Journal of Scientific Computing (2023) 96:89
https://doi.org/10.1007/s10915-023-02313-3

A Meshfree Collocation Scheme for Surface Differential Operators on Point Clouds

Abhinav Singh[1,2,3] · Alejandra Foggia[1,2,3] · Pietro Incardona[1,2,3] ·
Ivo F. Sbalzarini[1,2,3,4]

Received: 12 February 2023 / Revised: 19 July 2023 / Accepted: 25 July 2023

Abstract

We present a meshfree collocation scheme to discretize intrinsic surface differential operators over scalar fields on smooth curved surfaces with given normal vectors and a non-intersecting tubular neighborhood. The method is based on discretization-corrected particle strength exchange (DC-PSE), which generalizes finite difference methods to meshfree point clouds. The proposed Surface DC-PSE method is derived from an embedding theorem, but we analytically reduce the operator kernels along surface normals to obtain a purely intrinsic computational scheme over surface point clouds. We benchmark Surface DC-PSE by discretizing the Laplace–Beltrami operator on a circle and a sphere, and we present convergence results for both explicit and implicit solvers. We then showcase the algorithm on the problem of computing Gauss and mean curvature of an ellipsoid and of the Stanford Bunny by approximating the intrinsic divergence of the normal vector field. Finally, we compare Surface DC-PSE with surface finite elements (SFEM) and diffuse-interface finite elements (DI FEM) in a validation case.

Keywords Surface differential operators · Meshfree methods · Collocation · Partial differential equations on surfaces · Intrinsic calculus

This work was funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) under grants FOR-3013 ("Vector- and tensor-valued surface PDEs") (author A.F.), GRK-1907 ("RoSI: role-based software infrastructures") (author A.S.), and SB-350008342 ("OpenPME") (author P.I.), and by the Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) under funding code 031L0160 (project "SPlaT-DM – computer simulation platform for topology-driven morphogenesis") (author I.F.S.).

✉ Ivo F. Sbalzarini
ivos@mpi-cbg.de

Abhinav Singh
absingh@mpi-cbg.de

[1] Faculty of Computer Science, Technische Universität Dresden, Dresden, Germany

[2] Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany

[3] Center for Systems Biology Dresden, Dresden, Germany

[4] Cluster of Excellence Physics of Life, TU Dresden, Dresden, Germany

Published online: 07 August 2023

Mathematics Subject Classification 65M06 · 65D25 · 35R01

1 Introduction

Partial differential equations (PDEs) on curved surfaces and differentiable manifolds are an important tool in understanding and studying physical phenomena such as surface flows [25, 36] and active morphogenesis [21]. Analytically solving intrinsic PDEs in curved surfaces, however, quickly becomes impossible for nonlinear PDEs or for surfaces that do not possess a global parameterization. Therefore, numerical methods for solving intrinsic PDEs on curved surfaces are important, and a wide variety of both embedded and embedding-free schemes have been developed to consistently discretize intrinsic differential operators over scalar fields on surfaces.

Embedding-free methods require a (at least local) parameterization of the surface in order to discretize the differential operators via coordinate charts or a local basis of the manifold [37]. This includes methods based on local moving frames [6], a concept originally developed in continuous group theory, where the surface geometry is locally represented by intrinsic orthonormal bases. This has been used to solve surface PDEs over meshfree point clouds using moving least squares (MLS) approximations [19]. The concept of local moving frames has also been combined with discontinuous Galerkin discretization, e.g., to solve shallow-water equations on arbitrary rotating surfaces [7]. Other embedding-free Finite Element Methods (FEM) include intrinsic surface FEM (ISFEM), which discretizes differential operators on a triangulation of the surface [2, 14], and methods based on Discrete Exterior Calculus (DEC) [24].

Embedding methods discretize the surface problem in an embedding space of co-dimension 1 and use projections to restrict the differential operators computed in the embedding space to the surface manifold. This includes methods that use explicit tracer points to represent the surface, but interpolate to an embedding mesh to evaluate differential operators [18], diffuse-interface methods based on phase-field representations of the surface [23], embedding FEM such as TraceFEM [26] and diffuse-interface FEM [17, 30], narrow-band level-set methods based on orthogonal extension of the surface quantities [4, 9], level-set methods based on the closest-point transform [20, 29], and volume-of-fluid methods for surface PDE problems [16].

While each of these methods has its specific strengths, embedding methods usually generalize better to complex-shaped or arbitrary surfaces [29]. However, they tend to have higher computational cost, because computations are done in the higher-dimensional embedding space and additional extension (for level sets), right-hand-side evaluation (for phase fields), or interpolation (for closest-point transforms) steps are required, albeit specific optimizations are available, e.g., for level sets [27]. Embedding-free methods are usually more accurate, because they avoid the interpolation and projection errors arising when the discretization of the embedding space does not trace the surface exactly, but they tend to be more difficult to implement and harder to generalize to complex-shaped or moving/deforming surfaces.

Here, we present a meshfree collocation method for PDEs on smooth and orientable curved surfaces with non-intersecting tubular neighborhood. The method combines elements from embedding and embedding-free approaches. It is *algorithmically* embedding-free in the sense that surface quantities are represented on tracer points that are contained in the surface. This also discretizes and represents the surface itself as a point cloud. But the method is *mathematically* related to embedding approaches, since the stencils used to approximate differential operators at the surface points are computed in the embedding space by a reduction operation along the local normal vector, which needs to be known or computed. Intuitively, this projects the discrete operators, rather than projecting the flux vectors as typically done in embedding methods. The resulting method therefore shares properties of moving frame approaches, such as the low dimensionality (and hence low computational cost) and the meshfree character [6, 19]. It combines these with properties of embedding methods, such as their flexibility in generalizing to complex surfaces [29], and their ability to compute extrinsic differential-geometric quantities.

Our method is based on the Discretization-Corrected Particle Strength Exchange (DC-PSE) collocation scheme for arbitrary (surface) point clouds. DC-PSE is related to Generalized Finite Difference Methods (GFDM) [35] and to MLS [32]. Given the local surface normal $\boldsymbol{n}$, we derive intrinsic discrete operators by first creating an embedding narrow-band and placing collocation points along the normal from each surface point. We then determine the regular DC-PSE operator kernels in the embedding space. These kernels are subsequently reduced under the condition of orthogonal extension $\nabla f \cdot \boldsymbol{n} = 0$ for any (sufficiently) differentiable scalar field $f(\boldsymbol{x}) \in \mathbb{R}$ to derive intrinsic kernels at the surface points $\boldsymbol{x}$. This is possible due to the kernel nature of DC-PSE, and it preserves the information from the embedding space in a scheme that only requires computation over surface points.

This paper is organized as follows: Sect. 2 recollects the DC-PSE method for convenience and introduces the notation. In Sect. 3, we describe the Surface DC-PSE scheme for numerically consistent discretization of surface differential operators. We present validation and convergence result in Sect. 4 and conclude in Sect. 5.

2 Discretization-Corrected Particle Strength Exchange (DC-PSE)

DC-PSE is a numerical method for discretizing differential operators on irregular distributions of collocation points [32]. The method was originally derived as an improvement over the classic Particle Strength Exchange (PSE) [12] scheme, reducing its quadrature error on irregularly distributed collocation points, but mathematically amounts to a generalization of finite differences [32]. The PSE/DC-PSE class of collocation methods uses mollification with a symmetric smoothing kernel $\eta_\epsilon(\cdot)$ to approximate (sufficiently smooth) continuous functions $f(\boldsymbol{x}) \in \mathbb{R}$, $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d$,

$$f(\boldsymbol{x}_p) \approx f_\epsilon(\boldsymbol{x}_p) = \int_\Omega f(\boldsymbol{x})\, \eta_\epsilon(\boldsymbol{x}_p - \boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \tag{1}$$

where $f_\epsilon(\boldsymbol{x}_p)$ is a regularized approximation of the function f at location $\boldsymbol{x}_p \in \Omega$ of collocation point p. The scalar ϵ is the smoothing length (or the kernel width) of the mollification. Linear differential operators in $\mathbb{R}^d$,

$$\boldsymbol{D}^{\boldsymbol{\alpha}} = \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_d^{\alpha_d}}, \tag{2}$$

defined by the multi-index $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{Z}^d$ with $|\boldsymbol{\alpha}| = \sum_{i=1}^d \alpha_i$ are approximated by Taylor series expansion to find a discrete operator

$$\boldsymbol{Q}^{\boldsymbol{\alpha}} f(\boldsymbol{x}_p) = \boldsymbol{D}^{\boldsymbol{\alpha}} f(\boldsymbol{x}_p) + O\big(h(\boldsymbol{x}_p)^r\big) \tag{3}$$

at collocation point $\boldsymbol{x}_p$. The order of approximation r depends on the kernel η_ϵ used in Eq. (1), and $h(\boldsymbol{x}_p)$ is the average distance between collocation point p and its neighbors

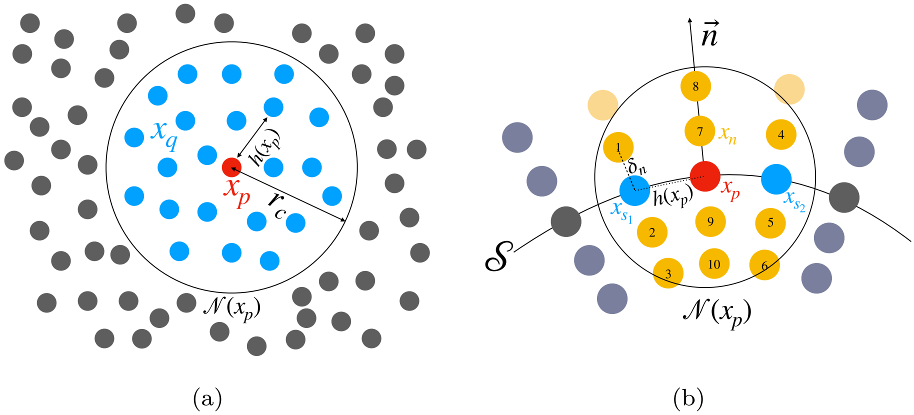


Fig. 1 **a** Illustration of the DC-PSE method. The collocation points $\boldsymbol{x}_q$ (blue) within the symmetric operator support $\mathcal{N}(\boldsymbol{x}_p)$ of radius r_c around the center point $\boldsymbol{x}_p$ (red) are used to approximate the differential operator at $\boldsymbol{x}_p$. The average distance between points in the operator support, $h(\boldsymbol{x}_p)$ defines the accuracy of the approximation. **b** Illustration of the Surface DC-PSE method. The intrinsic differential operator at a point $\boldsymbol{x}_p$ (red) on the surface $\mathcal{S}$ is evaluated over neighboring surface points $\boldsymbol{x}_s$ (blue). The operator kernel is constructed in the tubular neighborhood of radius r_c (circle) with $2N_n$ points $\boldsymbol{x}_n$ (yellow) replicated along the local surface normal $\boldsymbol{n}$ at distances δn. Point labels are used in the main text for the illustrative example (Color figure online)

within the kernel support. We use the arithmetic mean of the L_1-distances to compute h, but since all norms are equivalent, the convergence order (but not the actual error magnitude) is independent of the choice of average. The Taylor expansion yields integral constraints (also known as *continuous moment conditions*), which the kernel η_ϵ needs to fulfill in order to reach a certain convergence order r [12].

DC-PSE uses different kernels $\eta_\epsilon^p(\cdot,\cdot)$ for different collocation points p and directly acts on a given quadrature of Eq. (1) with collocation points $\boldsymbol{x}_q \in \Omega$, resulting in the discrete operator:

$$\boldsymbol{Q}_h^{\boldsymbol{\alpha}} f(\boldsymbol{x}_p) = \frac{1}{\epsilon(\boldsymbol{x}_p)^{|\boldsymbol{\alpha}|}} \sum_{\boldsymbol{x}_q \in \mathcal{N}(\boldsymbol{x}_p)} \left(f(\boldsymbol{x}_q) \pm f(\boldsymbol{x}_p)\right) \eta_\epsilon^p(\boldsymbol{x}_p, \boldsymbol{x}_q), \tag{4}$$

where $\mathcal{N}(\boldsymbol{x}_p)$ are all collocation points in the neighborhood (of a certain radius r_c defined by the kernel width) around point $\boldsymbol{x}_p$, as illustrated in Fig. 1a. The positive sign in the parenthesis is used for odd $|\boldsymbol{\alpha}|$, the negative sign for even $|\boldsymbol{\alpha}|$. This renders the operator conservative on symmetric collocation point distributions, i.e., when $\eta_\epsilon^p(\boldsymbol{x}_p, \boldsymbol{x}_q) = \eta_\epsilon^q(\boldsymbol{x}_q, \boldsymbol{x}_p)$. In DC-PSE, the kernels η_ϵ^p are thus not determined from continuous moment conditions, as in PSE, but directly from the *discrete moment conditions* that result from substituting Eq. (4) into the quadrature of Eq. (1) [32] for a given set $\{\boldsymbol{x}_q\}_{q=1}^N$. This adapts the kernels to the specific distribution of collocation points (hence the name "discretization-corrected") and avoids the quadrature error of PSE [12], leading to a scheme that is consistent with order r on almost[1] arbitrary collocation point sets. This means that at each collocation point, a potentially different kernel is used for the same differential operator if the neighboring collocation points within the kernel support are distributed differently. Evaluating such a kernel at the locations

[1] The collocation point distribution must not be degenerate in the sense that the Vandermonde matrix of the kernel system must have full rank [5]. A trivial example: placing all points along a line and then asking for an approximation of the derivative in the perpendicular direction cannot work.

of the collocation points yields a generalized finite-difference stencil, which reduces to the classic compact finite differences on regular grid arrangements of points [32].

DC-PSE kernels are determined at runtime by solving a small system of linear equations for each collocation point, resulting from the discrete moment conditions in its kernel neighborhood. For this, one can choose the function space such that the kernels are compact and symmetric. A frequent choice are polynomials windowed by truncated exponentials [33]

$$\eta_\epsilon^p(\boldsymbol{x}_p, \boldsymbol{x}_q) = \eta_\epsilon^p\left(\frac{\boldsymbol{x}_p - \boldsymbol{x}_q}{\epsilon(\boldsymbol{x}_p)}\right) := \left(\sum_{|\gamma|=\beta_{\min}}^{|\boldsymbol{\alpha}|+r-1} a_\gamma(\boldsymbol{x}_p)\left(\frac{\boldsymbol{x}_p - \boldsymbol{x}_q}{\epsilon(\boldsymbol{x}_p)}\right)^\gamma\right) \mathrm{e}^{-\left|\frac{\boldsymbol{x}_p-\boldsymbol{x}_q}{\epsilon(\boldsymbol{x}_p)}\right|^2} \tag{5}$$

of finite radius r_c. The polynomial coefficients a_γ are determined for a given $\boldsymbol{\alpha}$ and given collocation points $\boldsymbol{x}_q \in \mathcal{N}(\boldsymbol{x}_p)$, such that the following discrete moment conditions are satisfied:

$$Z_h^{\boldsymbol{\beta}} = \begin{cases} (-1)^{|\boldsymbol{\alpha}|}\boldsymbol{\alpha}!, & \boldsymbol{\beta} = \boldsymbol{\alpha} \\ 0, & \boldsymbol{\beta} \neq \boldsymbol{\alpha}, \quad \beta_{\min} \leq |\boldsymbol{\beta}| \leq |\boldsymbol{\alpha}| + r - 1, \ \beta_{\min} = \begin{cases} 0, & |\boldsymbol{\alpha}| \text{ odd} \\ 1, & |\boldsymbol{\alpha}| \text{ even} \end{cases} \\ < \infty, & |\boldsymbol{\beta}| = |\boldsymbol{\alpha}| + r \end{cases} \tag{6}$$

where

$$Z_h^{\boldsymbol{\beta}}(\boldsymbol{x}) = \frac{1}{\epsilon(\boldsymbol{x}_p)^d} \sum_{\boldsymbol{x}_q \in \mathcal{N}(\boldsymbol{x}_p)} \frac{(\boldsymbol{x}_p - \boldsymbol{x}_q)^{\boldsymbol{\beta}}}{\epsilon(\boldsymbol{x}_p)^{|\boldsymbol{\beta}|}} \eta_\epsilon^p\left(\frac{\boldsymbol{x}_p - \boldsymbol{x}_q}{\epsilon(\boldsymbol{x}_p)}\right) \tag{7}$$

is the discrete moment of order $\boldsymbol{\beta}$ of the kernel $\eta_\epsilon^{\boldsymbol{\alpha}}$, and $\beta_{\min}$ is the parity of $|\boldsymbol{\alpha}|$, because the zeroth moment $Z_h^{\mathbf{0}}$ vanishes for even operators. Under these conditions, DC-PSE is consistent with order r as long as

$$\frac{h(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)} \in O(1), \tag{8}$$

i.e., the kernel width ϵ scales proportionally with the average inter-point distance h around $\boldsymbol{x}_p$ [32].

3 Surface DC-PSE

We generalize DC-PSE to surface differential operators based on the following classic result [22, 29]: Let $\mathcal{S} \subset \mathbb{R}^d$ be a differentiable manifold that possesses a tubular neighborhood T and is orientable[2] and $f : \mathcal{S} \to \mathbb{R}$. Define $F : T \to \mathbb{R}$, such that the restriction $F|_{\mathcal{S}} = f$, and F is constant along the normal direction $\boldsymbol{n}$ of $\mathcal{S}$, i.e., $\nabla F \cdot \boldsymbol{n} = 0$. Then, on the surface $\mathcal{S}$,

$$\nabla_{\mathcal{S}} f = (\nabla F)|_{\mathcal{S}}, \tag{9}$$

where $\nabla_{\mathcal{S}} f$ is the intrinsic surface gradient. A similar result is true for the intrinsic divergence operator $(\nabla_{\mathcal{S}} \cdot\ )$ and for tangential vector field that is extended by constant extension to all surfaces displaced along the normal of $\mathcal{S}$ [22, 29].

Given this result, it is straightforward to see the advantages of a meshfree discretization: it allows for conforming discretization of the surface and for *exact* constant orthogonal extension by simply copying points along the normal. This creates an embedding narrowband of *exact* closest-point function values within the tubular neighborhood T without a

[2] Every boundary-less smooth surface embedded in $\mathbb{R}^d$ has a tubular neighborhood, and the orientability condition is not restrictive when considered locally [22].

need for interpolation. If T is non-intersecting with a radius of at least r_c everywhere, the conditions of the result in Eq. (9) are satisfied in the constructed embedding. Analogous to the closest-point method [29], one can then discretize differential operators in the embedding space. Due to the additive kernel nature of DC-PSE, the discrete operators in the embedding space can be reduced to only the surface points.

This reduction becomes clear from the formulation of the DC-PSE method. Indeed, we realize that the constant normal extension can be made internal to the operator evaluation by accumulating the kernel *coefficients* along the normals. To see this, consider the DC-PSE operator in Eq. (4) in the embedding space. The neighborhood $\mathcal{N}$ for the summation contains both surface points $\boldsymbol{x}_s$ and normally extended points $\boldsymbol{x}_n$, as shown in Fig. 1b. Because the $f(\boldsymbol{x}_n)$ are identical copies of the values of the respective surface points, we note that the prefactors $\big(f(\boldsymbol{x}_s) \pm f(\boldsymbol{x}_p)\big)$ in the summation of Eq. (4) are the same for all extended normal points and the corresponding surface point $\boldsymbol{x}_s$. Hence, for each given pair of a center point $\boldsymbol{x}_p$ and another surface point $\boldsymbol{x}_s$, the interactions with the corresponding normally extended points can be factored out from the kernel summation:

$$\begin{aligned} &\frac{f(\boldsymbol{x}_s) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)^{|\alpha|}} \sum_{\boldsymbol{x}_q=\{\boldsymbol{x}_s,\boldsymbol{x}_n:(\boldsymbol{x}_n-\boldsymbol{x}_s)||\boldsymbol{n}(\boldsymbol{x}_s)\}} \eta_\epsilon^p\left(\frac{\boldsymbol{x}_p-\boldsymbol{x}_q}{\epsilon\left(\boldsymbol{x}_p\right)}\right) \\ &= \frac{f(\boldsymbol{x}_s) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)^{|\alpha|}} \eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_s), \end{aligned} \tag{10}$$

defining the surface kernels $\eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_s)$. These can be evaluated over only the surface points $\boldsymbol{x}_s = \mathcal{N}_{\mathcal{S}}(\boldsymbol{x}_p)$ in the in-surface neighborhood $\mathcal{N}_{\mathcal{S}}(\boldsymbol{x}_p)$ around the surface point $\boldsymbol{x}_p$, see Fig. 1b, yielding the Surface DC-PSE operator:

$$\boldsymbol{Q}_{\mathcal{S}}^{\boldsymbol{\alpha}} f(\boldsymbol{x}_p) = \frac{1}{\epsilon(\boldsymbol{x}_p)^{|\alpha|}} \sum_{\boldsymbol{x}_s\in\mathcal{N}_{\mathcal{S}}(\boldsymbol{x}_p)} \big(f(\boldsymbol{x}_s) \pm f(\boldsymbol{x}_p)\big)\, \eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_s). \tag{11}$$

Importantly, the surface kernels $\eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_s)$, summed over all orthogonally extended points, can directly be computed when determining the kernel weights and without explicitly creating or storing the normally extended points $\boldsymbol{x}_n$.

Evaluating a Surface DC-PSE operator involves only the neighboring points on the surface and requires no narrow band or normally extended grid, even though the construction of the operators uses an embedding. This leads to a corresponding reduction in computational complexity for operator evaluation, as computations are only performed on a $(d-1)$-dimensional surface embedded in d-dimensional space. In comparison, the cost of operator evaluation for embedding methods such as the closest-point method is $O(k(d-1))$, where $k > 1$ is the narrow-band width, which scales proportionally with the order of convergence.

3.1 Surface DC-PSE Kernel Construction

Surface DC-PSE requires two algorithms that are not part of the standard, flat-space DC-PSE method: an algorithm to create the intrinsic neighborhood of a surface point p, and an algorithm to determine the surface kernels $\eta_{\mathcal{S}}$ at a surface point p. We follow the example of Ref. [5] and use explicit component notation and a concrete example in order to directly relate to implementations in computer code.

Figure 1b illustrates a piece of the tubular neighborhood of radius r_c (circle) of a curved surface $\mathcal{S}$ embedded in $\mathbb{R}^2$. The red point p at position $\boldsymbol{x}_p$ on the surface is the "center" collocation point at which we derive the discrete Surface DC-PSE operator $\boldsymbol{Q}_{\mathcal{S}}^{\boldsymbol{\alpha}} f(\boldsymbol{x}_p)$ for

Algorithm 1 Surface DC-PSE: construction of neighborhood of a point p.

Input:

1. Point set P on the surface $\mathcal{S}$
2. Cutoff radius for the operator support r_c
3. Indices $\mathcal{N}_\mathcal{S}$ of surface points in the neighborhood of p
4. Optional: spacing δn between the normally extended points. Default: average embedding-space distance between surface points
5. Optional: Number of normal copies of each surface point to be used during operator construction N_n (symmetric to either sides of the surface). Default: $N_n = \text{round}(r_c/\delta n)$

Output: List of distances between point p and all surface s_i and normal n_i points in its neighborhood $\mathcal{N}(\boldsymbol{x}_p)$: $\mathcal{N}_{\text{dist}}(\boldsymbol{x}_p)$

```
Require: $|\boldsymbol{n}_{s_i}| = |\boldsymbol{n}_p| = 1$
$\mathcal{N}_{\text{dist}} = [\,]$, $k = 0$
for all $s_i \in \mathcal{N}_\mathcal{S}$ do
    $\mathcal{N}_{\text{dist}}$.append([ ])
    for $i \in [-N_n, N_n]$ do
        $\boldsymbol{d}_{n_i} = \boldsymbol{x}_p - \boldsymbol{x}_{s_i} - i\delta n \cdot \boldsymbol{n}_{s_i}$
        if $|\boldsymbol{d}_{n_i}| \leq r_c$ then
            $\mathcal{N}_{\text{dist}}[k]$.append($\boldsymbol{d}_{n_i}/\epsilon_p$)        ▷ Add to set of the corresponding $s_i$.
        end if
    end for
    $k \mathrel{+}= 1$
end for

$\mathcal{N}_{\text{dist}}$.append([ ])
for $i \in [-N_n, N_n]\backslash\{0\}$ do        ▷ Normal points to $p$.
    $\boldsymbol{d}_{n_i} = -i\delta n \cdot \boldsymbol{n}_p$
    if $|\boldsymbol{d}_{n_i}| \leq r_c$ then
        $\mathcal{N}_{\text{dist}}[k]$.append($\boldsymbol{d}_{n_i}/\epsilon_p$)        ▷ Create a new set containing all points along $\boldsymbol{n}_p$.
    end if
end for
```

a scalar surface field f. Points in light blue are surface points within the embedding-space neighborhood of radius r_c (circle) of $\boldsymbol{x}_p$, and the yellow points are the orthogonal extensions $\boldsymbol{x}_n$. By default, the spatial separation δn between adjacent orthogonal extensions of the surface point p is the arithmetic mean of the distances between p and the other surface points within the kernel support, measured in the embedding space. This favors isotropic-resolution neighborhoods and, thus, low condition numbers of the DC-PSE kernel system matrix. The number of orthogonally extended points should be $N_n \approx \text{round}(r_c/\delta n)$ to either side of the surface, which is the default. Only surface points are actually allocated and stored.

In order to determine the DC-PSE kernel η_ϵ^p in the embedding space, the distances between $\boldsymbol{x}_p$ and all collocation points in its embedding-space neighborhood $\mathcal{N}$ are required. In the example of Fig. 1b, the neighborhood (circle) includes the surface points $\mathcal{N}_\mathcal{S} = \{s_1, s_2\}$ and the normally extended points $\{n_i\}_{i=1}^{10}$. The two pale-yellow points are not part of the neighborhood. Algorithm 1 constructs the neighbor set along with the corresponding distances. Surface normals at a given point are indexed by the point index for better readability, i.e., $\boldsymbol{n}(\boldsymbol{x}_p) := \boldsymbol{n}_p$. In the example of the figure, this results in the output

$$\mathcal{N}_{\text{dist}}(\boldsymbol{x}_p) = [[\boldsymbol{d}_{s_1}, \boldsymbol{d}_{n_1}, \boldsymbol{d}_{n_2}, \boldsymbol{d}_{n3}], [\boldsymbol{d}_{s_2}, \boldsymbol{d}_{n_4}, \boldsymbol{d}_{n_5}, \boldsymbol{d}_{n_6}], [\boldsymbol{d}_{n_7}, \boldsymbol{d}_{n_8}, \boldsymbol{d}_{n_9}, \boldsymbol{d}_{n_{10}}]],$$

where $\boldsymbol{d}_q$ is the distance between the collocation points p and q in the embedding space in units of $\epsilon_p := \epsilon(\boldsymbol{x}_p)$.

Using this neighborhood data structure, the embedding-space DC-PSE operator at point p in the example of the figure reads:

$$\begin{aligned} \boldsymbol{Q}^{\boldsymbol{\alpha}}_{\mathcal{S}} f(\boldsymbol{x}_p) &= \frac{f(\boldsymbol{x}_{s_1}) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)} \left(\eta^p_\epsilon(\boldsymbol{d}_{s_1}) + \eta^p_\epsilon(\boldsymbol{d}_{n_1}) + \eta^p_\epsilon(\boldsymbol{d}_{n_2}) + \eta^p_\epsilon(\boldsymbol{d}_{n_3})\right) \\ &+ \frac{f(\boldsymbol{x}_{s_2}) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)} \left(\eta^p_\epsilon(\boldsymbol{d}_{s_2}) + \eta^p_\epsilon(\boldsymbol{d}_{n_4}) + \eta^p_\epsilon(\boldsymbol{d}_{n_5}) + \eta^p_\epsilon(\boldsymbol{d}_{n_6})\right) \\ &+ \frac{f(\boldsymbol{x}_p) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)} \left(\eta^p_\epsilon(\boldsymbol{d}_{n_7}) + \eta^p_\epsilon(\boldsymbol{d}_{n_8}) + \eta^p_\epsilon(\boldsymbol{d}_{n_9}) + \eta^p_\epsilon(\boldsymbol{d}_{n_{10}})\right). \end{aligned} \tag{12}$$

Since the embedding-space kernels are evaluated at concrete distances, the $\eta^p_\epsilon(\boldsymbol{d}_q)$ are just scalar numbers. All kernel values that share the same pre-factor $\frac{f(\boldsymbol{x}_q) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)}$ are thus summed to the surface kernels $\eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_q)$, $q \in \mathcal{N}_{\mathcal{S}}$, obtaining:

$$\begin{aligned} \boldsymbol{Q}^{\boldsymbol{\alpha}}_{\mathcal{S}} f(\boldsymbol{x}_p) &= \frac{f(\boldsymbol{x}_{s_1}) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)} \eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_{s_1}) + \frac{f(\boldsymbol{x}_{s_2}) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)} \eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_{s_2}) \\ &+ \frac{f(\boldsymbol{x}_p) \pm f(\boldsymbol{x}_p)}{\epsilon(\boldsymbol{x}_p)} \eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_p). \end{aligned} \tag{13}$$

For even differential operators, i.e., derivatives with even $|\boldsymbol{\alpha}|$, the third term vanishes identically and can be skipped in the calculations. But this is not the case for odd-order derivatives. With this rearrangement, each evaluation of the operator at a point p only requires three kernel evaluations instead of the 12 that would be required in the embedding case. In addition, the normally extended points never need to be allocated and stored, as all kernel computations can happen on the fly. Algorithm 2 details the procedure for Surface DC-PSE operator construction. For the example from Fig. 1b, this results in the surface DC-PSE kernel values:

$$\mathcal{K}_{\mathcal{S}} = [\eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_{s_1}),\ \eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_{s_2}),\ \eta_{\mathcal{S}}(\boldsymbol{x}_p, \boldsymbol{x}_p)],$$

which can directly be used in Eq. (11) to evaluate surface differential operators.

Algorithm 2 Surface DC-PSE: construction of surface kernel at a point p.

Input:

1. List of neighbor distances $\mathcal{N}_{\text{dist}}$ for each pair $\boldsymbol{x}_{p,q}$, with $q \in \mathcal{N}(\boldsymbol{x}_p)$, as constructed by Algorithm 1

Output: Surface kernel values for each pair $\boldsymbol{x}_{p,s}$, with $s \in \mathcal{N}_{\mathcal{S}}$: $\mathcal{K}_{\mathcal{S}}$

```
$\mathcal{K}_{\mathcal{S}} = zeros(size(\mathcal{N}_{\mathcal{S}}) + 1)$
$\eta = \text{DCPSE}(p)$                                   ▷ Determine embedding-space DC-PSE kernel at $p$
for all $(i, j) \in \mathcal{N}_{\text{dist}}$ do
    $\mathcal{K}_{\text{emb}}[i][j] = \eta(\mathcal{N}_{\text{dist}}[i][j])$
end for
for $i \in [1, size(\mathcal{N}_{\mathcal{S}}) + 1]$ do
    for all $j \in \mathcal{K}_{\text{emb}}[i]$ do
        $\mathcal{K}_{\mathcal{S}}[i]$ += $\mathcal{K}_{\text{emb}}[i][j]$
    end for
end for
```

4 Results

We validate and benchmark the Surface DC-PSE method. First, we verify its convergence in test cases with known analytical solution. Then, we show applications to cases with more general surfaces where no analytical solution is available. Finally, we compare with surface finite-element methods (SFEM) in a validation study.

In all cases, orthogonal extension is exact, copying surface points along the known normals. Therefore, $\nabla F \cdot \boldsymbol{n}$ in Eq. 9 is always zero by construction. Numerically evaluating this term requires approximating the gradient ∇F, which we confirmed to converge with the order of accuracy of the discretization scheme used.

4.1 Laplace–Beltrami Operator on a Circle and a Sphere

We start by verifying convergence for the Laplace–Beltrami operator on the unit circle S^1. The collocation points are distributed regularly using equi-angular spacing. We use a normal spacing of $\delta n = 3/(N_p - 1)$ to compute the surface operators in Eq. (11) in a narrow band with $N_n = 4$ layers on each side of the surface. N_p is the number of surface points $\boldsymbol{x}_s$. We choose $r_c = 4.1\delta n$ as the operator support and $r = 2$ as the desired order of convergence. The Laplace–Beltrami operator is characterized by the multi-index $\boldsymbol{\alpha} = (2, 0) + (0, 2)$. Note that this multi-index is 2-dimensional, despite the circle being one-dimensional, since the operators are constructed in the embedding space, but evaluated intrinsically.

We test the numerical approximation of the surface operator on the function

$$f(\theta) = \sin(\theta) + \cos(\theta) \tag{14}$$

in polar coordinates. The error is computed against the analytical solution

$$\Delta_{\mathcal{S}} f(\theta) = \nabla_{\mathcal{S}} \cdot (\nabla_{\mathcal{S}} f(\theta)) = \nabla_{\theta}^2 f(\theta) = -(\sin(\theta) + \cos(\theta)). \tag{15}$$

The visualization of the numerical solution and the convergence plot of the absolute errors are shown in Fig. 2a, c. We observe second-order convergence to the analytical solution, as expected for $r = 2$.

We further test the method on the unit sphere S^2. The collocation points are distributed using the Fibonacci sphere technique [13]. We use a normal spacing of $\delta n = 0.8/(\sqrt[3]{N_p} - 1)$ to determine the surface operators in Eq. (11) in a narrow band with $N_n = 2$ layers on each side of the surface. N_p is the number of points on the sphere. We choose $r_c = 2.9\delta n$ as the operator support and $r = 2$ and $r = 4$ as the desired orders of convergence. The Laplace–Beltrami operator is characterized by the three-dimensional multi-index $\boldsymbol{\alpha} = (2, 0, 0) + (0, 2, 0) + (0, 0, 2)$. We test the numerical approximation of the surface operator on the scalar spherical harmonic function

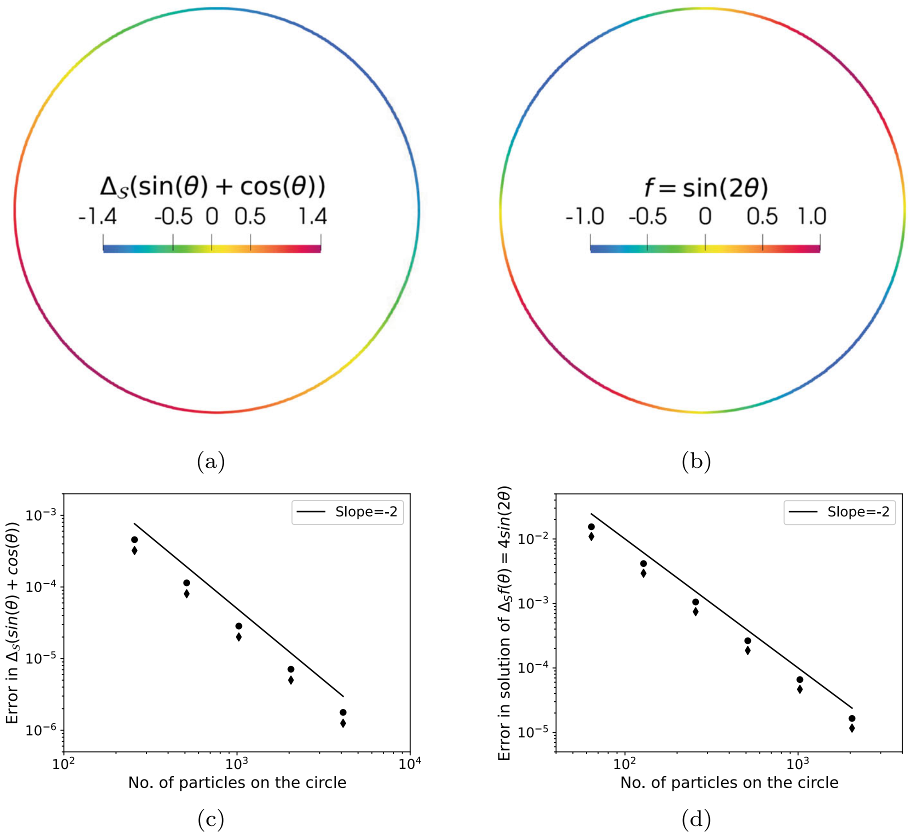
$$f(\theta, \phi) = Y_{l,m} \tag{16}$$

in spherical coordinates for the mode $l = 4$, $m = 0$ (Fig. 3a). The error is computed against the analytical solution

$$\Delta_{\mathcal{S}} f = \nabla_{\mathcal{S}} \cdot (\nabla_{\mathcal{S}} f(\theta, \phi)) = -l(l + 1)Y_{l,m} \tag{17}$$

and is plotted in Fig. 3c.

We also use this test case to benchmark against the Closest Point (CP) method [29] with L_2 and L_∞ errors plotted in Fig. 3c. While Surface DC-PSE is less accurate than the CP



(a)

(b)

(c)

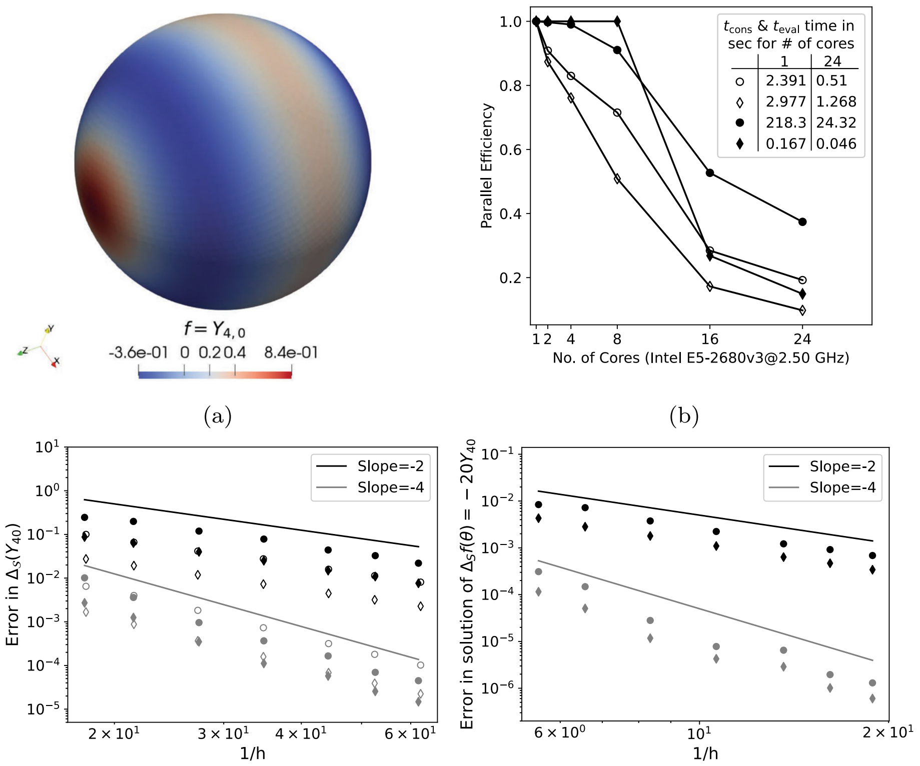
(d)

Fig. 2 Visualization and convergence of the Laplace–Beltrami operator on the unit circle. **a** Visualization of $\Delta_{\mathcal{S}}(\sin(\theta) + \cos(\theta))$ computed using second-order accurate Surface DC-PSE operators. **b** Visualization of the solution of the Poisson equation $\Delta_{\mathcal{S}} f = 4\sin(2\theta)$ solved using second-order accurate Surface DC-PSE operators. **c** Convergence plot of the Laplace–Beltrami operator in (**a**). L_∞(●) and L_2(◆) norms of the absolute errors are computed against the analytical solution in Eq. (15) for increasing numbers of points on the circle. **d** Convergence plot of the Poisson equation solution in (**b**). L_∞(●) and L_2(◆) norms of the absolute errors are computed against the analytical solution in Eq. (19) for increasing numbers of points on the circle

method for second-order operators, it outperforms for fourth-order operators, which is likely due to the better condition numbers of the small linear systems to be solved for each point.

Finally, we perform a strong scaling benchmark of the computation time with increasing numbers of CPU cores with both codes implemented in the parallel computing library OpenFPM [15, 34] in C++ and run on the same hardware. We plot the parallel efficiency (i.e., the speed-up divided by the number of cores) in Fig. 3b for both the construction and the evaluation of the operators of both methods. We further report the absolute wall-clock times on one and 24 cores in the inset table. These times show that evaluating the Surface DC-PSE operators over all points in the domain is about one order of magnitude faster than evaluating the CP transform [29]. In addition, Surface DC-PSE scales better with increasing numbers of parallel CPU cores. This is due to the simpler kernel evaluation of DC-PSE.[3]

[3] Up to 8 cores, the measured parallel efficiency of Surface DC-PSE was larger than 1.0, due to cache effects when storing the kernel coefficients.



(a)

(b)

(c)

(d)

Fig. 3 Visualization and convergence of the Laplace–Beltrami operator on the unit sphere. **a** Visualization of the spherical harmonic function $Y_{4,0}$. **b** Results of a strong scalability test for the computation of $\Delta_{\mathcal{S}} Y_{4,0}$ for average spacing $h = 0.05$ using Surface DC-PSE operators (•, ◆) and the Closest Point (CP) method (∘, ◇) [29] on increasing numbers of CPU cores. We separately show the parallel efficiency for the construction of Surface DC-PSE operators (•) and of the CP representation (∘), and for their evaluation to approximate $\Delta_{\mathcal{S}} Y_{4,0}$ across the entire domain using Surface DC-PSE (◆) and the CP method (◇). The absolute wall-clock times for one time step are shown in the inset table for 1 and 24 cores. **c** Convergence plot of $\Delta_{\mathcal{S}} Y_{4,0}$ for Surface DC-PSE (L_∞(•), L_2(◆)) and the CP method (L_∞(∘), L_2(◇)) using second-order (black) and fourth-order (gray) approximations. Norms of the absolute errors are computed against the analytical solution in Eq. (17) for increasing numbers of points (decreasing average spacing h). **d** Convergence plot for the solution of the Poisson equation $\Delta_{\mathcal{S}} f = -20Y_{4,0}$ across the entire domain using second-order (black) and fourth-order (gray) operators. L_∞(•) and L_2(◆) norms of the absolute errors are computed against the analytical solution in Eq. (21) for increasing numbers of surface points

Eventually, the efficiency for both methods drops, as is expected for strong scaling with constant communication overhead. The time required to construct the Surface DC-PSE operators with $N_n = 2$, however, is about two orders of magnitude larger than that for constructing the CP representation in a narrow band of radius 5 as required by the regression support. However, it still scales better with increasing CPU core count. For Eulerian simulations, where collocation points do not move, the kernels have to be determined only once at the beginning of a simulation, or they can be loaded from files for standard point distributions, potentially leading to an overall lower computational cost of Surface DC-PSE.

4.2 Poisson Equation on a Circle and a Sphere

Surface DC-PSE operators can also be used for implicit equations by solving a linear system of equations with a system matrix constructed using the Surface DC-PSE operators. We test this by solving the Poisson equation on the unit circle S^1:

$$\Delta_{\mathcal{S}} f = 4\sin(2\theta) \qquad \theta \in \Omega = S^1 \backslash (1, 0) \tag{18}$$

with Dirichlet boundary condition at one point $(1, 0)$ conforming to the analytical solution

$$f(\theta) = \sin(2\theta) \qquad \theta \in \Omega \cup (1, 0). \tag{19}$$

We use the same Surface DC-PSE operators as in the previous subsection to construct the system of equations, which is then solved using the KSPGMRES solver from PETSc [3]. Figure 2b, d show the solution f and the convergence plot of the absolute error with respect to the analytical solution.

Next, we test the method in three dimensions by solving the Poisson equation on the sphere S^2:

$$\Delta_{\mathcal{S}} f = -20Y_{4,0} \tag{20}$$

with Dirichlet boundary condition along the great circle parallel to the $y-z$ plane conforming to the analytical solution

$$f = Y_{4,0}. \tag{21}$$

We solve the resulting linear system with KSPGMRES from PETSc [3] without preconditioning. The convergence plots for orders $r = 2$ and $r = 4$ are shown in Fig. 3d. The collocation points on the sphere were constructed using the Fibonacci sphere technique [13]. While this generates pseudo-regular point distributions on the sphere, their average spacing does fluctuate a bit, explaining the slight waviness of the curve especially for the fourth-order operators.

4.3 Mean and Gauss Curvature Computation

We further verify Surface DC-PSE by computing the mean curvature H and Gauss curvature K of an ellipsoid

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{22}$$

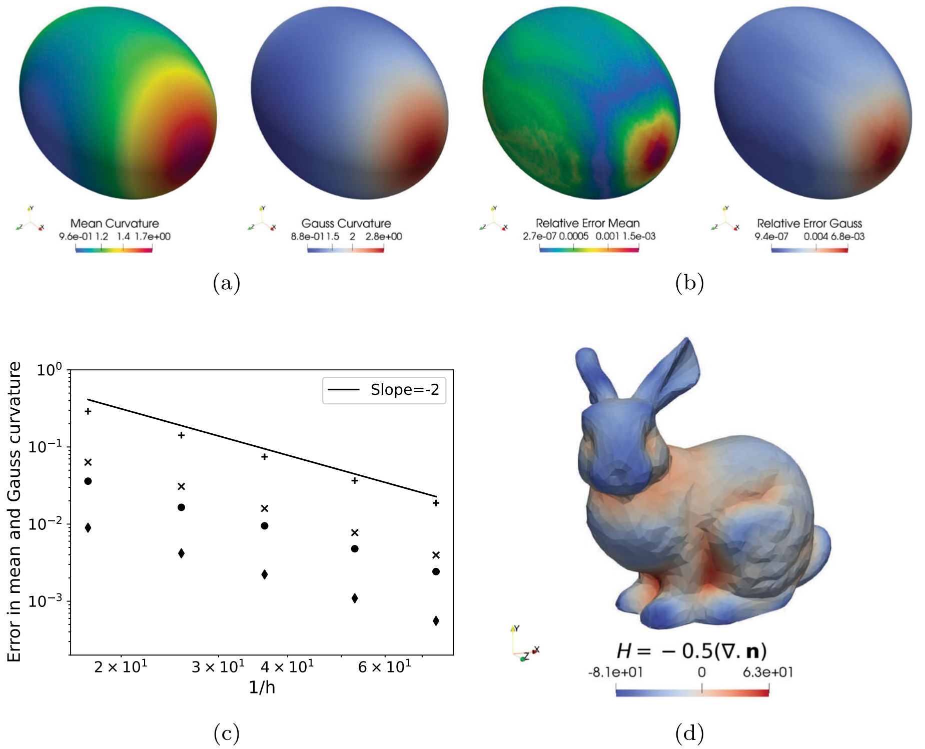
with $a = 1$, $b = 0.8$, $c = 0.75$ and parameterization (u, v)

$$x = a\cos u \sin v, \qquad y = b \sin u \sin v, \qquad z = c\cos v. \tag{23}$$

We compute both curvatures from the embedded shape tensor $\nabla_{\mathcal{S}}\boldsymbol{n}$, i.e., the extension of the intrinsic shape operator to the embedding space. We numerically compute this tensor in Cartesian coordinates as the 3×3 matrix

$$\nabla_{\mathcal{S}}\boldsymbol{n} = \begin{pmatrix} \nabla_{\mathcal{S}} n_1 \\ \nabla_{\mathcal{S}} n_2 \\ \nabla_{\mathcal{S}} n_3 \end{pmatrix},$$

where $\boldsymbol{n} = (n_1, n_2, n_3)$ are the components of the analytically given normal vectors at the collocation point. All intrinsic derivatives over the scalar fields n_1, n_2, n_3 are approximated by Surface DC-PSE operators. Mean curvature H is then computed as the trace of the embedded



(a)

(b)

(c)

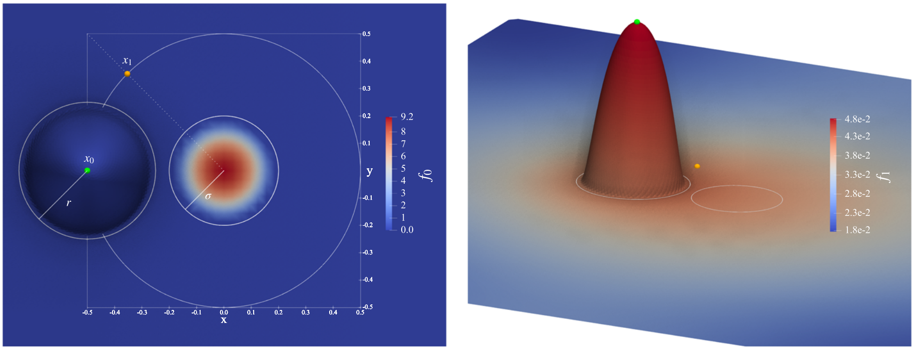
(d)

Fig. 4 Gauss and mean curvature computation using Surface DC-PSE. **a** Visualization of the mean and Gauss curvatures of an ellipsoid, numerically computed as the trace of the embedded shape tensor $-0.5\nabla_{\mathcal{S}} \cdot \boldsymbol{n} = -0.5\mathrm{Tr}(\nabla_{\mathcal{S}}\boldsymbol{n})$ of the surface normal vector field $\boldsymbol{n}$ and the product of the non-zero eigenvalues of $\nabla_{\mathcal{S}}\boldsymbol{n}$, respectively, using second-order accurate Surface DC-PSE operators. **b** Visualization of the relative errors in the computed curvatures from (**a**) on 32258 surface points in comparison with the analytical solution in Eq. (24). They range from about 10^{-7} to 10^{-3}. **c** Convergence plot of the mean and Gauss curvature computations. $L_\infty(\bullet, +)$ and $L_2(\blacklozenge, \times)$ absolute errors for mean and Gauss curvatures, respectively, are computed against the analytical solutions in Eq. (24) for decreasing average collocation point spacing h. **d** Visualization of the mean curvature computed on the Stanford bunny with 2960 points using second-order accurate Surface DC-PSE operators

shape tensor and Gauss curvature K as the product of its non-zero eigenvalues (principal curvatures). These numerical computations are verified against the analytical solutions

$$
\begin{aligned}
H &= \frac{abc[3(a^2+b^2)+2c^2+(a^2+b^2-2c^2)\cos(2v)-2(a^2-b^2)\cos(2u)\sin^2 v]}{8[a^2b^2\cos^2 v+c^2(b^2\cos^2 u+a^2\sin^2 u)\sin^2 v]^{3/2}}\\
K &= \frac{a^2b^2c^2}{\left[a^2b^2\cos^2 v+c^2\left(b^2\cos^2 u+a^2\sin^2 u\right)\sin^2 v\right]^2}.
\end{aligned}
\tag{24}
$$

We approximate the embedded shape tensor $\nabla_{\mathcal{S}}\boldsymbol{n}$ using Surface DC-PSE with $\delta n = 3.0/(N_p - 1)$, $r_c = 2.9\delta n$, $N_n = 2$, and $r = 2$. The results and the convergence plot of the absolute errors are shown in Fig. 4a, c. As specified by r, we observe second-order convergence to the analytical solution when decreasing the average spacing h between the points. The relative errors are visualized in Fig. 4b. They concentrate around extremal points of the curvature, as expected.



(a) (b)

Fig. 5 Illustration of the benchmark problem considering a surface $\mathcal{S}$ with an isolated "bump" given by the graph of a function over the two-dimensional domain $\Omega = [-2, 2]^2$. **a** Top view of the center part with the initial condition f_0 shown by color. The smallest white circle encloses the area where $f_0 \neq 0$ (radius $\sigma = 0.2$). The mid-sized white circle encloses the "bump" (radius $r = 0.25$). The solutions computed by the different numerical methods are probed and compared at the two surface points above $\boldsymbol{x}_0 = (-0.5,\ 0)$ (green dot) and $\boldsymbol{x}_1 = 0.25(-\sqrt{2},\ \sqrt{2})$ (orange dot). The largest white circle of radius 0.5 and the dotted diagonal line help locate the point $\boldsymbol{x}_1$ with respect to the "bump". **b** Perspective view of the surface with the Surface DC-PSE solution at final time $t = 1$ shown by color. For reference, we also show the same circles as shown in (**a**) for the initial condition and the "bump", as well as the probe points $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ (Color figure online)

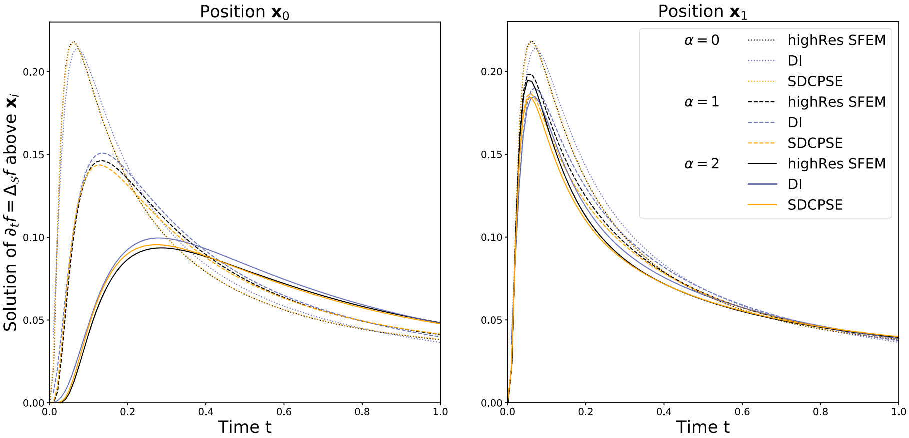
Finally, we apply the same mean-curvature computation to an arbitrary surface with no analytical solution, the Stanford bunny from the Stanford Computer Graphics Laboratory. We use the down-sampled version of the original data set with 2960 points on the surface, obtained from https://www.stlfinder.com/model/stanford-bunny-S4kAUsKI/3091553. The result is visualized in Fig. 4d, showing that the Surface DC-PSE qualitatively works also for non-algebraic surfaces.

4.4 Comparison with Surface Finite Element Methods

We validate Surface DC-PSE by comparing it with surface Finite Element Methods (FEM) on a test case specifically developed for benchmarking FEM solutions of surface problems [1]. The benchmark problem considers a two-dimensional surface $\mathcal{S}$ with an isolated "bump" described by the graph of the function $u(\boldsymbol{x}) = \alpha\zeta(\|\boldsymbol{x} - \boldsymbol{p}\|/r)$, where $\alpha \geq 0$ is the height of the bump, $\boldsymbol{p} = (-0.5,\ 0)$ is the position of its center, and $r = 0.25$ is the bump radius (see Fig. 5a). The function $\zeta(d) = \exp\left(-\frac{1}{1-d^2}\right)$ for $d < 0.975$ and 0 otherwise is a cut-off compressed Gaussian. The surface is thus defined as $\mathcal{S} = \{\boldsymbol{x}_{\mathcal{S}} = (x_1,\ x_2,\ u(x_1, x_2)) \in \mathbb{R}^3 \,|\, \boldsymbol{x} = (x_1,\ x_2) \in \Omega\}$ over the square $\Omega = [-2,\ 2]^2 \subset \mathbb{R}^2$. This surface is asymmetric along the x direction, containing regions of both positive and negative Gauss curvature (see Fig. 5b).

We numerically solve the diffusion equation $\partial_t f(\boldsymbol{x}_{\mathcal{S}}, t) = \Delta_{\mathcal{S}} f$ on $\mathcal{S}$ with no-flux Neumann boundary conditions at all borders of the domain Ω. The initial condition is the truncated Gaussian $f_0 = f(\boldsymbol{x}_{\mathcal{S}}, 0) = \sigma^{-2}\zeta(\|\boldsymbol{x}_{\mathcal{S}}\|/\sigma)$ centered at $\boldsymbol{x}_{\mathcal{S}} = (0, 0, 0)$ with $\sigma = 0.2$ (Fig. 5a).

The collocation points for Surface DC-PSE are regularly distributed in the flat parts of the surface, while in the region of the bump ($[-0.75, -0.25] \times [-0.25, 0.25]$) we use the Fibonacci sphere technique [13] to place the points. This results in a total of $N_p = 18\,439$



(a) (b)

Fig. 6 Comparison of Surface DC-PSE (SDCPSE) with two different surface Finite Element Methods (SFEM and DI) on the benchmark described in Fig. 5 and in the main text, solving the diffusion equation on a graph surface with an isolated "bump" [1]. Orange lines correspond to solutions obtained using Surface DC-PSE, blue lines correspond to those from DI FEM, and black lines correspond to those using high-resolution SFEM (reference solution). Solutions for three different bump heights α are displayed by different line styles (see inset legend). **a** Plot of the solution $f(\boldsymbol{x}_{\mathcal{S}}, t)$ at the bump maximum above the point $\boldsymbol{x}_0 = (-0.5, 0)$. **b** Plot of the solution on the surface above the point $\boldsymbol{x}_1 = 0.25(-\sqrt{2}, \sqrt{2})$

points on the surface, of which 16 441 are in the flat parts of the surface. The resulting average point spacing is $h = 0.03125$. We use a normal spacing of $\delta n = h$ to generate the surface operators with $N_n = 3$ layers to either side of the surface. We choose $r_c = 2.9\delta n$ as the operator support and $r = 2$ as the desired order of convergence. The Neumann boundary conditions at the edge of the surface are imposed using the method of images [8] with around 3000 "ghost points" outside the domain. Time integration over $t \in [0, 1]$ is done using the explicit fifth-order Dormand–Prince method [10] with a time-step size of $\delta t = 10^{-4}$.

We qualitatively compare the solution computed by Surface DC-PSE with those obtained by Finite Element Methods using the data reported in Ref. [1] for Surface FEM (SFEM) [11], Intrinsic Surface FEM (ISFEM) [2], trace FEM (TraceFEM) [26], and diffuse interface (DI) FEM [17, 30]. We report the results obtained by high-resolution SFEM and DI FEM. The solutions obtained using a lower-resolution SFEM, as well as using ISFEM and TraceFEM are visually indistinguishable and reported elsewhere [1].

The comparison is done at two points on the surface above $\boldsymbol{x}_0 = \boldsymbol{p} = (-0.5, 0)$ and $\boldsymbol{x}_1 = 0.25(-\sqrt{2}, \sqrt{2})$ for three different bump heights $\alpha \in \{0, 1, 2\}$ [1]. There is no analytical solution for this test case, but a high-resolution SFEM solution serves as a reference. This "highRes SFEM" reference solution was obtained using a ten-times finer mesh than Surface DC-PSE, with the finest space resolution (on the bump) set to $h \approx 0.0027$, a time step size of $\delta t = 10^{-4}$, a standard BDF-2 scheme, and a polynomial order of two [1]. Results are shown in Fig. 6.

We observe excellent agreement between the Surface DC-PSE solution and the "highRes SFEM" reference solution for $\alpha = 0$ (dotted lines in Fig. 6). For quantitative comparison, we report the absolute difference in the peak values of the solutions, e_{peak}. At both probe points, $e_{\text{peak}} \approx 7 \cdot 10^{-4}$ for the flat case, which is comparable to the other FEM methods [1].

For the curved cases ($\alpha \in \{1, 2\}$), Surface DC-PSE is still in very good agreement with the reference solution and closer to it than DI FEM (i.e., the L_2 difference between highRes SFEM and Surface DC-PSE is smaller in all cases than between highRes SFEM and DI FEM). For $\alpha = 1$ (dashed lines in Fig. 6), the absolute difference in the peak values between Surface DC-PSE and highRes SFEM is $e_{\text{peak}} \approx 2.6 \cdot 10^{-3}$ above $\boldsymbol{x}_0$ and $e_{\text{peak}} \approx 1.2 \cdot 10^{-2}$ above $\boldsymbol{x}_1$. The differences for DI FEM are of the same order. For the highest bump ($\alpha = 2$, solid lines in Fig. 6), we observe similar e_{peak} values and overall behavior for all methods. Consistently, and as expected, the curves have decreasing peak values for increasing α.

5 Conclusions

We have presented a meshfree collocation scheme for consistently approximating intrinsic differential operators on smooth curved surfaces represented by point clouds. The present scheme is based on the DC-PSE method for discretizing differential operators on (almost) arbitrarily distributed collocation points in flat spaces [32]. We have derived the present surface-intrinsic version by realizing that the kernel evaluations can be factored out across points created by *exact* constant orthogonal extension, and that the partial sums over the kernels can be precomputed and stored on the surface points only, defining effective surface kernels. This yields a method that is easy to implement and computationally efficient, as it only requires storing points *on* the surface. In this sense, Surface DC-PSE combines features from embedding methods with features from embedding-free methods. The operators are determined in an embedding formulation, but result in a surface-intrinsic algorithm for operator evaluation.

We have verified the method in different test cases with known analytical solutions. This included evaluating the Laplace–Beltrami operator on the unit circle and the unit sphere, solving Poisson equations on the unit circle and the unit sphere using an implicit solver, and computing mean and Gauss curvature of an ellipsoid via approximation of the embedded shape tensor. In all cases, the Surface DC-PSE method converged as expected. We then applied the method to compute the mean curvature of the Stanford bunny, showing an application to a non-parametric surface. We expect DC-PSE to be more robust (i.e., better conditioned) than Surface MLS in complex geometries [5] and likely computationally more efficient, since Surface MLS uses local moving frames in co-dimension 2, which is different from the present tubular approach in co-dimension 1. Finally, we validated Surface DC-PSE against two different Finite Element Methods (FEM) for surface problems, showing excellent agreement with the reference solution.

Despite its advantages, Surface DC-PSE also has a number of limitations: First, the normal field is required as an input, which can be limiting or introduce additional errors if the normals need to be numerically approximated. Second, for a given point distribution, the numerical error is limited by the curvature of the represented surface and depends on the average spacing between the surface points and the normally extended points (see Figs. 1b and 4b). The required minimum resolution can be determined based on an approximation of the curvature. Third, Surface DC-PSE is only correct for orientable surfaces that possess a non-intersecting tubular neighborhood (i.e., a tubular network) with a radius at least as large as the kernel radius everywhere. This guarantees that the line segments along which the surface points are extended never intersect. For surfaces that possess any non-intersecting tubular neighborhood, this can always be achieved by choosing the resolution h (locally) sufficiently small, since the tube radius scales with the radius of the DC-PSE kernel. Fourth, we limited our discussion to surfaces in co-dimension 1. While it may be possible to extend Surface DC-PSE to co-dimension 2 (e.g., curves embedded in $\mathbb{R}^3$), the method is likely not computationally efficient in those cases, as it would require constant orthogonal extension in the whole two-dimensional normal space of each surface point. Lastly, constructing the Surface DC-PSE kernels is computationally expensive, as it involves solving a small linear system of equations at each surface point in order to determine the embedding-space DC-PSE kernels. For Eulerian simulations, where the collocation points do not move, the kernels have to be determined once at the beginning of the simulation. However, if points move, e.g. in a Lagrangian simulation or simulations involving deforming surfaces, the kernels need to be recomputed at each time step. While the corresponding cost may be amortized by a gain in accuracy and stability [32], it is still significant.

In future work, we will consider extensions of Surface DC-PSE to Lagrangian problems involving moving and deforming surfaces. This also includes simulations of *deformable* surfaces, where the surface deformation itself results from intrinsic force-balance equations [21, 31]. We will also consider coupling Surface DC-PSE with regular DC-PSE in the surrounding space in order to describe coupled bulk-surface phenomena, as well as adaptive-resolution Surface DC-PSE with resolution and tube radius locally adjusted to the curvature of the surface in order to maintain error equi-distribution [28].

In summary, we have extended the meshfree collocation method DC-PSE to scalar-valued problems on curved surfaces, requiring only intrinsic surface points. Like DC-PSE, also Surface DC-PSE computes the operator kernels numerically at runtime and is consistent for any desired order of convergence r. This makes the presented algorithm particularly attractive for higher-order intrinsic operators, such as the fourth-order operators in Fig. 3, and for determining the system matrices of implicit equations on surfaces or implicit time integration schemes.

Acknowledgements We thank Dr. Simon Praetorius and Prof. Axel Voigt, both from the Faculty of Mathematics at Technische Universität Dresden, for providing the benchmark data of the FEM methods in Sect. 4.4. We thank Bryce Palmer (Michigan State University & Flatiron CCB) for his useful comments on the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) under grants FOR-3013 ("Vector- and tensor-valued surface PDEs") (author A.F.), GRK-1907 ("RoSI: role-based software infrastructures") (author A.S.), and SB-350008342 ("OpenPME") (author P.I.), and by the Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) under funding code 031L0160 (project "SPlaT-DM – computer simulation platform for topology-driven morphogenesis") (author I.F.S.).

Data Availibility The C++ source code of the Surface DC-PSE implementation is available in the numerics module of the open-source scalable scientific computing library OpenFPM: https://github.com/mosaic-group/openfpm_numerics. All examples shown in this manuscript are contained in the "example" folder of the OpenFPM repository at: https://github.com/mosaicgroup/openfpm_pdata/tree/master/example/Numerics/Surface_DCPSE.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevantto the content of this article.

References

1. Bachini, E., Brandner, P., Jankuhn, T., Nestler, M., Praetorius, S., Reusken, A., Voigt, A.: Diffusion of tangential tensor fields: numerical issues and influence of geometric properties (2022). https://doi.org/10.48550/arXiv.2205.12581
2. Bachini, E., Farthing, M.W., Putti, M.: Intrinsic finite element method for advection–diffusion–reaction equations on surfaces. J. Comput. Phys. **424**, 109827 (2021). https://doi.org/10.1016/j.jcp.2020.109827
3. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object-oriented numerical software libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (eds.) Modern Software Tools for Scientific Computing, pp. 163–202. Birkhäuser, Boston (1997). https://doi.org/10.1007/978-1-4612-1986-6_8
4. Bergdorf, M., Sbalzarini, I.F., Koumoutsakos, P.: A Lagrangian particle method for reaction–diffusion systems on deforming surfaces. J. Math. Biol. **61**(5), 649–663 (2010). https://doi.org/10.1007/s00285-009-0315-2
5. Bourantas, G.C., Cheeseman, B.L., Ramaswamy, R., Sbalzarini, I.F.: Using DC PSE operator discretization in Eulerian meshless collocation methods improves their robustness in complex geometries. Comput. Fluids **136**, 285–300 (2016). https://doi.org/10.1016/j.compfluid.2016.06.010
6. Chun, S.: Method of moving frames to solve conservation laws on curved surfaces. J. Sci. Comput. **53**(2), 268–294 (2012). https://doi.org/10.1007/s10915-011-9570-7
7. Chun, S., Eskilsson, C.: Method of moving frames to solve the shallow water equations on arbitrary rotating curved surfaces. J. Comput. Phys. **333**, 1–23 (2017). https://doi.org/10.1016/j.jcp.2016.12.013
8. Cottet, G.H., Koumoutsakos, P.D.: Vortex Methods: Theory and Practice. Cambridge University Press, Cambridge (2000)
9. Cottet, G.H., Maitre, E.: A semi-implicit level set method for multiphase flows and fluid–structure interaction problems. J. Comput. Phys. **314**, 80–92 (2016). https://doi.org/10.1016/j.jcp.2016.03.004
10. Dormand, J., Prince, P.: A family of embedded Runge–Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980). https://doi.org/10.1016/0771-050X(80)90013-3
11. Dziuk, G., Elliott, C.M.: Finite element methods for surface PDEs. Acta Numerica **22**, 289–396 (2013). https://doi.org/10.1017/S0962492913000056
12. Eldredge, J.D., Leonard, A., Colonius, T.: A general deterministic treatment of derivatives in particle methods. J. Comput. Phys. **180**(2), 686–709 (2002). https://doi.org/10.1006/jcph.2002.7112
13. González, Á.: Measurement of areas on a sphere using Fibonacci and latitude–longitude lattices. Math. Geosci. **42**(1), 49–64 (2009). https://doi.org/10.1007/s11004-009-9257-x
14. Grande, J., Olshanskii, M., Reusken, A.: A space-time FEM for PDEs on evolving surfaces. In: 11th World Congress on Computational Mechanics (WCCM XI), pp. 211–222 (2014)
15. Incardona, P., Leo, A., Zaluzhnyi, Y., Ramaswamy, R., Sbalzarini, I.F.: OpenFPM: a scalable open framework for particle and particle-mesh codes on parallel computers. Comput. Phys. Commun. **241**, 155–177 (2019). https://doi.org/10.1016/j.cpc.2019.03.007
16. James, A.J., Lowengrub, J.: A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant. J. Comput. Phys. **201**, 685–722 (2004). https://doi.org/10.1016/j.jcp.2004.06.013
17. Lehrenfeld, C., Reusken, A.: High Order Unfitted Finite Element Methods for Interface Problems and PDEs on Surfaces, pp. 33–63. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56602-3_2
18. Leung, S., Zhao, H.: A grid based particle method for moving interface problems. J. Comput. Phys. **228**(8), 2993–3024 (2009). https://doi.org/10.1016/j.jcp.2009.01.005
19. Liang, J., Zhao, H.: Solving partial differential equations on point clouds. SIAM J. Sci. Comput. **35**(3), A1461–A1486 (2013). https://doi.org/10.1137/120869730
20. Macdonald, C.B., Brandman, J., Ruuth, S.J.: Solving eigenvalue problems on curved surfaces using the closest point method. J. Comput. Phys. **230**(22), 7944–7956 (2011). https://doi.org/10.1016/j.jcp.2011.06.021
21. Mietke, A., Jülicher, F., Sbalzarini, I.F.: Self-organized shape dynamics of active surfaces. PNAS **116**(1), 29–34 (2019). https://doi.org/10.1073/pnas.1810896115
22. März, T., Macdonald, C.B.: Calculus on surfaces with general closest point functions. SIAM J. Numer. Anal. **50**(6), 3303–3328 (2012). https://doi.org/10.1137/120865537
23. Nestler, M., Nitschke, I., Voigt, A.: A finite element approach for vector- and tensor-valued surface PDEs. J. Comput. Phys. **389**, 48–61 (2019). https://doi.org/10.1016/j.jcp.2019.03.006

24. Nitschke, I., Reuther, S., Voigt, A.: Discrete exterior calculus (DEC) for the surface Navier–Stokes equation. In: Bothe, D., Reusken, A. (eds.) Transport Processes at Fluidic Interfaces, pp. 177–197. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56602-3_7
25. Nitschke, I., Reuther, S., Voigt, A.: Liquid crystals on deformable surfaces. Proc. R. Soc. A: Math. Phys. Eng. Sci. (2020). https://doi.org/10.1098/rspa.2020.0313
26. Olshanskii, M., Reusken, A.: Trace finite element methods for PDEs on surfaces. Lect. Notes Comput. Sci. Eng. **121**, 211–258 (2017). https://doi.org/10.1007/978-3-319-71431-8_7
27. Peng, D., Merriman, B., Osher, S., Zhao, H., Kang, M.: A PDE-based fast local level set method. J. Comput. Phys. **155**, 410–438 (1999). https://doi.org/10.1006/jcph.1999.6345
28. Reboux, S., Schrader, B., Sbalzarini, I.F.: A self-organizing Lagrangian particle method for adaptive-resolution advection–diffusion simulations. J. Comput. Phys. **231**(9), 3623–3646 (2012). https://doi.org/10.1016/j.jcp.2012.01.026
29. Ruuth, S.J., Merriman, B.: A simple embedding method for solving partial differential equations on surfaces. J. Comput. Phys. **227**(3), 1943–1961 (2008). https://doi.org/10.1016/j.jcp.2007.10.009
30. Rätz, A., Voigt, A.: PDEs on surfaces—a diffuse interface approach. Commun. Math. Sci. **4**(3), 575–590 (2006). https://doi.org/10.4310/CMS.2006.v4.n3.a5
31. Salbreux, G., Jülicher, F.: Mechanics of active surfaces. Phys. Rev. E **96**(3), 032404 (2017). https://doi.org/10.1103/PhysRevE.96.032404
32. Schrader, B., Reboux, S., Sbalzarini, I.F.: Discretization correction of general integral PSE operators for particle methods. J. Comput. Phys. **229**(11), 4159–4182 (2010). https://doi.org/10.1016/j.jcp.2010.02.004
33. Schrader, B., Reboux, S., Sbalzarini, I.F.: Choosing the best Kernel: performance models for diffusion operators in particle methods. SIAM J. Sci. Comput. **34**(3), A1607–A1634 (2012). https://doi.org/10.1137/110835815
34. Singh, A., Incardona, P., Sbalzarini, I.F.: A C++ expression system for partial differential equations enables generic simulations of biological hydrodynamics. Eur. Phys. J. E **44**(9), 117 (2021). https://doi.org/10.1140/epje/s10189-021-00121-x
35. Suchde, P., Kuhnert, J., Tiwari, S.: On meshfree GFDM solvers for the incompressible Navier–Stokes equations. Comput. Fluids **165**, 1–12 (2018). https://doi.org/10.1016/j.compfluid.2018.01.008
36. Voigt, A.: Fluid deformable surfaces. J. Fluid Mech. **878**, 1–4 (2019). https://doi.org/10.1017/jfm.2019.549
37. Wang, R., Yang, Z., Liu, L., Chen, Q.: Discretizing Laplace–Beltrami operator from differential quantities. Commun. Math. Stat. **1**(3), 331–350 (2013). https://doi.org/10.1007/s40304-013-0018-2