Available online at www.sciencedirect.com

ScienceDirect

Procedia IUTAM 18 (2015) 40 – 55

Procedia IUTAM

www.elsevier.com/locate/procedia

IUTAM Symposium on Particle Methods in Fluid Mechanics

A Self-Organizing Adaptive-Resolution Particle Method with Anisotropic Kernels

Christoph Häcki[a], Sylvain Reboux[a], Ivo F. Sbalzarini[a,b,*]

[a]MOSAIC Group, ETH Zurich, Universitätstr. 6, CH–8092 Zurich, Switzerland.
[b]Now at: MOSAIC Group, Max Planck Institute of Molecular Cell Biology and Genetics, Center of Systems Biology, Pfotenhauerstr. 108, D–01307 Dresden, Germany.

Abstract

Adaptive-resolution particle methods reduce the computational cost for problems that develop a wide spectrum of length scales in their solution. Concepts from self-organization can be used to determine suitable particle distributions, sizes, and numbers at runtime. If the spatial derivatives of the function strongly depend on the direction, the computational cost and the required number of particles can be further reduced by using anisotropic particles. Anisotropic particles have ellipsoidal influence regions (shapes) that are locally aligned with the direction of smallest variation of the function. We present a framework that allows consistent evaluation of linear differential operators on arbitrary distributions of anisotropic particles. We further extend the concept of particle self-organization to anisotropic particles, where also the directions and magnitudes of anisotropy are self-adapted. We benchmark the accuracy and efficiency of the method in a number of 2D and 3D test cases.

Keywords: Particle method; PSE; DC-PSE; anisotropy; adaptive resolution; self organization; anisotropic resolution

1. Introduction

Lagrangian particle methods are well suited to simulate convection-dominated problems and problems in complex geometries. Particle methods are adaptive, since particles naturally follow the flow, and the resolution can be locally adapted by changing the core sizes (smoothing lengths) of the particles [1]. Determining the total number of particles required to reach a certain error level, as well as a good placement of the particles in the computational domain, however, is not trivial. Principles from particle self-organization (inverse statistical mechanics) allow automatically finding near-optimal distributions of particles at runtime [2]. Additionally, particles are inserted in under-resolved regions and removed from over-resolved regions in order to ensure consistent function approximation everywhere and to dynamically adapt the total number of particles in the simulation [2].

The numerical error of a particle method at a given position in the computational domain depends on the distances to surrounding particles and on the local properties of the represented function [3]. Concentrating the particles in

* Corresponding author. Tel.: +49 351 2102525.
E-mail address: ivos@mpi-cbg.de

regions of large solution variation, one aims at reducing the upper error bound using a given number of particles. The goal is to equi-distribute the numerical error such that in any part of the domain, the minimum number of particles is used that is needed to guarantee a certain error bound. To this end, various adaptive-resolution methods have been introduced, including adaptive mesh-free finite-element methods [4], vortex methods with spatially varying core sizes [1, 5], and self-organizing particle methods [2].

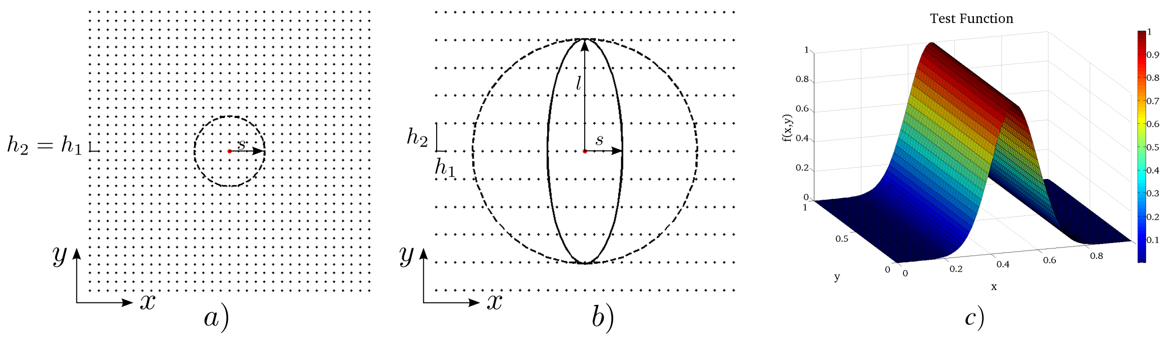


Fig. 1. Illustration of isotropic and anisotropic particles. (a) Isotropic particles with spherical kernels with $h_1 = h_2$. (b) Anisotropic particles with $h_2 = 3h_1$. Both particle distributions approximate the function plotted in (c) to the same accuracy, but the number of particles required is substantially lower when using anisotropic kernels.

While adaptive-resolution methods cope with spatially varying resolution requirements, they do not consider potential anisotropy in the numerical error. Anisotropy implies directionally dependent behavior. Anisotropic functions have different rates of change along different directions. For example, a function could stay constant in the y-direction, but rapidly vary in the x-direction (Fig. 1c). The local resolution required to reach a certain error bound is then dictated by the direction of fastest function variation. The number of particles required to provide this resolution can be significantly reduced if the kernel functions (shape functions) carried by the particles are locally adapted to the directional derivatives of the function (Fig. 1a vs. b). This means that the particle sizes are scaled differently along different directions, giving rise to anisotropic, non-spherical particles, and particle–particle interaction radii that depend on the direction of the interaction. In finite-element methods, this is done using anisotropic meshes [6]. Anisotropic particle methods include the adaptive SPH method of Shapiro and Owen et al. [7, 8]. Their method is based on tracking the evolution of anisotropic smoothing kernels in SPH. A linear transform is used for each particle to describe the mapping of the anisotropic kernel to an isotropic one. Adaptive SPH was also used by Liu et al. [9], who extended it to high-strain hydrodynamics with material strength. Anisotropic particles are also used in computer graphics. Examples include particle-based surface reconstruction using anisotropic kernels [10], which leads to smoother results using the same number of particles.

Here we combine the idea of anisotropic particles [7, 8] with the idea of particle self-organization [2]. We present a mathematical formalism for self-organizing anisotropic particle methods, where the number of particles, their distribution, their sizes, and their anisotropies are found by the method at runtime and do not need to be specified by the user. This results in a fully adaptive method. We have implemented the method in the parallel particle mesh (PPM) library [11, 12]. This required a number of technical extensions to the PPM library, including anisotropic domain decompositions, adaptive-resolution neighbor lists [13], and ghost mappings for anisotropic ghost layers that are described elsewhere [14].

This paper is organized as follows: Section 2 provides the necessary background on continuum particle methods, PSE-type operator approximations, and particle self-organization. For details, we refer to the corresponding original publications [15, 16, 17, 2]. In section 3 we recapitulate anisotropic smooth particle methods [7, 8] and extend them to PSE-type function and operator approximations. Section 4 presents a novel particle self-organization scheme for anisotropic kernels and derives the necessary expressions. In section 5 we present a set of 2D and 3D numerical experiments to illustrate the accuracy and efficiency of anisotropic adaptive-resolution methods compared to their isotropic counterparts. The results are summarized and discussed in section 6. This paper is a write-up of the Master thesis of Christoph Häcki, and we refer to the original thesis for more details as well as for results on the parallel scalability and efficiency of the PPM implementation [14].

2. Background

We briefly recapitulate the basics of smooth particle methods, PSE operators, and particle self-organization. For more details, we refer to the original publications [15, 16, 17, 2]. Since DC-PSE operators on non-uniform particle distributions are not conservative, we discretize the strong form of the governing equations, where the particles carry intensive properties.

2.1. Smooth particle methods

Continuum models can be simulated using smooth particle methods, where particles $p = 1,\dots,N$ are defined by their positions $\vec{x}_p$ and scaled, smooth kernel functions $\zeta_\varepsilon(\vec{y}) = \varepsilon^{-\mathrm{d}}\zeta(\vec{y}/\varepsilon)$ that they carry. The kernel functions are scaled by the function values $\vec{u}_p$, leading to a nonparametric regression-type function approximation

$$u^h_\varepsilon(\vec{x}) = \sum_{p=1}^{N} \vec{u}_p \zeta_\varepsilon(\vec{x}_p - \vec{x}) \tag{1}$$

for the continuous function $u(\vec{x})$. The core size (smoothing length) ε and the inter-particle distance h define the local resolution of the method. The core size ε additionally acts as a regularizer of the numerical solution. The error of this type of function approximation is bounded as [18]

$$u^h_\varepsilon(\vec{x}) = u(\vec{x}) + O(\varepsilon^r) + O\left(\frac{h}{\varepsilon}\right)^s, \tag{2}$$

where r is the number of vanishing moments of the kernel ζ and s depends on the number of continuous derivatives of ζ. For the method to be consistent, we require that $h < \varepsilon$.

The dynamics of the model is then governed by the evolution of the particle properties, which in its most general form may depend on the properties of all other particles in the system [15]:

$$\frac{\mathrm{d}\vec{x}_p}{\mathrm{d}t} = \vec{v}_p(\vec{x}_p, t) = \sum_{q=1}^{N} \vec{K}(\vec{x}_p, \vec{x}_q; \vec{u}_p, \vec{u}_q) \tag{3}$$

$$\frac{\mathrm{d}\vec{u}_p}{\mathrm{d}t} = \sum_{q=1}^{N} \vec{F}(\vec{x}_p, \vec{x}_q; \vec{u}_p, \vec{u}_q)\,. \tag{4}$$

The functions $\vec{K}$ and $\vec{F}$ define the model being simulated. This formulation is Lagrangian, since the particles travel with the flow velocity field $\vec{v}(\vec{x}, t)$.

2.2. PSE-type operator approximation

Differential operators can be discretized over particle function approximations using a number of methods, leading to different functions $\vec{K}$ and $\vec{F}$ in Eqs. (3) and (4). Based on the particle strength exchange (PSE) framework, a general treatment of derivatives in particle methods was introduced by Eldredge et al. [16]. In this framework, a linear differential operator $L^{\vec{\beta}}$ of degree $\vec{\beta}$ is approximated over a particle set of resolution h as:

$$L^{\vec{\beta}}_h u(\vec{x}_p) = \frac{1}{\varepsilon^{|\vec{\beta}|}} \sum_{q=1}^{N} \left(u(\vec{x}_q) \pm u(\vec{x}_p)\right) \eta^{\vec{\beta}}_\varepsilon(\vec{x}_p - \vec{x}_q). \tag{5}$$

The multi-index $\vec{\beta}$ indicates the type of derivative that is approximated:

$$L^{\vec{\beta}} = \frac{\partial^{|\vec{\beta}|}}{\partial x_1^{\beta_1} \partial x_2^{\beta_2} \dots \partial x_{\mathrm{d}}^{\beta_{\mathrm{d}}}}, \tag{6}$$

where $\vec{\beta} = (\beta_1, \beta_2, ..., \beta_\mathrm{d})$ and $|\vec{\beta}| = \beta_1 + \beta_2 + ... + \beta_\mathrm{d}$. The sign in the first parenthesis of Eq. (5) is positive for odd $|\vec{\beta}|$ and negative for even $|\vec{\beta}|$. We further define $\vec{y}^{\vec{\beta}} = y_1^{\beta_1} y_2^{\beta_2} ... y_\mathrm{d}^{\beta_\mathrm{d}}$, $\vec{\beta}! = \beta_1!\beta_2!...\beta_\mathrm{d}!$, and $\sum_{|\vec{\beta}|=i}^{j}$ sums over all multi-indexes $\vec{\beta}$ for which $i \le |\vec{\beta}| \le j$. The function $\eta_\varepsilon^{\vec{\beta}}(\vec{y}) = \varepsilon^{-\mathrm{d}} \eta^{\vec{\beta}}(\vec{y}/\varepsilon)$ is an operator kernel that depends on the type of derivative and the desired convergence order of the discretization.

In PSE-type methods, the operator kernels η are determined by solving a system of moment conditions [16]. Compared to the standard SPH choice $\eta = L^{\vec{\beta}} \zeta$, this has the advantage of providing the full order of convergence for all degrees of derivatives. Convergence of the original PSE scheme, however, depends on a regular particle distribution. If particles have different sizes or are irregularly distributed, the kernels need to be locally adapted for each particle depending on the distribution and sizes or neighboring particles. This is, e.g., done in DC-PSE methods [17]. DC-PSE locally satisfies conditions for the discrete moments of order $\vec{\alpha}$

$$Z_h^{\vec{\alpha}}(\vec{x}) = \frac{1}{\varepsilon^\mathrm{d}} \sum_{p \in \mathscr{N}(\vec{x})} \vec{z}^{\vec{\alpha}} \eta^{\vec{\beta}}(\vec{z}), \quad \vec{z} = \frac{\vec{x} - \vec{x}_p}{\varepsilon}, \tag{7}$$

where $\mathscr{N}(\vec{x})$ is the neighborhood around $\vec{x}$ within a given cutoff radius r_c. Kernels are then computed according to the template [17]

$$\eta^{\vec{\beta}}(\vec{z}, \vec{x}) = \left(\sum_{|\vec{\gamma}| = \vec{\alpha}_\mathrm{min}}^{|\vec{\beta}|+r-1} a_{\vec{\gamma}}(\vec{x}) \vec{z}^{\vec{\gamma}} \right) \mathrm{e}^{-|\vec{z}|^2}, \quad \vec{\alpha}_\mathrm{min} = \begin{cases} 0, & \text{if } |\vec{\beta}| \text{ odd} \\ 1, & \text{if } |\vec{\beta}| \text{ even} \end{cases}, \tag{8}$$

potentially leading to a different kernel for each particle. Substituting this kernel template into the discrete moment conditions leads to the linear system of equations for each particle. The kernel coefficients $a_{\vec{\gamma}}(\vec{x})$ are found by solving these linear systems. This guarantees the full rate of convergence on arbitrary particle distributions, as well as near boundaries.

Since each particle may carry a different kernel, the smoothing length ε can also be different for different particles, leading to the function approximation

$$u_\varepsilon^{h_p}(\vec{x}) = \sum_{p=1}^{N} \vec{u}_p \zeta_{\varepsilon_p}(\vec{x}_p - \vec{x}), \tag{9}$$

where ε_p is the smoothing length (size) of particle p and h_p the distance to its nearest neighbor. Two particles are considered neighbors if their cutoff radii r_c mutually enclose each other, hence $\mathscr{N}(\vec{x}_p) = \{\vec{x}_q : |(\vec{x}_p - \vec{x}_q)| < \min(r_{c,p}, r_{c,q})\}$. This guarantees that neighbor lists are symmetric. Efficient algorithms exist for constructing such neighbor lists [13]. The resulting operator approximation, however, is no longer conservative.

2.3. Particle self-organization

One of the key difficulties in smooth particle methods is to determine the total number of particles and their distribution required to reach a certain error level. Concepts from self-organization can be used to let the particles dynamically rearrange, split, and fuse at runtime. This has been proposed in a self-organizing particle method for advection-diffusion simulations [2]. In such methods, particles self-organize so as to approximately equi-distribute the numerical error. Exact error equi-distribution would lead to an optimal method in the sense that it uses the minimum number of particles required to reach the given error level everywhere in the domain. At each time step (or every few steps), particles self-organize in order to adapt to the current resolution requirements as defined by a monitor function or an error predictor. The numerical solution is then interpolated from the old set of particles to the new, adapted set before the simulation continues. Given an unadapted set of particles $\{\vec{x}_p\}_{p=1...N}$ with cutoff radii $r_{c,p}$, and a smooth field $\tilde{D}(\vec{x})$, the self-organization process determines a new set of particles whose distribution and sizes are adapted according to $\tilde{D}(\vec{x})$.

$\tilde{D}(\vec{x})$ hence plays the role of a monitor function, prescribing the required spatial resolution at each position [2]. A typical choice is:

$$\tilde{D}(\vec{x}) = \max\left(D_{\min}, \frac{D_{\max}}{\sqrt{1+|\nabla u(\vec{x})|^2}}\right), \tag{10}$$

where the parameter $D_{\max}$ is an upper bound for the distance between two neighboring particles and for the difference in function values between two particles separated by a distance of $\tilde{D}(\vec{x})$. The resolution $\tilde{D}(\vec{x})$ is bounded from below by $D_{\min}$ in order to globally limit the total number of particles in a simulation. The local distance to neighbors is defined as:

$$D(\vec{x}_p) = D_p = \min_{|\vec{x}_q-\vec{x}_p|\le\tilde{D}(\vec{x}_p)r^*} \tilde{D}(\vec{x}_q), \tag{11}$$

where $\varepsilon_p = r_{c,p} = D_p r^*$ is the smoothing length and $r_{c,p}$ the cutoff radius of particle p. Particles self-organize to provide a spatial resolution that is approximately equal to $\tilde{D}(\vec{x})$. This is done by approximately minimizing the potential energy $W(\vec{x}_1,...,\vec{x}_N) = \sum_p^N \sum_q^N V_{pq}$ with the pairwise interaction potential $V_{pq} = D_{pq}^2 V(|\vec{x}_p - \vec{x}_q|/D_{pq})$, where V is the generalized (using a ratio of 4/5 between the two length scales, instead of the usual 1/2) dimensionless Morse potential $V(r) = 0.8\cdot 2.5^{1-5r} - 2.5^{-4r}$ [20] and $D_{pq} = \min(D_p, D_q)$. Approximate minimization is done by performing a few steps of steepest descent along the negative gradient of W.

In order for the DC-PSE operators to be determined, each particle must have a minimum number of N^* neighbors (typically around 15, but the exact value of N^* depends on the order of the DC-PSE operator [19]). Particles are hence dynamically added and removed in order to guarantee that all linear systems are uniquely determined. This amounts to inserting particles in under-resolved regions and removing particle in over-resolved regions [2].

3. Anisotropic Smooth Particle Methods

The number of particles required to represent an anisotropic function up to a certain error can be significantly reduced by using anisotropic particles whose shapes and orientations are locally adapted to the function being represented. We show how the isotropic DC-PSE method presented in section 2 can be modified to account for anisotropy.

3.1. The smoothing tensor

In isotropic methods, the smoothing length ε_p is used to describe the region of influence of a particle p. In anisotropic methods, this is replaced by a smoothing tensor $\boldsymbol{T} = (h_{i,j})$. The column vectors of $\boldsymbol{T}$ define the orthogonal axes of an ellipsoid and are ordered by decreasing length. Isotropic particles correspond to the limit case $\boldsymbol{T} = \varepsilon_p \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity tensor.

3.2. Reference space

Function and operator approximations as discussed in section 2 rely on an isotropic smoothing length. Anisotropic particles are treated by defining a reference space in which the kernels appear isotropic [7, 8]. This is achieved by measuring distances in the metric induced by the smoothing tensor. The smoothing tensor can be interpreted as a mapping from the unit circle to the surface of the ellipsoid defining the particle's influence region. The inverse mapping is given by the tensors

$$\boldsymbol{G}_{2D} = \boldsymbol{T}_{2D}^{-1} = \frac{1}{\det(\boldsymbol{T}_{2D})}\begin{pmatrix} h_{22} & -h_{12} \\ -h_{21} & h_{11} \end{pmatrix} \tag{12}$$

$$\boldsymbol{G}_{3D} = \boldsymbol{T}_{3D}^{-1} = \frac{1}{\det(\boldsymbol{T}_{3D})}\begin{pmatrix} h_{22}h_{33}-h_{23}h_{32} & h_{13}h_{32}-h_{12}h_{33} & h_{12}h_{23}-h_{13}h_{22} \\ h_{23}h_{31}-h_{21}h_{33} & h_{11}h_{33}-h_{13}h_{31} & h_{13}h_{21}-h_{11}h_{23} \\ h_{21}h_{32}-h_{22}h_{31} & h_{12}h_{31}-h_{11}h_{32} & h_{11}h_{22}-h_{12}h_{21} \end{pmatrix}. \tag{13}$$

Since the smoothing tensors are orthogonal, the inverse always exists. Figure 2 illustrates the relation between them.

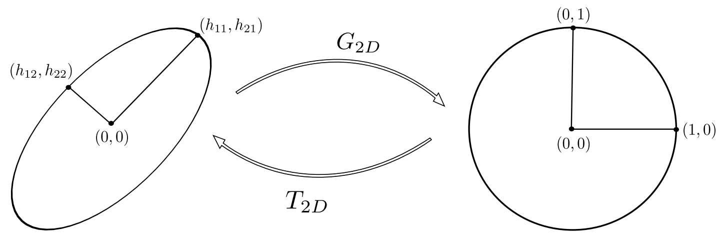


Fig. 2. A 2D example illustrating the smoothing tensor $\boldsymbol{T}$ of an anisotropic particle and its relation to the transform $\boldsymbol{G} = \boldsymbol{T}^{-1}$ of the surface of the anisotropic particle to the isotropic unit circle.

3.3. Function approximation

Function approximation using anisotropic particles is done by replacing the kernel $\zeta_{\varepsilon_p}(\vec{x}_p - \vec{x}) = \varepsilon^{-\mathrm{d}}\zeta((\vec{x}_p - \vec{x})/\varepsilon_p)$ by its anisotropic counterpart $\zeta_{\boldsymbol{G}_p}(\vec{x}_p - \vec{x}) = \|\boldsymbol{G}_p\|\,\zeta(\boldsymbol{G}_p(\vec{x}_p - \vec{x}))$. This leads to the function approximation

$$u^h_\varepsilon(\vec{x}) = \sum_{p=1}^{N} \vec{u}_p \,\|\boldsymbol{G}_p\|\,\zeta(\boldsymbol{G}_p(\vec{x}_p - \vec{x}))\,. \tag{14}$$

The matrix norm $\|\boldsymbol{G}\|$ has to be chosen such that the kernel $\zeta_{\boldsymbol{G}}$ has the same moments as the Dirac delta function up to the desired order of accuracy [18]. Setting $\|\boldsymbol{G}\| = |\det(\boldsymbol{G})|$ guarantees that $\int \zeta_{\boldsymbol{G}}(\vec{x})\mathrm{d}\vec{x} = 1$. Using integration by substitution for multiple variables, we can furthermore show that:

$$\int_U \zeta_{\boldsymbol{G}}(\boldsymbol{G}\vec{u})\mathrm{d}\vec{u} = \int_U \zeta(\boldsymbol{G}\vec{u})|\det(\boldsymbol{G})|\mathrm{d}\vec{u} = \int_{\boldsymbol{G}U} \zeta(\vec{v})\mathrm{d}\vec{v} = 1\,. \tag{15}$$

Choosing $\|\boldsymbol{G}\| = \det(\boldsymbol{G})$ hence fulfills the requirements for a valid function-approximation kernel.

3.4. Operator approximation

The operators presented in section 2.2 assume isotropic smoothing lengths. They are hence used to approximate derivatives in reference space, which amounts to measuring distances between particles using the norm induced by $\boldsymbol{G}$.

Using substitution of variables, one can show that derivates in reference space (reference-space coordinate $\tilde{\vec{x}}$) are related to derivatives in real space $\mathbb{R}^{\mathrm{d}}$ as:

$$\frac{\partial^{|\vec{\beta}|}u}{\partial x_1 \partial x_2 \dots \partial x_{|\vec{\beta}|}} = \sum_{m=1}^{\mathrm{d}} G_{m,1}\left(\sum_{n=1}^{\mathrm{d}} G_{n,2}\left(\dots\left(\sum_{k=1}^{\mathrm{d}} G_{k,|\vec{\beta}|}\frac{\partial^{|\vec{\beta}|}g}{\partial \tilde{x}_m \partial \tilde{x}_n \dots \partial \tilde{x}_u}\right)\right)\right). \tag{16}$$

We hence need all partial derivatives of order $|\vec{\beta}|$ in reference space to compute any partial derivative of the same order in real space. Gradients are related as $\nabla_{\vec{x}}u(\vec{x}) = \boldsymbol{G}^T\,\nabla_{\tilde{\vec{x}}}g(\tilde{\vec{x}})$, and the Hessian as $\boldsymbol{H}_{\vec{x}}(u(\vec{x})) = \boldsymbol{G}^T\,\boldsymbol{H}_{\tilde{\vec{x}}}(g(\tilde{\vec{x}}))\,\boldsymbol{G}$.

We adapt DC-PSE operators to anisotropic particles. By construction, the smoothing length in reference space always is $\varepsilon = 1$. The discrete moments from section 2.2 hence become:

$$Z_h^{\vec{\alpha}}(\vec{x}) = \sum_{p\in\mathcal{N}(\vec{x})} \vec{z}^{\vec{\alpha}}\,\eta^{\vec{\beta}}(\vec{z}), \quad \vec{z} = \boldsymbol{G}(\vec{x} - \vec{x}_p), \tag{17}$$

where neighborhood $\mathcal{N}(\vec{x})$ is defined in section 3.5. This change only impacts the weights of the linear system. Using these two adaptations in DC-PSE operators, we can compute derivatives in reference space. These are then transformed to real space using Eq. (16).

3.5. Anisotropic neighborhood definition

Anisotropic particles change the definition of the neighborhood. In order to determine whether $\vec{x}_p$ is in the neighborhood of $\vec{x}$, we need to consider the distance in reference space. The anisotropic neighborhood is hence defined as $\mathscr{N}(\vec{x}) = \{\vec{x}_p \,:\, |\boldsymbol{G}(\vec{x}-\vec{x}_p)| < r_c\}$.

4. Particle Self-Organization for Adaptive-Resolution Methods with Anisotropic Kernels

We extend isotropic adaptive-resolution particle methods with self-organizing particles [2] to anisotropic kernels. This includes three contributions: generalizing the smoothing length to a smoothing tensor, consistent removal and insertion of anisotropic particles, and anisotropic self-organization by energy minimization.

The error of the function approximation in Eq. (9) can be locally bounded by [3]:

$$\left| u(\vec{x}_p) - u_\varepsilon^{h_p}(\vec{x}_p) \right| \leq C h_p^m \|u(\vec{x})\|_{W_\infty^m(\theta)}\,, \tag{18}$$

where h_p is the local inter-particle spacing, $\|u(\vec{x})\|_{W_\infty^m(u)} = \max_{|\vec{\alpha}|=m} \|\partial u/\partial \vec{x}^{\vec{\alpha}}\|_{L^\infty(u)}$ is a measure for the magnitude of the derivatives of the function u, and θ is the ball with radius $r_{c,p}$ around particle p. The exponent m is the convergence order of the method.

In words, the error at position $\vec{x}_p$ is bounded from above by the partial derivative of degree $|\vec{\alpha}|$ that has the largest magnitude within the vicinity defined by the cutoff radius, scaled with the local inter-particle distance. As we are interested in a maximum-error bound over the entire domain, as well as in minimizing the total number of particles, we do not benefit from lower error bounds in some parts of the domain while having larger errors in other parts. Consequently, we wish to equi-distribute the error, which means that the upper bound at a specific position $e_x = C_x h_x \|u(\vec{x})\|_{W_\infty^m(\theta)}$ should be equal for all $\vec{x} \in u$. Variable smoothing lengths are necessary to achieve this goal. They are, however, not sufficient. Anisotropic kernels and cutoff radii are additionally required in order to exploit directionality in the rate of change of the function u. Doing so, the domain of influence of a particle is not a ball anymore, but the interior of an ellipsoid. The use of anisotropic particles can hence further reduce the total number of particles required to reach a certain maximum-error bound.

4.1. Smoothing tensor determination

In anisotropic particles, the smoothing length ε_p is replaced by a smoothing tensor. Likewise, the functions $\tilde{\boldsymbol{D}}(\vec{x})$ and $\boldsymbol{D}(\vec{x})$ are not scalar-valued anymore, but return tensors describing the required resolution at each position. The inverse smoothing tensor of particle p is then defined as:

$$\boldsymbol{G}_p = \frac{1}{r^*}\boldsymbol{D}(\vec{x}_p)^{-1}\,. \tag{19}$$

In the following, we describe how the monitor tensor function $\boldsymbol{D}(\vec{x})$ is constructed.

4.1.1. Monitor tensor axes directions

The column vectors of the monitor tensors are orthogonal, limiting the choice of possible tensors. We align the smallest axis of the monitor tensor with the gradient of the represented function u. The other directions are chosen in the plane normal to the gradient. As we are often interested in approximating surfaces (e.g., using level sets), the other two directions are less relevant for reducing the number of particles. Therefore, we choose the axes of the monitor tensor as:

$$\text{in 2D:}\quad \vec{\mathrm{d}}_1 = \nabla u(\vec{x}) = \begin{pmatrix} u_x \\ u_y \end{pmatrix}, \quad \vec{\mathrm{d}}_2 = \begin{pmatrix} -u_y \\ u_x \end{pmatrix}, \tag{20}$$

$$\text{in 3D:}\quad \vec{\mathrm{d}}_1 = \nabla u(\vec{x}) = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}, \quad \vec{\mathrm{d}}_2 = \begin{pmatrix} -u_y \\ u_x \\ 0 \end{pmatrix}, \quad \vec{\mathrm{d}}_3 = \begin{pmatrix} -u_x u_z \\ -u_y u_z \\ u_x^2 + u_y^2 \end{pmatrix}. \tag{21}$$

4.1.2. Monitor tensor axes lengths

The axes length are determined by separate, scalar monitor functions along each axis. Using monitor functions of the form given in Eq. (10) allows one to specify lower and upper bounds on the admissible axes lengths using the parameters $D_{\max}$ and $D_{\min}$. The length of the axis $\vec{\mathrm{d}}_1$ is given by:

$$h_1 = \max\left(D_{\min}, \frac{D_{\max}}{\sqrt{1+|\nabla u(\vec{x})|^2}}\right). \tag{22}$$

The first derivatives in the directions orthogonal to the gradient are zero. In order to get an approximation of how the function varies along these directions, we project the gradients at the neighboring particles onto the tensor axes and take the maximum. The axes lengths of the other directions, $\vec{\mathrm{d}}_2$ and $\vec{\mathrm{d}}_3$, are hence given by:

$$h_{2,3} = \max\left(D_{\min}, \frac{D_{\max}}{\sqrt{1+\nu^2}}\right) \quad , \quad \nu = \max_{\vec{x}_p \in \mathcal{N}(\vec{x})}\left(\frac{\nabla u(\vec{x}_p)\cdot\vec{\mathrm{d}}_{2,3}}{|\vec{\mathrm{d}}_{2,3}|}\right). \tag{23}$$

The tensor-valued monitor function $\tilde{\boldsymbol{D}}(\vec{x})$ is then defined as:

$$\text{in 2D:}\quad \tilde{\boldsymbol{D}}(\vec{x}) = \frac{1}{|\nabla u(\vec{x})|}\begin{pmatrix} -h_2 u_y & h_1 u_x \\ h_2 u_x & h_1 u_y \end{pmatrix}. \tag{24}$$

$$\text{in 3D:}\quad \tilde{\boldsymbol{D}}(\vec{x}) = \begin{pmatrix} -\tilde{h}_3 u_x u_z & -\tilde{h}_2 u_y & \tilde{h}_1 u_x \\ -\tilde{h}_3 u_y u_z & \tilde{h}_2 u_x & \tilde{h}_1 u_y \\ \tilde{h}_3 (u_x^2+u_z^2) & 0 & \tilde{h}_1 u_z \end{pmatrix} \quad , \quad \tilde{h}_i = \frac{h_i}{|\vec{\mathrm{d}}_i|}. \tag{25}$$

The column vectors represent the axes of the ellipsoid, sorted by decreasing length.

4.1.3. Considering the neighborhood of $\tilde{\boldsymbol{D}}(\vec{x})$ when computing $\boldsymbol{D}(\vec{x})$

The vicinity defined by $\tilde{\boldsymbol{D}}(\vec{x})$ according to Eqs. (24) and (25) may contain strongly varying gradients (e.g., in a corner of a rectangular surface). While Eq. (11) accounts for this in the isotropic case, we additionally need to consider the particle axes in the anisotropic case. In order to account for the largest change, we choose the smallest axis in the neighborhood to be the smallest axis of the final monitor tensor $\boldsymbol{D}(\vec{x})$. The other axes are given by the smallest projection of any neighbor's maximum axis onto the new orthogonal directions, where $\tilde{\vec{\mathrm{d}}}_{\vec{x}_p,i}$ is the i^{th} column-vector in $\tilde{\boldsymbol{D}}(\vec{x}_p)$. In 2D, the final monitor tensor is hence defined as:

$$\begin{aligned} \vec{x}_p &= \arg\min_{\vec{x}_p \in \mathcal{N}(\vec{x})} |\tilde{\vec{\mathrm{d}}}_{\vec{x}_p,2}|, \\ \boldsymbol{D}(\vec{x}) &= \begin{pmatrix} \nu_2 \tilde{D}_{11}(\vec{x}_p) & \tilde{D}_{12}(\vec{x}_p) \\ \nu_2 \tilde{D}_{21}(\vec{x}_p) & \tilde{D}_{22}(\vec{x}_p) \end{pmatrix}, \\ \nu_2 &= \min_{\vec{x}_q \in \mathcal{N}(\vec{x})} \max\left(\frac{\tilde{\vec{\mathrm{d}}}_{\vec{x}_q,1}\cdot\tilde{\vec{\mathrm{d}}}_{\vec{x}_p,2}}{|\tilde{\vec{\mathrm{d}}}_{\vec{x}_p,2}|}, \frac{\tilde{\vec{\mathrm{d}}}_{\vec{x}_q,2}\cdot\tilde{\vec{\mathrm{d}}}_{\vec{x}_p,2}}{|\tilde{\vec{\mathrm{d}}}_{\vec{x}_p,2}|}\right). \end{aligned}$$

In 3D, the expressions read:

$$\vec{x}_p = \arg \min_{\vec{x}_p \in \mathcal{N}(\vec{x})} |\tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,3}|,$$

$$\boldsymbol{D}(\vec{x}) = \begin{pmatrix} v_3\tilde{D}_{11}(\vec{x}_p) & v_2\tilde{D}_{12}(\vec{x}_p) & \tilde{D}_{13}(\vec{x}_p) \\ v_3\tilde{D}_{21}(\vec{x}_p) & v_2\tilde{D}_{22}(\vec{x}_p) & \tilde{D}_{23}(\vec{x}_p) \\ v_3\tilde{D}_{31}(\vec{x}_p) & v_2\tilde{D}_{32}(\vec{x}_p) & \tilde{D}_{33}(\vec{x}_p) \end{pmatrix},$$

$$v_2 = \min_{\vec{x}_q \in \mathcal{N}(\vec{x})} \max \left(\frac{\tilde{\tilde{\mathrm{d}}}_{\vec{x}_q,1} \cdot \tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,2}}{|\tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,2}|}, \frac{\tilde{\tilde{\mathrm{d}}}_{\vec{x}_q,2} \cdot \tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,2}}{|\tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,2}|}, \frac{\tilde{\tilde{\mathrm{d}}}_{\vec{x}_q,3} \cdot \tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,2}}{|\tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,2}|} \right),$$

$$v_3 = \min_{\vec{x}_q \in \mathcal{N}(\vec{x})} \max \left(\frac{\tilde{\tilde{\mathrm{d}}}_{\vec{x}_q,1} \cdot \tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,3}}{|\tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,3}|}, \frac{\tilde{\tilde{\mathrm{d}}}_{\vec{x}_q,2} \cdot \tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,3}}{|\tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,3}|}, \frac{\tilde{\tilde{\mathrm{d}}}_{\vec{x}_q,3} \cdot \tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,3}}{|\tilde{\tilde{\mathrm{d}}}_{\vec{x}_p,3}|} \right).$$

The tensor-valued function $\boldsymbol{D}(\vec{x})$ then defines the local neighborhood considering the anisotropy of u.

4.2. Particle insertion and removal

In self-organizing particle methods, particles are dynamically inserted and removed in order to adapt the total number of particles to the overall requirement of the simulation. Each particle requires a certain number N^* of neighbors in order for the operator kernel to be uniquely determined. We insert particles in the neighborhood of particle p if p has less than N^* neighbors. We do so by randomly sampling a point $\vec{y}$ on the unit sphere and inserting a new particle with position $\vec{x}_{\text{new}} = \vec{x}_p + \boldsymbol{T}_p\vec{y}$ in the neighborhood of particle p. The smoothing tensor of the new particle is set equal to that of its mother particle p. It will later be adapted during potential minimization.

Since the required resolution is defined by the tensor $\boldsymbol{D}(\vec{x})$, a particle can be removed whenever its nearest neighbor is closer than the Nyquist-Shannon sampling limit, hence:

$$\text{Delete } \vec{x}_p \quad \text{if } \exists q: \; |\boldsymbol{D}(\vec{x}_p)^{-1}(\vec{x}_p - \vec{x}_q)| < 1/2. \tag{26}$$

If this condition is fulfilled in multiple directions, the particle with the smallest distance to its nearest neighbor is deleted. This assures the required minimum smoothing length. The geometric situation around an anisotropic particle is illustrated in Fig. 3.

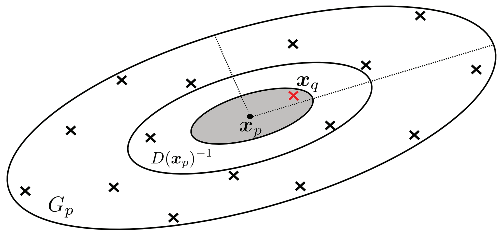


Fig. 3. Illustration of the different regions around an anisotropic particle. $\boldsymbol{D}(\vec{x}_p)^{-1}$ determines the actual size of the particle. $\boldsymbol{G}_p$ is the inverse smoothing tensor that defines the region within which we require a certain number of neighbors. Particle q is deleted because it lies within the fusion region of particle p.

4.3. Anisotropic particle self-organization by potential minimization

Particles self-organize driven by a potential-minimization process as outlined in section 2.3. The goal is to reach a distribution of particles and smoothing tensors close to $\boldsymbol{D}(\vec{x})$. The self-organization potential W depends on the distances between particles. For anisotropic particles, the distance metric has to include the monitor tensor, which specifies how distances are to be measured. This leads to the pairwise self-organization potential $V_{pq} = D_p^2 V(|\boldsymbol{D}(\vec{x}_p)^{-1}(\vec{x}_p - \vec{x}_q)|)$. The pre-factor D_p scales the self-organization force with the size of the anisotropic particle in direction of the interaction partner. We therefore propose

$$D_p = \left| \boldsymbol{D}(\vec{x}_p) \, \frac{D(\vec{x}_p)^{-1}(\vec{x}_p - \vec{x}_q)}{|\boldsymbol{D}(\vec{x}_p)^{-1}(\vec{x}_p - \vec{x}_q)|} \right| = \left| \frac{(\vec{x}_p - \vec{x}_q)}{|\boldsymbol{D}(\vec{x}_p)^{-1}(\vec{x}_p - \vec{x}_q)|} \right|, \tag{27}$$

which is the length scale of particle $\vec{x}_p$ along the direction pointing to $\vec{x}_q$.

We minimize the resulting potential energy by performing a few steps of steepest descent along the negative energy gradient

$$-\frac{\partial W(\vec{x}_1,\ldots,\vec{x}_N)}{\partial \vec{x}_k} = -\sum_p \sum_q \frac{\partial V_{pq}}{\partial \vec{x}_k} = -\sum_p \frac{\partial V_{pk}}{\partial \vec{x}_k} - \sum_q \frac{\partial V_{kq}}{\partial \vec{x}_k}. \tag{28}$$

The last step follows from the fact that the derivative of V_{pq} is only non-zero if $k = p$ or $k = q$. Using the scaled distance vector $\vec{\mathrm{d}}_{kq} = \boldsymbol{D}(\vec{x}_k)^{-1}(\vec{x}_k - \vec{x}_q)$, the partial derivatives $\partial V_{kq}/\partial \vec{x}_k$ and $\partial V_{pk}/\partial \vec{x}_k$ are given by:

$$\frac{\partial V_{kq}}{\partial \vec{x}_k} = D_k^2 \frac{\partial V(|\vec{\mathrm{d}}_{kq}|)}{\partial \vec{x}_k} = D_k^2 \frac{\partial |\vec{\mathrm{d}}_{kq}|}{\partial \vec{x}_k} V'(|\vec{\mathrm{d}}_{kq}|) = D_k^2 \frac{\partial \vec{\mathrm{d}}_{kq}}{\partial \vec{x}_k} \frac{\vec{\mathrm{d}}_{kq}}{|\vec{\mathrm{d}}_{kq}|} V'(|\vec{\mathrm{d}}_{kq}|) = D_k^2 \left(\vec{D}(\vec{x}_k)^{-1}\right)^T \frac{\vec{\mathrm{d}}_{kq}}{|\vec{\mathrm{d}}_{kq}|} V'(|\vec{\mathrm{d}}_{kq}|) \tag{29}$$

and analogously:

$$\frac{\partial V_{pk}}{\partial \vec{x}_k} = -\frac{\partial V_{pk}}{\partial \vec{x}_p} = -D_p^2 \left(\boldsymbol{D}(\vec{x}_p)^{-1}\right)^T \frac{\vec{\mathrm{d}}_{pk}}{|\vec{\mathrm{d}}_{pk}|} V'(|\vec{\mathrm{d}}_{pk}|). \tag{30}$$

A noteworthy difference to the isotropic case is the sub-optimality of the steepest descent method for anisotropic functions. The negative gradient of an anisotropic function does not point to the origin. The method could hence benefit from using a higher-order minimization method, such as conjugate gradients or the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method.

4.4. Overall algorithm

The workflow of the self-organization step for anisotropic particles is summarized in algorithm 1. Compared with its isotropic counterpart, the smoothing lengths and cutoff radii are replaced by the inverse smoothing tensors. The user-adjustable parameters of the method are: the lower and upper bounds on the resolution ($D_{\max}$, $D_{\min}$) and the neighborhood size r^*. The algorithm takes as an input a particle set P_{old} and returns a new particle set P_{new} with particle positions, sizes, and shapes adapted to the current resolution needs as governed by the function u and the monitor tensor function $\boldsymbol{D}$.

5. Numerical Experiments

We present several numerical experiments that demonstrate the convergence of DC-PSE operators with anisotropic kernels, the computational efficiency of the method, and anisotropic particle self-organization. All experiments were done using the PPM library [11, 12] on the Brutus cluster of ETH Zurich (quad-core AMD Opteron 8380 with 2.5GHz and 16 GB memory per node; 16 cores per node).

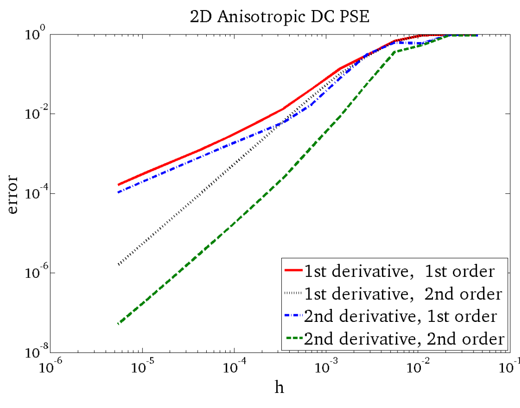
5.1. Convergence of anisotropic DC-PSE operators

As a test case, we approximate the first and second derivatives of the function $u(\vec{x}) = \mathrm{e}^{-2000|\vec{x}|_2^2}$ along x in 2D and 3D, and compare to the analytical solutions. We place N ($N = 2025$ in 2D and $N = 6859$ in 3D) particles on a uniform Cartesian grid in $[-1, 1]^{\mathrm{d}}$ and distort them randomly. The axes directions of the smoothing tensors are randomly

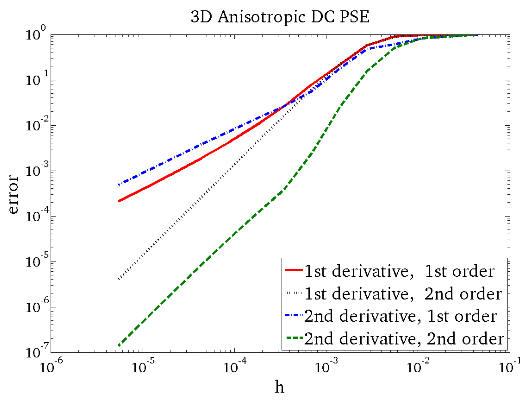
Algorithm 1 Particle self-organization with anisotropic kernels

```
Input: $P_{\text{old}} = \{\vec{x}_p^{\text{old}}, \vec{u}_p^{\text{old}}, \boldsymbol{G}_p^{\text{old}}\}$, $D_{\max}$, $D_{\min}$, $r^*$
Output: $P_{\text{new}} = \{\vec{x}_p^{\text{new}}, \vec{u}_p^{\text{new}}, \boldsymbol{G}_p^{\text{new}}\}$
 1: Create neighbor lists in $P_{\text{old}}$.
 2: Approximate $\nabla u(\vec{x}_p^{\text{old}})$ using an anisotropic DC-PSE operator (section 3.4).
 3: Compute the monitor tensors $\tilde{\boldsymbol{D}}(\vec{x}_p^{\text{old}})$ and $\boldsymbol{D}(\vec{x}_p^{\text{old}})$ (section 4.1).
 4: Compute the inverse smoothing tensors as $\boldsymbol{G}_p^{\text{old}} = 1/r^* \boldsymbol{D}(\vec{x}_p^{\text{old}})^{-1}$.
 5: Copy $P_{\text{old}}$ into $P_{\text{new}}$.
 6: while Any particle in $P_{\text{new}}$ has less than $N^*$ neighbors or Steepest descent is not done do
 7:     Remove particles where $|\boldsymbol{D}(\vec{x}_p^{\text{new}})^{-1}(\vec{x}_q^{\text{new}} - \vec{x}_p^{\text{new}})| < 1/2$.
 8:     Insert particles around particles that have less than $N^*$ neighbors.
 9:     Update neighbor lists within $P_{\text{new}}$ and between $P_{\text{new}}$ and $P_{\text{old}}$ using $\boldsymbol{G}_p^{\text{old}}$.
10:     Compute $\boldsymbol{D}(\vec{x}_p^{\text{new}})$ by 1st order interpolation from $\boldsymbol{D}(\vec{x}_p^{\text{old}})$.
11:     Update inverse smoothing tensor $\boldsymbol{G}_p^{\text{new}}$.
12:     Compute total energy of the adaptation potential and its gradient (section 4.3).
13:     Perform line search for gradient descent step size and move the particles one step down the energy gradient.
14: end while
15: Create cross-neighbor list between $P_{\text{new}}$ and $P_{\text{old}}$ using $\boldsymbol{G}_p^{\text{old}}$.
16: Interpolate the field values $\vec{u}_p^{\text{new}}$ from $\vec{u}_p^{\text{old}}$ using particle–particle interpolation from $P_{\text{old}}$ to $P_{\text{new}}$ [2].
```

rotated unit vectors, and their lengths are sampled uniformly at random in $[\varepsilon/1.5, 1.5\varepsilon]$ with $\varepsilon = 0.30$. We use DC-PSE operators of first and second order accuracy and quantify the relative maximum error (ℓ^∞ norm of the relative error). Figure 4 shows the log-log plots of the error versus the inter-particle spacing (resolution) h. We observe the expected convergence rates, which are equal to those for isotropic DC-PSE operators.



(a)



(b)

Fig. 4. Convergence of 2D (a) and 3D (b) anisotropic DC-PSE operators when approximating the first and second derivatives with respect to x of the function $\exp(-2000\,|\vec{x}|_2^2)$ in the domain $[-1,1]^{\mathrm{d}}$ with first and second order accuracy. The dotted line indicates convergence of order two.

5.2. *Computational efficiency of anisotropic DC-PSE operators*

Figure 5 compares the total runtimes, approximation errors, and number of particles needed by an isotropic and an anisotropic DC-PSE method with particle self-organization. The test case consists of approximating $\nabla u(x,y)$ for $u = \mathrm{e}^{-50(x-0.5)^2}$ (see Fig. 1c) for different ratios of anisotropy h_2/h_1. Using the anisotropic method for ratio $= 1$ exposes the overhead from the anisotropic kernels. This overhead is about 5% of the total runtime, since 90% of the time are used by the DC-PSE operators. Detailed timings and code profiling can be found elsewhere [14].

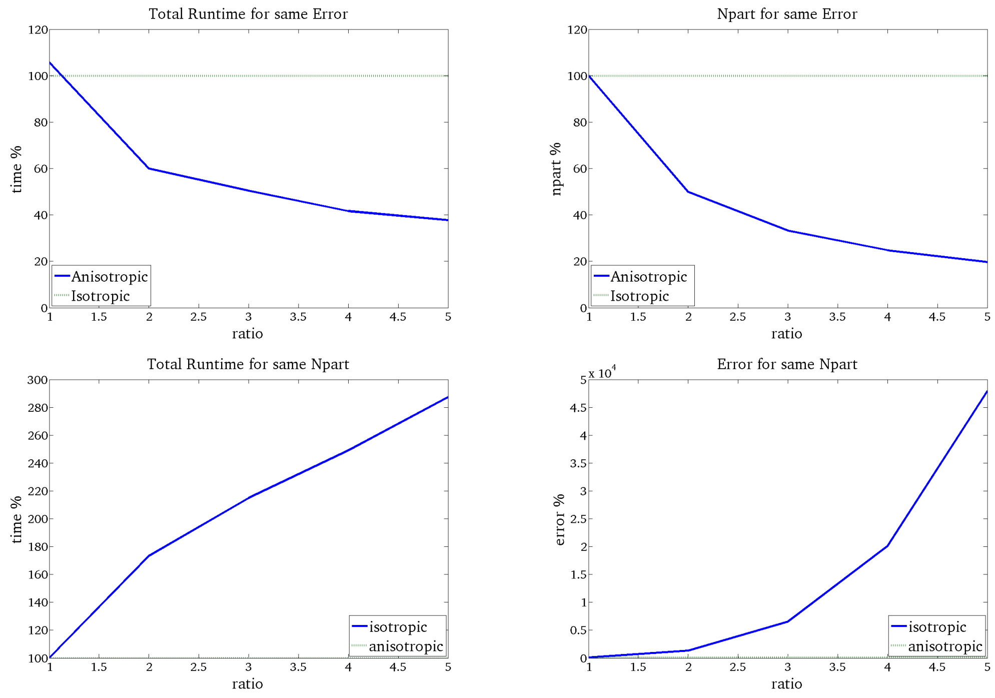


Fig. 5. Upper panels: Total runtime and number of particles required by the anisotropic self-organizing particle method compared to the isotropic variant (normalized to 100%) to approximate a gradient to the same error level. Lower panels: Total runtime and maximum error for isotropic and anisotropic particles for the same number of particles.

5.3. Anisotropic adaptive-resolution method with self-organizing particles

We demonstrate the effect of particle self-organization in a 2D and a 3D example. The test case consists in computing the gradient of the function $u(\vec{x}) = \mathrm{e}^{-50(x-0.5)^2}$ in 2D and of the function $u(\vec{x}) = \mathrm{e}^{-50((x-0.5)^2+(y-0.5)^2)}$ in 3D using 2nd-order DC-PSE operators. We compare the maximum error $\|\nabla u(\vec{x}) - \nabla_h u^h(\vec{x})\|_\infty$. We initially distribute $N = 20\,000$ isotropic particles of size $\varepsilon = 0.025$ on a regular Cartesian lattice. The particles are then let to self-organize with $D_{\mathrm{max}} \in [0.014, 0.25]$, $D_{\mathrm{min}} = 0.001$, $r^* = 2$. In 2D, the resulting numbers of particles range from 203 to 32 722 in the anisotropic method, and from 764 to 109 579 in the isotropic method, depending on the required accuracy level. In 3D, the final numbers of particles range from 4531 to 37 884 in the anisotropic method, and from 4548 to 52 376 in the isotropic method. Figure 6 shows the maximum error versus the average inter-particle spacing $h = (|u|/N)^{1/\mathrm{d}}$. The smoothing tensors are properly resolved, as reflected by the decrease in error for the same inter-particle spacing. The resulting particle distributions after self-organization are illustrated in Fig. 7 for 2D, and in Fig. 8 for 3D.

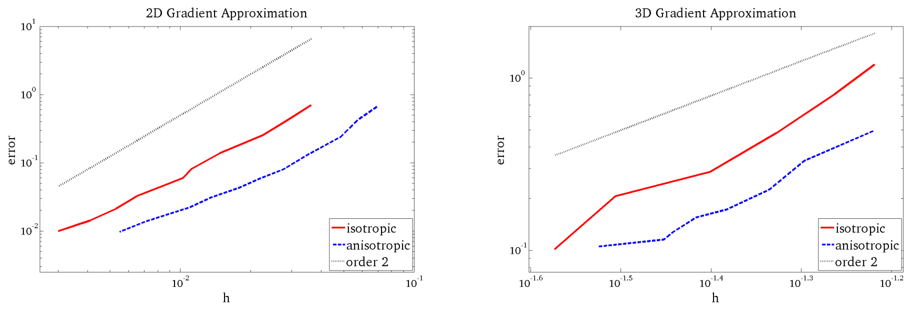
5.4. 2D Burgers equation

We apply the anisotropic adaptive-resolution self-organization method to the 2D Burgers equation

$$\frac{\partial u}{\partial t} + \mathrm{Re}\,\vec{v}\cdot\nabla u = \Delta u\,, \tag{31}$$

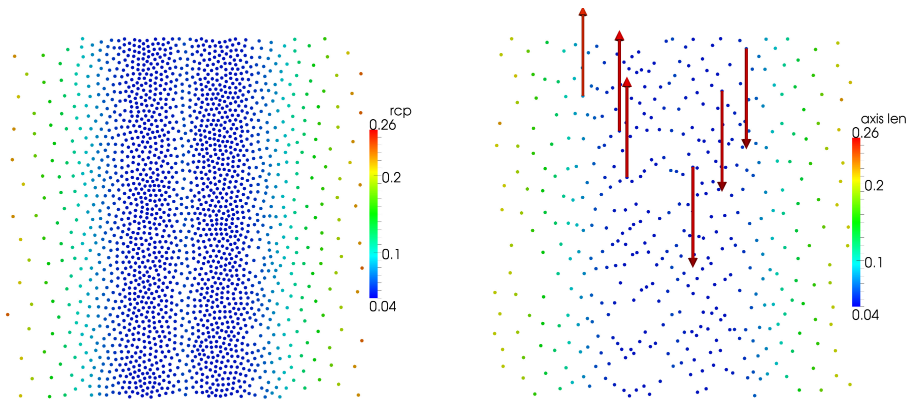
where Re is the Reynolds number and $\vec{v} = (u, u)$ the velocity. We solve Eq. (31) with initial condition $u(x, y, t = 0) = \cos(2\pi x)\cos(2\pi y)$ and doubly periodic boundary conditions in the computational domain $[0, 1]^2$. The solution of the Burgers equation creates regions with steep gradients, but small rates of change in the orthogonal directions.

The particle distribution and numerical solution computed by the present method are shown in Fig. 9(a) at $t = 0.126/\mathrm{Re}$ for $\mathrm{Re} = 100$, $D_{\mathrm{max}} = 0.15$, and $D_{\mathrm{min}} = 0.01$. Fig. 9(b) illustrates the smoothing tensors at the same time



(a) (b)

Fig. 6. Convergence of the maximum error with the average inter-particle spacing in 2D (a) and 3D (b) self-organizing adaptive-resolution particle approximations of a gradient using isotropic and anisotropic particles. The dotted lines indicate convergence order 2.



(a) (b)

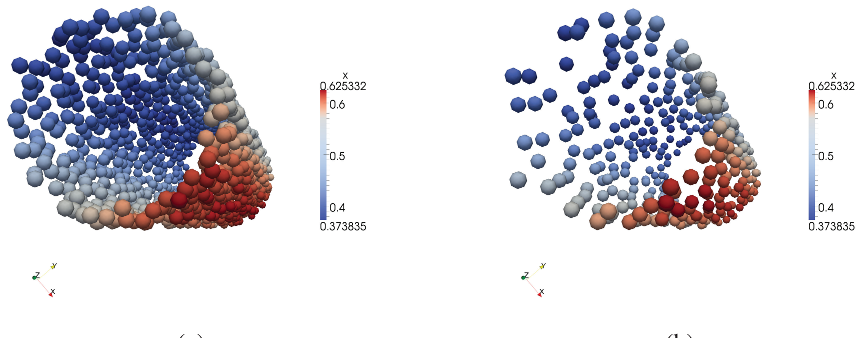
Fig. 7. Particle distributions resulting from self-organization of isotropic (a) and anisotropic (b) particles to represent the gradient of the function shown in Fig. 1(c). Color codes particle size as measured by the interaction cutoff radius $r_{c,p}$. The arrows in (b) indicate the resulting axes of anisotropy after particle self-organization.

by visualizing their principal axes. The steep gradients in the solution are properly resolved by the smaller axes of the anisotropic particles. Additionally, the longer axes are correctly determined to be significantly larger. Regions with isotropic particles correspond to saddle points, where the solution is isotropic.

6. Conclusions

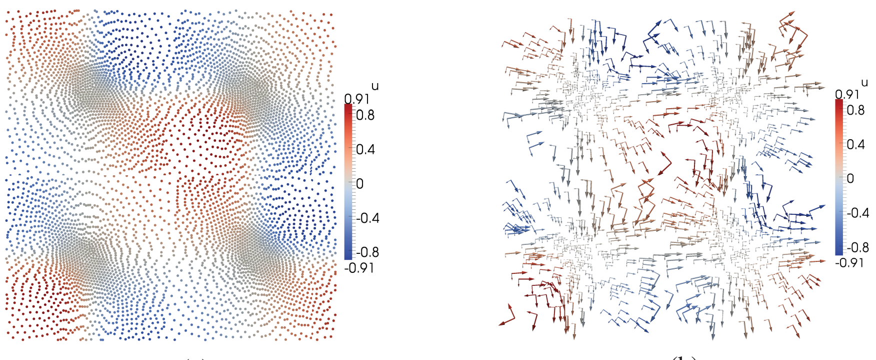
We have presented an adaptive-resolution particle method with anisotropic kernels where the particles and kernels self-organize as governed by the local resolution requirements of the solution. The self-organization process optimizes the particle distribution, their sizes, their anisotropies, their orientations, and their total number. This is done so as to lower the total number of particles and the computational cost needed to reach a certain error level everywhere in the computational domain. Using anisotropic particles allows one to further reduce the cost compared to isotropic adaptive-resolution methods.

The presented method combines the idea of anisotropic particles [7, 8] with the idea of particle self-organization [2]. We used DC-PSE operators [17] to discretize the strong form of the governing equations. We have presented an extension of DC-PSE operators to anisotropic particles. We then extended the concept of particle self-organization [2] to anisotropic particles, where the complete smoothing tensors are adapted through an energy-minimization process, such as to approximately equi-distribute the numerical error in the computational domain. The method dynamically inserts and removes particles in under- and over-resolved regions, respectively. The total number of particles present in a simulation is hence automatically adapted and may vary during the runtime of a simulation.



(a) (b)

Fig. 8. Particle distributions resulting from self-organization to the gradient of the function $\exp(-50((x-0.5)^2+(y-0.5)^2))$ in 3D. Color codes the x coordinate for visualization. For clarity, only particles with function values between 0.45 and 0.55 are visualized.



(a) (b)

Fig. 9. Particle distribution (a) and anisotropy (b) for 1000 self-organizing particles adapting to the solution of the 2D Burgers equation at time $t = 0.126/\mathrm{Re}$ with $\mathrm{Re} = 100$. Color codes the function value u.

We have benchmarked the presented method in a number of 2D and 3D test cases. The tests have shown that anisotropic DC-PSE operators yield the design rate of convergence, and that using anisotropic particles can lead to substantial computational savings. If the represented function is indeed isotropic, the present anisotropic method incurs an about 5% time overhead. The total number of particles used and the numerical error, however, are identical to those obtained with the isotropic self-organizing method. Already at mild anisotropy ratios of about 1.15, anisotropic DC-PSE operators break even with their isotropic counterparts. At anisotropy ratios of 5, the anisotropic method uses about 5 times less particles than the isotropic one, has a significantly lower numerical error, and runs about 3 times faster.

The presented method has been implemented in both 2D and 3D in the parallel particle mesh (PPM) library [11, 12]. This required a number of extensions to the PPM library core, including anisotropic domain decompositions, adaptive-resolution neighbor lists [13], and ghost mappings for anisotropic ghost layers. Details of the parallel implementation, as well as benchmarks of the parallel scalability and efficiency, have been presented elsewhere [14].

Currently, the presented method still has a number of limitations: First, it can only be used to discretize the strong form of an equation. Weak-form discretizations require the computation of particle volumes, which necessitates different strategies for inserting and removing particles. The present particle insertion and removal schemes rely on strong-form discretizations. Second, the method is not conservative. While this is not a problem specific to anisotropic kernels, it is a general limitation of DC-PSE operators evaluated over particles of non-uniform size. Third, even though an error analysis of the method has been done in reference space [14], a corresponding analysis in physical space is not yet available. Fourth, the axes of the smoothing tensor in the plane orthogonal to the gradient are pre-computed. This may be improved by computing the rates of change of the function in the orthogonal plane and also determining the corresponding axes dynamically at runtime.

Acknowledgements

We thank all member of the MOSAIC Group for many fruitful discussions. We particularly thank Omar Awile for his support with the PPM implementation.

References

[1] Hou TY. Convergence of a Variable Blob Vortex Method for the Euler and Navier-Stokes Equations. SIAM J Numer Anal. 1990;27(6):1387–1404.

[2] Reboux S, Schrader B, Sbalzarini IF. A self-organizing Lagrangian particle method for adaptive-resolution advection–diffusion simulations. J Comput Phys. 2012;231:3623–3646.

[3] Chen JS, Han W, You Y, Meng X. A reproducing kernel method with nodal interpolation property. Int J Numer Meth Engng. 2003;56:935–960.

[4] Eriksson K, Estep D, Hansbo P, Johnson C. Introduction to adaptive methods for differential equations. Acta numerica. 1995;4:105–158.

[5] Cottet GH, Koumoutsakos P, Ould Salihi ML. Vortex Methods with Spatially Varying Cores. J Comput Phys. 2000;162:164–185.

[6] Li X, Huang W. An anisotropic mesh adaptation method for the finite element solution of heterogeneous anisotropic diffusion problems. J Comput Phys. 2010;229(21):8072–8094.

[7] Shapiro PR, Martel H, Villumsen JV, Owen JM. Adaptive smoothed particle hydrodynamics, with application to cosmology: Methodology. Astrophys J Suppl Ser. 1996;103:269–330.

[8] Owen JM, Villumsen JV, Shapiro PR, Martel H. Adaptive smoothed particle hydrodynamics: Methodology. II. Astrophys J Suppl Ser. 1998;116:155–209.

[9] Liu MB, Liu GR, Lam KY. Adaptive smoothed particle hydrodynamics for high strain hydrodynamics with material strength. Shock Waves. 2006;15(1):21–29.

[10] Yu J, Turk G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In: Proc. 2010 ACM SIGGRAPH/Eurographics Symp. Computer Animation. ACM; 2010. p. 217–225.

[11] Sbalzarini IF, Walther JH, Bergdorf M, Hieber SE, Kotsalis EM, Koumoutsakos P. PPM – A Highly Efficient Parallel Particle-Mesh Library for the Simulation of Continuum Systems. J Comput Phys. 2006;215(2):566–588.

[12] Awile O, Demirel O, Sbalzarini IF. Toward an Object-Oriented Core of the PPM Library. In: Proc. ICNAAM, Numerical Analysis and Applied Mathematics, International Conference. AIP; 2010. p. 1313–1316.

[13] Awile O, Büyükkeçeci F, Reboux S, Sbalzarini IF. Fast neighbor lists for adaptive-resolution particle simulations. Comput Phys Commun. 2012;183:1073–1081.

[14] Häcki C. A Parallel Anisotropic Adaptive-Resolution Particle Method for Advection-Diffusion Simulations [Master thesis]. MOSAIC Group, ETH Zurich; 2011.

[15] Koumoutsakos P. Multiscale Flow Simulations Using Particles. Annu Rev Fluid Mech. 2005;37:457–487.

[16] Eldredge JD, Leonard A, Colonius T. A General Deterministic Treatment of Derivatives in Particle Methods. J Comput Phys. 2002;180:686–709.

[17] Schrader B, Reboux S, Sbalzarini IF. Discretization Correction of General Integral PSE Operators in Particle Methods. J Comput Phys. 2010;229:4159–4182.

[18] Cottet GH, Koumoutsakos P. Vortex Methods – Theory and Practice. New York: Cambridge University Press; 2000.

[19] Schrader B, Reboux S, Sbalzarini IF. Choosing the best kernel: performance models for diffusion operators in particle methods. SIAM J Sci Comput. 2012;34(3):A1607–A1634.

[20] D'Orsogna M, Chuang Y-L, Bertozzi A, Chayes L. Pattern formation stability and collapse in 2D driven particle systems. In: Device Applications of Nonlinear Dynamics, Understanding Complex Systems, vol. 7. Springer; 2006. p. 103–113.