, and converting vectors expressed in polar coordinates (r, φ) to Cartesian coordinates $r(\cos\varphi, \sin\varphi)$. The cost of these operations stems from evaluation of transcendental functions (either arccos, or cos and sin, *resp.*). As an alternative, one can represent an angle $\theta \in \mathbb{R}$ via the Cartesian coordinates $(x, y) := (\cos\theta, \sin\theta)$ of the corresponding unit vector, *plus* an integer $n \in \mathbb{Z}$ representing a whole number of rotations, *i.e.*, $\theta = \text{atan2}(y, x) + 2n\pi$. We call this tuple (x, y, n) the *wrapped angle* representation of θ. Arithmetic operations on wrapped angles can then be defined in a natural way—for our algorithm, we require only the operations described below, operating on *positive* angles, though this scheme can easily be extended to handle negative angles.

(Comparison.) Suppose we have two angles θ_1, θ_2 encoded as wrapped angle tuples, and wish to evaluate the boolean expression $\theta_1 < \theta_2$. First, we simply check if $n_1 = n_2$; if not, we simply need to evaluate $n_1 < n_2$. If $n_1 = n_2$, then we check whether the quadrant containing θ_1 precedes the quadrant of θ_2 using the signs of their (x, y) coordinates. Finally, if these quadrants are equal, we compute the 2D cross product $s = (x_1, y_1) \times (x_2, y_2)$, and return $s < 0$. In principle this final test could be performed via robust predicates (*à la* [3] to guarantee a total ordering on wrapped angles (even in floating point), though we did not use this strategy in our implementation.

Arithmetic operations can be derived by expressing them as operations on unit complex numbers, and carefully managing the integer component. The resulting operations in real coordinates are then:

(Addition.) Suppose we want to compute the angle $\theta_1 + \theta_2$. The Cartesian coordinates of the sum are given by $(x, y) = (x_1x_2 - y_1y_2, x_1y_2 + x_2y_1)$. If $(x, y, 0) < (x_1, y_1, 0)$ (*i.e.*, if the new angle is smaller than either of the old angles, ignoring the integer part), then it must be the case that the angle "wrapped around," and our new integer part is $n = n_1 + n_2 + 1$; otherwise it is simply $n_1 + n_2$. In practice, it is also helpful to normalize the Cartesian component to prevent numerical drift away from unit norm.

(Unsigned Difference.) Given two angles θ_1, θ_2, we can compute the angle $|\theta_2 - \theta_1|$ via a nearly identical strategy. Without loss of generality, suppose $\theta_1 < \theta_2$. We first evaluate the Cartesian part $(x, y) = (x_1x_2 + y_1y_2, x_1y_2 - x_2y_1)$, normalize, and let $n = n_2 - n_1 - a$, where $a = 1$ if $(x_2, y_2, 0) < (x_1, y_1, 0)$, and $a = 0$ otherwise.

(Modulus.) Often we need to compute the modulus φ of an angle θ with respect to another angle Θ (*e.g.*, when dealing with angles expressing tangent vectors at vertices). This can be achieved by simply starting with an angle $\xi = 0$, and adding copies of Θ while $\xi + \Theta < \theta$. The modulus is then given by $\varphi = |\theta - \xi|$.

2 Adaptive Error Estimator

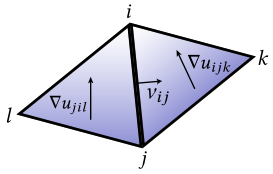
In this section we describe the local error estimators used for the adaptive mesh refinement, as described by [1, 2]. Consider the second order elliptic PDE

$$(c\mathbb{1} + \Delta)u = f$$

where c is a non-negative smooth function, and Δ is the positive semi-definite Laplace-Beltrami operator on the polyhedral surface described by the input triangulation. Note that this simultaneously includes discretizations of both PDEs considered in Section 5.2.1 (*i.e.*, the harmonic Green's function and short time heat kernel).

Consider an intrinsic triangulation M_h let u_h denote the finite element solution in some basis V_h obtained on M_h. At a high level, the local error estimator measures two things: (1) how well adapted is the mesh to represent the current FEM solution, and (2) how much information is lost by representing the right hand side in the FEM basis. Let $R_h := f - \Delta u_h - cu_h$ denote the *element residual*. The *a posteriori* error estimators are computed as follows:

- For each edge ij in the mesh, we compute the *jump residual* of ∇u_h along this edge. More precisely, consider the neighboring triangles ijk and jil, let v_{ij} denote the unit normal vector, and compute $\nabla u_h \cdot v_{ij}$ in each of the neighboring faces; the jump discontinuity, $J_{ij}(u_h)$ associated to ij is defined to be the difference of these values (*i.e.*, $J_{ij}(u_h)$ is the flux of u_h through the edge ij). This jump discontinuity describes exactly how well does the "shape" of u_h align with the elements in M_h.



$$J_{ij}(u_h) := \nabla u_{jil} \cdot v_{ij} - \nabla u_{ijk} \cdot v_{ij}$$

- Using this we can compute the *local error indicator* for a face ijk as

$$\eta_{ijk}(u_h)^2 := A_{ijk}^2 \|R_h\|_{L^2(ijk)}^2 + \sum_e \ell_e \|J_e(u_h)\|_{L^2}^2,$$

  where the sum is over all of the edges in the face ijk. The integrals above are computed by means of numerical quadrature. Note that when working with piecewise linear finite elements the jump residual is constant along an edge and so $\ell_e \|J_e(u_h)\|_{L^2}^2 = \ell_e^2 J_e(u_h)^2$.

- The last important computation we need is the *oscillation* of the data in each face. For a face $ijk \in \mathsf{F}$ we first compute by means of numerical quadrature the mean value of the element residual in ijk

$$\overline{R}_{h,ijk} := \frac{1}{A_{ijk}} \int_{ijk} R_h \, dx.$$

  We can use this to define the *oscillation* of the residual in the face ijk as

$$\mathrm{osc}_{ijk}(R_h) := A_{ijk} \|R_h - \overline{R}_{h,ijk}\|_{L^2(ijk)}.$$

  Again, this term is computed via numerical quadrature. It is exactly this oscillation term that indicates how much error is obtained by "averaging" the data f and the current solution u_h onto the finite element basis V_h.

With these computations, it is extremely easy to implement an adaptive refinement strategy by using these error estimators as a *criterion for refinement* in intrinsic Delaunay refinement (see Section 4.2). In addition to the minimum angle bounds we continue inserting vertices until the set of refined triangles, denoted by $\tilde{\mathsf{F}} \subset \mathsf{F}$, satisfies

$$\sum_{ijk \in \tilde{\mathsf{F}}} \eta_{ijk}(u_h) > \frac{1}{2} \sum_{ijk \in \mathsf{F}} \eta_{ijk}(u_h) \quad \text{and} \quad \sum_{ijk \in \tilde{\mathsf{F}}} \mathrm{osc}_{ijk}(R_h) > \frac{1}{2} \sum_{ijk \in \mathsf{F}} \mathrm{osc}_{ijk}(R_h).$$

References

[1] Khamron Mekchay and Ricardo H Nochetto. Convergence of adaptive finite element methods for general second order linear elliptic pdes. *SIAM Journal on Numerical Analysis*, 43(5):1803–1827, 2005.

[2] Pedro Morin, Ricardo H Nochetto, and Kunibert G Siebert. Convergence of adaptive finite element methods. *SIAM review*, 44(4):631–658, 2002.

[3] Jonathan Richard Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry*, 18(3):305–363, Oct 1997.