Repulsive Shells (Supplemental)

Algorithm 2 MidpointApproximation(σ, τ, α)

Input: A pair of non-intersecting triangles σ, τ given as triples of points in $\mathbb{R}^3$, and a power α for the tangent-point energy.

Output: Approximation of the tangent-point energy Φ using midpoint quadrature.

```
(a_σ, a_τ) ← (Area(σ), Area(τ))
(c_σ, c_τ) ← (Center(σ), Center(τ))
n_σ ← Normal(σ)
return a_σ a_τ |⟨n_σ, c_σ − c_τ⟩|^α / |c_σ − c_τ|^{2α}
```

Algorithm 3 AdaptiveMultipole($\sigma^0, \tau^0, \alpha, \theta$)

Input: A pair of non-intersecting triangles σ^0, τ^0 given as triples of points in $\mathbb{R}^3$, a power α for the tangent-point energy, and a parameter $\theta > 0$ for the multipole acceptance criterion (Equation 18).

Output: A multipole approximation of the tangent-point energy Φ, and its first-order derivatives $d_\sigma\Phi, d_\tau\Phi$ with respect to the vertex coordinates of σ^0 and τ^0.

```
if Intersect(σ^0, τ^0) then
    Return(∞)
Φ ← 0                                   ▹ energy approximation
d_σΦ ← 0 ∈ ℝ^3 × ℝ^3 × ℝ^3              ▹ derivative w.r.t. vertices of σ^0
d_τΦ ← 0 ∈ ℝ^3 × ℝ^3 × ℝ^3              ▹ derivative w.r.t. vertices of τ^0
Push(S, (σ^0, τ^0))                     ▹ initialize stack S with root node
while !Empty(S) do
    (σ, τ) ← Pop(S)
    if max(Diam(σ), Diam(τ)) < θ Dist(σ, τ) then
        Φ ← Φ + MidpointApproximation(σ, τ)
        d_σΦ ← d_σΦ + d_σMidpointApproximation(σ, τ) · Bary(σ, σ_0)
        d_τΦ ← d_τΦ + d_τMidpointApproximation(σ, τ) · Bary(τ, τ_0)
    else
        (σ_1, σ_2, σ_3, σ_4) ← Subdivide(σ)
        (τ_1, τ_2, τ_3, τ_4) ← Subdivide(τ)
        for i = 1, . . . , 4 do
            for j = 1, . . . , 4 do
                Push(S, (σ_i, τ_j))
return (Φ, d_σΦ, d_τΦ)
```

B ENERGY DIFFERENTIAL

Here we give explicit expressions for the first-order derivatives of the discrete tangent-point energy. Derivatives for the discrete elastic energy can be found in Heeren [2017, Section A.5]. Recall from Equation 15 that the kernel of the tangent-point energy is given by

$$K(x, y, n) := \frac{|\langle n, x - y\rangle|^\alpha}{|x - y|^{2\alpha}}. \tag{1}$$

The partial derivatives of the kernel are given by

$$\begin{aligned} \mathrm{d}_x K(x, y, n) = \alpha \frac{|\langle n, x - y\rangle|^{\alpha-1}}{|x - y|^{2\alpha}} n \\ - 2\alpha \frac{|\langle n, x - y\rangle|^{\alpha}}{|x - y|^{2\alpha+2}} (x - y) \in \mathbb{R}^3, \end{aligned} \tag{2}$$

$$\mathrm{d}_y K(x, y, n) = -\mathrm{d}_x K(x, y, n) \in \mathbb{R}^3, \tag{3}$$

and

$$\mathrm{d}_n K(x, y, n) = \alpha \frac{|\langle n, x - y\rangle|^{\alpha-1}}{|x - y|^{2\alpha}} (x - y) \in \mathbb{R}^3. \tag{4}$$

To obtain the derivatives with respect to nodal positions, we employ the chain rule—yielding the derivative computation in Algorithm 3.

C PSEUDOCODE

In this appendix we give complete pseudocode for our adaptive multipole scheme on a pair of triangles. The only methods not defined explicitly are:

- Area(σ)—returns area $\frac{1}{2}|(x_2 - x_1) \times (x_3 - x_1)|$ of a triangle σ with vertices $x_1, x_2, x_3 \in \mathbb{R}^3$.
- Center(σ)—returns triangle center $\frac{1}{3}(x_1 + x_2 + x_3)$.
- Normal(σ)—returns unit vector parallel to $(x_2 - x_1) \times (x_3 - x_1)$.
- Diam(σ)—returns the maximum edge length of σ.
- Intersect(σ, τ)—returns true if and only if σ, τ intersect.
- Dist(σ, τ)—returns the distance between triangles σ, τ, *i.e.*, the length of the shortest segment between them.
- Bary($\tilde{\tau}, \tau$)—returns for $\tilde{\tau} \subset \tau$ the barycentric coordinates of the vertices of $\tilde{\tau}$ with respect to the containing triangle τ as 3-by-3 matrix with columns corresponding to vertices of $\tilde{\tau}$

Algorithm 1 Subdivide(σ)

Input: A triangle σ given as a triples of points $x_1, x_2, x_3 \in \mathbb{R}^3$.

Output: The four triangles obtained by cutting σ along the segments between its edge midpoints.

```
(x_1, x_2, x_3) ← σ                                          ▹ get the vertices
(m_1, m_2, m_3) ← (x_1+x_2, x_2+x_3, x_3+x_1)/2              ▹ compute midpoints
σ_1 ← (x_1, m_1, m_3)
σ_2 ← (x_2, m_2, m_1)
σ_3 ← (x_3, m_3, m_2)
σ_4 ← (m_1, m_2, m_3)
return (σ_1, σ_2, σ_3, σ_4)
```

REFERENCES

Behrend Heeren. 2017. *Numerical methods in shape spaces and optimal branching patterns.* Ph. D. Dissertation. Universitäts-und Landesbibliothek Bonn.