A Particle-Partition of Unity Method Part VI: A p-robust Multilevel Solver

Michael Griebel[1]*, Peter Oswald[2]**, and Marc Alexander Schweitzer[1]***

[1] Institut für Numerische Simulation, Rheinische Friedrich–Wilhelms Universität Bonn, Wegelerstraße 6, D–53115 Bonn, Germany.

[2] International University Bremen, School of Engineering and Science, Campus Ring 1, D–28759 Bremen, Germany.

Abstract In this paper we focus on the efficient multilevel solution of linear systems arising from a higher order discretization of a second order partial differential equation using a partition of unity method. We present a multilevel solver which employs a tree-based spatial multilevel sequence in conjunction with a domain decomposition type smoothing scheme. The smoother is based on an overlapping subspace splitting, where the subspaces contain all interacting local polynomials. The resulting local subspace problems are solved exactly. This leads to a computational complexity of the order $O(Np^{3d})$ per iteration. The results of our numerical experiments indicate that the convergence rate of this multilevel solver is independent of the number of points N and the approximation order p. Hence, the overall complexity of the solver is of the order $O(\log(1/\epsilon)Np^{3d})$ to reduce the initial error by a prescribed factor ϵ.

1 Introduction

The particle–partition of unity method (PUM) [7, 8, 9, 10, 11, 18] is a meshfree Galerkin method for the numerical treatment of partial differential equations (PDE). In essence, it is a generalized finite element method (GFEM) which employs piecewise rational shape functions rather than piecewise polynomial functions. The shape functions $\varphi_i\psi_i^n$ of a PUM are products of a partition of unity (PU) $\{\varphi_i\}$ and local approximation functions ψ_i^n which are usually chosen as polynomials. In contrast to other GFEM approaches [19, 20], these shape functions are linearly independent and make up a basis of the discrete function space. This allows us to construct fast multilevel solvers in a similar fashion as in the finite element method (FEM) [12, 21].

In most meshfree discretizations the two most time-consuming tasks are the assembly of the stiffness matrix and the load vector, i.e. numerical integration, and the solution of the resulting linear system. With respect to the

* griebel@ins.uni-bonn.de

** p.oswald@iu-bremen.de

*** schweitzer@ins.uni-bonn.de

asymptotic complexity of these steps the assembly of the stiffness matrix is usually directly proportional to the number of nonzeros nnz of the matrix. The solution of the linear system, however, may only be achieved with similar complexity if an optimal iterative solver is employed. For example, the complexity of a direct solver such as LU- or Cholesky-decomposition is in general much larger. In the case of dense matrices a direct solver requires $O(\mathsf{dof}^3)$ operations and $O(\mathsf{dof}^2)$ storage. With more advanced sparse direct solvers these complexities can be reduced to some extent only. In the special case of regular meshes in two dimensions for instance a nested dissection solver requires $O(\mathsf{dof}^{3/2})$ operations and $O(\mathsf{dof}\ln(\mathsf{dof}))$ storage [5] whereas the optimal computational complexity is of the order $O(\mathsf{dof}) = O(\mathsf{nnz})$. Hence, this optimal storage and operation complexity will be lost when a direct solver is employed.

For a PUM discretization in d dimensions, the number of degrees of freedom is of the order $\mathsf{dof} = O(Np^d)$ where N denotes the number of points or particles and p is the approximation order. Here, the number of nonzeros nnz is of the order $O(Np^{2d})$. Thus, the optimal complexity for the assembly of the stiffness matrix is $O(Np^{2d})$. To allow for an efficient and scalable meshfree simulation the employed linear solver should have a similar complexity. Since there is no such optimal solver based on general algebraic methods, non-optimal (sparse) direct solvers are often employed in meshfree methods or generalized finite element methods [20]. Our goal is the development of an iterative solver with optimal complexity for the PUM; i.e., the number of iterations required to solve the linear system should be independent of the number of points N and the approximation order p, and the computational cost associated with a single iteration should be close to $O(Np^{2d})$.

In this paper we present a multilevel solver for partition of unity discretizations which employs a tree-based spatial multilevel construction and a domain decomposition type smoothing scheme. The respective subspace splitting is based on overlapping subspaces which contain all shape functions $\varphi_j \psi_j^m$ interacting on a given patch $\omega_i := \operatorname{supp}(\varphi_i)$. The computational complexity of a single iteration of this solver is $O(Np^{3d})$. Furthermore, the results of our numerical experiments indicate that the convergence rate is independent of N and p; i.e., the number of iterations required to solve the linear system within a given relative accuracy ϵ is independent of N and p. Therefore, the overall complexity of the solver is of the order $O(\log(1/\epsilon)Np^{3d})$ for any polynomial degree p which is optimal up to a factor of $O(p^d)$.

The remainder of this paper is organized as follows: In section 2 we shortly review the construction of PUM spaces and the Galerkin discretization of a linear elliptic PDE using our PUM. In section 3 we shortly review our tree-based multilevel cover construction and introduce a p-robust smoothing scheme based on domain decomposition ideas. Then, we present the results of our numerical experiments in section 4 which indicate that the convergence rate of our multilevel solver is independent of N and p. We consider scalar Poisson-type problems in two and three space dimensions and the system of the Navier–Lamé equations in two dimensions with up to 393216 degrees of freedom. Finally, we conclude with some remarks in section 5.

2 Partition of Unity Method

In the following, we shortly review the construction of partition of unity spaces and the meshfree Galerkin discretization of an elliptic PDE, see [7, 8, 18] for details.

2.1 Construction of Partition of Unity Spaces

In a PUM, we define a global approximation u^{PU} as a weighted sum of local approximations u_i,

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

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 $\{\psi_i^n\}$ and the order of approximation p_i for every single $u_i := \sum_n u_i^n \psi_i^n \in V_i^{p_i}$ 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 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 it still fulfills global regularity conditions. Hence, the global approximation space on Ω is defined as

$$V^{\mathrm{PU}} := \sum_i \varphi_i V_i^{p_i} = \sum_i \varphi_i \,\mathrm{span}\langle\{\psi_i^n\}\rangle = \mathrm{span}\langle\{\varphi_i \psi_i^n\}\rangle. \tag{2.2}$$

The starting point for any meshfree method is a collection of N independent points $P := \{x_i \in \mathbb{R}^d \mid x_i \in \overline{\Omega}, i = 1, \ldots, N\}$. 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 (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. Thus, the first (and most crucial) step in a PUM is the efficient construction of an appropriate cover $C_\Omega := \{\omega_i\}$. Throughout this paper we use a tree-based construction algorithm for d-rectangular covers C_Ω presented in [8, 18]. Here, the cover patches ω_i are products of intervals $(x_i^l - h_i^l, x_i^l + h_i^l)$ for $l = 1, \ldots, d$. With the help of weight functions W_k defined on these cover patches ω_k we can easily generate a partition of unity by *Shepard's method*, i.e., we define

$$\varphi_i(x) = \frac{W_i(x)}{\sum_{\omega_k \in C_\Omega^i} W_k(x)}, \tag{2.3}$$

where $C_i := \{\omega_j \in C_\Omega \,|\, \omega_i \cap \omega_j \neq \emptyset\}$ is the set of all geometric neighbors of a cover patch ω_i. Due to the use of d-rectangular patches ω_i, 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 $\operatorname{supp}(\mathcal{W}) = [0,1]$ such that $\operatorname{supp}(W_i) = \overline{\omega_i}$. It is sufficient for this construction to choose a one-dimensional weight function $\mathcal{W}$ with the desired regularity which is non-negative. The partition of unity functions φ_i inherit the regularity of the generating weight function $\mathcal{W}$.

In general, a partition of unity $\{\varphi_i\}$ can only recover the constant function on the domain Ω. Hence, 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 ψ_i^n. Since we use d-rectangular patches ω_i only, a local tensor product space is the most natural choice. Here, we use products of univariate Legendre polynomials as local approximation spaces $V_i^{p_i}$, i.e., we choose

$$V_i^{p_i} = \operatorname{span}\langle\{\psi_i^n \,|\, \psi_i^n = \prod_{l=1}^d \mathcal{L}_i^{\hat{n}_l}, \|\hat{n}\|_1 = \sum_{l=1}^d \hat{n}_l \leq p_i\}\rangle,$$

where $\hat{n}$ is the multi-index of the polynomial degrees $\hat{n}_l$ of the univariate Legendre polynomials $\mathcal{L}_i^{\hat{n}_l} : [x_i^l - h_i^l, x_i^l + h_i^l] \to \mathbb{R}$, and n is the index associated with the product function $\psi_i^n = \prod_{l=1}^d \mathcal{L}_i^{\hat{n}_l}$.

In summary, we can view the construction given above as follows

$$\begin{pmatrix} \{x_i\} \\ \mathcal{W} \\ \{p_i\} \end{pmatrix} \to \begin{pmatrix} \{\omega_i\} \\ \{W_i\} \\ \{V_i^{p_i} = \operatorname{span}\langle\psi_i^n\rangle\} \end{pmatrix} \to \begin{pmatrix} \{\varphi_i\} \\ \{V_i^{p_i}\} \end{pmatrix} \to V^{\mathrm{PU}} = \sum \varphi_i V_i^{p_i},$$

where the set of points $P = \{x_i\}$, the generating weight function $\mathcal{W}$ and the local approximation orders p_i are assumed to be given. For the approximation of vector fields we employ vector-valued shape functions; i.e., we change the definition of our local approximation spaces $V_i^{p_i} = \operatorname{span}\langle\psi_i^{n,l}\rangle = \operatorname{span}\langle\psi_i^n \boldsymbol{e}_l\rangle$ where $\boldsymbol{e}_l$ denotes an appropriate unit vector but leave the scalar partition of unity functions φ_i unchanged. Throughout this paper we use a fixed polynomial degree p on all patches ω_i, i.e, $p_i = p$ for all $i = 1, \dots, N$.

2.2 Variational Formulation and Galerkin Discretization

The imposition of essential boundary conditions within meshfree methods is more involved than in the FEM for a number of reasons and many different approaches have been proposed [18]. We use Nitsche's method [14] to enforce Dirichlet boundary conditions which leads to a non-standard weak formulation. The main advantages of this approach are that it does not require a second function (or multiplier) space and that it leads to a positive definite linear system, see [11, 18] for a more detailed discussion of Nitsche's method in the PUM context. Here, we only state the resulting weak formulation $a(u,v) = l(v)$ of the Poisson problem

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega \subset \mathbb{R}^d, \\ u &= g_D && \text{on } \Gamma_D \subset \partial\Omega, \\ u_n &= g_N && \text{on } \Gamma_N = \partial\Omega \setminus \Gamma_D, \end{aligned} \tag{2.4}$$

and for the system of the Navier–Lamé equations

$$-\mu \boldsymbol{\Delta} u - (\lambda + \mu)\nabla(\nabla \cdot u) = f \quad \text{in} \quad \Omega \subset \mathbb{R}^d, \quad d = 2,3 \tag{2.5}$$

with suitable boundary conditions $u_D = g_D$ on $\Gamma_D \subset \partial\Omega$ and $\boldsymbol{\sigma}(u) \cdot n = g_N$ on $\Gamma_N = \partial\Omega \setminus \Gamma_D$. The parameters λ and μ are the so-called Lamé parameters of the material and are related to the Poisson ratio ν and the Young modulus E via $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$.

The bilinear form $a(u,v)$ associated with (2.4) using Nitsche's method is given by

$$a(u,v) = \int_\Omega \nabla u \nabla v + \int_{\Gamma_D} u(\beta v - v_n) - u_n v$$

and the respective right-hand side is given by

$$l(v) = \int_\Omega f v + \int_{\Gamma_D} g_D(\beta v - v_n) + \int_{\Gamma_N} g_N v.$$

The subscript n denotes the normal derivative and β is the Nitsche regularization parameter. For (2.5) we obtain the bilinear form

$$a(u,v) = \int_\Omega \boldsymbol{\sigma}(u) : \boldsymbol{\epsilon}(v) + \int_{\Gamma_D} 2\mu\beta_{\boldsymbol{\epsilon}} u \cdot v + \lambda\beta_{\mathrm{div}}(u \cdot n)(v \cdot n) - \big((\boldsymbol{\sigma}(u) \cdot n) \cdot v + u \cdot (\boldsymbol{\sigma}(v) \cdot n)\big)$$

where $\beta_{\boldsymbol{\epsilon}}$ and β_{div} denote the two Nitsche regularization parameters involved, $\boldsymbol{\sigma}(u) := \lambda \nabla \cdot u I + 2\mu\, \boldsymbol{\epsilon}(u)$ is the symmetric stress tensor and $\boldsymbol{\epsilon}(u) := \frac{1}{2}(\partial_i u_j + \partial_j u_i)$ denotes the strain tensor associated with the displacement field $u = (u_i)$, $i = 1, \ldots, d$. The respective linear form $l(v)$ on the right-hand side is given by

$$l(v) = \int_\Omega f \cdot v + \int_{\Gamma_N} g_N \cdot v + \int_{\Gamma_D} 2\mu\beta_{\boldsymbol{\epsilon}} g_D \cdot v + \lambda\beta_{\mathrm{div}}(g_D \cdot n)(v \cdot n) - g_D \cdot (\boldsymbol{\sigma}(v) \cdot n).$$

Note that Nitsche's method introduces regularization parameters which depend on the employed discretization space. Hence, as we refine the discretization space, the regularization parameters and therefore the bilinear form $a(\cdot,\cdot)$ and the linear form on the right-hand side $l(\cdot)$ change. Note also that the regularization parameters can be computed automatically during a simulation without much computational cost for a specific discretization space, see [11, 18] for details.

Finally, for the Galerkin discretization of (2.4) or (2.5) we have to compute the stiffness matrix

$$A = \left(A_{(i,n),(j,m)}\right), \text{ with } A_{(i,n),(j,m)} = a\left(\varphi_j \psi_j^m, \varphi_i \psi_i^n\right),$$

and the right-hand side vector

$$\hat{f} = \left(f_{(i,n)}\right), \text{ with } f_{(i,n)} = l(\varphi_i \psi_i^n).$$

The stable approximation of the respective integrals is somewhat more involved in the PUM than in the FEM. Due to the meshfree construction given above the shape functions $\varphi_i \psi_i^n$ are piecewise rational functions only, so that the respective integrands have a number of jumps within the integration domain which need to be resolved.

To estimate the computational cost associated with the Galerkin discretization, let us first assume that the shape functions are (piecewise) polynomials like in the GFEM. Then each integral associated with a particular entry $A_{(i,n),(j,m)}$ of the stiffness matrix can be computed as the sum of integrals over the elements within $\operatorname{supp}(\varphi_i) \cap \operatorname{supp}(\varphi_j)$. Now, all integrands[3] are polynomial functions as well and the integrals can be evaluated exactly. If the shape functions are polynomials of degree p the integrals are polynomials of degree $(p+1)^2$ and can be evaluated with $O(dp^2)$ operations. Therefore, we can assemble the stiffness matrix with $O(N(dp^2 + p^{2d}))$ operations, if we evaluate all polynomials simultaneously.

In the PUM, however, the analytical integration of the integrals is in general not possible. Hence, we need to employ a numerical integration scheme. The cost $\mathcal{C}_{\mathrm{NI}}$ associated with the numerical integration of a single entry of the stiffness matrix is given by

$$\mathcal{C}_{\mathrm{NI}} = O(n_{\mathrm{IC}}\; n_{\mathrm{IN}}\; \mathcal{C}_{\mathrm{EI}})$$

where n_{IC} denotes the number of integration cells, n_{IN} the number of integration nodes per cell, and $\mathcal{C}_{\mathrm{EI}}$ the cost associated with the evaluation of the integrand.[4] In our implementation we use a subdivision sparse grid integration scheme [8, 18] where n_{IC} is essentially determined by the jumps of the derivatives of φ_i; i.e., $n_{\mathrm{IC}} = O(3^d(l+1)^d)$ is given by the order l of the employed spline weight function $\mathcal{W}$. In the following we restrict ourselves to the case of $l = 1$. Within each of these integration cells the integrands are smooth functions and the integrals can be approximated efficiently by a higher order quadrature formula. To this end, we use a sparse grid quadrature scheme [6] based on univariate Gauss–Patterson [16] rules. The number of quadrature points of such a sparse grid formula is given $n_{\mathrm{IN}} = O(2^q q^{d-1})$ where q denotes the refinement level of the employed univariate quadrature rule; i.e.,

[3] In fact, only the integrals associated with the stiffness matrix and the mass matrix are polynomials. The exact integration of the load vector is usually not possible.

[4] Note that this bound on the computational complexity does not directly involve the required quality of the numerical integration scheme. The parameters, however, must be chosen such that the numerical integration is accurate enough.

the number of quadrature points of the univariate rule is of the order $O(2^q)$. The polynomial exactness of the Gauss–Patterson rule on level q is $3 \cdot 2^{q-1} - 1$.

Let us assume that the integrands on the integration cells can be approximated accurately by a polynomial of degree $(p+1)^2$.[5] Hence, we can estimate the cost $\mathcal{C}_{\mathrm{A,NI}}$ associated with the assembly of the stiffness matrix in our implementation by

$$\mathcal{C}_{\mathrm{A,NI}} = O(Np(\ln p)^{d-1}(dp + p^d + p^{2d})).$$

Hence, our implementation is optimal up a factor of $O(p(\ln p)^{d-1})$, under the assumptions stated above. The results of our numerical experiments indicate that our numerical integration scheme gives an accurate and stable approximation of the stiffness matrix.[6]

3 Multilevel Solution of Resulting Linear System

In the following we focus on the solution of the large sparse linear system $A\tilde{u} = \hat{f}$ where $\tilde{u}$ denotes a coefficient vector and $\hat{f}$ denotes a moment vector. This solution step is a very time consuming part of any numerical simulation. The use of an inappropriate solver can drive up the compute time as well as the storage demand dramatically.

Classical direct solvers for dense matrices like Gaussian elimination or LU-decomposition have a storage requirement of $O(\mathsf{dof}^2)$ and the number of operations even scales with $O(\mathsf{dof}^3)$, where dof denotes the number of degrees of freedom. More advanced sparse direct solvers can reduce these complexities to some extent only. In the special case of regular meshes in two dimensions for instance a nested dissection solver requires $O(\mathsf{dof}^{3/2})$ operations and $O(\mathsf{dof}\ln(\mathsf{dof}))$ storage [5]. For our PUM space we have $\mathsf{dof} = O(Np^d)$ where $N = \mathrm{card}(C_\Omega)$ denotes the number of patches ω_i and p the order of approximation. The number of nonzeros entries of a PUM stiffness matrix is of the order $O(Np^{2d})$. Hence, this optimal storage and operation complexity will be lost when a direct solver is employed.

Alternatively, an iterative scheme like the Jacobi- or Gauss–Seidel method can be used. Here, we do not have a significant increase in the storage requirements, but the number of operations necessary to obtain the solution of

[5] This assumption can be justified by the structure of φ_i and our choice of α and $\mathcal{W}$.

[6] In fact, we use an adaptive version of the quadrature scheme with a dynamic stopping criterion to ensure the quality of the numerical integration also for problems with non-constant coefficients, see [8, 18]. Note that it is essential to analyze the interplay of the numerical integration error and the approximation error to be able to develop an assembly scheme for the stiffness matrix with optimal complexity and optimal approximation properties. Such an analysis would allow to determine the required tolerance of the numerical quadrature automatically and can help to minimize the computational costs associated with the assembly of the stiffness matrix.

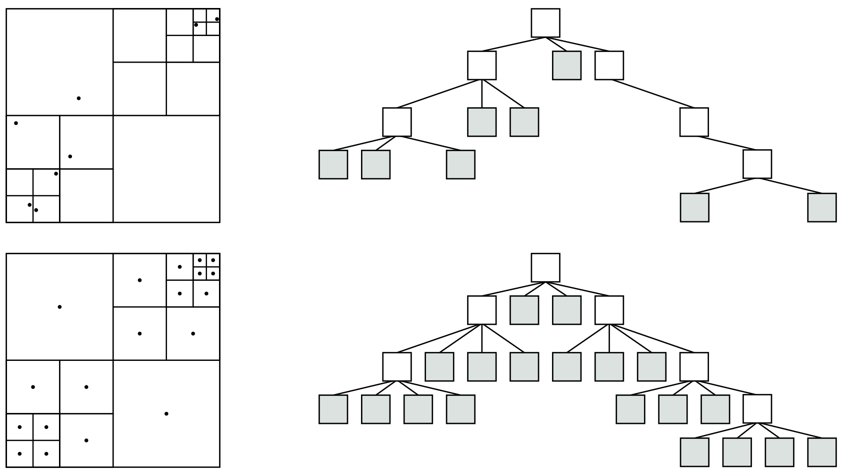

Figure 1. Hierarchical cover construction in two dimensions. The cell decomposition induced by the initial point set (upper left) and its corresponding tree representation (upper right, white: INNER tree nodes, gray shaded: LEAF tree nodes). Here, the leaves of the tree correspond to the given initial points x_i. The final cell decomposition with all generated points x_L (lower left) and its tree representation (lower right) after the completion of the cover construction. Now, the leaves of the tree correspond to the points $x_L \in P_J$.

the linear system up to a prescribed accuracy does not scale with the optimal complexity. A sophisticated class of iterative methods which not only show an optimal scaling in the storage demand but also in the operation count are so-called multilevel iterative solvers or multigrid methods [12, 21]. These solvers, however, are not general algebraic methods but involve a substantial amount of information about the discretization and possibly the PDE. We have developed a first multilevel solver for PUM discretizations in [9, 18]. The convergence rate of this solver is independent of the number of patches N, but not of the approximation order p. Now we present an extension of this multilevel solver which gives a convergence rate that is also independent of p. To this end, let us shortly review the multilevel construction from [9, 18].

The first step toward the design of an efficient multilevel solver is the construction of an appropriate sequence of function spaces. To this end, we have developed a hierarchical multilevel cover construction algorithm [8, 9, 18] which gives a sequence of point sets $P_k = \{x_{i,k}\}$ and covers $C^k_\Omega = \{\omega_{i,k}\}$. The algorithm is based on so-called d-binary trees (quadtrees, octrees) and ensures the covering property on all levels $k = 0, \dots, J$, i.e., $\bigcup_{\omega_{i,k} \in C^k_\Omega} \omega_{i,k} \supset \Omega$, see Figures 1 and 2. We define the cover patches $\omega_{i,k} = \alpha\, \mathcal{C}_{i,k}$ via a scaling of the cells $\mathcal{C}_{i,k}$ of the tree decomposition by a scalar factor $\alpha \in (1, 2)$, see Figure 3. The overlap parameter α must be larger than 1 to obtain (at least) a $\mathcal{C}^0$

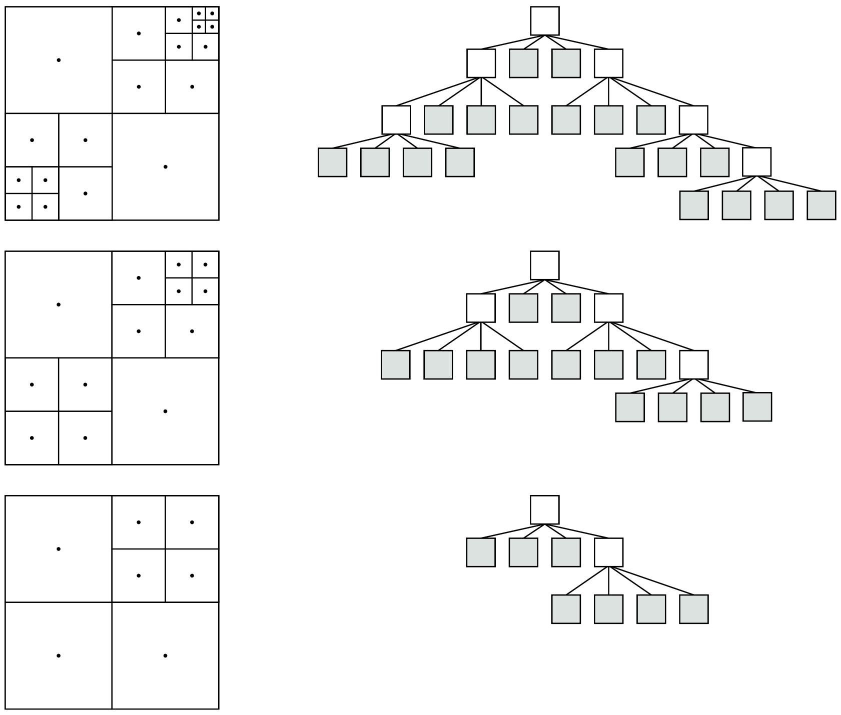

Figure 2. Multilevel cover sequence in two dimensions. The cell decompositions and its respective tree representation (upper right, white: INNER tree nodes, gray shaded: LEAF tree nodes) for the fine level point set $P_J = P_4$ (upper row), and two coarser level point sets P_3 (center row) and P_2 (lower row). The leaves of the respective tree correspond to the points $x_L \in P_k$.

PU (independent of the employed weight function $\mathcal{W}$) and should be smaller than 2 to ensure the linear independence of the resulting shape functions.

Note that the underlying tree data structure also allows for an efficient neighbor search for general point sets P_J and it reduces the computational effort associated with numerical integration, see [8, 9, 18] for details. Furthermore, it can be used for the approximation of the domain Ω, see Figure 4. Via the general PUM construction given in section 2 we then obtain the sequence of PUM function spaces V_k^{PU} associated with the sequence of covers C_Ω^k. Note that these spaces are nonnested, i.e., $V_{k-1}^{\mathrm{PU}} \not\subset V_k^{\mathrm{PU}}$, and that the shape functions $\varphi_{i,k}\psi_{i,k}^n$ are non-interpolatory. Thus, the natural injection and a direct interpolation between two successive PUM spaces V_{k-1}^{PU} and V_k^{PU} are not available. Therefore, we need to construct appropriate prolongation operators $I_{k-1}^k : V_{k-1}^{\mathrm{PU}} \to V_k^{\mathrm{PU}}$ and restriction operators $I_k^{k-1} : V_k^{\mathrm{PU}} \to V_{k-1}^{\mathrm{PU}}$ to transfer information between the PUM spaces.

To this end, we can use an L^2-projection as prolongation between two spaces V_{k-1}^{PU} and V_k^{PU}. However, a global L^2-projection is prohibitively ex-

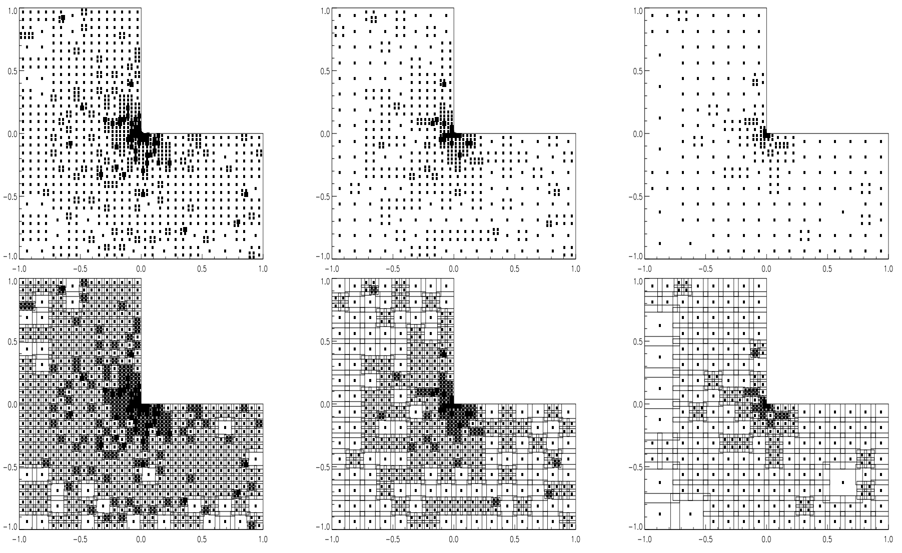

Figure 3. Point sets P_k (upper row) and covers C_Ω^k (lower row) for $k = 10, \dots, 8$ for a graded initial point set.

pensive. Yet, within the PUM context we can construct a very cheap prolongation operator using a localized L^2-projection approach. Based on the error estimates for the PUM [1, 2] and the geometric hierarchy of our tree construction, we have developed this cheap but qualitatively good projection. Here, we only give a short review over the construction principles, see [9, 18] for details.

The localization of the L^2-projection within our PUM consists of two steps. At first consider the basic PUM error estimate

$$\|v - v^{\mathrm{PU}}\|^2_{L^2(\Omega)} \le C \sum_i \|v - v_i\|^2_{L^2(\omega_i \cap \Omega)}, \tag{3.6}$$

where $v^{\mathrm{PU}} := \sum_i \varphi_i \sum_n u_i^n \psi_i^n$ and $v_i := \sum_n u_i^n \psi_i^n$. From (3.6) we know that it is sufficient to control the local errors $\|v - v_i\|_{L^2(\omega_i \cap \Omega)}$ on each cover patch ω_i. Now choose $v = u^{\mathrm{PU}}_{k-1} = \sum_j \varphi_{j,k-1} u_{j,k-1} = \sum_j \varphi_{j,k-1} \sum_m u^m_{j,k-1} \psi^m_{j,k-1}$ and $v^{\mathrm{PU}} = I^k_{k-1} u^{\mathrm{PU}}_{k-1} = \sum_i \varphi_{i,k} u_{i,k} = \sum_i \varphi_{i,k} \sum_n u^n_{i,k} \psi^n_{i,k}$ so that (3.6) reads

$$\|u^{\mathrm{PU}}_{k-1} - I^k_{k-1} u^{\mathrm{PU}}_{k-1}\|^2_{L^2(\Omega)} \le C \sum_i \|u^{\mathrm{PU}}_{k-1} - u_{i,k}\|^2_{L^2(\omega_{i,k} \cap \Omega)}. \tag{3.7}$$

Hence, we observe that we can approximate the global coarse function u^{PU}_{k-1} locally on the fine cover patches $\omega_{i,k}$ using the local basis functions $\psi^n_{i,k}$, rather than approximating u^{PU}_{k-1} by the global shape functions $\varphi_{i,k} \psi^n_{i,k}$ on the finer level k. Now in a second step we establish an upper bound for each of the

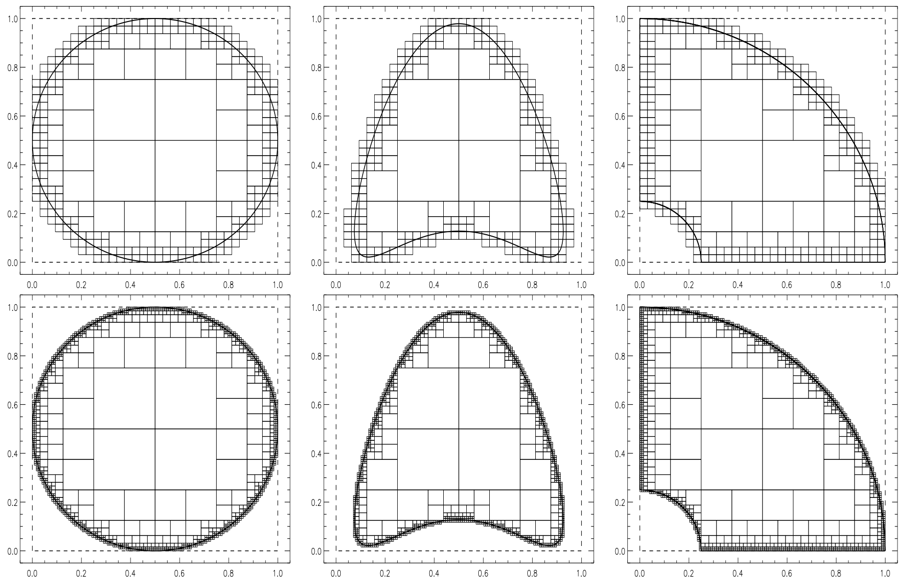

Figure 4. Tree-based approximation of a spherical domain (left), a smooth non-convex domain (center), and a quarter of a spherical domain with a spherical hole (right) on level $k = 5$ (upper row) and level $k = 7$ (lower row).

terms on the right-hand side of (3.7) utilizing the geometric hierarchy of our tree. Due to our tree-based cover construction we can find exactly one coarse patch $\omega_{\tilde{i},k-1}$ for every fine patch $\omega_{i,k}$ such that $\omega_{i,k} \subset \omega_{\tilde{i},k-1}$ holds. Hence, we can introduce the respective coarse local function $u_{\tilde{i},k-1}$ associated with the unique coarse patch $\omega_{\tilde{i},k-1}$ into each term $\|u^{\mathrm{PU}}_{k-1} - u_{i,k}\|_{L^2(\omega_{i,k}\cap\Omega)}$ of (3.7). Finally, we obtain the estimate

$$\begin{aligned}\|u^{\mathrm{PU}}_{k-1} - u_{i,k}\|_{L^2(\omega_{i,k}\cap\Omega)} \leq \|u^{\mathrm{PU}}_{k-1} - u_{\tilde{i},k-1}\|_{L^2(\omega_{i,k}\cap\Omega)} +& \\ \|u_{\tilde{i},k-1} - u_{i,k}\|_{L^2(\omega_{i,k}\cap\Omega)}&\end{aligned} \tag{3.8}$$

by the triangle inequality. This estimate allows us to approximate each coarse local function $u_{\tilde{i},k-1}$, independent of all other local components $u_{j,k-1}$ of u^{PU}_{k-1}, on the respective fine cover patch $\omega_{i,k}$ with $\omega_{i,k} \subset \omega_{\tilde{i},k-1}$ since the first term of (3.8) is small by definition of u^{PU}_{k-1}. Hence, we can set up our prolongation operators via the so-called local-to-local L^2-projection. To this end, we project each local approximation $u_{i,k-1}$ on level $k-1$ independently to the finer level k using the hierarchical condition $\omega_{i,k} \subseteq \omega_{\tilde{i},k-1}$ instead of the geometric neighbor relation $\omega_{i,k} \cap \omega_{j,k-1} \neq \emptyset$ only. The respective matrix representation of this prolongation is given by

$$\begin{aligned} I^k_{k-1} := \widetilde{\Pi}^k_{k-1} &:= (\widetilde{M}^k_k)^{-1}(\widetilde{M}^k_{k-1}) && \text{with} \\ (\widetilde{M}^k_k)_{(i,n),(i,m)} &:= \langle \psi^m_{i,k}, \psi^n_{i,k} \rangle_{L^2(\omega_{i,k}\cap\Omega)} && \text{and} \\ (\widetilde{M}^k_{k-1})_{(i,n),(\tilde{i},m)} &:= \langle \psi^m_{\tilde{i},k-1}, \psi^n_{i,k} \rangle_{L^2(\omega_{i,k}\cap\Omega)}. \end{aligned}$$

The storage requirement of this local-to-local projection is minimal. We need to store only a single block-entry $(\widetilde{M}^k_k)^{-1}_{i,i}(\widetilde{M}^k_{k-1})_{i,\tilde{i}}$ for each patch $\omega_{i,k}$ on level k. Moreover, the respective integrals involve only the local basis functions $\psi^m_{\tilde{i},k-1}$ and $\psi^n_{i,k}$ and can be computed very efficiently. Overall, the local-to-local projection operator can be computed with $O(Np^{3d})$ operations in general (and with $O(Np^d)$ if we use orthogonal polynomials locally). Furthermore, it is exact for polynomials of degree p and therefore suitable also for higher order approximations.

In summary, we now have a sequence of stiffness matrices A_k coming from the direct Galerkin approximation of the respective bilinear form $a_k(\cdot,\cdot)$ on each level k and a sequence of high-quality transfer operators I^k_{k-1}, I^{k-1}_k based on localized L^2-projections. As the final ingredient for our multilevel solver, see Algorithm 3.1, we now need to construct a sequence of appropriate smoothing operators S_k.

Algorithm 3.1 (Multilevel Algorithm $M^{\nu_1,\nu_2}_\gamma(k, x_k, b_k)$).

```
1. If $k > 0$:
   a) For $l = 1, \dots, \nu_1$: Set $x_k = S^{\mathrm{pre}}_k(x_k, b_k)$.
   b) Set $d_{k-1} := I^{k-1}_k(b_k - A_k x_k)$.
   c) Set $e_{k-1} := 0$.
   d) For $i = 1, \dots, \gamma$: $e_{k-1} = M^{\nu_1,\nu_2}_\gamma(k-1, e_{k-1}, d_{k-1})$.
   e) Set $x_k = C_k(x_k, e_{k-1}) := x_k + I^k_{k-1} e_{k-1}$.
   f) For $l = 1, \dots, \nu_2$: Set $x_k = S^{\mathrm{post}}_k(x_k, b_k)$.
2. Else:
   a) Set $x_k = A^{-1}_k b_k$.
```

Many iterative solvers such as the classical Jacobi- and Gauss–Seidel iterations, the overlapping domain decomposition methods and even multigrid methods can be interpreted in the framework of subspace correction methods (SCM) [3, 4, 13, 15, 21, 22]. Hence, let us shortly review the abstract setting of an SCM.

The general idea is as follows: First, we write the discretization space $\mathcal{V} = \sum_{j=1}^{\mathcal{N}} \mathcal{V}_j$ as the sum[7] of subspaces $\mathcal{V}_j$ with maps $P_j : \mathcal{V}_j \to \mathcal{V}$.[8] Then, we choose symmetric positive definite bilinear forms $b_j(\cdot,\cdot)$ on each $\mathcal{V}_j$ represented by operators B_j such that solutions to the systems of linear equations $B_j u_j = f_j$ on $\mathcal{V}_j$ are easily computable, and B_j^{-1} can be considered as an approximate inverse to the restriction of A to $\mathcal{V}_j$. Finally, we combine these local

[7] Note that we do not assume that the splitting is a direct sum.

[8] Actually, it is sufficient to require $\mathcal{V} = \sum_j P_j \mathcal{V}_j$, i.e., the condition $\mathcal{V}_j \subset \mathcal{V}$ is not necessary.

approximate inverses B_j^{-1} appropriately to define a global approximate inverse to A on the discretization space $\mathcal{V}$. There are essentially two approaches to the definition of an approximate inverse of A by the B_j^{-1}, the additive approach and the multiplicative approach.

In the so-called parallel subspace correction (PSC) or additive Schwarz method we set up an iterative solution process via the operator

$$M_{\mathrm{PSC}} := \mathbb{I} - \omega \sum_{j=1}^{\mathcal{N}} P_j T_j = \mathbb{I} - \omega \Big(\sum_{j=1}^{\mathcal{N}} P_j B_j^{-1} R_j \Big) A, \tag{3.9}$$

where ω is a relaxation parameter and the involved operators are defined by

$$\begin{aligned} a(u,v) = \langle Au, v\rangle_{\mathcal{V}}, \quad & b_j(u_j, v_j) = \langle B_j u_j, v_j\rangle_{\mathcal{V}_j}, \\ \langle R_j u, v_j\rangle_{\mathcal{V}} = \langle u, P_j v_j\rangle_{\mathcal{V}}, \; & b_j(T_j u, v_j) = a(u, P_j v_j). \end{aligned}$$

The iteration operator of the successive subspace correction (SSC) or multiplicative Schwarz method is given by

$$M_{\mathrm{SSC}} := \prod_{j=1}^{\mathcal{N}} \big(\mathbb{I} - P_j T_j \big) = \prod_{j=1}^{\mathcal{N}} \big(\mathbb{I} - P_j B_j^{-1} R_j A \big). \tag{3.10}$$

Note that the PSC operator (3.9) can also be interpreted as a preconditioned Richardson iteration where the preconditioner is given by

$$\mathcal{C}_{PSC} := \sum_{j=1}^{\mathcal{N}} P_j B_j^{-1} R_j. \tag{3.11}$$

Let us now restrict ourselves to the case of $B_j := A|_{\mathcal{V}_j}$ which means that we consider exact subspace solvers only. Then, we have essentially two degrees of freedom in the design of our smoothing scheme: The splitting of the discretization space and the type of the iteration, namely the additive scheme (3.9) or the multiplicative scheme (3.10). To define an appropriate splitting of our PUM space V^{PU} (we omit the level index k in the following), let us consider the specific structure of the PUM shape functions. The product structure of the shape functions $\varphi_i \psi_i^n$ implies two natural subspace definitions. For instance, we can define the subspaces $\mathcal{V}_n := \mathrm{span}_i \langle \varphi_i \psi_i^n \rangle := \{ v \in V^{\mathrm{PU}} \,|\, v = \sum_i \varphi_i v_i^n \psi_i^n \}$. These subspaces, however, contain functions with *global* support on the domain Ω, see Figure 5 (left), and the dimension of each subspace is of the order $O(N)$. Therefore, a direct solution of $A|_{\mathcal{V}_n}$ is not feasible. We would need to resort to fast iterative solution techniques for these subspace problems. Furthermore, we are interested in smoothing schemes S_k for Algorithm 3.1 based on our spatial multilevel construction. Hence, there is no additional benefit from the fact that the solutions to $A|_{\mathcal{V}_n}$ contain global information and the computational cost associated with the solution of the subspace problems make this splitting unsuitable for our construction.

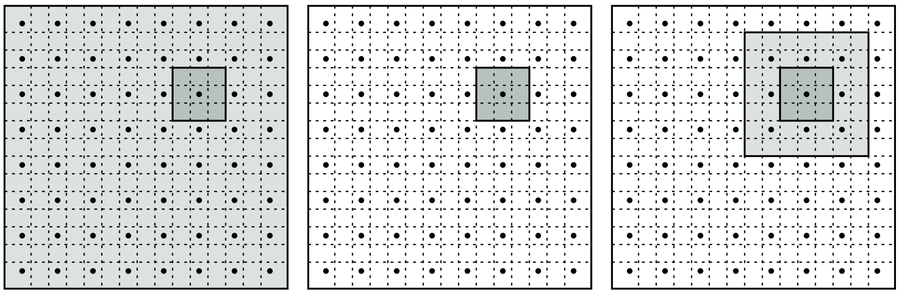
Figure 5. Subdomains (light gray shaded) associated with the subspaces $\mathcal{V}_n$ (left), $\mathcal{V}_i$ (center), and $\widetilde{\mathcal{V}}_l$ (right) and the support of the a single shape function $\varphi_i \psi_i^n$ (dark gray shaded) based on a cover with $\alpha = 1.5$.

A more appropriate subspace definition is given by $\mathcal{V}_i := \varphi_i V_i^p = \mathrm{span}_n \langle \varphi_i \psi_i^n \rangle := \{v \in V^{\mathrm{PU}} \mid v = \sum_n \varphi_i v_i^n \psi_i^n\}$. These spaces contain functions with local supports only, see Figure 5 (center). Furthermore, the dimension of the subspace $\mathcal{V}_i$ is given by the dimension $O(p^d)$ of the local approximation spaces V_i^p. Hence, we can compute the inverse $(A|_{\mathcal{V}_i})^{-1}$ of each of the subspace problems with acceptable complexity of $O(p^{3d})$; i.e., one iteration of a PSC or SSC iteration based on this splitting is of the order $O(Np^{3d})$.

Note that both subspace definitions lead to a *direct* splitting of our PUM function space $V^{\mathrm{PU}} = \sum_i \mathcal{V}_i = \sum_n \mathcal{V}_n$; i.e., every basis function $\varphi_i \psi_i^n$ is contained in exactly one subspace. In terms of the index pairs (i, n) we have a disjoint decomposition of the index set $\{(i, n)\}$ which induces a specific partitioning of the PUM stiffness matrix $A = (A_{(i,n),(j,m)})$. Using the subspaces $\mathcal{V}_i$ we obtain the so-called polynomial block-form. Here, a single block $A_{i,j} = (A_{(i,n),(j,m)})$ corresponds to a local discretization of the PDE on the domain $\omega_i \cap \omega_j \cap \Omega$.

Note that a PSC iteration (3.9) based on the direct splitting $V^{\mathrm{PU}} = \sum_i \mathcal{V}_i$ corresponds to the classical block-Jacobi iteration and the SSC iteration (3.10) corresponds to the block-Gauss–Seidel iteration (*BGS*) where we have only a small overlap between the supports of functions from different subspaces, see Figure 5 (center). Even though we consider a direct splitting and employ an exact solver $(A|_{\mathcal{V}_i})^{-1}$ within a specific subspace $\mathcal{V}_i$ there are still couplings between the subspaces due to the overlap of the supports via the global problem A. The quality of the PSC and SSC iterations is obviously determined by the strength of these couplings. The two parameters within our PUM which can influence the strength of the couplings between two different subspaces $\mathcal{V}_i$ and $\mathcal{V}_j$, and hence the quality of the iterations, are the overlap parameter α used in our cover construction and the polynomial degree p. Since we are interested in a smoothing scheme that is truly robust, i.e., that works with the same quality at least for a large range of parameters α and p, this direct splitting approach cannot be pursued.

One approach to overcome this problem is to consider subspace splittings $V^{\mathrm{PU}} = \sum_l \widetilde{\mathcal{V}}_l$ which are no longer direct splittings, i.e., a basis function $\varphi_i\psi_i^n$ may now belong to several subspaces $\widetilde{\mathcal{V}}_l$. Consider the subspace definition

$$\widetilde{\mathcal{V}}_l := \sum_{\omega_i\cap\omega_l\neq\emptyset} \mathcal{V}_i = \mathrm{span}_{(i,n),i\in C_l}\langle\varphi_i\psi_i^n\rangle \tag{3.12}$$

where $C_l := \{i\,|\,\omega_i \cap \omega_l \neq \emptyset\}$ denotes the neighborhood of the cover patch ω_l, see Figure 5 (right). The subspace $\widetilde{\mathcal{V}}_l$ contains *all* functions $\varphi_i\psi_i^n$ whose support ω_i has a non-vanishing intersection with the patch ω_l. Hence, when we solve the subspace problem $A|_{\widetilde{\mathcal{V}}_l}$ we resolve all couplings involving the basis functions $\varphi_l\psi_l^q$. Thus, for each patch ω_l there is one subspace problem $A|_{\widetilde{\mathcal{V}}_l}$ which resolves *all* couplings involving the associated basis functions $\varphi_l\psi_l^q$ independent of the overlap parameter α and the polynomial degree p. Note that this splitting is similar to the one employed in [17].

Since the subspace splitting into the $\widetilde{\mathcal{V}}_l$ is not a direct splitting it does not correspond to a simple partitioning scheme of the stiffness matrix A. Here, we rather have to assemble the (discrete) local subproblems $A_{l,l}$ from the global linear system via the so-called Galerkin products $A_{l,l} := P_l^T A P_l$ where P_l denotes the discrete extension operator which embeds the subspace $\widetilde{\mathcal{V}}_l$ in the global PUM space V^{PU}. In our case P_l is just a mask matrix, i.e., a reduced identity matrix. With the matrices A and P_l the application of the SSC iteration operator (3.10) to a linear system $A\tilde{u} = \hat{f}$ can be realized by Algorithm 3.2. In the following we refer to this iteration as a multiplicative overlapping Schwarz (*MOS*) smoother.

Algorithm 3.2 (Successive subspace correction method).

```
1. For all $l = 1, \dots, N$:
    a) Compute local residual $\hat{f}_l := P_l^T(\hat{f} - A\tilde{u})$.
    b) Solve subspace problem $(P_l^T A\ P_l)\tilde{u}_l = A_{l,l}\tilde{u}_l = \hat{f}_l$.
    c) Update global iterate $\tilde{u} = \tilde{u} + P_l\tilde{u}_l$.
```

In Figure 6 we give the smoothing results obtained after one iteration of the *BGS* and the *MOS* smoother for $p = 1$ and $p = 5$. From these surface plots we can clearly observe that the *MOS* smoother gives much smoother iterates than the *BGS* smoother. More notably, the quality of the *BGS* smoother deteriorates for higher order approximations. The results for $p = 5$ are not as smooth as for $p = 1$. For the *MOS* smoother we find a completely different behavior. There is no deterioration in the quality for larger p. In fact it even seems that the results for $p = 5$ are better than for $p = 1$.

Note that Algorithm 3.1 corresponds to an SSC iteration based on a multilevel subspace splitting. Similarly, we can define a PSC type multilevel iteration and the associated preconditioner (3.11). For instance if apply (3.9) not only the sum of all local subspaces (3.12) on a particular level k but rather to the sum of all local subspaces $\widetilde{\mathcal{V}}_{l,k}$ on all levels k, i.e., $\sum_k\sum_l \widetilde{\mathcal{V}}_{l,k}$, together

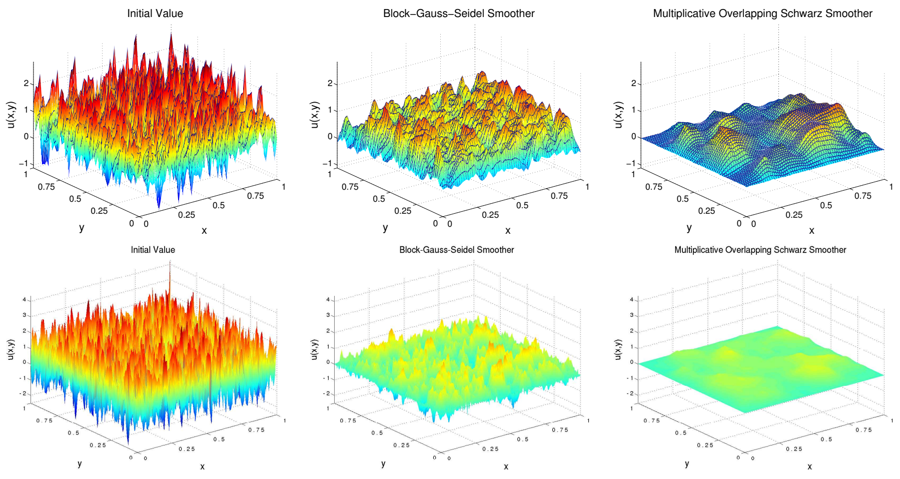

Figure 6. Random valued initial guess (left) and smoothing results using a block-Gauss–Seidel (center) and a multiplicative overlapping Schwarz smoother (right). Depicted are the current iterates after a single application of the smoother. The discretization was based on a uniform node arrangement on level 5 and employed polynomials of degree $p = 1$ (upper row) and $p = 5$ (lower row).

with the mask matrices $P_{l,k}$, the prolongations I_{k-1}^{k} and the restrictions I_{k}^{k-1}, we obtain a multilevel PSC iteration. The application of the associated multilevel additive Schwarz ($MAOS$) preconditioner (3.11) can be realized with Algorithm 3.3.

ALGORITHM 3.3 (Multilevel parallel subspace correction preconditioner).

```
1. Set initial value $\tilde{u}_J = 0$.
2. For all levels $k = J, \dots, 1$:
   a) Set right-hand side $\hat{f}_{k-1} = I_k^{k-1} \hat{f}_k$.
   b) Set initial value $\tilde{u}_{k-1} = 0$.
3. For all levels $k = 0, \dots, J$:
   a) For all $l = 1, \dots, N_k$:
      i. Set local right-hand side $\hat{f}_{k,l} := P_{k,l}^T(\hat{f}_k)$.
      ii. Solve subspace problem $(P_{l,k}^T A_k P_{l,k})\tilde{u}_{l,k} = (A_k)_{l,l}\tilde{u}_{k,l} = \hat{f}_{k,l}$.
      iii. Update global iterate $\tilde{u}_k = \tilde{u}_k + P_{l,k}\tilde{u}_{l,k}$.
4. For all levels $k = 1, \dots, J$:
   a) Update global iterate $\tilde{u}_k = \tilde{u}_k + I_{k-1}^k \tilde{u}_{k-1}$.
```

In many cases the convergence behavior of the PSC schemes and the respective SSC schemes are similar. Yet, the PSC methods have certain advantages with respect to parallelization.

4 Numerical Results

In all our experiments with our multilevel solver (see Algorithm 3.1) and the multilevel preconditioner (see Algorithm 3.3) we solve a linear system of the form

$$A\tilde{u} = \hat{f} = 0.$$

We choose a vanishing right-hand side vector $\hat{f} = 0$ and a random initial guess u_0^{PU} with $\|u_0^{\mathrm{PU}}\|_{L^2} \approx 1$. Here, we approximate the L^2-norm of the function u_0^{PU} by

$$\|u_0^{\mathrm{PU}}\|_{L^2}^2 \approx \tilde{u}_0^T M \tilde{u}_0 =: \|\tilde{u}_0\|_{L^2} = 1,$$

where $\tilde{u}_0$ denotes the random valued coefficient vector associated with the function u_0^{PU} and $M := \langle \varphi_j \psi_j^m, \varphi_i \psi_i^n \rangle_{L^2}$ is the mass matrix. The stopping criterion for the iteration is $\|\tilde{u}_r\|_{L^2} < 10^{-13}$ (or $r = 50$ for Algorithm 3.1 and $r = 250$ for Algorithm 3.3) where r denotes the number of iterations. Hence, the average convergence or error reduction rate of our multilevel iteration with respect to the L^2-norm[9] of the error is given by

$$\rho_{L^2} := \left(\frac{\tilde{u}_r^T M \tilde{u}_r}{\tilde{u}_0^T M \tilde{u}_0} \right)^{\frac{1}{r}} = \left(\frac{\|\tilde{u}_r\|_{L^2}}{\|\tilde{u}_0\|_{L^2}} \right)^{\frac{1}{r}} = \|\tilde{u}_r\|_{L^2}^{\frac{1}{r}}.$$

Furthermore, we also compute the more common convergence rates

$$\rho_{l^2} := \left(\frac{\|\tilde{u}_r\|_{l^2}}{\|\tilde{u}_0\|_{l^2}} \right)^{\frac{1}{r}} \quad \text{and} \quad \rho_R := \left(\frac{\|A\tilde{u}_r\|_{l^2}}{\|A\tilde{u}_0\|_{l^2}} \right)^{\frac{1}{r}}$$

which are based on the l^2-norm of the current coefficient vector $\tilde{u}_r$ and the respective residual vector $A\tilde{u}_r$. These convergence rates are given for multilevel iterations $M_1^{1,1}$, the so-called $V(1,1)$-cycle, based on the local-to-local L^2-projection using the non-overlapping BGS smoother and the overlapping MOS smoother. The quality of both these SSC smoothers is dependent on the ordering of the respective subspaces, i.e., of the cover patches. In all our experiments we use a space filling curve ordering scheme based on the Hilbert curve for the cover patches, see [9, 18] for details. Besides the convergence rates ρ^{BGS} and ρ^{MOS} we also give the finest discretization level J, the polynomial degree p of the local approximation spaces V_i^p, the respective number of degrees of freedom dof, and the number of nonzeros entries nnz of the stiffness matrix A_J on the finest level. In all experiments, we used a linear B-spline as the generating weight function $\mathcal{W}$, a uniform node arrangement as initial point set, and Legendre polynomials up to degree 9 as local approximation spaces in our PUM discretization.

[9] Note that even for uniform covers C_Ω we have no uniform correspondence (with respect to the number of degrees of freedom) between the L^2-norm of a function u^{PU} and the l^2-norm of its corresponding coefficient vector $\tilde{u}$ due to the use of local polynomials, just like in the p-version of the finite element method.

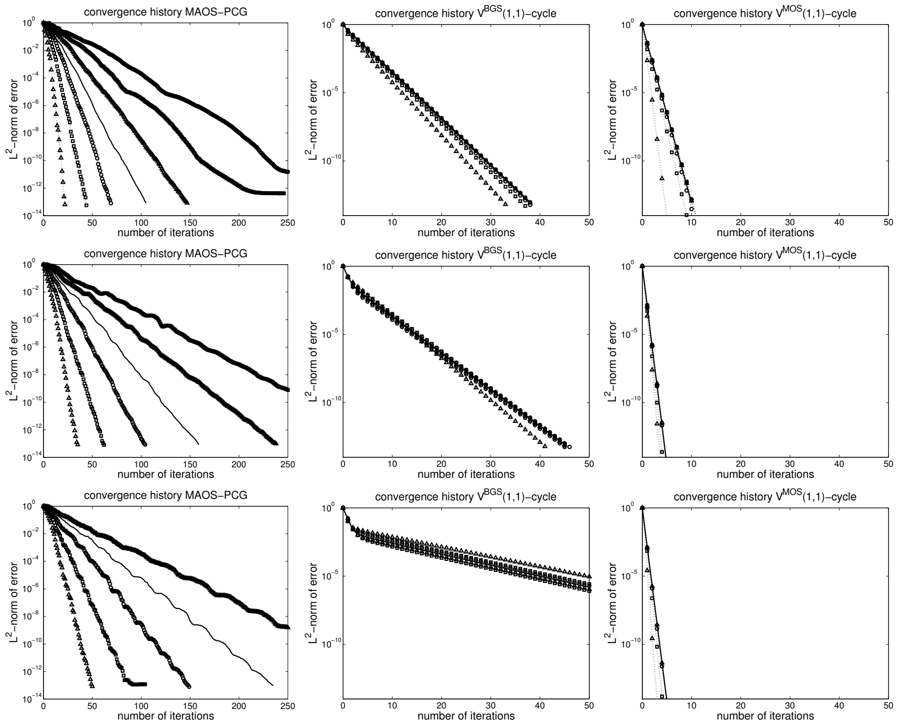

Figure 7. Convergence history for a preconditioned conjugate gradient solver with the *MAOS* preconditioner (left), the $V(1,1)$-cycle using the *BGS* smoother (center) and the *MOS* smoother (right), for the Poisson problem (4.13) in two dimensions. Depicted are the results for $p = 1$ on levels $2, \dots, 8$ (top row), $p = 3$ on levels $2, \dots, 7$ (center row), and $p = 5$ on levels $2, \dots, 6$ (bottom row).

In our first experiment, we considered the Poisson problem

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = [0,1]^d \subset \mathbb{R}^d, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{4.13}$$

in two and three dimensions with $f = 0$ and Dirichlet boundary conditions $g = 0$. The results of this experiment are summarized in Figure 7 and in Tables 1 and 2. From the plots depicted in Figure 7 we clearly observe that the convergence behavior of a conjugate gradient method preconditioned with the *MAOS* scheme (Algorithm 3.3, Figure 7 (left)) is dependent on the number of points N as well as the polynomial degree p. The multilevel iteration $M_1^{1,1}$ (Algorithm 3.1), i.e., the $V(1,1)$-cycle, converges independent of the number of points N for the *BGS* smoother (Figure 7 (center)) but not independent of the polynomial degree p. Only the $V(1,1)$-cycle with the *MOS* smoother (Figure 7 (right)) shows a convergence behavior that is independent of the number of points N and the polynomial degree p. Hence, in contrast to FEM

Table 1. Convergence rates ρ^{BGS} obtained for a $V(1,1)$-cycle using the BGS smoother, and the rates ρ^{MOS} using the overlapping MOS smoother for the Poisson problem (4.13) in two dimensions.

J	p	dof	nnz	$\rho^{BGS}_{L^2}$	$\rho^{BGS}_{l^2}$	ρ^{BGS}_{R}	$\rho^{MOS}_{L^2}$	$\rho^{MOS}_{l^2}$	ρ^{MOS}_{R}
8	1	196608	5280804	0.261	0.256	0.220	0.039	0.037	0.020
8	2	393216	21123216	0.166	0.160	0.150	0.002	0.001	0.001
7	3	163840	14592400	0.324	0.337	0.322	0.001	0.001	0.001
6	4	61440	8122500	0.580	0.607	0.581	0.002	0.001	0.001
6	5	86016	15920100	0.692	0.736	0.695	0.001	0.001	0.001
5	6	28672	6927424	0.791	0.868	0.754	0.002	0.004	0.001
5	7	36864	11451456	0.811	0.892	0.781	0.001	0.003	0.001
5	8	46080	17892900	0.829	0.910	0.813	0.002	0.002	0.001
4	9	14080	6400900	0.850	0.926	0.830	0.001	0.002	0.001

Table 2. Convergence rates ρ^{BGS} obtained for a $V(1,1)$-cycle using the BGS smoother, and the rates ρ^{MOS} using the overlapping MOS smoother for the Poisson problem (4.13) in three dimensions.

J	p	dof	nnz	$\rho^{BGS}_{L^2}$	$\rho^{BGS}_{l^2}$	ρ^{BGS}_{R}	$\rho^{MOS}_{L^2}$	$\rho^{MOS}_{l^2}$	ρ^{MOS}_{R}
5	1	131072	13289344	0.319	0.312	0.266	0.034	0.032	0.020
4	2	40960	9733600	0.192	0.182	0.163	0.002	0.002	0.001
3	3	10240	4259200	0.443	0.460	0.430	< 0.001	< 0.001	< 0.001
3	4	17920	13043800	0.657	0.689	0.637	< 0.001	< 0.001	< 0.001

the additive Schwarz or PSC preconditioner (Algorithm 3.3) does not yield a similar convergence behavior as the multiplicative Schwarz or SSC iteration (Algorithm 3.1).

The rates ρ^{BGS}, see Tables 1 and 2, measured for the $V(1,1)$-cycle with the BGS smoother are clearly p-dependent and deteriorate with increasing p. We measure $\rho^{BGS}_{L^2} = 0.261$ for $p = 1$ and $\rho^{BGS}_{L^2} = 0.850$ for $p = 9$. The rates ρ^{MOS} obtained with the MOS smoother, however, are almost constant for all polynomial degrees p. Furthermore, these rates are very small. We find $\rho^{MOS}_{L^2}$ to be no worse than 0.002 for $p \geq 2$; i.e., our solver converges up to machine accuracy in less than 6 iterations. Note that due to storage limitations not all experiments with increasing p could be carried out up to the same discretization level J. Here, we were limited by the number of nonzeros nnz of the stiffness matrix A_J on the finest level J. From the numbers given in Tables 1 and 2 we see that nnz on the finest level J is similar for all experiments and ranges from 5 million to 21 million entries.

In our second experiment we considered the Navier–Lamé equations

$$\begin{aligned} -\mu\boldsymbol{\Delta} u - (\lambda+\mu)\nabla(\nabla\cdot u) &= f \quad \text{in} \quad \Omega = [0,1]^2 \subset \mathbb{R}^2, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{4.14}$$

in two dimensions with $f = 0$ and Dirichlet boundary conditions $g = 0$. The measured convergence rates ρ for our multilevel solver with the BGS smoother

Table 3. Convergence rates ρ^{BGS} obtained for a $V(1,1)$-cycle using the BGS smoother, and the rates ρ^{MOS} using the overlapping MOS smoother for the elasticity problem (4.14) in two dimensions.

J	p	dof	nnz	$\rho^{BGS}_{L^2}$	$\rho^{BGS}_{l^2}$	ρ^{BGS}_{R}	$\rho^{MOS}_{L^2}$	$\rho^{MOS}_{l^2}$	ρ^{MOS}_{R}
8	1	393216	21123216	0.664	0.658	0.552	0.128	0.124	0.074
7	2	196608	21013056	0.347	0.336	0.277	0.008	0.007	0.003
6	3	81920	14440000	0.671	0.689	0.655	0.010	0.011	0.009
5	4	30720	7952400	0.786	0.813	0.764	0.002	0.001	< 0.001
5	5	43008	15586704	0.791	0.831	0.778	0.002	0.002	< 0.001
4	6	14336	6635776	0.827	0.882	0.805	0.002	0.003	0.001
4	7	18432	10969344	0.857	0.908	0.831	0.002	0.002	0.001
3	8	5760	3920400	0.871	0.927	0.836	0.001	0.001	< 0.001

and the MOS smoother are given in Table 3. Again we observe that the rates ρ^{BGS} for the BGS smoother deteriorate for larger p. The rates ρ^{MOS} obtained for the MOS smoother on the other hand are very small and stay constant e.g. $\rho^{MOS}_{L^2} \approx 0.002$ for increasing $p \geq 2$.

In summary, the results of our numerical experiments indicate that only the multilevel solver (Algorithm 3.1) with the overlapping MOS smoother (Algorithm 3.2) converges independent of N and p. Furthermore, the measured convergence rates ρ^{MOS} are very small. It took no more than 6 iterations for the solver to converge up to machine accuracy for $p \geq 2$.

Note, however, that the computational costs associated with the overlapping smoothing scheme involve a rather large and d-dependent constant. Due to the overlap of the subspace splitting, the local problems are of dimension $O(3^d p^d)$ whereas the dimension of the local problems without overlap is $O(p^d)$, see Figure 5 (center) and (right). Since these local problems lead to dense matrices in both cases, the storage requirement of the overlapping MOS smoother is larger by a factor of 3^{2d} than the storage demand of the non-overlapping BGS smoother. Similarly, the number of operations required by the MOS smoother is larger by a factor of 3^{3d} due to the direct solution of the larger local problems. Hence, the overall compute time of the multilevel solver with the non-optimal BGS smoother may be significantly smaller in the practical range of N and p. Furthermore, the multilevel solver with the BGS smoother is applicable to larger problems due to its smaller storage demand.

5 Concluding Remarks

We presented a multilevel solver for PUM discretizations which employs a tree-based spatial multilevel sequence in conjunction with an overlapping domain decomposition type smoothing scheme. The results of our numerical experiments indicate that the convergence rate ρ of this multilevel solver is independent of the number of points N and the approximation order p; i.e., the solver is robust with respect to p. The overall computational complexity of our solver is of the order $O(Np^{3d})$ to reduce the initial error by a prescribed factor which is optimal up to a factor of $O(p^d)$. Note, however, that the constants involved are rather large.

These results hold also for irregular point distributions and different cycle types. For instance, also the $V(1,0)$-cycle with the MOS smoother based on a graded point set converges independent of N and p. Furthermore, the rate of the multilevel solver with the MOS smoother seems to robust also for general domains, i.e., when the domain is not resolved on the coarsest level.

Acknowledgement. Finally, we would like to acknowledge the support of the Sonderforschungsbereich 611 „Singuläre Phänomene und Skalierung in mathematischen Modellen“ funded by the Deutsche Forschungsgemeinschaft.

References

1. I. Babuška and J. M. Melenk. The Partition of Unity Finite Element Method: Basic Theory and Applications. *Comput. Meth. Appl. Mech. Engrg.*, 139:289–314, 1996. Special Issue on Meshless Methods.
2. I. Babuška and J. M. Melenk. The Partition of Unity Method. *Int. J. Numer. Meth. Engrg.*, 40:727–758, 1997.
3. J. H. Bramble and X. Zhang. Handbook of Numerical Analysis. In P. G. Ciarlet and J. L. Lions, editors, *The Analysis of Multigrid Methods*, volume VII, pages 173–416. Elsevier, 2000.
4. W. Dahmen. Multiscale Analysis, Approximation, and Interpolation Spaces. In C. K. Chui and L. L. Schumaker, editors, *Approximation Theory VIII*, volume 2, pages 47–88. World Scientific, 1995.
5. J. A. George. Nested Dissection of a Regular Finite Element Mesh. *SIAM J. Num. Anal.*, 10:345–363, 1973.
6. T. Gerstner and M. Griebel. Numerical Integration using Sparse Grids. *Numer. Alg.*, 18:209–232, 1998.
7. M. Griebel and M. A. Schweitzer. A Particle-Partition of Unity Method for the Solution of Elliptic, Parabolic and Hyperbolic PDE. *SIAM J. Sci. Comput.*, 22(3):853–890, 2000.
8. M. Griebel and M. A. Schweitzer. A Particle-Partition of Unity Method—Part II: Efficient Cover Construction and Reliable Integration. *SIAM J. Sci. Comput.*, 23(5):1655–1682, 2002.
9. M. Griebel and M. A. Schweitzer. A Particle-Partition of Unity Method—Part III: A Multilevel Solver. *SIAM J. Sci. Comput.*, 24(2):377–409, 2002.
10. M. Griebel and M. A. Schweitzer. A Particle-Partition of Unity Method—Part IV: Parallelization. In M. Griebel and M. A. Schweitzer, editors, *Meshfree Methods for Partial Differential Equations*, volume 26 of *Lecture Notes in Computational Science and Engineering*, pages 161–192. Springer, 2002.
11. M. Griebel and M. A. Schweitzer. A Particle-Partition of Unity Method—Part V: Boundary Conditions. In S. Hildebrandt and H. Karcher, editors, *Geometric Analysis and Nonlinear Partial Differential Equations*, pages 517–540. Springer, 2002.

12. W. Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer, 1985.
13. W. Hackbusch. *Iterative Solution of Large Sparse Linear Systems of Equations.* Springer, 1994.
14. J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abh. Math. Sem. Univ. Hamburg*, 36:9–15, 1970–1971.
15. P. Oswald. *Multilevel Finite Element Approximation.* Teubner Skripten zur Numerik. Teubner, 1994.
16. T. N. L. Patterson. The Optimum Addition of Points to Quadrature Formulae. *Math. Comp.*, 22:847–856, 1968.
17. L. F. Pavarino. Additive Schwarz Methods for the p-Version Finite Element Method. *Numer. Math.*, 66:493–515, 1994.
18. M. A. Schweitzer. *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*, volume 29 of *Lecture Notes in Computational Science and Engineering.* Springer, 2003.
19. T. Strouboulis, I. Babuška, and K. Copps. The Design and Analysis of the Generalized Finite Element Method. *Comput. Meth. Appl. Mech. Engrg.*, 181(I 3):43–69, 2000.
20. T. Strouboulis, K. Copps, and I. Babuška. The Generalized Finite Element Method. *Comput. Meth. Appl. Mech. Engrg.*, 190:4081–4193, 2001.
21. J. Xu. Iterative Methods by Space Decomposition and Subspace Correction. *SIAM Review*, 34(4):581–613, 1992.
22. H. Yserentant. Old and New Convergence Proofs for Multigrid Methods. *Acta Numerica 93*, pages 285–326, 1993.