Supplemental Material for A robust solver for elliptic PDEs in 3D complex geometries

Matthew J. Morse[a], Abtin Rahimian[b], Denis Zorin[a]

[a]*Courant Institute of Mathematical Sciences, New York University, New York, NY 10003*
[b]*Department of Computer Science, University of Colorado - Boulder, Boulder, CO 80309*

1. Kernels

Here we list the elliptic PDE's investigated in this work along with the associated kernels for their single- and double-layer potentials. In this section, $\boldsymbol{x}$ and $\boldsymbol{y}$ are in $\mathbb{R}^3$, $\boldsymbol{x}$ is the point of evaluation and $\boldsymbol{y}$ is a point on the boundary and $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{y}$. Recall that $\boldsymbol{n}$ is the outward pointing unit normal at $\boldsymbol{y}$ to the domain boundary Γ. We denote the single layer kernel, also known as the *fundamental solution* or *Green's function* of the PDE, by S and the double layer kernel by D.

1. *Laplace equation*:

$$\Delta u = 0$$
$$S(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{4\pi}\frac{1}{\|\boldsymbol{r}\|}, \quad D(\boldsymbol{x},\boldsymbol{y}) = -\frac{1}{4\pi}\frac{\boldsymbol{r}\cdot\mathbf{n}}{\|\boldsymbol{r}\|^3}$$

2. *Stokes equation*:

$$\mu\Delta u - \nabla p = 0,\ \nabla\cdot u = 0$$
$$S(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{8\pi\mu}\left(\frac{1}{\|\boldsymbol{r}\|} + \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{\|\boldsymbol{r}\|^3}\right), \quad D(\boldsymbol{x},\boldsymbol{y}) = -\frac{3}{4\mu\pi}\frac{\boldsymbol{r}\otimes\boldsymbol{r}}{\|\boldsymbol{r}\|^5}(\boldsymbol{r}\cdot\mathbf{n})$$

3. *Elasticity equation*:

$$\mu\Delta u - \frac{\mu}{1-2\nu}\nabla(\nabla\cdot u) = 0$$
$$S(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{16\pi\mu(1-\nu)}\left(\frac{3-4\nu}{\|\boldsymbol{r}\|} + \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{\|\boldsymbol{r}\|^3}\right),$$
$$D(\boldsymbol{x},\boldsymbol{y}) = -\frac{1-2\nu}{8\mu(1-\nu)}\left(\frac{1}{\|\boldsymbol{r}\|^3}(\boldsymbol{r}\otimes\boldsymbol{n} - (\boldsymbol{r}\cdot\boldsymbol{n})I - \boldsymbol{n}\otimes\boldsymbol{r}) - \frac{3}{1-2\nu}\frac{(\boldsymbol{r}\cdot\boldsymbol{n})(\boldsymbol{r}\otimes\boldsymbol{r})}{\|\boldsymbol{r}\|^5}\right)$$

2. Find the closest point on a patch

We include our algorithm to find the closest point $\boldsymbol{y}$ on a patch $\boldsymbol{P}$ to a point $\boldsymbol{x}\in\mathbb{R}^3$ in the section for completeness. For a surface or quadrature patch $\boldsymbol{P}$ and point $\boldsymbol{x}\in\mathbb{R}^3$, we need to compute a point $\boldsymbol{y} = \boldsymbol{P}(s^*,t^*)$ such that

$$(s^*,t^*) = \underset{(s,t)\in[-1,1]^2}{\arg\min}\ \|\boldsymbol{x} - \boldsymbol{P}(s,t)\|_2^2 = \underset{(s,t)\in[-1,1]^2}{\arg\min}\ \boldsymbol{r}(s,t)\cdot\boldsymbol{r}(s,t) \tag{1}$$

Email addresses: mmorse@cs.nyu.edu (Matthew J. Morse), arahimian@acm.org (Abtin Rahimian), dzorin@cs.nyu.edu (Denis Zorin)

where $\boldsymbol{r} = \boldsymbol{r}(s,t) = \boldsymbol{x} - \boldsymbol{P}(s,t)$; let $g(s,t) = \boldsymbol{r} \cdot \boldsymbol{r}$. We first consider the unconstrained problem

$$(s^*, t^*) = \underset{(s,t)\in\mathbb{R}^2}{\arg\min} \|\boldsymbol{x} - \boldsymbol{P}(s,t)\|_2^2 = \underset{(s,t)\in\mathbb{R}^2}{\arg\min}\, \psi(s,t) \tag{2}$$

We solve this optimization problem with Newton's method. The first and second derivatives of ψ can be evaluated efficiently, since they are polynomials of fixed order. The gradient and Hessian of the objective function are:

$$\nabla\psi = \begin{pmatrix} -\boldsymbol{P}_s \cdot \boldsymbol{r} \\ -\boldsymbol{P}_t \cdot \boldsymbol{r} \end{pmatrix}, \quad \nabla^2\psi = \begin{pmatrix} \boldsymbol{P}_s \cdot \boldsymbol{P}_s - \boldsymbol{r} \cdot \boldsymbol{P}_{ss} & \boldsymbol{P}_s \cdot \boldsymbol{P}_t - \boldsymbol{r} \cdot \boldsymbol{P}_{st} \\ \boldsymbol{P}_s \cdot \boldsymbol{P}_t - \boldsymbol{r} \cdot \boldsymbol{P}_{st} & \boldsymbol{P}_t \cdot \boldsymbol{P}_t - \boldsymbol{r} \cdot \boldsymbol{P}_{tt} \end{pmatrix}. \tag{3}$$

The optimality conditions are

$$\boldsymbol{P}_s^* \cdot \boldsymbol{r}^* = 0, \quad \boldsymbol{P}_t^* \cdot \boldsymbol{r}^* = 0, \quad (u,v) = (s^*, t^*). \tag{4}$$

at a local optimum (s^*, t^*).

Let $\psi_i = \psi(s_i, t_i)$, where (s_i, t_i) is the value of the solution during the ith iteration of Newton's method. To solve for the descent direction in Newton's method, we need to solve

$$\nabla^2\psi_i\, \eta_i = -\nabla\psi_i \tag{5}$$

where $\eta_i = (\Delta s_i, \Delta t_i)$ is the ith Newton update to (s_i, t_i) such that

$$s_{i+1} = \alpha_i \Delta s_i + s_i, \quad t_{i+1} = \alpha_i \Delta t_i + t_i \tag{6}$$

We use four iterations of a backtracking line search with an Armijo condition to compute the step length α_i to ensure an appropriate size step is taken in case the initial guess is outside the region of quadratic convergence. We compute the solution (s^*, t^*) by iterating

$$(s_n, t_n) = (s_{n-1}, t_{n-1}) + \alpha_{n-1}\eta_{n-1}, \quad \text{while } \boldsymbol{P}_s \cdot \boldsymbol{r} > \epsilon_{\text{opt}}, \quad \boldsymbol{P}_t \cdot \boldsymbol{r} > \epsilon_{\text{opt}}, \tag{7}$$

until convergence, i.e., $\psi_i \approx \epsilon_{\text{opt}}$, $\boldsymbol{r} \approx \boldsymbol{n}(\boldsymbol{y})$.

If $(s^*, t^*) \in (-1,1)^2$, then the solution to the unconstrained problem is also the solution to the constrained problem. However, if the closest point lies in $\mathbb{R} \setminus [-1,1]^2$, we need to ensure the inequality constraints are satisfied. Additionally, if (s^*, t^*) is on the boundary of $[-1,1]^2$, either s^* or t^* should be exactly zero; with the optimization scheme above, we can only claim that $|s^*| < \epsilon_{\text{opt}}$ (similarly for t^*). To address both of these troubles, we can solve a one-dimensional projection of Eq. (5) on to the boundary of $[-1,1]^2$. For example, to find the closest point along the edge $v = 0$, the Newton iteration becomes

$$s_n = s_{n-1} + \alpha_{n-1} \frac{-\boldsymbol{P}_s \cdot \boldsymbol{r}}{\boldsymbol{P}_s \cdot \boldsymbol{P}_s - \boldsymbol{r} \cdot \boldsymbol{P}_{ss}}, \tag{8}$$

where $\boldsymbol{P}_s$, $\boldsymbol{P}_{ss}$ and $\boldsymbol{r}$ are evaluated at s_{n-1}. Since the boundary is composed of $[-1,t], [1,t], [s,-1], [s,1]$ for $s, t \in [-1,1]$, we solve Eq. (8) once for each interval.

This final algorithm to compute the closest point is as follows:

1. We solve Eq. (5) on an extended parameter domain $[-1-c, 1+c]^2$, and terminate the Newton iteration if (s_i, t_i) walks outside this boundary. If the Newton iteration terminates inside $[-1,1]^2$, then we've found the closest point. We typically choose $c = .2$.

2. If the solution is outside $[-1,1]^2$, we solve Eq. (5) along each component of the boundary of $[-1,1]^2$, also on an extended parameter domain $[-1-c, 1+c]$, by choosing an initial guess contained within the interval. The solution to these four problems that yields a minimal distance to $\boldsymbol{x}$ to used as the closest point, if the solution is inside $[-1,1]$.

3. If the closest point on the boundary is still outside of $[-1,1]^2$, the closest point to $\boldsymbol{x}$ is chosen from $\boldsymbol{P}(-1,-1), \boldsymbol{P}(-1,1), \boldsymbol{P}(1,-1)$, and $\boldsymbol{P}(1,1)$ closest to $\boldsymbol{x}$.

This gives us an algorithm to compute the closest point on a quadrature patch $\boldsymbol{P}$ to $\boldsymbol{x}$. The 1D and 2D Newton minimizations converge in ten iterations on average.

3. Complexity

The parameters that directly impact complexity are:

- The number of patches N *after* admissibility refinement. This is a function of N_{init}, the geometry of Γ, the definition of f, and the choices of parameters a and b in check point construction.
- Quadrature order q and the degree of smoothness k of Γ and f. We assume that k is sufficiently high to obtain optimal error behavior for a given q by letting $k = 2q$ in [MRZ20, Heuristic 1].
- `hedgehog` interpolation order p.
- The numbers of evaluation points in different zones $\mathcal{N}_{\text{far}}$, $\mathcal{N}_{\text{inter}}$, and $\mathcal{N}_{\text{near}}$, with $\mathcal{N}_{\text{tot}} = \mathcal{N}_{\text{far}} + \mathcal{N}_{\text{inter}} + \mathcal{N}_{\text{near}}$.

The complexity is also affected by the geometric characteristics of Γ. These include:

- The *maximum patch length* $L_{\max} = \max_{\boldsymbol{P}} L(\boldsymbol{P})$
- The *relative minimal patch length* $L_{\min} = \beta_0 L_{\max}$, $\beta_0 \leq 1$.
- The *minimal feature size relative to* $L_{\max}$, $\ell_{min} = \alpha_0 L_{\max}$, which is defined in terms of the *local feature size* and the *medial axis* of Γ. The medial axis of Γ, denoted $M(\Gamma)$, is the set of points in $\mathbb{R}^3$ with more than one closest point on Γ. For $\boldsymbol{y} \in \Gamma$, the local feature size $\ell(\boldsymbol{y})$ is the distance from $\boldsymbol{y}$ to $M(\Gamma)$. We assume that the local feature size is bounded below by $\alpha_0 L_{\max}$, i.e., $\ell(\boldsymbol{y}) \geq \alpha_0 L_{\max} = \ell_{min}$ for $\boldsymbol{y} \in \Gamma$.
- The *maximum variation of area distortion* of the parametrization C_J. The variation of the area distortion of a patch $\boldsymbol{P}$ is $C_J(\boldsymbol{P}) = \max_{(s,t)} \sqrt{g(s,t)} / \min_{(s,t)} \sqrt{g(s,t)}$, where $g(s,t)$ is the determinant of the metric tensor of $\boldsymbol{P}$ at the point (s,t). We define $C_J = \max_{\boldsymbol{P} \in \Gamma} C_J(\boldsymbol{P})$. This value is an indicator of how non-uniform the parametrization of $\boldsymbol{P}$ is and allows us to estimate how the patch length decreases with refinement.

We assume that the α_0, β_0 and C_J are independent of N_{init}. We also assume that principal curvatures are bounded globally on Γ and independent of N_{init}.

3.1. Admissibility

The patch refinement procedure in [MRZ20, Section 3.2.1] to enforce [MRZ20, Section 3.2.1,Criteria 1 and 2] of admissibility and achieve given approximation errors of the geometry ϵ_{g} and boundary data ϵ_{f} is a local operation on each patch. If we assume that $L_{\min}$, $L_{\max}$, the partial derivatives of all patches composing $\hat{\Gamma}$, and the partial derivatives of f are bounded, then errors ϵ_{g} and ϵ_{f} can always be achieved after a fixed number of refinement steps. As a consequence, this stage must have complexity $O(N_{\text{init}})$.

We focus on the additional refinement needed to satisfy [MRZ20, Section 3.2.1,Criteria 3]: ensuring that each check center $\hat{\boldsymbol{c}}$ is closest to its corresponding quadrature point $\boldsymbol{y}$. This can be restated in terms of local feature size: for a quadrature patch $\boldsymbol{P} \in \hat{\Gamma}$ and quadrature node $\boldsymbol{x} \in \boldsymbol{P}$ with check center $\hat{\boldsymbol{c}}$, $\|\boldsymbol{x} - \hat{\boldsymbol{c}}\|_2 \leq \ell(\boldsymbol{x}) \leq \alpha_0 L_0$. We will first relate the number of required refinement steps η to satisfy [MRZ20, Section 3.2.1,Criteria 3] to the shape parameters α_0 and C_J, then we will show that this number does not depend on N under our assumptions.

Recall that the distance from a check center to the surface for a patch $\boldsymbol{P}$ is given by $R + r(p+1)/2 = (a + (p+1)b/2)L(\boldsymbol{P}) = KL(\boldsymbol{P})$. After η refinement steps, the area of each child of $\boldsymbol{P}$ relative to $\boldsymbol{P}$ itself will have decreased by at least by $C_J(\boldsymbol{P})(1/4)^\eta$. Since the distance from $\hat{\boldsymbol{c}}$ to the surface is proportional to $L(\boldsymbol{P})$, we can estimate the required level of uniform refinement to satisfy [MRZ20, Section 3.2.1,Criteria

3] by requiring that the check center distance is less than the minimal local feature size, then taking the maximum value of $L(\boldsymbol{P})$ over all patches:

$$KL_{\max}\sqrt{C_J}(1/2)^{\eta} \leq \ell_{min} = \alpha_0 L_{\max}$$

This yields

$$\eta = \lceil -\log_2 \frac{\alpha_0}{K\sqrt{C_J}} \rceil, \tag{9}$$

which we note depends only on nondimensional quantities α_0, K and C_J characterizing the shape of the surface and its parametrization. If we assume these to be independent of N, then the number of required levels of refinement η are also independent of N. This means that the number of patches N generated [MRZ20, Algorithm 2] is a linear function of N_{init}, bounded by $4^{\eta} N_{\text{init}}$.

Next, we estimate the complexity of work per patch in [MRZ20, Algorithm 2] to determine if a given patch requires refinement. As described in [MRZ20, Section 3.4], for each patch, we query the AABB tree T_B for patches that are at the distance $R + r(p+1)/2 = KL(\boldsymbol{P})$ from a check center $\hat{\boldsymbol{c}}$. The cost of the query is logarithmic in the number of patches N_{init} and proportional to the number of patches $N(\hat{\boldsymbol{c}})$ returned. This means that we need to estimate the number of patches that can be within the distance $KL(\boldsymbol{P})$ from $\hat{\boldsymbol{c}}$.

Consider an area element dA of $\hat{\Gamma}$ at a point $\boldsymbol{x}_0$. The parallel surface of dA, given by $\boldsymbol{x}_0 + h\boldsymbol{n}(\boldsymbol{x}_0)$ does not have self-intersections when $|h| \leq \ell_{min}$ and has a corresponding area element given by $dA^h = (1 + h\kappa_1)(1 + h\kappa_2)dA$ [Kre99, Section 6.2], where κ_1 and κ_2 are the principal curvatures of $\hat{\Gamma}$ at $\boldsymbol{x}_0$. The volume of the truncated cone bounded by dA and dA^h of height ℓ_{min} can be computed directly from the integral $\int_0^{\ell_{min}} dA^h dh$:

$$dV = dA\ell_{min}(1 + \frac{1}{2}(\kappa_1 + \kappa_2)\ell_{min} + \frac{1}{3}\kappa_1\kappa_2\ell_{min}^2) = dA\ell_{min}(1 + \frac{1}{2}H\ell_{min} + \frac{1}{3}K\ell_{min}^2)$$

where K and H are Gaussian and mean curvatures respectively. As principal curvatures satisfy $\kappa_i \geq -1/\ell_{min}$, this expression has minimal value for $\kappa_1 = \kappa_2 = -1/\ell_{min}$:

$$dV \geq \frac{1}{3}\ell_{min} dA \tag{10}$$

In other words, each surface element dA has (at least) a volume $\frac{1}{3}\ell_{min}dA$ with no other surface elements inside associated with it. From this, we can estimate the total area of surface contained within distance $KL(\boldsymbol{P})$ from $\hat{\boldsymbol{c}}$ by equating Eq. (10) with the volume of a sphere of raidus $KL(\boldsymbol{P})$, producing $4\pi K^3 L(\boldsymbol{P})^3/\ell_{min}$. Since the area of each patch is at least L_{min}^2, the number of patches $KL(\boldsymbol{P})$ from $\hat{\boldsymbol{c}}$ is bounded by

$$N(\hat{\boldsymbol{c}}) \leq 4\pi K^3 \frac{L(\boldsymbol{P})^3}{\ell_{min} L_{\text{min}}^2} \leq 4\pi K^3 \frac{L_{\text{max}}^3}{\ell_{min} L_{\text{min}}^2} = \frac{4\pi K^3}{\alpha_0 \beta_0^2} \tag{11}$$

This is independent of N_{init}, which means that the complexity of nearest patch retrieval is $O(N_{\text{init}} \log N_{\text{init}})$, with constant given by the product of (11) and 4^{η}, with η given by (9).

To complete the complexity estimate of the admissibility refinement, we need to estimate the cost of computing the closest point on each patch. The complexity of the Newton's method for finding roots of polynomials in Section 2 depends only on the polynomial degree and the desired accuracy of the optimization, which we can assume to be bounded by floating-point precision [SS17]. We conclude that the overall complexity of admissibility refinement is $O(N_{\text{init}} \log N_{\text{init}})$ with constants proportional to the patch degree and optimization accuracy.

3.2. Upsampling

We estimate the complexity of the upsampling algorithm in [MRZ20, Section 3.5] in terms of N, the number of patches produced by admissibility refinement, and a parameter ϵ, which is the desired accuracy achieved by the final upsampled patches at the check points. As the distance from the surface to the check points $\boldsymbol{c}_i$ is bounded from below by $aL_{\min}$, the $\tilde{V}$ term in [MRZ20, Heuristic 1] is bounded from above by $CL_{\min}^{-2q-1}$, for a constant C independent of q. Furthermore, since $\hat{\Gamma}$ and f are assumed to be smooth, the density and its derivatives can also be assumed to be bounded. The overall form of the estimate in [MRZ20, Heuristic 1] can then be bounded and written as $\tilde{C}(q)L_{\min}^{-2q-1}\tilde{L}^{2q}$ for some constant $\tilde{C}(q)$. The maximum patch length obtained by refinement $\tilde{L}$ is

$$\tilde{L} = L_{\max}^{\text{fine}} \leq L_{\max} 2^{-\tilde{\eta}}, \tag{12}$$

where $\tilde{\eta}$ is the maximum amount of required patch refinement. By setting $C(q)L_{\min}^{-2q-1}\tilde{L}^{2q} \leq \epsilon$ and using Eq. (12), we can obtain an upper bound for $\tilde{\eta}$ as a function of $L_{\min}$, $L_{\max}$, and ϵ:

$$\tilde{\eta} \leq -\frac{1}{2q}\log_2\left(\frac{\epsilon}{L_{\min}^{-2q-1}L_{\max}^{2q}C(q)}\right) = \log_2 \epsilon^{-1/(2q)} + \bar{C}(q, L_{\min}, L_{\max}), \tag{13}$$

for some constant $\bar{C}(q, L_{\min}, L_{\max})$.

The number of points generated by upsampling is $O(4^{\tilde{\eta}}N)$. Taking powers of both sides of Eq. (13) yields an estimate in terms of ϵ_{target}: $O((2^{\tilde{\eta}})^2 N) \leq O(\epsilon^{-2/(2q)}N) = O(\epsilon^{-1/q}N)$. As discussed in Section 3.1, the closest point computation needed to determine if a checkpoint is in Ω_I has $\log(N)$ cost per point, leading to $O(\epsilon^{-1/q}N\log(N))$ overall complexity and an upsampling factor of $\epsilon^{-1/q}$. Since we desire upsampled quadrature with an accuracy of 10^{-12}, we set ϵ as such to arrive at the desired complexity.

3.3. Point marking

In the point marking algorithm of [MRZ20, Section 3.6], we first use the Laplace FMM to cull points far from Γ, which requires $O(N + \mathcal{N}_{\text{tot}})$ time. Let $\bar{L} = \frac{1}{M}\sum_{\boldsymbol{P}\in vPcoarse} L(\boldsymbol{P})$ be the average patch length. After FMM culling, the remaining unmarked evaluation points are those whose distances from Γ are approximately $\bar{L}$ or less. For each unmarked point $\boldsymbol{x}$, we query the AABB tree T_T for the nearest triangle in the linear approximation of $\mathcal{P}_{\text{coarse}}$.

Since there are $O(N)$ such triangles in T_T, we can perform this query in $O(\log N)$ time [Sam06]. This triangle provides a candidate closest patch that is distance d_0 from $\boldsymbol{x}$. We then use to query T_B for all bounding boxes at distance d_0 from $\boldsymbol{x}$. This query too can be performed in $O(\log N)$ time [Sam06] and returns a bounded number of boxes and that each is processed in constant time, as discussed in Section 3.1. As the number of unmarked points after culling is bounded above by $\mathcal{N}_{\text{tot}}$, the overall complexity of our marking scheme is $O(\mathcal{N}_{\text{tot}}\log N)$.

3.4. Integral evaluation complexity

We assume that geometric admissibility criteria are already satisfied. All integral evaluation is accelerated using an FMM with complexity $O(N + \mathcal{N}_{\text{tot}})$.

Far zone. The complexity of far evaluation is just the complexity of computing the integrals on $\mathcal{P}_{\text{coarse}}$ using standard quadrature and FMM acceleration, i.e., $O(q^2N + \mathcal{N}_{\text{far}})$.

Intermediate zone. The complexity of the intermediate zone evaluation is similar to that of the far zone. However the computation is performed on $\mathcal{P}_{\text{fine}}$ rather than $\mathcal{P}_{\text{coarse}}$, which is up to m times finer than $\mathcal{P}_{\text{coarse}}$, with $m = O(\epsilon^{-1/q})$ and $\epsilon = 10^{-12}$. The density values must be interpolated from points in $\mathcal{P}_{\text{coarse}}$ to points in $\mathcal{P}_{\text{fine}}$: this can be computed in $O(mq^4N)$ time using a 2D version of the barycentric interpolation formula [BT04]. This yields an overall complexity of $O(mq^4N + mq^2N + \mathcal{N}_{\text{inter}})$. Although not asymptotically dominant, for all practical target errors, the quadrature evaluation is the dominant cost in practice due to suppressed FMM-related constants, as demonstrated in Section 4.

Near zone. [MRZ20, Section 3.1] requires a closest point computation, an intermediate-zone evaluation at p check points and an extrapolation for each target point in Ω_N. The intermediate zone calculation is the dominant cost, resulting in a complexity of $O(mq^4N + mq^2N + p\mathcal{N}_{\text{near}})$.

GMRES solve. As a result of the second-kind integral formulation in [MRZ20, Section 2] the cost of solving [MRZ20, Equation 5] via GMRES is asymptotically equal to the cost of a single singular integral evaluation, since the low number of iterations are independent of N. In our algorithm, this is a special case of near-zone evaluation with $\mathcal{N}_{\text{near}} = q^2N$, producing a complexity of $O(mq^4N + mq^2N + pq^2N) = O((m + p + mq^2)q^2N)$.

Overall complexity for uniform point distribution. We now suppose that we wish to evaluate the solution u determined by a density ϕ at a set of uniformly distributed points throughout Ω. We also assume that $\hat{\Gamma}$ is discretized uniformly by N patches, i.e., $L_{\max} = O(N^{-1/2})$ and that the distances between samples in Ω and from samples to $\hat{\Gamma}$ are also $O(N^{-1/2})$. Since the total number of evaluation points is proportional to $1/L_{\max}^3$, this implies that $\mathcal{N}_{\text{tot}} = O(N^{3/2})$.

The size of the intermediate zone Ω_I is bounded by the estimate discussed in Section 3.2. Letting d_I be the shortest distance along a normal vector of $\hat{\Gamma}$ which is contained in Ω_I, following the discussion in Section 3.2 yields the following relation:

$$\tilde{C}(n) d_I^{-2q-1} L_{\max}^{2q} \leq \epsilon. \tag{14}$$

Solving for d_I gives us

$$d_I \leq \left(\frac{\epsilon}{C(n)}\right)^{-\frac{1}{2q-1}} (L_{\max})^{\frac{2q}{2q-1}}. \tag{15}$$

We are interested in the regime as $N \to \infty$, or $L_{\max} \to 0$. Since $L_{\max}^{\frac{2q}{2q-1}} \leq \sqrt{L_{\max}} = O(N^{-1/4})$, this gives us

$$d_I \leq \left(\frac{\epsilon}{C(n)}\right)^{-\frac{1}{2q-1}} N^{-1/4} = O(\epsilon^{-1/2q} N^{-1/4}) = O(\sqrt{m} N^{-1/4}), \tag{16}$$

after recalling from above that $m = O(\epsilon^{-1/q})$ is the average upsampling rate to produce $\mathcal{P}_{\text{fine}}$ from $\mathcal{P}_{\text{coarse}}$. The size of the near zone is, by construction, of the order $L_{\max}$. It follows that $\mathcal{N}_{\text{inter}} = O(\sqrt{m} N^{5/4})$, and $\mathcal{N}_{\text{near}} = O(N)$.

The overall complexity for this evaluation is the sum of the cost of each separate evaluation:

$$\begin{aligned} &O(q^2N + \mathcal{N}_{\text{far}} + mq^4N + mq^2N + \mathcal{N}_{\text{inter}} + mq^4N + mq^2N + p\mathcal{N}_{\text{near}}) \\ &= O\left((m + mq^2)q^2N + \mathcal{N}_{\text{tot}} + (p-1)\mathcal{N}_{\text{near}}\right) \end{aligned}$$

Using the estimates for $\mathcal{N}_{\text{tot}}$ and $\mathcal{N}_{\text{near}}$ and dropping dominated terms, we obtain $O((m + mq^2)q^2N + N^{3/2})$ for the overall complexity. This suggests that for a given q and ϵ, the minimal cost is obtained from choosing the number of discretization points $N = O(m^2)$, i.e., $N = O(\epsilon^{-2/q})$.

4. Comparison with [YBZ06]

To understand the performance of [YBZ06] and `hedgehog` and see the implications of this complexity difference in practice, we now compare the performance of `hedgehog` with that of [YBZ06] on several concrete numerical examples. The metric we are interested is *cost for a given relative error*. Assuming the surface discretization is $O(N)$, we measure the *cost* of a method as its total wall time during execution T divided by the total wall time of an FMM evaluation on the same $O(N)$ discretization, T_{FMM}. By normalizing by the FMM evaluation cost, we minimize the dependence of the cost on machine- and implementation-dependent machine-dependent parameters, such as clock speed, cache size, performance optimizations, etc. We run the tests in this section on the sphere geometry shown in [MRZ20, Figure 8-left] and continue to focus on the singular quadrature scheme of [YBZ06] as described in [MRZ20, Section 6.2].

4.1. Complexity comparison

The algorithm of [YBZ06] substantially differs from `hedgehog` in two main ways. First, on-surface singular integral evaluation is computed in [YBZ06] by subtracting the inaccurate part of the FMM-accelerated smooth quadrature rule using a partition-of-unity (POU) function, then adding an accurately computed part singular integral close to singularity via polar quadrature. Second, [YBZ06] sets more algorithms parameters *a priori* rather than determining them adaptively. Specific choices used in [YBZ06] may be considered optimal for the uniform volume point distribution described in Section 3.4, but need to be adjusted based on additional analysis for other distribution types. Additionally, [YBZ06] has a trade-off between accuracy and complexity proportional to the POU radius d_P, which `hedgehog` does not have.

The intermediate and far zone complexity estimates are similar for both `hedgehog` and [YBZ06]. The near-zone complexity for the algorithm of [YBZ06] has an additional term of the form $O(Nd_P^2/L_{\max}^2)$, where d_P is the radius of the POU function. For simplicity, we use $L_{\max}$ as a measure of surface sampling density as in Sections 3.1 and 3.2, since $L_{\max}$ and the h from [YBZ06] differ by a constant.

The error of [YBZ06]'s singular evaluation is $O(d_P^{-2q-1}L_{\max}^{2q})$, for an optimally chosen local quadrature rule. We note that the factor d_P^{-2q-1} is entirely an artifact of using a compactly supported POU function to localize the singular integral computation. As observed in [YBZ06], to achieve optimal convergence as the surface is refined, d_P needs to decrease slower than $L_{\max}$, i.e., slower than $N^{-1/2}$, under the assumptions on point distribution in Ω from Section 3.4. In [YBZ06], $d_P = O(N^{-1/2(1+\gamma)})$ is suggested. As a result, the overall complexity is $O(N^{1+\gamma})$ and grows faster than N.

By choosing $\gamma = \frac{1}{2}$, [YBZ06]'s final complexity becomes $O(N^{3/2})$ in order to produce an error proportional to $N^{(-2q+1)/4}$. In other words, the work needed for an error ϵ is proportional to $\epsilon^{-6/(2q-1)}$, which is asymptotically higher than `hedgehog` (with ϵ from Section 3.2). On the other hand, our method has the disadvantage of requiring p check point evaluations for every sample point in $\mathcal{N}_{\text{near}}$. This requires an FMM call that is $(m+p)$-times larger than [YBZ06]. In common use cases, such as solving [MRZ20, Equation 5] via GMRES, repeated `hedgehog` evaluations through a more expensive FMM can require more work in practice for lower accuracy than [YBZ06].

4.2. Experimental comparison.

To understand the performance of these two methods and see the implications of this complexity difference in practice, we now compare the performance of `hedgehog` with that of [YBZ06] on several concrete numerical examples. The metric we are interested is *cost for a given relative error*. Assuming the surface discretization is $O(N)$, we measure the *cost* of a method as its total wall time during execution T divided by the total wall time of an FMM evaluation on the same $O(N)$ discretization, T_{FMM}. By normalizing by the FMM evaluation cost, we minimize the dependence of the cost on machine- and implementation-dependent machine-dependent parameters, such as clock speed, cache size, performance optimizations, etc.

Comparison on C^∞ surface of [YZ04]. An important contribution of [YBZ06] was the use of a C^∞ surface representation, first introduced in [YZ04], allowing for exponential accuracy via the trapezoidal rule, and easy resampling for singular quadrature. To fairly compare the two quadrature methods, we have implemented a modified version of `hedgehog` on the surface representation of [YZ04]. The algorithm of [MRZ20, Section 3.1] has the following modifications: (i) we discretize the vertex-centered patches of [YZ04] with the tensor-product trapezoidal rule for compactly supported functions with spacing parameter h, as in [YBZ06]; (ii) the upsampled quadrature rule uses a trapezoidal rule with spacing $h/4$; (iii) density interpolation is computed with FFT's, as in [YBZ06]; the rest of the algorithm proceeds unchanged. This essentially matches [MRZ20, Section 3.1] but uses the discretization scheme of [YBZ06] instead of Clenshaw-Curtis.

For each of the tests in this section, we choose some initial spacing parameter h_0 to discretize the surface of [YZ04] as in [YBZ06] and use the same $16\times$ upsampled grid to evaluate both `hedgehog` and [YBZ06]. We apply the modified `hedgehog` algorithm and the scheme of [YBZ06] with spacing h_0 and compute the relative error and collect timing statistics. We repeat this test with $h_0/2^i$ for $i = 1, \ldots 4$ and plot the results. This ensures that the smooth quadrature rule used by both methods have the same resolution.

We choose the floating partition of unity size in [YBZ06] to be $\sqrt{h}$ as in the original work. As in the previous section, we choose the parameters r and R of `hedgehog` to be $O(\sqrt{h})$ to observe standard convergence behavior. For both quadrature methods, we use a multipole order of 16 for PVFMM with at most 250 points in each leaf box and with the same initial spacing.

In Figs. 1 and 2, we summarize our results for two test cases. In Fig. 1, we evaluate [MRZ20, Equation 8] using one-sided `hedgehog` and the singular quadrature method of [YBZ06] with the density $\phi = 1$, in order to demonstrate their behavior without interaction with GMRES. In Fig. 2, we construct a boundary condition using [MRZ20, Equation 25] with random charge values and solve [MRZ20, Equation 5] using two-sided `hedgehog` and with the singular quadrature method of [YBZ06] inside of GMRES. We then evaluate the singular integral at a finer discretization of the surface using either one-sided `hedgehog` or [YBZ06], respectively. From left to right, each plot details the total cost of each scheme, the cost of each subroutine for `hedgehog` (denoted HH) and the singular quadrature scheme of [YBZ06] (denoted POU), and the relative error as a function of h. Each data point in the plots, from right to left, is the result of running the method on a discretization with spacing $h_0/2^i$ for $i = 0, \ldots, 4$. We plot the cost of both schemes the cost of each algorithmic step as a function of their computed relative error. In each figure, we present results for a Laplace problem (top) and an elasticity problem (bottom), to highlight the difference in performance between scalar and vector kernels.

As expected, the `hedgehog` total cost curves lie somewhere between 1 and 10, since the required FMM evaluation is $(m + p)$-times larger than N. This step is the dominant cost: the next most expensive step is density interpolation, which is two orders of magnitude faster. Initially, the main cost of [YBZ06] is FMM evaluation time, but eventually the local correction cost begins to dominant. Note that the `hedgehog` and [YBZ06]-FMM curves are not quite flat, due to the initial quadratic complexity of a shallow FMM tree.

From Figs. 1 and 2, we observe a higher convergence rate for `hedgehog` than [YBZ06], except for the elasticity solve in Fig. 2-bottom where the methods perform about equally. This allows the cost of `hedgehog` to decrease below [YBZ06] for errors less than 10^{-7} for Laplace problems. More importantly, however, [YBZ06] outperforms `hedgehog` for elasticity problems for all tested discretizations, and also for low and moderate accuracy Laplace problems. This is due to the greater cost of a vector FMM evaluation compared to a scalar one: the $m + p$ factor saved in the FMM evaluation of [YBZ06] can be accelerated more efficiently with the method's small dense linear algebra computations. This means that a local singular quadrature method of *worse* complexity can beat a global method, simply by virtue of reducing the FMM size. Moreover, our implementation of [YBZ06] is not highly optimized, so we can expect a well-engineered POU singular quadrature implementation such as [MCIGO19] to widen this gap. By noting the large difference between the `hedgehog` FMM cost and the `hedgehog` density interpolation, we

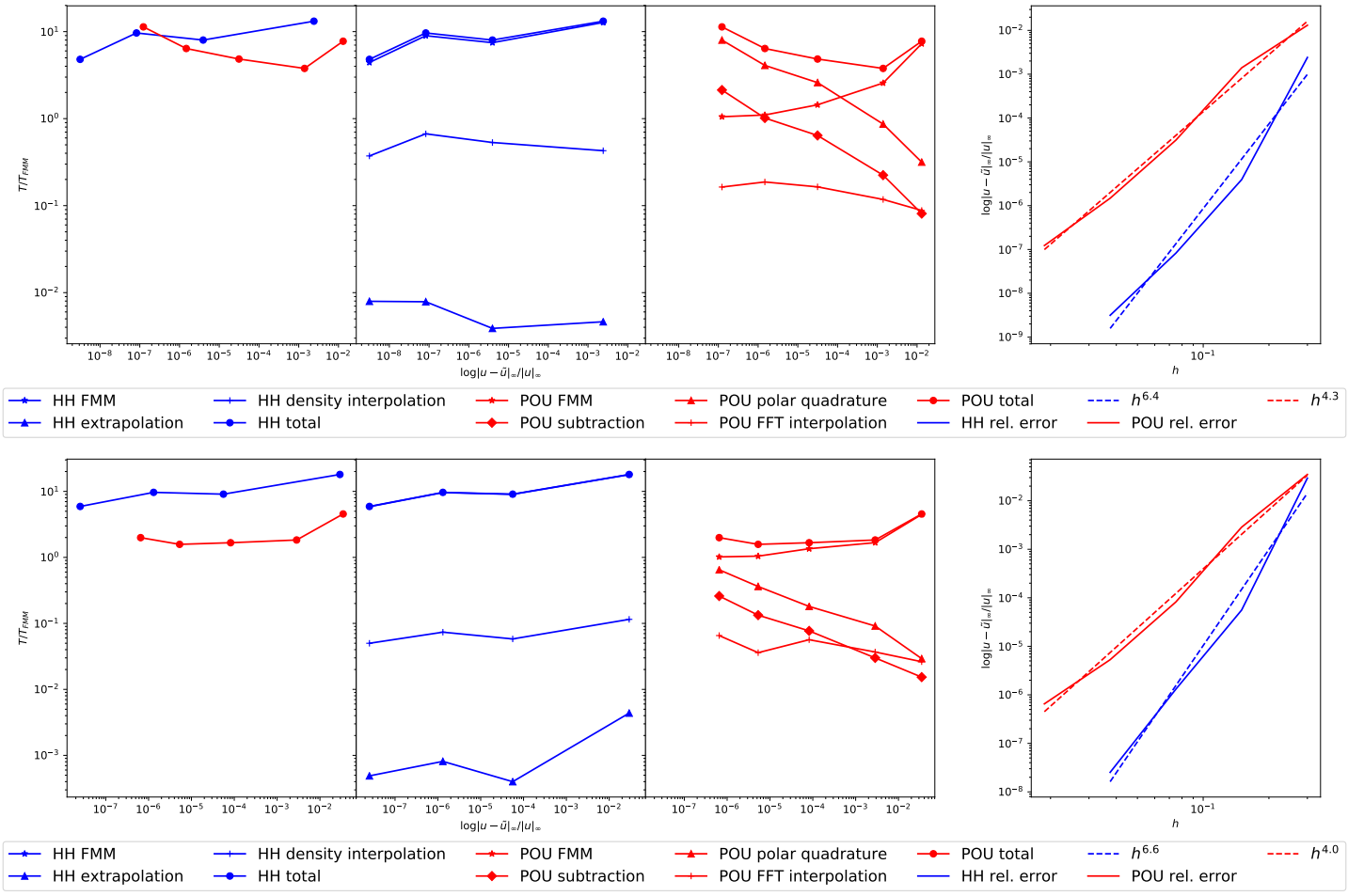

Figure 1: Comparison of `hedgehog` (HH) versus [YBZ06] (POU) on the surface representation of [YZ04] evaluating double-layer potential with $\phi = 1$. *Laplace (top) and elasticity (bottom) problems solved on the sphere shown in [MRZ20, Figure 8-left]. From left to right, we plot the total cost of each scheme, the cost of each subroutine for* `hedgehog` *(blue) and the singular quadrature scheme of [YBZ06] (red), and the relative error as a function of h. The plots show the cost and relative error for $h_0 = .3$ representing the right-most data point and each point to the left corresponding to a spacing of $h_i = h_0/2^i$. For the Laplace problem, we choose $r = .186\sqrt{h}$, $R = 1.12\sqrt{h}$ and $p = 6$ for* `hedgehog` *parameters; for the elasticity problem, we choose $r = .133\sqrt{h}$, $R = .8\sqrt{h}$ and $p = 6$. The initial spacing parameter is $h_0 = .3$.*

can reasonably infer that a local `hedgehog` scheme should narrow this gap and outperform [YBZ06], assuming that this transition does not dramatically affect error convergence.

5. Comparison with [WK19a, WK19b]

Our work most closely resembles the advancements presented in [WK19a, WK19b]. We have presented a *global* singular/near-singular quadrature method, i.e., the potential values at the check points are computed with a quadrature rule from the entire boundary. [WK19a] proposed a global QBX method that computes QBX expansion coefficients via FMM translation operators from within an FMM tree. Our method is *target-specific* as in [ST18], creating one set of check points for each target point. [WK19a] was further refined to include target-specific QBX expansions in [WK19b].

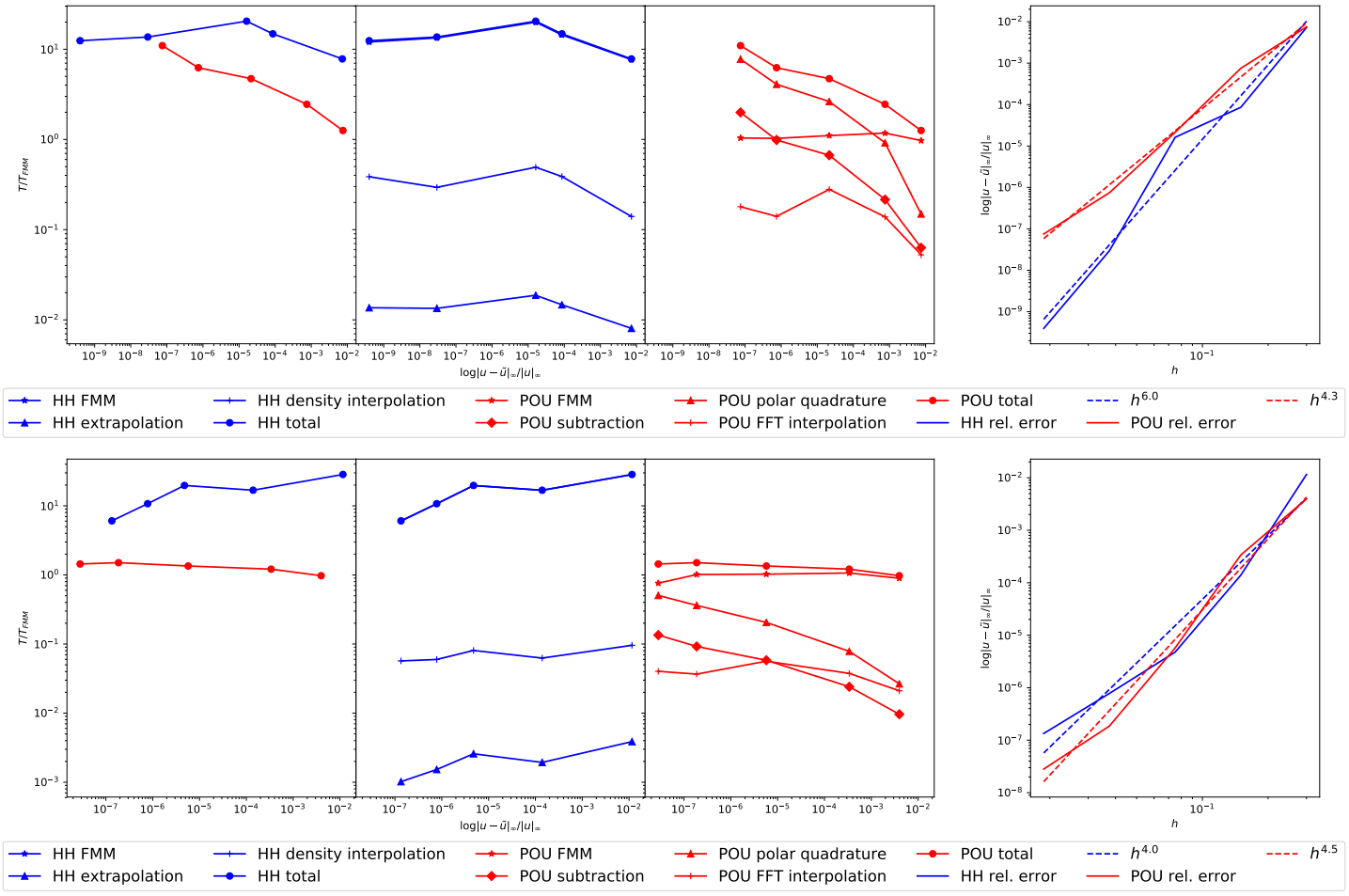

Figure 2: Comparison of `hedgehog` versus [YBZ06] on the surface representation of [YZ04] solving via GMRES for u_c. *This figure's format is similar to Fig. 1. For the Laplace problem, we choose* $r = .028\sqrt{h}$, $R = .172\sqrt{h}$ *and* $p = 6$ *for* `hedgehog` *parameters; for the elasticity problem, we choose* $r = .042\sqrt{h}$, $R = .253\sqrt{h}$ *and* $p = 6$. *The initial spacing parameter is* $h_0 = .3$.

Our admissibility algorithm is similar to the Stage-1 refinement of [WK19a]. Both approaches first resolve the boundary data and input geometry, then enforce a criteria that will guarantee accurate smooth quadrature rules at prescribed point locations. The improvement in our approach is the decoupling of the spatial data structure for the required geometry queries to enforce admissibility and the data structure for FMM acceleration. This allows for less memory overhead and faster spatial queries and FMM evaluations by leveraging existing software packages. Additionally, our algorithm is formulated in terms of patches and bounding boxes rather than in terms of quadrature point locations. This allows us to perform fewer spatial queries on a smaller data structure to enforce our criteria and make guarantees about the proximity of a patch to a check point that is independent of the quadrature order. As in [WK19a], we also fix the check point location before upsampling, which decouples the coarse and upsampled discretization. We both compute upsampled discretizations based on empirical heuristics to approximate quadrature error behavior.

However, the primary improvement of `hedgehog` over [WK19a] is *algorithmic simplicity*. Our only requirement is a standard point FMM without modifications. This allows us to utilize existing optimized

algorithms for spatial queries and fast summation, which have been extensively optimized. Most importantly, it prevents the QBX-FMM error coupling handled carefully in [WK19a, WK19b]. The price we must pay for this simplicity is a larger point FMM evaluation, since we are using the discretization of $\mathcal{P}_{\text{fine}}$ as source points. Since we are using the kernel-independent FMM, we must use a higher multipole order to counteract the accumulation of translation operator error inherent in this approach [YBZ04]. A standard FMM method would not have this downside, but we believe that PVFMM's impressive performance optimizations make this is reasonable trade-off.

6. Geometry approximation error

Let θ be a scalar function defined on the surface of $\partial\Omega$ with $|\theta| \leq 1$ and let δ be a small real constant. Suppose the boundary of the domain Ω is perturbed by δ along the normal field of $\partial\Omega$, scaled by θ, to produce the perturbed domain Ω_δ with boundary $\partial\Omega_\delta$. More concretely, for $\boldsymbol{y} \in \partial\Omega$ and $\boldsymbol{y}_\delta \in \partial\Omega_\delta$, $\boldsymbol{y}_\delta = \boldsymbol{y} + \delta\theta\boldsymbol{n}(\boldsymbol{y})$. We can define the *Eulerian shape derivative* of u with respect to θ, denoted u_θ, at a point $\boldsymbol{x} \in \Omega_\delta \cap \Omega$ as the rate of change in u at $\boldsymbol{x}$ as $\delta \to 0$. This quantity is of interest to us because the solution to [MRZ20, Equation 2] on $\Omega_\delta \cap \Omega$ can be written as $u + \delta u_\theta$, where u is the solution to [MRZ20, Equation 2] on Ω. Moreover, we can compute the shape derivative by solving a Laplace problem on the unperturbed domain [Pir82]:

$$\Delta u_\theta = 0 \text{ in } \Omega,\ u_\theta = -\theta\frac{\partial u}{\partial n} \text{on } \partial\Omega. \tag{17}$$

where u is the solution of the [MRZ20, Equation 2] on Ω. For small δ, this means that the error in the solution introduced by a boundary perturbation along the field θ can be estimated by $\delta \sup_\Omega \|u_\theta\|$. Assuming the boundary is smooth and the gradient of the solution u is bounded, then

$$\|u_\theta\| \leq C_g \sup_{\partial\Omega} \left|\theta\frac{\partial u}{\partial n}\right| \leq C_g \sup_{\partial\Omega} \left|\frac{\partial u}{\partial n}\right| \tag{18}$$

for some real constant C_g. The right-hand side of Eq. (18) yields a constant C_g', such that if $\epsilon_{\text{g}} < \zeta\epsilon_{\text{target}}/C_g'$ for some $\zeta < 1$, the change in the solution is less than ϵ_{target} for a sufficiently small ϵ_{g}. The constant depends implicitly on the surface geometry: for example, if an area element of $\partial\Omega$ is close to a sharp, concave corner, then $\frac{\partial u}{\partial n}$ can be arbitrarily large.

[BT04] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *Siam Review*, 46(3):501–517, 2004.

[Kre99] Rainer Kress. Linear integral equations, volume 82 of applied mathematical sciences, 1999.

[MCIGO19] Dhairya Malhotra, Antoine Cerfon, Lise-Marie Imbert-Gérard, and Michael O'Neil. Taylor states in stellarators: A fast high-order boundary integral solver. *arXiv preprint arXiv:1902.01205*, 2019.

[MRZ20] Matthew J Morse, Abtin Rahimian, and Denis Zorin. A robust solver for elliptic pdes in 3d complex geometries. *arXiv preprint arXiv:2002.04143*, 2020.

[Pir82] Olivier Pironneau. Optimal shape design for elliptic systems. In *System Modeling and Optimization*, pages 42–66. Springer, 1982.

[Sam06] Hanan Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[SS17] Dierk Schleicher and Robin Stoll. Newton's method in practice: Finding all roots of polynomials of degree one million efficiently. *Theoretical Computer Science*, 681:146–166, 2017.

[ST18] Michael Siegel and Anna-Karin Tornberg. A local target specific quadrature by expansion method for evaluation of layer potentials in 3D. *Journal of Computational Physics*, 364:365–392, 2018.

[WK19a] Matt Wala and Andreas Klöckner. A fast algorithm for Quadrature by Expansion in three dimensions. *Journal of Computational Physics*, 388:655–689, 2019.

[WK19b] Matt Wala and Andreas Klöckner. Optimization of fast algorithms for global Quadrature by Expansion using target-specific expansions. *Journal of Computational Physics*, page 108976, 2019.

[YBZ04] Lexing Ying, George Biros, and Denis Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, 2004.

[YBZ06] Lexing Ying, George Biros, and Denis Zorin. A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains. *Journal of Computational Physics*, 219(1):247–275, 2006.

[YZ04] Lexing Ying and Denis Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 271–275. ACM, 2004.