Journal of Computational Physics 231 (2012) 3623–3646

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A self-organizing Lagrangian particle method for adaptive-resolution advection–diffusion simulations

Sylvain Reboux, Birte Schrader, Ivo F. Sbalzarini [*]

MOSAIC Group, ETH Zurich, Universitaetstr. 6, CAB E64.1, CH-8092 Zurich, Switzerland Swiss Institute of Bioinformatics, Zurich, Switzerland

ARTICLE INFO

Article history:
Received 10 December 2010
Received in revised form 9 November 2011
Accepted 3 January 2012
Available online 1 February 2012

Keywords:
Particle method
Adaptive resolution
Self-organization
Pseudo force
Adaptivity
Lagrangian method
Advection–diffusion

ABSTRACT

We present a novel adaptive-resolution particle method for continuous parabolic problems. In this method, particles self-organize in order to adapt to local resolution requirements. This is achieved by pseudo forces that are designed so as to guarantee that the solution is always well sampled and that no holes or clusters develop in the particle distribution. The particle sizes are locally adapted to the length scale of the solution. Differential operators are consistently evaluated on the evolving set of irregularly distributed particles of varying sizes using discretization-corrected operators. The method does not rely on any global transforms or mapping functions. After presenting the method and its error analysis, we demonstrate its capabilities and limitations on a set of two- and three-dimensional benchmark problems. These include advection–diffusion, the Burgers equation, the Buckley–Leverett five-spot problem, and curvature-driven level-set surface refinement.

1. Introduction

In Lagrangian particle methods (LPM) for advection–reaction–diffusion problems, field variables are discretized on an unstructured set of nodes called *particles* that are advected by the flow map in a Lagrangian way. The nodes do not have to satisfy any topological connectivity constraints, i.e., they do not have to form a (structured or unstructured) mesh or lattice of any kind. This makes the method inherently adaptive with respect to the flow map. However, advection may lead to the formation of holes or clusters in the distribution of particles, jeopardizing the consistency of the method [14,20]. This issue is commonly addressed by a remeshing procedure that consists in periodically reinitializing the particles onto a regular Cartesian mesh [22].

In systems with large spatial inhomogeneities, however, uniform remeshing is undesirable, and LPM have to be equipped with adaptive-resolution or multi-resolution capabilities in order to remain computationally efficient. A number of such frameworks have been proposed and we briefly summarize them below. For an in-depth review in the context of flow simulations we refer to Koumoutsakos [23]. We distinguish between *adaptive-resolution* and *multi-resolution* methods. In adaptive-resolution methods, the resolution of the discretization is given by a unique-valued map $\mathbf{x} \in \mathbb{R}^d \mapsto D(\mathbf{x}) \in \mathbb{R}^+$ assigning to each location $\mathbf{x}$ a local target resolution $D > 0$. This is in contrast to multi-resolution methods where the solution is represented on multiple resolution levels simultaneously at any given location.

Adaptive-resolution LPM were first introduced as vortex methods with spatially varying core sizes by Hou [20] and further developed by Cottet et al. [15]. The method relies on a mapping from the physical space, where particle sizes are locally adapted, to a reference space with uniform resolution. All operators, including remeshing, are applied in the uniform

[*] Corresponding author. Tel.: +41 44 632 6344; fax: +41 44 632 1562.
E-mail address: ivos@ethz.ch (I.F. Sbalzarini).

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2012.01.026

reference space. This imposes the condition that particles living in low-resolution areas must not travel too far into any high-resolution area between two remeshing steps Cottet et al. [15]. Also, the coordinate transform (mapping function) from physical space to reference space needs to be explicitly known. This was later relaxed in the adaptive global mapping (AGM) [8] formulation by numerically approximating the mapping function on the same particles that also represent the flow fields, and numerically evaluating the Jacobian of the mapping in order to map differential operators between physical space and reference space. This concept is related to *r*-adaptive finite element methods, in which computational elements are dynamically moved to areas where increased resolution is needed [11]. The problem can be formulated as the equi-distribution of a *monitor function* [10]. AGM evaluates this monitor function at each particle location and computes the global mapping from physical space to reference space. The concept of *r*-adaptivity, however, is limited to locally "distorting" the resolution map around the given set of particles. It does not allow for creation or removal of particles during adaptation.

Multi-resolution LPM include wavelet reproducing-kernel particle methods (WRKPM) [28], wavelet particle methods (WPM) [7], adaptive mesh refinement (AMR) as applied to particle methods [8,32], and adaptive tree codes [31]. Wavelet particle methods combine a sparse multi-resolution representation of the solution with a Lagrangian adaptation mechanism [7]. AMR-type particle methods employ hierarchies of overlapping mesh patches onto which the particles are remeshed at every time step [8,32].

Here, we propose an adaptive-resolution LPM that does not require global transforms or mapping functions. The method is based on the concept of self-organization with *pseudo forces* driving the particles to areas where higher resolution is needed, and dynamic insertion and removal of particles in under- and over-resolved regions, respectively. In addition, the core sizes and interaction cutoff radii of the particles are locally adapted to the required resolution. This generates a self-organizing configuration of particles that collectively represent the solution with a locally adapted resolution. Adaptation is done using a Lagrangian mechanism with pseudo forces that are determined by approximate equi-distribution of a monitor function defining the desired target resolution everywhere. Since the particles self-organize, the monitor function does not need to be known *a priori* and is allowed to evolve during a simulation. Moreover, the total number of particles required to reach a certain error level does not need to be known or imposed, but the self-organization mechanism finds it automatically.

Pseudo forces have previously been used in moving-mesh methods [1]. There, the mesh nodes interact with each other through pseudo forces that depend on a measure of the local truncation error. The use of pseudo forces to adapt particle locations has been described to stabilize smoothed particle hydrodynamics (SPH) simulations [18] using an artificial pressure term based on the Lennard–Jones potential. Dynamic insertion and removal of particles has previously been considered in the context of hybrid particle-mesh methods for convection–reaction–diffusion problems [42].

The present method combines these concepts and determines the adaptation pseudo forces and the particle insertion/removal strategy based on the fact that under certain conditions interacting particles spontaneously self-assemble into organized structures [43,13,33,35,34]. We exploit this to design adaptation pseudo forces that lead to particle distributions with the desired resolution and regularity properties.

Particle self-organization also dispenses with the need for remeshing onto uniform or adaptive-resolution Cartesian meshes. This is because the pseudo forces and particle insertion/removal guarantee consistent representation of the solution at all times and (by construction) prevent the formation of holes in the particle distribution. Remeshing is hence replaced by interpolation to the newly adapted set of particles after self-organization. After moving the particles according to the adaptation pseudo forces and inserting or removing particles where needed, the field quantities are interpolated from the old set of particles to the new one. Since both sets are irregularly distributed, the interpolation schemes required differ from those used in remeshing procedures. Rather, they are conceptually related to the interpolation scheme used in Behrens's adaptive semi-Lagrangian method [5] for radial basis functions. Differential operators can be consistently approximated on the evolving, irregular set of particles using the DC-PSE scheme [39], which can be seen as a generalization of vorticity redistribution schemes [41,4,25] to arbitrary differential operators.

This paper is organized as follows: In Section 2 we recall the basics of Lagrangian particle methods and introduce a novel variant in which particles self-organize by gradient descent on a pseudo-potential energy. The algorithm is described in detail in Section 3 and numerical benchmarks are presented in Section 4. We conclude with a summary and a discussion of possible extensions in Section 5.

2. A self-organizing adaptive-resolution Lagrangian particle method

We first review the basic concepts of adaptive-resolution LPM and introduce the operators used here to approximate spatial derivatives on scattered sets of particles and to interpolate between two sets of particles. We then introduce the pseudo forces and particle insertion/removal strategies that lead to self-organizing adaptive-resolution particle arrangements.

2.1. Adaptive-resolution Lagrangian particle methods for parabolic problems

We focus on parabolic problems of the form:

$$\frac{\partial f}{\partial t} + \nabla \cdot (\mathbf{u} f) = \mathcal{L}(f), \tag{2.1}$$

where $\mathcal{L}$ is an elliptic differential operator, $\mathbf{u}$ a *given* advection velocity field, and $f : \mathbb{R}^d \to \mathbb{R}$ a continuous scalar field representing the concentration of the transported quantity.

In LPM, the field *f* is discretized on a set of particles that carry local or compact kernel (basis) functions. Eq. (2.1) is then solved by a method of lines: the particles follow the streamlines of the flow and carry the quantity corresponding to the field *f*. The accuracy of the method depends on how the differential operator $\mathcal{L}$ is discretized over the particles and on the regularity of the particle distribution. In order to avoid spurious distortions of the field and to ensure sufficient sampling, particles can be periodically reinitialized on a Cartesian mesh using a remeshing procedure [22]. This also enables discretizing the operator $\mathcal{L}$ using mesh-based schemes, such as finite differences.

LPM can be classified according to whether the particles carry extensive or intensive quantities. If they carry extensive quantities, the LPM discretizes the *weak form* of Eq. (2.1). In this case, the particles have a physical *volume* and their extensive *strengths* are computed by integrating *f* over the volume of the particle. The kernels carried by extensive particles must overlap with those of their neighbors at all times for the function approximation to be consistent [14,20]. This can, e.g., be guaranteed by remeshing. The particle positions, strengths, and volumes then evolve according to a system of ordinary differential equations [23]. Directly attributing the local value of *f* to each particle, disregarding particle volumes, leads to LPM for the strong form of Eq. (2.1) where the particles carry intensive quantities. There, as in generalized finite-difference methods, the function approximation is given by

$$f(\mathbf{x}) \approx \sum_{p=1}^{N} f_p \zeta_{\epsilon_p}(\mathbf{x};\mathbf{x}_p) = f^h(\mathbf{x}), \tag{2.2}$$

where $\mathbf{x}_p$ is the position of particle p ($p = 1,\ldots,N$) and $f_p = f(\mathbf{x}_p)$ its *intensity*. The kernel functions $\zeta(\mathbf{x};\mathbf{x}_p)$ may be different on each particle, as indicated by parameterizing them with the particle position $\mathbf{x}_p$ (see details in Appendix A). They are assumed to have compact support of radius $r_{c,p}$ and are rescaled to characteristic width ϵ_p as: $\zeta_{\epsilon_p}(\mathbf{x};\mathbf{x}_p) = \zeta(\mathbf{x}/\epsilon_p;\mathbf{x}_p)$. The *cutoff radii* $r_{c,p}$ are an additional property of the particles and different particles can have different cutoff radii. The *core sizes* ϵ_p are scaling parameters that define the local spatial resolution of the method in the neighborhood of particle *p*. They can also be different on each particle, depending on the local resolution required. However, ϵ_p and $r_{c,p}$ are of the same order of magnitude and proportional to each other.

The dynamics of the particles is governed by the following system of ODEs:

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p, t) = \mathbf{u}_p(t), \tag{2.3a}$$

$$\frac{df_p}{dt} = \mathbf{u}_p \cdot \nabla f_p + \frac{\partial}{\partial t} f_p = \mathcal{L}^h(f_p, t) - f_p \nabla^h \cdot \mathbf{u}, \tag{2.3b}$$

where $\mathcal{L}^h$ is a discrete approximation of the differential operator $\mathcal{L}$, expressed in the form:

$$\mathcal{L}^h(f_p, t) = \sum_{q=1}^{N} (f_q \pm f_p)\eta_p(\mathbf{x}_q). \tag{2.4}$$

The operator kernels η_p (not to be confused with the basis functions $\zeta_{\epsilon_p}(\mathbf{x};\mathbf{x}_p)$ above) are discretized versions of the generalized integral operators proposed by Eldredge et al. [17] and depend on the desired order of accuracy (see [39] and Appendix A).

In this paper we discretize the strong form of Eq. (2.1) on particles, rather than its weak form. This avoids defining and evolving particle volumes and renders particle–particle interpolation simpler. However, it leads to more restrictive regularity constraints on the solution than those of a weak-form discretization, and it hampers conservativeness of the method.

2.2. Approximation of derivatives and particle–particle interpolation

The present adaptive-resolution method relies on discretizing the function *f* on irregularly distributed particles with varying core sizes. These particles self-organize by means of pseudo forces in order to rearrange according to the required resolution. After this adaptation, we need to interpolate the particle intensities f_p from the old set of particles (circles in Fig. 2.1) to the new one (crosses in Fig. 2.1) and determine the temporal change of f_p by approximating the right-hand side of Eq. (2.3b). We obtain a consistent approximation of derivatives of *f* on arbitrary distributions of particles with varying core sizes using DC-PSE operators [39], which rely on solving a small[1] linear system of equations at each particle to determine the kernel weights. After interpolating the f_p values from the old particles to the new ones, there are two ways the right-hand side of Eq. (2.3b) can be computed in a collocation setting: (i) using the new set of particles as both source and collocation points (dashed arrows in Fig. 2.1) or (ii) using the old set of particles as source points, but the new set as collocation points (solid arrows in Fig. 2.1). We denote by *collocation points* the particles at whose location the operator is evaluated, i.e., where the derivative approximation is computed. The *source points* are the surrounding particles whose f_p values are used to do so.

[1] The number of equations is given by the number of moment conditions that are to be satisfied and hence by the order of accuracy of the operator.

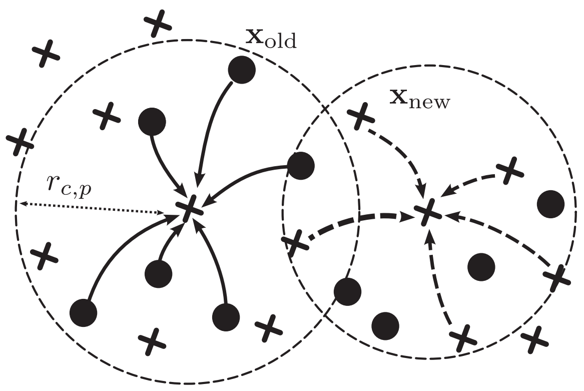

Fig. 2.1. Interpolation of function values from the old set of particles (circles) to a new set of particles (crosses). After interpolation, the differential operators can be approximated either by using the values on the old particles (left circle, solid arrows), or the values on the new particles (right circle, dashed arrows). While the two ways are algebraically equivalent, they differ in computational cost when using DC-PSE operators; see main text for details.

The two ways are algebraically equivalent for corresponding kernel choices. In fact, in both cases the kernels for derivate approximation and interpolation could be combined into one, as has been done by Wee and Ghoniem [45] for the case where $\mathcal{L}$ is the Laplacian and the particles are remeshed. The two ways, however, differ with respect to their computational cost when using DC-PSE operators: Variant (i) uses two kernels with different sets of source particles, hence requiring two different systems of linear equations to be solved on each particle. In variant (ii) both kernels use the same set of source particles and their weights can be determined from the same system of linear equations. The computational cost of the second way is thus about half of that of the first way. We hence prefer the second way, but do not combine the two kernels into one for the sake of clarity and simplicity of the presentation.

In order to be able to use the values of the old particles to approximate derivatives on the new particles, the DC-PSE operators need to be evaluated at off-particle locations, i.e., at collocation points that are not in the set of source points. This requires the zeroth-order moment of the DC-PSE kernels to vanish. With DC-PSE operators this is possible since the zeroth-order moment is a free parameter that can be used to tune the stability properties of the operators [39]. Setting the zeroth-order moment to zero and evaluating the operators at off-particle locations makes DC-PSE a particle-analog of derivative-reproducing kernel (DRK) Galerkin collocation methods [12,46,44], which are conceptually related to moving least-squares (MLS) schemes [26,6].

The kernel for the zeroth derivative can be used to evaluate the function f itself at arbitrary locations, also between particles. We exploit this to construct particle–particle interpolation schemes that satisfy the same moment conditions as the derivative approximations. However, interpolating f_p from one set of irregularly distributed particles to another additionally requires the kernel to be *interpolating*, i.e., to satisfy the Kronecker delta property at the particle locations (see Appendix A for details).

Under some mild assumptions (see Appendix B) about the smoothness of f and the regularity of the particle distribution, upper bounds for the local approximation errors can be expressed in terms of the DC operator's order of accuracy m, the local inter-particle spacing h_p, and the magnitude of the derivatives of f.

Chen et al. [12] report the following error estimate for the interpolant f^h defined by Eqs. (2.2) and (A.9):

$$|f - f^h|_{\mathbf{x}=\mathbf{x}_p} \leqslant C h_p^m |f|_{W_\infty^m(\mathcal{B}_p)}, \tag{2.5}$$

where $\mathcal{B}_p$ is the ball of radius $r_{c,p}$ around particle p, C is a positive constant, and $|f|_{W_\infty^m(\Omega)} = \max_{|\boldsymbol{\alpha}|=m} \|\partial^{|\boldsymbol{\alpha}|} f / \partial \mathbf{x}^{\boldsymbol{\alpha}}\|_{L^\infty(\Omega)}$. We use the notation:

$$\frac{\partial^{|\boldsymbol{\alpha}|} f}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}} \tag{2.6}$$

for differential operators in $\mathbb{R}^d$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ is a multi-index and $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \cdots + \alpha_d$.

The point-wise truncation error for differential operators of the type $\mathcal{L} = \partial^{|\boldsymbol{\alpha}|} f / \partial \mathbf{x}^{\boldsymbol{\alpha}}$ is bounded by (see Appendix B):

$$|\mathcal{L}(f) - \mathcal{L}^h(f)|_{\mathbf{x}=\mathbf{x}_p} \leqslant C h_p^{m-|\boldsymbol{\alpha}|} |f|_{W_\infty^m(\mathcal{B}_p)}. \tag{2.7}$$

2.3. Self-organizing Lagrangian particles

The point-wise error bounds for the approximation of derivatives and for particle–particle interpolation stated above motivate a spatially adapted resolution (i.e., h is a function of space) such as to equi-distribute the error across all particles. This would then result in the minimum number of particles needed for these approximations to reach below a certain error level everywhere in the domain. The above operators for approximating derivatives are consistent on almost any particle distribution [39], except those for which the associated Vandermonde matrix **V** is not invertible (see Eq. (A.3)), in which case we randomly displace or insert particles until **V** becomes invertible.

The self-organization of the particles is driven by pseudo forces that arise from pairwise particle–particle interactions with a pair potential that is scaled with the local target resolution. After a short relaxation time, the particle density and their core sizes thus follow the spatial features of the field function f such as to approximately equi-distribute the approximation error (see also Appendix B). Particles further self-organize into configurations that are non-degenerate with respect to the above scattered-points interpolation scheme, ensuring that the field is well sampled everywhere.

We describe below how such self-organization potentials can be constructed from a target resolution field (monitor function), and how we handle dynamic insertion and removal of particles in regions where this is needed.

2.3.1. Resolution field

We denote by $\widetilde{D}(\mathbf{x})$ the desired local target resolution of the spatial discretization. This defines the smallest scales that ought to be resolved by the numerical approximation in the neighborhood of **x**.

In order to be able to determine the locally required resolution at every point in the computational domain, $\widetilde{D}(\mathbf{x})$ needs to be expressed as a function of known or computable properties of f. Although many choices are possible, we here choose the simple form:

$$\widetilde{D}(\mathbf{x}) = \frac{D_0}{\sqrt{1+|\nabla f(\mathbf{x})|^2}}, \tag{2.8}$$

which is often used as a monitor function in adaptive-resolution methods, including moving-mesh methods. $D_0 > 0$ is a user-defined parameter that sets the coarsest resolution in the computational domain and hence an upper bound on the inter-particle spacing h. We refer to Appendix B for a discussion of how the choice of $\widetilde{D}$ influences the accuracy of the method.

Each particle p is assigned the minimum value of $\widetilde{D}$ over all its neighbors within a certain cutoff radius:

$$D_p = D(\mathbf{x}_p) = \min_{|\mathbf{x}_q-\mathbf{x}_p|\leqslant r_{c,p}} \widetilde{D}(\mathbf{x}_q). \tag{2.9}$$

The cutoff radius of particle p is $r_{c,p} = D_p r^*$, where $r^* > 1$ is a global parameter that depends on the kernels used for approximating derivatives and for particle–particle interpolation. See Section 2.3.4 for how r^* is determined. The core size of particle p is set to $\epsilon_p = r_{c,p} = D_p r^*$.

2.3.2. Self-organization potential

The pairwise interaction potential for the adaptation pseudo forces between particles p and q is scaled to the locally required resolution as:

$$V_{pq} = D_{pq}^2 V(|\mathbf{x}_p - \mathbf{x}_q|/D_{pq}), \tag{2.10}$$

where $D_{pq} = \min(D_p, D_q)$ and $V(r)$ is a normalized symmetric pair potential. This form ensures that the adaptation pseudo forces scale with D_{pq} and that the length scale of the potential corresponds to the local resolution requirement.

Locally minimizing the total potential energy of the particles:

$$W(\mathbf{x}_1,\ldots,\mathbf{x}_N) \equiv \sum_p \sum_q V_{pq} \tag{2.11}$$

with respect to the particle positions $(\mathbf{x}_1,\ldots,\mathbf{x}_N)$ leads to a distribution of particles that provides a spatial resolution close to the target resolution $\widetilde{D}(\mathbf{x})$, such that the characteristic inter-particle spacing h_p near a particle at $\mathbf{x}_p$ is smaller than or equal to $\widetilde{D}(\mathbf{x}_p)$.

Many choices are possible for $V(r)$, but only those that lead to low-discrepancy particle distributions where the distance between any pair of nearest neighbors is $\approx D_{pq}$ should be considered. In special cases, self-organization potentials with provable ground states can be designed in a principled way [43,13]. This has, e.g., been used to make particles relax to Cartesian [33,35] and hexagonal [34] lattices at minimum energy. How the systematic design of self-organization potentials could be generalized to irregular adaptive-resolution cases, however, is an open problem.

We propose two parameter-free self-organization potentials from well-studied classes:

$$V_1(r) = 0.8\cdot 2.5^{1-5r} - 2.5^{-4r}, \tag{2.12}$$

$$V_2(r) = r^{-2}/2 + r^{-6}/6, \tag{2.13}$$

which are plotted in Fig. 2.2. V_1 is a *h-stable* attractive/repulsive Morse potential [30,16]. V_2 is a purely repulsive potential. We modify both potentials to linearly decay below $r = 0.5$. This ensures that particles that are too close to each other fuse (circles in Fig. 2.2; see also Section 2.3.4).

We illustrate the qualitative differences between these potentials by equilibrating a fixed number of particles in the 2D unit square with a high-resolution field $\widetilde{D} = 0.01$ inside a square of size 0.1×0.1 at the center, and a lower resolution

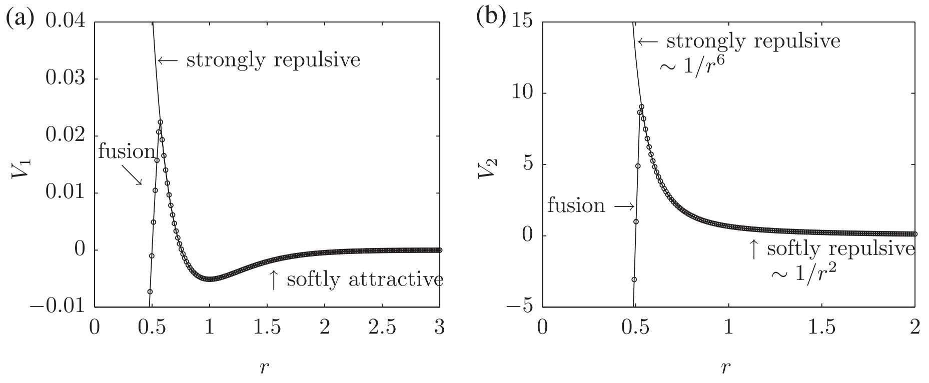

Fig. 2.2. Examples of normalized self-organization potentials. (a) Stable attractive/repulsive Morse potential $V_1(r)$ (Eq. (2.12)) to be used in open domains, (b) Purely repulsive potential $V_2(r)$ (Eq. (2.13)) for finite and periodic domains. The lines with symbols show the modified potentials with strong short-range attraction to induce particle fusion in over-resolved regions.

$\widetilde{D} = 0.1$ elsewhere. The ratio between the high and low resolutions is chosen small for the sake of visualization. At the edges of the high-resolution square $\widetilde{D}(\mathbf{x})$ abruptly jumps by a factor of 10. For the potential V_2 we also consider the same setting with a gradually varying resolution field. Initially, all particles are placed inside the high-resolution region and let to self-organize to steady state.

The resulting particle arrangements after energy minimization by gradient descent are shown in Fig. 2.3. They illustrate how particles self-organize in regions of uniform resolution (the small high-resolution square) and in regions where the resolution abruptly or gradually changes (the edges of the high-resolution square).

The h-stable attractive/repulsive Morse potential V_1 keeps the local particle density constant [16], with a characteristic local spacing of D_{pq}, and covers an increasingly large domain with increasing particle number (Fig. 2.3(a) and (b)). This is well suited for free-space boundary conditions where the support of the particle distribution can vary.

The purely repulsive potential V_2 is convex and leads to an energy that can be robustly minimized, causing particles to rapidly spread over the entire computational domain. This is desired in finite and periodic domains, where this potential ensures that the ratio between the distance of two particles and their core sizes reflects the local resolution requirement. The absolute value of the inter-particle spacing can be adjusted by changing the total number of particles in the computational domain. However, this only works well if the ratio between the target resolutions $\widetilde{D}$ of any two nearest-neighbor particles is less than r^*. This follows from Eq. (2.9) together with the neighborhood definition given in Section 2.3.3. If the resolution field varies faster than that, the particles inside the high-resolution region tend to be expelled into the low-resolution region. This leads to the high-resolution region being under-resolved and the low-resolution region over-resolved, as illustrated in Fig. 2.3(c) and (d), where the resolution discontinuously jumps by a factor of 10 at the edge of the high-resolution square. When the jump in the resolution field $\widetilde{D}$ is replaced by a linear decay with a slope of less than $(r^* - 1)$ (which implies that the ratio of $\widetilde{D}$ is less than r^* between nearest-neighbor particles), the local resolution is correct everywhere. This is shown for the same case in Fig. 2.3(e) and (f), where the two squares mark the beginning and the end of the linear slope.

In summary, we always use the self-organization potential V_1 in open domains with free-space boundaries or when the resolution field has gradients $|\nabla\widetilde{D}| > (r^* - 1)$. The potential V_2 is used in finite domains and domains with periodic boundary conditions when $|\nabla\widetilde{D}| < (r^* - 1)$.

2.3.3. Neighbor lists

The cutoff radius of the operator and function approximations presented here is a function of space. Defining the neighbors of particle p, denoted by the index set $\mathcal{N}_p$, as those particles within a ball of radius $r_{c,p}$ around $\mathbf{x}_p$ could hence lead to the situation where particle p is a neighbor of particle q, but not vice versa. Such asymmetric neighbor lists are undesirable for the computational efficiency of the method. In order to guarantee symmetric neighbor lists, we consider as neighbors of particle p all particles q at a distance less than $\min(r_{c,p}, r_{c,q})$ from particle p (see Fig. 2.4(a)). This ensures that $q \in \mathcal{N}_p \iff p \in \mathcal{N}_q$ and that particles in coarsely resolved regions ($r_{c,p}$ large) do not interact with potentially large clusters of particles in nearby finely resolved regions where $r_{c,q}$ is small. Such neighbor lists can efficiently be built using adaptive-resolution cell lists [3].

2.3.4. Insertion/removal of particles and choice of r^*

Finding the global minimum of the potential energy of a large collection of interacting particles is rarely feasible and always computationally expensive. Finding a particle distribution that locally minimizes the energy is comparatively much easier. For sufficiently smooth self-organization potentials, simple gradient descent algorithms are able to robustly approach local minima of the potential energy, which is sufficient for the present method.

The number of iterations required by a gradient descent algorithm depends on how far the initial condition is from the energy basin where the algorithm terminates. If the initial particle distribution is very different from the final, adapted one,

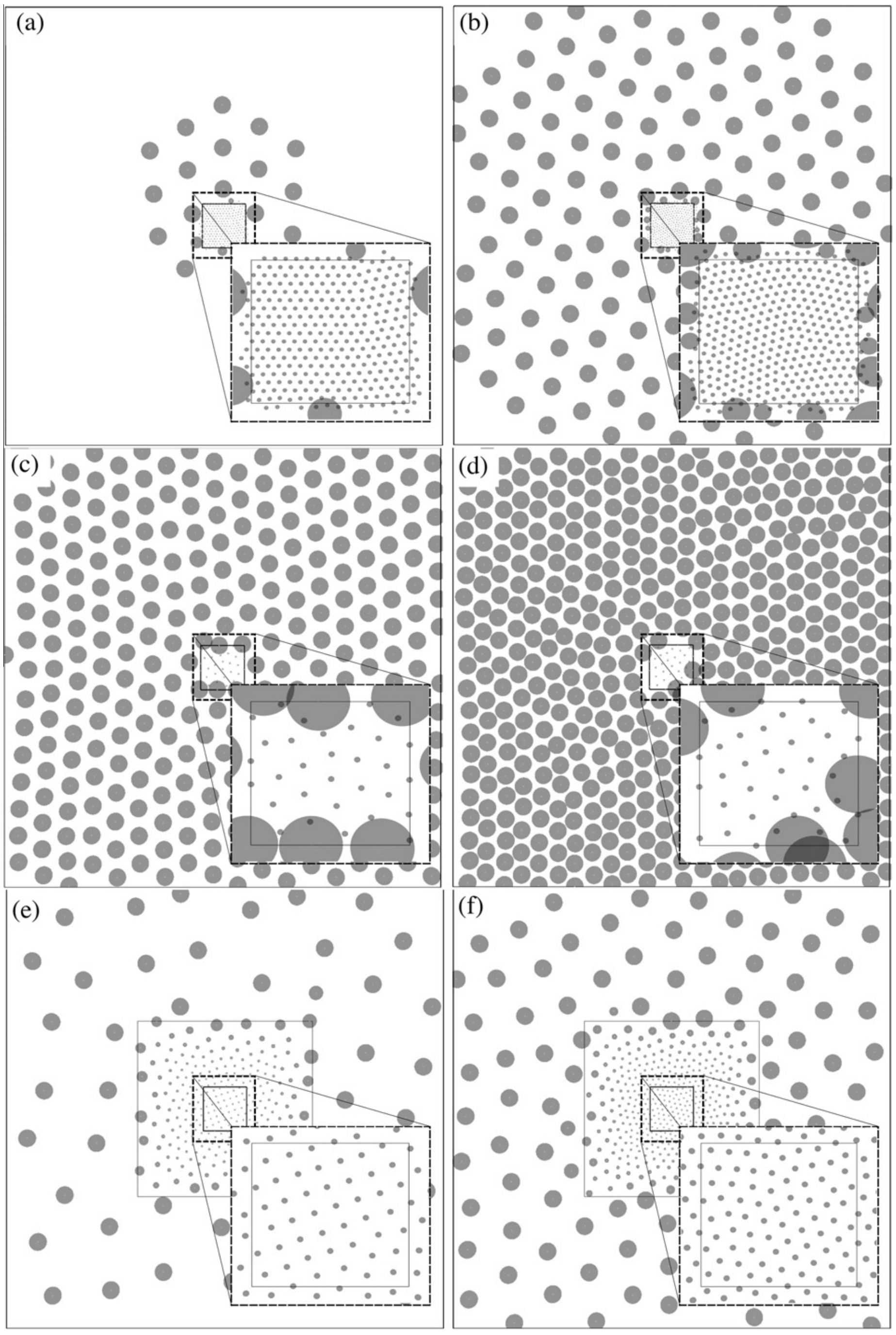

Fig. 2.3. Examples of steady-state particle distributions under different self-organization potentials in the 2D unit square: (a and b) $V_1(r)$ (Eq. (2.12)), (c and d) $V_2(r)$ (Eq. (2.13)) with a discontinuous resolution field, (e and f) $V_2(r)$ (Eq. (2.13)) with a continuous resolution field; Left column: $N = 324$ particles, right column: $N = 529$ particles. Gray disks indicate the core sizes of the particles, scaled down by a factor 10 for better visualization.

the number of iterations required can be prohibitively large. This may happen, for example, when an initially uniform solution field develops steep gradients in a small region of the computational domain. In order to resolve those gradients, many particles from across the computational domain need to move to the region where refinement is needed. This major particle migration would quickly become the bottleneck in large simulations. Moreover, if fine scales in the solution develop and disappear over time, not only the distribution of particles, but also their total number has to be adapted in order to maintain the same accuracy.

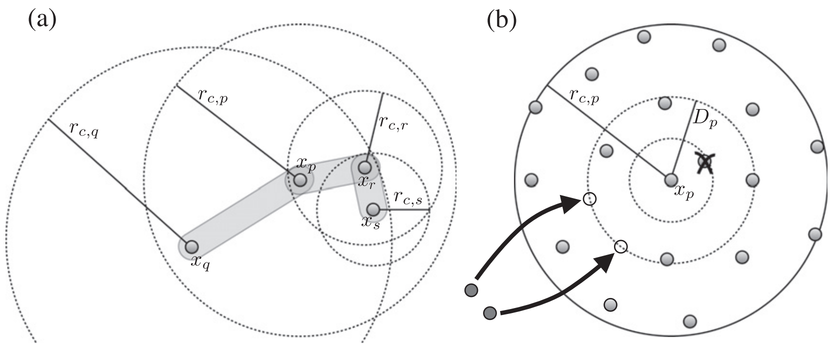

Fig. 2.4. (a) Illustration of the present definition of neighborhood. Particles that are neighbors of each other are grouped together by shaded links. This definition of neighborhood ensures symmetric neighbor lists. (b) Insertion and removal of particles. Particles that are too close to others are removed and new particles are inserted in under-resolved regions; see main text for details.

We achieve faster energy minimization and adaptive particle numbers by dynamically removing particles from over-resolved regions and inserting new ones in under-resolved regions. We do this by fusing particles that are too close to each other and generating new particles in regions where the already existing ones have fewer neighbors than a critical number (see Fig. 2.4(b)). We now outline how this critical number is determined and what "too close" means in the context of the present method.

We wish that near any particle p, the neighboring particles locally adapt toward an irregular distribution with characteristic spacing D_p. The first requirement that needs to be fulfilled is that each particle must have a minimum number of N^* neighbors within its cutoff radius $r_{c,p}$ in order for the discretization to be consistent. This number is equal to the number of moment conditions that need to be fulfilled by the discretized operators, which depends on the order of accuracy of the spatial discretization of the elliptic operator $\mathcal{L}$ in Eq. (2.1), see Appendix A. For 4th-order interpolation and 2nd-order approximation of the Laplacian, e.g., $N^* = 10$ in 2D and $N^* = 20$ in 3D. The actual number of neighbors of any particle is ideally identical to N^* as any additional neighbors increase the computational cost without increasing the accuracy of the discretization. We thus take N^* as the critical number of neighbors for the particle insertion/removal strategy.

In a first approximation, we start the algorithm by setting r^* such that $|\mathcal{N}_p| = N^*$ for particles arranged on a triangular lattice with spacing D_p. For a triangular lattice in 2D, for example, $r^* = 1, \sqrt{3}, 2, \sqrt{7}$, or 3 leads to $|\mathcal{N}_p| = 6, 12, 18, 30$, or 36, respectively. In 3D, $r^* = 1, \sqrt{2}, \sqrt{3}$, or 2 leads to $|\mathcal{N}_p| = 12, 18, 42$, or 54. The local particle density is then adapted during energy minimization by inserting particles in regions where $|\mathcal{N}_p| < N^*$ and fusing particles that are closer to each other than $D_{pq}/2$, which means that they are too close for the local resolution required. The total number of particles N in a simulation is hence not a free parameter of the method, but is determined adaptively by the self-organization algorithm at run-time.

We find that this insertion/removal strategy is effective in dealing with global changes in the required resolution, allowing the gradient descent minimizer to reach a local minimum within few iterations (typically less than 10).

If necessary, N can be bounded from above by imposing a lower bound on the resolution field $\widetilde{D}(\mathbf{x})$. For $D_0 \downarrow 0$, the number of particles contained in a d-dimensional domain Ω scales as $N \propto \int_\Omega \widetilde{D}(\mathbf{x})^{-d} d\mathbf{x}$ and may grow arbitrarily large. This is, e.g., the case when the solution develops infinitely steep gradients. In such cases, we impose a minimum threshold $D_{\min}$ on the resolution field as follows: $\widetilde{D}(\mathbf{x}) \leftarrow \max(\widetilde{D}(\mathbf{x}), D_{\min})$. This guarantees that N does not exceed $N_{\max} \approx |\Omega|/D_{\min}^{-d}$. However, this also means that the field f will be under-resolved in regions where $\widetilde{D} < D_{\min}$, and that the desired accuracy cannot be guaranteed there.

2.4. Boundary conditions

We demonstrate the present method mainly on problems with free-space and periodic boundary conditions. Other types of boundary conditions, however, can be treated in the standard ways.

Homogeneous Neumann and Dirichlet boundary conditions can be imposed using mirror particles in a small neighborhood outside the computational domain (method of images). These mirror particles are only used to evaluate the right-hand side of Eq. (2.3b). They are not considered for the adaptation pseudo forces, nor for particle–particle interpolation. Instead, they are re-generated after each particle self-organization. When computing the adaptation pseudo forces, the boundaries of the domain are treated as rigid walls that confine the particles to the computational domain.

Inhomogeneous and mixed boundary conditions can be enforced by locally modifying the intensities of the particles in the neighborhood of the boundary [24] or by treating them as artificial reaction terms [36].

3. Implementation

We first describe how the pseudo forces for resolution adaptation are computed from the self-organization potential. Then, we give an overview of the workflow of the overall algorithm.

3.1. Steepest descent on W

Self-organization of the particle positions and core sizes is driven by pseudo forces that locally decrease the total potential energy W of the interacting particle system. After fusing particles that are closer to each other than $D_{pq}/2$ and inserting new particles where needed, a single step of a gradient descent is performed. This displaces each particle by a step $\mathbf{w}_p$ against the gradient of the interaction potential energy W:

$$\mathbf{w}_p = -\gamma \partial W(\mathbf{x}_1, \ldots, \mathbf{x}_N)/\partial \mathbf{x}_p, \tag{3.1}$$

$$= -\gamma \left[\sum_{q \in \mathcal{N}_p} \partial V_{pq}/\partial \mathbf{x}_p + \sum_{q \text{s.t.} p \in \mathcal{N}_q} \partial V_{qp}/\partial \mathbf{x}_p \right], \tag{3.2}$$

where the step size γ is determined by a line search.

Since both the potential V and the neighborhood relations are symmetric, the gradient descent flow simplifies to:

$$\mathbf{w}_p = -2\gamma \sum_{q \in \mathcal{N}_p} \partial V_{pq}/\partial \mathbf{x}_p. \tag{3.3}$$

Using the chain rule and the rescaled form of V_{pq} from Eq. (2.10) leads to:

$$\mathbf{w}_p = -2\gamma \sum_{q \in \mathcal{N}_p} D_{pq} \left[V'(r)\mathbf{e}_{pq} + (2V(r) - rV'(r)) \nabla_{\mathbf{x}_p} D_{pq} \right]_{r = r_{pq}/D_{pq}}, \tag{3.4}$$

where $\mathbf{e}_{pq}$ is the unit vector pointing from particle p to particle q and r_{pq} is the distance between particles p and q. Note that once the field f is properly resolved (i.e., the total number of particles N is sufficiently large), $\nabla_{\mathbf{x}_p} D_{pq} \ll 1$ and the second term in Eq. (3.4) can be neglected.

During particle self-organization, the insertion/removal and steepest descent algorithms are iterated until all particles have at least N^* neighbors and the stopping criterion $\max_p \max_{q \in \mathcal{N}_p} (D_{pq}/r_{pq}) \leqslant d_c$ is met. This ensures that no two particles are too close to each other. We find that a value of $d_c = 2.5$ leads to a small number of iterations (typically less than 10) while ensuring that the local density of particles matches the target resolution. Choosing a smaller value for d_c leads to more regular particle distributions at the expense of a larger number of gradient-descent iterations.

3.2. Overall workflow of the method

The whole simulation workflow starts from determining a good initial particle distribution and then enters a time loop where Eq. (2.3) is solved and the particle distribution is continuously adapted.

3.2.1. Initialization

There are several possibilities of placing the particles at the beginning of a simulation: Particles can be initialized on a (adaptive-resolution) Cartesian mesh, placed uniformly at random in the computational domain, or sampled from a probability density function that is proportional to the initial monitor function.

In practice, we find that random placement of a fixed number of particles and subsequent self-organization to the initial condition is sufficient. This leads to a simple and robust initialization strategy where the number of particles required to represent the initial condition on a certain error level is determined by the adaptation algorithm and does not need to be know *a priori*. Fig. 3.1 illustrates this initialization procedure for the example of a Gaussian pulse (see Section 4.3 below). We initially randomly place a fixed number of 800 particles with uniform core sizes (Fig. 3.1(a)). We then iterate the adaptation algorithm (step 4(e) of the algorithm below) until it terminates, without performing any time steps. Fig. 3.1(b) and (c) show two different resulting particle distributions for $D_0 = 0.2$ and $D_0 = 0.05$, respectively. The total resulting number of particles is 3027 in the first case and 27 631 in the second.

This resulting particle distribution is then used to represent the initial condition of the problem before entering time stepping.

3.2.2. Time stepping

The system of ODEs in Eq. (2.3) is solved until final time $t = T$ using any time-stepping scheme. In each time step, we perform the following operations to evaluate the right-hand side of the discretized equations and re-organize the particles:

1. Choose the time-step size δt from the CFL condition based on the globally smallest value of the inter-particle spacing.
2. Advect the particles with the Lagrangian velocity $\mathbf{u}$ between t and $t + \delta t$.
3. Construct the neighbor lists.
4. If it is time to re-organize the particles, do:
   (a) Construct the DC-PSE operators.
   (b) Evaluate the field derivatives and $\widetilde{D}(\mathbf{x}_p)$.
   (c) Compute D_p using Eq. (2.9).

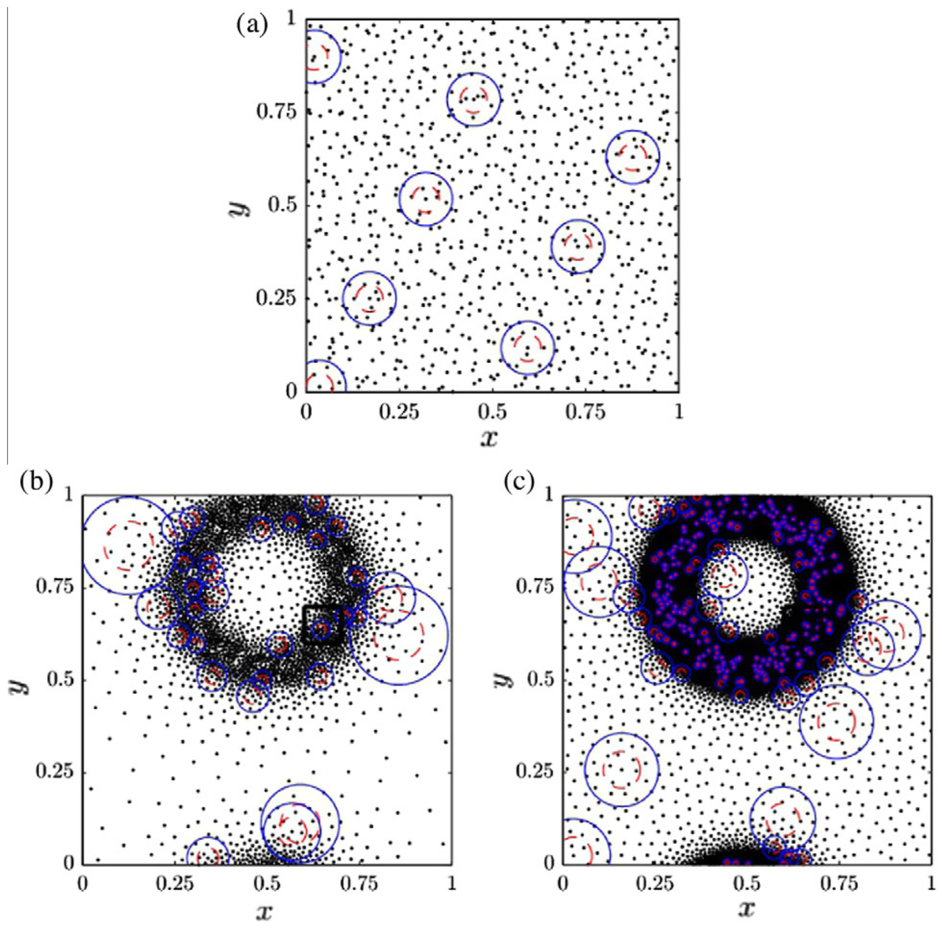

Fig. 3.1. Adaptation of the particle distribution to the initial condition given in Eq. (4.6). (a) Initial set of 800 uniformly randomly placed particles of equal size. (b) Particle distribution after self-organization with $D_0 = 0.2$ ($N = 3027$ particles, 7 gradient-descent iterations). (c) Particle distribution after self-organization with $D_0 = 0.05$ ($N = 27631$ particles, 21 gradient-descent iterations). Circles represent D_p (dashed) and $r_{c,p}$ (solid) for a sample set of particles.

(d) Save the set of points $\mathbf{x}_p$ as $\mathbf{x}_p^{\text{old}}$.
(e) Adapt the particles to a new distribution $\mathbf{x}_p{}^{\text{new}}$ by iterating:
  i. Fuse particles where $|\mathbf{x}_q - \mathbf{x}_p| < D_{pq}/2$.
  ii. Insert new particles at random locations in the neighborhood of particles that have fewer than N^* neighbors.
  iii. Construct the neighbor lists within $\mathbf{x}_p^{\text{new}}$ and between $\mathbf{x}_p^{\text{new}}$ and $\mathbf{x}_p{}^{\text{old}}$.
  iv. Compute D_p of $\mathbf{x}_p^{\text{new}}$ by first-order interpolation from D_p of $\mathbf{x}_p^{\text{old}}$.
  v. Adapt the cutoff radii $r_{c,p}$ and core sizes ϵ_p.
  vi. Compute the total energy of the self-organization potential and its gradient.
  vii. Perform a line search for the gradient-descent step size and move the particles by one step down the energy gradient.
  viii. If the stopping criterion of the gradient descent is met and every particle has at least N^* neighbors, go to step 4(f). Else go to step 4(e)-i.

(f) Compute on each particle the interpolation kernels ζ_p and the kernels of the discretized right-hand side of Eq. (2.3b), re-using the matrix inverse from step 4(a).
(g) Interpolate the particle intensities f_p from $\mathbf{x}_p^{\text{old}}$ to $\mathbf{x}_p^{\text{new}}$.

5. Construct the DC-PSE operators for the right-hand side of Eq. (2.3b) and evaluate them using $\mathbf{x}_p^{\text{old}}$ as source points and $\mathbf{x}_p^{\text{new}}$ as collocation points. Update the particle intensities f_p.
6. Advance time $t \leftarrow t + \delta t$ and loop back to step 1, unless $t > T$.

4. Numerical experiments and benchmarks

We present an array of numerical experiments and benchmarks that are designed to demonstrate the capabilities and limitations of the present method. The first benchmark in Section 4.1 demonstrates the consistency of the operator approximation and particle–particle interpolation schemes. The second benchmark considers a pure advection problem for a passive scalar. This test case was also considered by Bergdorf and Koumoutsakos [7], which allows comparing the results. The third test case in Section 4.3 adds diffusion and considers an advection–diffusion problem with known analytical solution. It also compares the behavior of the present method in 2D and 3D. The fourth test case in Section 4.4 is the 2D unsteady Burgers equation, which serves as a benchmark for nonlinear transport problems. The fifth test case in Section 4.5 demonstrates the efficiency and accuracy of the present method for the real-world application of the nonlinear 2D Buckley–Leverett problem. The sixth problem considers curvature-driven surface refinement in 3D. We conclude this section by analyzing and commenting on the computational cost of the method. In all cases, convergence is shown in terms of the effective inter-particle spacing $h = (|\Omega|/N)^{1/d}$, where $|\Omega|$ is the volume of the computational domain, N the actual number of particles used, and d the space dimension. In order to illustrate the adaptation capabilities of the present method, we use test cases that develop steep gradients in small parts of the computational domain. The actual number of particles hence rarely exceeds 10^5, since the larger parts of the computational domains require only low resolution.

Problems 1 to 5 are defined on finite or periodic domains with smooth resolution fields and hence use the self-organization potential V_2 (Eq. (2.13)) plotted in Fig. 2.2(b). Problem 6 has free-space boundaries and uses the self-organization potential V_1 (Eq. (2.12)) shown in Fig. 2.2(a).

4.1. Consistency of derivative approximation and interpolation

We assess the convergence of the interpolation kernels and of the discretized Laplace operator on the test function:

$$f(x,y) = \tanh\left(\frac{x^2+y^2-0.2^2}{0.01}\right) \tag{4.1}$$

in the domain $\Omega = [-1,1]^2$. This test function has a steep sigmoidal transition of tunable slope. Successively reducing the coarse-scale parameter D_0, we measure the errors as follows: For an initial value $D_0^{(0)}$, particles are adapted to the field f and their intensities are set to the exact values: $f_p^{(0)} = f\left(\mathbf{x}_p^{(0)}\right)$. The particles are then adapted to a finer resolution $D_0^{n+1} \leftarrow 0.95D_0^n$ using the self-organization scheme described in Section 2.3, and the new function values $f_p^{(n+1)}$ are interpolated from the old values $f_p^{(n)}$. The point-wise interpolation error is then computed as $\left|f_p^{(n+1)} - f\left(\mathbf{x}_p^{(n+1)}\right)\right|$. The matrices that have to be inverted for each particle to compute the interpolation kernels are re-used to compute an approximation Δ^h of the Laplacian using the old particles $\left\{\mathbf{x}_p^{(n)}, f_p^{(n)}\right\}$ as source points. The point-wise error of this approximation is then computed as $\left|\Delta^h f_p^{(n+1)} - \Delta f\left(\mathbf{x}_p^{(n+1)}\right)\right|$. Finally, the particle intensities are re-set to the exact values $f_p^{(n+1)} = f\left(\mathbf{x}_p^{(n+1)}\right)$ and the whole procedure is repeated for $n \leftarrow n+1$.

All kernels are computed with $m = 4$, leading to fourth-order convergence of the interpolation functions and second-order convergence of the Laplacian approximation, as verified in Fig. 4.1(a). D_0 decreases from 0.4 to 0.006 and the number of particles increases from 10^2 to 2×10^5, approximately. Each value of D_0 corresponds to a different set of particles to represent the test function f. The ruggedness of the convergence plot in Fig. 4.1 for low resolutions can thus be interpreted as the sensitivity of the error norm to the underlying particle distribution. This is not specific to the present method. The same effect in the L_∞-norm of the error also occurs, e.g., in Cartesian finite-difference schemes when rotating the mesh. An example particle distribution created by the self-organization process is shown in Fig. 4.1(b).

4.2. Advection of a passive scalar

We illustrate the resolution adaptivity of the present method by considering 2D advection of a passive scalar f by a given velocity field $\mathbf{u}$, which can be written in non-conservative form as:

$$\frac{\partial f}{\partial t} + \mathbf{u}\cdot\nabla f = 0. \tag{4.2}$$

In order to allow comparison with the Lagrangian wavelet-particle method of Bergdorf and Koumoutsakos [7], we consider the same test problem they did. This comprises the advection of an initial "blob":

$$f(x,y,0) = \sum_{i=-1}^{i=1}\sum_{j=-1}^{j=1} \frac{\sqrt{2}}{2}\ \mathrm{erf}\left(c_1\left(c_2 - \sqrt{(x-x_0+i)^2+(y-y_0+j)^2}\right)+1\right) \tag{4.3}$$

with $(x_0,y_0) = (0.5,0,75)$, $c_1 = 21.269446$, and $c_2 = 0.16811704$ by the divergence-free velocity field:

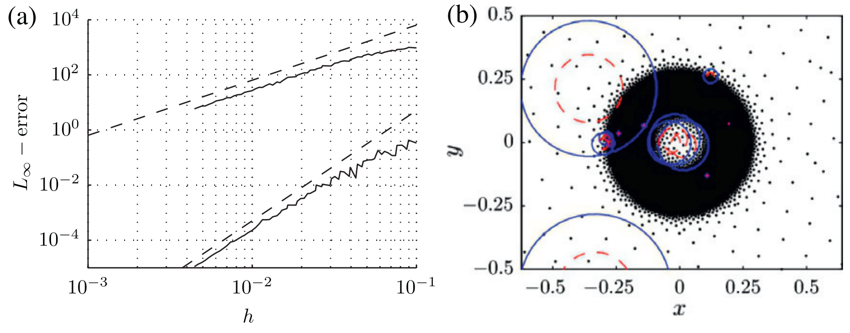

Fig. 4.1. (a) Errors in the function approximation ($\max_p|f^h(\mathbf{x}_p) - f(\mathbf{x}_p)|$, lower curve) and in the approximation of the Laplacian ($\max_p|\Delta^h f^h(\mathbf{x}_p) - \Delta f(\mathbf{x}_p)|$, upper curve) for the test function given in Eq. (4.1), plotted against the average inter-particle spacing $h = \sqrt{|\Omega|/N}$ in 2D. Dashed lines indicate second- and fourth-order convergence. (b) Example distribution of $N = 40698$ particles adapted to the test function in Eq. (4.1); Circles represent D_p (dashed) and $r_{c,p}$ (solid) for a sample set of particles.

$$\mathbf{u} = 2\cos(\pi t/T)\begin{pmatrix} -\sin^2(\pi x)\sin(\pi y)\cos(\pi y) \\ \sin^2(\pi y)\sin(\pi x)\cos(\pi x) \end{pmatrix} \tag{4.4}$$

in the computational domain $\Omega = [0,1]^2$ with doubly periodic boundary conditions. We simulate the time evolution of the advected field f up to final time $T = 2.5$. The analytical solution at this final time is identical to the initial condition given in Eq. (4.3). The maximum distortion of the field f occurs at $t = T/2$ and is shown in Fig. 4.2(a).

Like Bergdorf and Koumoutsakos [7], we use a fourth-order Runge–Kutta time stepping scheme with $\delta t = 0.025$. Interpolation is performed using fourth-order kernels ($m = 4$ in Eq. (A.2)) and the error is computed as:

$$e(\mathbf{x}) = (f(\mathbf{x},T) - f(\mathbf{x},0))\|f(\mathbf{x},T)\|_\infty^{-1}. \tag{4.5}$$

Fig. 4.2(b) shows the L_∞ norm of $e(\mathbf{x})$ as a function of the average inter-particle spacing $h = 1/\sqrt{N}$. The convergence is fourth-order, as expected. Quantitatively, the errors are competitive with those published for the wavelet-particle method [7] for the same test case.

4.3. Rigid-body rotation with diffusion in 2D and 3D

As an advection–diffusion problem with known analytical solutions in both 2D and 3D we consider the d-dimensional Gaussian pulse:

$$f(\mathbf{x},0) = \exp\left(-Pe\frac{|\mathbf{x}-\mathbf{x}_0|^2}{4}\right), \tag{4.6}$$

initially centered at $\mathbf{x}_0$, diffusing and being advected by rigid-body rotation about the center of the computational domain $\Omega = [-1,1]^d$. This is described by the equation:

$$\frac{\partial f}{\partial t} + \mathbf{u}\cdot\nabla f = \frac{1}{Pe}\Delta f, \tag{4.7}$$

where Pe is the dimensionless Péclet number, and $\mathbf{u}$ is the velocity field of the rigid-body rotation. After one revolution, at $T = 1$, the exact solution is given by

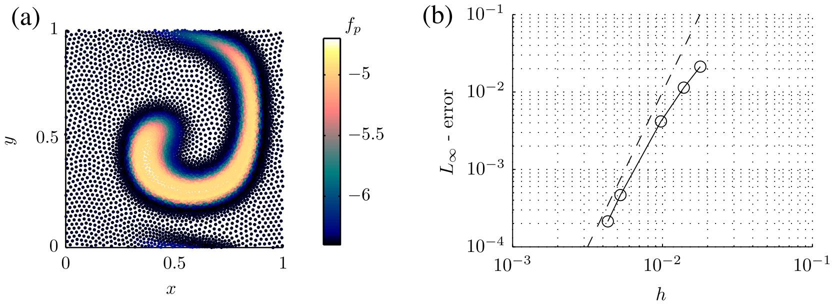

Fig. 4.2. Advection of a passive scalar by the velocity field given in Eq. (4.4) with doubly periodic boundary conditions. Panel (a) shows the particles at the time of maximum distortion, $t = T/2$. Color codes the particle intensities f_p. Panel (b) shows the L_∞ error for the advection of the function given in Eq. (4.3) versus $h = 1/\sqrt{N}$. The dashed line has slope 4.

$$f(\mathbf{x},1) = 2^{-d/2}\exp\left(-Pe\frac{|\mathbf{x}-\mathbf{x}_0|^2}{8}\right). \tag{4.8}$$

All kernels are computed with $m = 4$ and the characteristic width of the initial pulse, $2/Pe$, is such that f is negligible near the boundaries of Ω at all times $t \leqslant T$. The method converges with the expected second-order accuracy, the error being dominated by the diffusion term (the advection term is, in this case, computed exactly). We compare the results with those obtained using a remeshed LPM where advection is also computed exactly and diffusion is simulated using second-order centered finite differences after the particles have been interpolated onto a uniform Cartesian mesh of resolution h using the third-order M_4' interpolation kernel [29]. In both cases time stepping is done using a forward Euler scheme with $\delta t = h_{\min}^2/4$, where $h_{\min}$ is the smallest distance between any two particles ($h_{\min} \equiv h$ for the remeshed LPM).

We first consider the 2D case ($d = 2$) with $\Omega = [-1,1]^2$, $\mathbf{x}_0 = (0.58, 0.02)$, and $\mathbf{u} = 2\pi(y, -x)$. An example particle distribution after self-organization to the initial condition in Eq. (4.6) is shown in Fig. 4.3 for $Pe = 10^4$. The steep gradients of the Gaussian pulse are well resolved. Fig. 4.4(a) shows the maximum point-wise error as defined in Eq. (4.5) at $T = 1$ for different resolutions and for $Pe = 1000$.

Since the present method discretizes the strong form of the governing equation, it does not conserve mass exactly. While the L_∞ error reported in Fig. 4.4 also includes the mass error, and hence is an upper bound on it, we also separately quantify the relative mass loss. We do this by interpolating the particle intensities f_p onto a high-resolution Cartesian mesh (interpolation error is negligible) and evaluating the total relative mass loss:

$$\left(\int f(\mathbf{x},t)\mathrm{d}\mathbf{x} - \int f^h(\mathbf{x},t)\mathrm{d}\mathbf{x}\right)\Big/\int f(\mathbf{x},t)\mathrm{d}\mathbf{x} \tag{4.9}$$

over time. Fig. 4.5 shows the evolution for three different resolutions D_0 and the respective maximum numbers of particles used by the method. In all cases, the loss rate decreases with time as diffusion smoothes out the initially steep gradients in the concentration field.

We illustrate the behavior of the present method in 3D by considering the case $d = 3$ with $\Omega = [-1.5, 1.5]^3$, $\mathbf{x}_0 = (0.58, 0.02, 0.02)$, and $\mathbf{u} = 2\pi(y, -x, 0)$. The method converges with the expected second-order accuracy of the diffusion operator, as shown in Fig. 4.4(b) for $Pe = 100$.

For solutions with large gradients, the finest scales that are resolved by the present method are of length $D_0/\max|\nabla f|$, which in this test case decreases as Pe^{-2}. This length is approximately $0.23D_0$ for $Pe = 100$. In the 3D case we find that the corresponding number of particles required to achieve the same accuracy on a uniform Cartesian mesh is about 50 times larger than when using the present adaptive-resolution method.

4.4. 2D Burgers equation

We demonstrate the performance of the present method on a nonlinear problem by considering the 2D unsteady Burgers equation:

$$\frac{\partial f}{\partial t} + \mathbf{u}\cdot\nabla f = \frac{1}{Re}\Delta f, \tag{4.10}$$

where Re is the Reynolds number and $\mathbf{u} = (f, f)$. We solve Eq. (4.10) subject to $u(x, y, t = 0) = \sin(2\pi x)\cos(2\pi y)$ and doubly periodic boundary conditions in the computational domain $\Omega = [0,1]^2$. For large Re, the solution of the Burgers equation develops steep gradients over time, requiring an increasingly high resolution.

The solution as computed by the present method is shown in Fig. 4.6 at $T = 0.17$ for $Re = 1000$ and $D_0 = 0.15$. The steep gradients in Fig. 4.6(a) correspond to the dense regions in Fig. 4.6(b). The ratio of scales between fine and coarse regions

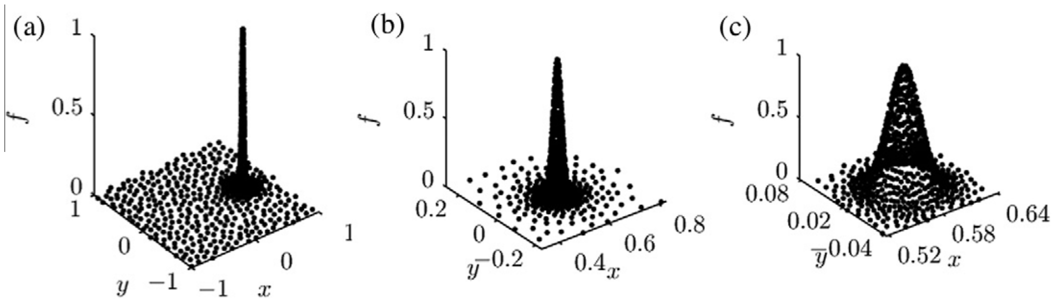

Fig. 4.3. Example particle distribution $\{x_p, y_p, f(x_p, y_p, 0)\}$ after self-organization to the initial condition (Eq. (4.6)) of the Gaussian pulse advection–diffusion problem for $Pe = 10^4$. The resulting number of particles is $N = 1300$. (a) Entire computational domain Ω; (b)/(c) successive close-ups on the Gaussian pulse.

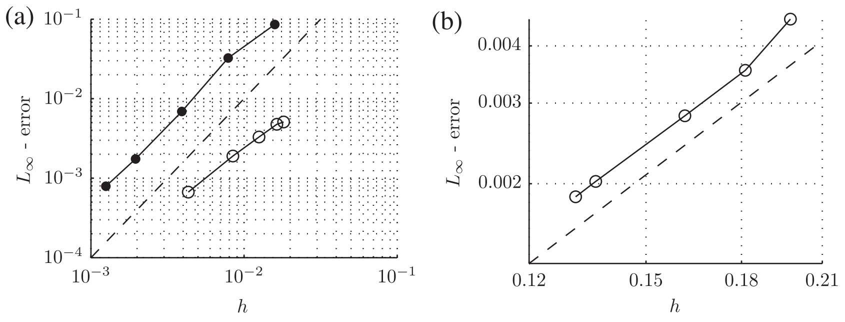

Fig. 4.4. Maximum point-wise error versus average inter-particle spacing h for the Gaussian pulse advection–diffusion problem after one revolution at $T = 1$. (a) Two-dimensional case with $Pe = 1000$, $m = 4$, and $h = 2/\sqrt{N}$; filled circles: remeshed LPM, open circles: present method. (b) Three-dimensional case with $Pe = 100$, $m = 4$, and $h = 2/N^{1/3}$. Both dashed lines have slope 2.

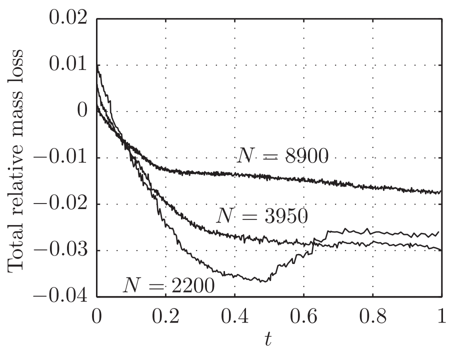

Fig. 4.5. Total relative mass loss (Eq. (4.9)) versus time during the advection–diffusion of a 2D Gaussian pulse for three different resolutions D_0 and the resulting maximum numbers of particles N: $(D_0,N) =$ (0.04,8900); (0.06,3950); (0.08,2200).

is approximately 12 in this case and depends mostly on the gradient of the solution f, and not on the user-defined resolution limit D_0. This indicates that all scales in the solution are properly resolved.

For comparison, we also solve Eq. (4.10) using a remeshed LPM in the weak formulation, where particles have a volume and carry an extensive strength. Eq. (4.10) can be rewritten in conservative form as a transport equation for the quantity f with a flow of velocity $\mathbf{u}/2$:

$$\frac{\partial f}{\partial t} + \nabla \cdot \left(\frac{\mathbf{u}}{2} f\right) = \frac{1}{\mathrm{Re}} \Delta f. \tag{4.11}$$

The particles are initialized on the nodes of a uniform Cartesian mesh covering Ω with a resolution of h. Advection with the velocity $\mathbf{u}/2$ is performed using forward Euler with $\delta t = h^2 Re/4$. After each time step the particles are remeshed using the M_4' interpolation kernel with $\epsilon = h$ [29]. The diffusion term is computed on the mesh using centered second-order finite differences.

Since no analytical solution is available for this problem, we use a numerical reference solution computed on a 2048×2048 mesh using the remeshed LPM. A second-order interpolation of this reference solution is used to compute point-wise errors at all particle locations. The maximum of these point-wise errors as a function of $h = 1/\sqrt{N}$ is shown in Fig. 4.7 at $T = 0.1$ for $Re = 100$. Convergence with the average inter-particle spacing h is second-order in both cases. We use the present method with $m = 4$, such that the DC-PSE operators have fourth order for interpolation and second order for the approximation of the Laplacian. Time integration is done using forward Euler with $\delta t = h_{\min}^2 Re/4$, where $h_{\min}$ is the smallest distance between any two particles in the domain.

The maximum number of particles used by the present method in this case is about 8 times smaller than that of the remeshed LPM, independent of the target error level. This ratio, however, depends on the solution itself and increases as finer scales and steeper gradients develop.

4.5. The five-spot problem: 2D Buckley–Leverett equation

As a real-world application we consider another 2D nonlinear problem, known as the five-spot problem or the waterflooding problem. This popular test case for oil reservoir modeling describes the injection of a wetting fluid

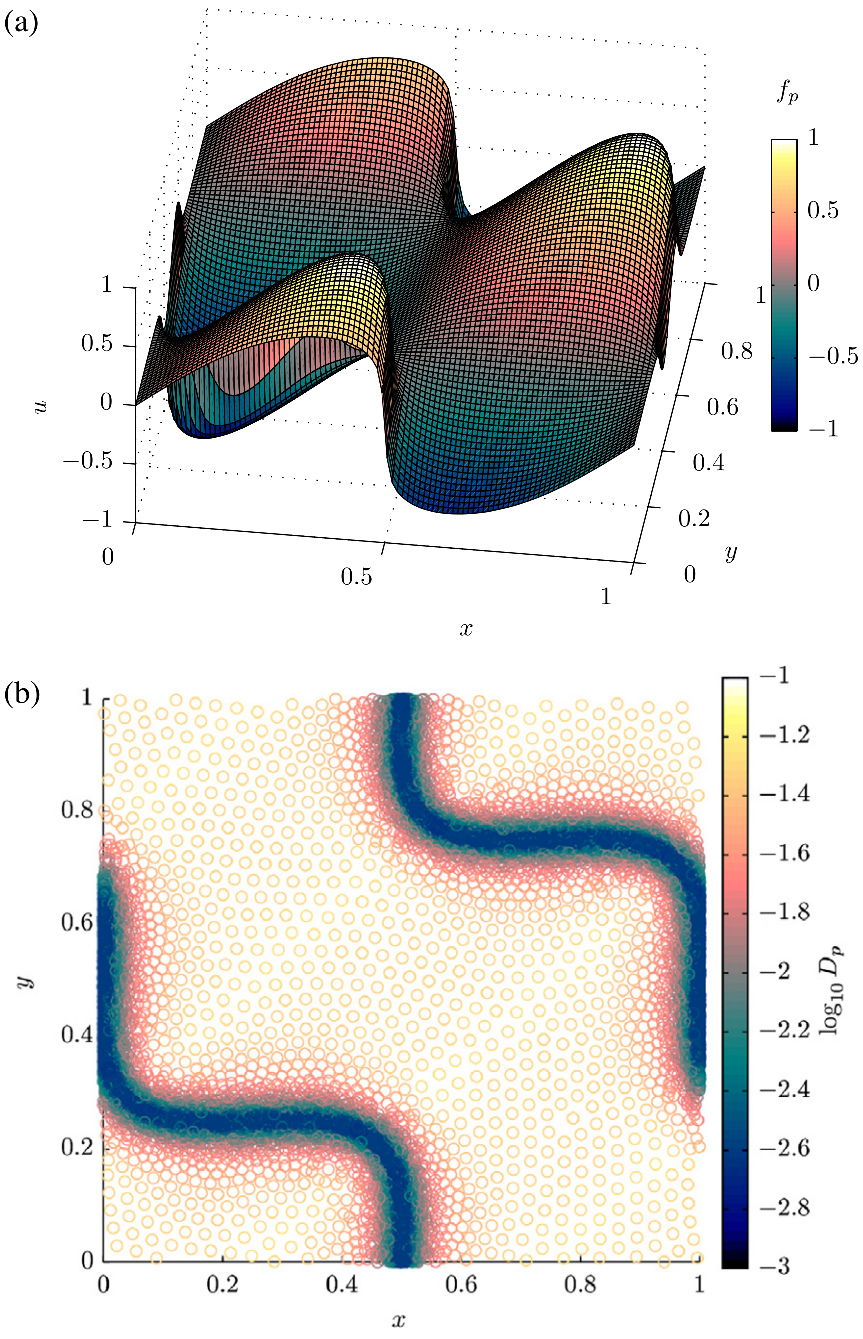

Fig. 4.6. Numerical solution of the 2D Burgers equation for $Re = 1000$ at $T = 0.17$ using the present method. (a) Particle intensities f_p interpolated onto a regular Cartesian mesh for visualization purposes; color codes the function value. (b) Particle positions (circles) and sizes (color: $\log_{10}(D_p)$). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(water) at the center of a porous medium initially saturated with a non-wetting fluid (oil). The oil is flushed away by the pressurized water and sucked out from the four corners of the reservoir. Details of this test case can be found, e.g., in Iske and Kaser [21]. When neglecting gravity and capillary effects, the problem reduces to the viscous Buckley–Leverett equation:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla g(f) = \nu \Delta f \tag{4.12}$$

for the water saturation f. Here, the flux function g is of the form:

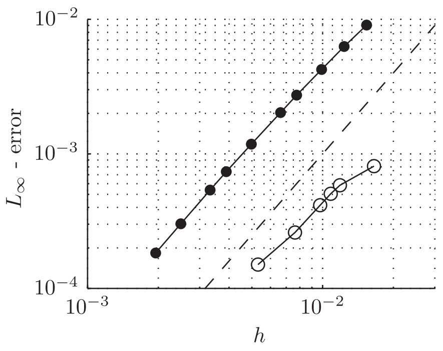

Fig. 4.7. Maximum point-wise error versus average inter-particle spacing $h = 1/\sqrt{N}$ for the 2D Burgers equation at $T = 0.1$ with $Re = 100$ and $m = 4$; filled circles: remeshed LPM, open circles: present method. The dashed line has slope 2.

$$g(f) = \frac{f^2}{f^2 + \mu(1-f)^2}, \tag{4.13}$$

where $\mu > 0$ is the ratio between the two fluids' viscosities. The artificial diffusion term on the right-hand side of Eq. (4.12), with $\nu > 0$, is a standard regularization technique to render the equation parabolic and guarantee the existence of smooth solutions.

The oil is pumped out at the four corners of the domain $\Omega = [-0.5,0.5]^2$ and water is injected at the origin $\mathbf{x} = \mathbf{0}$. Using the same simplifications as Iske and Kaser [21], we assume that the velocity field is stationary and given by $\mathbf{u} = -\nabla p$ with

$$p = \sum_{i=1}^{4} \log(|\mathbf{x} - \mathbf{c}_i|) - \log(|\mathbf{x}|),$$

as plotted in Fig. 4.8(a).

At $t = 0, f \equiv 1$ inside a disk of radius 0.02 centered at the injection well $\mathbf{x} = \mathbf{0}$. Eq. (4.12) is then solved using the present method. Time stepping is done using forward Euler with $\delta t = h^2_{\min}/4\nu$. Particle distributions and saturation fields at different times are shown in Fig. 4.9. The particles self-organize to concentrate near the steep water/oil front. The number of particles in the simulation grows from 373 to 7300 as the front elongates and propagates across the reservoir (see Fig. 4.8(b)).

4.6. Curvature-driven surface refinement

As a geometric example application we consider the evolution of a curved surface embedded in $\mathbb{R}^3$. The surface is represented implicitly as a level set [40] that is discretized using the particles as collocation points [19].

We consider a surface of revolution generated by three arcs of circles, resembling a small bud pinching off from a larger sphere (see Fig. 4.10). This models the geometry of a dividing yeast cell. The radii of the bud and of the sphere are fixed to 0.1 and 0.3, respectively. The neck between the bud and the sphere has a radius of curvature of 0.1. The distance L between the center of the bud and that of the sphere is varied parametrically. The level set is known analytically as a function of L. When L approches 0.6, the neck becomes a thin tether of vanishing thickness and the surface develops a singularity.

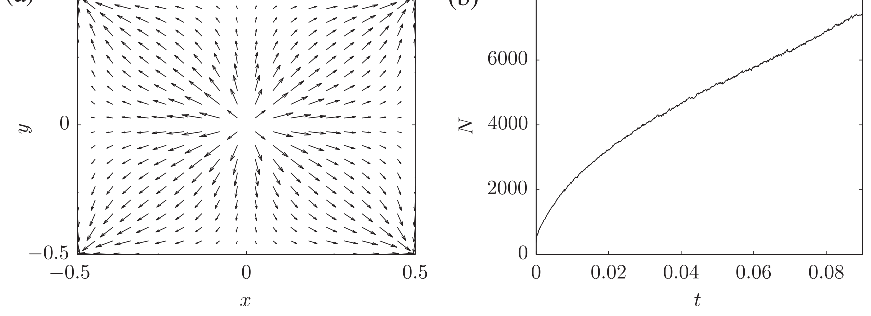

Fig. 4.8. (a) Velocity field of the five-spot problem. The injection well is in the center of the domain and the fluid is pumped out from the four corners. (b) Time evolution of the number of particles N during a simulation using the present method.

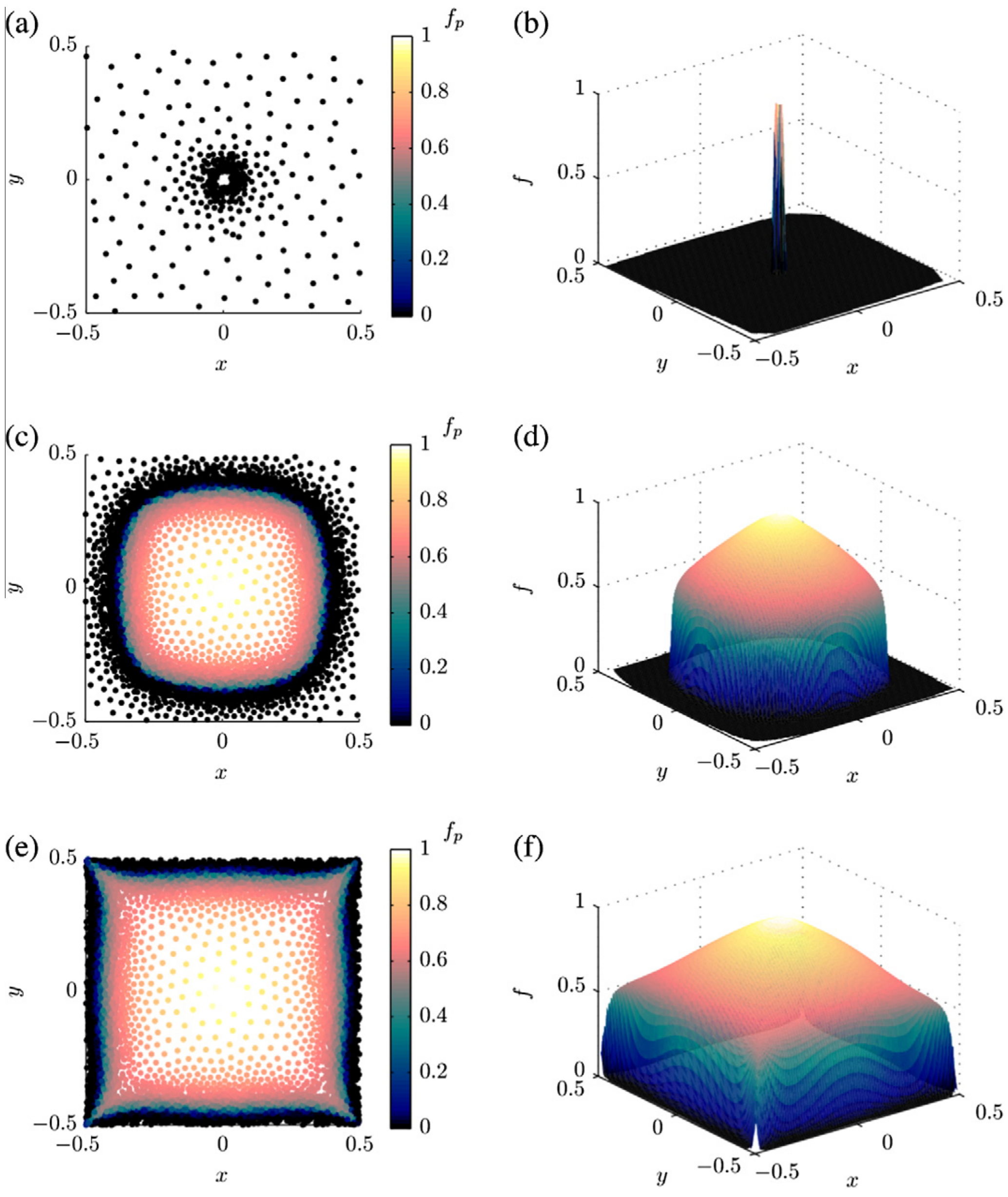

Fig. 4.9. Visualization of the particle distribution (left column) and water saturation field f (right column) for the five-spot problem at different times: (a)/(b) $t = 0$; (c)/(d) $t = 0.056$; (e)/(f) $t = 0.09$.

In order to efficiently resolve the geometry, the density of particles needs to be larger (and their core sizes smaller) in regions where the surface has a high curvature. We hence use the monitor function:

$$\widetilde{D}(\mathbf{x}) = \frac{D_0}{\sqrt{1 + \max(\kappa(\mathbf{x}), 1/d(\mathbf{x}))^2}}, \tag{4.14}$$

where $\kappa(\mathbf{x})$ is the larger of the two principal curvatures at $\mathbf{x}$, and $d(\mathbf{x}) \geqslant 0$ is the distance to the opposing surface. The latter term matters only in the neck region where the surfaces from the two sides approach each other toward the singularity at $d \to 0$.

The particle sizes span a continuous spectrum of scales and the geometry is well resolved everywhere. Particles are only placed in a narrow band around the surface and the rest of the volume remains empty [9]. The width of the narrow band varies in space and is set to $4\widetilde{D}(\mathbf{x})$.

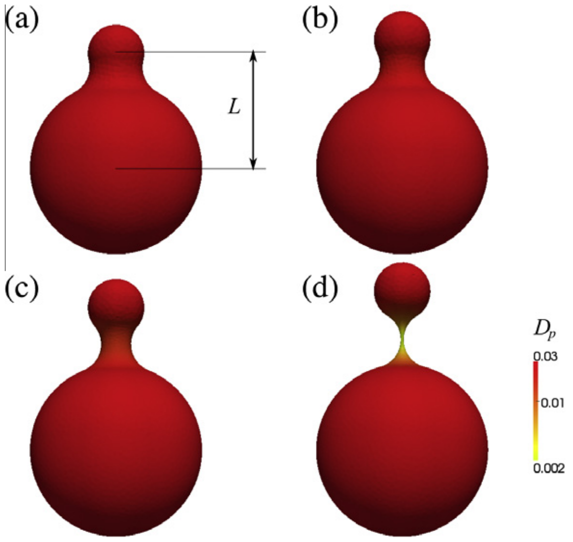

Fig. 4.10. Evolution of a level-set surface represented by self-organizing particles. The distances L between the sphere centers and the total numbers of particles N are: (a) $L = 0.40$, $N = 2.75 \times 10^4$; (b) $L = 0.45$,$N = 2.8 \times 10^4$; (c) $L = 0.50$, $N = 3.0 \times 10^4$; and (d) $L = 0.56$,$N = 4.8 \times 10^4$. The core sizes of the particles are proportional to the resolution D_p, represented by the color code on the isosurface of the level function. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In order to account for the free-space boundaries at the edge of the narrow band and to cope with the strong variations in the resolution field, we use the Morse potential V_1 (Eq. (2.12)) for particle self-organization (see Section 2.3.2).

The coarsest resolution is set to $D_0 = 0.1$. The number of particles increases from 2.75×10^4 to 4.8×10^4 as L is increased from 0.4 to 0.56. At all times, the particles self-organize to fill the narrow band with the desired local inter-particle spacing, as shown in Fig. 4.11 for the interesting region around the neck.

4.7. Computational cost

We quantify the overall computational cost of the present method by comparing the CPU times needed to numerically solve the Burgers Eq. (4.10) for different resolutions at $Re = 100$. The CPU times are averaged over the last 30 time steps of each simulation, and N is the total number of particles at the end of a simulation. All benchmarks are implemented using version 1.2.1 of the PPM library [37,2], available from www.ppm-library.org. The Fortran 90 code was compiled with the Intel Fortran compiler version 12 with -O3 optimization and run on an Intel Xeon 2.8 GHz core with 8 GB of RAM.

Fig. 4.12(a) shows the times spent in each step of the simulation algorithm. The overall scaling of the computational cost appears linear with the total number of particles in the simulation, but has an upper bound of $O(N\log N)$ due to the adaptive tree used in the neighbor-list algorithm [3]. In the benchmarks presented here, particle self-organization (step 4 in the algorithm in Section 3.2.2) is done every 10 time steps and represents 36% of the total CPU time.[2] The remaining 64% are mostly spent constructing the DC-PSE operators for the diffusion term in Eq. (4.10), which must be re-done at every time step (step 5). Of the time spent for the particles to self-organize, 57% comes from constructing the DC-PSE operators (step 4(a)) for evaluating the monitor function and for interpolating the particle intensities to the adapted particle positions. 38% of the self-organization time is spent constructing neighbor lists (step 4(e)-iii) using the algorithm of Awile et al. [3]. Insertion/removal of particles and gradient descent on the self-organization energy (step 4(e) without 4(e)-iii), jointly account for the remaining 5% of the self-organization time (1.8% of the total simulation time).

Fig. 4.12(b) shows the number of gradient-descent iterations needed for the energy minimization in the self-organization of particles (step 4(e) in the algorithm in Section 3.2.2). Starting from a uniformly random distribution of $N = 1000$ particles, adaptation to the initial condition requires 23 iterations. Once the particles are adapted, two gradient-descent iterations every 10 time steps are usually sufficient. Larger particle rearrangements (insertion and removal of particles) happen every 50–100 time steps and require between 8 and 18 gradient-descent iterations. Regardless of the number of gradient-descent iterations, however, the most costly parts of the self-organization process (steps 4(a)–4(d) and 4(f)–4(g)) are done only once. Energy minimization (step 4(e)) accounts for 15% of the total simulation time (43% of the self-organization time).

It has been shown [39] that in certain cases the cost of constructing DC-PSE operators may be amortized by the resulting gain in accuracy. Here, we see that the computational overhead introduced by the self-organization procedure is comparable to that from the DC-PSE operators and, similarly, may be amortized by the gain in spatial resolution for a given number of

[2] If particle self-organization were done at every time step, it would account for about 85% of the total CPU time of the present test case.

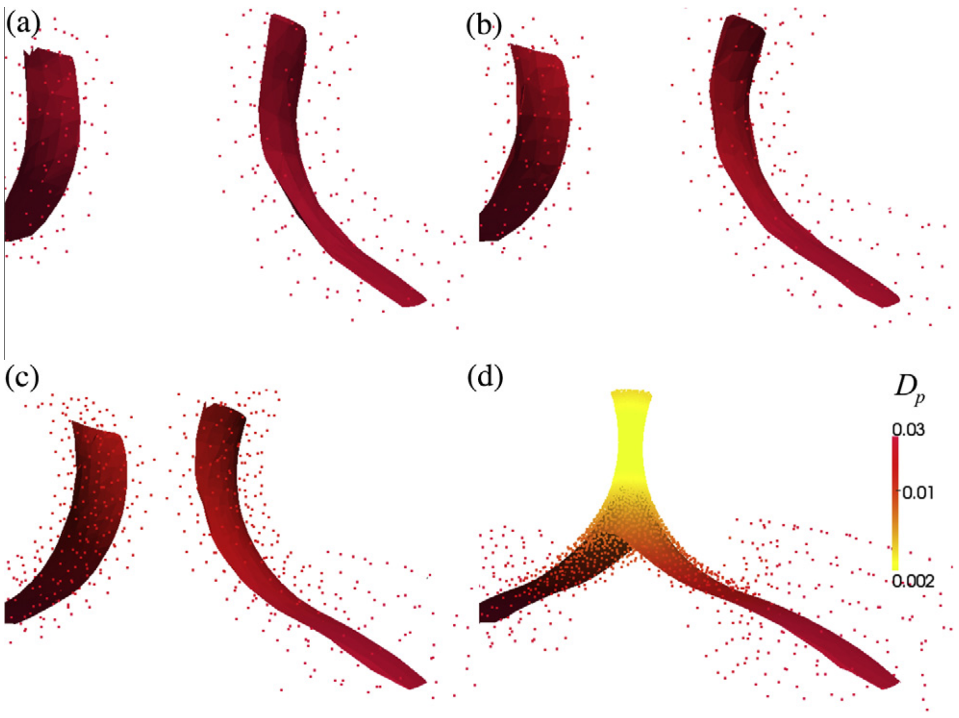

Fig. 4.11. Isosurface and particle distribution near the neck between the two spheres at separation distances (a) $L = 0.40$; (b) $L = 0.45$; (c) $L = 0.50$; and (d) $L = 0.56$. The particles are confined to a narrow band with free-space boundary conditions and spatially varying width using the self-organization potential V_1 (Eq. (2.12)). Color codes the local resolution (particle sizes) D_p. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

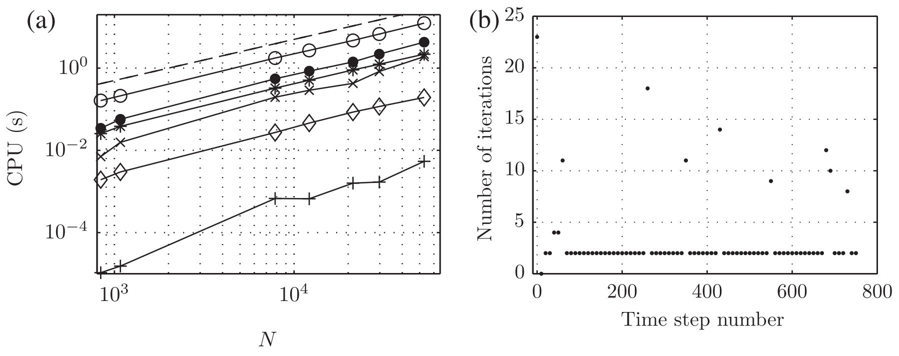

Fig. 4.12. (a) CPU time per time step versus final number of particles N for solving the Burgers Eq. (4.10) until $T = 0.1$ for $Re = 100$. Open circles: total time; filled circles: time for particle self-organization; stars: time for constructing DC-PSE operators; crosses: time for constructing neighbor lists; diamonds: time for the rest of the self-organization algorithm (including insertion/removal of particles and gradient descent on the potential); pluses: time for computing the derivatives and interpolations using DC-PSE operators. The dashed line without symbols has slope 1. (b) Number of iterations of the self-organization gradient descent (step 4(e) in the algorithm in Section 3.2.2) required at each adaptation for $N = 4 \times 10^4$. Self-organization is done every 10 time steps.

particles, or by needing less particles to achieve a certain error level. In the present test case, we measure a net speed-up of approximately 2 over the CPU time required by the non-adaptive remeshed finite-difference LPM code with remeshing at every time step (with the same parameters as described in Section 4.4) to reach the same error level. Remeshing less frequently is, in the present case, less efficient since it increases the spatial discretization error and requires a larger number of particles in order to reach the target accuracy.

5. Conclusions and discussion

We have introduced an adaptive-resolution Lagrangian particle method for continuous parabolic problems. In the present method particles self-organize according to adaptation pseudo-forces such as to approximately equi-distribute the numerical approximation error. This causes the total number of particles in the simulation to approach the smallest number required to represent the solution field with a given accuracy everywhere. In contrast to previous adaptive-resolution particle methods, the present approach does not require any implicit or explicit mapping functions into a reference space of uniform resolution, nor does it require global transforms.

The presented method relies on pairwise interaction potentials according to which the particles self-organize in an energy minimization process. Together with dynamic insertion and removal of particles where needed, this leads to robust and efficient adaptation of the particle density and sizes to the features of the evolving field functions. Remeshing is replaced by interpolation from the old set of particles before self-organization to the new, adapted set of particles. The self-organization potential is chosen according to the boundary conditions and the gradients of the monitor function such that the solution field is always well sampled and that no holes or clusters develop in the particle distribution. Consistent approximations of differential operators on scattered sets of particles with varying core sizes, as well as particle–particle interpolation schemes that satisfy certain moment conditions, can be constructed as DC-PSE operators [39]. Constructing these DC-PSE operators requires inverting a small linear system of equations for each particle. These systems, however, only need to be solved once and all operator and interpolation kernels can be constructed from the same inverse.

The additional computational cost incurred by the self-organization may be amortized by the gain in accuracy. Compared to non-adaptive methods, the overhead of self-organization is amortized whenever the solution has multi-scale features. In these cases, the present method requires fewer particles than non-adaptive methods. This advantage is more pronounced in 3D than in 2D.

We have shown that the truncation errors of the discretization schemes correspond to those predicted by theory, both for particle–particle interpolation and for the DC-PSE operators. We have validated the present method on two- and three-dimensional advection–diffusion problems where analytical solutions are available and have shown that the method can be used to efficiently address also more complex, nonlinear problems.

The presented method has four parameters that control its behavior: the coarsest resolution to be used D_0, the neighborhood size r^*, the termination threshold d_c of the energy minimization, and every how many time steps particles are re-organized. For d_c we recommend a standard value of 2.5. The smaller this value, the more gradient-descent iterations are needed and the more regular (but still adaptive-resolution) the resulting particle distribution becomes. The value for r^* is determined as outlined in Section 2.3.4. The number of time steps between particle re-organization depends on how fast the resolution requirements evolve in a given problem. A conservative, but computationally expensive setting would be to do it at every time step. Finally, D_0 is set according to what minimal resolution one requires in the solution. This is needed since the total number of particles N is *not* a free parameter of the method, but is determined adaptively to fulfill the resolution requirements. For some problems it may also be beneficial to use a monitor function different from the one in Eq. (2.8). We have presented an example in Section 4.6 and refer to Appendix B for how the choice of monitor function influences the accuracy of the method. For particle self-organization we always use the potentials presented in Section 2.3.2; they have no parameters. Additional parameters, albeit not specific to the present method, are the order of accuracy m of the DC-PSE operators [39] and the time-step size δt, which is given by the time stepping algorithm used.

In its current form the present method has a couple of limitations. The most important one probably is that the method is not conservative. Exact conservation of mass could be enforced by symmetric DC-PSE operators. Constructing such operators, however, is an open problem for convergence orders larger than one. Another limitation of the present formulation is that in explicit time-stepping schemes, such as Euler or Runge–Kutta, the time-step size is dictated by the CFL condition in the highest-resolution region. In applications where most of the particles are located in coarsely resolved regions, this is inefficient. Multi-resolution time-stepping schemes, such as multirate Runge–Kutta schemes [38], are available to alleviate this. In most applications of adaptive-resolution methods, however, the majority of the particles are located in high-resolution regions.

Current and future work is concerned with extending the presented method to the weak form of the governing equations and with restoring conservativeness. This requires particles with a non-zero physical volume that carry the extensive quantity associated with the field f. Weak formulations are favorable if f is discontinuous.

Acknowledgments

We thank Christian L. Müller (MOSAIC Group, ETH Zurich) for many inspiring discussions and Nicolas Fiétier (MOSAIC Group, ETH Zurich) for proofreading the manuscript. Birte Schrader was financed by grant #200021-115764 from the Swiss National Science Foundation (to Ivo F. Sbalzarini). Sylvain Reboux was financed by a grant from the Swiss SystemsX.ch initiative (to Ivo F. Sbalzarini), Grant LipidX-2008/011, evaluated by the Swiss National Science Foundation.

Appendix A. Interpolating DC-PSE operators

The present self-organizing particle method relies on accurate particle–particle interpolation schemes and on consistent approximation of derivatives of f on arbitrary distributions of particles with varying core sizes. We outline how this can be achieved in the DC-PSE framework [39] using the same approach as in Reproducing Kernel Particle Methods [27]. Since we use the same framework for both interpolation and approximation of derivatives, we first refer to a generic kernel function ϕ

before specializing to ζ for particle–particle interpolation (see Eq. (2.2)) and η for approximating the elliptic operator $\mathcal{L}$ (see Eq. (2.4)).

For any point $\mathbf{x} \in \mathbb{R}^d$, we can define a kernel function $\mathbf{y} \mapsto \phi(\mathbf{y};\mathbf{x}) \in \mathbb{R}$ as the product of a smooth weight function w (in this paper we choose $w(x) = \exp(-c^2x^2/2)$, with $c > 0$, but other choices are also possible) and a polynomial correction function:

$$\phi(\mathbf{y};\mathbf{x}) = w^2\left(\frac{|\mathbf{y}-\mathbf{x}|}{\epsilon(\mathbf{x})}\right)\left[\mathbf{p}\left(\frac{\mathbf{y}-\mathbf{x}}{\epsilon(\mathbf{x})}\right)\mathbf{c}^T(\mathbf{x})\right], \tag{A.1}$$

where the arguments are rescaled with the spatially varying resolution $\epsilon(\mathbf{x})$. The row vector $\mathbf{p}(\mathbf{x})$ is the complete basis of monomials $\{\mathbf{x}^{\mathbf{k}}\}_{|\mathbf{k}|\leqslant m}$ for a multi-index $\mathbf{k}$, and $\mathbf{c}^T$ is a column vector of unknown coefficients. These coefficients, which depend on $\mathbf{x}$, are determined by enforcing *discrete moment conditions* of the form:

$$\sum_q \left[\left(\frac{\mathbf{x}_q-\mathbf{x}}{\epsilon(\mathbf{x})}\right)^{\mathbf{k}}\phi(\mathbf{x}_q;\mathbf{x})\right] = b_{\mathbf{k}}, \quad \text{for } |\mathbf{k}| < m, \tag{A.2}$$

where m is the desired order of accuracy of ϕ. The support of w is assumed to be local, such that the summation in Eq. (A.2) is only done over particles q in some neighborhood of the point $\mathbf{x}$. In addition, here we always choose $b_{\mathbf{0}} = 0$ in order to obtain kernels with a vanishing zeroth-order moment that can be consistently evaluated at off-particle locations (i.e., when $\mathbf{y}$ does not coincide with any $\mathbf{x}_q$).

Using Eqs. (A.1) and (A.2) can be written in matrix form as:

$$\mathbf{A}(\mathbf{x})\mathbf{c}^T(\mathbf{x}) = \mathbf{b}^T, \tag{A.3}$$

where $\mathbf{b} = \{b_{\mathbf{k}}\}_{|\mathbf{k}|\leqslant m}$ and $\mathbf{A}(\mathbf{x})$ is the matrix $(\mathbf{w}(\mathbf{x})\mathbf{V}(\mathbf{x}))^T\mathbf{w}(\mathbf{x})\mathbf{V}(\mathbf{x})$ with $\mathbf{V}(\mathbf{x})$ the Vandermonde matrix associated with the d-dimensional polynomial basis $\mathbf{p}(\mathbf{x})$ and the set of points $\{\mathbf{x}_p\}$belonging to the neighborhood of $\mathbf{x}$. The diagonal weight matrix $\mathbf{w}(\mathbf{x})$ has entries $w(|\mathbf{x}-\mathbf{x}_p|)$ for the same set of points $\{\mathbf{x}_p\}$.

The matrix $\mathbf{A}(\mathbf{x})$ in Eq. (A.3) contains information about the spatial distribution of the particles $\{\mathbf{x}_p\}$ around $\mathbf{x}$ (weighted by w), while the right-hand side $\mathbf{b}$ determines the approximation properties of the kernel. For example choosing the vector:

$$\mathbf{b}^T = \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}\mathbf{p}\Big|_{\mathbf{x}=\mathbf{0}}, \tag{A.4}$$

for a given multi-index $\boldsymbol{\alpha}$, leads to kernel functions $\eta_p(\mathbf{x}) \equiv \phi(\mathbf{x};\mathbf{x}_p)$ that approximate the derivative of degree $\boldsymbol{\alpha}$ at arbitrary locations $\mathbf{x}$, hence:

$$\frac{\partial^{|\boldsymbol{\alpha}|}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}f \approx \epsilon(\mathbf{x})^{-|\boldsymbol{\alpha}|}\sum_p f_p\eta_p(\mathbf{x}). \tag{A.5}$$

Similarly, choosing:

$$\mathbf{b}^T = \Delta\mathbf{p}|_{\mathbf{x}=\mathbf{0}} \tag{A.6}$$

yields an approximation of the Laplace operator evaluated at arbitrary $\mathbf{x}$:

$$\Delta f(\mathbf{x}) \approx \epsilon(\mathbf{x})^{-2}\sum_p f_p\eta_p(\mathbf{x}). \tag{A.7}$$

We note here that the same way of constructing the right-hand side $\mathbf{b}$ of the discrete moment conditions in Eq. (A.2) can also be used in general DC-PSE operators with non-vanishing zeroth-order moments [39]. The resulting approximation of the Laplacian at source point locations $\mathbf{x} = \mathbf{x}_q$, for example, then becomes:

$$\Delta^h f(\mathbf{x}_q) \equiv \epsilon(\mathbf{x}_q)^{-2}\sum_p (f_p - f_q)\eta_q(\mathbf{x}_p) \approx \Delta f(\mathbf{y})|_{\mathbf{y}=\mathbf{x}_q} \tag{A.8}$$

with the same η_p as in Eq. (A.7). Conveniently, the same η_p also yield an approximation of the gradient of f at $\mathbf{x} = \mathbf{x}_q$ as:

$$\nabla^h f(\mathbf{x}_q) \equiv \epsilon(\mathbf{x}_q)^{-2}\sum_p f_p(\mathbf{x}_q - \mathbf{x}_p)\eta_q(\mathbf{x}_p) \approx \nabla f(\mathbf{y})|_{\mathbf{y}=\mathbf{x}_q}.$$

Note that, in contrast to general DC-PSE operators, the sum in Eq. (A.5) only involves the intensities of the source particles. This is a direct consequence of the kernels having vanishing zeroth-order moments. In general DC-PSE operators the sum also involves the intensity of the collocation particle, hence allowing for non-zero zeroth-order moments. Operators with a non-vanishing zeroth-order moment, however, can only be consistently evaluated at source particle locations.

If the weight function w is strictly positive, the invertibility of $\mathbf{A}$ in Eq. (A.3) depends only on that of the Vandermonde matrix $\mathbf{V}$. If $\mathbf{V}$ is invertible, $\mathbf{A}$ is symmetric and positive definite (as the product of a real matrix and its transpose) and can efficiently be inverted using, e.g., Cholesky decomposition. Nevertheless, this operation represents most of the computational cost of solving Eq. (A.3) for the unknown kernel coefficients $\mathbf{c}$. Solving this system for multiple right-hand sides $\mathbf{b}$ in order to, e.g., compute derivatives of different orders, then comes at little additional cost.

The same framework can also be used to construct accurate particle–particle interpolation schemes that satisfy the same moment conditions as the DC-PSE operators. This is done by choosing $\boldsymbol{\alpha} = \mathbf{0}$ in Eq. (A.4), which yields operators that approximate the function f itself at any point $\mathbf{x}$ given the function values f_p at scattered neighboring points $\{\mathbf{x}_p\}$. Without further precautions, however, this approximation will not have the interpolating Kronecker-delta property, which may lead to undesired interpolation errors.

Interpolating kernel functions $\zeta_p(\mathbf{x}) \equiv \zeta(\mathbf{x};\mathbf{x}_p)$ that exactly fulfill the property $f^h(\mathbf{x}_p) = f_p$ for all p can be constructed by re-using the same matrix $\mathbf{A}$ and its Cholesky decomposition (or inverse) that was already computed for approximating derivatives.

Following Chen et al. [12], interpolating kernels are obtained by expressing ζ as the sum of the non-interpolating kernel ϕ and a correction function $\hat{\zeta}$, thus:

$$\zeta_p(\mathbf{x}) = \phi(\mathbf{x};\mathbf{x}_p) + \hat{\zeta}_p(\mathbf{x}). \tag{A.9}$$

The non-interpolating kernels $\phi(\mathbf{x};\mathbf{x}_p)$ are obtained by solving Eq. (A.3) with right-hand side:

$$\mathbf{b}^T = \mathbf{p}(\mathbf{0}) - \sum_q \mathbf{p}\left(\frac{\mathbf{x}-\mathbf{x}_q}{\epsilon(\mathbf{x})}\right)\hat{\zeta}_q(\mathbf{x}) \tag{A.10}$$

for smooth correction functions:

$$\hat{\zeta}_p(\mathbf{x}) = \hat{\varphi}\left(\frac{\mathbf{x}-\mathbf{x}_p}{a_p}\right) \tag{A.11}$$

that satisfy $\hat{\varphi}((\mathbf{x}_q - \mathbf{x}_p)/a_p) = \delta_{pq}$, where δ_{pq} is the Kronecker delta. The resulting kernels ζ_p satisfy the moment conditions in Eq. (A.2) for $\mathbf{b}^T = \mathbf{p}(\mathbf{0})$, which ensures that the approximation is consistent, as well as the Kronecker delta property $\zeta_p(\mathbf{x}_q - \mathbf{x}_p) = \delta_{pq}$, which ensures that the approximation is interpolating.

Like Wang et al. [44], we take $\hat{\varphi}$ to be the quartic spline with cutoff radius 1 and choose a_p such that it is smaller than the distance between particle p and its nearest neighbor.

Appendix B. Truncation error analysis of DC-PSE operators with *h*-refinement

A Taylor series expansion of f around $\mathbf{x}_p$ and multiplication with $\zeta_{\epsilon_p}(\mathbf{y}-\mathbf{x}_p;\mathbf{x}_p)$ yields, for all $\mathbf{y}$ in the ball $\mathcal{B}_p$ of radius $r_{c,p}$ centered at $\mathbf{x}_p$:

$$(f(\mathbf{y}) - f(\mathbf{x}_p))\zeta_{\epsilon_p}(\mathbf{y}-\mathbf{x}_p;\mathbf{x}_p) = \sum_{|\mathbf{k}|=1}^{m-1} \frac{(\mathbf{y}-\mathbf{x}_p)^{\mathbf{k}}}{\mathbf{k}!}\zeta_{\epsilon_p}(\mathbf{y}-\mathbf{x}_p;\mathbf{x}_p)\frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{y}^{\mathbf{k}}}f(\mathbf{y})\bigg|_{\mathbf{y}=\mathbf{x}_p} + r(\mathbf{y}), \tag{B.1}$$

with the remainder satisfying

$$r(\mathbf{y}) = \sum_{|\mathbf{k}|=m} (\mathbf{y}-\mathbf{x}_p)^{\mathbf{k}}\zeta_{\epsilon_p}(\mathbf{y}-\mathbf{x}_p;\mathbf{x}_p)s_{\mathbf{k}}(\mathbf{y}), \quad \text{with} \quad |s_{\mathbf{k}}(\mathbf{y})| \leqslant \sup_{y\in\mathcal{B}}\left|\frac{1}{\mathbf{k}!}\frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{y}^{\mathbf{k}}}f(\mathbf{y})\right|.$$

Applying Eq. (B.1) for all particles q in the neighborhood $\mathcal{N}_p$ of particle p and summing over particles yields:

$$\begin{aligned}\sum_{q\in\mathcal{N}_p}(f_q - f_p)\zeta_{\epsilon_p}(\mathbf{x}_q-\mathbf{x}_p;\mathbf{x}_p) &= \sum_{|\mathbf{k}|=1}^{m-1}\frac{1}{\mathbf{k}!}\frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{y}^{\mathbf{k}}}f(\mathbf{y})\bigg|_{\mathbf{y}=\mathbf{x}_p}\sum_{q\in\mathcal{N}_p}\left[(\mathbf{x}_q-\mathbf{x}_p)^{\mathbf{k}}\zeta_{\epsilon_p}(\mathbf{x}_q-\mathbf{x}_p;\mathbf{x}_p)\right] + \sum_{q\in\mathcal{N}_p} r(\mathbf{x}_q)\\ &= \sum_{|\mathbf{k}|=1}^{m-1}\epsilon_p^{|\mathbf{k}|}\frac{Z_p^{\mathbf{k}}}{\mathbf{k}!}\frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{y}^{\mathbf{k}}}f(\mathbf{y})\bigg|_{\mathbf{y}=\mathbf{x}_p} + \sum_{q\in\mathcal{N}_p} r(\mathbf{x}_q),\end{aligned}$$

with the discrete moments $Z_p^{\mathbf{k}}$ defined as:

$$Z_p^{\mathbf{k}} = \epsilon_p^{-|\mathbf{k}|}\sum_{q\in\mathcal{N}_p}\left[(\mathbf{x}_q-\mathbf{x}_p)^{\mathbf{k}}\zeta_{\epsilon_p}(\mathbf{x}_q-\mathbf{x}_p;\mathbf{x}_p)\right]. \tag{B.2}$$

Using the following moment conditions for ζ:

$$Z_p^{\mathbf{k}} = Y_{\mathbf{k}} \quad \text{for } |\mathbf{k}| < m, \tag{B.3}$$

with

$$Y_{\mathbf{k}} = \begin{cases} (-1)^{|\boldsymbol{\alpha}|}\boldsymbol{\alpha}!, & \mathbf{k} = \boldsymbol{\alpha},\\ 0, & \text{else},\end{cases} \tag{B.4}$$

we obtain an approximation for any partial derivative $\partial^{|\boldsymbol{\alpha}|}f/\partial\mathbf{x}^{\boldsymbol{\alpha}}$ as:

$$\sum_{q\in\mathcal{N}_p}(f_q-f_p)\zeta_{\epsilon_p}(\mathbf{x}_q-\mathbf{x}_p;\mathbf{x}_p)=\epsilon_p^{|\boldsymbol{\alpha}|}\frac{\partial^{|\boldsymbol{\alpha}|}f(\mathbf{y})}{\partial\mathbf{y}^{\boldsymbol{\alpha}}}\bigg|_{\mathbf{y}=\mathbf{x}_p}+\sum_{q\in\mathcal{N}_p}r(\mathbf{x}_q) \tag{B.5}$$

$$=\epsilon_p^{|\boldsymbol{\alpha}|}\frac{\partial^{|\boldsymbol{\alpha}|}f(\mathbf{y})}{\partial\mathbf{y}^{\boldsymbol{\alpha}}}\bigg|_{\mathbf{y}=\mathbf{x}_p}+\text{h.o.t.} \tag{B.6}$$

For each particle p, we choose a kernel function of the form:

$$\zeta_p(\mathbf{x})=P_p(\mathbf{x})\exp(-|c\mathbf{x}|^2), \tag{B.7}$$

where $P_p(\mathbf{x})$ is a polynomial of degree m and $c > 0$. The discrete moments in Eq. (B.2) are then given by

$$Z_p^{\mathbf{k}}=\epsilon_p^{-|\mathbf{k}|}\sum_{q\in\mathcal{N}_p}\left[\epsilon_p^{\mathbf{k}}(\mathbf{z}_{pq})^{\mathbf{k}}P_p(\mathbf{z}_{pq})\exp\left(-|c\mathbf{z}_{pq}|^2\right)\right] \tag{B.8}$$

$$=\sum_{q\in\mathcal{N}_p}\left[(\mathbf{z}_{pq})^{\mathbf{k}}P_p(\mathbf{z}_{pq})\exp\left(-|c\mathbf{z}_{pq}|^2\right)\right], \tag{B.9}$$

where

$$\mathbf{z}_{pq}=\frac{\mathbf{x}_q-\mathbf{x}_p}{\epsilon_p}.$$

The partial derivative of f can thus be approximated by the DC-PSE operator:

$$\left[\frac{\partial^{|\boldsymbol{\alpha}|}f(\mathbf{x}_p)}{\partial\mathbf{x}^{\boldsymbol{\alpha}}}\right]^h\equiv\sum_{q\in\mathcal{N}_p}(f_q-f_p)L_p(\mathbf{x}_q)\approx\frac{\partial^{|\boldsymbol{\alpha}|}f(\mathbf{y})}{\partial\mathbf{y}^{\boldsymbol{\alpha}}}\bigg|_{\mathbf{y}=\mathbf{x}_p}, \tag{B.10}$$

where

$$L_p(\mathbf{x}_q)=\epsilon_p^{-|\boldsymbol{\alpha}|}\zeta_{\epsilon_p}(\mathbf{x}_q-\mathbf{x}_p;\mathbf{x}_p)=\epsilon_p^{-|\boldsymbol{\alpha}|}P_p\left(\frac{\mathbf{x}_q-\mathbf{x}_p}{\epsilon_p}\right)\exp\left(-\left|c\frac{\mathbf{x}_q-\mathbf{x}_p}{\epsilon_p}\right|^2\right). \tag{B.11}$$

From Eq. (B.5), assuming that the local inter-particle spacing is $h_p=\mathcal{O}(\epsilon_p)$, we find that for $\mathbf{x} = \mathbf{x}_p$ the upper bound on the point-wise error in the spatial derivative decreases as $h_p^{m-|\boldsymbol{\alpha}|}|f|_{W_\infty^m(\mathcal{B}_p)}$, thus:

$$\left|\left[\frac{\partial^{|\boldsymbol{\alpha}|}f(\mathbf{x}_p)}{\partial\mathbf{x}^{\boldsymbol{\alpha}}}\right]^h-\frac{\partial^{|\boldsymbol{\alpha}|}f(\mathbf{x}_p)}{\partial\mathbf{x}^{\boldsymbol{\alpha}}}\right|\leqslant h_p^{-|\boldsymbol{\alpha}|}\sum_{q\in\mathcal{N}_p}|r(\mathbf{x}_q)|\leqslant Ch_p^{m-|\boldsymbol{\alpha}|}|f|_{W_\infty^m(\mathcal{B}_p)}\|\zeta_{\epsilon_p}\|_\infty. \tag{B.12}$$

The norm $\|\zeta_{\epsilon_p}\|_\infty$ of the kernel function depends on the order of approximation m, the parameter c, and the particle distribution.

Other possible choices than Eq. (2.8) for the target resolution field (monitor function) $\widetilde{D}(\mathbf{x})$ can in some cases lead to a more accurate discretization of the differential operators in Eq. (2.1). The choice of $\widetilde{D}(\mathbf{x})$ is intrinsically linked with the discretization errors of the function and operator approximations. For example, the local truncation error of the spatial derivative of degree $\boldsymbol{\alpha}$ computed with DC-PSE operators on particles spaced by a distance D_p and with a cutoff radius $r_{c,p} = D_p r^*$ scales like $D_p^{m-|\boldsymbol{\alpha}|}|f|_{W_\infty^m(\mathcal{B}_p)}$. In this case, the optimal choice for the resolution field (in the sense that it would equi-distribute the truncation error across all particles) is of the form $D_p\propto\left(|f|_{W_\infty^m(\mathcal{B}_p)}\right)^{-1/(m-|\boldsymbol{\alpha}|)}$, where the ball $\mathcal{B}_p$ contains all particles at a distance less than $D_p r^*$ from $\mathbf{x}_p$. One then has to solve Eq. (2.9) with:

$$\widetilde{D}(\mathbf{x})=D_0\left(\max_{|\boldsymbol{\beta}|=m}\left|\frac{\partial^{|\boldsymbol{\beta}|}}{\partial\mathbf{x}^{\boldsymbol{\beta}}}f(\mathbf{x})\right|\right)^{-1/(m-|\boldsymbol{\alpha}|)}. \tag{B.13}$$

Note, however, that albeit Eq. (B.13) together with Eq. (2.9) is optimal (in some sense), it may not always be amenable to efficient and accurate numerical computation.

References

[1] D.A. Anderson, M.M. Rai, The use of solution adaptive grids in solving partial–differential equations, Appl. Math. Comput. 10 (1982) 317–338.
[2] O. Awile, O. Demirel, I.F. Sbalzarini, Toward an object-oriented core of the PPM library, in: Proceedings of the ICNAAM, International Conference on Numerical Analysis and Applied Mathematics, 2010.
[3] O. Awile, F. Büyükkeçeci, S. Reboux, I.F. Sbalzarini, Fast neighbor lists for adaptive-resolution particle simulations, Comput. Phys. Commun. (2012), doi:10.1016/j.cpc.2012.01.003.
[4] L.A. Barba, A. Leonard, C.B. Allen, Advances in viscous vortex methods – meshless spatial adaption based on radial basis function interpolation, Int. J. Numer. Methods Fluids 47 (5) (2005) 387–421.
[5] J. Behrens, A. Iske, Grid-free adaptive semi-lagrangian advection using radial basis functions, Comput. Math. Appl. 43 (3–5) (2002) 319–327.

[6] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, P. Krysl, Meshless methods: an overview and recent developments, Comput. Methods Appl. Mech. Eng. 139 (1996) 3–47.
[7] M. Bergdorf, P. Koumoutsakos, A Lagrangian particle–wavelet method, Multiscale Model. Simul. 5 (3) (2006) 980–995.
[8] M. Bergdorf, G.-H. Cottet, P. Koumoutsakos, Multilevel adaptive particle methods for convection–diffusion equations, Multiscale Model. Simul. 4 (1) (2005) 328–357.
[9] M. Bergdorf, I.F. Sbalzarini, P. Koumoutsakos, A Lagrangian particle method for reaction–diffusion systems on deforming surfaces, J. Math. Biol. 61 (2010) 649–663, doi:10.1007/s00285-009-0315-2.
[10] C.J. Budd, W. Huang, R.D. Russell, Adaptivity with moving grids, Acta Numer. 18 (2009) 111–241.
[11] W. Cao, H. Weizhang, R. Russell, An *r*-adaptive finite element method based upon moving mesh PDEs, J. Comput. Phys. 149 (2) (1999) 221–244.
[12] J.S. Chen, W.M. Han, Y. You, X.P. Meng, A reproducing kernel method with nodal interpolation property, Int. J. Numer. Methods. Eng. 56 (7) (2003) 935–960.
[13] H. Cohn, A. Kumar, Algorithmic design of self-assembling structures, Proc. Natl. Acad. Sci. USA 106 (24) (2009) 9570–9575, doi:10.1073/pnas.0901636106.
[14] G.-H. Cottet, P. Koumoutsakos, Vortex Methods – Theory and Practice, Cambridge University Press, New York, 2000.
[15] G.H. Cottet, P. Koumoutsakos, M.L.O. Salihi, Vortex methods with spatially varying cores, J. Comput. Phys. 162 (1) (2000) 164–185.
[16] M. D'Orsogna, Y.-L. Chuang, A. Bertozzi, L. Chayes, Pattern formation stability and collapse in 2D driven particle systems, in: S. Baglio, A. Bulsara (Eds.), Device Applications of Nonlinear Dynamics, Understanding Complex Systems, vol. 7, Springer, Berlin, Heidelberg, 2006, pp. 103–113. ISBN: 978-3-540-33877-2.
[17] J.D. Eldredge, A. Leonard, T. Colonius, A general deterministic treatment of derivatives in particle methods, J. Comput. Phys. 180 (2) (2002) 686–709.
[18] X.-J. Fan, R.I. Tanner, R. Zheng, Smoothed particle hydrodynamics simulation of non-Newtonian moulding flow, J. Non-Newton. Fluid Mech. 165 (5–6) (2010) 219–226.
[19] S.E. Hieber, P. Koumoutsakos, A Lagrangian particle level set method, J. Comput. Phys. 210 (2005) 342–367.
[20] T.Y. Hou, Convergence of a variable blob vortex method for the Euler and Navier–Stokes equations, SIAM J. Numer. Anal. 27 (6) (1990) 1387–1404.
[21] A. Iske, M. Kaser, Two-phase flow simulation by AMMoC, an adaptive meshfree method of characteristics, Comput. Model. Eng. Sci. 7 (2) (2005) 133–148.
[22] P. Koumoutsakos, Inviscid axisymmetrization of an elliptical vortex, J. Comput. Phys. 138 (2) (1997) 821–857.
[23] P. Koumoutsakos, Multiscale flow simulations using particles, Annu. Rev. Fluid Mech. 37 (2005) 457–487.
[24] P. Koumoutsakos, A. Leonard, F. Pépin, Boundary-conditions for viscous vortex methods, J. Comput. Phys. 113 (1) (1994) 52–61.
[25] I. Lakkis, A. Ghoniem, A high resolution spatially adaptive vortex method for separating flows. Part I: two-dimensional domains, J. Comput. Phys. 228 (2) (2009) 491–515.
[26] P. Lancaster, K. Salkauskas, Surfaces generated by moving least-squares methods, Math. Comput. 37 (1981) 141–158.
[27] W. Liu, S. Jun, Y. Zhang, Reproducing kernel particle methods, Int. J. Numer. Methods Fluids 20 (8–9) (1995) 1081–1106.
[28] W. Liu, W. Hao, Y. Chen, S. Jun, J. Gosz, Multiresolution reproducing kernel particle methods, Comput. Mech. 20 (4) (1997) 295–309.
[29] J.J. Monaghan, Extrapolating B splines for interpolation, J. Comput. Phys. 60 (1985) 253–262.
[30] P. Morse, Diatomic molecules according to the wave mechanics. II. Vibrational levels, Phys. Rev. 34 (1) (1929) 57–64.
[31] J.A. Munoz-Gomez, P. Gonzalez-Casanova, G. Rodriguez-Gomez, Adaptive node refinement collocation method for partial differential equations, in: ENC'06: Proceedings of the Seventh Mexican International Conference on Computer Science, 2006, pp. 70–80.
[32] J.T. Rasmussen, G.-H. Cottet, J.H. Walther, A multiresolution remeshed Vortex-In-Cell algorithm using patches, J. Comput. Phys. 230 (2011) 6742–6755.
[33] M. Rechtsman, F. Stillinger, S. Torquato, Designed interaction potentials via inverse methods for self-assembly, Phys. Rev. E 73 (1) (2006) 011406, doi:10.1103/PhysRevE.73.011406.
[34] M.C. Rechtsman, F.H. Stillinger, S. Torquato, Optimized interactions for targeted self-assembly: application to a honeycomb lattice, Phys. Rev. Lett. 95 (22) (2005) 228301, doi:10.1103/PhysRevLett.95.228301.
[35] M.C. Rechtsman, F.H. Stillinger, S. Torquato, Self-assembly of the simple cubic lattice with an isotropic potential, Phys. Rev. E 74 (2) (2006), doi:10.1103/PhysRevE.74.021404.
[36] E. Ryan, A. Tartakovsky, C. Amon, A novel method for modeling Neumann and Robin boundary conditions in smoothed particle hydrodynamics, Comput. Phys. Commun. 181 (12) (2010) 2008–2023.
[37] I.F. Sbalzarini, J.H. Walther, M. Bergdorf, S.E. Hieber, E.M. Kotsalis, P. Koumoutsakos, PPM – a highly efficient parallel particle-mesh library for the simulation of continuum systems, J. Comput. Phys. 215 (2) (2006) 566–588.
[38] M. Schlegel, O. Knoth, M. Arnold, R. Wolke, Multirate Runge–Kutta schemes for advection equations, J. Comput. Appl. Math. 226 (2) (2009).
[39] B. Schrader, S. Reboux, I.F. Sbalzarini, Discretization correction of general integral PSE operators for particle methods, J. Comput. Phys. 229 (11) (2010) 4159–4182.
[40] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, Cambridge, UK, 1999.
[41] S. Shankar, L. van Dommelen, A new diffusion procedure for vortex methods, J. Comput. Phys. 127 (1996) 88–109.
[42] O. Shipilova, H. Haario, A. Smolianski, Particle transport method for convection problems with reaction and diffusion, Int. J. Numer. Methods Fluids 54 (10) (2007) 1215–1238.
[43] S. Torquato, Inverse optimization techniques for targeted self-assembly, Soft Matter 5 (6) (2009) 1157–1173, doi:10.1039/b814211b.
[44] Y.-M. Wang, S.-M. Chen, C.-P. Wu, A meshless collocation method based on the differential reproducing kernel interpolation, Comput. Mech. 45 (6) (2010) 585–606.
[45] D. Wee, A.F. Ghoniem, Modified interpolation kernels for treating diffusion and remeshing in vortex methods, J. Comput. Phys. 213 (1) (2006) 239–263.
[46] C.-P. Wu, J.-S. Wang, Y.-M. Wang, A DRK interpolation-based collocation method for the analysis of functionally graded piezoelectric hollow cylinders under electro-mechanical loads, Comput. Model. Eng. Sci. 52 (1) (2009) 1–37.