A PARTICLE-PARTITION OF UNITY METHOD–PART II: EFFICIENT COVER CONSTRUCTION AND RELIABLE INTEGRATION

MICHAEL GRIEBEL† AND MARC ALEXANDER SCHWEITZER†

Abstract. In this paper we present a meshfree discretization technique based only on a set of irregularly spaced points $x_i \in \mathbb{R}^d$ and the partition of unity approach. In this sequel to [13] we focus on the cover construction and its interplay with the integration problem arising in a Galerkin discretization. We present a hierarchical cover construction algorithm and a reliable decomposition quadrature scheme. Here, we decompose the integration domains into disjoint cells on which we employ local sparse grid quadrature rules to improve computational efficiency. The use of these two schemes already reduces the operation count for the assembly of the stiffness matrix significantly. Now, the overall computational costs are dominated by the number of the integration cells. We present a regularized version of the hierarchical cover construction algorithm which reduces the number of integration cells even further and subsequently improves the computational efficiency. In fact, the computational costs during the integration of the nonzeros of the stiffness matrix are comparable to that of a finite element method, yet the presented method is completely independent of a mesh. Moreover, our method is applicable to general domains and allows for the construction of approximations of any order and regularity.

Key words. meshfree method, gridless discretization, partition of unity method, Galerkin method, sparse grids, numerical integration, mesh generation

AMS subject classifications. 65C20, 65N30, 65D30, 65N50

1. Introduction. Meshfree methods (MM) are promising approaches to overcome the problem of mesh generation which still is the most time-consuming part of any finite element (FE) simulation. Meshfree methods are based only on a (finite) collection of independent points within the domain of interest, i.e. there are no fixed connections between any two points like in a conventional mesh. These points can now be used as collocation nodes [1, 9, 10, 11, 19], for the construction of approximate densities [21, 22, 23] or even for the construction of trial and test spaces for a Galerkin method [2, 3, 4, 8, 13].

The shape functions of a meshfree Galerkin method are in general more complex than FE shape functions. In a meshfree method the shape functions are (in general) piecewise rational functions, whereas in a finite element method (FEM) they are piecewise polynomials. This is due to the fact that the construction of a meshfree shape function is based only on independent points instead of a mesh. Therefore, the integration of meshfree shape functions is more complicated than the integration of FE shape functions. Hence, the assembly of the stiffness matrix and right hand side vector in a meshfree Galerkin method is far more expensive than in the FEM. Meshfree Galerkin methods therefore could not be applied to real world problems up to now and are considered to be in an experimental state only.

In this paper we present a numerical method based only on a set of irregularly spaced points and the partition of unity approach [2]. In a partition of unity method (PUM), we define a global approximation u_{PU} simply as a weighted sum of local

†Sonderforschungsbereich 256 *Nonlinear Partial Differential Equations*, Project D *Meshfree numerical methods for the simulation of 3D flows with free boundaries*, Institut für Angewandte Mathematik, Universität Bonn, Wegelerstr. 6, D-53115 Bonn, {griebel, schweitz}@iam.uni-bonn.de

approximations u_i,

$$u_{\mathrm{PU}}(x) := \sum_{i=1}^{N} \varphi_i(x) u_i(x).$$

These local approximations u_i are completely independent of each other, i.e. the local supports $\omega_i := \mathrm{supp}(u_i)$, the local basis $\{\psi_i^k\}$ and order of approximation p_i for every single $u_i := \sum u_i^k \psi_i^k$ can be chosen independently of all other u_j. Here, the functions φ_i form a partition of unity (PU). They are used to splice[1] the local approximations u_i together in such a way that the global approximation u_{PU} benefits from the local approximation orders p_i yet still fulfills global regularity conditions, see [13]. For a general partial differential equation (PDE) $Lu = f$ the fully assembled approximation functions $\varphi_i \psi_i^k$ have to be used within the Galerkin procedure. Hence, for the approximation of a PDE we have to integrate the product functions $\varphi_i \psi_i^k$ in the assembly of the stiffness matrix. This integration is one major issue of concern with partition of unity methods. The dominant factor for the approximation quality of the method certainly is the local basis function ψ_i^k but, other than with the p-version of the FEM, this high order function is *not* the cause of concern during the integration—the partition of unity functions φ_i are.

The algebraic structure of the functions φ_i is (in general) more complex than that of finite element shape functions, since the φ_i have to repair any spatial irregularity induced by the overlaps of the ω_i. Hence, the construction of a partition of unity with a simple algebraic structure is the most crucial step in a PUM. Therefore, the design and implementation of a PUM for general covers $C_\Omega := \{\omega_i \,|\, i = 1, \ldots, N\}$ including a fast yet reliable quadrature scheme is quite involved.

One approach to utilize at least some of the PUM benefits is the so-called *Generalized Finite Element Method* (GFEM) [7, 30]. Here, the construction of the PU $\{\varphi_i\}$ is left to an h-version FEM. On top of this PU, local basis functions ψ_k^i can still be selected with all the freedom the PU approach allows for, e.g. ψ_k^i which are adapted to known local behavior of the solution. But the dependence on the h-mesh construction for the PU is of course a major drawback of the GFEM. Furthermore, one needs to supply appropriate quadrature schemes for the reliable integration of these general local basis functions ψ_k^i independent of the facts that the cover is a mesh and that the partition of unity is piecewise linear.

Truly meshfree Galerkin methods [8, 13] have to be concerned with the construction of a cover from a given set of points $P = \{x_i \in \Omega\}$ and hence have to cope with geometric searching and sorting problems. To tackle these problems an algorithm for finding a set covering a single point based on a tree concept was proposed in [17]. But still the cover C_Ω was assumed to be given.

In this paper we present a general cover construction algorithm based only on a set of irregularly spaced points $P = \{x_i \in \Omega\}$. We partition the domain into overlapping d-rectangular patches ω_i which we assign to the given points x_i to cover the complete domain. We use d-binary trees (binary trees, quadtrees, octrees) for the construction of these patches ω_i. While the data structures used here are similar to those used

[1] Due to this splicing property of the PU one is tempted to construct the global solution u_{PU} by solving only local problems for the u_i on the support patches ω_i. Here, one would use the local basis functions ψ_i^k as trial and test functions and disregard the partition of unity functions φ_i during the construction of the global solution u_{PU}. For the mass matrix problem—and related interpolation problems—this is a proper approach which directly leads to a system of linear equations already in block diagonal form without any lumping.

in [17], we are interested in the *construction of a cover* C_Ω with desirable features. For instance, the subsets $N_x \subset C_\Omega$ of cover patches ω_k which cover a point $x \in \Omega$, i.e. $N_x = \{\omega_k \in C_\Omega \,|\, x \in \omega_k\}$, should be small. Although a cover C_Ω generated by this general algorithm is minimal in the sense that $\mathrm{card}(N_x) \ll \mathrm{card}(C_\Omega)$ is very small for all $x \in \Omega$, the resulting partition of unity functions φ_i are (in general) still more complex than in the GFEM. The piecewise character of the constructed functions φ_i though can further be significantly simplified by making slight changes to the general cover construction. With this refined algorithm we construct covers for general domains that stay close to k-irregular grids. This construction not only minimizes the number of patches $\mathrm{card}(N_x)$ which cover a single point $x \in \Omega$, but also leads to partition of unity functions φ_i with simpler algebraic structure. A k-irregular grid is completely sufficient for a PUM, since the φ_i will repair the jump within the spatial resolution and ensure the global regularity conditions imposed on the approximation u_{PU}. At the same time, the cost of the assembly of the stiffness matrix and right hand side vector is significantly reduced since the simpler algebraic structure of the φ_i allows for cheaper quadrature rules.

Furthermore, we introduce a numerical quadrature scheme for the fully assembled approximation functions $\varphi_i \psi_i^k$ which further reduces the integration costs. Here, we decompose the integration domain to resolve the algebraic structure of the partition of unity functions φ_i; i.e. we decompose the integration domains into disjoint cells on which the integrands are smooth functions. We then use a sparse grid quadrature rule [12] with a dynamic stopping criterion locally on the cells. Hence, the number of integration points on each cell is minimal with respect to accuracy. It turns out that overall, the number of operations needed by our method during the assembly of the stiffness matrix is comparable to that of a finite element method. Yet at the same time it is a truly meshfree method, i.e. it is completely independent of a mesh. Hence, our partition of unity method is a very flexible and efficient numerical discretization technique and it is a strong competitor for conventional finite element methods. With the proposed method the treatment of real world problems using meshfree Galerkin methods might now be in reach now.

The remainder of the paper is organized as follows: in §2 we give a short review over the construction of partition of unity spaces for meshfree Galerkin methods. We then present in §3 a general hierarchical cover construction algorithm based only on a set of irregularly spaced points which allows for a fast neighbor search. Here, we make use of d-binary trees (quadtrees, octrees, etc.) to assign parts of the domain to each of the given points to cover the complete domain. The Galerkin discretization of a PDE using PUM shape functions is given in §4. In §5 we introduce an appropriate numerical quadrature scheme for PUM shape functions. The scheme is based on a decomposition approach to resolve the piecewise character of the partition of unity functions φ_i. On the cells of this decomposition we employ local sparse grid quadrature rules with a dynamic stopping criterion. This reduces the computational costs on each cell substantially, yet still ensures a reliable accuracy of the integration. In §6 we present a refinement of the general cover construction algorithm given in §3; a similar cover construction algorithm was recently proposed in [18]. It also accounts for the geometric neighboring relations of the partition of unity functions φ_i, i.e. the neighboring relations of the cover patches ω_i, which have a significant effect on the number of integration cells. With this improved algorithm the number of integration cells and the corresponding computational effort during the integration of the stiffness matrix entries is significantly reduced. Numerical results for elliptic problems in two and three dimensions are given in §7.

2. Construction of trial and test spaces. In the following, we give a short recap of how to construct partition of unity spaces for a meshfree Galerkin method, see [13] for details. The starting point for any meshfree method is a collection of N independent points $P := \{x_i \in \mathbb{R}^d \,|\, x_i \in \overline{\Omega}, i = 1, \ldots, N\}$. In the PU approach we need to construct a partition of unity $\{\varphi_i\}$ on the domain of interest Ω to define an approximate solution

$$u_{\mathrm{PU}}(x) := \sum_{i=1}^{N} \varphi_i(x) u_i(x), \tag{2.1}$$

where the union of the supports $\mathrm{supp}(\varphi_i) = \overline{\omega_i}$ covers the domain $\overline{\Omega} \subset \bigcup_{i=1}^{N} \omega_i$ and $u_i \in V_i^{p_i}(\omega_i)$ is some locally defined approximation of order p_i to u on ω_i. Given a cover $C_\Omega = \{\omega_i \,|\, i = 1, \ldots, N\}$ we then can define such a partition of unity and local approximations u_i by using *Shepard functions* as φ_i and local approximation spaces $V_i^{p_i}$ on the patches ω_i.

A naive approach toward the construction of such a cover C_Ω would be the design of patches $\tilde{\omega}_i$ in such a way that every given point $x_i \in \tilde{\omega}_j$ for some $j \neq i$. But this procedure (in general) does not lead to a cover of the complete domain Ω, i.e. $\bigcup_j \tilde{\omega}_j \not\supset \overline{\Omega}$, since the points $x_i \in P$ may not be uniformly distributed in the domain Ω. In [13] the following algorithm was proposed which tackles this problem by using a set $\tilde{P} = P \cup Q$ of original points $P = \{x_i\}$ and additional (user supplied) points $Q = \{\xi_k\}$. Note that the additional points ξ_k are introduced to guarantee that the patches ω_i completely cover the entire domain Ω. However, they are *not* used to construct additional cover patches, i.e. the algorithm constructs a cover patch ω_i only for $x_i \in P$ not for $\xi \in \tilde{P} \setminus P$.

Algorithm 1 (Direct Cover Construction).

1. Given: the domain $\Omega \subset \mathbb{R}^d$, a scalar $\alpha \geq 1$, the set of points $P = \{x_i \in \mathbb{R}^d \,|\, x_i \in \overline{\Omega}, i = 1, \ldots, N\}$ for the partition of unity construction and a set of points $Q = \{\xi_i \in \mathbb{R}^d \,|\, \xi_i \in \overline{\Omega}\}$ to resolve the domain Ω.
2. Set $\tilde{P} = P \cup Q$.
3. For all $x_k \in P$: Set $\omega_k := \bigotimes_{i=1}^{d} [x_k^i - h_k^i, x_k^i + h_k^i]$ with $h_k^i = 0$ for all $i = 1, \ldots, d$.
4. For all $y \in \tilde{P}$:
   (a) Set $R = 0$. Evaluate the set $S_{y,R}$ of all points $x_k \in P$ that fall within a searching square B_R which is centered in y and whose side length is equal to $2R$. If $S_{y,R} = \emptyset$ (or $S_{y,R} = \{y\}$ if $y = x_k$), increase the size of the searching square, i.e. R, and try again.
   (b) Compute the distances $d_{y,k} := \|y - x_k\|$ for all $x_k \in S_{y,R}$ with $x_k \neq y$.
   (c) Determine a point $x_l \neq y$ with $d_{y,l} = \min_k d_{y,k}$.
   (d) If $y \notin \omega_l$ increase h_l^i for all $i = 1, \ldots, d$ appropriately such that $y \in \omega_j$ holds.
5. For all $x_k \in P$: Set $\omega_k = \alpha \omega_k$, i.e. $h_k^i = \alpha h_k^i$ for all $i = 1, \ldots, d$.

Now that we have found a cover C_Ω of the domain Ω, we construct a partition of unity $\{\varphi_i\}$ by defining weight functions W_i on the cover patches ω_i. From these weight functions W_i we can easily generate a partition of unity by Shepard's method, i.e. we define

$$\varphi_i(x) = \frac{W_i(x)}{\sum W_k(x)}. \tag{2.2}$$

Since the cover patches ω_i constructed by Algorithm 1 are d-rectangular, i.e. they are products of intervals, the most natural choice for a weight function W_i is a product of one-dimensional functions, i.e. $W_i\,(x) = \prod_{l=1}^{d} W_i^l\,(x^l) = \prod_{l=1}^{d} \mathcal{W}\,(\frac{x-x_i^l+h_i^l}{2h_i^l})$ with $\mathrm{supp}(\mathcal{W}) = [0,1]$ such that $\mathrm{supp}(W_i) = \overline{\omega_i}$. It is sufficient for this construction to choose a one-dimensional weight function $\mathcal{W}$ which is non-negative. Throughout this paper we use normed B-splines [27] as the generating weight function $\mathcal{W}$.

In general, a partition of unity $\{\varphi_i\}$ can of course only recover the constant function on the domain Ω. Hence, the consistency error in the L^2-norm of a discretization with the $\{\varphi_i\}$ would be of first order only. Therefore, we need to improve the approximation quality to use the method for the discretization of a PDE. To this end, we multiply the partition of unity functions φ_i locally with polynomials. Since we use d-rectangular patches ω_i only, a local tensor product space is the most natural choice. Throughout this paper, we use complete Legendre polynomials[2] as local approximation spaces $V_i^{p_i}$, i.e. we choose $V_i^{p_i} = \mathrm{span}\,(\{\psi_i^k \,|\, \psi_i^k = \prod_{l=1}^{d} \mathcal{L}_i^{k_l}, \sum_{l=1}^{d} k_l \leq p_i\})$ where $\mathcal{L}_i^{k_l}$ is the one-dimensional Legendre polynomial of degree k_l on the interval $[x_i^l - h_i^l, x_i^l + h_i^l]$.

The selection of optimal local approximation orders p_i and basis functions ψ_i^k are by nature problem-dependent. The regularity of the analytical solution space, e.g. $H^k(\Omega)$ or $L^p(\Omega)$, and information about the analytical solution u itself may provide some insight. This and other adaptivity related issues are subject of future research and will not be treated in this paper.

Following the construction given above, we can construct approximate solutions u_{PU} of any order and regularity without additional constraints on the cover C_Ω. The resulting shape functions $\varphi_i\psi_i^k$ though have some surprising properties.

1. The partition of unity functions φ_i are (in general) non-interpolatory. Furthermore, there are more degrees of freedom in a PUM space than there are points $x_i \in P$ due to the use of (multi-dimensional) local approximation spaces $V_i^{p_i}$.
2. The regularity of the shape functions $\varphi_i\psi_i^k$ is independent of the number of degrees of freedom. The shape functions inherit the regularity of the partition of unity functions φ_i (if we assume that the local approximation spaces $V_i^{p_i}$ are at least of the same regularity). Therefore, we can increase the regularity of an approximation u_{PU} by changing the B-spline used in (2.2) independent of the local approximation spaces $V_i^{p_i}$. Note that this is different from finite element methods. In the FEM the global regularity of an approximation is given by the element regularity which on the other hand is implemented by constraints imposed on the local degrees of freedom. Hence, a higher regularity may only be achieved by increasing the number of degrees of freedom of an element.
3. The PUM shape functions are piecewise rational functions due to the use of piecewise polynomial weights in (2.2).

The cover C_Ω itself influences the computational work of the method significantly. For one, the neighbor relations $\omega_i \cap \omega_j \neq \emptyset$ of the cover C_Ω already define the sparsity pattern of the stiffness matrix. Second, the evaluation of a single partition of unity function φ_i, see (2.2), involves the evaluation of the weights of all neighboring patches $N_i := \{\omega_j \in C_\Omega \,|\, \omega_i \cap \omega_j \neq \emptyset\}$. Hence, it is necessary to control the number of neigh-

[2]Note that due to their product structure the shape functions $\varphi_i\psi_i^k$ of our PUM do not inherit orthogonality properties of the local basis functions ψ_i^k. Hence, the chosen Legendre polynomials $\mathcal{L}^k$ will not lead to a diagonal matrix for the mass matrix problem, as well as the integrated Legendre polynomials $\mathcal{L}_I^k$ will not lead to a diagonal matrix for the Poisson problem.

bors $\mathrm{card}(N_i)$ to limit the computational work during the assembly of the stiffness matrix. Furthermore, the smoothness of a PU function φ_i is strongly dependent on the amount of overlap $\omega_i \cap \omega_j$ of the neighboring patches $\omega_j \in N_i$, see [13] for details.

3. Hierarchical Cover Construction. Therefore, we need to construct a cover C_Ω which minimizes the number of neighbors $\mathrm{card}(N_i)$ for each patch ω_i, but ensures significantly large overlaps $\omega_i \cap \omega_j$ to allow for the use of a cheaper quadrature scheme for each nonzero entry of the stiffness matrix.

With Algorithm 1 for the cover construction the control over the neighborhoods N_i is somewhat limited. Hence, it is very difficult to limit the density of the stiffness matrix. Even more problematic though is the fact that Algorithm 1 needs an additional input[3] Q besides the set of points P and the domain $\Omega \subset \mathbb{R}^d$ to ensure that the complete domain is indeed covered by C_Ω. This of course makes Algorithm 1 significantly less useful, especially in time-dependent settings.

In the following we propose a new algorithm which employs a decomposition approach for the domain Ω to assign sets $\omega_i \subset \mathbb{R}^d$ to the points $x_i \in P$ in such a way that they cover the domain $\Omega \subset \bigcup \omega_i$. This hierarchical algorithm does not need an additional input Q.

Algorithm 2 (Hierarchical Cover Construction).

1. Given: the domain $\Omega \subset \mathbb{R}^d$, a bounding box $R_\Omega = \bigotimes_{i=1}^d [l_\Omega^i, u_\Omega^i] \supset \overline{\Omega}$, the initial point set $P = \{x_j \in \mathbb{R}^d \,|\, x_j \in \overline{\Omega}\}$ and a parameter $k \in \mathbb{N}$.
2. Build a d-binary tree (quadtree, octree) over R_Ω, such that per leaf L at most one $x_i \in P$ lies within the associated cell $C_L := \bigotimes_{i=1}^d [l_L^i, u_L^i]$, and the difference of the levels of two adjacent cells is at most k, see Figure 3.1.
3. For all cells $C_L = \bigotimes_{i=1}^d [l_L^i, u_L^i]$ with $C_L \cap \Omega \neq \emptyset$:
   (a) If there is an $x_j \in P$ with $x_j \in C_L$, set $x_L = x_j$. Else, set $x_L =$ any element of C_L, e.g. the center $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ of the cell C_L.
   (b) Set $P = P \cup \{x_L\}$.
   (c) Set $\omega_L = R_L = \bigotimes_{i=1}^d [x_L^i - h_L^i, x_L^i + h_L^i] \supset C_L$, where $h_L^i = \frac{\alpha}{2} \max\{u_L^i - x_L^i, x_L^i - l_L^i\}$ with $\alpha > 1$.

With Algorithm 2 we not only ensure the covering property $\Omega \subset \bigcup \omega_i$ without additional input data, but we also control the neighborhoods N_i, i.e. the nonzeros of the stiffness matrix, and ensure the smoothness of the functions φ_i. The neighborhoods N_i of the cover patches ω_i constructed by Algorithm 2 are small yet the amount of overlap of any two neighboring patches is of significant size. Certainly, these features do come at a prize we have to pay: the algorithm automatically introduces additional points x_{N+k} into the set P, see Figures 3.1 and 3.2.[4] This increases the number of unknowns, i.e. the number of rows of the stiffness matrix, and seemingly the overall computational cost. But as it turns out, the number of nonzeros of a stiffness matrix based on our algorithm is comparable to the number of nonzeros of a stiffness matrix based on Algorithm 1 for uniformly distributed points P (see Table 6.1 where

[3] The selection of the set Q within Algorithm 1 is quite involved to ensure the sparsity of the stiffness matrix and to cover the complete domain at the same time if the points $x_i \in P$ are non-uniformly distributed.

[4] The additional points are necessary to ensure the shape-regularity of the tree cells and patches. A similar tree-based algorithm for the construction of shape-regular triangulations with an almost-minimal number of vertices was proposed in [5]. In analogy, the presented algorithm may have a similar almost-optimal property: If m is the minimal number of shape-regular d-rectangles required to cover the given point set in such way that all d-rectangles contain at most a single point, then the cover C_Ω constructed by the presented algorithm is of size $\mathrm{card}(C_\Omega) = O(m)$.

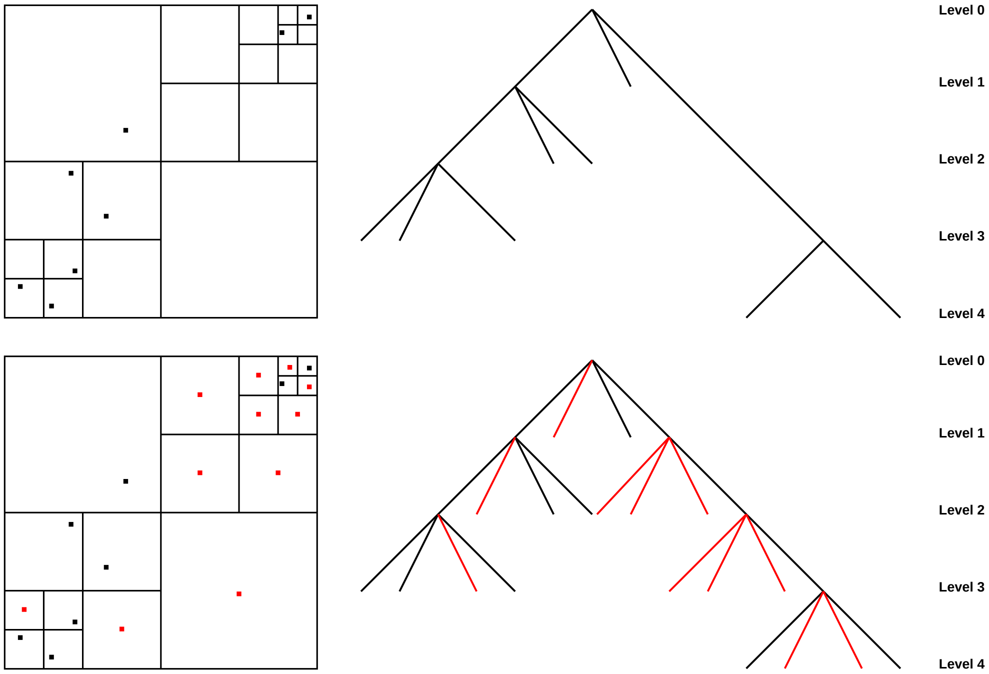

FIGURE 3.1. *Hierarchical cover construction with Algorithm 2 in two dimensions. The initial cell decomposition induced by P (upper left) and its corresponding tree representation (upper right) after step 2 of Algorithm 2. The final cell decomposition (lower left) and its tree representation after the completion of Algorithm 2.*

the initial point set P for the cover construction is a Halton[5] point set). And it is significantly less for highly irregular point sets P (see Table 6.2). Furthermore, the proposed algorithm enables the user to control the amounts of overlap $\omega_i \cap \omega_j$ completely by the choice of the parameter k in step 2 (which corresponds to the local imbalance of the tree), the choice of $x_L \in C_L$ in step 3a and the choice of α in step 3c. Hence, this construction leads to smoother partition of unity functions φ_i and allows for the use of cheaper quadrature schemes (compared with Algorithm 1) during the assembly of the stiffness matrix, although the functions φ_i are still more complex than FE shape functions (see Figure 3.3). In summary, the proposed algorithm is applicable to general domains Ω and any initial distribution of points P without an additional input Q, yet also reduces the computational costs during the assembly of the stiffness matrix and right hand side vector.

Note that we capture the domain Ω by the cells C_L and subsequently by the $\omega_L = R_L$ only. This though does *not* limit the domain or boundary resolution of our method. At this stage in the construction process, we are only interested in generating a cover C_Ω of the domain Ω. Only during the integration of the stiffness matrix and right hand side vector entries we need to restrict the evaluation of the associated shape functions $\varphi_L \psi_L^k$ to the computational domain Ω, i.e. to the integration domain $\omega_L \cap \Omega$.

[5]Halton–sequences are pseudo Monte Carlo sequences, which are used in sampling and numerical integration. Consider $n \in \mathbb{N}_0$ given as $\sum_j n_j p^j = n$ for some prime p. We can define the transformation H_p from $\mathbb{N}_0$ to $[0,1]$ with $n \mapsto H_p(n) = \sum_j n_j p^{-j-1}$. Then, the (p,q) Halton–sequence with N points is defined as $\text{Halton}_0^N(q,p) := \{(H_p(n), H_q(n)) \,|\, n = 0, \ldots, N\}$.

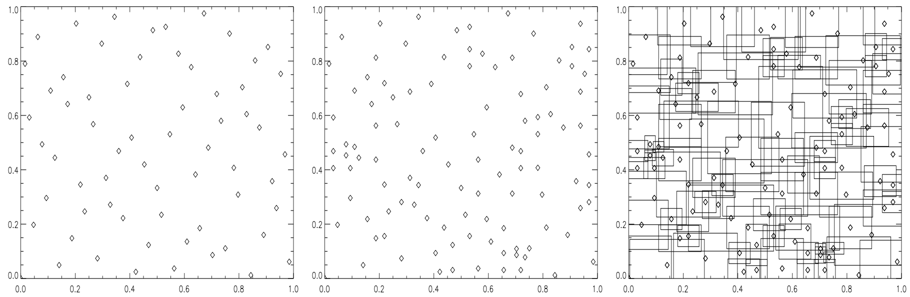

FIGURE 3.2. $P = Halton_0^{63}(2,3)$ *point set in* $R_\Omega = \Omega = [0,1]^2$ *(left),* P *with* $\mathrm{card}(P) = 106$ *(center) after Algorithm 2 with* $k = \infty$ *and* $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ *for generated points* x_L*, and the generated cover* C_Ω *(right) with* $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$.

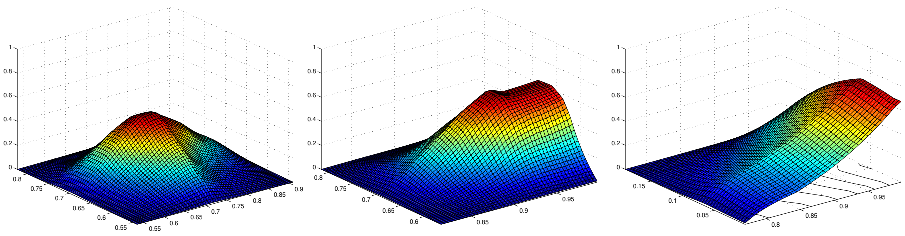

FIGURE 3.3. *The partition of unity function* φ_i *on* $\Omega \cap \omega_i$ *generated by Algorithm 2 with the input data from 3.2 for an interior point (left), a boundary point (center) and a corner point (right) using linear B-splines.*

We postpone the issue of domain and boundary resolution to §5. Note further that due to steps 3a and 3c we generally produce d-rectangular cover patches ω_i independent of the shape of the bounding box R_Ω, see Figure 3.2. Note also that the construction allows for the fast evaluation of a single partition of unity function φ_i, see (2.2), due to the efficient neighbor search in the hierarchical tree data structure. Note finally that the introduction of a hierarchical cover induces a hierarchy for the associated function space[6] which we may exploit in the design of fast multilevel solvers for the linear equations arising from a PUM discretization. This issue though is subject of future research.

4. Galerkin method with the PUM space. We want to solve elliptic boundary value problems of the type

$$\begin{array}{rcll} Lu & = & f & \text{in } \Omega \subset \mathbb{R}^d\,, \\ Bu & = & g & \text{on } \partial\Omega\,, \end{array} \tag{4.1}$$

where L is a symmetric partial differential operator of second order and B expresses suitable boundary conditions. The implementation of Neumann boundary conditions with our partition of unity method is straightforward and similar to their treatment within the FEM. The realization of essential boundary conditions with meshfree methods is more involved than with a finite element method due to the non-interpolatory

[6]For covers from Algorithm 1 the introduction of such a hierarchy would at least be artificial. Since in Algorithm 1 the selection of the supports ω_i is independent of a hierarchical ordering on the points, this hierarchy on the points would not lead to a hierarchy for the supports of the shape functions in a natural way.

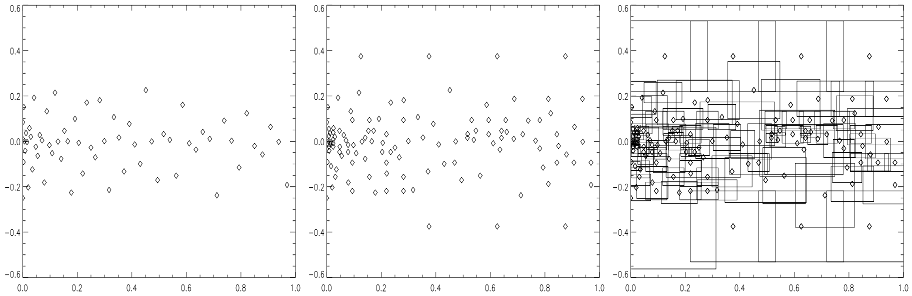

FIGURE 3.4. *P is a Halton$_0^{63}(2,3)$ point set in $R_\Omega = \Omega = [0,1] \times [-0.5, 0.5]$ graded by $(x,y) \mapsto (x^2, \pm y^2)$ (left), P with* $\mathrm{card}(P) = 121$ *(center) after Algorithm 2 with $k = \infty$ and $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L, and the generated cover C_Ω (right) with $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$.*

character of the meshfree shape functions. There are several different approaches to the implementation of essential boundary conditions with meshfree approximations, see [13, 16, 27]. Throughout this paper we use Lagrangian multipliers to enforce essential boundary conditions, see [13, 27].

In the following let $a(\cdot,\cdot)$ be the continuous and elliptic bilinear form induced by L on $H^1(\Omega)$. We discretize the partial differential equation using Galerkin's method. Then, we have to compute the stiffness matrix

$$A = (a_{ij}), \text{ with } a_{ij} = a(\varphi_j\psi_j^l, \varphi_i\psi_i^k) \in \mathbb{R}^{\dim(V_j^{p_j})\times\dim(V_i^{p_i})},$$

and the right hand side vector

$$\hat{f} = (f_i), \text{ with } f_i = \langle f, \varphi_i\psi_i^k\rangle_{L^2} = \int_\Omega f\varphi_i\psi_i^k \in \mathbb{R}^{\dim(V_i^{p_i})}.$$

If we restrict ourselves for reasons of simplicity to the case $L = -\Delta$ we have to compute the integrals $\int_\Omega \varphi_i\psi_i^k f$ for the right hand side and the integrals $\int_\Omega \nabla(\varphi_i\psi_i^k)\nabla(\varphi_j\psi_j^l)$ for the stiffness matrix. Recall that φ_i is defined by (2.2), i.e.

$$\varphi_i(x) = \frac{W_i(x)}{\sum W_k(x)}.$$

Now we carry out the differentiation in $\int_\Omega \nabla(\varphi_i\psi_i^k)\nabla(\varphi_j\psi_j^l)$. With the notation $\mathcal{S} := \sum W_k$, $\mathcal{T} := \sum \nabla W_k$ and $\mathcal{G}_i := \nabla W_i\mathcal{S} - W_i\mathcal{T}$ we end up with the integrals

$$\begin{aligned} a(\varphi_j\psi_j^l, \varphi_i\psi_i^k) \quad = \quad & \int_\Omega \mathcal{S}^{-4}\mathcal{G}_i\psi_i^k\mathcal{G}_j\psi_j^l \quad + \quad \int_\Omega \mathcal{S}^{-2}W_i\nabla\psi_i^k W_j\nabla\psi_j^l \quad + \\ & \int_\Omega \mathcal{S}^{-3}\mathcal{G}_i\psi_i^k W_j\nabla\psi_j^l \quad + \quad \int_\Omega \mathcal{S}^{-3}W_i\nabla\psi_i^k\mathcal{G}_j\psi_j^l \end{aligned} \tag{4.2}$$

for the stiffness matrix and the integrals

$$\langle f, \varphi_i\psi_i^k\rangle_{L^2} = \int_\Omega \mathcal{S}^{-1}W_i\psi_i^k f \tag{4.3}$$

for the right hand side. Due to the facts that we use piecewise polynomial weights W_i for the Shepard construction (2.2) and that the support patches ω_i overlap each

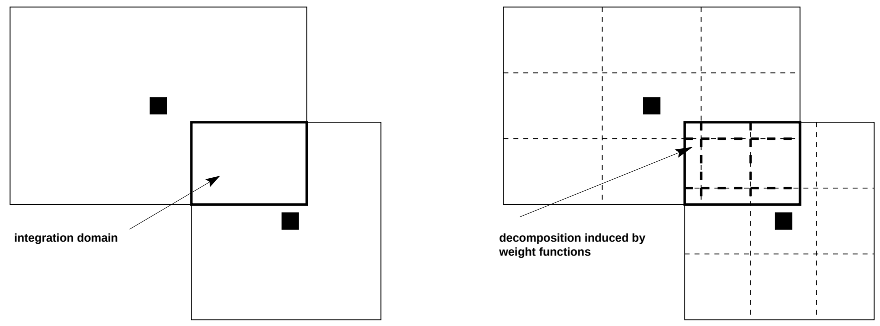

FIGURE 5.1. *Integration domain $\Omega_{ij} = \omega_i \cap \omega_j$ (left). The decomposition $E_{\omega_{ij}}$ of the integration domain ω_{ij} via the subdivision induced by the weight functions W_i and W_j (right). Here, the weights are tensor products of quadratic B-splines.*

other, the functions $\mathcal{T}$ and $\mathcal{G}_i$ may have quite a number of jumps of significant size. Therefore, the integrals (4.2) and (4.3) should not be computed by a simple quadrature scheme which does not respect these discontinuities and the algebraic structure of the shape functions. Instead, we need to decompose the integration domain in such a way that the piecewise character of the integrands is resolved.

5. Decomposition Sparse Grid Quadrature Scheme. Let us assume that the PU is given by an h-mesh construction like we have in the GFEM. Then, we know how to resolve the piecewise character of the integrands: we subdivide the integration domains with the help of the geometric elements of the h-mesh. However, with our general PUM we do not have a mesh or geometric elements. But we have support patches ω_i and weight functions W_i which define the partition of unity functions φ_i by (2.2). From this information only, we have to find an appropriate subdivision of the support patches ω_i and subsequently the integration domains. Furthermore, we have to cope with rational integrands on the cells of such a subdivision in our general PUM. The foundation for the proposed quadrature scheme is a decomposition approach which was first presented in [27]. Here, we give a short review over the construction principles for the decomposition $D_{\omega_{ij}} := \{D^n_{\omega_{ij}}\}$ of the integration domains $\omega_{ij} := \omega_i \cap \omega_j \cap \Omega$.

The integration domains ω_{ij} may be decomposed into disjoint cells $D^n_{\omega_{ij}}$ by exploiting the tensor product structure of the cover patches ω_i and the weight functions W_i used during the construction (2.2) of the partition of unity $\{\varphi_i\}$. This decomposition of an integration domain ω_{ij} can efficiently be computed by splitting ω_{ij} via its caps $\omega_{ij} \cap \omega_k$ with the neighboring cover patches $\omega_k \in N_{ij} := N_i \cap N_j$ using a second tree data structure.

Consider the integration domain $\omega_{ij} = \omega_i \cap \omega_j \subset \Omega$. The intersection ω_{ij} of two cover patches ω_i, ω_j which are tensor products of intervals is also a tensor product of intervals, see Figure 5.1 (left). Moreover, the employed weight functions W_k are tensor products of normed B-splines of order l, i.e. they are piecewise polynomials of degree l. Therefore, the weight function W_k induces a subdivision of the respective cover patch ω_k into $(l+1)^d$ sub-patches $\{\omega_k^q\}$ on which $W_k\big|_{\omega_k^q}$ is polynomial. Furthermore, these sub-patches $\{\omega_k^q\}$ are also tensor products of intervals. With the help of these sub-patches $\{\omega_i^q\}$, $\{\omega_j^q\}$ we can define a first decomposition $E_{\omega_{ij}} = \{E^n_{\omega_{ij}}\}$ of ω_{ij}, see Figure 5.1 (right). On the cells $E^n_{\omega_{ij}}$ of this decomposition we have that $W_i\big|_{E^n_{\omega_{ij}}}$

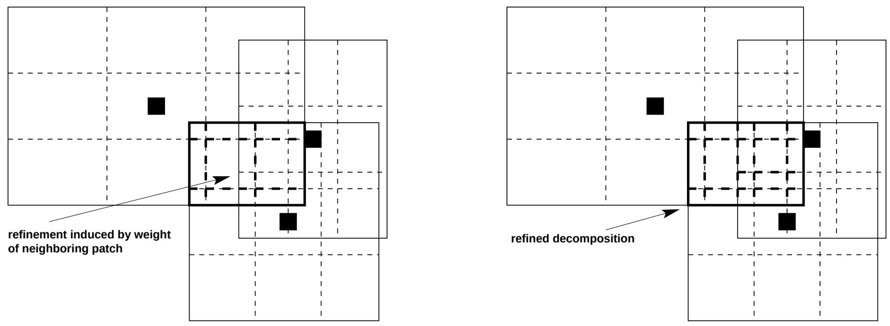

FIGURE 5.2. *Refinement of the decomposition $E_{\omega_{ij}}$ of the integration domain ω_{ij} via the subdivision induced by the weight function W_k (tensor product of quadratic B-splines) of one neighboring point x_k (left). The resulting decomposition $D_{\omega_{ij}}$ after the refinement step for the neighboring weight function W_k (right).*

and $W_j\big|_{E^n_{\omega_{ij}}}$ are polynomials of degree l, but all other $W_k\big|_{E^n_{\omega_{ij}}}$ may still be piecewise polynomial only. Therefore, we further refine the decomposition $E_{\omega_{ij}}$ by subdividing the cells $E^n_{\omega_{ij}}$ with the help of the $\{\omega^q_k\}$ sub-patches for all $\omega_k \in N_{ij}$, see Figure 5.2. The resulting decomposition $D_{\omega_{ij}} = \{D^n_{\omega_{ij}}\}$ consists of d-rectangular cells $D^n_{\omega_{ij}}$ on which all weight functions $W_k|_{D^n_{\omega_{ij}}}$ are polynomials of degree l. The number of cells $\mathrm{card}(D_{\omega_{ij}})$ of the decomposition $D_{\omega_{ij}} = \{D^n_{\omega_{ij}}\}$ depends on the polynomial degree l of the weight functions W_k used during the Shepard construction (2.2) for the partition of unity, the number of neighbors $\mathrm{card}(N_i)$ and their geometric location.

Since all weights W_k are polynomial on the cells $D^n_{\omega_{ij}}$, the functions $\mathcal{T}$ and $\mathcal{G}_i$ are non-singular rational functions on $D^n_{\omega_{ij}}$. Hence, any standard quadrature rule for smooth functions is applicable for the numerical integration of the weak form on the cells $D^n_{\omega_{ij}}$ (if we assume that the local basis functions ψ^k_i and ψ^l_j are smooth on ω_{ij}). Independent of the local quadrature rule used on $D^n_{\omega_{ij}}$ we can utilize the product structure of the shape functions $\varphi_i\psi^k_i$ to reduce the computational costs of an evaluation of the weak form at a quadrature point. Here, we simultaneously evaluate the complete block $a_{ij} = a(\varphi_j\psi^l_j, \varphi_i\psi^k_i) \in \mathbb{R}^{\dim(V^{p_j}_j)\times\dim(V^{p_i}_i)}$ of the stiffness matrix rather than evaluating every single $(a_{ij})_{kl} = a(\varphi_j\psi^l_j, \varphi_i\psi^k_i) \in \mathbb{R}$ for fixed k and l. Besides the reduction in the number of evaluations of φ_i and φ_j, this also allows for a hierarchical evaluation of the local basis functions ψ^k_i and ψ^l_j (which is available for the chosen Legendre polynomials) which reduces the computational costs of an evaluation of the weak form significantly (esp. for higher order approximations).

For the selection of a quadrature rule on the cells $D^n_{\omega_{ij}}$ we now can assume the smoothness of the integrands. But still the quadrature rule has to be applicable to general situations (general covers, weights and local basis functions ψ^k_i, etc.). Hence, we have to find a fast converging, cheap quadrature rule on $D^n_{\omega_{ij}}$ which allows for a reliable dynamic stopping criterion for a wide range of integrands.

So-called *sparse grid quadrature* [12] rules are multi-dimensional interpolatory rules with a substantially smaller number of integration nodes compared with a tensor product rule. They are defined as special products of one-dimensional interpolatory quadrature rules. Although the number of evaluations of the integrand is significantly less for a sparse grid quadrature rule, the order of the achieved error is comparable to that of a full tensor product rule. Here, we only state the fundamental construction principles and error bounds, see [12] and the references cited therein for further details.

Consider a sequence of nested one-dimensional quadrature rules for univariate functions $\{Q_l^1 \,|\, Q_l^1 f := \sum_{i=1}^{n_l^1} w_{li} f(x_{li}),\ n_l^1 = O(2^l),\ f : \mathbb{R} \to \mathbb{R}\}$ with weights w_{li}, nodes x_{li} and error bound $|Q_l^1 f - \int f| = O(2^{-lr})$ where f is assumed to be r-times continuously differentiable. This bound holds for example for the Clenshaw–Curtis and Gauß–Patterson rules. With the help of the difference quadrature rules Δ_k^1

$$\Delta_k^1 f := (Q_k^1 - Q_{k-1}^1) f \quad \text{with} \quad Q_0^1 f := 0.$$

we can define the sparse grid quadrature rule Q_l^d on level l in d-dimensions as

$$Q_l^d f := \sum_{\sum_{i=1}^d k_i \le l+d-1} (\Delta_{k_1}^1 \otimes \ldots \otimes \Delta_{k_d}^1) f$$

with $f : \mathbb{R}^d \to \mathbb{R}$, $l \in \mathbb{N}$ and $k \in \mathbb{N}^d$. Due to the restriction $\sum_{i=1}^d k_i \le l + d - 1$ in the summation, the number n_l^d of quadrature points x_i^d of the resulting sparse grid quadrature rule Q_l^d is

$$n_l^d = O(2^l l^{d-1})$$

only. Hence, the number of function evaluations for a sparse grid quadrature rule is dramatically less (see Figure 5.3) than for a full tensor product rule where the integrand has to be evaluated at $O(2^{ld})$ quadrature points. This reduction of the computational costs though does not compromise the approximation quality significantly for smooth functions. When f is assumed to be r-times continuously differentiable the estimate

$$|Q_l^d f - \int f| = O(2^{-lr} l^{(d-1)(r+1)})$$

holds.

In summary, sparse grid quadrature rules are not only cheaper to evaluate (esp. in higher dimensions) compared with tensor product rules, but rather their overall efficiency with respect to accuracy is significantly better. In [12] the fast convergence of sparse grid quadrature rules based on Gauß–Patterson rules (see Figure 5.3) is shown for a wide variety of function classes. In fact, sparse grid quadrature rules based on Gauß–Patterson rules converge exponentially for smooth integrands. Since, the integrands we are interested in are smooth on the cells $D_{\omega_{ij}}^n$ of the constructed decomposition $D_{\omega_{ij}}$ we use Gauß–Patterson sparse grid rules for the numerical integration of the stiffness matrix entries.

To ensure a reliable accuracy of our quadrature scheme, we use a simple three level dynamic stopping criterion [24]. The quadrature on a cell $D_{\omega_{ij}}^n$ is stopped if

$$|Q_{l-1}^d f - Q_{l-2}^d f| \le c_1 \epsilon_a + c_2 \epsilon_r |Q_{l-1}^d f| \quad \text{and} \quad |Q_l^d f - Q_{l-1}^d f| \le \epsilon_a + c_3 \epsilon_r |Q_l^d f|$$

hold for the integrand $(a_{ij})_{kl}$ of minimal order $k = l = 0$ as well as for the integrand of maximal order $k = \dim(V_i^{p_i}) - 1$, $l = \dim(V_j^{p_j}) - 1$. Here, c_1, c_2 and c_3 are non-negative constants and ϵ_a and ϵ_r are user supplied absolute and relative tolerances. These tolerances which determine the accuracy of the integration though have to be chosen with respect to the approximation space. Here, the diameters $\operatorname{diam}(\omega_i)$, $\operatorname{diam}(\omega_j)$ of the cover patches ω_i, ω_j, the number of integration cells $\operatorname{card}(D_{\omega_{ij}})$, their respective diameters $\operatorname{diam}(D_{\omega_{ij}}^n)$ and the local approximation orders p_i, p_j have to be considered. An automatic selection of the tolerances ϵ_a and ϵ_r which minimizes the

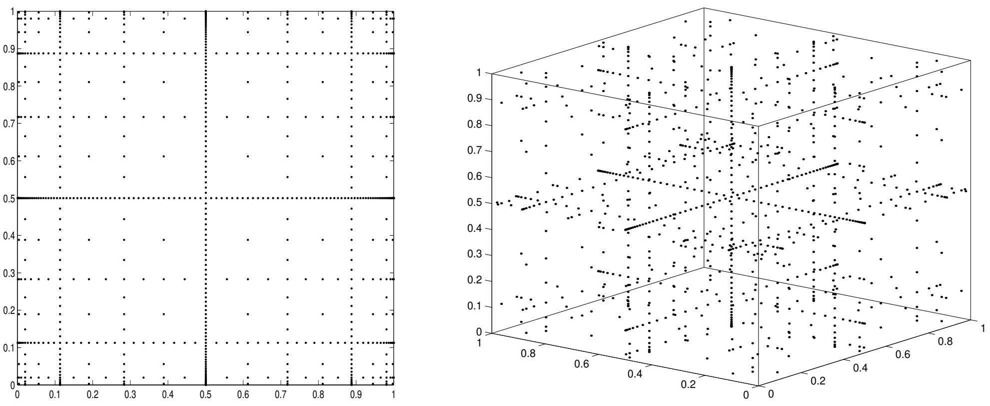

FIGURE 5.3. *Quadrature nodes of sparse grid Gauß–Patterson rules, level $l = 6$ with 769 nodes in two dimension (left) and level $l = 5$ with 1023 nodes in three dimension (right).*

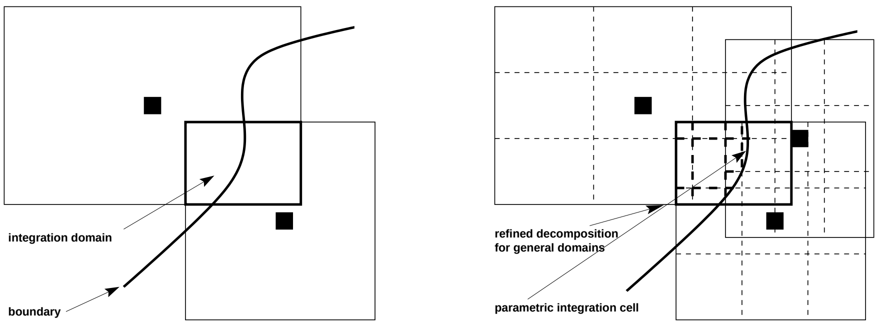

FIGURE 5.4. *Integration domain $\Omega_{ij} = \omega_i \cap \omega_j \cap \Omega$ for general domains Ω (left). The decomposition $D_{\omega_{ij}}$ of a general integration domain ω_{ij} via the subdivision induced by the weight functions W_i, W_j and a neighboring patch (right), compare Figures 5.1 and 5.2. Here, the weights are tensor products of quadratic B-splines.*

computational work but at the same time does not compromise the accuracy of the discretization [29, Chapter 4] is subject of future research.

Note that the decomposition approach given above is not restricted to domains Ω which are unions of d-rectangles but rather applicable to general domains. For integration domains $\omega_i \cap \omega_j \cap \Omega \neq \omega_i \cap \omega_j$ we apply the construction to the fictitious integration domain $\tilde{\omega}_{ij} := \omega_i \cap \omega_j$. From this decomposition $D_{\tilde{\omega}_{ij}} = \{D^n_{\tilde{\omega}_{ij}}\}$ we then select the subset $\hat{D}_{\omega_{ij}} := \{\hat{D}^n_{\omega_{ij}} \in D_{\tilde{\omega}_{ij}} \mid \hat{D}^n_{\omega_{ij}} \cap \Omega \neq \emptyset\}$ of all cells which overlap the actual integration domain $\omega_{ij} = \tilde{\omega}_{ij} \cap \Omega$. This intermediate cover $\hat{D}_{\omega_{ij}}$ consists of d-rectangular cells $\hat{D}^n_{\omega_{ij}} \cap \Omega \neq \emptyset$ which cover the integration domain ω_{ij}, see Figure 5.4. Now, the remaining task is to resolve (with the necessary accuracy) the boundary $\partial\Omega$ of the domain which runs through some of the cells $\hat{D}^n_{\omega_{ij}}$. We assume that a representation for the boundary $\partial\Omega$ is given as part of the computational domain Ω. That is, we assume the domain Ω and its boundary $\partial\Omega$ are given as a collection of mappings $\mathcal{R}^m_\Omega : [-1,1]^d \to \overline{\Omega} \subset \mathbb{R}^d$ from a reference cell into the physical space $\overline{\Omega} \subset \mathbb{R}^d$ with $\bigcup \mathcal{R}^m([-1,1]^d) = \overline{\Omega}$. These mappings $\mathcal{R}^m_\Omega$ may be coming from a CAD system, i.e. the mappings themselves are only an approximation to the true domain Ω, or may be coming from a given analytical representation of the domain Ω.

With the help of these mappings $\mathcal{R}_\Omega^m$ we can compute the parametric integration cell $D_{\omega_{ij}}^n := \hat{D}_{\omega_{ij}}^n \cap \Omega$ which lies within the integration domain ω_{ij}, see Figure 5.4. The final decomposition $D_{\omega_{ij}} = \{D_{\omega_{ij}}^n \,|\, D_{\omega_{ij}}^n := \hat{D}_{\omega_{ij}}^n \cap \Omega\}$ for general domains Ω therefore consists of d-rectangular cells $D_{\omega_{ij}}^n = \hat{D}_{\omega_{ij}}^n$ if $\hat{D}_{\omega_{ij}}^n \subset \Omega$ and parametric cells $D_{\omega_{ij}}^n$ if the corresponding d-rectangular cell $\hat{D}_{\omega_{ij}}^n$ overlaps the boundary $\partial\Omega$.

Note that the mappings $\mathcal{R}_\Omega^m$ do *not* influence the shape functions of our partition of unity method. While in a finite element method the supports of the shape function have to align with the boundary $\partial\Omega$ of the domain, in our PUM the supports of the shape functions only have to cover the domain Ω and its boundary $\partial\Omega$. The mappings of the domain representation $\mathcal{R}_\Omega^m$ are merely used to implement integration domains ω_{ij} with complicated geometry.

Overall the numerical quadrature of the stiffness matrix entries is completed in three steps:

1. First we compute the decomposition $D_{\tilde{\omega}_{ij}}$ for the domain $\tilde{\omega}_{ij} = \omega_i \cap \omega_j$, see Figures 5.1 and 5.2. With the help of the domain representation mappings $\mathcal{R}_\Omega^m$ we then select the integration cells $\hat{D}_{\omega_{ij}}^n = D_{\tilde{\omega}_{ij}}^n \cap \Omega \neq \emptyset$ which overlap the computational domain Ω. Furthermore, we use the mappings $\mathcal{R}_\Omega^m$ to construct the final decomposition $D_{\omega_{ij}} = \{D_{\omega_{ij}}^n \,|\, D_{\omega_{ij}}^n := \hat{D}_{\omega_{ij}}^n \cap \Omega\}$, see Figure 5.4.
2. For each of these integration cells $D_{\omega_{ij}}^n$ we then compute the Jacobian $J_{T_{\omega_{ij}}^n}$ of the mapping $T_{\omega_{ij}}^n : [-1,1]^d \to D_{\omega_{ij}}^n$ from the reference integration domain $[-1,1]^d$ onto the integration cell $D_{\omega_{ij}}^n$ in the configuration space.
3. Finally, we evaluate the entries of the stiffness matrix and right hand side vector by computing the integrals (4.2) and (4.3) on the integration cells $D_{\omega_{ij}}^n$. That is, we compute them on the reference integration domain $[-1,1]^d$ using the transformations $T_{\omega_{ij}}^n$ and the respective Jacobians $J_{T_{\omega_{ij}}^n}$

$$\int_{D_{\omega_{ij}}^n} \mathcal{F} = \int_{[-1,1]^d} \mathcal{F} \circ T_{\omega_{ij}}^n |J_{T_{\omega_{ij}}^n}|.$$

   The Jacobian $J_{T_{\omega_{ij}}^n}$ for a simple d-rectangular integration cell is of course constant and this transformation does not increase the costs of the numerical integration. But for a parametric cell the transformation $T_{\omega_{ij}}^n$ involves the mappings $\mathcal{R}_\Omega^m$ of the domain representation. Therefore, the Jacobian may well be space-dependent and has to be evaluated at every integration node of the quadrature rule.[7]

   Again, the error during the numerical quadrature has to be controlled by the selection of ϵ_a and ϵ_r to ensure that the order of approximation is not compromised by the integration error.

The overall computational costs of the proposed quadrature scheme depends on the number of cells $\operatorname{card}(D_{\omega_{ij}})$ of the decomposition, i.e. on the order l of the weight functions, the geometric location of the neighbors $\omega_j \in N_i$, their number $\operatorname{card}(N_i)$ and the local quadrature rule used on the cells $D_{\omega_{ij}}^n$. Due to the use of the sparse

[7]Note that in general the transformations $T_{\omega_{ij}}^n$ may lead to (locally) non-smooth integrands where an isotrop sparse grid quadrature scheme may not be well-suited on all integration cells $D_{\omega_{ij}}^n$. Here, a further decomposition of the integration cells $D_{\omega_{ij}}^n$ where the integrands are non-smooth or even adaptive quadrature rules (sparse grid or other) should be employed on such integration cells $D_{\omega_{ij}}^n$.

grid rules on the cells, the computational costs is significantly reduced compared with tensor product rules.

6. Hierarchical Regular Cover Construction. A further reduction of the computational effort necessary during the assembly of the stiffness matrix can only be achieved by reducing the number of cells of the decomposition[8] $D_{\omega_{ij}}$. This can be attained by the alignment of the cover patches ω_k and their subdivisions $\{\omega_k^q\}$.

Taking into account that we limit ourselves to the use of tensor product B-splines as weight functions for Shepard's construction (2.2) we can align the cover patches to simplify the algebraic structure of the resulting partition of unity functions φ_i. Here, we eliminate some of the flexibility in step 3 of Algorithm 2 for the choices of x_L and α. This though does not lead to a significantly larger number of neighbors. Hence, the number of nonzeros of the stiffness matrix stays (almost) constant, see Tables 6.1 and 6.2. However, the number of integration cells $\mathrm{card}(D_{\omega_{ij}})$ is substantially reduced by this modification, see Tables 6.4 and 6.5.

Recall that we split the integration domain ω_{ij} into several cells by its caps $\omega_{ij}\cap\omega_k^q$ with the cells ω_k^q of the subdivision induced by the weight W_k on $\omega_k \in N_{ij}$ during the construction of the decomposition $D_{\omega_{ij}}$. Hence, we align these caps $\omega_{ij} \cap \omega_k^q$, which subsequently induce at least one integration cell $D^n_{\omega_{ij}}$, if we align the neighboring cover patches ω_k with respect to their subdivisions $\{\omega_k^q\}$. Therefore, many of the $\omega_{ij} \cap \omega_k^q$ will lead to the same integration cell $D^n_{\omega_{ij}}$ and the overall number of integration cells $\mathrm{card}(D_{\omega_{ij}})$ will be reduced significantly, see Tables 6.4 and 6.5. This alignment of the cover patches ω_k and their subdivisions $\{\omega_k^q\}$ is achieved by the following algorithm.

Algorithm 3 (Hierarchical Regular Cover Construction).

1. Given: the domain $\Omega \subset \mathbb{R}^d$, a bounding box $R_\Omega = \bigotimes_{i=1}^d [l^i_\Omega, u^i_\Omega] \supset \overline{\Omega}$, the initial point set $P = \{x_j \in \mathbb{R}^d \,|\, x_j \in \overline{\Omega}\}$ and a parameter $k \in \mathbb{N}$.
2. Build a d-binary tree (quadtree, octree) over R_Ω, such that per leaf L at most one $x_i \in P$ lies within the associated cell $C_L := \bigotimes_{i=1}^d [l^i_L, u^i_L]$, and the difference of the levels of two adjacent cells is at most k.
3. For all cells $C_L = \bigotimes_{i=1}^d [l^i_L, u^i_L]$ with $C_L \cap \Omega \neq \emptyset$:
   (a) Set $x^i_L = l^i_L + \frac{1}{2}(u^i_L - l^i_L)$.
   (b) If there is an $x_j \in P$ with $x_j \in C_L$, set $P = P \setminus \{x_j\}$.
   (c) Set $P = P \cup \{x_L\}$.
   (d) Set $\omega_L = R_L = \bigotimes_{i=1}^d [x^i_L - h^i_L, x^i_L + h^i_L] \supset C_L$, where $h^i_L = \frac{\alpha_l}{2}(u^i_L - l^i_L)$.

Here, the parameter α_l in the computation of the support size in step 3d is only dependent on the weight function used in (2.2), i.e. the order l of the B-spline. By construction the one-dimensional distances from a point $x_L \in P$ to its direct neighboring point $x_j \in P$, i.e. the point x_j corresponding to the sibling tree cell $C_j = \bigotimes_{i=1}^d [l^i_j, u^i_j]$, are $|x^i_L - x^i_j| = u^i_L - l^i_L = u^i_j - l^i_j$ where $C_L = \bigotimes_{i=1}^d [l^i_L, u^i_L]$ is the cell associated with x_L. Hence, if we choose α_l in such a way that condition (6.1) is fulfilled, we not only align the patch ω_L with its direct neighboring patch ω_j, but rather also their corresponding subdivisions $\{\omega_L^q\}$ and $\{\omega_j^q\}$ induced by the weight

[8]The decomposition itself though is minimal in the sense that it has a minimal number $\mathrm{card}(D_{\omega_{ij}})$ of integration cells necessary to resolve the piecewise character of the partition of unity functions. In our construction (2.2) of the partition of unity we have to allow for higher orders l of the B-spline weights to be able to construct global solutions u_{PU} with higher order regularity, i.e. $u_{\mathrm{PU}} \in \mathcal{C}^{l-1}$. Therefore, the remaining influences on the computational effort involved with the numerical integration of the stiffness matrix entries are the geometric neighboring relations of our cover patches ω_i.

N	$A_{1,R}$	$A_{1,S}$	card(P)	A_2	A_3
1024	18840	21530	1729	20023	21751
4096	79102	91902	6364	73908	78588
16384	325730	377388	27673	326919	360053
65536	1267300	1431210	101314	1245108	1259134

TABLE 6.1

The number $\sum \operatorname{card}(N_i)$ of neighbors for covers generated by Algorithm 1 with $\alpha = 1.5$ using rectangular patches ($A_{1,R}$), and using square patches ($A_{1,S}$). The number of cover patches card(P) *after the cover construction and the number of neighbors for Algorithm 2 with $k = \infty$, $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L and $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$ (A_2, see also Figure 3.2), and for Algorithm 3 with $k = \infty$ and $\alpha_l = 2$ (A_3, see also Figure 6.1). The initial point set P for all algorithms was Halton$_0^{N-1}(2,3)$.*

N	$A_{1,R}$	$A_{1,S}$	card(P)	A_2	A_3
1024	33878	39114	1897	23193	26203
4096	127950	149460	7501	92235	104985
16384	507010	591794	30040	369754	416292

TABLE 6.2

The number $\sum \operatorname{card}(N_i)$ of neighbors for covers generated by Algorithm 1 with $\alpha = 1.5$ using rectangular patches ($A_{1,R}$), and using square patches ($A_{1,S}$). The number of cover patches card(P) *after the cover construction and the number of neighbors for Algorithm 2 with $k = \infty$, $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L and $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$ (A_2, see also Figure 3.4), and for Algorithm 3 with $k = \infty$ and $\alpha_l = 2$ (A_3, see also Figure 6.3). The initial point set P for all algorithms was a graded Halton$_0^{N-1}(2,3)$.*

functions W_L and W_j. Moreover, this alignment of the patches does not increase the number of neighbors $\operatorname{card}(N_L)$. With the notation $h_l^i := \frac{\alpha_l}{l+1}(u_L^i - l_L^i)$ for the B-spline interval size, the condition reads

$$x_L^i + \frac{l+1}{2}h_l^i = x_j^i - \left(\frac{l+1}{2} - m\right)h_l^i = x_L^i + (u_L^i - l_L^i) - \left(\frac{l+1}{2} - m\right)h_l^i \qquad (6.1)$$

for the i-th coordinate with $i = 1, \ldots, d$. Here, the parameter $m \in \mathbb{N}$ indicates the amount of overlap $\omega_L \cap \omega_j \sim \bigotimes_{i=1}^d m h_l^i$ for the neighbor $\omega_j \in N_L$. Any integer m with $1 \le m \le \frac{l+1}{2}$ leads to minimal neighborhoods N_L and minimal decompositions $D_{\omega_{Lj}}$, i.e. the number $\sum_L \operatorname{card}(N_L)$ of nonzero entries of the stiffness matrix and the number $\sum_{L,j}\operatorname{card}(D_{\omega_{Lj}})$ of integration cells are (almost) constant. Therefore, it is advisable to choose the largest such integer to control the gradients of the partition of unity function φ_i. Solving (6.1) for α_l we have

$$\alpha_l = \frac{l+1}{l+1-m}.$$

With the choice of $l = 2n-1$ and maximal $m = n$, this yields $\alpha_l = 2$, in general we have $1 < \alpha_l \le 2$. Due to this construction many of the points $x_i \in P$ are covered only by the corresponding ω_i. Therefore, we have $\varphi_i(x_j) = \delta_{ij}$ for many partition of unity functions φ_i and points $x_j \in P$, see Figure 6.2. In fact, $\varphi_i(x) = 1$ holds not only for the point $x = x_i$ if we have $\alpha_l < 2$ but rather on a sub-patch $\tilde{\omega}_i \subset \omega_i$ with $x_i \in \tilde{\omega}_i \sim \bigotimes_{i=1}^d h_l^i$, i.e. $\varphi_i|_{\tilde{\omega}_i} \equiv 1$, see Figure 6.2.

When we compare the covers C_Ω (Figures 3.2 and 6.1) and functions φ_i (Figures 3.3 and 6.2) generated by Algorithms 2 and 3, we clearly see the effect of the alignment of the cover patches.

Note that the change of the point set P in step 3c in Algorithm 3 is admissible due to the non-interpolatory character of the PUM shape functions $\varphi_i\psi_i^k$. We can

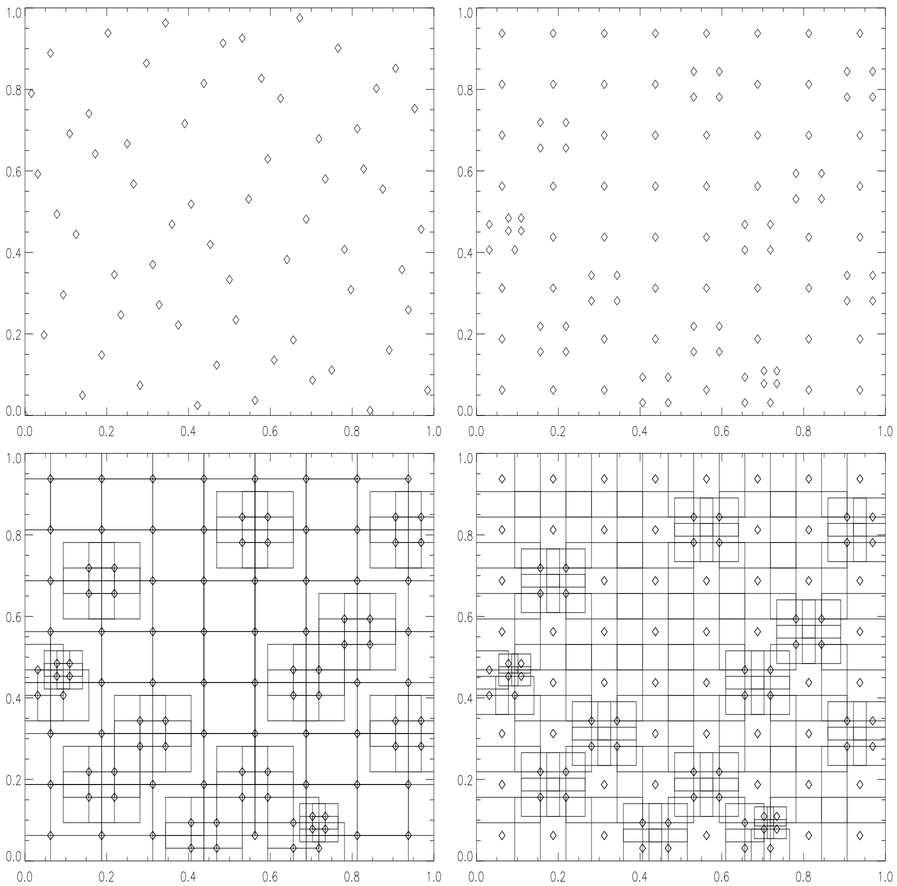

FIGURE 6.1. *$P = Halton_0^{63}(2,3)$ point set in $R_\Omega = \Omega = [0,1]^2$ (upper left), P with* $\mathrm{card}(P) = 106$ *(upper right) after Algorithm 3 with $k = \infty$, and the generated cover C_Ω with $\alpha_l = 2$ (lower left) and $\alpha_l = 1.5$ (lower right).*

interpret this change in the point set P as a change of the weight functions W_k used during the Shepard construction (2.2). So far the weight functions W_k and the cover patches ω_k were assumed to be centered in the given point x_k (compare §2), but this is of course not a necessary condition for the PUM to work. Therefore, we may view the construction given above as a more general approach toward assigning weight functions W_k to a given point $x_k \in P$. The weight functions W_k and cover patches ω_k are now centered in $l_L + \frac{1}{2}(u_L - l_L)$ rather than in the given point x_k. Note that the constructed point set P of newly introduced and shifted points x_k is only part of the implementation of the function space. The initial point set P of step 1 is still the set of all relevant points for the resolution of the solution and the geometry of the domain. Therefore, a copy $\tilde{P}$ of the initial point set P is stored separately and the points $x_l \in \tilde{P}$ are used in time-dependent settings to generate covers for future time steps [13]. Hence by the introduction of general weight functions W_k as part of the cover construction, we can also write step 3 of Algorithm 3 equivalently as

3'. For all cells $C_L = \bigotimes_{i=1}^{d} [l_L^i, u_L^i]$ with $C_L \cap \Omega \neq \emptyset$:
   (a) Set $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$.

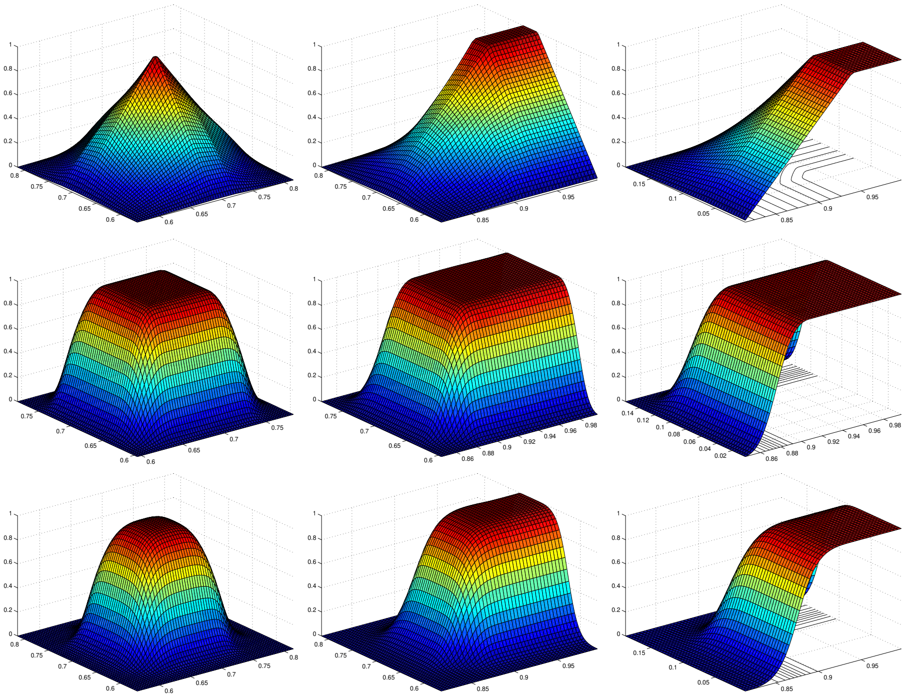

FIGURE 6.2. *The partition of unity function φ_i on $\Omega \cap \omega_i$ generated by Algorithm 3 with the input data (upper row $l = 1$, $\alpha_l = 2$, center row $l = 2$, $\alpha_l = 1.5$, lower row $l = 3$, $\alpha_l = 2$) from 6.1 for an interior point (left), a boundary point (center) and a corner point (right).*

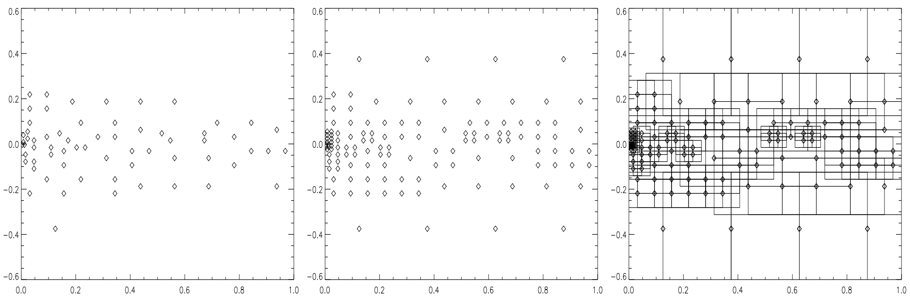

FIGURE 6.3. *P is a Halton$_0^{63}(2,3)$ point set in $R_\Omega = \Omega = [0,1] \times [-0.5, 0.5]$ graded by $(x,y) \mapsto (x^2, \pm y^2)$ (left), P with* $\operatorname{card}(P) = 121$ *(center) after Algorithm 3 with $k = \infty$, and the generated cover C_Ω with $\alpha_l = 2$ (right).*

(b) If there is *no* $x_j \in P$ with $x_j \in C_L$, set $P = P \cup \{x_L\}$.

(c) Set $\omega_L = R_L = \bigotimes_{i=1}^d [x_L^i - h_L^i, x_L^i + h_L^i] \supset C_L$, where $h_L^i = \frac{\alpha_l}{2}(u_L^i - l_L^i)$.

(d) Set the associated weight function $W_L(x) := \Pi_{i=1}^d \mathcal{W}(\frac{x - x_L^i + h_L^i}{2h_L^i})$.

which leaves the given points at their original location. Note also that the cover patches $\omega_L = R_L$ constructed with Algorithm 3 and the bounding box R_Ω always have the same aspect ratio, see Figure 6.1. If we apply the algorithm given above to $\Omega = R_\Omega = [0,1]^d$ with $k = \infty$ and $\alpha_l = 2$ to a uniformly distributed set of points P, we

$d=2$					$d=3$				
N	card(P)	A_2	$A_{3,2}$	$A_{3,1.5}$	N	card(P)	A_2	$A_{3,2}$	$A_{3,1.5}$
1024	1729	11.58	12.58	8.51	1024	3543	26.81	30.90	21.23
4096	6364	11.61	12.35	8.48	8192	26699	29.16	34.44	22.41
16384	27673	11.81	13.01	8.65	65536	199417	31.53	34.80	23.30
65536	101314	12.29	12.43	8.56	524288	1694568	32.31	34.94	24.04

TABLE 6.3

The average number s_{C_Ω} (6.3) of nonzero blocks a_{ij} per row of the stiffness matrix for the different cover construction algorithms in two (left) and three dimensions (right). Algorithm 2 with $k=\infty$, $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L and $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$ (A_2), and Algorithm 3 with $k=\infty$, $\alpha_l = 2$ ($A_{3,2}$) and $\alpha_l = 1.5$ ($A_{3,1.5}$) . The initial point set P was Halton$_0^{N-1}(2,3)$ in $[0,1]^2$ in two dimensions (left) and Halton$_0^{N-1}(2,3,5)$ in $[0,1]^3$ in three dimensions (right).

N	card(P)	A_2^1	$A_{3,2}^1$	$A_{3,1.3}^1$	A_2^2	$A_{3,1.5}^2$	A_2^3	$A_{3,2}^3$
1024	1729	31.54	6.39	7.03	48.82	8.16	76.51	10.77
4096	6364	32.22	5.29	6.60	50.34	7.46	78.51	9.93
16384	27673	32.18	6.28	6.99	49.51	8.27	77.20	10.76
65536	101314	34.65	5.16	6.62	53.75	7.40	82.47	9.85

TABLE 6.4

The average number a_{C_Ω} (6.2) of integration cells per nonzero block a_{ij} of the stiffness matrix for the different cover construction algorithms. Algorithm 2 with $k=\infty$, $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L and $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$ (A_2^l), and Algorithm 3 with $k=\infty$ (A_{3,α_l}^l) using a linear, a quadratic and a cubic B-spline during the Shepard construction (2.2). The initial point set P was Halton$_0^{N-1}(2,3)$ in $[0,1]^2$.

construct a uniform grid[9] (or at least an r-irregular grid with very small r depending only on the quality of the initial point set P, see Figure 6.1). Here, also the cells $D_{\omega_{ij}}^n$ of the decomposition $D_{\omega_{ij}}$ are (geometrically) identical to a bilinear finite element. Furthermore, the partition of unity $\{\varphi_i\}$ generated by (2.2) will again be piecewise linear for $l=1$ just like their FE counterpart in the GFEM (see Figure 6.2). Hence, in this situation our method does reconstruct functions φ_i that are identical to bilinear finite element functions and also our general decomposition algorithm will recover the corresponding geometric elements. Hence, the number of integrals to be evaluated here with our method or a finite element method are the same. We give the average number

$$a_{C_\Omega} := \frac{\sum_{i=1, j\in N_i}^{\mathrm{card}(P)} \mathrm{card}(D_{\omega_{ij}})}{\sum_{i=1}^{\mathrm{card}(P)} \mathrm{card}(N_i)} \tag{6.2}$$

of integration cells per nonzero block a_{ij} of the stiffness matrix in Tables 6.4 and 6.5. Furthermore, we give the average number

$$s_{C_\Omega} := \frac{\sum_{i=1}^{\mathrm{card}(P)} \mathrm{card}(N_i)}{\mathrm{card}(P)} \tag{6.3}$$

of nonzero blocks a_{ij} per block-row of the stiffness matrix in Table 6.3 which corresponds to the number of entries in a finite element stencil. For a one-dimensional uniform grid the average a_{C_Ω} is $\frac{4}{3}$ for hat-functions (at interior points, where we have

[9]The covers from Algorithm 3 with $k=0$ will correspond to a uniform grid regardless of the initial point set P when we have the order $l = 2n-1$ and maximal $m=n$.

N	card(P)	A_2^1	$A_{3,2}^1$	$A_{3,1.3}^1$	A_2^2	$A_{3,1.5}^2$	A_2^3	$A_{3,2}^3$
1024	3543	250.03	16.34	20.71	464.61	17.61	877.89	29.19
8192	26699	302.20	15.96	22.48	553.66	18.59	1048.01	29.41
65536	199417	363.77	12.09	20.63	668.04	16.59	563.76	24.63
524288	1694568	–	12.44	26.52	–	15.72	–	25.07

TABLE 6.5

The average number a_{C_Ω} (6.2) of integration cells per nonzero block a_{ij} of the stiffness matrix for the different cover construction algorithms. Algorithm 2 with $k = \infty$, $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L and $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$ (A_2^l), and Algorithm 3 with $k = \infty$ (A_{3,α_l}^l) using a linear, a quadratic and a cubic B-spline during the Shepard construction (2.2). The initial point set P was $\mathrm{Halton}_0^{N-1}(2,3,5)$ in $[0,1]^3$.

$N_i = \{\omega_{i-1}, \omega_i, \omega_{i+1}\}$). This situation corresponds of course to the case $l = 1$, $m = 1$, $\alpha_l = 2$ in Algorithm 3. Due to the tensor product approach, we have $(\frac{4}{3})^d$ as the optimal ratio of integration cells to nonzero blocks for $l = 1$ in the d-dimensional case. We certainly cannot expect to meet this optimal ratio for an irregular cover.

From the averages displayed in Tables 6.4 and 6.5 for the two-dimensional and three-dimensional case we see that the averages a_{C_Ω} are (almost) independent of the number of points N of the initial point set P for Algorithm 3 as well as for Algorithm 2. Furthermore, we clearly see the substantial reduction in the number of integration cells for Algorithm 3 compared with Algorithm 2. The average a_{C_Ω} for Algorithm 3 drops by more than a factor of $\frac{1}{6}$ in two dimensions and by more than a factor of $\frac{1}{18}$ in three dimensions compared with the average a_{C_Ω} for covers constructed by Algorithm 2. This significant improvement in the number of integration cells is of course due to the alignment of the patches we have with Algorithm 3 but not with Algorithm 2, see Figures 6.1 and 3.2, Figures 6.3 and 3.4. Another advantage of Algorithm 3 over Algorithm 2 is the fact that due to the alignment of the weight subdivisions $\{\omega_k^q\}$ the aspect ratio and volume of every integration cell $D_{\omega_{ij}}^n$ is bounded. This is not the case for covers constructed with Algorithm 2. In fact, in the three-dimensional example of Table 6.5 our decomposition algorithm would have generated a number of integration cells with very large aspect ratios and almost vanishing volume in the case $N = 524288$ for a cover constructed with Algorithm 2.

The average a_{C_Ω} for covers from Algorithm 3 (with $l = 1$) is about three times the optimal ratio of $(\frac{4}{3})^d$. For one this factor can be explained by the sudden change in the spatial resolution of the cover, i.e. the level difference k of two neighboring patches, which leads to nonaligned subdivisions $\{\omega_k^q\}$, see Figures 6.1 and 6.3, and therefore increases the number of integration cells. Moreover though, the optimal ratio of $(\frac{4}{3})^d$ holds for interior patches only. For cover patches which overlap the boundary of the domain this ratio is 2 (or even 2.25 at corners) in two dimensions since the subdivisions $\{\omega_k^q\}$ are only aligned with each other but *not* with the boundary $\partial\Omega$, see Figure 6.4. Hence, we expect the average number of integration cells a_{C_Ω} to decrease for larger N since the volume to surface ratio improves. This can be observed from the numbers $A_3^{1_1}$ displayed in Tables 6.4 and 6.5. A similar argument can be made in the case of a quadratic B-spline[10].

When we use a cubic B-spline ($l = 3$) we construct smooth approximations $u_{\mathrm{PU}} \in$

[10]In the case $l = 2$, we have an optimal ratio of $\frac{5}{3}$ in one dimension. But due to the fact, that the overlap $m = n$ for $l = 2n$ is smaller in relation to the overall number of cells card$(\{\omega_k^q\})$, see Figure 6.1, the generalization of this optimum to higher dimensions is *not* given by $(\frac{5}{3})^d$; e.g. in two dimensions the optimum is $\frac{22}{9}$ only, see Figure 6.4.

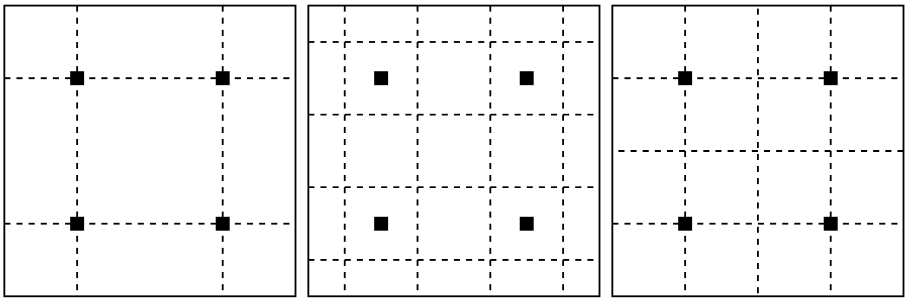

FIGURE 6.4. *The integration cells induced by four neighboring linear B-splines (left), quadratic B-splines (center) and cubic B-splines (right) in two dimensions. The integration cells $D^n_{\omega_{ij}}$ align with the boundary for the cubic B-spline (right) but not for linear (left) or quadratic (center) B-spline due to the overlap condition for the support patches ω_i.*

$\mathcal{C}^2$ and the optimal ratio for interior patches is $(\frac{8}{3})^d$, i.e. it is about 7 in two dimensions and 19 in three dimensions. The averages a_{C_Ω} given in Tables 6.4 and 6.5 in this case are about 10 in two dimensions and 26 in three dimensions, i.e. they are closer to their optimal ratio of $(\frac{8}{3})^d$ than a_{C_Ω} is its optimum for the linear B-spline. This can be explained by the fact that for $l = 3$ the boundary effect mentioned above does not exist, see Figure 6.4. Here, the cells $\{\omega^q_k\}$ induced by the cubic spline align with the boundary of $[0,1]^d$. Hence, only the irregularity of the cover causes an increase in the number of integration cells. Overall the number of integration cells for an approximate solution $u_{\mathrm{PU}} \in \mathcal{C}^2$ is about twice the number of integration cells we have when we construct an approximate solution $u_{\mathrm{PU}} \in \mathcal{C}^0$.

So far we were only concerned with the computational cost during the integration and the influence the shape functions $\varphi_i\psi^k_i$ have on the computational efficiency of our PUM. Another important issue though is the stability of the basis of our PUM space. Here, we also have to address the question if the functions $\varphi_i\psi^k_i$ are indeed a basis. In the case of $l = 1$ and $\alpha_l = 2$ the alignment of the cover patches ω_i and their respective weight subdivisions $\{\omega^q_i\}$ leads to the reconstruction of the finite element hat functions for the PU. Hence, our PUM reduces to the GFEM in this situation. It is well-known [2, 3, 27, 30] that the GFEM (in general) generates linear dependent shape functions $\varphi_i\psi^k_i$, the so-called nullity of the method. This is essentially due to the fact that in the GFEM the partition of unity functions φ_i already reconstruct the linear polynomial.

Consider the one-dimensional situation, where we have one element and two nodes with their associated hat function as φ_i. Assume that we use linear polynomials as local approximations spaces V_i. The shape functions $\varphi_i\psi^k_i$ are (global) polynomials due to this construction. The number of shape functions is four and the maximal polynomial degree is two. Since the quadratic polynomials in one dimension can be generated by three basis functions, we see that the GFEM shape functions are linear dependent.

With our approach, the φ_i reconstruct the linear polynomial only away from the boundary, close to the boundary we have $\varphi_i \equiv 1$. Therefore, the shape functions are not linear dependent. But since the small boundary layer where $\varphi_i \equiv 1$ decreases with larger N the condition number κ of the mass matrix is dependent on N, i.e. the basis is no longer stable. A simple cure for this stability problem is to use $m < 1$ in (6.1) when we have $l = 1$, i.e. we limit ourselves to $1 < \alpha_l < 2$ when $l = 1$. With $\alpha_l < 2$ we can find a sub-patch $\tilde{\omega}_i \subset \omega_i$ where $\varphi_i|_{\tilde{\omega}_i} \equiv 1$ for many i. Therefore, the partition of unity functions φ_i no longer reconstruct the linear polynomial independent of N and the resulting shape functions form a stable basis. We therefore allow for any value $1 < \alpha_l < 2$ in Algorithm 3 if $l = 1$. The number of integration cells increases somewhat due to this generalization. The patches ω_i are still aligned but their respective weight subdivisions are not. The optimal ratio increases from $(\frac{4}{3})^d$ to 2^d. From the averages displayed in Tables 6.4 and 6.5 we see that a_{C_Ω} for the choice of $\alpha_l = 1.3$ is about a factor of 1.5 to 2 larger than the average is for the optimal choice of $\alpha_l = 2$. But still the number of integration cells is substantially less compared with the covers from Algorithm 2.

A similar problem arises for higher order splines $l > 1$ only when $\alpha_l > 2$, e.g. we need $\alpha_l = 4$ with $l = 2$. Therefore, we can stay with our choices of $\alpha_l = 1.5$ if $l = 2$ and $\alpha_l = 2$ if $l = 3$.

7. Numerical Experiments. In this section we present some results of our numerical experiments for elliptic PDE using the h-version and the p-version of our partition of unity method.

We apply our PUM to elliptic problems on the unit cell $\Omega = (0,1)^d$ in two and three dimensions. Here, we consider the Laplace equation

$$-\Delta u = f \tag{7.1}$$

with Dirichlet boundary conditions $u = g$ on $\partial\Omega$, and the equation

$$-\Delta u + u = f \tag{7.2}$$

of Helmholtz type with Neumann boundary conditions $\nabla u = g$ on $\partial\Omega$. Furthermore, we use the presented method to study heat conduction in lattice materials [20, 26] in three dimensions.

The local approximation spaces $V_i^{p_i}$ used in our numerical experiments are complete Legendre polynomials with $p_i = p$ for all patches ω_i. The weight functions W_i used in the Shepard construction (2.2) are linear splines ($l = 1$, $\alpha_l = 1.3$). We give the relative error

$$e = \frac{\|u - u_{\mathrm{PU}}\|}{\|u\|}$$

in the L^∞-, L^2- and the energy-norm, which is computed with the help of the integration scheme presented in §5. Moreover, we also give the convergence rates

$$\rho = \frac{\log\left(\frac{\|u-u_{\mathrm{PU},L}\|}{\|u-u_{\mathrm{PU},L-1}\|}\right)}{\log\left(\frac{\mathsf{dof}_L}{\mathsf{dof}_{L-1}}\right)},$$

where $\mathsf{dof} := \sum \dim\left(V_i^{p_i}\right)$, with respect to two successive refinement levels L and $L-1$. These convergence rates ρ correspond to an algebraic error estimate

$$\|u - u_{\mathrm{PU},L}\| = O\left(\mathsf{dof}_L^\rho\right) \tag{7.3}$$

which is valid for the h-version of the PUM [2, 3]. We can relate these rates ρ to the common h^α notation by $\alpha = -\rho d$ since $N^{-\frac{1}{d}} \sim h$ for uniform point sets. Hence, the optimal convergence rates ρ for our PUM based on uniform point sets and linear

N	card(P)	p	dof	e_∞	ρ^A_∞	e_2	ρ^A_2	e_E	ρ^A_E
16	28	1	84	5.739_{-2}	–	6.462_{-2}	–	2.676_{-1}	–
64	106	1	318	1.817_{-2}	-8.639_{-1}	1.726_{-2}	-9.917_{-1}	1.403_{-1}	-4.850_{-1}
256	406	1	1218	5.700_{-3}	-8.633_{-1}	4.405_{-3}	-1.017_{0}	6.963_{-2}	-5.217_{-1}
1024	1729	1	5187	1.926_{-3}	-7.488_{-1}	1.072_{-3}	-9.753_{-1}	3.462_{-2}	-4.823_{-1}
4096	6364	1	19092	1.101_{-3}	-4.291_{-1}	2.749_{-4}	-1.044_{0}	1.762_{-2}	-5.183_{-1}
16384	27673	1	83019	4.431_{-4}	-6.193_{-1}	6.749_{-5}	-9.555_{-1}	8.677_{-3}	-4.819_{-1}
65536	101314	1	303942	2.099_{-4}	-5.757_{-1}	1.716_{-5}	-1.055_{0}	4.374_{-3}	-5.278_{-1}

TABLE 7.1

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.1) in two dimensions with solution (7.5).

N	card(P)	p	dof	e_∞	ρ_∞	e_2	ρ_2	e_E	ρ_E
64	106	1	318	2.942_{-2}	–	1.476_{-2}	–	1.341_{-1}	–
64	106	2	636	7.960_{-4}	-5.208_{0}	3.448_{-4}	-5.420_{0}	5.761_{-3}	-4.541_{0}
64	106	3	1060	1.465_{-5}	-7.821_{0}	1.046_{-5}	-6.843_{0}	1.986_{-4}	-6.592_{0}
64	106	4	1590	9.093_{-8}	-1.253_{1}	1.091_{-7}	-1.125_{1}	2.612_{-6}	-1.068_{1}
64	106	5	2226	8.003_{-9}	-7.223_{0}	2.640_{-9}	-1.106_{1}	7.121_{-8}	-1.071_{1}
64	106	6	2968	8.579_{-10}	-7.762_{0}	2.588_{-10}	-8.073_{0}	8.429_{-9}	-7.418_{0}
64	106	7	3816	3.710_{-10}	-3.336_{0}	6.482_{-11}	-5.509_{0}	2.481_{-9}	-4.866_{0}
64	106	8	4770	9.538_{-11}	-6.087_{0}	2.183_{-11}	-4.877_{0}	9.217_{-10}	-4.437_{0}
64	106	9	5830	4.137_{-11}	-4.163_{0}	8.451_{-12}	-4.729_{0}	4.100_{-10}	-4.037_{0}
64	106	10	6996	1.401_{-11}	-5.939_{0}	3.075_{-12}	-5.545_{0}	1.700_{-10}	-4.829_{0}
64	106	11	8268	8.220_{-12}	-3.192_{0}	1.535_{-12}	-4.159_{0}	8.807_{-11}	-3.937_{0}
64	106	12	9646	3.836_{-12}	-4.944_{0}	7.533_{-13}	-4.618_{0}	4.751_{-11}	-4.004_{0}

TABLE 7.2

Errors (e) and convergence rates (ρ) in different norms for problem (7.2) in two dimensions with solution (7.5).

polynomials are $\rho_2 = -1$ and $\rho_E = -\frac{1}{2}$ in two dimensions ($\rho_2 = -\frac{2}{3}$ and $\rho_E = -\frac{1}{3}$ in three dimensions). For the pointwise convergence[11] we have $\rho_2 < \rho_\infty < \rho_E$.

For the p-version, we expect an exponential convergence for smooth solutions u since the local error estimate

$$\|u - u_{\mathrm{PU},L}\| = O\left(\exp\left(-b\sqrt{\mathsf{dof}_L}\right)\right) \tag{7.4}$$

holds on every cover patch ω_i for smooth solutions u.

Example 1 (Unit Square). In our first example we consider the Dirichlet problem (7.1) in $\Omega = (0,1)^2$. We choose f and g such that the solution u is given by

$$u(x) = \|x\|_2^5. \tag{7.5}$$

We apply the h-version of our PUM with linear polynomials to approximate (7.1). The covers C_Ω are generated using N points of the Halton$(2,3)$ sequence with increasing N. They exhibit a somewhat locally varying patch size, see Figure 6.1. Consequently, there will be some fluctuation in the measured convergence rates ρ.

The results for the h-version experiment with linear polynomials are given in Table 7.1. The measured rates ρ show the algebraic convergence (7.3) of our PUM

[11] In the finite element method we have the estimate $\|u-u_h\|_\infty = O(h^2|\log h|^{\mu(d)})$, where $\mu(2) = 1$ and $\mu(d) = \frac{d}{4}+1$ for $d \geq 3$ [25, 28]. The L^∞-norm is usually approximated by the maximum over the nodal values, where we can observe a super-convergence of order h^2. For our approximation to the L^∞-norm though we do not use the points $x_i \in P$ but all quadrature points since the PUM shape functions $\varphi_i\psi_i^n$ are non-interpolatory and the Legendre polynomials ψ_i^n of odd degree vanish at x_i. Hence, we cannot expect to measure h^2 super-convergence in the L^∞-norm.

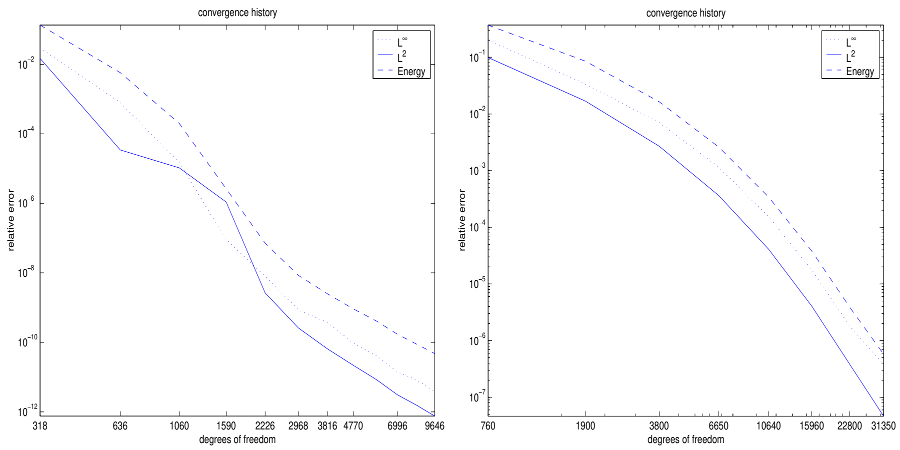

FIGURE 7.1. *Convergence history for problem (7.2) in two dimensions (left) with solution (7.5). Convergence history for problem (7.2) in three dimensions (right) with solution (7.6).*

N	card(P)	p	dof	e_∞	ρ_∞^A	e_2	ρ_2^A	e_E	ρ_E^A
16	50	1	200	4.565_{-1}	–	1.182_0	–	1.828_0	–
128	414	1	1656	2.580_{-1}	-2.699_{-1}	1.499_{-1}	-9.769_{-1}	4.273_{-1}	-6.876_{-1}
1024	3543	1	14172	7.914_{-2}	-5.505_{-1}	5.452_{-2}	-4.711_{-1}	2.683_{-1}	-2.168_{-1}
8192	26699	1	106796	3.119_{-2}	-4.610_{-1}	9.091_{-3}	-8.869_{-1}	1.197_{-1}	-3.996_{-1}

TABLE 7.3

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.1) in three dimensions with solution (7.6).

in the L^2- and energy-norm with rates ρ near the optimal values of $\rho_2 = -1$ and $\rho_E = -\frac{1}{2}$ for the h-version. In the L^∞-norm we measure a convergence rate ρ_∞ of -0.6 which is between -1 and $-\frac{1}{2}$ as expected.[11]

Let us now consider the Neumann problem (7.2). Again, the solution is given by (7.5). Here, we apply the p-version of our PUM. Since the solution (7.5) is not analytic in Ω we may not expect an exponential convergence of the p-version. From the numbers given in Table 7.2 and the convergence history displayed in Figure 7.1 (left) we can observe that the convergence starts off at an exponential rate but breaks down to a polynomial rate as anticipated for higher polynomial degrees p.

Example 2 (Unit Cube). In our second example we consider (7.1) with Dirichlet boundary conditions on the unit cube in three dimension. Here, we now choose f and g such that the solution u is given by

$$u(x) = \exp\left(4\|x\|_1\right). \tag{7.6}$$

Again, we use the h-version of our PUM with linear polynomials to approximate (7.1). The covers are generated using N points of the Halton$(2,3,5)$ sequence. The numerical results are given in Table 7.3. Here, it seems that the convergence in the L^2- and the energy-norm is actually better than the theory (for uniform point sets) suggests. We measure a convergence rate ρ_2 close to -1 but would only expect a convergence rate of about $-\frac{2}{3}$. This behavior though is due to the fact that the size of patches ω_i based on a Halton sequence varies much more in three dimensions than in two dimensions (for a small number of samples). Therefore, a larger fluctuation in the measured convergence rates ρ will occur. If we use a different number N of Halton

N	card(P)	p	dof	e_∞	ρ^A_∞	e_2	ρ^A_2	e_E	ρ^A_E
16	50	1	200	7.184_{-1}	–	4.321_{-1}	–	7.741_{-1}	–
32	106	1	424	4.125_{-1}	-7.383_{-1}	1.384_{-1}	-1.515_0	4.846_{-1}	-6.232_{-1}
64	190	1	760	2.020_{-1}	-1.224_0	9.887_{-2}	-5.766_{-1}	3.701_{-1}	-4.618_{-1}
128	414	1	1656	1.966_{-1}	-3.457_{-2}	8.198_{-2}	-2.405_{-1}	3.351_{-1}	-1.277_{-1}
256	673	1	2692	1.838_{-1}	-1.387_{-1}	3.729_{-2}	-1.621_0	2.630_{-1}	-4.985_{-1}
512	1415	1	5660	8.673_{-2}	-1.011_0	3.464_{-2}	-9.924_{-2}	2.364_{-1}	-1.434_{-1}
1024	3543	1	14172	8.565_{-2}	-1.366_{-2}	2.628_{-2}	-3.008_{-1}	1.936_{-1}	-2.180_{-1}
2048	5874	1	23496	6.756_{-2}	-4.692_{-1}	9.380_{-3}	-2.038_0	1.340_{-1}	-7.267_{-1}
4096	10606	1	42424	6.734_{-2}	-5.719_{-3}	9.039_{-3}	-6.263_{-2}	1.296_{-1}	-5.676_{-2}
8192	26699	1	106796	3.934_{-2}	-5.821_{-1}	7.021_{-3}	-2.738_{-1}	1.016_{-1}	-2.643_{-1}
16384	45151	1	180604	2.197_{-2}	-1.109_0	2.334_{-3}	-2.096_0	6.745_{-2}	-7.790_{-1}

TABLE 7.4

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.2) in three dimensions with solution (7.6). The covers C_Ω are based on a Halton$(2,3,5)$ point set.

N	card(P)	p	dof	e_∞	ρ^A_∞	e_2	ρ^A_2	e_E	ρ^A_E
64	64	1	256	4.124_{-1}	–	1.382_{-1}	–	4.846_{-1}	–
512	512	1	2048	1.838_{-1}	-3.887_{-1}	3.734_{-2}	-6.295_{-1}	2.634_{-1}	-2.931_{-1}
4096	4096	1	16384	6.751_{-2}	-4.816_{-1}	9.409_{-3}	-6.628_{-1}	1.351_{-1}	-3.211_{-1}
32768	32768	1	131072	2.199_{-2}	-5.395_{-1}	2.332_{-3}	-6.708_{-1}	6.787_{-2}	-3.311_{-1}

TABLE 7.5

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.2) in three dimensions with solution (7.6). The covers C_Ω are based on a uniform point set.

points for the initial cover or refine the covers using only twice as many points instead of eight times as many, this behavior becomes much more obvious. To this end we also give the numerical results of an h-version experiment with linear polynomials applied to the Neumann problem (7.2) with solution (7.6) in Table 7.4. Here, we clearly see the fluctuation in the convergence rates due to the unstructured refinement induced by the Halton sequence. If we use the grid points of a uniform grid to generate the covers C_Ω we can observe a very good correspondence of the measured convergence rates ρ with those of the theory, see Table 7.5 where the corresponding numerical results for the Neumann problem (7.2) with solution (7.6) are given.

We now turn to the p-version of our partition of unity method in three dimensions. The numerical results for problem (7.2) with Neumann boundary conditions are given in Table 7.6. For the smooth solution (7.6) we expect an exponential convergence behavior of the p-version of our PUM. From the measured values of ρ and the convergence history given in Figure 7.1 (right) we clearly see this behavior.

In summary we have that on simple domains in two and three dimensions the PUM works with the anticipated convergence properties. Now we turn to more complicated (yet still academic) computational domains in three dimensions.

Example 3 (Lattice Material). In our third example we study the problem of heat conduction in a lattice material. To this end, we use our PUM to discretize the PDE

$$-\Delta u + u = f \text{ in } \Omega \subset (0,1)^d \text{ where } d = 2,3$$

on a domain Ω which describes a lattice material with a characteristic cell width of $\frac{4}{14}$ and a cell number of 3. As boundary conditions we use the Neumann conditions

$$\nabla u = g = \begin{cases} -10(\frac{1}{4} - 2x): & \Gamma_I := \{x \in \partial\Omega \,|\, x_0 = 0 \text{ and } \|x\|^2 < \frac{1}{4}\} \\ 0: & \Gamma_O := \partial\Omega \setminus \Gamma_I \end{cases}$$

N	card(P)	p	dof	e_∞	ρ_∞	e_2	ρ_2	e_E	ρ_E
64	190	1	760	2.020_{-1}	–	9.887_{-2}	–	3.701_{-1}	–
64	190	2	1900	3.355_{-2}	-1.959_{0}	1.682_{-2}	-1.933_{0}	8.527_{-2}	-1.602_{0}
64	190	3	3800	7.049_{-3}	-2.251_{0}	2.698_{-3}	-2.640_{0}	1.644_{-2}	-2.375_{0}
64	190	4	6650	1.150_{-3}	-3.240_{0}	3.631_{-4}	-3.584_{0}	2.606_{-3}	-3.291_{0}
64	190	5	10640	1.536_{-4}	-4.283_{0}	4.095_{-5}	-4.643_{0}	3.408_{-4}	-4.328_{0}
64	190	6	15960	1.763_{-5}	-5.339_{0}	4.062_{-6}	-5.699_{0}	3.810_{-5}	-5.404_{0}
64	190	7	22800	1.800_{-6}	-6.397_{0}	3.809_{-7}	-6.636_{0}	3.882_{-6}	-6.403_{0}
64	190	8	31350	3.818_{-7}	-4.869_{0}	4.652_{-8}	-6.603_{0}	5.759_{-7}	-5.992_{0}

TABLE 7.6

Errors (e) and convergence rates (ρ) in different norms for problem (7.2) in three dimensions with solution (7.6).

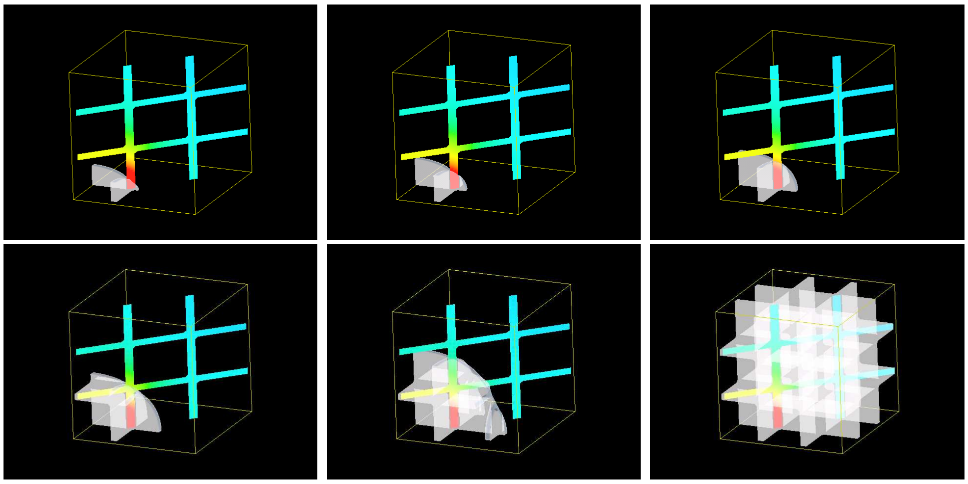

FIGURE 7.2. *Isosurfaces of approximate solution and diagonal slice through lattice domain.*

to simulate heat-introduction into the material at the contact points Γ_I to a heat source and out-flow conditions on the remaining boundary Γ_O.

We use a Halton$_0^{4095}(2,3,5)$ set distributed in the bounding box $R_\Omega := (0,1)^3$ as initial point set P for our cover construction. The local approximation spaces $V_i^{p_i}$ that are assigned to the 5187 patches which overlap the computational domain Ω are quadratic Legendre polynomials. In Figure 7.2 some isosurfaces of the computed solution and a diagonal slice through the material are displayed. From these illustrations we observe the heat-introduction into the material at the contact points Γ_I and the heat-propagation through the material. We clearly see the expected radial shape of the isosurfaces due to the prescribed profile of the Neumann boundary conditions on Γ_I.

8. Concluding Remarks. We presented a meshfree Galerkin method for the discretization of a PDE. The method is based on the partition of unity approach and utilizes a novel tree-based cover construction algorithm. The introduction of this algorithm for the cover construction problem and the presented numerical quadrature scheme—both of which are applicable to general domains—improve the computational efficiency during the assembly of the stiffness matrix substantially.

The results of our numerical experiments showed the exponential convergence of the p-version of our PUM for smooth solutions and the anticipated algebraic convergence of the h-version in two and three dimensions. Furthermore, we applied our PUM to the heat conduction problem in lattice materials. Although these examples are still of academic nature, we believe that the substantial improvement of the computation efficiency due to the ideas and algorithms presented in this paper makes the treatment of real world problems with meshfree Galerkin methods seizable in the near future, at least for Neumann boundary conditions. The implementation of Dirichlet boundary conditions in meshfree methods is in general a challenging problem which needs further investigation.

The presented hierarchical cover construction algorithm not only reduces the computational costs significantly but also introduces a hierarchy on the PUM function space. This may be exploited in the development of multilevel solvers [14]. The parallelization of our PUM [15] may be simplified by the tree-based cover construction. Here, we can apply a space filling curves parallelization approach [6] which allows for a cheap dynamic load balancing strategy.

REFERENCES

[1] N. R. Aluru, *A Point Collocation Method Based on Reproducing Kernel Approximations*, Int. J. Numer. Meth. Engrg., 47 (2000), pp. 1083–1121.

[2] I. Babuška and J. M. Melenk, *The Partition of Unity Finite Element Method: Basic Theory and Applications*, Comput. Meth. Appl. Mech. Engrg, 139 (1996), pp. 289–314. Special Issue on Meshless Methods.

[3] ———, *The Partition of Unity Method*, Int. J. Numer. Meth. Engrg., 40 (1997), pp. 727–758.

[4] T. Belytschko, Y. Y. Lu, and L. Gu, *Element-free Galerkin methods*, Int. J. Numer. Meth. Engrg., 37 (1994), pp. 229–256.

[5] M. Bern, D. Eppstein, and J. Gilbert, *Provably Good Mesh Generation*, J. of Comp. Sys. Sci., 48 (1994), pp. 384–409.

[6] A. Caglar, M. Griebel, M. A. Schweitzer, and G. Zumbusch, *Dynamic Load-Balancing of Hierarchical Tree Algorithms on a Cluster of Multiprocessor PCs and on the Cray T3E*, in Proceedings 14th Supercomputer Conference, Mannheim, H. W. Meuer, ed., Mannheim, Germany, 1999, Mateo.

[7] C. A. M. Duarte, I. Babuška, and J. T. Oden, *Generalized Finite Element Methods for Three Dimensional Structural Mechanics Problems*, Computers and Structures, 77 (2000), pp. 215–232.

[8] C. A. M. Duarte and J. T. Oden, *hp Clouds – A Meshless Method to Solve Boundary Value Problems*, Num. Meth. for PDE, 12 (1996), pp. 673–705.

[9] G. Fasshauer, *Solving Differential Equations with Radial Basis Functions: Multilevel Methods and Smoothing*, Adv. Comp. Math., 11 (1999), pp. 139–159.

[10] C. Franke and R. Schaback, *Convergence Orders of Meshless Collocation Methods using Radial Basis Functions*, Adv. in Comput. Math., 8 (1998), pp. 381–399.

[11] ———, *Solving Partial Differential Equations by Collocation using Radial Basis Functions*, Appl. Math. and Comput., 93 (1998), pp. 73–82.

[12] T. Gerstner and M. Griebel, *Numerical Integration using Sparse Grids*, Numer. Algorithms, 18 (1998), pp. 209–232.

[13] M. Griebel and M. A. Schweitzer, *A Particle-Partition of Unity Method for the Solution of Elliptic, Parabolic and Hyperbolic PDE*, SIAM J. Sci. Comp., 22 (2000), pp. 853–890.

[14] ———, *A Particle-Partition of Unity Method—Part III: A Multilevel solver*, SFB Preprint, Sonderforschungsbereich 256, Institut für Angewandte Mathematik, Universität Bonn, 2001. submitted.

[15] ———, *A Particle-Partition of Unity Method—Part IV: Parallelization*, SFB Preprint, Sonderforschungsbereich 256, Institut für Angewandte Mathematik, Universität Bonn, 2001. in preparation.

[16] F. C. Günther and W. K. Liu, *Implementation of Boundary Conditions for Meshless Methods*, Comput. Meth. Appl. Mech. Engrg., 163 (1998), pp. 205–230.

[17] X. Han, S. Oliveira, and D. Stewart, *Finding sets Covering a Point with Application to Mesh-Free Galerkin Methods*, SIAM J. Comput., 30 (2000), pp. 1368–1383.

[18] O. Klaas and M. S. Shepard, *Automatic Generation of Octree-based Three-Dimensional Discretizations for Partition of Unity Methods*, Comput. Mech., 25 (2000), pp. 296–304.

[19] T. J. Liszka, C. A. M. Duarte, and W. W. Tworzydlo, *hp-Meshless Cloud Method*, Comp. Meth. Appl. Mech. Engrg., 139 (1996), pp. 263–288.
[20] A.-M. Matache, I. Babuška, and C. Schwab, *Generalized p-FEM in Homogenization*, Numer. Math., 86 (2000), pp. 319–375.
[21] J. J. Monaghan, *Why Particle Methods Work*, SIAM J. Sci. Stat. Comput., 3 (1982), pp. 422–433.
[22] ———, *An Introduction to SPH*, Comput. Phys. Comm., 48 (1988), pp. 89–96.
[23] H. Neunzert, A. Klar, and J. Struckmeier, *Particle Methods: Theory and Applications*, Tech. Rep. 95-153, Arbeitsgruppe Technomathematik, Universität Kaiserslautern, 1995.
[24] E. Novak, K. Ritter, R. Schmitt, and A. Steinbauer, *On a recent interpolatory method for high dimensional integration*, J. Comput. Appl. Math., 15 (1999), pp. 499–522.
[25] R. Rannacher, *Zur L^∞-Konvergenz linearer finiter Elemente beim Dirichletproblem*, Math. Z., 149 (1976), pp. 69–77.
[26] C. Schwab and A.-M. Matache, *High order generalized FEM for lattice materials*, in Proceedings of the 3rd European Conference on Numerical Mathematics and Advanced Applications, 1999, Finnland, P. Neittaanmäki, T. Tiihonen, and P. Tarvainen, eds., 2000.
[27] M. A. Schweitzer, *Ein Partikel–Galerkin–Verfahren mit Ansatzfunktionen der Partition of Unity Method*, Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, 1997.
[28] R. Scott, *Optimal L^∞-Estimates for the Finite Element Method on Irregular Meshes*, Math. Comp., 30 (1976), pp. 681–697.
[29] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice–Hall, 1973.
[30] T. Strouboulis, I. Babuška, and K. Copps, *The Design and Analysis of the Generalized Finite Element Method*, Comp. Meth. Appl. Mech. Engrg., 181 (1998), pp. 43–69.