



Point Cloud Generation for Meshfree Methods: An Overview

Pratik Suchde^{1,2} · Thibault Jacquemin² · Oleg Davydov³

Received: 6 January 2022 / Accepted: 5 September 2022 / Published online: 28 September 2022
© The Author(s) 2022

Abstract

Meshfree methods are becoming an increasingly popular alternative to mesh-based methods of numerical simulation. The biggest stated advantage of meshfree methods is the avoidance of generating a mesh on the computational domain. However, even today a surprisingly large amount of meshfree literature ironically uses the nodes of a mesh as the point set that discretizes the domain. On the other hand, already existing efficient meshfree methods to generate point clouds are apparently not very well known among meshfree communities, which has led to recent work redeveloping existing algorithms. In this paper, we present a brief overview of point cloud generation methods for domains and surfaces and discuss their features and challenges, in particular in the context of applicability to industry-relevant complex geometries.

1 Introduction

Most numerical methods for solving partial differential equations require the generation of a mesh over the computational domain. Despite advances in mesh generation technology and computer hardware, the generation and management of meshes is often the most difficult and time consuming part of the simulation procedure on geometrically complex domains. The efficiency of mesh generation limits the overall accuracy, robustness and speed of the numerical simulation process. Moreover, high quality mesh generation cannot always be entirely automated, and often requires a lot of manual work for complicated domains [107]. Meshfree methods arose in the first instance in order to prevent this need of mesh generation. They have been widely used especially for applications where the computational domain can undergo rapid or huge changes in time, such as large deformations and displacements. In such cases, mesh adaptation or regeneration has to be done automatically and may easily become a computational bottleneck.

As a result, over the past two decades, meshfree methods have become a popular alternative to mesh-based simulations. The initial step of mesh generation is replaced by the generation of a meshfree point cloud that does not need to be topologically connected, unlike meshes, which is an easier task per se. There remains, however, a more subtle question whether *good* meshfree point clouds are much easier to generate than good meshes. Most published research articles on meshfree methods take this for granted and do not discuss how the point cloud generation should be done. Often they test proposed meshfree algorithms on point clouds generated as nodes of meshes which are obtained by a standard mesh generation method. This may lead to the misconception that point cloud generation should be as tough as mesh generation. Therefore it is important to highlight research in which effort has been made to introduce original meshfree point cloud generation methods.

A wide variety of meshfree methods have been developed, based on both weak and strong formulations of the underlying partial differential equations. We refer to [18, 30, 64, 85, 136, 174] for surveys of various types of meshfree methods. The goal of the present paper is to cover the domain discretization process across the entire spectrum of meshfree methods, and is not restricted to any particular subclass of methods. For this, we define meshfree methods in a very broad sense [106], by including all methods in which approximations of unknown functions are determined only from the locations of a scattered set of nodes. This includes the so-called ‘truly meshfree’ methods which require no mesh, pre-defined or otherwise, but also includes

✉ Pratik Suchde
pratik.suchde@itwm.fraunhofer.de; pratik.suchde@uni.lu

¹ Fraunhofer ITWM, Fraunhofer-Platz 1,
67663 Kaiserslautern, Germany

² University of Luxembourg, 2 avenue de l’université,
4365 Esch-sur-alzette, Luxembourg

³ Department of Mathematics, University of Giessen,
Arndtstrasse 2, 35392 Giessen, Germany

meshfree methods with background grids which could be either locally or globally defined, and could be used for post-processing or for integrating weak forms (such that any volume or surface integration required is done independently of the approximation procedure [146]). We also cover particle-based methods such as Discrete Element Methods (DEM) and molecular dynamics.

Among major techniques for unstructured mesh generation [63], *octree* and *advancing front* methods may serve as prototypes for purely meshfree versions. While meshfree octree-type methods have not been developed yet, the advancing front technique appears to be very promising. In fact, automated and efficient meshfree point cloud generation methods of advancing front type are successfully used in a few commercial meshfree codes [62, 140], especially in meshfree fluid solvers. However, these ideas are not very well known in many meshfree communities, with recent meshfree literature sometimes redeveloping the same techniques already in use since two decades.

One difficulty for identifying methods for meshfree point cloud generation in the literature is the variability of the terminology used for this process. The largely unambiguous term of mesh generation has several meshfree counterparts. In addition to the term *point cloud generation*, which we shall use in this paper, the same has also been referred to as ‘point cloud distribution’ or ‘initialization’ [43], ‘point generation’ [115], ‘node placement’ [138, 139], ‘node positioning’ [209], ‘point scattering’ [104], ‘point set generation’ or ‘creation’ [101, 102], ‘particle setup methods’ [65], ‘particle packing’ or ‘sphere packing’ [150], and even ‘model generation’ [69]. In some meshfree communities like Smoothed Particle Hydrodynamics (SPH), where node distributions change with time, it is referred to as the ‘generation of initial conditions’ [40], while in Radial Basis Functions (RBF) based methods the term ‘node generation’ is the most common [59, 179].

In this paper, we present a brief survey of different methods used for point cloud generation in meshfree research and software. We also consider initial domain discretization procedures for time dependent problems. The generation of point clouds of varying spatial resolutions is also covered. However, methods for modification of point clouds during simulations, for example those used to maintain regularity of moving point clouds are out of scope of this paper, despite the overlap of techniques with point cloud generation. The selection of local sets of influence is not considered either.

Another question we do not touch in this paper is how to facilitate visualization of the results of numerical simulations when they are given in the form of values on a point cloud. The problem of approximate evaluation of functions known at scattered locations in more than one variable has its own extensive literature, see e.g. [217].

Until recently, no open-source point cloud generators have been available, but this has changed in the last few years, so that many of the methods presented in this paper may be checked by the reader using open-source software. Milewski introduced a MATLAB point generator in two spatial dimensions [125, 126]. Mishra developed another MATLAB toolbox for two-dimensional point generators [128, 129], based on [59]. In the last three years, three open-source three-dimensional point generators have been developed by Slak and Kosec [88, 184], Negi and Ramachandran [133, 135], and van der Sande and Fornberg [210, 211]. Some simple tools for point cloud generation are available with the package `mFDlab` for meshfree finite difference methods [34]. We note that for the present work, the terms *point*, *node*, and *particle* are equivalent, and are used interchangeably.

We start discussing the topic at hand with the two most popular methods: mesh generation for meshfree methods in Sect. 2, and uniform lattice structures in Sect. 3. We then consider random and quasi-random generators for point clouds in Sect. 4. In Sect. 5, we talk about the use of initially over- and under-sampled point clouds followed by a thinning or filling procedure respectively. Section 6 discusses the use of iterative methods, pre-simulations and so-called relaxation phases. We then present advancing front techniques for point cloud generation in Sect. 7. Point cloud discretizations for boundaries and surfaces are discussed in Sect. 8. In Sect. 9, we give a brief comparison between the different methods, discuss how they could be applied to complex examples and present a numerical solution to a Poisson problem on point clouds generated with different methods. A short conclusion is given in Sect. 10.

2 Mesh Generation for Point Cloud Generation

Across meshfree literature, it is quite common to see meshfree simulations that begin with mesh generation on the computational domain, see for example [24, 72, 82, 121, 144, 153, 160, 219]. Then the vertices of the mesh are used as the global point cloud on which simulations are performed. Alternatively, instead of the vertices, mesh element barycenters may also be used as the nodes of the desired point cloud. Any of the huge variety of existing mesh generating methods can be used for this. An overview of different mesh generators falls outside the scope of the present work, and we refer the reader to the books and surveys [22, 31, 50, 63, 113, 151]. While the use of a mesh generator *seemingly* defeats the main reason for the existence of meshfree methods, this can be justified in certain circumstances as explained below.

Fig. 1 An example of a 2D mesh (left) with near degenerate elements (marked in blue), the nodes of which produce a decent point cloud (right)

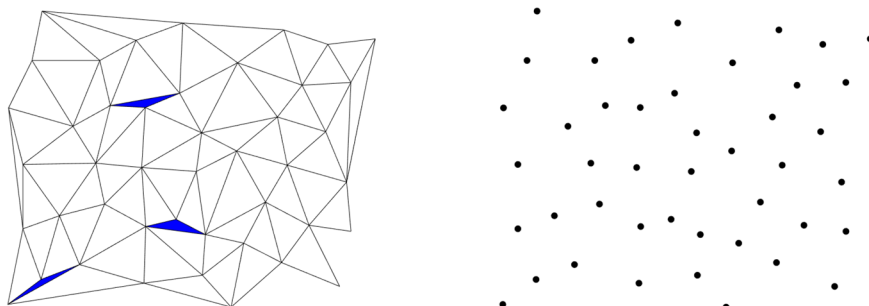
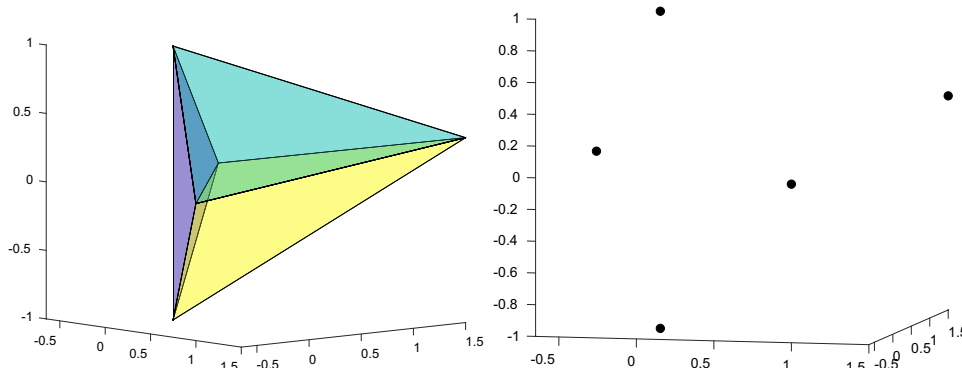


Fig. 2 Delaunay triangulation of five points in 3D with a sliver (left). The five points are nicely distributed as is clearly seen from a different viewpoint (right)



First, using the nodes of a mesh as the point cloud serves as an easy approach for comparing the results between meshfree and mesh-based methods, which has naturally been an essential part in the validation of many meshfree techniques. This is especially relevant for literature that focuses on specific aspects of meshfree techniques development or on the numerical validation of theoretical work. Indeed, a large number of commercial and open-source mesh-based simulation software packages exist that include mesh generators. On the other hand, there are few meshfree point cloud generators, commercial or open-source, none of which were available until recently. As a result, mesh generation for the creation of point clouds has the least overhead in terms of implementation, when compared to other point cloud generation methods.

Second, it is generally expected that the quality requirements of a mesh generated to create a point cloud for a meshfree method is much lower than when the actual mesh is used in the mesh-based simulation. Qualities of the mesh such as skewness, orthogonality, or shape regularity of the elements do not play a very important role in order to obtain a well distributed point cloud. For example, since only the nodes of the mesh are being used, the presence of degenerate mesh elements can often be permitted. A two-dimensional example of this is illustrated in Fig. 1. This advantage is even more relevant in three dimensional domains, where even Delaunay meshes may contain near flat tetrahedra, referred to as slivers [31], as illustrated in Fig. 2. As a result, using an efficient but crude mesh

generator, without subsequent post-processing or optimization may already significantly reduce the cost of point cloud generation as compared to the cost of generating a good mesh. That said, quantifying this advantage remains an open problem, in particular because the quality requirements on a point cloud remain unclear.

Mesh quality is pretty well understood in particular for the finite element method (see, for example, [51, 70, 92, 180]), with criteria like shape regularity of elements motivated in part by theoretical error bounds and confirmed in numerical experiments. Contrary to this, meshfree literature provides little more than a vague understanding that smoothly varying density of points and local regularity (that is, the point cloud locally looking close to optimal sphere packing) is advantageous. Local regularity is often prescribed by the minimum distance between points (often referred to as a ‘separation distance’), and the maximum radius of a ball that does not contain any points of the cloud (‘fill distance’), each of which could be a function of the point density. A few other measures for defining point cloud quality have also been proposed, including spatial disorder measures [7], energy definitions (see Sect. 6), optimal recovery [171], and angle uniformity measures [143]. These measures have been built into various point cloud generation algorithms. However, the impact of local regularity, or any other point cloud quality measure, on the accuracy and stability of meshfree methods has not been studied from either a theoretical or a computational perspective.

Third, meshes may be used at some stages of the algorithm or play an auxiliary role, only contributing a fraction of the total computational cost. Some examples:

- Construction of meshes on each part of a non-disjoint domain decomposition into simpler shapes for which meshing is cheap. If a mesh were to be used directly, extra work is needed at the intersecting regions. On the other hand, if only the nodes are being used, the union of the nodes of each mesh can be taken directly as the point cloud [4, 132, 161]. This method has also been referred to as chimera cloud of points [3].
- Meshfree/particle methods with background grids [17, 84, 201, 208]. Several meshfree methods use so-called background grids for the computation of integrals needed in weak formulations. In this case, since a background grid generation is part of the process, the resulting mesh may also be used for the generation of the point cloud. Since the computation of integrals does not put high demands on the mesh quality, the cost of creating and maintaining background grids is low. The same also holds for particle-in-cell methods [77] and hybrid meshfree-meshed methods [71].
- Lagrangian meshfree methods [15, 196, 203, 215]. For these methods, the most important advantage of the meshfree framework is not the avoidance of creating a mesh, but rather the relative ease of incorporating moving Lagrangian frameworks and dealing with large displacements and deformations of the domain. Deformations in point clouds can easily be fixed with local procedures [197, 198]. Thus, the meshfree equivalent of remeshing, which is typically a global procedure, can be significantly cheaper computationally. As a result, even if the meshing process were still used, other advantages of the meshfree framework are still relevant.

A mesh and, in particular, mesh connectivity information and other data structures created to maintain the mesh may also be used to facilitate various computational procedures, for example for efficient identification of neighbourhoods or boundary points, inside-outside checks and other searching tasks arising in meshfree algorithms. A few decades ago, the use of a background mesh significantly sped up these tasks in meshfree methods [83]. However, with the subsequent advent of efficient neighbour searching algorithms for meshfree methods (for example, [9, 12, 41, 147, 202]), this advantage does not seem to hold anymore.

We conclude this section by some general comments on meshing. Over the years many algorithms have been developed to handle and optimize different ingredients of the popular Delaunay-type meshing process that typically starts by discretizing the edges before a discretization of the faces and finally of the volume for 3D problems. The amount

of research invested into optimizing and polishing meshing contributes very much to the prevalence of the mesh-based methods in numerical simulations. Nevertheless, very often the geometry is not designed considering the constraints of the meshing algorithm. As a result, in more complex cases the definition of the domain may not be appropriate for the desired discretization (e.g. element size, shape, density). The geometry may include nodes, edges or surfaces the user does not want to include in the final discretization. However, they end up in the mesh due to the inherent workflow of Delaunay meshing. Another aspect which limits mesh generators is the above mentioned goal of producing high quality elements. The meshing algorithms use various quality metrics to optimize the mesh in the course of the generation process. However, for more complex problems, the mesh generation may fail if some metrics cannot be met. Therefore, for complex or poorly defined geometries, extensive additional work may be needed to achieve a discretization with desired quality. We report below at the end of Sect. 9.1 about our own experience of meshing a complex geometry.

3 Cartesian Grids, Lattices, and Other Uniform Point Clouds

Beyond the use of the nodes of an unstructured mesh, the most popular method to generate point clouds in meshfree literature is the use of a regular lattice structure. Several different lattice structures can be found in the literature.

- *Cartesian grids*: The most common version of this is a Cartesian grid with uniformly distributed points in each dimension, giving a uniform cubic lattice [34, 36, 89, 93, 141, 145, 164, 181].

In order to reduce the effect of preferred coordinate directions inherent in Cartesian grids, several other lattice based distributions listed below have been used, which are inspired by various crystal lattice structures [47, 58].

- *Body centered cubic (bcc) lattice* This is created by starting with a regular cubic lattice and adding nodes at the center of each cube, and is equivalent to a uniform staggered Cartesian grid. In two dimensions, the bcc lattice is the same as one formed by considering only element centers of a hexagonal tiling. These have been used for point cloud generation, for example, in [152].
- *Cubic closed pack (ccp) lattice* The ccp lattice is created by starting with a regular cubic lattice and adding nodes at the face centers of each cube [109, 189]. It has also been called the face centered cubic (fcc or cF) close packing.

Fig. 3 Point clouds based on a Cartesian grid (left) and hexagonal grid (right) in a two-dimensional domain

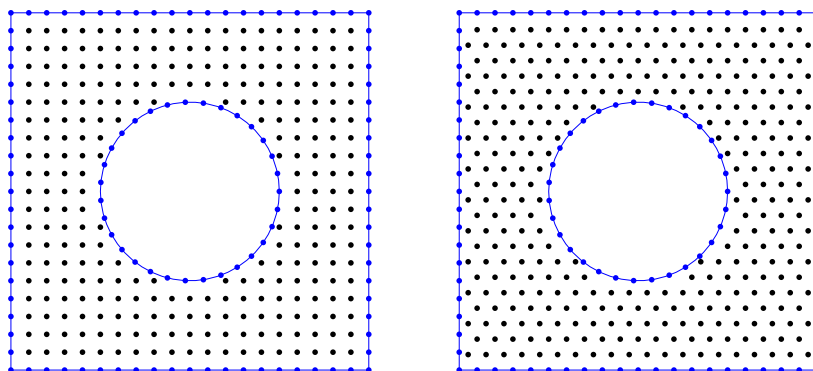
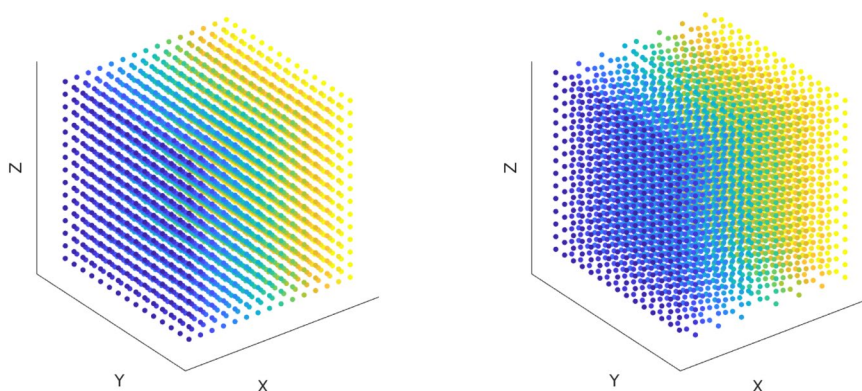


Fig. 4 Point clouds based on a Cartesian grid (left) and a bcc lattice (right) in a three-dimensional domain. The colouring according to x is only to enhance visualization



- *Hexagonally closed-packed (hcp) lattice* The hcp lattice is based an arrangement of nodes in a hexagonal crystal, and has been used to create point clouds for many particle-based simulations [61, 80]. It is also used in two-dimensional pseudospectral and generalized finite difference methods based on radial basis functions [60].

We refer to Diehl et al. [40] for an overview of such crystal lattice structures in the SPH context. A Cartesian grid and a hexagonal hcp grid are illustrated in Fig. 3 for a two-dimensional domain. In Fig. 4, we show a Cartesian grid compared to a bcc lattice for a three-dimensional domain.

Such lattice based point clouds are typically created in the bounding box of the domain. Points outside the domain are then removed. For non-trivial domains, the boundary configuration is obtained in one of three ways:

- The boundary is represented simply by the closest lattice points, as a “staircase” boundary [177, 182].
- Lattice points near the boundary are orthogonally projected to the boundary [34, 36].
- The boundary discretization is computed independently of the lattice. Details on different methods for this are provided in Sect. 8. The union of these boundary points and the lattice points in the interior of the domain form

the point cloud [5, 23, 75, 95]. Optionally, lattice points close to the boundary are deleted [34, 36, 126].

The first two of these cases are illustrated in Fig. 5, with a Cartesian grid for the interior points. An independently created boundary discretization is seen in Fig. 3. This figure also illustrate the fact that this method normally leads to high irregularity of point clouds near boundaries, which may affect the quality of simulations. To tackle this, cloud improvement methods may be applied as in Sects. 5 and 6, which however destroys the grid structure near the boundary. Alternatively, points from a finer grid could be added near the boundary [42].

For complex domains, the identification and deletion of points outside the computational domain can make these methods more expensive than they look. Inside–outside checks may be cheap in level set type methods where the boundary is given by an implicit function, but they can be quite intensive with complex CAD structures or intricate surface meshes defining the boundary.

A single Cartesian grid does not provide spatially varying point densities. To achieve this, over- or undersampling may be used as discussed in Sect. 5. Alternatively, block structured Cartesian lattices [1] consisting of blocks with different spacings may be employed, but we have not seen this in the meshfree literature.

Fig. 5 Boundary points generated from Cartesian grids. A staircase boundary (left), and orthogonal projection to the boundary (right)

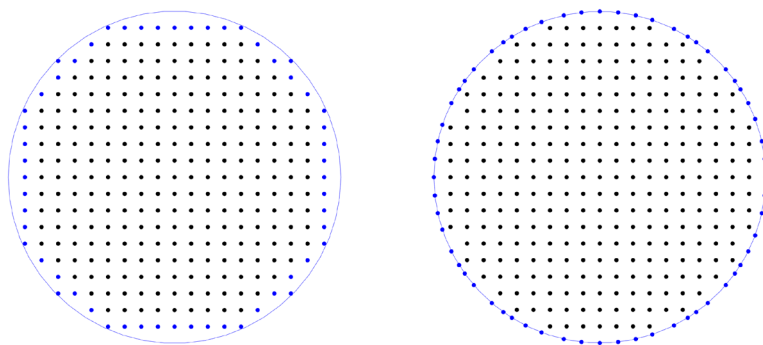
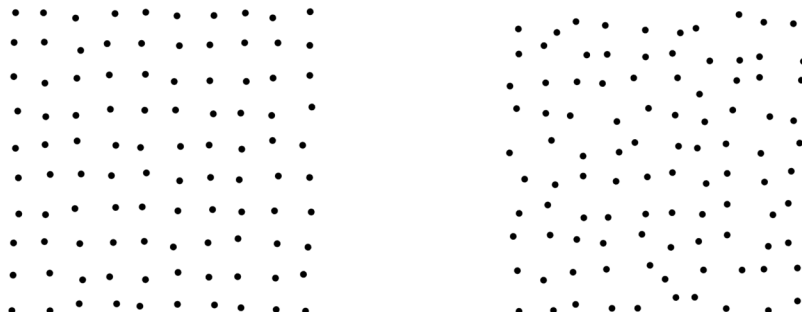


Fig. 6 Examples of 2D point clouds created by perturbing a uniform 10×10 grid. With a small perturbation width $a = 0.15$ (left), and a larger perturbation of $a = 0.3$ (right). See Eq. (1)



Some papers combine a Cartesian grid with additional point clouds that conform to specific boundaries. These methods start with a domain decomposition. For parts of the domain of particular interest, such as the domain around an enclosed boundary, a point cloud is constructed based on the shape of that boundary [166, 167]. For example, in [207], concentric circles are created around a circular enclosed boundary, and a Cartesian grid is used elsewhere in the domain.

Curvilinear lattices used in the structured grid generation do not seem to have appeared in meshfree literature except of very simple model cases like radially symmetric grids for an annulus.

The reasons for the use of Cartesian grids and other lattice structures largely overlap with those listed in the previous section. Each of the lattice based structures are among the simplest to implement, which makes the use of these methods quite popular for testing various meshfree techniques. However, they do not seem to have been developed enough to be used in complex applications.

4 Random and Quasi-random Points

Another frequently seen method to generate point clouds is to use random number generators. This can be done in several ways.

- In meshfree literature, one of the most common methods to create non-uniformly distributed point clouds is

by performing random perturbations of a point distribution given by any of the methods of Sect. 3 [14, 16, 52, 67, 126, 169, 204–206, 208, 209].

An example of this is shown in Fig. 6 for a two-dimensional domain, and in Fig. 7 for a three-dimensional domain. For both examples, a uniform grid is considered on a square (cube in 3D), with uniform spacing δ in each direction. To create the perturbed point cloud, each point in the uniform grids is perturbed using a random number generator. A point indexed i at location $\mathbf{x}_i = (x_i, y_i, z_i)$ is perturbed as

$$\begin{aligned} x_i &= x_i + a \delta \text{rand}(-1, 1), \\ y_i &= y_i + a \delta \text{rand}(-1, 1), \\ z_i &= z_i + a \delta \text{rand}(-1, 1), \end{aligned} \tag{1}$$

where $\text{rand}(-1, 1)$ are pseudo-randomly generated numbers in the range $[-1, 1]$. Limiting the perturbation by $a \delta$ for a perturbation width $a < 0.5$ ensures that there is a minimum distance of $(1 - 2a) \delta$ between every pair of points in the resultant point cloud. In Fig. 6, we illustrate the impact of the perturbation width a , by perturbing a uniform point cloud on a square with $a = 0.15$ and $a = 0.3$. Figure 7 shows a three-dimensional example with $a = 0.3$.

- Several probabilistic algorithms have been employed to create point sets. One such case is the use of randomly generated points, based on a probability distribution describing a prescribed point density [44, 81, 86, 218, 225]. An algorithm of this type that uses so-called

Fig. 7 An example of a 3D point cloud (right) created by perturbing a uniform grid of points (left). The colouring according to x is only to enhance visualization

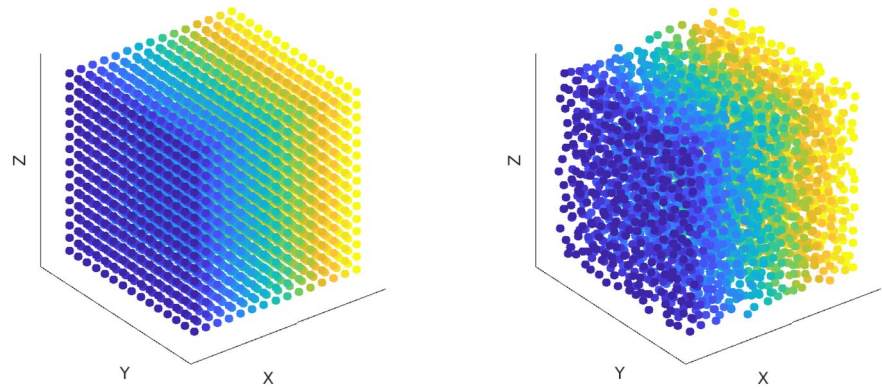
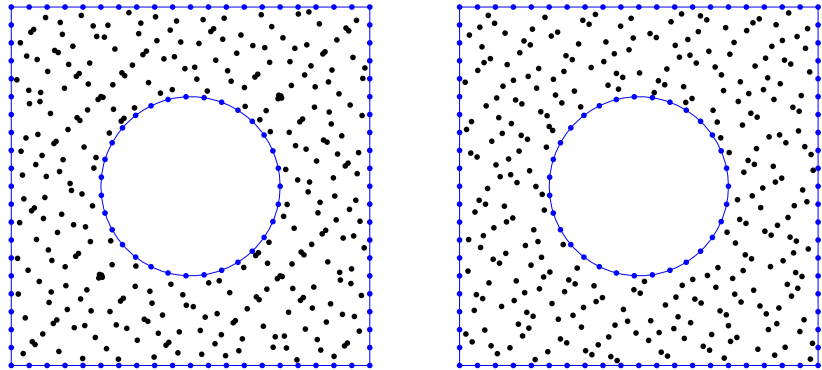


Fig. 8 Sobol (left) and Halton (right) points in a two-dimensional domain



rejection sampling is explained with an example in Sect. 9.1.

- As a replacement for pseudo-random points, quasi-randomly generated points have also been used [55, 73, 127]. This is done with the help of so-called low-discrepancy sequences. Common examples of these are the Sobol sequence [186], and the Halton sequence [76], see Fig. 8. These sequences are a well justified tool for the evaluation of multi-dimensional integrals [39]. Another similar example is the Poisson disk sampling, which has been widely used in graphics communities [26], and is now also used to generate meshfree point clouds [179].

Generation of random or quasi-random points on complex domain boundaries can be quite challenging. Therefore for non-trivial domains the same methods described in Sect. 3 for lattices are adopted: staircase boundary, orthonormal projection, or independent boundary discretization. Similar to the case of uniform grids or lattices, resulting random point clouds may be especially irregular near the boundaries. Moreover, similar to the lattices of Sect. 3, for complex computational domains, randomly generated points would have to be created on a bounding box of the geometry, and inside-outside checks are needed to identify and delete points outside the domain.

These methods are typically employed to illustrate the applicability of particular meshfree techniques on scattered point distributions. It does serve this purpose of validation quite well, with minimum implementation effort. It is worth noting here that open-source implementations for the Halton and Sobol quasi-random point sets are freely available for any spatial dimension. These methods are also popular to generate point clouds which serve as the starting point distribution for iterative methods described in Sect. 6.

5 Over- and Undersampling/Thinning and Filling

The above techniques for point cloud generation, especially lattice-based grids and the random generators, are often combined in different ways, especially to generate point clouds with spatially varying densities. We elaborate on two classes of such methods below.

5.1 Oversampling

The first approach in this context is to start with an excessively fine point cloud, obtained by any of the methods explained in Sects. 3 or 4. This is then coarsened or thinned to achieve the actual point cloud.

Fig. 9 Using a thinning procedure on an oversampled point cloud (left) consisting of 8000 points to obtain the desired point cloud (right), which contains 1318 points

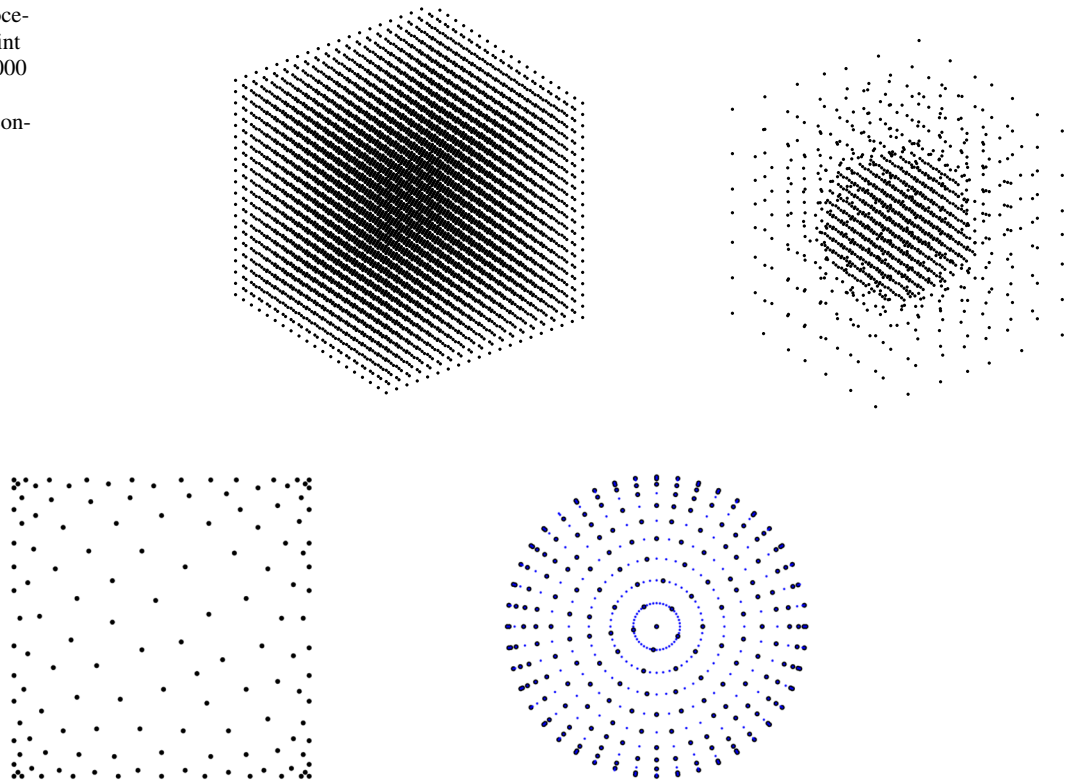


Fig. 10 Approximate Fekete points of degree 15 for the unit square (left) and degree 20 for the unit disk (right). The points for the square are chosen from the 150×150 Cartesian grid whereas those for the

disk from a cloud of points on concentric circles shown in blue. The approximate Fekete points were determined using open source code from [191]

- The thinning process can be done by merging or removing points located too close to each other, based on a prescribed *separation distance* (minimum distance between points), or a desired density of points. An example of such a thinning process is shown in Fig. 9. Starting with a $20 \times 20 \times 20$ grid on $[-1, 1]^3$, points are thinned to ensure a separation given by $r_{\min}h$, for the separation value $r_{\min} = 0.25$, and a point density specified by an inter-point spacing function $h(x, y, z)$ given by

$$h(x, y, z) = 0.3(1 + x^2 + y^2 + z^2). \quad (2)$$

This point cloud density specification used here, and throughout this work, is in terms of point spacing. Thus, higher values of h imply a coarser point cloud. If two points are closer than $r_{\min}h$ apart, they are merged into a single point between the two original points at the arithmetic mean location. This process is repeated until all points satisfy the locally varying minimum separation distance prescribed by $r_{\min}h$. One of the advantages of merging the points in a central location (rather than deleting any one of them) is that it can reduce the influence of any preferred directions present in the original oversampled point cloud.

- Alternatively, the thinning process can also be done by choosing specific points based on desired qualities in the point set. Methods of this type have been developed especially for global polynomial approximation and numerical integration [38, 158, 190].

One such method is the so-called *approximate Fekete points* [190, 226], in which points from an initial oversampled grid are selected by applying a QR decomposition with partial pivoting to the global Vandermonde matrix of a high order polynomial basis. This process leads to point clouds that resemble the distribution of classical Chebyshev points and are supposed to be advantageous for spectral type methods, see Fig. 10. In [108] approximate Fekete points are computed with respect to RBF interpolation matrices. Multi-objective optimization has also been used to select nodes optimized for specific discretization methods [168], here for global RBF collocation.

Such thinning processes are also useful in generating multilevel subsets with decreasing separation distances [57] which are useful for multilevel algorithms [53, 54, 216].

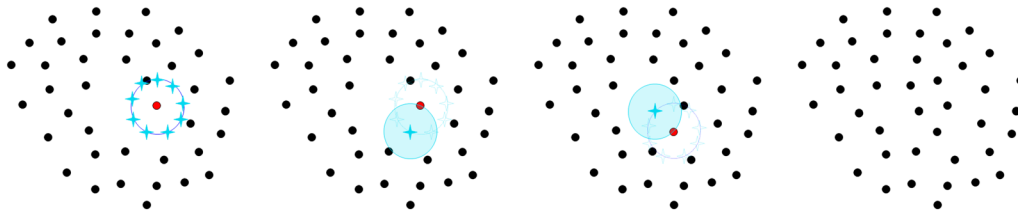


Fig. 11 Filling points using a discrete hole search: Discrete hole locations (left), adding a point (middle two), final point cloud (right)

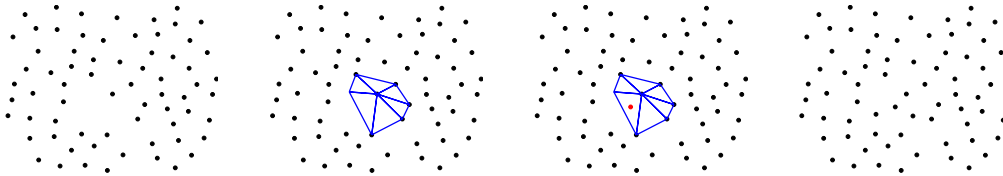


Fig. 12 Filling points using local triangulations: Initial under-sampled points (first), local triangulation (second), adding a point (third), final point cloud after complete addition process (last)

We distinguish this use of oversampled point clouds from another notion also referred to as “oversampling” in which both a fine and a coarse point cloud are used in the simulation, one on the test and another on the trial side of a discretization [96, 172].

5.2 Undersampling

Conversely, some authors start with a very coarse point cloud to discretize the domain. A filling or refinement process is then carried out in order to reach the desired density of points. These are typically based on hole search algorithms to detect regions where points need to be added. Extra points are added in specific regions, for example near boundaries.

One or several points, or even a fine point cloud constructed by any of the methods listed in earlier sections may be added in a part of the domain which is a special region of interest. The main challenges of this approach are to add a point from the finer point cloud only if it is at a minimum separation from all points in the coarse point cloud, and to achieve smoothly varying point densities.

Several approaches for adding points to an undersampled point cloud are known:

- A discrete hole search [176] based on a prescribed point density. An example of this in two spatial dimensions is shown in Fig. 11. At each point a circle (sphere in 3D) is constructed, with the radius dependent on the desired point density at that location. Discrete points on that circle/sphere are then added to the point cloud if they satisfy a minimum separation criterion.

- Hole searching can also be done using locally defined tessellations followed by the addition of points in ‘large’ triangles or tetrahedra, relative to the point spacing required. This process is illustrated in Fig. 12. More details can be found, for example, in [175, 198]. Similar approaches have also been carried out using global tessellations [27]. Point generation using hole search from global tessellations have been referred to as optimization driven point insertion [148, 149].
- Another option is when a pre-defined density function is not given, locations of additional points can be determined by an *a posteriori* error indicator, as is the case in adaptive refinement methods for stationary problems [19, 28, 35, 42, 45, 49, 90, 98, 99, 143, 155, 183, 220]. Similar to adaptive refinement algorithms well known in the finite element method [2], a numerical simulation is performed on the coarse grid, and an approximate solution is computed. An indicator is applied to this temporary solution, in order to determine which parts of the domain require a better resolution by inserting additional points into the cloud. This process is repeated several times until a satisfactory point cloud is created.

In this context, it is important to mention that the ease of adaptive refinement is an important advantage of meshfree methods over mesh-based ones. Due to the presence of an underlying mesh structure in mesh-based methods, adaptive mesh refinement typically encounters issues with conformation and hanging nodes (see, for example, [13, 91, 224] and references therein). In contrast, the lack of topological connectivity in meshfree methods means that adaptive point cloud refinement can be done easily without having to deal

with similar issues. This ease of incorporation of adaptive procedures is not just restricted to the thinning and filling methods listed in this section, but also holds for the iterative point movement methods to be discussed in Sect. 6.

The approaches of starting with oversampled or under-sampled point clouds are especially useful when spatially varying node densities are desired. Additionally, they are also useful when specific global conditions on the point locations are desired.

While the thinning and filling algorithms are essential after starting with an under- or over-sampled point cloud, several of the thinning and filling approaches listed above can also be used as an optional second step to perform local modifications of a point cloud generated by any other method. This can be quite useful to avoid irregularities in a generated point cloud, which occur in several of the methods explained above, especially near the boundary.

Some of the approaches for the deletion/merging of close points from a fine point cloud, and the addition of points in “holes” in a coarse point cloud follow similar (time-dependent) procedures adopted in many Lagrangian meshfree methods [175]. In fully Lagrangian meshfree methods, nodes are moved with an underlying velocity field. This can lead to a distortion in the point cloud. In many meshfree Lagrangian methods (but not all), this is fixed locally by adding points in holes and removing or merging close points, which makes point cloud distortion easier to fix than mesh distortion. All algorithms used in that context can be carried over to point cloud generation by over- or undersampling. We refer to [43, 175, 198, 200] for more details on fixing distortion in Lagrangian meshfree methods. Similarly, various approaches for point cloud thinning or refining can also be adopted from the above mentioned adaptive refinement algorithms in meshfree methods, or from similar notions in point cloud based object reconstruction in a computer graphics context [48, 162].

It must be noted here that the thinning and filling methods to generate a point cloud are often used together. When space varying point densities are desired, it is possible that an initial uniformly spaced grid is under-sampled in some parts of the domain relative to the desired point density, and over-sampled in other parts, thus requiring both filling and thinning algorithms simultaneously.

6 Iterative Methods and Cloud Improvement

In mesh generation, it is common to add a post-processing step in order to iteratively improve the mesh obtained by a primary technique, such as Delaunay triangulation or advancing front method. This is achieved by modifying the nodes or the mesh or both, and the goal is usually the

improvement of the shape quality of the elements, see e.g. [63].

Similar ideas are often used for the improvement of meshfree point clouds created by any of the methods explained in the earlier sections. Moreover, sometimes the initial cloud is a rough first guess, and an iterative point movement procedure becomes the main vehicle for point cloud generation, rather than a polishing tool. This approach may also be found in mesh generation (for example, [156, 185, 223]). We describe below several popular meshfree methods of this type.

6.1 Energy Minimization or Inter-particle Forces

The most common iterative approach is the minimization of a prescribed energy function on the node set [94, 213]. Usually the energy function is defined via an inter-particle repulsion force [134, 221, 222]. Non-uniform point densities can be achieved by prescribing spatially varying energy functionals or inter-particle repulsion forces.

- The most common version of this is one that uses *Voronoi decompositions* [11, 40, 65]. This approach has been widely adopted in the SPH community. In each iteration, a global Voronoi tessellation of the domain is constructed. Particles are then moved using a force

$$\mathbf{F}_i = m(\mathbf{x}_i - \mathbf{z}_i), \quad (3)$$

where \mathbf{x}_i is the location of the point i , \mathbf{z}_i is the location of the centroid of the corresponding Voronoi cell, and m is a constant. Spatially varying particle densities can be achieved by constructing weighted Voronoi tessellations [8, 40]. The movement of particles under such a forcing tries to achieve a (weighted) centroidal Voronoi tessellation (CVT), where every particle is coincident with the centroid of its Voronoi cell. The construction of a global Voronoi tessellation means that these methods are no longer purely meshfree. However, the main character of the methods remain meshfree in nature since all approximations are done only on the point cloud. Moreover, they avoid the main obstacles of mesh generation because there is no goal to obtain a tessellation whose dual Delaunay triangulation is a high quality mesh.

- Other purely meshfree energy minimizers typically rely on the inverse of inter-particle distances, or their logarithms. Commonly used examples are the so-called Riesz energy function and log-energy function [94, 213],

$$E_i^{s,\kappa} = \sum_{\substack{j \in S_i, \\ j \neq i}} \frac{\kappa(\mathbf{x}_i, \mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|^s}, \quad (4)$$

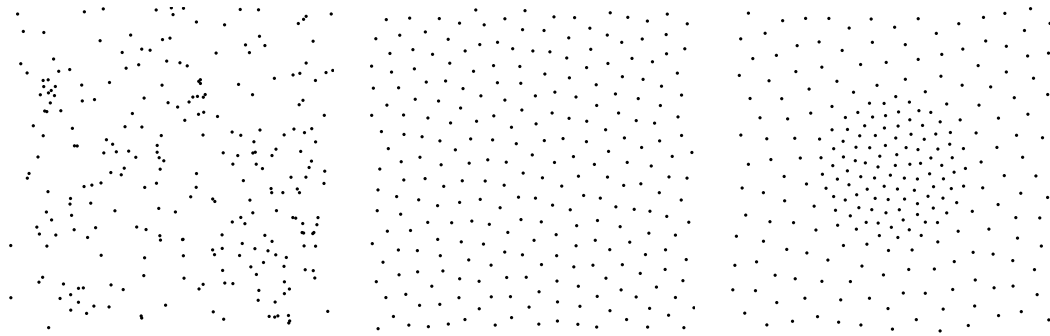


Fig. 13 Using CVT based energy minimization to obtain point clouds. Randomly generated starting point cloud (left), resultant point cloud after a regular CVT based energy minimization (center), and

resultant point cloud after a weighted CVT based energy minimization (right). This figure was generated using open source code from [10, 11]

$$E_i^{s,\kappa} = \sum_{\substack{j \in S_i, \\ j \neq i}} \log \frac{\kappa(\mathbf{x}_i, \mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \tag{5}$$

where S_i is a neighbourhood of point \mathbf{x}_i , the parameter $s > 0$ determines the extent of repulsion, and κ is a function of the point locations used to achieve non-uniform point densities. A specific case of the Riesz energy function, with $s = 2$ in Eq. (4), mimics gravitational or electrostatic repulsion that are proportional to the inverse square distance between particles [59, 211].

Various optimization algorithms have been used to carry out the actual energy minimization, with gradient descent algorithms widely adopted. The Voronoi decomposition based iterative process is generally done using the Lloyd algorithm [111], which is a special case of a gradient descent approach. Here, in each iteration, every particle is moved to the centroid of the corresponding Voronoi cell. Other commonly used strategies include other gradient-based descent methods [110], Monte Carlo [223] and quasi-Monte Carlo [213] approaches.

An example using Voronoi decompositions-based energy minimizations is shown in Fig. 13. Starting with a randomly generated point cloud on $[-1, 1]^2$, a modified Lloyd algorithm [11] is run to create a point cloud with uniform spacing. This is compared with a multiplicatively weighted Voronoi diagram for non-uniform point spacing. Here, the density of the point cloud is prescribed by the function

$$\rho = \max(0.5, 5 \exp(-10(x^2 + y^2))). \tag{6}$$

6.2 Pre-simulations

Another approach is to start from some initial configuration and then perform numerical simulations to obtain the

actual node placement [192]. These simulations can be done based on equations of motion,

$$\frac{d\mathbf{v}}{d\tau} = \mathbf{F}_{\text{ext}} + \mathbf{F}_{\text{contact}} + \mathbf{R}, \tag{7}$$

$$\frac{d\mathbf{x}}{d\tau} = \mathbf{v}, \tag{8}$$

where \mathbf{x} is the position of the particles, \mathbf{v} the velocity and τ is a pseudo time term. Different force terms can be used in Eq. (7) in this context. The most common is a prescribed external force \mathbf{F}_{ext} such as gravity or an external pressure gradient. Optionally, a contact force between particles $\mathbf{F}_{\text{contact}}$ is also used, as is a relaxation term \mathbf{R} , possibly dependent on \mathbf{v} [32], to ensure convergence of the iterations. Other inter-particle interaction forces could also be used, which makes this procedure similar to the energy minimization explained above. The distinction is that the pre-simulation methods use “time” integration procedures, rather than optimization algorithms, to obtain the desired point clouds.

One example is the so-called bubble simulation procedure [138, 139], which prescribes inter-particle forces and damping forces on particles with an assumed finite non-zero diameter. The resultant equations of motions are then solved with an ODE solver until an equilibrium configuration is attained.

A simulation approach is also used to obtain the initial packing of particles in many Discrete Element Method (DEM) simulations [123, 150]. The DEM particles are initially placed in a dilute uniform configuration [123]. DEM simulations are then run on that configuration of particles until they reach a steady state, which is subsequently used as the initial condition for the actual problem being simulated. An example of this is shown for a simple three-dimensional geometry in Fig. 14. Here, starting from a dilute particle configuration, the pre-simulations cause the particles to fall due to gravity and settle

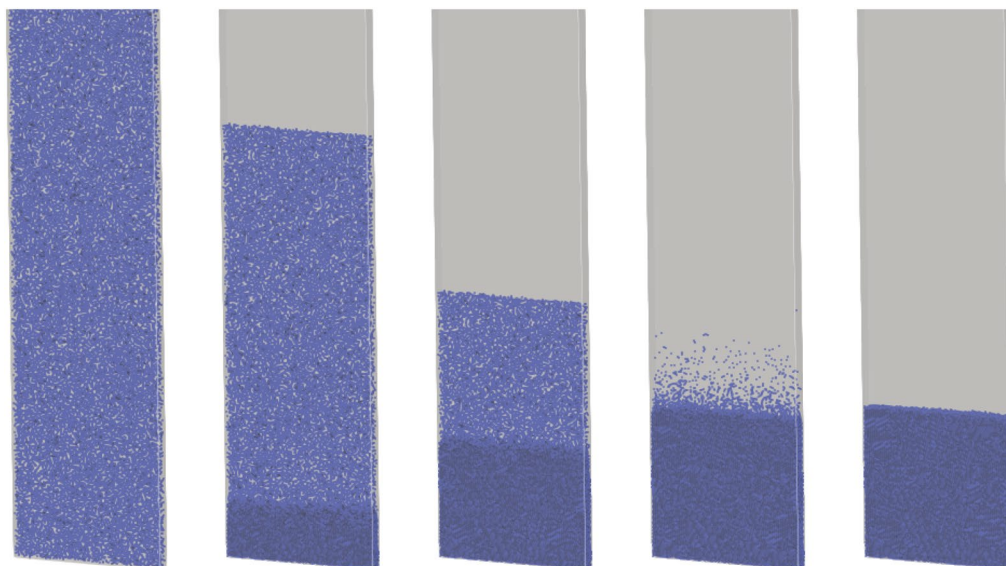


Fig. 14 Obtaining the initial particle packing in a Discrete Element Method (DEM) using a pre-simulation of loosely packed particles falling under gravity

down in a dense configuration. This final state of the pre-simulation is the initial configuration of the particles in the main simulation [142]. Another approach considered by some DEM simulations is to run a pre-simulation with a specified inflow of particles, or random insertion from domain boundaries.

Pre-simulation methods have also been widely used in SPH (for example, [32]). Here, rather than simplified equations of motion, the pre-simulation is done using continuous momentum conservation equations, with an added damping term based on inter-particle distances, or with the use of fictitious pressure potentials [212].

Iterative methods are reported to produce useful point clouds. However, they are rather computationally expensive, since a high number of iterations may be required. In some cases, auxiliary computations such as the Voronoi tessellation significantly add to the cost. Furthermore, the results may heavily depend on the initial placement of points. To reduce the number of iterations required to get to a point cloud with good performance, the initial placement may need to be “improved” [94, 221]. This becomes especially relevant for complex domains. Similar to the methods in the earlier sections, there are issues at the boundary. A further complication in highly complex domains is the need for boundary penetration checks during the movement of the points.

In any case, these methods appear quite useful as a post-processing step following some more efficient point cloud generation with already reasonable results.

7 Meshfree Advancing Front Methods

Advancing front techniques arose in the context of volume mesh generation in the 1980s [112, 137, 154]. These methods start with a surface mesh which discretizes the domain boundaries, which could be either pre-defined (for example, an STL mesh), or computed using a surface meshing technique. This is used to progressively create mesh elements from the boundary inwards until the whole domain is meshed. We refer to [114, 118, 173] for a detailed introduction to advancing front methods for mesh generation and their capabilities.

These techniques were first used for point cloud generation in meshfree methods by Löhner and Oñate over two decades ago [115]. They introduce a meshfree version of the advancing front method in two and three spatial dimensions, with straightforward generalizations for higher dimensions, and demonstrate the use of variable point densities and point generation for complex domains. They further show that meshfree advancing front point cloud generation (henceforth, AFPCG) is an order of magnitude faster than advancing front mesh generation.

This work has been used by many authors to create point clouds for meshfree simulations [119, 122, 124, 155, 170, 194], and has been generalized to different ends [79]. While the original work required a surface mesh, it has been generalized for different boundary specifications [43, 79]. Different methods of specifying the point density

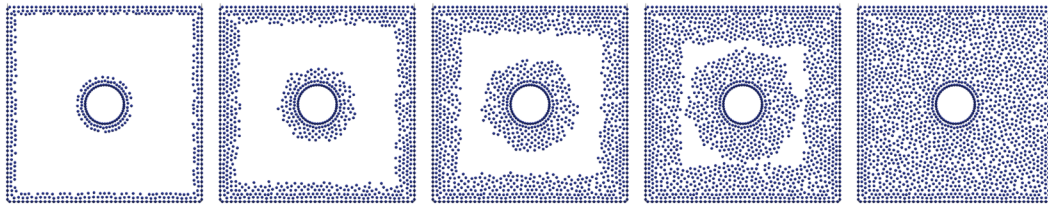


Fig. 15 Generating node distributions in a two-dimensional domain with a uniform point spacing using a meshfree advancing front method. The images show the point cloud after 2, 5, 8, 11, and 15 iterations of interior point generation, respectively

and for advancing the front have been considered [43]. Seibold [175] uses AFPCG, while calling the approach “successive insertion”, and further complements it with point thinning and filling methods like those explained in Sect. 5. The method is also used in a couple of commercial meshfree codes [62, 140], which also report parallel implementations. This work has even been generalized to fill volume domains with objects of different shapes and sizes instead of points [56, 116, 117, 163], such as spheres, ellipsoids, and coils, for the purpose of discrete element method (DEM) simulations.

A similar method was developed independently by Li et al. [101, 102], referring to their advancing front strategy as a biting method. Their method is based on sphere packing and then using the centers of the spheres as the point cloud. They employ a different approach to advancing the front. While [115] advance points in the front directly (details below), [101, 102] require the computation of intersections of arcs. As a result, this algorithm is more computationally expensive.

However, long existing AFPCG methods do not seem to be very well known in many meshfree communities. As a result, a significant amount of recent research has been done into developing similar advancing front methods.

Fornberg and Flyer [59] suggested an AFPCG technique for two-dimensional problems, which was later extended to three-dimensional problems in [211]. This work has become popular in RBF based meshfree methods. The method of [59, 211] has two key differences from older meshfree advancing front methods. Firstly, for curved geometries, they discretize a bounding box of the geometry, and then discard points outside the domain. Furthermore, the initial front is not based on the entire boundary, but rather only on a specific part of it. This simplifies the initial step of boundary discretization; however modifications near the boundary may be needed to better fit the point cloud to it. For those domains where the bounding box is much bigger than the actual computational domain, significant unnecessary computations will be performed.

Slak and Kosec [184] presented an AFPCG technique developed independently of [115] and related work. The main difference to the above mentioned generalizations of

[115] is that certain seed nodes are chosen in the interior of the domain, which results in a different final set of nodes. The work of [184] is also the first one providing an analysis of the computational complexity of a meshfree AFPCG.

Below, we provide an overview of advancing front methods for meshfree point cloud generation. They can be broken down into the following steps:

1. Create a set of points on the boundary. These boundary points act as the initial front(s), which will be advanced.
2. Using the boundary points as sources, a first set of interior points is generated in the domain.
3. Newly generated interior points are then used as a source to create another set of interior points, until the whole domain is filled.

This process is illustrated in Fig. 15 for a two dimensional domain and a uniform density of points.

As mentioned above, advancing front point generation methods start with the creation of a point distribution on each of the (possibly disconnected) domain boundaries. An overview of different methods for doing this is given in Sect. 8. Once the boundary point configuration is computed, the boundary points must be equipped with a surface normal. This normal field is used to prescribe the direction in which the the first set of interior points will be created. This normal field must be consistent in the sense that it should make the discrete surface oriented, i.e., all normals should either be facing inwards or outwards. The normal field can be prescribed directly if already available from the domain boundary specification (see Sect. 8). Alternatively, the normal field can be computed from the boundary discretization using surface normal computation methods [103, 130, 165, 198].

7.1 Advancing the Front

The set of “active” points which are being used as sources to create new points is referred to as the “front” which is advanced. As a point is used to create (or fails to create) new points, it is removed from the front, while the newly added

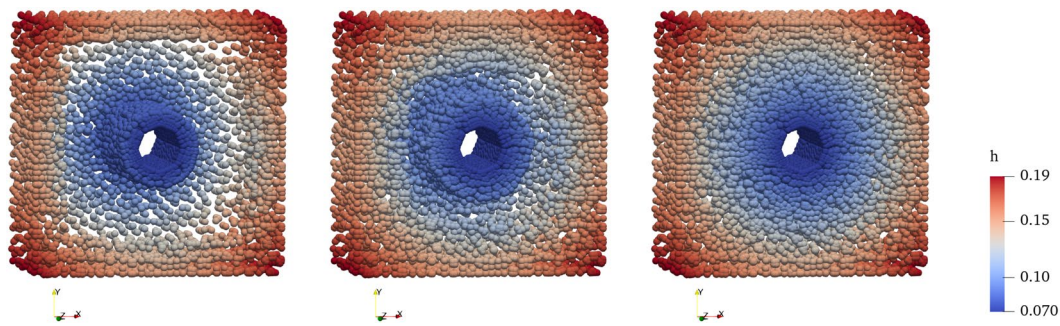


Fig. 16 Generating node distributions in a three-dimensional domain with a varying point density using a meshfree advancing front method. The images show the point cloud after 2, 4, and 7 iterations

points are added to the front. This advancing of the front of active points has also been referred to as a marching process.

Consider a point \mathbf{x}_i being used as a source to generate new points. The point \mathbf{x}_i generates a set of candidate points C_i . Admissible points in C_i are then added to the point cloud and to the active front, and \mathbf{x}_i is removed from the front. The admissibility criteria are as follows:

- Each point should be a prescribed minimum distance away from all other points.
- Points should be inside the domain.

Variations in this framework are obtained by different methods of choosing the candidate points C_i .

- Stencil-based criteria [115]: By considering different stencils, similar to those used in finite difference methods, centered at \mathbf{x}_i .
- By considering points on the surface of a sphere centered at \mathbf{x}_i . The points on the sphere can be chosen either randomly, or by a discrete set on the sphere [59, 184]. Instead of considering points on the surface of a sphere, points could also be considered within an annular spherical region around the candidate point [179].
- By a hole filling algorithm [43, 175]: A discrete hole search is carried out to identify regions where no points are present, and points in the center of identified holes are chosen in C_i .

In each case, the point density specification governs the distances of the candidate points to \mathbf{x}_i . Either by differently sized stencils, or by the radius of the sphere, or the allowed size of the holes.

This front is advanced until the entire domain is filled with points. The minimum specified distance in the admissibility criteria ensures an automatic checking for intersection of the front with itself, or with another disconnected part of the front. When the entire domain is filled, each of the points

of interior point generation respectively. All points are shown as spheres of the same size, with the colour representing the inter-point spacing function h at that location

in the active front will fail to generate new points, thus resulting in an empty front. However, this could produce a “poor” quality point cloud locally where the advancing fronts meet. To fix this, local modifications in these regions can be made either using the thinning and filling algorithms mentioned in Sect. 5, or using a few iterations of iterative algorithms mentioned in Sect. 6 on a few selected points.

Note that merging of the advancing front is one of the biggest challenges in mesh-based advancing front techniques, which is much simpler for meshfree variants since less merging checks are required. As a result, Löhner and Oñate [115] present time comparisons to show that meshfree advancing front techniques were an order of magnitude faster than contemporary volume mesh generators with advancing fronts.

The process of establishing a point cloud using an AFPCG is shown in Fig. 15 for a two dimensional domain with uniform point spacing, and in Fig. 16 for a three dimensional domain with a spatially varying density. Figure 16 shows the discretization of a unit cube with cylindrical obstacle in the middle. Note that points are being filled simultaneously from both the cube boundary inwards and from the cylinder boundary outwards. The point cloud density is prescribed directly as a function of distance from the center of the cube. With a minimum inter-point spacing of $h = 0.07$ at the cylinder boundary, the resolution is linearly increased at a rate of 0.2.

All point clouds generated with the advancing front method are created using the software suite MESHFREE¹, with permission.

8 Discretization of Boundaries and Surfaces

While the bulk of this article focuses on point cloud generation for volume domains, in this section we discuss point generation for surfaces and curves. This is of importance

¹ <https://www.meshfree.eu>

for volume point cloud generation, in the form of boundary discretizations, as explained earlier. It can also be used for the discretization of manifolds. The need for this arises due to the increasing requirement of solving PDEs on manifolds with meshfree methods [29, 60, 66, 74, 100, 103, 157, 187, 188, 199].

As discussed earlier, the topic of point cloud generation for volume domains is a largely overlooked aspect of meshfree methods. This holds even more so for boundaries and surface domains.

Many procedures for meshfree surface discretization are similar to those explained for volume discretization above. Several possibilities are listed below.

- *Parametrization* In certain cases, a parametrization of the domain boundary can be used. The domain could be bounded by a set of curves (for domains in \mathbb{R}^2) or surfaces (for domains in \mathbb{R}^3) for which a parametrization is either known, for example NURBS curves and surfaces [159], or is easy to determine. In this case, the boundary discretization is achieved through a discretization of the parameter space [5, 97, 131, 179]. Obtaining good parametrizations may get extremely complicated for non-trivial geometries. Isogeometric collocation methods [6, 87] could be adopted for parametric boundary discretization in the case of a NURBS geometry. Here points on the surfaces are added by directly discretizing the parameter space defining the NURBS.
- *Surface mesh* For the more general case, with complex domains, the domain geometry is often specified by a surface mesh, which is much easier to generate than a volume mesh of the entire domain. The surface mesh is used to generate points on the surface [43, 69, 115]. This can be done by directly using the nodes or centroids of surface mesh elements. This initial node set can then be refined or coarsened to achieve the desired point density, with surface thinning or filling algorithms [198], similar to those explained in Sect. 5. It must be noted here that many meshfree methods use a surface mesh to define the domain boundary. This is especially relevant for practical applications with complex geometries. Furthermore, for time-dependent geometries, the surface mesh only defines the initial domain boundary.
- *CAD surface* It is often desirable to prescribe the bounding geometry directly by CAD surfaces, without even a surface mesh. For this case, points are placed directly on CAD entities. It must be mentioned that a detailed study of efficient point cloud generation directly on a CAD surface has not been done. While several commercial software packages mention the

ability to go directly from CAD to point clouds, we have not found publications detailing such algorithms.

- *Projection from the interior* As discussed in Sect. 3, boundary points may be generated by projecting interior points to the boundary.
- *Implicit surfaces* A point cloud can be obtained by iso-surface extraction methods, such as the marching cubes algorithm [120], if a signed distance function or another level set function is available, see e.g. [36].
- *Advancing front* Advancing front methods can also be applied to manifold discretizations [46, 198]. Just as in the volume case, the process starts with a discretization of the manifold boundaries, with successive filling in the interior. For a closed manifold with no boundaries, the process would begin with an arbitrary choice of a seed point somewhere in the domain. These methods could be especially useful if the surface is prescribed by CAD data.
- *Iterative procedures and minimum energy points* Iterative energy minimizing procedures as explained in Sect. 6 are also commonly used for boundary discretizations [21, 60]. Similar to the volume point generation case, the biggest advantage of these procedures is that they are reported to produce good quality point sets. Many particular examples of well distributed point clouds are obtained for the sphere in \mathbb{R}^3 by minimizing some energy, maximizing determinants, or from spherical designs [25]. A large collection of such points sets for the sphere can be found in [193].
- *Random points, over- and undersampling* approaches can also be used [179] to discretize boundaries, with procedures similar to that described in Sects. 4 and 5.

9 Comparisons and Applications

In this section we provide a comparison of the features of the different methods, and explain how they can be applied and combined.

Different features of each of the point cloud generation methods discussed above are summarized in Table 1. Among the different methods discussed, iterative and cloud improvement methods of Sect. 6 can serve as a post-processing tool to improve an existing point cloud, and as a stand-alone method starting with a uniform initial point cloud. Similarly, the thinning and filling approaches of Sect. 5 can be used both as a post-processing method, and a stand-alone method starting with an over- or under-sampled point cloud. Random numbers based methods of Sect. 4 are primarily used as a first step in the generation of point clouds, except of the situations where they provide irregular point clouds for testing different meshfree techniques. We note that other than mesh generation for point clouds, each of the other

Table 1 Summary of different types of point cloud generation, the required implementational effort, their applicability to problems with spatially varying densities, and their use as a post-processing method to modify an existing point cloud

Point cloud generation method	Implementation overhead	Spatially varying densities	Post-processing method
Nodes of generated mesh	Low*	✓	
Uniform grids and lattices	Low		
Random points	Moderate	✓	
Over / under sampling	Moderate	Small variations only	
Thinning / filling	High	✓	✓
Iterative/pre-simulation	Moderate	✓	✓
Advancing front	High	✓	

*The implementation overhead for mesh generation is low because of the vast variety of existing open-source and commercial tools for the purpose

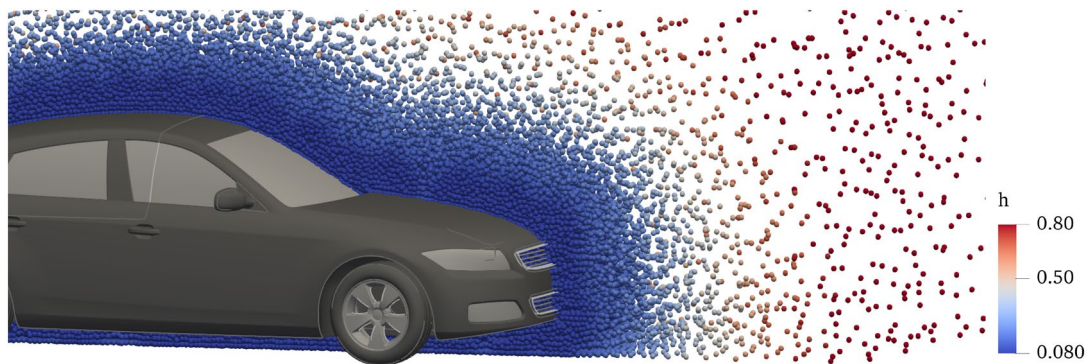


Fig. 17 Point cloud around the DrivAer car model. The colour represents the point density given by the inter-point spacing function h . Only a clip of the point cloud is shown with the half closer to the viewing angle hidden

direct point cloud generation methods mentioned above is usually combined with a certain amount of post-processing.

In terms of modifying or post-processing a given point cloud, iterative methods are very good to prevent points from coming too close together. However, since the number of points is typically fixed in an iterative process, they can only be used to achieve an optimum separation for a given number of points and a point density. These iterative methods can not be used to achieve a prescribed minimum and maximum separation, which is often desired, as is the case in one of the examples below. Thus, for such applications, thinning/filling is a better post-processing option than the iterative methods. Moreover, most thinning/filling algorithms are local in nature, making them easier to parallelize, while the iterative methods are global processes.

To show how the different methods discussed above can be applied and combined, we discuss their use on two complex examples. As mentioned earlier, since there is no well understood measure of what constitutes a good point cloud, comparisons between point clouds generated by different methods can only be done to a limited extent. Thus, we do not attempt to compare the quality of different point clouds, nor do we present complete time comparisons between

different methods. The goal of these examples is only to discuss the applicability of different point cloud generation methods on complex domains, and to highlight various technical details involved when different methods are being applied. We emphasize that performing a thorough comparison of numerical results on point clouds generated is *not* a goal here. The latter likely depends on the choice of the meshfree method and requires its own dedicated investigation that falls out of the scope of the present overview paper.

We start with a detailed discussion on the challenges involved in generating a point cloud with a varying density around a car geometry in Sect. 9.1. We then consider a Poisson problem on a simpler geometry, and present numerical solutions with a generalized finite difference method GFDM on the point clouds generated with different methods.

9.1 DrivAer Car Geometry

We consider the process of point cloud generation around the open source DrivAer car geometry [78],² in which the car model is prescribed by a surface mesh. This geometry and

² The FastBack and engine bay flow configuration is used here.

a point cloud discretizing the domain are shown in Fig. 17. The dimensions of the car model are about $4.6 \times 2 \times 1.42$, with all dimensions in m . The computational domain of interest is the region around the car, inside a box of size $9 \times 4 \times 3$.

For complex geometries such as this one, it is often convenient to prescribe the density of a point cloud as a function of the distance to specific boundaries, as shall be done here. Within a distance of $L_{min} = 0.2$ to specific tagged boundaries (the car geometry here), the point spacing is set to $h_{min} = 0.08$. This is linearly increased at a rate of $dh/dl = 0.5$ till a maximum of $h_{max} = 0.8$ is reached.

We also prescribe a minimum and maximum separation distance between two points. Relative to a given point density, the minimum separation controls the minimum distance between two points, while the maximum separation governs the size of the largest hole allowed in the point cloud. There is no unique way to prescribe the minimum separation and hole sizes. While in some literature, a minimum separation distance is prescribed directly as a function of space, it is also very common to have a point density function prescribed as a function of space, with the minimum separation and maximum hole size given by fixed factors of the density. We use this latter approach here. The minimum separation and maximum hole size are prescribed by fixed factors of the point spacing function h , given by $r_{min} = 0.25h$ and $r_{max} = 0.42h$.

We generate point clouds with several different point generation methods, with details explained below. With the exception of mesh generation for point clouds, all other point cloud generation methods start by generating points on the boundary. Here, we place points on the surface mesh of both the car and the outer box of the domain, such that the prescribed point density is satisfied. The points placed on the car geometry are shown in Fig. 18. Starting with this boundary point discretization, volume point cloud generation in the interior of the domain is done as described below.

9.1.1 Advancing Front Point Cloud Generation

We first consider an AFPCG method to generate the interior points. The front is advanced using a hole filling algorithm, with a prescribed search radius of holes as $0.5(r_{min} + r_{max})$. This ensures that the required minimum and maximum separation are satisfied everywhere, with the possible exception where different advancing fronts intersect. The generated point cloud is then run through with a thinning and filling algorithm, to ensure that the minimum and maximum separation is satisfied even at locations where different fronts intersected. We observe that two thinning and filling cycles each are sufficient. The resultant point cloud generated is shown in Fig. 17. Points around the detailed underbody of the DrivAer car model are shown in Fig. 18.

We note here that for the AFPCG, the distance to the boundary computation required for the point cloud density specification does not need to be explicitly calculated. It can be approximated while advancing the front of active points.

Running the computation serially on a dedicated node, the generation of the point cloud shown in Fig. 17 took 133 s. We note here that the computational time depends on the complexity of the geometry. The surface mesh of the car contains 2.85×10^6 triangles. If the car geometry is replaced with a cube representing its bounding box, filling the domain with the same density of points using the advancing front method takes a fraction of the time: 16s to generate 137346 points, compared to 133s to generate 147947 points with the car geometry. Generating a point cloud with a uniform density of points would be even faster.

9.1.2 Random Point Generation

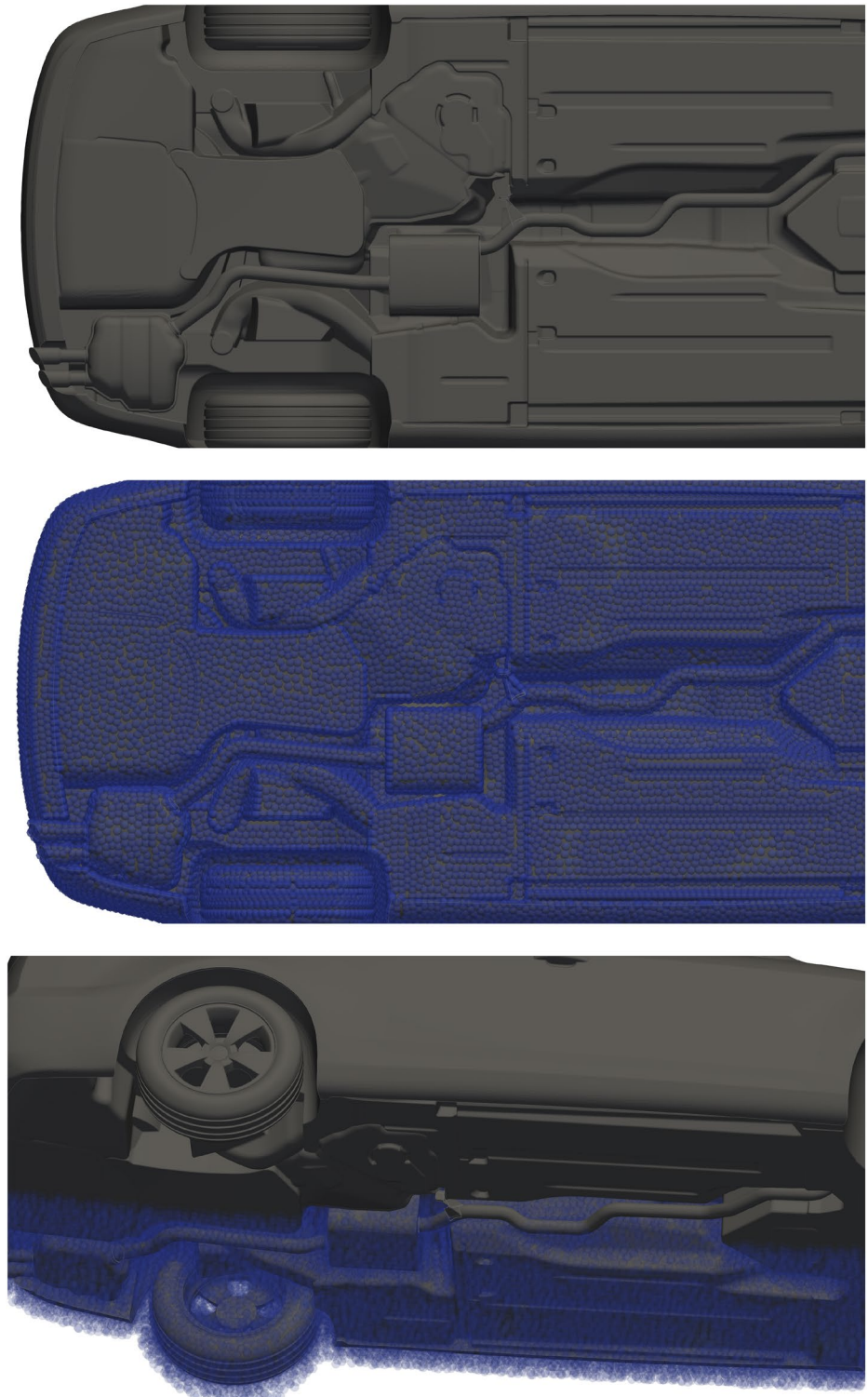
We start by generating points in the bounding box of the domain using a pseudo-random number generator. To take the prescribed point density into account, rejection sampling [214] was used. This is done by considering uniform sampling of points in the domain, with a uniform sampling of a density check variable. A randomly generated point is added to the point cloud only if the corresponding density check variable is greater than the desired density at that location.

The second step after the random point generation is to detect points outside the desired domain, and delete them. In this example, points inside the car geometry need to be removed. This is done by computing a signed distance function (only the sign is important here [20]) using a fast marching method [178].

The final step is to ensure that the point cloud satisfies the desired minimum and maximum separation. This is done using multiple cycles of thinning and filling. The filling process is done with a hole search algorithm which searches for holes of size r_{max} , while the thinning algorithm merges any two points closer than r_{min} apart. We note that the same thinning and filling implementations used in the AFPCG are used here.

Two different randomly generated point clouds are considered here, with different number of randomly sampled points: 50,000 and 140,000. Since the same minimum and maximum separation is used, the final number of points after thinning and filling is similar in both cases, 145,022 in the first case, and 148,441 in the latter. The complete process of interior point generation takes 209 s in the former case, and 183 s in the latter. In the former case, the number of filling cycles required to reach the desired point density is much higher, and thus it takes longer. This is illustrated in Fig. 19.

Fig. 18 Point cloud around the underbody of DrivAer car model. The underbody geometry (top), boundary points only (center), and a clip of the point cloud after 2 iterations of interior point addition in an AFPCG (bottom). Note that the points in the center image are shown with a 50% transparency, while those in the bottom image are shown with a 90% transparency



9.1.3 Uniform Point Cloud

Starting with a spatially uniform point cloud using a Cartesian grid or a uniform lattice is not feasible in such an example with a huge difference in the coarsest and finest resolution required. No matter what the resolution of a considered

uniform grid would be, a large number of thinning and/or filling cycles would be required to attain the desired minimum and maximum separation. We thus do not generate a uniform point cloud for this example.

For the sake completeness, we mention possible adaptations of uniform point clouds that could be considered

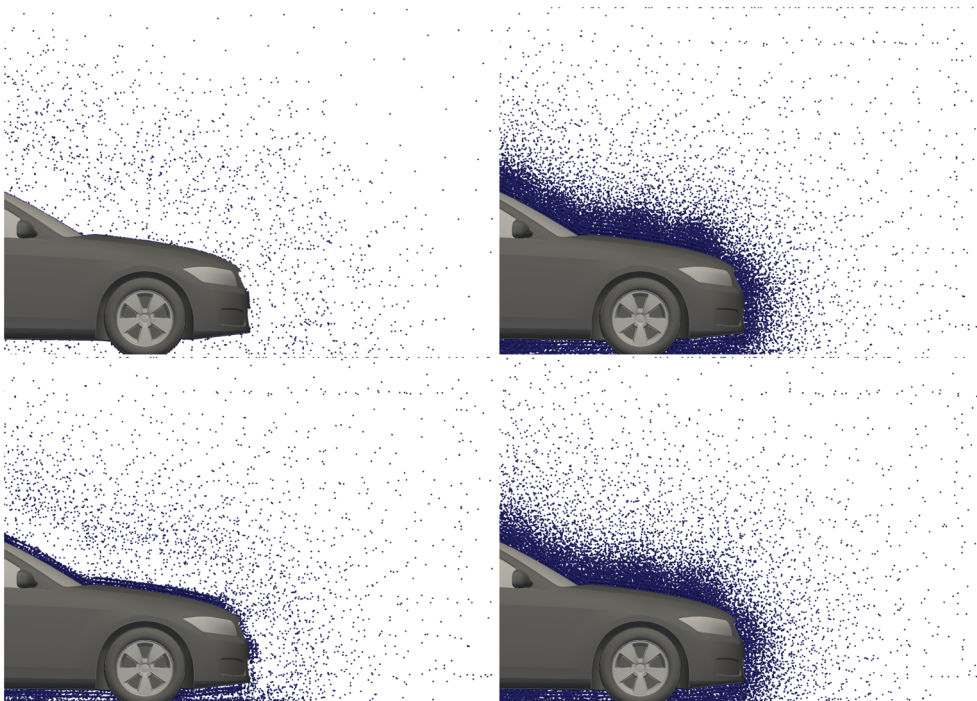


Fig. 19 Random point cloud generation around a car. Initial points clouds after random point generation (left column), and final point clouds after thinning and filling (right column), with 50,000 (top row) and 140,000 (bottom row) randomly sampled points initially, and

comparable final numbers in both cases. Only a clip of each of the point clouds are shown, with the halves closer to the viewing angle hidden

for the current application. One possibility is to consider domain decompositions, and use a different notion of a uniform point cloud. For example, the domain could be partitioned into several blocks with a uniform point cloud in each block for the corresponding mean density there. However, this would result in a non-smooth distribution of points. This drawback could be reduced by partitioning the domain into a high number of blocks, with an octree type structure, based on the desired density.

9.1.4 Mesh Generation

We use Gmsh [70] to create a volume mesh around the car model. The DrivAer car model [78] is available as both CAD and STL surface mesh. In the point cloud generation methods used above, we used the STL surface mesh. Here, we first try to generate a mesh from the CAD prescription directly. However, our attempts at this suggested that a lot of manual work would be required for this because (i) the CAD model included many free edges and very small curves and surfaces. (ii) at several intersections of different parts of the geometry, one surface was tangent to the other. For instance, near the mirror and at the exhaust system. This caused a problem in inserting elements at these regions.

Thus, to reduce the amount of manual work needed, we start with the STL surface mesh. Using Blender [33] and

Fusion360 (Autodesk), we create a simpler CAD model of the car geometry. A CAD geometry was still preferred over working directly with the STL file primarily because the specification of the desired node density as a function of the distance to the CAD model of the car can be done easily. Using the resultant simplified CAD model, we create a mesh with a similar node density in Gmsh. The resultant mesh, and the point cloud formed by the vertices of the mesh are shown in Fig. 20.

A visual comparison of the point clouds generated by the three methods considered here is shown in Fig. 21.

9.2 Poisson Problem

For a representational comparison of point clouds generated by different methods, we consider the numerical solution of a Poisson problem

$$\Delta\phi = f, \quad (9)$$

with Dirichlet boundary conditions. The geometry considered is the “Forearm Link” geometry that is shipped with the MATLAB PDE toolbox, and is shown in Fig. 22. The geometry is defined by a STL surface mesh. The size of the bounding box of the geometry is approximately $134 \times 33 \times 61$.

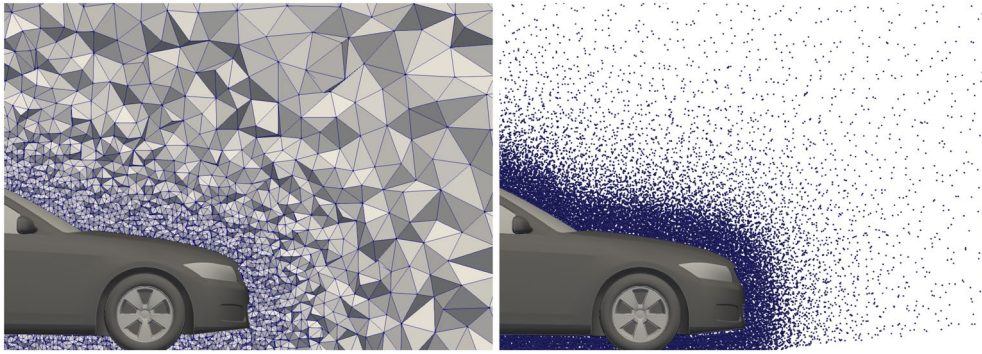


Fig. 20 Point cloud generation around a car using the Gmsh mesh generator. A slice of the mesh (left) and the corresponding clip of the point cloud composed of nodes of the mesh (right). The point cloud only shows the points behind the shown mesh slice

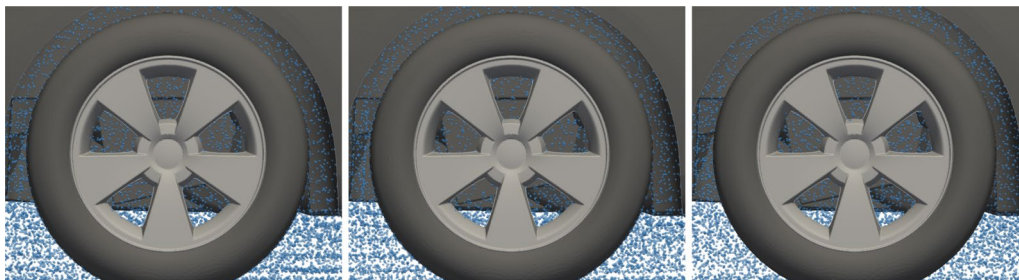
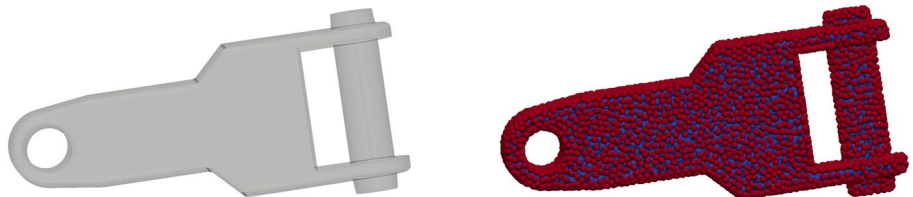


Fig. 21 Comparison of points clouds around the car generated by three methods. AFPCG (left), randomly generated point cloud (center), and nodes of a mesh (right)

Fig. 22 Forearm Link geometry (left), and a point cloud discretizing the computational domain (right). Boundary points are marked in red, and interior points are marked in blue



We consider a manufactured solution to Eq. (9), given by

$$\phi_{\text{exact}} = \sin(x/10), \quad (10)$$

at a point $\mathbf{x} = (x, y, z)$. The right hand side f of the Poisson equation is set such that ϕ_{exact} is a solution of Eq. (9). Boundary conditions are also set to match ϕ_{exact} . Point clouds to discretize the resultant computational domain are generated with four different approaches, each with a spatially constant density.

1. *Nodes of a mesh* We start by generating a volume mesh in Gmsh 4.3.0 [70], using the Delaunay algorithm with default parameters, and a spacing of 2.3.

This results in a mesh with approximately 10^4 vertices, which are used as the point cloud.

2. *Advancing front point cloud generation* We then create a point cloud with an AFPCG using a uniform spacing such that the total number of nodes matches the number of nodes in the mesh as closely as possible.
3. *Cartesian grid* Then using the boundary node set of the AFPCG, we create a uniform Cartesian grid in the interior so as to match the total number of nodes to the first two point clouds as closely as possible.
4. *Randomly generated points* Once again, we start with the boundary node set used in the AFPCG point cloud. Then we randomly generate 1.5×10^4 points in the bounding box of the domain. Points outside the geometry are identified and deleted. This is followed by a few thinning and filling cycles in the interior of the domain. We generated several point clouds with this process till the number of nodes closely matched the number of nodes in the above three point clouds.

Table 2 Poisson problem on Forearm Link geometry with different point generation methods: total number of points N in the domain including both interior and boundary points, the number of boundary points N_{boundary} , and the relative L^2 errors ϵ_{2r} [see Eq. (13)] in numerical solutions of the Poisson problem with GFDM

	N	N_{boundary}	ϵ_{2r}
Mesh nodes	10,802	5931	2.42×10^{-3}
Advancing front	10,631	4292	2.30×10^{-3}
Cartesian grid	10,777	4292	2.46×10^{-3}
Random point cloud	10,026	4292	2.75×10^{-3}

Other than the nodes of the mesh, the other three point clouds were generated within the framework of the software MESHFREE [62]. The point clouds generated by each of the above four methods have been made freely available as supplementary material with this work. While a Cartesian grid was not feasible in the previous example due to the large variance in point densities, a Cartesian grid can be used in the present case since we are using uniform point densities. It is also important to note that no thinning or filling operations were done on the Cartesian grid. As a result, this point cloud has the highest irregularity near the boundary.

The Poisson equation is solved on each of the four point clouds using a meshfree Generalized Finite Difference Method (GFDM), which is a strong form meshfree collocation method. In the GFDM formulation used here, the Laplacian of a scalar valued function u is approximated as

$$\Delta u(\mathbf{x}_i) \approx \sum_{j \in S_i} c_{ij} u(\mathbf{x}_j), \tag{11}$$

where the neighbourhoods or support domains S_i are chosen based on proximity, and are given by the 35 nearest neighbours. We note that a smaller number of neighbours is often used in some GFDM applications, especially for stationary point clouds, while the number of neighbours used here is typical for Lagrangian approaches. The coefficients c_{ij} are computed using a weighted minimization

$$\min \sum_{j \in S_i} w_{ij} c_{ij}^2, \tag{12}$$

subject to the exactness of Eq. (11) for quadratic polynomials. The weights considered here are $w_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^{2\mu}$, with $\mu = 3$. This choice of μ has a theoretical justification given in [37]. For the sake of brevity, we do not present an in-depth introduction to GFDMs here, and refer the interested reader to [68, 85, 105, 195] for more details. For all simulations considered here, the implementations of the GFDM are from the open source repository mFDlab [34].

Errors in the numerical solutions are measured in a relative L^2 sense

$$\epsilon_{2r} = \sqrt{\frac{\sum_{i \in \Omega \setminus \partial\Omega} (\phi_i^h - \phi_{\text{exact}}(\mathbf{x}_i))^2}{\sum_{i \in \Omega \setminus \partial\Omega} (\phi_{\text{exact}}(\mathbf{x}_i))^2}}, \tag{13}$$

where $\Omega \setminus \partial\Omega$ is the set of all interior points in the domain, and ϕ_i^h is the numerical solution to the Poisson problem at point \mathbf{x}_i .

The number of points in the point cloud, and the errors in the solution when using only Dirichlet boundary conditions at all boundaries are summarized in Table 2. We emphasize here that these errors are purely representational and do not serve to indicate the superiority of one point cloud generation method over another. Note that the different errors observed are not just a factor of different point placements, and hence different point cloud qualities, but also of the ratio of boundary points to interior points. For reference, a finite element solution with linear shape functions on the same mesh results in a nodal error of $\epsilon_{2r}(\text{FEM}) = 3.17 \times 10^{-3}$.

We note that the condition numbers of the global discretised linear system (without preconditioning) are 60 and 67 for the mesh nodes and advancing front method respectively on one hand, and significantly higher, 4235 and 1786 for the Cartesian grid and random point cloud respectively on the other hand. This stems presumably from the high irregularity near the boundary for the Cartesian grid and the random point cloud.

10 Conclusion

In this paper, we gave an overview of different methods of point cloud generation, and highlighted their advantages and disadvantages. The aim is not just to collect various methods used in different meshfree communities in one place, but also to encourage dedicated investigation into efficient point cloud generation techniques.

The topic of point cloud generation is often neglected across meshfree literature. Many articles on meshfree methods do not even mention how the domain discretization is achieved. A wide section of meshfree literature still uses the generation of a mesh or lattice structures to obtain the point cloud, primarily because of convenience and familiarity with those methods, which leads to low overhead in terms of implementation. In order to test meshfree techniques on irregular points, point clouds generated using a mesh or lattice structure are often perturbed using random number generation. Several iterative methods are used as a second step after a preliminary point cloud has been generated with some other method. These serve to improve the quality of the point cloud, primarily to ensure a minimum separation between points or to achieve a prescribed spatially varying density of the point cloud. Thinning and filling algorithms

have also been used towards the same end, while also being coupled with initially over- or under-sampled point clouds. Recent work has brought attention back to advancing front point cloud generation.

In addition to the overview we provided examples to illustrate how these different methods can be combined and applied to complex models.

What we presented in this paper is a snapshot of the current state of the point cloud generation methods that are being actively developed by many authors. We believe that meshfree communities may benefit from more effort devoted to in-depth comparison between different existing methods on the basis of a better understanding of what defines a good point cloud. In particular, investigation of quantifiable point cloud quality measures that correlate to numerical stability and accuracy could be helpful.

Acknowledgements Pratik Suchde would like to acknowledge partial support from the European Unions Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Actions Grant Agreement No. 892761. The car DrivAer model is courtesy of the Chair of Aerodynamics and Fluid Mechanics, Technical University of Munich. The authors would like to thank Dr. Jens Bender for the DEM set up and code used to generate Fig. 14, and Dr. Joerg Kuhnert for creating Fig. 11. The authors would like to thank Prof. Stéphane P.A. Bordas for helpful discussions.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Aftosmis MJ, Berger M, Melton J (1998) Adaptive Cartesian mesh generation. CRC Handbook of Mesh Generation (Contributed Chapter), Boca Raton
- Ainsworth M, Oden JT (2000) A posteriori error estimation in finite element analysis. Wiley, Hoboken
- Anandhanarayanan K (2010) Development of three-dimensional grid-free solver and its applications to multi-body aerospace vehicles. Defence Sci J. <https://doi.org/10.14429/dsj.60.583>
- Anandhanarayanan K, Nagarathinam M, Deshpande S (2005) Development and applications of a gridfree kinetic upwind solver to multi-body configurations. In 23rd AIAA applied aerodynamics conference, p 4846
- Angulo A, Pozo LP, Perazzo F (2009) A posteriori error estimator and an adaptive technique in meshless finite points method. Eng Anal Bound Elem 33(11):1322–1338
- Anitescu C, Jia Y, Zhang YJ, Rabczuk T (2015) An isogeometric collocation method using superconvergent points. Comput Methods Appl Mech Eng 284:1073–1097
- Antuono M, Bouscasse B, Colagrossi A, Marrone S (2014) A measure of spatial disorder in particle methods. Comput Phys Commun 185(10):2609–2621
- Aurenhammer F, Edelsbrunner H (1984) An optimal algorithm for constructing the weighted Voronoi diagram in the plane. Pattern Recogn 17(2):251–257
- Awile O, Büyükkeçeci F, Reboux S, Sbalzarini IF (2012) Fast neighbor lists for adaptive-resolution particle simulations. Comput Phys Commun 183(5):1073–1081
- Balzer M (2009) Capacity-constrained Voronoi tessellations. <https://github.com/michaelbalzer/ccvt>
- Balzer M, Schlömer T, Deussen O (2009) Capacity-constrained point distributions: a variant of Lloyd's method. In ACM SIGGRAPH 2009 Papers, SIGGRAPH '09, New York, NY, USA, Association for Computing Machinery
- Band S, Gissler C, Teschner M (2020) Compressed neighbour lists for SPH. Comput Graph Forum 39(1):531–542
- Bangerth W, Kim I, Sheen D, Yim J (2017) On hanging node constraints for nonconforming finite elements using the Douglas–Santos–Sheen–ye element as an example. SIAM J Numer Anal 55(4):1719–1739
- Bašić J, Degiuli N, Ban D (2018) A class of renormalised meshless laplacians for boundary value problems. J Comput Phys 354:269–287
- Bašić J, Degiuli N, Šime Malenica, Ban D (2020) Lagrangian finite-difference method for predicting green water loadings. Ocean Eng 209:107533
- Bašić J, Degiuli N, Blagojević B, Ban D (2022) Lagrangian differencing dynamics for incompressible flows. J Comput Phys 462:111198
- Belytschko T, Lu YY, Gu L (1994) Element-free Galerkin methods. Int J Numer Methods Eng 37(2):229–256
- Belytschko T, Krongauz Y, Organ D, Fleming M, Krysl P (1996) Meshless methods: an overview and recent developments. Comput Methods Appl Mech Eng 139(1):3–47
- Benito J, Urena F, Gavete L, Alvarez R (2003) An H-adaptive method in the generalized finite differences. Comput Methods Appl Mech Eng 192(5–6):735–759
- Bharadwaj A, Ghosh S (2020) Data reconstruction at surface in immersed-boundary methods. Comput Fluids 196:104236
- Borodachov SV, Hardin DP, Saff EB (2019) Discrete energy on rectifiable sets. Springer, New York
- Botsch M, Kobbelt L, Pauly M, Alliez P, Lévy B (2010) Polygon mesh processing. CRC Press, Boca Raton
- Bourantas G, Mountris K, Loukopoulos V, Lavier L, Joldes G, Wittek A, Miller K (2018) Strong-form approach to elasticity: hybrid finite difference-meshless collocation method (fdmcm). Appl Math Model 57:316–338
- Bourantas GC, Zwick BF, Joldes GR, Loukopoulos VC, Tavner AC, Wittek A, Miller K (2019) An explicit meshless point collocation solver for incompressible Navier–Stokes equations. Fluids 4(3):164

25. Brauchart JS, Grabner PJ (2015) Distributing many points on spheres: minimal energy and designs. *J Complex* 31(3):293–326
26. Bridson R (2007) Fast Poisson disk sampling in arbitrary dimensions. *SIGGRAPH Sketches* 10:1278780–1278807
27. Calvo N (2005) Generación de mallas tridimensionales por métodos duales. PhD thesis, Universidad Nacional del Litoral
28. Cavoretto R, De Rossi A (2020) A two-stage adaptive scheme based on rbf collocation for solving elliptic pdes. *Comput Math Appl* 79(11):3206–3222
29. Chen M, Ling L (2020) Extrinsic meshless collocation methods for PDEs on manifolds. *SIAM J Numer Anal* 58(2):988–1007
30. Chen J-S, Hillman M, Chi S-W (2017) Meshfree methods: progress made after 20 years. *J Eng Mech* 143(4):04017001
31. Cheng S-W, Dey TK, Shewchuk J (2012) Delaunay mesh generation. CRC Press, Boca Raton
32. Colagrossi A, Bouscasse B, Antuono M, Marrone S (2012) Particle packing algorithm for SPH schemes. *Comput Phys Commun* 183(8):1641–1653
33. Community BO (2018) Blender—a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam
34. Davydov O (2020) mFDlab: a laboratory for meshless finite difference (mFD) methods. <https://bitbucket.org/meshlessFD/mfdlab>
35. Davydov O, Oanh DT (2011) Adaptive meshless centres and RBF stencils for Poisson equation. *J Comput Phys* 230:287–304
36. Davydov O, Safarpour M (2021) A meshless finite difference method for elliptic interface problems based on pivoted QR decomposition. *Appl Numer Math* 161:489–509
37. Davydov O, Schaback R (2018) Minimal numerical differentiation formulas. *Numer Math* 140(3):555–592
38. De Marchi S, Piazzon F, Sommariva A, Vianello M (2015) Polynomial meshes: computation and approximation. In *Proceedings of the 15th international conference on computational and mathematical methods in science and engineering*, Citese. pp 414–425
39. Dick J, Kuo FY, Sloan IH (2013) High-dimensional integration: the quasi-Monte Carlo way. *Acta Numer* 22:133–288
40. Diehl S, Rockefeller G, Fryer CL, Riethmiller D, Statler TS (2015) Generating optimal initial conditions for smoothed particle hydrodynamics simulations. *Publications of the Astronomical Society of Australia*, 32
41. Domínguez JM, Crespo AJC, Gómez-Gesteira M, Marongiu JC (2011) Neighbour lists in smoothed particle hydrodynamics. *Int J Numer Meth Fluids* 67(12):2026–2042
42. Driscoll TA, Heryudono AR (2007) Adaptive residual subsampling methods for radial basis function interpolation and collocation problems. *Comput Math Appl* 53(6):927–939
43. Drumm C, Tiwari S, Kuhnert J, Bart H-J (2008) Finite pointset method for simulation of the liquid–liquid flow field in an extractor. *Comput Chem Eng* 32(12):2946–2957
44. Du Q, Gunzburger M, Ju L (2002) Meshfree, probabilistic determination of point sets and support regions for meshless computing. *Comput Methods Appl Mech Eng* 191(13):1349–1366
45. Duarte CA, Oden JT (1996) An HP adaptive method using clouds. *Comput Methods Appl Mech Eng* 139(1–4):237–262
46. Duh U, Kosce G, Slak J (2021) Fast variable density node generation on parametric surfaces with application to mesh-free methods. *SIAM J Sci Comput* 43(2):A980–A1000
47. Dunlap RA (2018) Crystalline structure. *Novel microstructures for solids*. Morgan Claypool Publishers, Williston, pp 2053–2571
48. Dyn N, Iske A, Wendland H (2008) Meshfree thinning of 3d point clouds. *Found Comput Math* 8(4):409–425
49. Ebrahimnejad M, Fallah N, Khoei A (2015) Adaptive refinement in the meshless finite volume method for elasticity problems. *Comput Math Appl* 69(12):1420–1443
50. Edelsbrunner H (2001) *Geometry and topology for mesh generation*. Cambridge University Press, Cambridge
51. Ern A, Guermond J-L (2013) *Theory and practice of finite elements*, vol 159. Springer, New York
52. Fang J, Parriaux A (2008) A regularized Lagrangian finite point method for the simulation of incompressible viscous flows. *J Comput Phys* 227(20):8894–8908
53. Farrell P, Gillow K, Wendland H (2017) Multilevel interpolation of divergence-free vector fields. *IMA J Numer Anal* 37(1):332–353
54. Fasshauer GE (1999) Solving differential equations with radial basis functions: multilevel methods and smoothing. *Adv Comput Math* 11(2–3):139–159
55. Fasshauer GE (2007) *Meshfree approximation methods with MATLAB*, vol 6. World Scientific, Singapore
56. Feng YT, Han K, Owen DRJ (2003) Filling domains with disks: an advancing front approach. *Int J Numer Meth Eng* 56(5):699–713
57. Floater MS, Iske A (1998) Thinning algorithms for scattered data interpolation. *BIT Numer Math* 38(4):705–720
58. Flowers P, Theopold K, Langley R, Robinson W (2019) *Chemistry 2e*. Rice University, Houston
59. Fornberg B, Flyer N (2015) Fast generation of 2-d node distributions for mesh-free pde discretizations. *Comput Math Appl* 69(7):531–544
60. Fornberg B, Flyer N (2015) *A primer on radial basis functions with applications to the geosciences*. Society for Industrial and Applied Mathematics, Philadelphia
61. Förster P U (2014) *Turbulence in SPH*. Bachelor’s thesis, Ludwig-Maximilians-University Munich
62. Fraunhofer Society. MESHFREE. <https://www.meshfree.eu>
63. Frey PJ, George P-L (2008) *Mesh generation: application to finite elements*. Wiley, Hoboken
64. Fries T-P, Matthies HG (2004) Classification and overview of meshfree methods. *Informatik-Berichte der Technischen Universität Braunschweig*, 2003-03
65. Fu L, Ji Z (2019) An optimal particle setup method with Centroidal Voronoi Particle dynamics. *Comput Phys Commun* 234:72–92
66. Fuselier EJ, Wright GB (2013) A high-order kernel method for diffusion and reaction-diffusion equations on surfaces. *J Sci Comput* 56(3):535–565
67. Gavete L, Urena Benito J (2016) Generalized finite differences for solving 3D elliptic and parabolic equations. *Appl Math Model* 40(2):955–965
68. Gavete L, Gavete M, Benito J (2003) Improvements of generalized finite difference method and comparison with other meshless method. *Appl Math Model* 27(10):831–847
69. Gerace S, Erhart K, Kassab A, Divo E (2017) A model-integrated localized collocation meshless method (mims). *Comput Assist Methods Eng Sci* 20(3):207–225
70. Geuzaine C, Remacle J-F (2009) Gmsh: a 3-d finite element mesh generator with built-in pre- and post-processing facilities. *Int J Numer Methods Eng* 79(11):1309–1331
71. Ghoneim A (2015) A meshfree interface-finite element method for modelling isothermal solutal melting and solidification in binary systems. *Finite Elem Anal Des* 95:20–41
72. Goyal VK, Huertas CA, Vasko TJ (2013) Smooth particle hydrodynamics for bird-strike analysis using ls-dyna. *Am Trans Eng Appl Sci* 2(2):83–107
73. Griebel M, Schweitzer MA (2002) A particle-partition of unity method-part III: a multilevel solver. *SIAM J Sci Comput* 24(2):377–409

74. Gross B, Trask N, Kuberry P, Atzberger P (2020) Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: a generalized moving least-squares (GMLS) approach. *J Comput Phys* 409:109340
75. Gu Y, Wang L, Chen W, Zhang C, He X (2017) Application of the meshless generalized finite difference method to inverse heat source problems. *Int J Heat Mass Transf* 108:721–729
76. Halton JH (1964) Algorithm 247: radical-inverse quasi-random point sequence. *Commun ACM* 7(12):701–702
77. Harlow FH (1964) The particle-in-cell computing method for fluid dynamics. *Methods Comput Phys* 3:319–343
78. Heft AI, Indinger T, Adams NA (2012) Introduction of a new realistic generic car model for aerodynamic investigations. Technical report, SAE Technical Paper
79. Ho HP, Chen Y, Liu H, Shi P (2005) Level set active contours on unstructured point cloud. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE. vol 2, pp. 655–662
80. Holmes DW, Williams JR, Tilke P, Leonardi CR (2016) Characterizing flow in oil reservoir rock using SPH: absolute permeability. *Comput Part Mech* 3(2):141–154
81. Horton A, Wittek A, Joldes GR, Miller K (2010) A meshless total Lagrangian explicit dynamics algorithm for surgical simulation. *Int J Numer Methods Biomed Eng* 26(8):977–998
82. Huang T-H, Wei H, Chen J-S, Hillman MC (2020) Rkpm2d: an open-source implementation of nodally integrated reproducing kernel particle method for solving partial differential equations. *Comput Particle Mech* 7(2):393–433
83. Idelsohn SR, Onate E (2006) To mesh or not to mesh. That is the question... *Comput Methods Appl Mech Eng* 195:4681–4696
84. Idelsohn S, Oñate E, Pin FD (2004) The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *Int J Numer Meth Eng* 61(7):964–989
85. Jacquemin T, Tomar S, Agathos K, Mohseni-Mofidi S, Bordas SP (2020) Taylor-series expansion based numerical methods: a primer, performance benchmarking and new approaches for problems with non-smooth solutions. *Arch Comput Methods Eng* 27(5):1465–1513
86. Jaworska I (2019) Application of the multipoint meshless fdm to chosen demanding problems. In AIP Conference Proceedings, vol 2078, p 020071. AIP Publishing LLC
87. Jia Y, Anitescu C, Zhang YJ, Rabczuk T (2019) An adaptive isogeometric analysis collocation method with a recovery-based error estimator. *Comput Methods Appl Mech Eng* 345:52–74
88. Jožef Stefan Institute. Medusa. <https://gitlab.com/e62Lab/medusa>
89. Kaennakham S, Chuathong N (2019) An automatic node-adaptive scheme applied with a RBF-collocation meshless method. *Appl Math Comput* 348:102–125
90. Kim H, Atluri S (2000) Arbitrary placement of secondary nodes, and error control, in the meshless local Petrov–Galerkin (MLPG) method. *CMES* 1(3):11–30
91. Kirk BS, Peterson JW, Stogner RH, Carey GF (2006) libmesh: a c++ library for parallel adaptive mesh refinement/coarsening simulations. *Eng Comput* 22(3):237–254
92. Knupp PM (2001) Algebraic mesh quality metrics. *SIAM J Sci Comput* 23(1):193–218
93. Kumar S, Danas K, Kochmann DM (2019) Enhanced local maximum-entropy approximation for stable meshfree simulations. *Comput Methods Appl Mech Eng* 344:858–886
94. Kunc O, Fritzen F (2019) Generation of energy-minimizing point sets on spheres and their application in mesh-free interpolation and differentiation. *Adv Comput Math* 45(5):3021–3056
95. Kwan-yu Chiu E, Wang Q, Hu R, Jameson A (2012) A conservative mesh-free scheme and generalized framework for conservation laws. *SIAM J Sci Comput* 34(6):A2896–A2916
96. Larsson E, Shcherbakov V, Heryudono A (2017) A least squares radial basis function partition of unity method for solving PDES. *SIAM J Sci Comput* 39(6):A2538–A2563
97. Lee CK (2000) A new finite point generation scheme using metric specifications. *Int J Numer Meth Eng* 48(10):1423–1444
98. Lee CK, Shuai Y (2007) An automatic adaptive refinement procedure for the reproducing kernel particle method part II: adaptive refinement. *Comput Mech* 40(3):415–427
99. Lee C, Zhou C (2004) On error estimation and adaptive refinement for element free Galerkin method: part II: adaptive refinement. *Comput Struct* 82(4):429–443
100. Leung S, Zhao H (2009) A grid based particle method for moving interface problems. *J Comput Phys* 228(8):2993–3024
101. Li X-Y, Teng X-Y, Ungor A (2000) Point placement for meshless methods using sphere packing and advancing front methods. ICCES'00, Los Angeles, CA
102. Li X, Teng S, Ungör A (2000) Generating a good quality point set for the meshless methods. *Comput Model Eng Sci (CMES)* 1(1):10–17
103. Liang J, Zhao H (2013) Solving partial differential equations on point clouds. *SIAM J Sci Comput* 35(3):A1461–A1486
104. Lin Q, Rokne J (2006) Construction and analysis of meshless finite difference methods. *Comput Mech* 37(3):232–248
105. Liszka T, Orkisz J (1980) The finite difference method at arbitrary irregular grids and its application in applied mechanics. *Comput Struct* 11(1–2):83–95
106. Liu G-R (2009) Meshfree methods: moving beyond the finite element method. CRC Press, Boca Raton
107. Liu G-R, Gu Y-T (2005) An introduction to meshfree methods and their programming. Springer, New York
108. Liu T, Platte RB (2021) Node generation for RBF-FD methods by QR factorization. *Mathematics* 9(16):1845
109. Liu M, Liu G, Lam K (2003) Coupling meshfree particle method with molecular dynamics novel approach for multiscale simulations. In Advances In Meshfree And X-Fem Methods. World Scientific. pp 211–216
110. Liu Y, Wang W, Lévy B, Sun F, Yan D-M, Lu L, Yang C (2009) On centroidal voronoi tessellation-energy smoothness and fast computation. *ACM Trans Graph (ToG)* 28(4):1–17
111. Lloyd S (1982) Least squares quantization in PCM. *IEEE Trans Inf Theory* 28(2):129–137
112. Lo SH (1985) A new mesh generation scheme for arbitrary planar domains. *Int J Numer Meth Eng* 21(8):1403–1426
113. Lo DS (2014) Finite element mesh generation. CRC Press, Boca Raton
114. Löhner R (1996) Progress in grid generation via the advancing front technique. *Eng Comput* 12(3–4):186–210
115. Löhner R, Onate E (1998) An advancing front point generation technique. *Commun Numer Methods Eng* 14(12):1097–1108
116. Löhner R, Onate E (2004) A general advancing front technique for filling space with arbitrary objects. *Int J Numer Methods Eng* 61(12):1977–1991
117. Löhner R, Onate E (2009) An advancing front technique for filling space with arbitrary separated objects. *Int J Numer Methods Eng* 78(13):1618–1630
118. Löhner R, Parikh P (1988) Generation of three-dimensional unstructured grids by the advancing-front method. *Int J Numer Meth Fluids* 8(10):1135–1149
119. Löhner R, Sacco C, Onate E, Idelsohn S (2002) A finite point method for compressible flow. *Int J Numer Methods Eng* 53(8):1765–1779
120. Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3d surface construction algorithm. *ACM SIGGRAPH Comput Graph* 21(4):163–169

121. Lukyanov A (2019) Meshless numerical methods applied to multiphysics and multiscale problems. PhD thesis, Delft University of Technology
122. Ma Z, Wang H, Pu S (2014) GPU computing of compressible flow problems by a meshless method with space-filling curves. *J Comput Phys* 263:113–135
123. Matuttis H-G, Chen J (2014) Understanding the discrete element method: simulation of non-spherical particles for granular and multi-body systems. Wiley, Hoboken
124. Michel I, Seifarth T, Kuhnert J, Suchde P (2020) A meshfree generalized finite difference method for solution mining processes. *Comput Particle Mech* 18:1–14
125. Milewski S (2017) Generate nodes mesh. https://www.cce.pk.edu.pl/slawek/generate_nodes_mesh.zip
126. Milewski S (2017) Development of simple effective cloud of nodes and triangular mesh generators for meshless and element-based analyses—implementation in Matlab. *Comput Assist Methods Eng Sci* 24(3):157–180
127. Mirzaei D, Schaback R, Dehghan M (2012) On generalized moving least squares with diffuse derivatives. *IMA J Numer Anal* 32(3):983–1000
128. Mishra P (2019) NodeLab. <https://github.com/pankajkmishra/NodeLab>
129. Mishra P (2019) NodeLab: a MATLAB package for meshfree node-generation and adaptive refinement. *J Open Source Softw* 4(40):1173
130. Mitra NJ, Nguyen A (2003) Estimating surface normals in noisy point cloud data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pp 322–328
131. Mokhtaram M, Noor MM, Abd Nazir MJ, Abidin AZ, Yassin AM (2020) Enhanced meshfree rpim with nurbs basis function for analysis of irregular boundary domain. *Malays J Civil Eng* 32(1):1–8
132. Munikrishna N (2007) On viscous flux discretization procedures for finite volume and meshless solvers. PhD thesis, Indian Institute of Science, Bangalore
133. Negi P, Ramachandran P (2021) Algorithms for uniform particle initialization in domains with complex boundaries. *Comput Phys Commun* 265:108008
134. Negi P, Ramachandran P (2019) Algorithms for uniform particle initialization in domains with complex boundaries. [arXiv:1910.07898v3](https://arxiv.org/abs/1910.07898v3)
135. Negi P, Ramachandran P (2019) SPHGeom. https://gitlab.com/pypr/sph_geom
136. Nguyen VP, Rabczuk T, Bordas S, Duflot M (2008) Meshless methods: a review and computer implementation aspects. *Math Comput Simul* 79(3):763–813
137. Nguyen-Van-Phai (1982) Automatic mesh generation with tetrahedron elements. *Int J Numer Methods Eng* 18(2):273–289
138. Nie Y, Zhang W, Liu Y, Wang L (2010) A node placement method with high quality for mesh generation. *IOP Conf Ser* 10:012218
139. Nie Y, Zhang W, Qi N, Li Y (2014) Parallel node placement method by bubble simulation. *Comput Phys Commun* 185(3):798–808
140. NOGRID GmbH. NOGRID. <https://www.nogrid.com>
141. Nomeritae N, Bui HH, Daly E (2018) Modeling transitions between free surface and pressurized flow with smoothed particle hydrodynamics. *J Hydraul Eng* 144(5):04018012
142. Norouzi HR, Zarghami R, Sotudeh-Gharebagh R, Mostoufi N (2016) Coupled CFD-DEM modeling: formulation, implementation and application to multiphase flows. Wiley, Hoboken
143. Oanh DT, Davydov O, Phu HX (2017) Adaptive RBF-FD method for elliptic problems with point singularities in 2D. *Appl Math Comput* 313:474–497
144. Obeidat A, Bordas SPA (2018) Three-dimensional remeshed smoothed particle hydrodynamics for the simulation of isotropic turbulence. *Int J Numer Meth Fluids* 86(1):1–19
145. Obeidat A, Bordas SP (2019) An implicit boundary approach for viscous compressible high reynolds flows using a hybrid remeshed particle hydrodynamics method. *J Comput Phys* 391:347–364
146. Onate E, Idelsohn S, Zienkiewicz O, Taylor R (1996) A finite point method in computational mechanics, applications to convective transport and fluid flow. *Int J Numer Methods Eng* 39(22):3839–3866
147. Onderik J, Durikovic R (2008) Efficient neighbor search for particle-based fluids. *J Appl Math Stat Inform (JAMSI)* 4(1):29–43
148. Ortega E, Onate E, Idelsohn S, Flores R (2013) A meshless finite point method for three-dimensional analysis of compressible flow problems involving moving boundaries and adaptivity. *Int J Numer Methods Fluids* 73(4):323–343
149. Ortega E, Onate E, Idelsohn S, Flores R (2014) Comparative accuracy and performance assessment of the finite point method in compressible flow problems. *Comput Fluids* 89:53–65
150. O’Sullivan C (2011) Particulate discrete element modelling: a geomechanics perspective. CRC Press, Boca Raton
151. Owen SJ (1998) A survey of unstructured mesh generation technology. *IMR* 239:267
152. Pahar G, Dhar A (2016) A robust volume conservative divergence-free ISPH framework for free-surface flow problems. *Adv Water Resour* 96:423–437
153. Pepper DW, Wang X, Carrington DB (2013) A meshless method for modeling convective heat transfer. *J Heat Transf* 135(1):1–7
154. Peraire J, Vahdati M, Morgan K, Zienkiewicz O (1987) Adaptive remeshing for compressible flow computations. *J Comput Phys* 72(2):449–466
155. Perazzo F, Löhner R, Perez-Pozo L (2008) Adaptive methodology for meshless finite point method. *Adv Eng Softw* 39(3):156–166
156. Persson P-O, Strang G (2004) A simple mesh generator in MATLAB. *SIAM Rev* 46(2):329–345
157. Petras A, Ling L, Piret C, Ruuth S (2019) A least-squares implicit rbf-fd closest point method and applications to pdes on moving surfaces. *J Comput Phys* 381:146–161
158. Piazzon F, Sommariva A, Vianello M (2017) Caratheodory–Tchakaloff subsampling. *Dolomites Res Notes Approx* 10(1):82
159. Piegil L, Tiller W (1996) *The NURBS book*. Springer, Cham
160. Praveen C, Deshpande SM (2007) Kinetic meshless method for compressible flows. *Int J Numer Meth Fluids* 55(11):1059–1089
161. Praveen C, Ghosh A, Deshpande S (2009) Positivity preservation, stencil selection and applications of LSKUM to 3-d inviscid flows. *Comput Fluids* 38(8):1481–1494
162. Qi J, Hu W, Guo Z (2019) Feature preserving and uniformity-controllable point cloud simplification on graph. In *2019 IEEE international conference on multimedia and expo (ICME)*, IEEE, pp 284–289
163. Recarey C, Pérez I, Roselló R, Muniz M, Hernández E, Giraldo R, Onate E (2019) Advances in particle packing algorithms for generating the medium in the discrete element method. *Comput Methods Appl Mech Eng* 345:336–362
164. Ren H, Zhuang X, Cai Y, Rabczuk T (2016) Dual-horizon peridynamics. *Int J Numer Meth Eng* 108(12):1451–1476
165. Reséndiz-Flores E, Kuhnert J, Saucedo-Zendejo F (2018) Application of a generalized finite difference method to mould filling process. *Eur J Appl Math* 29(3):450–469
166. Rhee J, Huh J, Kim K, Jung S (2015) Three dimensional meshless point generation technique for complex geometry. In *World Congress on aeronautics, nano, bio, robotics, and energy*. Incheon, Korea

167. Rhee JS, Huh JY, Han D, Kim KH, Jung SY (2016) A development of meshless point generation technique for analysis of the unsteady flow around the multibody. In 30th congress of the international council of the aeronautical sciences, Korea
168. Roque C, Madeira J, Ferreira A (2014) Node adaptation for global collocation with radial basis functions using direct multi-search for multiobjective optimization. *Eng Anal Boundary Elem* 39:5–14
169. Šarler B, Vertnik R (2006) Meshfree explicit local radial basis function collocation method for diffusion problems. *Comput Math Appl* 51(8):1269–1282
170. Saucedo-Zendejo FR (2022) A novel meshfree approach based on the finite pointset method for linear elasticity problems. *Eng Anal Boundary Elem* 136:172–185
171. Schaback R (2015) A computational tool for comparing all linear PDE solvers. Optimal methods are meshless. *Adv Comput Math* 41(2):333–355
172. Schaback R (2016) All well-posed problems have uniformly stable and convergent discretizations. *Numer Math* 132(3):597–630
173. Schöberl J (1997) NETGEN an advancing front 2d/3d-mesh generator based on abstract rules. *Comput Vis Sci* 1(1):41–52
174. Schweitzer MA (2008) Meshfree and generalized finite element methods. Habilitation, Institute for Numerical Simulation, University of Bonn. http://wissrech.ins.uni-bonn.de/research/pub/schweitzer/schweitzer_habilitation.pdf
175. Seibold B (2006) M-Matrices in Meshless Finite Difference Methods. PhD thesis, Kaiserslautern University
176. Seidel T, König C, Schäfer M, Ostermann I, Biedert T, Hietel D (2014) Intuitive visualization of transient groundwater flow. *Comput Geosci* 67:173–179
177. Seleson P, Littlewood DJ (2018) Numerical tools for improved convergence of meshfree peridynamic discretizations. In: Voyiadjis GZ (ed) *Handbook of nonlocal continuum mechanics for materials and structures*. Springer International Publishing, Cham, pp 1–27
178. Sethian JA (1996) A fast marching level set method for monotonically advancing fronts. *Proc Natl Acad Sci* 93(4):1591–1595
179. Shankar V, Kirby R, Fogelson A (2018) Robust node generation for mesh-free discretizations on irregular domains and surfaces. *SIAM J Sci Comput* 40(4):A2584–A2608
180. Shewchuk JR (2002) What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures. Technical report, In Proc. of the 11th International Meshing Roundtable
181. Shi B, Zhang Y, Zhang W (2019) Run-out of the 2015 shenzhen landslide using the material point method with the softening model. *Bull Eng Geol Env* 78(2):1225–1236
182. Silling SA, Askari E (2005) A meshfree method based on the peridynamic model of solid mechanics. *Comput Struct* 83(17–18):1526–1535
183. Slak J, Kosec G (2019) Adaptive radial basis function-generated finite differences method for contact problems. *Int J Numer Meth Eng* 119(7):661–686
184. Slak J, Kosec G (2019) On generation of node distributions for meshless PDE discretizations. *SIAM J Sci Comput* 41(5):A3202–A3229
185. Smirnov A, Zhang H (2008) Physically based node distributions for mesh generation. *Int J Model Simul* 28(2):182–187
186. Sobol I (1967) On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput Math Math Phys* 7(4):86–112
187. Sokolov A, Davydov O, Kuzmin D, Westermann A, Turek S (2019) A flux-corrected RBF-FD method for convection dominated problems in domains and on manifolds. *J Numer Math* 27(4):253–269
188. Sokolov A, Davydov O, Turek S (2017) Numerical study of the RBF-FD level set based method for partial differential equations on evolving-in-time surfaces. In *International workshop on meshfree methods for partial differential equations*. Springer, pp 117–134
189. Soleimani M (2017) Numerical simulation and experimental validation of biofilm formation. PhD thesis, Institut für Kontinuumsmechanik, Gottfried Wilhelm Leibniz Universität
190. Sommariva A, Vianello M (2009) Computing approximate Fekete points by QR factorizations of Vandermonde matrices. *Comput Math Appl* 57(8):1324–1336
191. Sommariva A, Vianello M WAM 2D 280210. <https://www.math.unipd.it/alvise/DIDATTICA/software.html>
192. Song B, Pazouki A, Pöschel T (2018) Instability of smoothed particle hydrodynamics applied to Poiseuille flows. *Comput Math Appl* 76(6):1447–1457
193. spherepts G Wright (2015) <https://github.com/gradywright/spherepts>
194. Su X, Sasaki D, Nakahashi K (2012) A hybrid scheme for the near wall treatment of building cube method. *J Fluid Sci Technol* 7(2):197–208
195. Suchde P (2018) Conservation and Accuracy in Meshfree Generalized Finite Difference Methods. PhD thesis, University of Kaiserslautern, Germany
196. Suchde P (2021) A meshfree Lagrangian method for flow on manifolds. *Int J Numer Meth Fluids* 93(6):1871–1894
197. Suchde P, Kuhnert J (2018) Point cloud movement for fully Lagrangian meshfree methods. *J Comput Appl Math* 340:89–100
198. Suchde P, Kuhnert J (2019) A fully Lagrangian meshfree framework for PDEs on evolving surfaces. *J Comput Phys* 395:38–59
199. Suchde P, Kuhnert J (2019) A meshfree generalized finite difference method for surface PDEs. *Comput Math Appl* 78(8):2789–2805
200. Suchde P, Kuhnert J, Schröder S, Klar A (2017) A flux conserving meshfree method for conservation laws. *Int J Numer Meth Eng* 112(3):238–256
201. Sulsky D, Chen Z, Schreyer H (1994) A particle method for history-dependent materials. *Comput Methods Appl Mech Eng* 118(1):179–196
202. Tapia-Fernández S, Romero I, García-Beltrán A (2017) A new approach for the solution of the neighborhood problem in mesh-free methods. *Eng Comput* 33(2):239–247
203. Tiwari S, Kuhnert J (2003) Particle method for simulation of free surface flows. In: Hou TY, Tadmor E (eds) *Hyperbolic problems: theory, numerics, applications*. Proceedings of the Ninth International Conference on Hyperbolic Problems held in CalTech, Pasadena, March 25–29, 2002. Springer, Berlin, pp 889–898
204. Tota P, Wang Z (2007) Meshfree euler solver using local radial basis functions for inviscid compressible flows. In 18th AIAA computational fluid dynamics conference, p 4581
205. Trask N, Maxey M, Hu X (2016) Compact moving least squares: an optimization framework for generating high-order compact meshless discretizations. *J Comput Phys* 326:596–611
206. Trask N, Perego M, Bochev P (2017) A high-order staggered meshless method for elliptic problems. *SIAM J Sci Comput* 39(2):A479–A502
207. Trask N, Maxey M, Hu X (2018) A compatible high-order meshless method for the stokes equations with applications to suspension flows. *J Comput Phys* 355:310–326
208. Trask N, Bochev P, Perego M (2020) A conservative, consistent, and scalable meshfree mimetic method. *J Comput Phys* 409:109187
209. Trobec R, Kosec G, Šterk M, Šarler B (2012) Comparison of local weak and strong form meshless methods for 2-d diffusion equation. *Eng Anal Boundary Elem* 36(3):310–321

210. van der Sande K Node generation, 2019. https://github.com/kierav/node_generation
211. van der Sande K, Fornberg B (2021) Fast variable density 3-D node generation. *SIAM J Sci Comput* 43(1):A242–A257
212. Vela Vela L, Sanchez R, Geiger J (2018) Alaric: an algorithm for constructing arbitrarily complex initial density distributions with low particle noise for sph/sphhd applications. *Comput Phys Commun* 224:186–197
213. Vlasiuk O, Michaels T, Flyer N, Fornberg B (2018) Fast high-dimensional node generation with variable density. *Comput Math Appl* 76(7):1739–1757
214. von Neumann J (1951) Various techniques used in connection with random digits. In: Householder AS, Forsythe GE, Germond HH (eds) Monte Carlo method, chapter 13, National Bureau of standards applied mathematics series, vol 12. US Government Printing Office, Washington, pp 36–38
215. Wang S, Zhang YO, Wu JP (2018) Lagrangian meshfree finite difference particle method with variable smoothing length for solving wave equations. *Adv Mech Eng* 10(7):1687814018789248
216. Wendland H (1999) Numerical solution of variational problems by radial basis functions. *Approximation Theory IX* 2:361–368
217. Wendland H (2005) Scattered data approximation. Cambridge University Press, Cambridge
218. Wittek A, Joldes GR, Miller K (2019) Meshless algorithms for computational biomechanics of the brain. In: Miller K (ed) Biomechanics of the brain. Springer International Publishing, Cham, pp 273–301
219. Xiao Y, Dong H, Zhan H, Gu Y (2017) A new particle generation method for arbitrary 2d geometries in sph modeling. *Int J Comput Methods* 14(03):1750023
220. Yvonnet J, Coffignal G, Ryckelynck D, Lorong P, Chinesta F (2006) A simple error indicator for meshfree methods based on natural neighbors. *Comput Struct* 84(21):1301–1312
221. Zamolo R, Nobile E (2018) Two algorithms for fast 2d node generation: Application to rbf meshless discretization of diffusion problems and image halftoning. *Comput Math Appl* 75(12):4305–4321
222. Zamolo R, Nobile E, Šarler B (2019) Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from rbf-fd meshless discretizations. *J Comput Phys* 392:311–334
223. Zhang H, Smirnov AV (2005) Node placement for triangular mesh generation by Monte Carlo simulation. *Int J Numer Meth Eng* 64(7):973–989
224. Zhao X, Conley R, Ray N, Mahadevan VS, Jiao X (2015) Conformal and non-conformal adaptive mesh refinement with hierarchical array-based half-facet data structures. *Proc Eng* 124:304–316
225. Zhou D, Seibold B, Shirokoff D, Chidyagwai P, Rosales RR (2015) Meshfree finite differences for vector Poisson and pressure Poisson equations with electric boundary conditions. In: Griebel M, Schweitzer MA (eds) Meshfree methods for partial differential equations VII. Springer International Publishing, Cham, pp 223–246
226. Žitňan P (2011) The collocation solution of Poisson problems based on approximate Fekete points. *Eng Anal Boundary Elem* 35(3):594–599

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.