

RADIAL BASIS FUNCTION TECHNIQUES FOR NEURAL FIELD MODELS ON SURFACES*

SAGE B. SHAW[†], ZACHARY P. KILPATRICK[†], AND DANIELE AVITABILE[‡]

Abstract. We present a numerical framework for solving neural field equations on surfaces using radial basis function (RBF) interpolation and quadrature. Neural field models describe the evolution of macroscopic brain activity, but modeling studies often overlook the complex geometry of curved cortical domains. Traditional numerical methods, such as finite element or spectral methods, can be computationally expensive and challenging to implement on irregular domains. In contrast, RBF-based methods provide a flexible alternative by offering interpolation and quadrature schemes that efficiently handle arbitrary geometries with high-order accuracy. We first develop an RBF-based interpolatory projection framework for neural field models on general smooth surfaces. Quadratures for both flat and curved domains are derived in detail, ensuring high-order accuracy and stability as they depend on RBF hyper-parameters (basis functions, augmenting polynomials, and stencil size). Through numerical experiments, we demonstrate the convergence of our method, highlighting its advantages over traditional approaches in terms of flexibility and accuracy. We conclude with an exposition of numerical simulations of spatiotemporal activity on complex surfaces, illustrating the method's ability to capture complex wave propagation patterns.

Key words. radial basis functions, nonlocal equations, mathematical neuroscience

MSC codes. 37N30, 47G20, 65J05

DOI. 10.1137/25M1753784



See reproducibility of
computational results
at end of the article.

1. Introduction. Neural field equations are nonlinear integro-differential equations that model large-scale neuronal activity, offering tractable formulations for analysis and simulation across diverse domains [13]. Such activity is strongly determined by the architecture of synaptic connections between neurons [20]. Associated models typically grapple with the high volume of neurons and synapses, especially in the primate cortex, with approximations such as coarsening, homogenization, and timescale separations [11]. Neural fields treat the cortex as a continuum excitable medium and model connectivity via a spatial weight kernel in a nonlinear integral operator, allowing the application of theory from nonlinear waves and other methods from partial differential equations (PDEs) [13]. The kernel defines a spatially dependent synaptic coupling strength based on the locations of pre- and post-synaptic neurons. Analyses and simulations typically center on this nonlocal term, which drives local dynamics throughout the domain.

*Submitted to the journal's Numerical Algorithms for Scientific Computing section April 22, 2025; accepted for publication (in revised form) September 17, 2025; published electronically April 3, 2026.
<https://doi.org/10.1137/25M1753784>

Funding: This work was initiated through discussions at the Institute for Computational and Experimental Research in Mathematics (ICERM), which is supported by the National Science Foundation under grant DMS-1929284. The first and second authors were supported by the National Science Foundation under grant DMS-2207700 and by the National Institutes of Health BRAIN Initiative under grant R01EB029847.

[†]Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO 80309 USA (Sage.Shaw@colorado.edu, zpkillpat@colorado.edu).

[‡]Department of Mathematics, Amsterdam Center for Dynamics and Computation, Vrije Universiteit Amsterdam, Amsterdam, 1081 HV, The Netherlands, and Inria MathNeuro team, Montpellier, France (d.avitabile@vu.nl).

Standard derivations of neural fields are grounded in the known spatial and functional structure of synaptic connectivity throughout the cerebral cortex, justifying the typical spatial coarse graining required to obtain continuum models [58, 65]. For analytical and numerical convenience, most modeling studies consider canonical spatial domains and connectivity functions (e.g., distance-dependent weight kernels on planar domains) [21]. Amari (1977) [2] introduced a simplified Heaviside transfer function to enable the use of interface methods and explicit construction of wave solutions. Weight kernels with rational Fourier transforms reduce nonlocal neural fields to local PDEs, facilitating efficient simulation and analysis [39]. However, these simplifications often ignore geometrical complexity. Here we advance a method that is highly flexible to the formulation of neural fields on arbitrary domains with curvature and nonstandard weight kernels.

Although the cortex is three-dimensional, many imaging modalities (e.g., fMRI, VSD) access activity on its two-dimensional surface [33, 40]. Prior models considering curved domains appeal to idealizations like spheres [15, 62] and tori [37], allowing for spectral solution methods leveraging known orthonormal eigenbases. General curved surfaces that cannot be expressed as Cartesian products lack this structure. Recent work has explored neural fields on arbitrary smooth surfaces [44], but without convergence guarantees and with only first-order accuracy. A recent framework for projection-based neural field solvers [5] offers an operator-theoretic approach to convergence analysis, decomposing numerical error into contributions from spatial projection, time integration, and quadrature. This unified perspective applies to both Galerkin and collocation methods, clarifying how errors scale with resolution and guiding the design of efficient discretizations for simulations on complex geometries.

In this work, we develop a high-order numerical method for simulating neural field equations on complex geometries and advance the associated convergence theory within a projection-based framework. Our approach combines radial basis function quadrature (RBF-QF) [51, 52, 53] with RBF interpolation to achieve high-order accuracy using relatively few degrees of freedom. This is especially important for simulations on curved surfaces, where evaluating nonlocal integral operators at $\mathcal{O}(n^2)$ cost can be prohibitive.¹ We formulate projection and collocation schemes for neural fields and present convergence results for projection methods (section 2); we describe the RBF-QF algorithm and its extension to manifolds (section 3); and we verify performance through numerical tests and simulations on nontrivial geometries (4 and 5).

2. Collocation schemes for the neural field model. We consider, as a model problem, a neural field posed on a compact domain $(t, \mathbf{x}) \in [0, T] \times (D \subset \mathbb{R}^{\dim})$:

$$(2.1) \quad \begin{aligned} \partial_t u(t, \mathbf{x}) &= -u(t, \mathbf{x}) + g(t, \mathbf{x}) + \int_D w(\mathbf{x}, \mathbf{y}) f(u(\mathbf{y}, t)) d\mu(\mathbf{y}), \\ u(0, \mathbf{x}) &= v(\mathbf{x}). \end{aligned}$$

We assume the integral is on a volume or surface D , and so is expressed in terms of a measure μ . Heuristically, a collocation scheme for (2.1) can be conceived in two steps, as follows:

Step 1: collocation. Select n points $\Xi = \{\mathbf{x}_i : i \in \mathbb{N}_n := \{1, \dots, n\}\} \subset D$ in the domain, where n may depend on a discretization parameter h so $n(h) \rightarrow \infty$ as $h \rightarrow 0$. We impose that (2.1) holds at each node, that is, *we collocate the*

¹On certain domains, FFT-based acceleration reduces complexity to $\mathcal{O}(n \log n)$, but such approaches are not applicable to general manifolds.

equation on Ξ . This leads to a functional equation relating $\{\partial_t u(t, \mathbf{x}_i)\}_i$ and $\{u(t, \mathbf{x}_i)\}_i$ to the function $u(t, \cdot)$, appearing under the integral of (2.1).

Step 2: quadrature. Select a quadrature for the integral in terms of $\{u(t, \mathbf{x}_i)\}_i$ and obtain a system of coupled, nonlinear ordinary differential equations (ODEs).

Avitabile (2023) [5] builds on previous work by Atkinson (2005) [4] to show that convergence estimates for collocation (and other) schemes are still possible when disregarding Step 2. Step 1 is a fully functional scheme for which one can estimate convergence rates of the collocation (or other projection types) as $n \rightarrow \infty$. Step 2 further approximates the projection with quadrature, establishing a new scheme (a *discrete collocation scheme* in Atkinson's terminology [4]).

Henceforth we assume the following hypotheses, providing a natural functional setup for the neural field problem posed on $\mathbb{X} = C(D)$ [5, 50]

Hypothesis 2.1 (general hypotheses).

1. The cortical domain $D \subset \mathbb{R}^{\dim}$ is compact.
2. The temporal domain $J \equiv [0, T] \subset \mathbb{R}$ is compact.
3. The synaptic kernel $w: D \times D \rightarrow \mathbb{R}$ is a function in $C(D \times D)$.
4. The firing rate $f: \mathbb{R} \rightarrow \mathbb{R}$ is a bounded and everywhere differentiable Lipschitz function, which guarantees $f, f' \in B(\mathbb{R})$.
5. The initial condition $v: D \rightarrow \mathbb{R}$ is a function in $\mathbb{X} = C(D)$.
6. The forcing $g: D \times J \rightarrow \mathbb{R}$ is a function in $C(J, \mathbb{X})$.

To set the stage for our collocation scheme, we rewrite the neural field (2.1) in operator form,

$$(2.2) \quad \begin{aligned} u'(t) &= -u(t) + WF(u(t)) + g(t) =: N(t, u(t)), & t \in J \equiv [0, T], \\ u(0) &= v, \end{aligned}$$

where we define the integral operator W and nonlinear operator F :

$$(2.3) \quad W: \mathbb{X} \rightarrow \mathbb{X}, \quad v \mapsto \int_D w(\cdot, \mathbf{y})v(\mathbf{y})d\mu(\mathbf{y}), \quad F: \mathbb{X} \rightarrow \mathbb{X}, \quad v \mapsto f(v).$$

Note that (2.1) evolves the $\mathbb{R}$ -valued function $u: D \times J \rightarrow \mathbb{R}$, whereas (2.2) evolves the $\mathbb{X}$ -valued function $U: J \rightarrow \mathbb{X}$, $t \mapsto u(\cdot, t)$. With a small abuse of notation, we adopt the same symbol for both functions (u), as for the forcing function g .

Equation (2.2) describes a Cauchy problem on the Banach space $\mathbb{X}$, which is well-posed under Hypothesis 2.1. That is, there exists a unique $u \in C^1(J, \mathbb{X})$ satisfying (2.2), as shown in [5, Lemma 2.7]. (See [50] for analogous results on unbounded cortices.)

2.1. Step 1: Collocation via RBF interpolatory projection. RBFs provide a flexible and accurate way to interpolate functions and have been successfully applied to nonlinear PDEs with complex spatiotemporal dynamics, like reaction-diffusion systems [55]. Since our goal is to approximate nonlinear integro-differential (neural field) equations, RBF-based collocation schemes are a natural choice. We present these schemes using the abstract notion of an *interpolating projector*, which maps functions onto their interpolants (e.g., RBFs).

In this setup, one seeks a solution u to the neural field equations as a mapping on $[0, T]$ to $\mathbb{X} = C(D)$, the space of continuous functions on D . An interpolatory projector P_n linearly projects any function $v \in \mathbb{X}$ to a function $v_n = P_nv$ in an n -dimensional subspace $\mathbb{X}_n = \text{Span}\{L_1, \dots, L_n\} \subset \mathbb{X}$, with the property

$$(2.4) \quad P_n: \mathbb{X} \rightarrow \mathbb{X}_n, \quad P_nv = v, \quad \text{on } \Xi.$$

The approximant then takes the form

$$v_n(\mathbf{x}) := (P_n v)(\mathbf{x}) = \sum_{j=1}^n v(\mathbf{x}_j) L_j(\mathbf{x}), \quad \mathbf{x} \in D.$$

The collocation scheme approximates the solution u of (2.2), by a function $u_n \in C^1(J, \mathbb{X}_n)$ that satisfies the following Cauchy problem on the n -dimensional subspace $\mathbb{X}_n \subset \mathbb{X}$:

$$(2.5) \quad \begin{aligned} u'_n(t) &= -u_n(t) + P_n W F(u_n(t)) + P_n g(t) = P_n N(t, u_n(t)), & t \in J, \\ u_n(0) &= P_n v, \end{aligned}$$

that is, the Cauchy problem obtained from (2.2) upon applying the projection P_n to the vectorfield N , and to the initial condition v .

In view of [5, Proposition 4.1], this is equivalent to imposing that (2.1) holds at all $\mathbf{x}_i \in \Xi$ (hence we have collocated the equation, and completed Step 1),

$$\begin{aligned} u'_n(t)(\mathbf{x}_i) &= [P_n N(t, u_n(t))](\mathbf{x}_i), & (i, t) \in \mathbb{N}_n \times J, \\ u_n(0)(\mathbf{x}_i) &= (P_n v)(\mathbf{x}_i). \end{aligned}$$

In [5, Theorem 3.1] it is shown that, under Hypothesis 2.1, also the projected problem is well-posed, admitting a unique solution $u_n \in C^1(J, \mathbb{X}_n)$.

The projected problem (2.5) evolves the function $u_n(t)$ in $\mathbb{X}_n$. This form is useful in the analysis of the scheme, but for implementations we pursue an ODE approximating the n coefficients $\alpha_i(t) := u_n(t)(\mathbf{x}_i)$. Since $u_n(t) \in \mathbb{X}_n$, we set $u_n(t) = \sum_{j \in \mathbb{N}_n} \alpha_j(t) L_j$, exploit the Lagrange property of the basis $\{L_j\}$, and obtain an ODE in $\mathbb{R}^n$:

$$(2.6) \quad \begin{aligned} \alpha'(t) &= -\alpha(t) + K(\alpha(t)) + \gamma(t), & t \in J, \\ \alpha(0) &= \nu, \end{aligned}$$

where

$$\begin{aligned} \alpha_i(t) &= u_n(t)(\mathbf{x}_i), & \nu_i &= v(\mathbf{x}_i), \\ \gamma_i(t) &= g(\mathbf{x}_i, t), & K_i(\alpha) &= \int_D w(\mathbf{x}_i, \mathbf{y}) f\left(\sum_{j \in \mathbb{N}_n} \alpha_j L_j(\mathbf{y})\right) d\mu(\mathbf{y}). \end{aligned}$$

We stress that (2.5) and (2.6) are equivalent Cauchy problems on the n -dimensional phase spaces $\mathbb{X}_n$ and $\mathbb{R}^n$, respectively.

2.2. Step 2: Quadrature and discrete projection schemes. The ODE (2.6) on $\mathbb{R}^n$ is not yet implementable on a computer, because generally the integrals in K can not be evaluated in closed form and must be approximated by numerical quadrature. Such an approach generates a computationally implementable algorithm using *discrete projection schemes*.

We use a quadrature scheme induced by interpolation. We will later discuss the details of the quadrature, which approximates the integral operator

$$Q: \mathbb{X} \rightarrow \mathbb{R}, \quad Qv = \int_D v(\mathbf{y}) d\mu(\mathbf{y})$$

with a sum of the form

$$Q_n: \mathbb{X} \rightarrow \mathbb{R}, \quad Q_n v = \sum_{j=1}^n v(\mathbf{x}_j) \mu_j.$$

We can apply the quadrature scheme Q_n to the operator W in (2.5), define

$$(2.7) \quad W_n: \mathbb{X} \rightarrow \mathbb{X}_n, \quad (W_n v)(\mathbf{x}) = Q_n(w(\mathbf{x}, \cdot)v),$$

and arrive at the scheme

$$(2.8) \quad \begin{aligned} \tilde{u}'_n(t) &= -\tilde{u}_n(t) + P_n W_n F(\tilde{u}_n(t)) + P_n g(t), \quad t \in J, \\ \tilde{u}_n(0) &= P_n v, \end{aligned}$$

the ODE system for u at the collocated points, approximated by quadrature.

Following similar steps to subsection 2.1, we obtain

$$(2.9) \quad \begin{aligned} \tilde{\alpha}'(t) &= -\tilde{\alpha}(t) + \tilde{K}(\tilde{\alpha}(t)) + \gamma(t), \quad t \in J, \\ \tilde{\alpha}(0) &= \nu \end{aligned}$$

for the evolution of the vector of coefficients $\alpha(t) = [\alpha_1(t), \dots, \alpha_n(t)]^T$, where

$$\begin{aligned} \tilde{\alpha}_i(t) &= \tilde{u}_n(t)(\mathbf{x}_i), \quad \nu_i = v(\mathbf{x}_i), \\ \gamma_i(t) &= g(t, \mathbf{x}_i), \quad \tilde{K}_i(\tilde{\alpha}) = \sum_{j=1}^n w(\mathbf{x}_i, \mathbf{x}_j) f \left(\sum_{k \in \mathbb{N}_n} \tilde{\alpha}_k L_k(\mathbf{x}_j) \right) \mu_j. \end{aligned}$$

2.3. Convergence of the collocation scheme. In [5], Atkinson's approach for studying projection methods in Fredholm and Hammerstein integral equations [3, 4] is extended to time-dependent neural field equations. The central result of the paper is a bound for the error $\|u - u_n\|_{C(J, \mathbb{X})}$ in terms of the projection error $\|u - P_n u\|_{C(J, \mathbb{X})}$, that is, a bound for the collocation scheme in Step 1, in terms of the error of the interpolating projector.

THEOREM 2.2 (abridged from [5], Theorem 3.3, and the discussion on p. 570). *Under Hypothesis 2.1, if $P_n v \rightarrow v$ for all $v \in \mathbb{X}$, then $u_n \rightarrow u$ in $C(J, \mathbb{X})$. Further, there exist positive constants m and M , independent of n , such that*

$$m\|u - P_n u\|_{C(J, \mathbb{X})} \leq \|u - u_n\|_{C(J, \mathbb{X})} \leq M\|u - P_n u\|_{C(J, \mathbb{X})}.$$

Since the time interval J is compact it holds that $\|u - P_n u\|_{C(J, \mathbb{X})} = \|u(t_*) - P_n u(t_*)\|_{\mathbb{X}}$ for some $t_* \in J$, and thus the convergence rate of the collocation scheme is estimated from the interpolation error $\|v - P_n v\|_{\mathbb{X}}$ for some $v \in \mathbb{X}$.

The error of the discrete collocation scheme can in principle be estimated, given $\|u - \tilde{u}_n\| \leq \|u - u_n\| + \|u_n - \tilde{u}_n\|$, but we do not take this route here. Controlling $\|u_n - \tilde{u}_n\|$ requires rigorous error estimates for quadrature rules on curved domains D , which, to the best of our knowledge, are not yet available when the underlying interpolant is an RBF. Instead, we proceed in the remaining sections as follows:

1. We employ RBF interpolants whose interpolation error $\|P_n v - v\|$ is available in the literature and deduce rates for the error of collocation scheme $\|u - u_n\|$, as per Theorem 2.2.
2. We select a quadrature scheme built on RBFs, for which we provide numerical evidence of convergence rates. A guiding principle is to select a quadrature rule that matches the error of the collocation scheme.
3. We provide numerical evidence of convergence rates for the error $\|u - \tilde{u}_n\|$ of the fully discrete scheme in cases where an analytical solution to the neural field is available in closed form.

3. RBF interpolation and quadrature. We use RBFs for the interpolatory projector P_n and quadrature formula Q_n introduced in section 2. In particular, we use RBF-QF, which provide an interpolation-based technique to calculate quadrature weights for arbitrary sets of quadrature nodes with high-order accuracy. This quadrature is geometrically flexible: it works in domains of arbitrary dimension and in complex geometry. The approximation power comes from the remarkable properties of radial basis function (RBF) interpolation. In this section, we describe the algorithm used to interpolate functions, generate quadrature weights in flat and curved domains, and discuss the consequences of hyperparameter choices. This general approach can then be leveraged to implement our neural field approximation scheme.

3.1. Local interpolation. The first use of RBF interpolation was by Hardy in 1971 [32] to construct a topography from scattered elevation measurements. The interpolant was a linear combination of radially symmetric basis functions called *multiquadrics*. Critically, the centers of these basis functions coincided with the locations of the measurements, and as a result, the interpolants were guaranteed to exist and be unique. The Mairhuber–Curtis theorem shows that such guarantees break down when basis functions are chosen independently of the interpolation nodes [24, 25, 43], as occurs in multivariate polynomial interpolation. Later developments showed that other radially symmetric basis functions could be used similarly, with resulting interpolants exhibiting spectral convergence as the number of nodes increased [17, 67]. For a comprehensive introduction to RBF interpolation, see [25].

The price of spectral convergence, however, is that finding interpolant coefficients requires solving an $n \times n$ linear system, which becomes prohibitively expensive and ill-conditioned as n (the number of interpolation nodes) increases. There are two approaches to remedy this limitation. First, one can choose compactly supported basis functions, often called Wendland functions [25, 64], to produce a sparse, banded linear systems, which can be solved efficiently. We opt for the second approach: local RBF interpolation using polyharmonic splines and appended polynomial basis functions, which is the focus of this subsection. For each stencil, the unisolvency of the appended polynomial basis is ensured by selecting distinct, well-distributed nodes, so that the associated Vandermonde matrix is full rank. On curved surfaces, we work in local coordinate parameterizations of each stencil, so the same unisolvency conditions as in the planar case apply [10].

RBF interpolation is often touted as a “mesh-free” method and this is certainly true of global RBF interpolation, which yields a smooth interpolant over the entire domain. Local RBF interpolation is arguably mesh-free as well, but it produces a piecewise interpolant that is smooth over each element in a partition of the domain but potentially discontinuous along the boundaries of these elements. Generally, this partition is taken to be the Voronoi partition and is not explicitly calculated. It is thus mesh-free in the sense that the mesh is implicit. We emphasize this because the RBF-QF algorithm for surfaces will explicitly require a triangulation in lieu of a Voronoi partition and is thus not a mesh-free algorithm. We now provide a self-contained description of the interpolation algorithm.

Given a compact domain $D \subset \mathbb{R}^{\dim}$, we seek an approximation to a sufficiently smooth function $f : D \rightarrow \mathbb{R}$, using a discrete finite set of interpolation nodes $\Xi_n \subset D$ and a partition of the domain $\{E_j\}_{j=1}^m$. We will approximate f piecewise on each element E_j , where each piecewise component will be a linear combination of basis functions. For each element E_j , there is an associated subset of interpolation nodes $S(E_j) \subseteq \Xi_n$, which we call a stencil. Generally, the elements E_j are small polygons

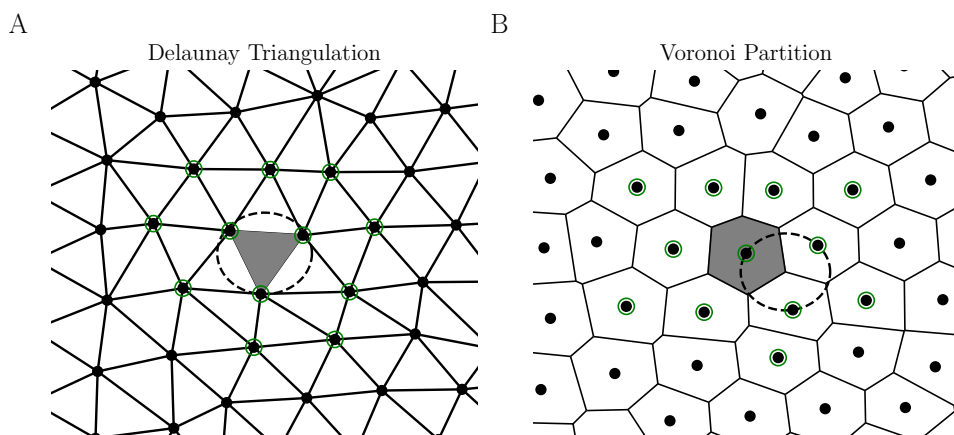


FIG. 3.1. **Stencils.** Each panel shows a subset of the interpolation nodes Ξ_n (black dots) for a domain $D \subset \mathbb{R}^2$, the boundaries (solid lines) enclosing partition elements E_j , and an example element (shaded gray) and its stencil points (green circles). (Color images are available online.) A. Delaunay triangulations partition elements using interpolation nodes as vertices. An element's circumscribing circle does not contain any additional interpolation nodes. B. Voronoi partitions assign one interpolation node per element. A circle centered at a Voronoi vertex can always be drawn to contain only the centers of neighboring elements.

that shrink or are divided as more points are added. The stencil $S(E_j)$ contains the closest k points in Ξ_n to a suitably defined center of E_j , making $k = |S(E_j)|$ a hyperparameter called the *stencil size*. In what follows, we will sometimes use E to refer to a generic element in the partition $\{E_j\}_{j=1}^m$.

Partitions are demonstrated in two examples depicted in Figure 3.1. The Delaunay triangulation is popular for partitioning planar domains and widely integrated into many standard mathematical software libraries [28]. Interpolation nodes are used as the vertices of triangular elements that form the partition (Figure 3.1A). The defining feature of the Delaunay triangulation is that a circle circumscribing each triangular element contains no other interpolation nodes except its vertices. Alternatively, Voronoi partitions (Figure 3.1B) assign each element a single interpolation node and define it as the set of points in D closer to that node than to any other. The Voronoi diagram is the graph dual of the Delaunay triangulation for a given set of points.²

We will refer to $\{E_j\}_{j=1}^m$ as the partition, mesh, or triangulation interchangeably and the subsets E_j as elements (of the partition), patches, or triangles depending on the context. While our presentation allows for general choices of interpolation nodes and meshes, not all such combinations yield accurate or stable interpolants. A useful quantity associated with the nodes is the *covering radius* [18, 25]

$$(3.1) \quad h = \sup_{\mathbf{x} \in D} \min_{\mathbf{y} \in \Xi_n} \|\mathbf{x} - \mathbf{y}\|,$$

the supremum of the radii of circles with centers $\mathbf{x} \in D$ that contain no interpolation nodes. The covering radius is analogous to mesh spacing and scales asymptotically as $h \sim n^{-1/\dim}$, as in regular grids.

²This is almost always true. Nonuniqueness can arise when a Voronoi vertex is shared by more than three cells (e.g., in a Cartesian grid), but this is rare for randomly chosen points and can be resolved by small perturbations.

This notion arises naturally in both the Delaunay triangulation and Voronoi partition. In the Delaunay case, the radius of any element's circumscribed circle (see Figure 3.1A) provides a lower bound on h , and if the triangulation covers D , then h is the maximum of these radii across all triangular elements. In the Voronoi case, the covering radius is the maximum distance from a node to a vertex of its associated patch. Figure 3.1B shows a circle centered on a Voronoi vertex with radius equal to the distance to the adjacent nodes. In both cases, the covering radius is the maximum over a finite set of such radii, making it computationally straightforward.

Going forward, we assume each stencil has fixed size $|S(E)| = k$ for all $E \in \{E_j\}_{j=1}^m$. We introduce a parameter, “deg,” to control the order of accuracy of our interpolant. Over each element E , the approximation is expressed as a linear combination of RBFs $\varphi_{\mathbf{y}}$ and polynomial basis functions $\pi_{\boldsymbol{\alpha}}(\mathbf{x}) = \mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha}$ is a multi-index, and $\{\pi_{\boldsymbol{\alpha}}\}_{|\boldsymbol{\alpha}| \leq \text{deg}}$ forms a basis for $\mathbb{P}_{\text{deg}}(D)$, the space polynomials of degree at most deg. A function $f \in C(D)$ is then approximated using the piecewise-defined function

$$(3.2) \quad s(\mathbf{x}) := \sum_{j=1}^m s_{E_j}(\mathbf{x}) \chi_{E_j}(\mathbf{x}), \quad \mathbf{x} \in D,$$

where $\chi_E: D \rightarrow \mathbb{R}$ is the indicator function on E and where s_E is the function

$$s_E(\mathbf{x}) = \sum_{\mathbf{y} \in S(E)} c_{E,\mathbf{y}} \varphi_{\mathbf{y}}(\mathbf{x}) + \sum_{|\boldsymbol{\alpha}| \leq \text{deg}} d_{E,\boldsymbol{\alpha}} \pi_{\boldsymbol{\alpha}}(\mathbf{x}).$$

The RBFs $\psi_{\mathbf{y}}$ are each translates of the same *basic function* $\psi_{\mathbf{y}}(\mathbf{x}) = \Phi(\|\mathbf{x} - \mathbf{y}\|)$ [25], but not all radially symmetric functions are suitable basic functions. We will restrict ourselves to polyharmonic splines with fixed-order parameter ℓ

$$(3.3) \quad \Phi(r) = \begin{cases} r^{\ell} & \text{for } \ell \text{ odd,} \\ r^{\ell} \log(r) & \text{for } \ell \text{ even.} \end{cases}$$

To determine the coefficients $c_{\mathbf{y}}$ and $d_{\boldsymbol{\alpha}}$ we enforce two sets of conditions known as *interpolation conditions* and *moment conditions*, respectively:

$$\begin{aligned} (\text{interpolation}) \quad & f(\mathbf{x}) = s_E(\mathbf{x}) && \text{for } \mathbf{x} \in S(E), \\ (\text{moment}) \quad & 0 = \sum_{\mathbf{y} \in S(E)} d_{\boldsymbol{\alpha}} \mathbf{y}^{\boldsymbol{\alpha}} && \text{for } |\boldsymbol{\alpha}| \leq \text{deg.} \end{aligned}$$

Note that though we refer to these as “interpolation conditions”, a point $\mathbf{x} \in S(E)$ but $\mathbf{x} \notin E$ is interpolated by the local interpolant s_E , but not by the global piecewise interpolant s . The full approximation interpolates f at each $\mathbf{x} \in \Xi$ since every node lies in some element E , but each local interpolant s_E satisfies additional conditions not directly reflected in the global approximation.

The moment conditions ensure *polynomial reproduction*: for any $p \in \mathbb{P}_{\text{deg}}(D)$, the local interpolant satisfies $s_E = p$ on its domain. This property underpins the approximation power of the local interpolation scheme.

Thus, the coefficients are determined by solving the linear system

$$(3.4) \quad \begin{bmatrix} A & \Pi \\ \Pi^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_E \\ \mathbf{0} \end{bmatrix},$$

where the full block matrix is the *interpolation matrix*. Here, $A_{ij} = \Phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$ is the *RBF interpolation matrix*, $\Pi_{i,\boldsymbol{\alpha}} = \mathbf{x}_i^{\boldsymbol{\alpha}}$ is the *polynomial matrix*, and $(\mathbf{f}_E)_i = f(\mathbf{x}_i)$ for $\mathbf{x}_i \in S(E)$.

Before turning to error estimation and convergence theory, we comment on key parameters in the RBF interpolant. We recommend first choosing $\deg$, the degree of the appended polynomial, which governs the order of accuracy as $h \rightarrow 0$. This choice determines the size of the polynomial matrix Π , which must have more rows than columns to ensure the system (3.4) is invertible. Specifically, the stencil size must satisfy $k \geq \binom{\deg + \dim}{\deg}$.

We will specify our choice of k for all numerical experiments, though there are often reasons to choose k larger than the minimum. As a general principle, increasing $\deg$ improves accuracy, while increasing k enhances stability. For instance, near domain boundaries, stencils may become one-sided, potentially reducing accuracy due to Runge's phenomenon. Enlarging stencils in such regions can mitigate this effect and improve accuracy without changing h . For further discussion on the interplay between stencil size and polynomial degree, see [8, 9, 26].

We must also choose ℓ , the degree of the polyharmonic spline (see (3.3)). Although increasing ℓ can reduce error, the improvement diminishes quickly, and the nonsingularity of the interpolation matrix requires $\deg \geq \lfloor \ell/2 \rfloor + 1$ [25, 64]. In practice, we therefore adopt parameters that balance accuracy, stability, and cost: stencil sizes $k = 21$ or 32 , slightly larger than the minimum needed for unisolvency, improve stability near boundaries without substantially increasing cost [9, 26]; and we fix $\ell = 3$, since larger values yield little accuracy gain but worsen conditioning [64]. With appended polynomials up to degree $\deg = 4$, these choices consistently performed well, and convergence tests (sections 3.2, 3.3) confirm robust accuracy across both flat and curved geometries.

We conclude this section by interpreting the interpolant s defined in (3.2) as a projection $P_n f$, aligning this step with the framework in section 2. While the projection operator $P_n : C(D) \rightarrow C(D)$ is expected to preserve continuity, the interpolant s is constructed locally and may be discontinuous across element boundaries. A continuous projector P_n can be obtained by applying weighted averages to neighboring local interpolants.

For example, consider a Voronoi partition $\{V_i\}_{i=1}^n$ of D , and let $\{T_{i,j,k}\}$ denote the associated Delaunay triangulation, where $1 \leq i, j, k \leq n$ index the vertices of the triangle. For any $\mathbf{x} \in T_{i,j,k} \subseteq D$, let $b_{i,\mathbf{x}}, b_{j,\mathbf{x}}, b_{k,\mathbf{x}}$ be the barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \Xi$. We define the piecewise projector

$$(P_n f)(\mathbf{x}) = \sum_{T_{i,j,k}} \chi_{T_{i,j,k}}(\mathbf{x}) \left(b_{i,\mathbf{x}} s_{V_i}(\mathbf{x}) + b_{j,\mathbf{x}} s_{V_j}(\mathbf{x}) + b_{k,\mathbf{x}} s_{V_k}(\mathbf{x}) \right),$$

which is continuous across D and smooth within each triangle. With P_n defined, the Lagrange basis $\{L_1, \dots, L_n\}$ is given by $L_i = P_n v_i$ where each $v_i \in C(D)$ satisfies $v_i(\mathbf{x}_j) = \delta_{ij}$.

Second, the projection method requires $P_n f \rightarrow f$ as $n \rightarrow \infty$ but the relationship between n and our interpolant is not apparent. To relate them, we consider a sequence of interpolation node sets $\{\Xi_n\}$ with covering radii $h(\Xi_n) \rightarrow 0$ as in (3.1). Such sequences can be easily constructed by refining triangulations or by placing points to efficiently fill the domain.

This framing aligns with interpolation convergence theorems in the literature, which establish error bounds for polyharmonic spline interpolation such as

$$\|f - P_n f\|_\infty \leq C h^{\deg}$$

or better in some cases [18, 25, 34, 64]. These results apply to *global* interpolation ($k = n$). While we found no published proof that *local* interpolation achieves the same

rate, we expect such a result could follow from adaptations of the existing global proofs [25, 64], though it would likely require geometric justification that the cone condition can be relaxed, except near boundaries. In practice, studies using local interpolation offer numerical evidence of convergence, often showing convergence rates faster than $O(h^{\deg})$ [8, 9, 26], which we observe in section 4.

3.2. Quadrature in flat domains. The theory of RBF-QF has its roots in finite difference methods, known as RBF-FD. Both quadrature and differentiation are linear functionals—linear maps into $\mathbb{R}$ —and share a common theoretical foundation. Derivative approximation using RBFs was introduced first [35], and the modern RBF-FD method based on local interpolation was discovered independently by several groups shortly afterward [19, 59, 60, 66]. Due to this history, some authors refer to RBF-QF as a variant of RBF-FD, though it produces quadrature weights rather than finite difference weights. We begin with the general theory for approximating linear functionals, then specialize to quadrature. This approach clarifies the distinction between RBF-QF and RBF-FD and highlights when and why a mesh is needed.

To approximate a linear functional $\mathcal{L}$ (such as quadrature or differentiation at a point) applied to a function $f : D \rightarrow \mathbb{R}$, we first consider the local RBF interpolant $s : D \rightarrow \mathbb{R}$ which, by (3.2), gives

$$\mathcal{L}f \approx \mathcal{L}s = \sum_E \mathcal{L}s_E \chi_E = \sum_E \sum_{\mathbf{y} \in S(E)} c_{E,\mathbf{y}} \mathcal{L}(\varphi_{\mathbf{y}} \chi_E) + \sum_{|\alpha| \leq \deg} d_{E,\alpha} \mathcal{L}[\chi_E \pi_\alpha].$$

The coefficients $c_{E,\mathbf{y}}$ and $d_{E,\alpha}$ are given in vector form in (3.4). Representing the previous equation as a sum over dot products, we have

$$\begin{aligned} \mathcal{L}f &\approx \sum_E [\mathcal{L}\varphi \quad \mathcal{L}\pi] \begin{bmatrix} c \\ d \end{bmatrix} = \sum_E [\mathcal{L}\varphi \quad \mathcal{L}\pi] \begin{bmatrix} A & \Pi \\ \Pi^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f}_E \\ \mathbf{0} \end{bmatrix}, \\ &= \sum_E [\mathbf{w}_E \quad \gamma_E] \begin{bmatrix} \mathbf{f}_E \\ \mathbf{0} \end{bmatrix} = \sum_{\mathbf{x}_i \in \Xi} w_{\mathbf{x}_i} f(\mathbf{x}_i), \\ &= \mathbf{w}^T \mathbf{f}, \end{aligned}$$

where $\mathbf{w}_E$ denotes the elementwise functional weights, and $\mathbf{w}$ the combined functional weights,

$$(3.5) \quad \mathbf{w}_E = \begin{bmatrix} A & \Pi \\ \Pi^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{L}\varphi \\ \mathcal{L}\pi \end{bmatrix}, \quad (\mathbf{w})_{\mathbf{x}_i} = w_{\mathbf{x}_i} = \sum_{E: \mathbf{x}_i \in S(E)} (\mathbf{w}_E)_{\mathbf{x}_i}.$$

If $\mathcal{L}$ is a differential (integral) operator, then $\mathbf{w}$ represents finite difference (quadrature) weights. Moreover, the expressions involving $w_{\mathbf{x}_i}$ and $\mathbf{w}$ help outline the implementation of the algorithm.

First, for each element E , we identify the stencil $S(E)$ and evaluate the functionals $\mathcal{L}(\varphi_{\mathbf{x}_i} \chi_E)$ and $\mathcal{L}(\mathbf{x}^\alpha \chi_E)$. We then form the system in (3.5) and solve for the weights $\mathbf{w}_E$, whose order reflects the ordering of the nodes in the stencil $S(E)$. Each node $\mathbf{x}_i$ may appear in multiple stencils and thus be assigned several weights $(\mathbf{w}_E)_{\mathbf{x}_i}$. The final quadrature weights are obtained by summing all contributions associated with each node across stencils.

Since we focus on quadrature, we set $\mathcal{L}f = Qf = \int_D f(y) d\mu(y)$ as in section 2. While our theory suggests $Q_n = QP_n$, we use the simpler form $Q_n = Qs = \mathbf{w}^T \mathbf{f}$, which offers the same accuracy and is easier to compute. Each functional evaluation $\mathcal{L}\varphi, \mathcal{L}\pi$ thus involves analytically integrating an RBF or polynomial basis function over an element E . The resulting quadrature approximation is then $Q_n f = \mathbf{w}^T \mathbf{f}$.

3.3. Quadrature on smooth closed manifolds. When the cortical domain D is a 2-manifold with curvature, applying RBF-QF is more involved. Convergence theory for such schemes remains an active area of research [31], while implementations have been developed by Reeger and Fornberg [51, 52, 53]. We summarize their approach for smooth closed surfaces and refer to [53] for the treatment of surface boundaries. Numerical evidence of convergence rates is presented in subsection 4.4.

The method begins with a set of interpolation nodes Ξ and defines two sets of elements. The first is a planar triangulation $\mathcal{E} = \cup_{j=1}^m E_j$ with nodes exactly equal to Ξ . The method is designed for curved (nonplanar) domains where the cortical surface D does not coincide with $\mathcal{E}$. It constructs a second set of elements forming a partition $D = \cup_{j=1}^m \mathcal{T}_j$, where each $\mathcal{T}_j$ is a curved triangular patch on D , in contrast to the flat triangles E_j in $\mathcal{E}$ (see Figure 3.2). The two families share the interpolation nodes, $\Xi \subseteq D \cap \mathcal{E}$, and are paired such that for each j there exists a bijection $\mu_j(E_j) = \mathcal{T}_j$. We defer a precise definition of the mappings to a later section and simply note that once the families $\{E_j\}$, $\{\mathcal{T}_j\}$, and $\{\mu_j\}$ are in place, one can write, without approximation,

$$(3.6) \quad \int_D f(\mathbf{x}) dS = \sum_{j=1}^m \int_{\mathcal{T}_j} f(\mathbf{x}) dS = \sum_{j=1}^m \int_{E_j} f(\mu_j(\boldsymbol{\xi})) \|(\partial_{\xi_1} \mu_j \times \partial_{\xi_2} \mu_j)(\boldsymbol{\xi})\| d\boldsymbol{\xi},$$

in which $\|(\partial_{\xi_1} \mu_j \times \partial_{\xi_2} \mu_j)(\boldsymbol{\xi})\|$ is the Jacobian of μ_j at $\boldsymbol{\xi}$. The method introduced by Reeger and Fornberg amounts to [51, 52, 53] the following: (i) defining the mappings $\{\mu_j\}$; (ii) selecting a quadrature scheme for the integral over E_i , on the right-hand side of (3.6).

The mappings $\{\mu_j\}$ depend on a set of *projection points* $\{\mathbf{p}_j\}$, as shown in Figure 3.2. For now, we assume these points are given and explain later how to determine them. For each fixed j , let π_j denote the plane in which the triangle E_j . We define

$$\mu_j: \pi_j \rightarrow D, \quad \boldsymbol{\xi} \mapsto \mathbf{x},$$

where $\mathbf{x} \in D$ is the closest point to $\boldsymbol{\xi} \in \pi_j$ along the line connecting $\mathbf{p}_j$ and $\boldsymbol{\xi}$. Such a point exists provided $\boldsymbol{\xi}$ lies in a sufficiently small neighborhood of E_j and the node set Ξ is dense enough. The mapping μ_j then sends E_j to a curved triangle

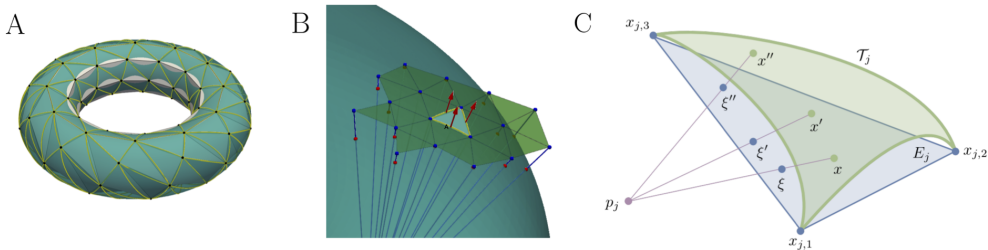


FIG. 3.2. A. The surface of a torus (D_T , transparent light green) is sampled at some well-distributed points (Ξ , black) which are then used for a triangulation ($\{E_j\}_{j=1}^M$, white flat triangles). Using the method described in subsection 3.3, these planar triangles are projected onto the torus to form a partition of surface triangles ($\{\mathcal{T}_j\}_{j=1}^M$, outlined in yellow). B. Stencil and projected stencil of a flat triangle (E_j , white, but appears light green because it is behind the surface) from a surface triangulation. Nearby surface points form the stencil ($S(E_j)$, red) and are projected along lines (blue) emanating from the projection point, onto co-planar points (ξ , blue). The projection point is computed from the vertices of the triangle and approximated edge normals (red arrows). C. A detailed diagram relating a flat triangle E_j and its associated surface triangle $\mathcal{T}_j$. Points $\mathbf{x} \in \pi_j$ on the plane of E_j are projected onto $\boldsymbol{\xi} \in \mathcal{T}_j$ along the lines through the projection point $\mathbf{p}_j$.

$\mathcal{T}_j \subseteq D$. This construction explains why $\{\mathbf{p}_j\}$ are called projection points: if $\mathbf{p}_j$ were a light source, then $\mathcal{T}_j$ would be the shadow of E_j cast onto the curved surface D . The point $\mathbf{x} \in \mathcal{T}_j$ is defined as the closest such intersection since the projection line may intersect D more than once. Finally, we note that μ_j is defined for points on π_j even outside E_j .

By construction, E_j and $\mathcal{T}_j$ intersect at the nodes $\{\mathbf{x}_{j,1}, \mathbf{x}_{j,2}, \mathbf{x}_{j,3}\} \subset \Xi$ (see Figure 3.2). The mapping μ_j sends each edge of the planar triangle E_j connecting $\mathbf{x}_{j,k}$ and $\mathbf{x}_{j,l}$ to the corresponding edge of the curved patch $\mathcal{T}_j$, preserving endpoints. Each such edge, together with the projection point $\mathbf{p}_j$, lies in a common *cutting plane*; and thus $\mathbf{p}_j$ is the intersection of three cutting planes.

The definition of the mappings $\{\mu_j\}$ and the properties above are valid for any choice of projection points $\{\mathbf{p}_j\}$. However, these points must be selected carefully to ensure that $\{\mathcal{T}_j\}$ forms a valid partition of D . Reeger and Fornberg outline a method to choose projection points based on the normal vectors $\{\mathbf{n}_j\}$ of the planar triangles $\{E_j\}$. For any pair of adjacent triangles E_j and $E_{j'}$, sharing an edge $E_j \cap E_{j'}$, a necessary condition for $\{\mathcal{T}_j\}$ to partition D is that the maps μ_j and $\mu_{j'}$ both send the shared edge to the same curved edge in their respective curved triangles, that is,

$$\mu_j(E_j \cap E_{j'}) = \mathcal{T}_j \cap \mathcal{T}_{j'} = \mu_{j'}(E_j \cap E_{j'}) \quad \text{for all } j \text{ and } j',$$

which implies that $\mathbf{p}_i$ and $\mathbf{p}_j$ must lie in the same cutting plane. They resolve this by choosing the cutting plane orthogonal to $\mathbf{n}_j - \mathbf{n}_{j'}$ with the condition $\mathbf{p}_j \cdot (\mathbf{n}_j - \mathbf{n}_{j'}) = 0$. This approach determines the distribution $\{\mathbf{p}_j\}$, as each planar triangle is associated with three such cutting planes, uniquely fixing its projection point.

Once the bijections $\{\mu_j\}$ are defined, a quadrature rule for integrands $g: E_j \rightarrow \mathbb{R}$ can be written as

$$\int_{E_j} g(\boldsymbol{\xi}) d\boldsymbol{\xi} \approx \sum_{r=1}^{q_j} g(\boldsymbol{\xi}_{j,r}) \rho_{j,r},$$

where the quadrature nodes satisfy

$$\mu_j(\boldsymbol{\xi}_{j,r}) = \mathbf{x}_{j,r}, \quad r \in \mathbb{N}_{q_j}, \quad j \in \mathbb{N}_m,$$

and $\{\mathbf{x}_{j,1}, \dots, \mathbf{x}_{j,q_j}\} = S(E_j)$; that is, the quadrature nodes for E_j are the preimages of the stencil points under μ_j (see subsection 3.1 for a definition of $S(E_j)$). Importantly, the nodes $\{\boldsymbol{\xi}_{j,r}\}_{r=1}^{q_j}$ lie in the plane π_j , but not necessarily within the triangle E_j itself, as illustrated in Figure 3.2. The points $\mu_j^{-1}(S(E_j)) \in \pi_j$ now form a planar stencil for E_j , and we use the procedure outlined in subsection 3.2. Specifically, we treat each E_j as a flat domain and apply RBF-QF locally, using the basis functions evaluated at the stencil nodes and integrating them over E_j . This yields weights that, when paired with function evaluations on the surface at $\mathbf{x}_{j,r} = \mu_j(\boldsymbol{\xi}_{j,r})$, define a quadrature rule that approximates integrals over the curved domain D .

4. Numerical experiments for neural fields. This section presents numerical experiments using RBF-QF, both as a standalone quadrature method and within a method of lines neural field simulation. We begin by introducing notation we will use throughout the section.

We consider three spatial domains of integration: (i) $D_1 = [0, 1]^2$, the unit square; (ii) $D_{2\pi} = [-\pi, \pi]^2$ the square of width 2π centered at the origin; and (iii) D_T , a torus with major radius $R = 3$ and minor radius of $r = 1$, parameterized by angles $(\varphi, \theta) \in D_{2\pi}$ as

$$(4.1) \quad \mathbf{x}(\varphi, \theta) = \begin{bmatrix} x(\varphi, \theta) \\ y(\varphi, \theta) \\ z(\varphi, \theta) \end{bmatrix} = \begin{bmatrix} (R + r \cos \theta) \cos \varphi \\ (R + r \cos \theta) \sin \varphi \\ r \sin \theta \end{bmatrix}.$$

We denote the inverse map by $\psi : D_T \rightarrow D_{2\pi}$.

Unless otherwise noted, we use the standard Euclidean distance metric in $\mathbb{R}^2$ or $\mathbb{R}^3$, as appropriate, for stencil selection and evaluating basic function inputs. For certain test functions and neural field solutions on D_1 or $D_{2\pi}$, we instead use a doubly periodic distance metric with periods $\Lambda = 1$ and $\Lambda = 2\pi$, respectively, defined by

$$(4.2) \quad \text{dist}(\mathbf{x}, \mathbf{y}) = \min_{i, j = -1, 0, 1} \left\| \mathbf{x} - \mathbf{y} + \Lambda \left(i \begin{bmatrix} 1 \\ 0 \end{bmatrix} + j \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \right\|.$$

Most of our test functions, as well as both neural field solutions, use scaled Gaussian functions defined with respect to this periodic distance:

$$(4.3) \quad \text{Gauss}(\mathbf{x}, \mathbf{y}; \sigma) = \frac{1}{2\pi\sigma^2} \exp \left[-\frac{1}{2\sigma^2} \text{dist}^2(\mathbf{x}, \mathbf{y}) \right].$$

In this setting, the kernel function is defined as $w(\mathbf{x}, \mathbf{y}) = \tilde{w}(\text{dist}(\mathbf{x}, \mathbf{y}))$, where $\tilde{w}$ is a standard radial function (e.g., Gaussian). While $\tilde{w}$ is not periodic, periodicity is enforced via the modified distance function in (4.2), which wraps Euclidean distance across boundaries, effectively defining w on a flat torus. This approach ensures that w depends on the shortest periodic path between $\mathbf{x}$ and $\mathbf{y}$, not just their Euclidean separation.

In neural field simulations we use the sigmoidal firing-rate function

$$(4.4) \quad f[u] = \left(1 + \exp(-\gamma(u - \vartheta)) \right)^{-1}$$

with gain $\gamma = 5$ and threshold $\vartheta = 1/2$.

To test the quadrature within a method of lines neural field simulation, we proceed with a *method of manufactured solutions* to construct equations with known analytic solutions. Specifically, we choose a solution $u(t, \mathbf{x})$ and define a forced neural field equation parameterized by a weight kernel w and the firing rate function f :

$$(4.5) \quad \partial_t u(t, \mathbf{x}) = -u(t, \mathbf{x}) + \int_D w(\mathbf{x}, \mathbf{y}) f[u(\mathbf{y})] dy + F(t, \mathbf{x}),$$

where the forcing term F is chosen to ensure that u satisfies the equation exactly.

4.1. Node selection. We briefly describe our selection of quadrature nodes and triangular meshes in each domain. Rather than detailing each construction, we provide a simplified description and refer interested readers to the code availability section (section 6) for implementation details, including quadrature weight generation, node placement, and time integration. For the square domains D_1 and $D_{2\pi}$ we compare two node families. The first places interior nodes at the vertices of a regular equilateral triangle tiling, with equally spaced nodes along the boundary. We refer to this as the *regular grid*, since interior stencils are rotated translates of each other and yield identical weights (Figure 4.1E). The second family distributes random interior nodes using a deterministic point-repulsion process, again placing equally spaced nodes along the boundary. An example is shown in Figure 4.1C. In both cases, we use Delaunay triangulation of the quadrature nodes to define the mesh.

For quadrature nodes on the torus D_T , we use a deterministic regular grid of *spiral nodes*. They are defined by a triangular tiling of the parameter space $D_{2\pi}$ that avoids thin triangles when mapped onto the torus. The nodes lie on lines of constant θ (rotation around the minor axis) and are equally spaced in the φ -direction, but staggered so they form approximately 60° angles with the θ -constant lines. These angled lines in trace closed spiral paths on the torus surface. While the tiling is regular in parameter space, it does not account for the curvature of the surface, and thus triangle sizes vary—becoming larger in regions of positive curvature and smaller in regions of negative curvature.

4.2. Quadrature experiments on the unit square. It is standard to verify quadrature convergence by measuring relative error as the mesh is refined (i.e., as the spacing $h \rightarrow 0$). We have performed numerous such experiments and present a representative sample in subsection 4.3. Before turning to convergence rates, we first examine specific meshes to illustrate how their structure influences the resulting quadrature weights and errors. This focus is worthwhile because unlike conventional quadrature methods, which require specific node placements (e.g., Gaussian or Newton–Cotes), RBF-based quadrature can accommodate arbitrary node sets. In this setting, mesh spacing h is a useful summary statistics but does not fully determine the node layout or resulting weights.

To demonstrate more precisely, we introduce a qualitative test in which a localized test function smooths the quadrature operator. Specifically, we define a family of steep Gaussian test functions $f_{\mathbf{x}_0} : D_1 \rightarrow \mathbb{R}$ by³

$$(4.6) \quad f_{\mathbf{x}_0}(\mathbf{x}) = 2\pi 10^2 \text{Gauss}(\mathbf{x}, \mathbf{x}_0; 10),$$

where each function is centered at $\mathbf{x}_0 \in D_1$. These Gaussians, defined in (4.3), using the doubly periodic distance (4.2), integrate to the same constant regardless of their center. The mass is sharply localized near $\mathbf{x}_0$, with rapid decay in all derivatives away from the center. While the periodic distance is not smooth, the resulting test functions are effectively smooth to machine precision. An example is shown in Figure 4.1A.

Figure 4.1B shows the quadrature nodes, colored by their associated weights. We observe that weights along the boundary are consistently smaller. This is partly because boundary nodes appear in fewer stencils and partly due to mild clustering near the edges—an intentional design to mitigate boundary-related errors such as the Runge phenomenon. Five nodes have negative weights (highlighted with black circles), though each is small in magnitude. The histogram to the right in Figure 4.1B reveals a bimodal distribution: the larger mode is centered near $1/n = 0.0005$, consistent with the average area per node, while the smaller mode corresponds to the boundary-adjacent nodes. This empirical structure reflects both the local stencil geometry and the rule’s enforcement of exactness for constant functions.

We generate approximately $n = 2000$ random and regular nodes (as described in section 4) and apply the RBF-QF algorithm from subsection 3.2 to compute quadrature nodes Ξ and associated weights $\{w_{\xi}\}_{\xi \in \Xi}$. To assess spatial error, we introduce a relative error as a function of the test function center:

$$(4.7) \quad E(\mathbf{x}_0) = \frac{Qf_{\mathbf{x}_0} - Q_n f_{\mathbf{x}_0}}{Qf_{\mathbf{x}_0}}.$$

³The prefactor of $2\pi 10^2$ has been used in the numerical experiment, but it has no impact on the relative error presented below, and can be safely ignored

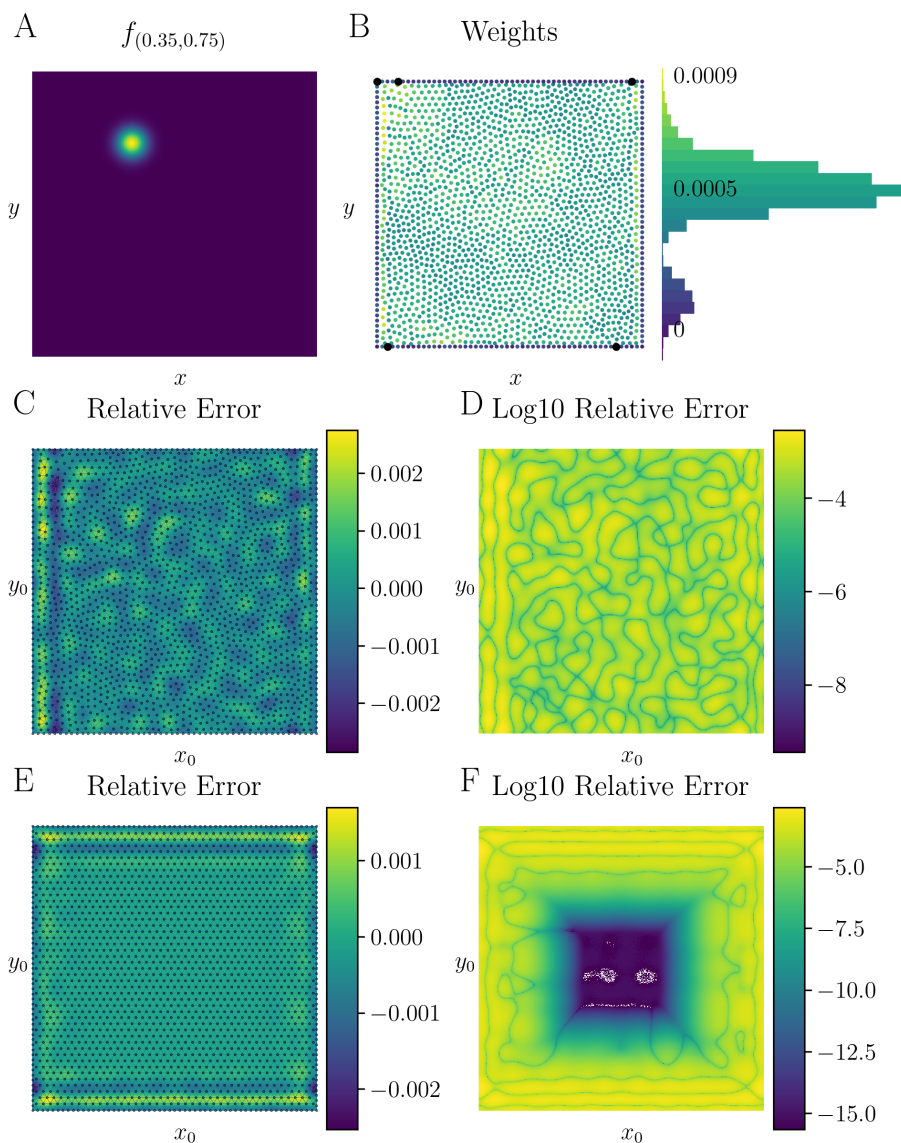


FIG. 4.1. **RBF-QF on the unit square.** A. A single test function as in (4.6). B. Quadrature node locations for $n = 2000$ points, colored by their associated weights. Boundary-adjacent nodes have smaller weights due to asymmetric stencils, and five nodes yield negative weights (black circles). Right: a histogram of weights showing a bimodal distribution—one mode near $1/n = 0.0005$ and a second associated with boundary clustering. C–D. Relative error (C) and $\log_{10}$ relative error (D) of the quadrature rule applied to Gaussian test functions $f_{\mathbf{x}_0}$ centered throughout the domain. E–F. The same as panels C–D, but using a near-regular triangular grid. Interior weights approach $1/n$ and the error approaches machine precision away from the boundary.

Large values of $|E(\mathbf{x}_0)|$ indicate substantial quadrature error near $\mathbf{x}_0$, suggesting the local node configuration contributes disproportionately to the total error. This can be interpreted as measuring the relative error in approximating the convolution $(1 * f_{(\cdot)})(\mathbf{x})$ via our quadrature rule.

We use localized Gaussian test functions to visualize how quadrature error varies spatially for different node configurations. In Figure 4.1, we demonstrate this on two node sets in the unit square (see section 4). Figure 4.1C shows a heatmap of $E(\mathbf{x}_0)$, the relative signed quadrature error of our Gaussian test functions $f_{\mathbf{x}_0}$. We again use $n = 2000$ quadrature nodes, the basic function r^3 , a stencil size of $k = 21$, and append third-degree polynomials. The error is relatively small, varies continuously, and is not systematically positive or negative. Most of the error magnitudes are well below the extremal values, highlighting the importance of testing across a variety of test functions.

For this particular mesh, the points are roughly evenly spaced except near the top and bottom boundary where they cluster. Stencils near the boundary are necessarily one-sided which can lead to Runge phenomenon [26, 27]. Clustering nodes near these boundaries is one way to combat this and demonstrates the utility of our spatially localized test functions. We observe that the error is generally higher along the left and right boundaries, where no clustering is applied, and comparatively lower near the top and bottom edges, where node clustering helps control boundary effects.

Any test function can exhibit surprisingly low error due to the random nature of the quadrature nodes. The function $E(\mathbf{x}_0)$ is continuous and oscillatory, and since the quadrature rule integrates constants exactly, the average of E over the domain is zero. The continuity of $E(\mathbf{x}_0)$ follows from the fact that $Q_n f_{\mathbf{x}_0}$ is a continuous function of $\mathbf{x}_0$, and $Q f_{\mathbf{x}_0}$ is constant in $\mathbf{x}_0$. The average value of E is zero because the quadrature weights are exact for constant functions, a fact that can be verified by a short computation. Thus, there must exist closed curves along which $E(\mathbf{x}_0) = 0$. We visualize this zero-level set by plotting $\log |E(\mathbf{x}_0)|$ in Figure 4.1D. While the overall structure remains similar across different meshes, the precise locations of high and low error regions will vary. This implies that, even for a fixed test function, quadrature error exhibits mesh-dependent spatial variation—even when meshes have similar n and point densities. For this reason, it is common in RBF quadrature studies to generate multiple random node sets for each choice of mesh parameters.

In practice, the presence of small negative weights does not prevent convergence. As noted earlier, Figure 4.1B shows five such weights, each small in magnitude, and their influence appears minimal in Figure 4.1C–D. It is often proposed that a quadrature rule is stable provided

$$\sum_n w_{\xi_n} - \sum_n |w_{\xi_n}| = 0,$$

i.e., if all weights are nonnegative [31]. High-order Newton–Cotes rules, for instance, fail this criterion and are indeed unstable. However, this notion of stability seems overly restrictive for our purposes. A more lenient condition—one that permits convergence—is that the difference between the total signed and absolute weights remains bounded as $n \rightarrow \infty$ [49, 61]. Indeed, the RBF-QF method has been analyzed under this framework [31], and our experiments confirm that it converges robustly even in the presence of small negative weights. Alternative strategies, such as L^1 optimization of stencil weights [1], have been proposed to mitigate or eliminate negative weights while maintaining accuracy, but we do not pursue these modifications here.

Last, Panels E and F of Figure 4.1 illustrate a case where the quadrature rule achieves spectral accuracy away from the boundary. The test functions and algorithmic parameters remain unchanged, but the nodes are now arranged on a near-regular triangular grid. In this symmetric configuration, the error is extremely small away from the boundary and reaches machine precision near the domain center.

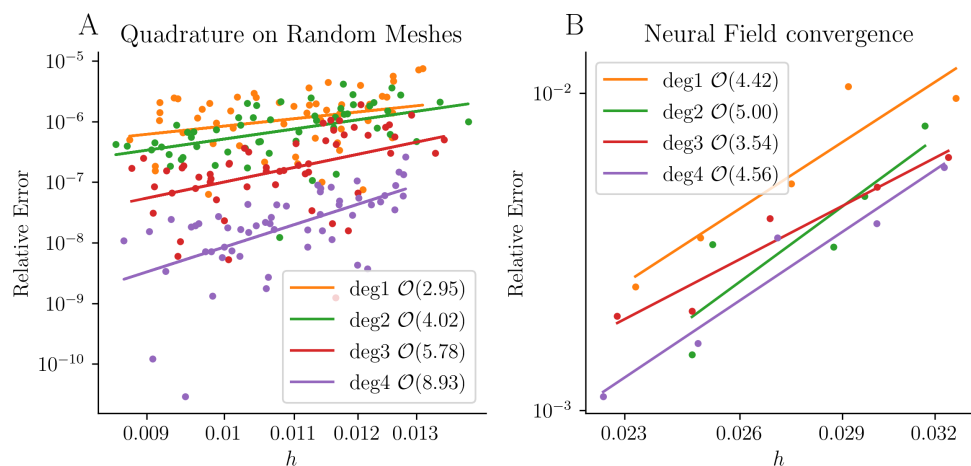


FIG. 4.2. A. Convergence of RBF-QF quadrature for the function $T_5(2x-1)T_4(2y-1)+1$ on the unit square. We use the RBF $\varphi(r)=r^3$, stencil size $k=21$, and append polynomial terms up to the specified degree (denoted by color). B. Convergence of a neural field simulation using the same quadrature rules (see the main text for details).

The white pixels visible in Panel F result from floating-point coincidence: the analytic integral and quadrature evaluation yield exactly the same double-precision value.

This level of accuracy arises from geometric regularity. When the mesh elements and their associated stencils are rotated translates of each other, the resulting quadrature weights are identical. Coupled with exact integration of constant functions, this symmetry forces all interior weights to equal the average area per node. Importantly, this value is independent of the degree of appended polynomial basis terms, which determine the formal convergence rate. As a result, the quadrature rule achieves arbitrarily high order in the interior, analogous to the spectral accuracy of the trapezoidal rule for periodic functions in one dimension. We observe the same phenomenon on other geometries: when nodes are placed on regular, nearly uniform grids on the square (Figure SM1), sphere (Figure SM2), or cyclide (Figure SM3), the quadrature achieves high accuracy, as documented in the supplementary materials (M175378_Supplementary_Materials_1.pdf [local/web 657KB]).

4.3. Convergence on the unit square. We evaluate the convergence of RBF-QF and neural field simulations on the unit square, using randomly chosen quadrature nodes and triangulations (see section 4). The test function is $f(x,y)=T_5(2x-1)T_4(2y-1)$, a product of Chebyshev polynomials (Figure 4.2C). Using $\varphi(r)=r^3$ and stencil size $k=21$, we compute quadrature weights. As shown in Figure 4.2A, errors decrease rapidly with increasing appended polynomial degree, often faster than $\mathcal{O}(h^{\deg})$. Panel B shows similar convergence for the simulation of the neural field solution, which we now describe.

We next present measured convergence rates for a method of lines simulation of a neural field, using RBF-QF to discretize the weight kernel. The manufactured solution is

$$u(t, \mathbf{x}) = f^{-1} [\text{Gauss}(\mathbf{x}, \mathbf{x}_0(t); \sigma = 11/10) + 1/10],$$

$$\mathbf{x}_0(t) = [\cos(t)/5, \sin(t)/5]^T,$$

which is indeed a solution to (4.5) on $(t, \mathbf{x}) \in [0, 1/10] \times D_{2\pi}$. Here f is defined in (4.4), and the weight kernel is $w(\mathbf{x}, \mathbf{y}) = \text{Gauss}(\mathbf{x}, \mathbf{y}; \sigma = 1/40)$ (see (4.3)). These expressions are used to construct a consistent forcing function. Figure 4.2B reports convergence results for different polynomial degrees. In each case, the observed order of accuracy exceeds the expected $\mathcal{O}(h^{\text{deg}})$ rate. For random meshes, the noise in the error curves is expected: each random placement of nodes slightly alters local point density and interpolation accuracy, and these local variations accumulate in the global error measure, producing the observed fluctuations.

4.4. Convergence on a torus. We next consider convergence results for quadrature and neural field simulations on a torus. For these experiments, we use spiral nodes for the quadrature (see section 4). While the algorithm does not require such regularity, using spiral nodes simplifies the setup and reduces error variability.

To test the surface RBF-QF algorithm directly, we use the function $f(x, y, z) = \sin(7x) + 1$ on the torus D_T with $\varphi(r) = r^3$, stencil size $k = 21$, and a polynomial basis appended up to a specified degree. While we lack a direct mesh radius on the surface, the nodes are well spaced, so we use $n^{-1/2} = \mathcal{O}(h)$ as a proxy measure of resolution [51, 53]. Figure 4.3A shows that for all polynomial degrees tested, convergence exceeds the expected rate $\mathcal{O}(n^{-\text{deg}/2})$.

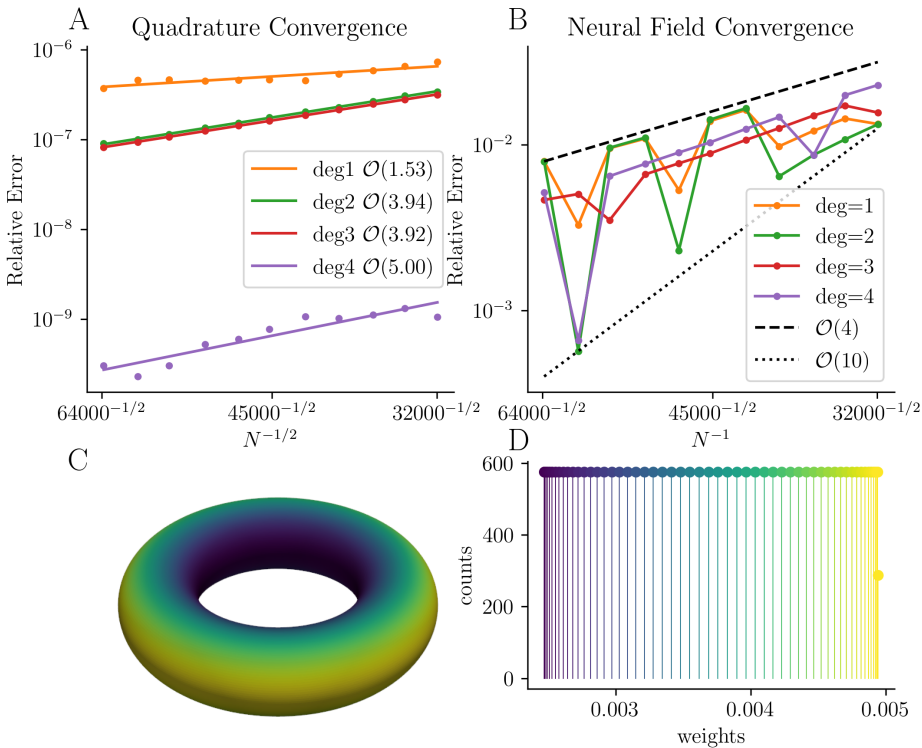


FIG. 4.3. A. Convergence of surface quadrature on the torus D_T using the test function $f(x, y, z) = \sin(7x) + 1$. In all cases, observed rates exceed the degree of appended polynomials. B. Convergence of a neural field simulation using the same surface quadrature rule. C. Torus surface colored by the quadrature weights for $k = 12$ and polynomial degree 2. Due to mesh regularity, weights are constant along θ and larger in regions of positive curvature where triangles are bigger. D. Histogram of the quadrature weights shown in panel C.

To test our method of lines for the neural field simulation on a surface, we construct a manufactured solution analogous to that used in the flat case. Specifically, we let

$$u(t, \mathbf{x}) = f^{-1} [\text{Gauss}(\psi(\mathbf{x}), \mathbf{x}_0; \sigma = 11/10) + 1/10], \quad \mathbf{x}_0 = [\cos(t)/5, \sin(t)/5]^T,$$

solve (4.5) with f as in (4.4) and weight kernel

$$w(\mathbf{x}, \mathbf{y}) = \text{Gauss}(\psi(\mathbf{x}), \psi(\mathbf{y}); \sigma = 1/40) \frac{1}{(R + r(\mathbf{y}) \cos \theta(\mathbf{y}))}.$$

This mirrors the previous solution but is defined on the surface domain $D_{2\pi}$ (in φ - θ coordinates), with the weight kernel adjusted by the inverse Jacobian of the torus parametrization $([R + r \cos \theta]^{-1})$. Although the analytical solution is identical, the quadrature weights differ due to the use of surface-specific integration techniques.

Results are shown in Figure 4.3. Panel A shows quadrature convergence on the torus using the test function $f(x, y, z) = \sin(7x) + 1$ with relative error decreasing as $n^{-1/2}$ and higher decay rates for higher-degree appended polynomials. Panel B reports convergence for the manufactured neural field solution, again showing high-order accuracy. Panel C visualizes the quadrature weights across the torus surface for $n = 32000$, while D presents a histogram of these weights. Weights are constant along lines of constant θ due to the symmetry of the node placement; their distribution reflects both curvature effects and histogram binning.

5. Showcase of neural field dynamics on curved geometries. Thus far, we have described and tested a numerical method for solving neural field equations on smooth, closed manifolds. In this section, we demonstrate its utility with simulations that reveal rich spatiotemporal dynamics on curved geometries. Each example extends classical results from planar or one-dimensional settings to non-Euclidean domains, identifying impacts and considerations of curvature on solution dynamics, opening new avenues for mathematical and scientific investigation. All computations use the same geodesic distance approximation, described in the code availability section.

5.1. Labyrinthine patterns on a deformed sphere. Previously, Coombes, Schmidt, and Bojak [23] investigated a neural field model in two spatial dimensions that exhibits *labyrinthine patterns* under appropriate initial conditions. These patterns emerge when a lateral inhibitory weight kernel is used with an unstable, near symmetric bump initial activity profile. Owing to the D_4 symmetry of the dominant instability, four arms of a cross extend outward from the bump and begin to branch, producing a single connected region of high neural activity, forming a complex labyrinth of thin, repeatedly branching corridors.

We perform a similar numerical experiment on a family of deformed spheres, demonstrating that surface curvature has a significant effect on the qualitative properties of the resulting labyrinthine patterns. Unlike the original experiment by Coombes, Schmidt, and Bojak [23], which employed a discontinuous Heaviside firing-rate function and a nonsmooth weight kernel, we use smooth, qualitatively similar functions that still give rise to complex spatiotemporal activity. Specifically, we define the weight kernel and firing-rate function as

$$(5.1) \quad w(\mathbf{x}, \mathbf{y}) = A_e \text{Gauss}(\mathbf{x}, \mathbf{y}, \sigma_e) - A_i \text{Gauss}(\mathbf{x}, \mathbf{y}, \sigma_i),$$

$$(5.2) \quad f(u) = \begin{cases} 0, & u < 0.06, \\ p(u), & 0.06 \leq u < 0.54, \\ 1, & u \geq 0.54, \end{cases}$$

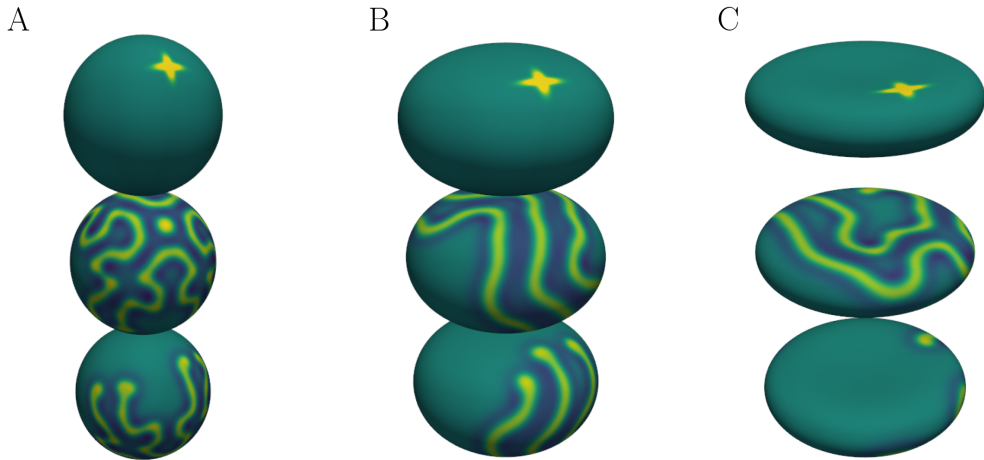


FIG. 5.1. **Effect of surface deformation on labyrinthine patterns.** *Labyrinthine patterns on a deformed spherical surface D_γ , with increasing deformation parameter from left to right: A. $\gamma = 0$ (perfect sphere). B. $\gamma = 0.4$ (moderate deformation). C. $\gamma = 0.8$ (flattened, blood-cell-like shape). An initial unstable, cross-shaped activity pattern (top) evolves to meandering spatiotemporal patterns (middle: north pole viewpoint; bottom: south pole viewpoint). Yellow indicates high neural activity ($u > 0$), green denotes baseline activity ($u \approx 0$), and dark green/blue indicates suppressed activity ($u < 0$). Surface curvature has a profound effect on the form and complexity of the resulting patterns. Click image to view full animation (Movie S1A, Movie S1B, Movie S1C).*

where $A_e = 5$, $A_i = 5$, $\sigma_e = 0.05$, $\sigma_i = 0.1$. Although $A_e = A_i$, the kernel satisfies $w(\mathbf{x}, \mathbf{x}) > 0$ since the Gaussian is normalized—its peak is higher for smaller σ —producing the desired lateral inhibitory property. The function $p(u)$ is a ninth-order polynomial chosen to ensure that f is four times continuously differentiable, yielding a smooth spline approximation to the Heaviside function $H(u - 0.3)$, remaining constant outside a narrow transition interval centered at the threshold.

The underlying surface D_γ is a one-parameter family of deformed spheres, with deformation governed by $\gamma \in [0, 1)$. When $\gamma = 0$, the surface is a standard sphere; as γ increases, the geometry is increasingly compressed along the vertical axis, reducing the pole-to-pole Euclidean distance to $2(1 - \gamma)$. The surface is implicitly defined by

$$1 = x^2 + y^2 + z^2 \left(1 - \frac{\gamma}{1 + (x^2 + y^2)/2.89} \right)^{-1}$$

and is visualized in Figure 5.1 for $\gamma = 0, 0.4, 0.8$.

We initialize the activity $u(0, \mathbf{x})$ on the undeformed sphere using the function

$$u(0, \mathbf{x}) = 5 \cdot \exp \left(-10 \left[\left(\cos \left(4 \arctan \left(\frac{y}{x} \right) \right) + 3 \right) \sqrt{x^2 + y^2} \right]^2 \right) H(z),$$

which defines a symmetric, cross-shaped pattern localized in the northern hemisphere. To orient this pattern appropriately, we apply a rigid rotation that maps the north pole $[0, 0, 1]$ to the point $[0.5, 0.3, \sqrt{0.5^2 + 0.3^2}]$, aligning the initial condition with the desired surface region before the deformation is applied.

We then simulate the neural field on each deformed surface using the kernel basis function $\varphi(r) = r^3$, a stencil size of $k = 32$, and append third-degree polynomials for accurate quadrature. Time integration is performed using the Adams–Bashforth 5

method with a fixed time step of $\Delta t = 10^{-2}$. The resulting dynamics, corresponding to various values of the surface deformation parameter $\gamma = 0, 0.4, 0.8$, are shown in Figure 5.1. The top row depicts the initial condition, rendered from a viewpoint where the *north pole* is visible. The middle row shows the activity at time $t = 200$ from the same viewpoint, while the bottom row presents the state at $t = 200$ with the surface inverted through the x - y plane to reveal the *south pole*.

Labyrinthine corridors remain connected but their wandering termini tend to veer away from regions of high curvature, illustrating how surface geometry profoundly influences neural field dynamics. This is akin to the pinning of propagating waves [12, 22] or attraction/repulsion of localized activity [36] observed by introducing inhomogeneities into the weight kernel of neural fields on flat domains.

5.2. Traveling spot steered by surface bumps. Our next example demonstrates how surface curvature influences the trajectory of traveling spot solutions in a neural field model with synaptic depression [16, 56]. The model consists of two coupled equations: one for the neural activity u , and one for the synaptic efficacy q , ranging from 0 (depleted) to 1 (fully available). The dynamics are given by

$$(5.3) \quad \partial_t u = -u + \int_D w(\cdot, \mathbf{y}) q(\cdot, \mathbf{y}) f[u(\cdot, \mathbf{y})] d\mathbf{y}, \quad \tau \partial_t q = 1 - q - \beta q f[u].$$

We use the laterally inhibitory weight kernel (5.1) with $A_e = 5$, $A_i = 7$, $\sigma_e = 0.05$, $\sigma_i = 0.1$. Such weights often produce *spot* solutions (localized circular active regions, also called pulses or bumps) in planar neural fields without adaptation ($\beta = 0$).

Incorporating synaptic depression ($\beta > 0$), as with other forms of adaptation [45], causes spots to travel. Introducing depression ($q < 1$) on one side of the spot, effective lateral inhibition is asymmetric and activity will increase on the side of the spot farthest from the depressed region [16, 56]. The spot then propagates, leaving a trail of synaptic depression in its wake.

We probe how surface curvature affects trajectories of such traveling spots by considering a surface that we refer to as a *bumpy sphere* (Figure 5.2), defined by the equation

$$1 + \sum_{\mathbf{x}_i} \left(\frac{2\pi}{10^2} \text{Gauss}(\mathbf{x}, \mathbf{x}_i, 1/10) \right) 1/10 = x^2 + y^2 + z^2,$$

where $\{\mathbf{x}_i\}_{i=1}^{100}$ are a set of 100 *bump centers* (randomly chosen, though roughly evenly spaced) on the unit sphere (see the remarks in the code availability section).

Figures 5.2A and B each show two snapshots from the same simulation, each depicting two views of the bumpy sphere. The left view is colored by the activity variable, while the right shows synaptic efficacy in grayscale. Both are rotated to center the spot in view, and the grayscale surface includes a red curve tracing the trajectory of the spot up to that time. Figure 5.2C presents the surface in spherical coordinates, with latitude and longitude along the vertical and horizontal axis, and color representing radial elevation. The red curve shows the full trajectory of the traveling spot. If the surface were perfectly spherical, the spot would follow a straight path along a great circle, akin to what was found on the planar case [16]. Instead, we observe consistent deviations in the trajectory that arise from geometric inhomogeneities. Two white curves show tangent great circles at times shown in panels A and B, illustrating how the trajectory is deflected by geometric inhomogeneities. This reinforces the point that surface irregularities create a similar potential surface which shapes the dynamics of evolving spatiotemporal solutions, akin to those created in neural fields on flat domains with weight inhomogeneities [38, 48].

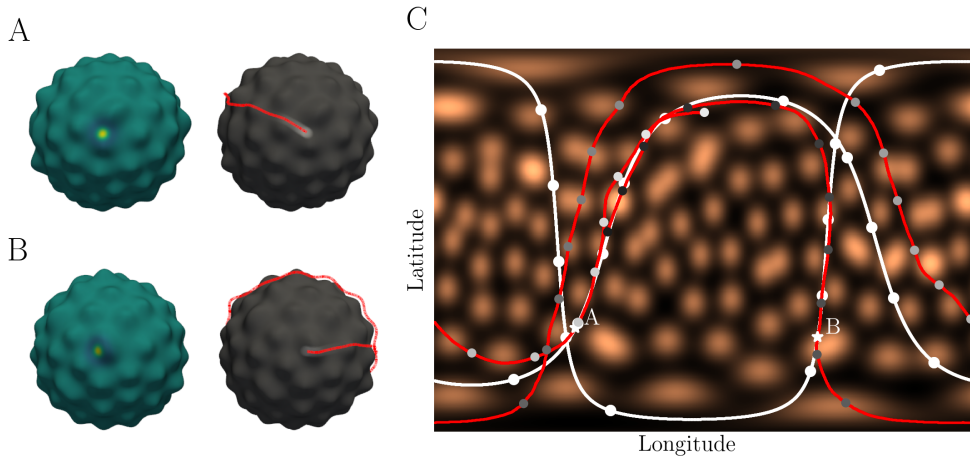


FIG. 5.2. **Traveling spot steered by bumps in curvature.** A–B. Snapshots from a simulation of a traveling spot on a bumpy sphere. Each shows the activity variable (left) and synaptic efficacy (right). The trajectory of the spot up to that point is shown as a red curve hovering above the surface. C. Projection of the bumpy surface into spherical coordinates, with longitude on the horizontal axis, latitude on the vertical axis, and surface elevation encoded by a heatmap. The traveling spot trajectory (red) sometimes hugs but also deviates from great-circle tangents (white), showing how bumps deflect motion. [Click to view full animation \(Movie S2\)](#).

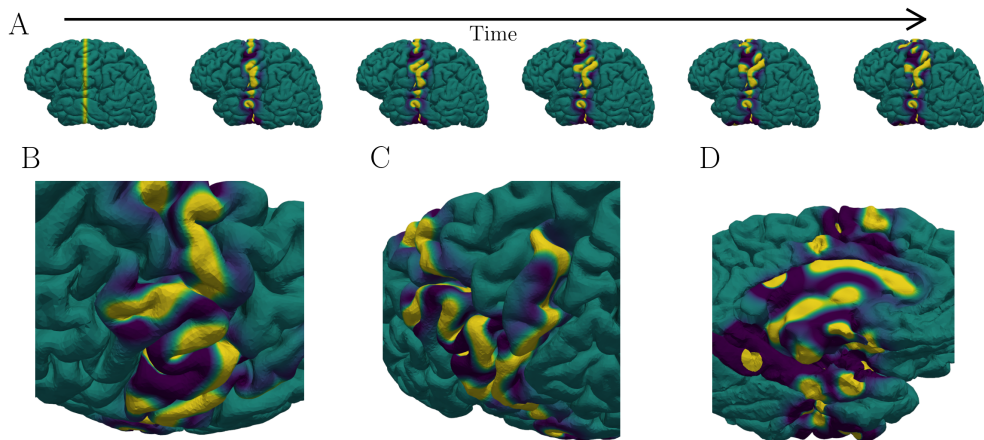


FIG. 5.3. **Labyrinthine dynamics on a cortex.** Here we show a simulation on the left hemisphere of a realistic human cortex using a laterally inhibitory weight kernel that favors the formation of labyrinthine patterns. A. Snapshots at evenly spaced early time points show the initial stripe of activity breaking into multiple labyrinth-like corridors that begin to grow and evolve. B–D. Final-time snapshots from different viewing angles highlight distinct cortical regions: B. frontal lobe; C. parietal lobe; and D. limbic cortex/medial surface. [Click to view full animation \(Movie S3\)](#).

5.3. Labyrinthine patterns on a human cortex. Finally, we build a simulation showcasing the evolution of a neural field on a cortical surface extracted from human data (Figure 5.3). See code availability for a link to the repository containing this mesh, representing the left hemisphere of a human cortex. It was generated using MNE-Python, which integrates FreeSurfer’s anatomical reconstruction pipeline to segment T1-weighted MRI scans and extract detailed cortical surfaces. The resulting

triangulated mesh captures the geometry of the pial surface, including sulci and gyri, enabling anatomically realistic neural field simulations. We use the same laterally inhibitory weight function defined in (5.1) with $A_e = A_i = 5$, $\sigma_e = 3$, $\sigma_i = 6$, along with the geodesic distance metric and parameters from subsection 5.1. The initial condition consists of a band of activity, which rapidly fragments into localized regions of varying size shaped by the surface curvature. The resulting dynamics include both stationary spots and labyrinthine corridors that tend to follow gyri and sulci—regions aligned with locally minimal curvature.

The mesh used here is visually compelling and anatomically detailed, though originally intended for visualization rather than numerical simulation. It exhibits some geometric irregularities—including a small duplicated region, uneven node density, and inconsistencies in surface normals. This leads to rapidly oscillating quadrature weights, including some large negative values. However, this is not indicative of an inherent instability of the RBF quadrature method. On other domains, including the flat-domain tests in section 4.2 and the torus in section 4.4, the method produces stable, accurate results with only small negative weights, even over long simulations. These observations demonstrate that the method is robust when applied to high-quality, well-distributed node sets. The oscillatory weights in the cortex case stem from the severe irregularity of the publicly available mesh we employed, rather than from the quadrature scheme itself. Nonetheless, the resulting dynamics reveal rich qualitative structure that underscores the potential of this framework and motivates further refinement and study.

Although a rigorous analytical treatment of these effects on curved surfaces remains an open question, these results highlight the potential for geometry to shape the dynamics of cortical activity. Our framework provides a powerful computational tool for probing such curvature-driven effects in structured neural field models.

6. Conclusion. We have presented a high-order, mesh-flexible solver for neural fields on smooth, closed surfaces using RBF-based interpolation and quadrature. The method is numerically stable and accurate and requires only a triangulated mesh and approximate vertex normals. Unlike spectral methods, which rely on structured domains, our approach—like finite element methods—supports arbitrary geometries without requiring element construction. Although the resulting quadrature matrices are sparse, pairwise interactions lead to $\mathcal{O}(n^2)$ complexity in the number of nodes. This scaling motivates the use of high-order schemes that achieve accuracy with fewer degrees of freedom, especially in cortex-scale modeling or inverse problems involving kernel learning.

Simulations on bumpy spheres and realistic cortical surfaces show how geometry can steer and constrain activity, extending phenomena observed in planar neural fields with inhomogeneous coupling (e.g., wave slowing, deflection, or pinning) to non-Euclidean domains. In Figure 5.1, a gyrus-like ridge steers a labyrinthine wave pattern along low-curvature paths until repulsion from adjacent corridors forces a transition; similar behavior is observed on real cortical gyri in Figure 5.3. The bumpy sphere simulation in Figure 5.2 further demonstrates that curvature can deflect traveling spot trajectories. While its effects on wave speed and stability remain unclear, prior work suggests that curvature can pin or disrupt waves [12]. Multispot simulations reveal curvature-modulated crowding and spot annihilation [36], raising broader questions about how surface geometry and the excitatory-inhibitory balance of the kernel interact to steer dynamics and whether a critical angle of incidence maximizes deflection. These findings motivate future reductions to effective equations and further analysis of curvature-driven pinning, transitions, and stability [14, 44].

Surface differential operators (e.g., diffusion or advection) commonly arise in neural field models with local dynamics [7, 37]. RBF-based finite difference methods provide high-order, geometry-flexible approximations of such operators [30, 41, 46, 47, 55, 57], with the same system matrix used for quadrature weights also yielding finite difference weights, offering computational savings when coupling local and nonlocal dynamics. Other mesh-free approaches, such as partition of unity methods [6, 63] and moving least squares [29, 42], share similar flexibility for irregular node layouts and could be adapted to neural field models on surfaces. We focus on RBF-based methods for their direct unification of interpolation, quadrature, and differentiation, noting that recent RBF-FD advances in stabilization and adaptive refinement [9, 26] further expand the toolkit for high-order, geometry-flexible PDE solvers.

Our method depends only on surface geometry and supports arbitrary kernels, offering a promising platform for pairing with experimental data to infer connectivity. This opens the door to data-driven modeling, model inversion, and further theoretical exploration of how cortical geometry shapes large-scale neural activity.

Code availability. The code used to generate the numerical simulations and figures can be found in the repository [www.github.com/shawsa/neural-fields-rbf](https://github.com/shawsa/neural-fields-rbf). For curved domains, the first-order approximation to the geodesic distance via the Fast Marching Algorithm [54] can be found in the MeshLib library: meshlib.io. The realistic cortical mesh was adapted from the MNE python library: mne.tools.

Reproducibility of computational results. This paper has been awarded the “SIAM Reproducibility Badge: Code and data available” as a recognition that the authors have followed reproducibility principles valued by SISC and the scientific computing community. Code and data that allow readers to reproduce the results in this paper are available at <https://github.com/shawsa/neural-fields-rbf> and in the supplementary materials (M175378_Supplementary_Materials_1.pdf [local/web 657KB]).

REFERENCES

- [1] D. ABRAHAMSEN AND B. FORNBERG, *Explicit time stepping of PDEs with local refinement in space-time*, J. Sci. Comput., 81 (2019), pp. 1945–1962, <https://doi.org/10.1007/s10915-019-01065-3>.
- [2] S. AMARI, *Dynamics of pattern formation in lateral-inhibition type neural fields*, Biol. Cybern., 27 (1977), pp. 77–87, <https://doi.org/10.1007/BF00337259>.
- [3] K. E. ATKINSON, *The Numerical Solution of Integral Equations of the Second Kind*, Cambridge Monogr. Appl. Comput. Math., Cambridge University Press, Cambridge, 1997.
- [4] K. E. ATKINSON AND W. HAN, *Theoretical Numerical Analysis: A Functional Analysis Framework*, Texts Appl. Math. 39, Springer, New York, 2005.
- [5] D. AVITABILE, *Projection methods for neural field equations*, SIAM J. Numer. Anal., 61 (2023), pp. 562–591, <https://doi.org/10.1137/21M1463768>.
- [6] I. BABUŠKA AND J. M. MELENK, *The partition of unity method*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 727–758, [https://doi.org/10.1002/\(SICI\)1097-0207\(19970228\)40:4%3C727::AID-NME86%3E3.0.CO;2-N](https://doi.org/10.1002/(SICI)1097-0207(19970228)40:4%3C727::AID-NME86%3E3.0.CO;2-N).
- [7] E. BASPINAR, D. AVITABILE, M. DESROCHES, AND M. MANTEGAZZA, *GABAergic neurons can facilitate the propagation of cortical spreading depolarization: Experiments in mouse neo-cortical slices and a novel neural field computational model*, PLoS Comput Biol., 21 (2025), e1013099, <https://doi.org/10.1371/journal.pcbi.1013099>.
- [8] V. BAYONA, N. FLYER, AND B. FORNBERG, *On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries*, J. Comput. Phys., 380 (2019), pp. 378–399, <https://doi.org/10.1016/j.jcp.2018.12.013>.
- [9] V. BAYONA, N. FLYER, B. FORNBERG, AND G. A. BARNETT, *On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs*, J. Comput. Phys., 332 (2017), pp. 257–273, <https://doi.org/10.1016/j.jcp.2016.12.008>.

- [10] V. BAYONA, A. PETRAS, C. PIRET, AND S. J. RUUTH, *A meshfree RBF-FD constant along normal method for solving PDEs on surfaces*, SIAM J. Sci. Comput., 46 (2024), pp. A3897–A3921, <https://doi.org/10.1137/23M1621265>.
- [11] M. BREAKSPEAR, *Dynamic models of large-scale brain activity*, Nat. Neurosci., 20 (2017), pp. 340–352, <https://doi.org/10.1038/nn.4497>.
- [12] P. C. BRESSLOFF, *Traveling fronts and wave propagation failure in an inhomogeneous neural network*, Phys. D, 155 (2001), pp. 83–100, [https://doi.org/10.1016/S0167-2789\(01\)00266-4](https://doi.org/10.1016/S0167-2789(01)00266-4).
- [13] P. C. BRESSLOFF, *Spatiotemporal dynamics of continuum neural fields*, J. Phys. A, 45 (2011), 033001, <https://doi.org/10.1088/1751-8113/45/3/033001>.
- [14] P. C. BRESSLOFF, *Stochastic neural field theory of wandering bumps on a sphere*, Phys. D, 399 (2019), pp. 138–152, <https://doi.org/10.1016/j.physd.2019.04.010>.
- [15] P. C. BRESSLOFF AND J. D. COWAN, *A spherical model for orientation and spatial-frequency tuning in a cortical hypercolumn*, Philos. Trans B, 358 (2003), pp. 1643–1667, <https://doi.org/10.1098/rstb.2002.1109>.
- [16] P. C. BRESSLOFF AND Z. P. KILPATRICK, *Two-dimensional bumps in piecewise smooth neural fields with synaptic depression*, SIAM J. Appl. Math., 71 (2011), pp. 379–408, <https://doi.org/10.1137/100799423>.
- [17] M. BUHMANN AND N. DYN, *Spectral convergence of multiquadric interpolation*, Proc. Edinb. Math. Soc., 36 (1993), pp. 319–333, <https://doi.org/10.1017/S0013091500018411>.
- [18] M. D. BUHMANN, *Radial basis functions*, Acta Numer., 9 (2000), pp. 1–38.
- [19] T. CECIL, J. QIAN, AND S. Osher, *Numerical methods for high dimensional Hamilton–Jacobi equations using radial basis functions*, J. Comput. Phys., 196 (2004), pp. 327–347, <https://doi.org/10.1016/j.jcp.2003.11.010>.
- [20] S. COOMBES, *Large-scale neural dynamics: Simple and complex*, Neuroimage, 52 (2010), pp. 731–739, <https://doi.org/10.1016/j.neuroimage.2010.01.045>.
- [21] S. COOMBES, P. BEIM GRABEN, R. POTTHAST, AND J. WRIGHT, *Neural Fields: Theory and Applications*, Springer, New York, 2014.
- [22] S. COOMBES AND C. LAING, *Pulsating fronts in periodically modulated neural field models*, Phys. Rev. E, 83 (2011), 011912, <https://doi.org/10.1103/PhysRevE.83.011912>.
- [23] S. COOMBES, H. SCHMIDT, AND I. BOJAK, *Interface dynamics in planar neural field models*, J. Math. Neurosci., 2 (2012), pp. 1–27, <https://doi.org/10.1186/2190-8567-2-9>.
- [24] P. C. CURTIS, *n-parameter families and best approximation*, Pacific J. Math., 9 (1959), pp. 1013–1027, <https://doi.org/10.2140/pjm.1959.9.1013>.
- [25] G. F. FASSHAUER, *Meshfree Approximation Methods with MATLAB*, World Scientific Publishing, River Edge, NJ, 2007.
- [26] N. FLYER, B. FORNBERG, V. BAYONA, AND G. A. BARNETT, *On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy*, J. Comput. Phys., 321 (2016), pp. 21–38, <https://doi.org/10.1016/j.jcp.2016.05.026>.
- [27] B. FORNBERG AND J. ZUEV, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Comput. Math. Appl., 54 (2007), pp. 379–398, <https://doi.org/10.1016/j.camwa.2007.01.028>.
- [28] S. FORTUNE, *Chapter 27: Voronoi diagrams and Delaunay triangulations*, in Handbook of Discrete and Computational Geometry, 3rd ed., J. E. Goodman, J. O’Rourke, and C. D. Tóth, eds., Chapman and Hall/CRC, Boca Raton, FL, 2017, pp. 705–721.
- [29] T.-P. FRIES AND H. G. MATTHIES, *Classification and Overview of Meshfree Methods*, Department of Mathematics, Technical University Braunschweig, Germany, 2010, https://leopard.tu-braunschweig.de/servlets/MCRFileNodeServlet/dbbs.derivate_00001418/Document.pdf.
- [30] E. J. FUSELIER AND G. B. WRIGHT, *A high-order kernel method for diffusion and reaction-diffusion equations on surfaces*, J. Sci. Comput., 56 (2013), pp. 535–565, <https://doi.org/10.1007/s10915-013-9688-x>.
- [31] J. GLAUBITZ AND J. A. REEGER, *Towards stability results for global radial basis function based quadrature formulas*, BIT, 63 (2023), <https://doi.org/10.1007/s10543-023-00956-0>.
- [32] R. L. HARDY, *Multiquadric equations of topography and other irregular surfaces*, J. Geophys. Res., 76 (1971), pp. 1905–1915, <https://doi.org/10.1029/JB076i008p01905>.
- [33] X. HUANG, W. XU, J. LIANG, K. TAKAGAKI, X. GAO, AND J.-Y. WU, *Spiral wave dynamics in neocortex*, Neuron, 68 (2010), pp. 978–990, <https://doi.org/10.1016/j.neuron.2010.11.007>.
- [34] M. J. JOHNSON, *On the Error in Surface Spline Interpolation of a Compactly Supported Function*, Manuscript, Department of Mathematics, University of Kuwait, 1998.

- [35] E. KANSA, *Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—II Solutions to parabolic, hyperbolic and elliptic partial differential equations*, Comput. Math. Appl., 19 (1990), pp. 147–161, <https://www.sciencedirect.com/science/article/pii/089812219090271K>.
- [36] Z. P. KILPATRICK AND B. ERMENTROUT, *Wandering bumps in stochastic neural fields*, SIAM J. Appl. Dyn. Syst., 12 (2013), pp. 61–94, <https://epubs.siam.org/doi/10.1137/120877106>.
- [37] F. KNEER, E. SCHÖLL, AND M. A. DAHLEM, *Nucleation of reaction-diffusion waves on curved surfaces*, New J. Phys., 16 (2014), 053010, <https://iopscience.iop.org/article/10.1088/1367-2630/16/5/053010>.
- [38] C. KUEHN AND J. M. TÖLLE, *A gradient flow formulation for the stochastic Amari neural field model*, J. Math. Biol., 79 (2019), pp. 1227–1252, <https://doi.org/10.1007/s00285-019-01393-w>.
- [39] C. R. LAING AND W. C. TROY, *PDE methods for nonlocal models*, SIAM J. Appl. Dyn. Syst., 2 (2003), pp. 487–516, <https://epubs.siam.org/doi/10.1137/030600040>.
- [40] S.-H. LEE, R. BLAKE, AND D. J. HEEGER, *Traveling waves of activity in primary visual cortex during binocular rivalry*, Nat. Neurosci., 8 (2005), pp. 22–23, <https://doi.org/10.1038/nl1365>.
- [41] E. LEHTO, V. SHANKAR, AND G. B. WRIGHT, *A radial basis function (RBF) compact finite difference (FD) scheme for reaction-diffusion equations on surfaces*, SIAM J. Sci. Comput., 39 (2017), pp. A2129–A2151, <https://epubs.siam.org/doi/10.1137/16M1095457>.
- [42] D. LEVIN, *The approximation power of moving least-squares*, Math. Comp., 67 (1998), pp. 1517–1531, <https://doi.org/10.1090/S0025-5718-98-00974-0>.
- [43] J. C. MAIRHUBER, *On Haar’s theorem concerning Chebychev approximation problems having unique solutions*, Proc. Amer. Math. Soc., 7 (1956), pp. 609–615, <https://history-of-approximation-theory.com/papers>.
- [44] R. MARTIN, D. J. CHAPPELL, N. CHUZHANOVA, AND J. J. CROFTS, *A numerical simulation of neural fields on curved geometries*, J. Comput. Neurosci., 45 (2018), pp. 133–145, <https://doi.org/10.1007/s10827-018-0697-5>.
- [45] M. R. OWEN, C. R. LAING, AND S. COOMBES, *Bumps and rings in a two-dimensional neural field: Splitting and rotational instabilities*, New J. Phys., 9 (2007), pp. 378–378, <https://doi.org/10.1088/1367-2630/9/10/378>.
- [46] A. PETRAS, L. LING, AND S. J. RUUTH, *An RBF-FD closest point method for solving pdes on surfaces*, J. Comput. Phys., 370 (2018), pp. 43–57, <https://doi.org/10.1016/j.jcp.2018.05.022>.
- [47] C. PIRET, *The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces*, J. Comput. Phys., 231 (2012), pp. 4662–4675, <https://doi.org/10.1016/j.jcp.2012.03.007>.
- [48] D. POLL AND Z. P. KILPATRICK, *Stochastic motion of bumps in planar neural fields*, SIAM J. Appl. Math., 75 (2015), pp. 1553–1577, <https://doi.org/10.1137/140999505>.
- [49] G. PÓLYA, *Über die konvergenz von quadraturverfahren*, Math. Z., 37 (1933), pp. 134, 264–286, <https://doi.org/10.1007/BF01474574>.
- [50] R. POTTHAST AND P. BEIM GRABEN, *Existence and properties of solutions for neural field equations*, Math. Methods Appl. Sci., 33 (2010), pp. 935–949, <https://doi.org/10.1002/mma.1199>.
- [51] J. REEGER, B. FORNBERG, AND M. WATTS, *Numerical quadrature over smooth, closed surfaces*, Proc. A, 472 (2016), 20160401, <https://doi.org/10.1098/rspa.2016.0401>.
- [52] J. A. REEGER AND B. FORNBERG, *Numerical quadrature over the surface of a sphere*, Stud. Appl. Math., 137 (2016), pp. 174–188, <https://doi.org/10.1111/sapm.12106>.
- [53] J. A. REEGER AND B. FORNBERG, *Numerical quadrature over smooth surfaces with boundaries*, J. Comput. Phys., 355 (2018), pp. 176–190, <https://doi.org/10.1016/j.jcp.2017.11.010>.
- [54] J. A. SETHIAN, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 1591–1595, <https://doi.org/10.1073/pnas.93.4.1591>.
- [55] V. SHANKAR, G. B. WRIGHT, R. M. KIRBY, AND A. L. FOGELSON, *A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction-diffusion equations on surfaces*, J. Sci. Comput., 63 (2015), pp. 745–768, <https://doi.org/10.1007/s10915-014-9914-1>.
- [56] S. SHAW AND Z. P. KILPATRICK, *Representing stimulus motion with waves in adaptive neural fields*, J. Comput. Neurosci., 52 (2024), pp. 145–164, <https://doi.org/10.1007/s10827-024-00869-z>.
- [57] S. B. SHAW, *Radial Basis Function Finite Difference Approximations of the Laplace-Beltrami Operator*, Master’s thesis, Boise State University, 2019.
- [58] S. SHIPP, *Structure and function of the cerebral cortex*, Curr. Biol., 17 (2007), pp. R443–R449, <https://doi.org/10.1016/j.cub.2007.03.044>.

- [59] C. SHU, H. DING, AND K. YEO, *Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier–Stokes equations*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 941–954, [https://doi.org/10.1016/S0045-7825\(02\)00618-7](https://doi.org/10.1016/S0045-7825(02)00618-7).
- [60] A. I. TOLSTYKH AND D. SHIROBOKOV, *On using radial basis functions in a “finite difference mode” with applications to elasticity problems*, Comput. Mech., 33 (2003), pp. 68–79, <https://doi.org/10.1007/s00466-003-0501-9>.
- [61] L. N. TREFETHEN, *Exactness of quadrature formulas*, SIAM Rev., 64 (2022), pp. 132–150, <https://doi.org/10.1137/20M1389522>.
- [62] S. VISSER, R. NICKS, O. FAUGERAS, AND S. COOMBES, *Standing and travelling waves in a spherical brain model: The Nunez model revisited*, Phys. D, 349 (2017), pp. 27–45, <https://doi.org/10.1016/j.physd.2017.02.017>.
- [63] H. WENDLAND, *Fast evaluation of radial basis functions: Methods based on partition of unity*, in Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt University Press, Nashville, TN, 2002, pp. 473–496, <https://www.amazon.com/Approximation-Theory-Applications-Innovations-Mathematics/dp/0826514162>.
- [64] H. WENDLAND, *Scattered Data Approximation*, Cambridge Monogr. Appl. Comput. Math., Cambridge University Press, Cambridge, 2004.
- [65] H. R. WILSON AND J. D. COWAN, *A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue*, Kybernetik, 13 (1973), pp. 55–80, <https://doi.org/10.1007/BF00288786>.
- [66] G. B. WRIGHT, *Radial Basis Function Interpolation: Numerical and Analytical Developments*, Ph.D. thesis, University of Colorado at Boulder, 2003.
- [67] J. YOON, *Spectral approximation orders of radial basis function interpolation on the Sobolev space*, SIAM J. Math. Anal., 33 (2001), pp. 946–958, <https://doi.org/10.1137/S0036141000373811>.