1

# Treatment of general domains in two space dimensions in a Partition of Unity Method

Marc Alexander Schweitzer[*] and Maharavo Randrianarivony[†]

Institut für Numerische Simulation, Universität Bonn, Wegelerstr. 6, D-53115 Bonn, Germany
schweitzer@ins.uni-bonn.de, randrian@ins.uni-bonn.de

**Summary.** This paper is concerned with the approximate solution of partial differential equations using meshfree Galerkin methods on arbitrary domains in two space dimensions which we assume to be given as CAD/IGES data. In particular we focus on the particle-partition of unity method (PPUM) yet the presented technique is applicable to most meshfree Galerkin methods. The basic geometric operation employed in our cut-cell type approach is the intersection of a planar NURBS patch and an axis-aligned rectangle. Note that our emphasis is on the speed of the clipping operations since these are invoked frequently while trying to attain a small number of patches for the representation of the intersection. We present some first numerical results of the presented approach.

**Key words:** meshfree method, partition of unity method, complex domain

## 1.1 Introduction

One major advantage of meshfree methods (MM) over traditional mesh-based approximation approaches is that the challenging task of mesh generation can (in principle) be avoided. Thus the numerical treatment of partial differential equations (PDE) on complex time-dependent domains can be simplified substantially by MM.

Collocation techniques for instance are essentially independent of the computational domain. They employ a collection of points for the discretization process only and require no explicit access to the domain or the boundary. In meshfree Galerkin methods however we have to integrate the respective bilinearform and linearform over the domain $\Omega$ (and parts of the boundary $\partial\Omega$).

[*] This work was supported in part by the Sonderforschungsbereich 611 *Singular phenomena and scaling in mathematical models* funded by the *Deutsche Forschungsgemeinschaft*.

[†] Hausdorff Center for Mathematics

Hence, we must be concerned with the issues of a meshfree domain representation and the efficient numerical integration of the meshfree shape functions over $\Omega$ and $\partial\Omega$. These issues often lead to the perception that a background mesh is needed in meshfree Galerkin methods. This is in fact not the case, we must rather compute an appropriate decomposition of the domain into pairwise disjoint cells which respect the algebraic structure of the employed shape functions as well as the geometry of the domain. Thus, we must be able to compute this decomposition efficiently *on the fly* since the construction of the meshfree shape functions is independent of the domain and the domain may change during the simulation.

In the following we present a two step procedure for the efficient computation of a decomposition in two space dimensions for the particle–partition of unity method (PPUM) [9, 10, 20] where we assume the domain to be given as CAD/IGES data. Note that our approach is easily extendable to other meshfree Galerkin methods.

The remainder of this paper is organized as follows: In section 1.2 we give a short review of the construction of meshfree shape functions in the PPUM which is essentially independent of the domain $\Omega$. Moreover we introduce the employed weak formulation and an initial decomposition which respects the algebraic structure of the constructed shape functions and covers the domain and its boundary with pairwise disjoint cells. The main contribution of this paper, the treatment of general domains in two space dimensions, is discussed in section 1.3. Here, we introduce the fundamental IGES entities used in our implementation for the domain representation and present an efficient cell-clipping approach for the computation of the intersection of an axis-aligned rectangle with a NURBS patch. First numerical results are reported in section 1.4 before we conclude with some remarks in section 1.5.

## 1.2 Particle–Partition of Unity Method

In this section let us shortly review the core ingredients of the PPUM, see [9, 10, 20] for details. In a first step, we need to construct a PPUM space $V^{\mathrm{PU}}$, i.e., we need to specify the PPUM shape functions $\varphi_i \vartheta_i^n$. With these shape functions, we then set up a sparse linear system of equations $A\tilde{u} = \hat{f}$ via the classical Galerkin method where we realize essential boundary conditions via Nitsche's method [16]. The arising linear system is then solved by our multilevel iterative solver [10].

An arbitrary function $u^{\mathrm{PU}} \in V^{\mathrm{PU}}$ is defined as the linear combination

$$u^{\mathrm{PU}}(x) = \sum_{i=1}^{N} \varphi_i(x) u_i(x) \quad \text{with} \quad u_i(x) = \sum_{m=1}^{d_i} u_i^m \vartheta_i^m(x) \tag{1.1}$$

and the respective PPUM space $V^{\mathrm{PU}}$ is defined as

$$V^{\mathrm{PU}} := \sum_{i=1}^{N} \varphi_i V_i \quad \text{with} \quad V_i := \mathrm{span}\langle \vartheta_i^m \rangle. \tag{1.2}$$

Here, we assume that the functions $\varphi_i$ form an admissible partition of unity (PU) on the domain $\Omega$; i.e., the supports $\omega_i := \mathrm{supp}(\varphi_i)$ cover the complete domain $\Omega$ and its boundary $\partial\Omega$, and refer to the spaces $V_i$ with $\dim(V_i) = d_i$ as local approximation spaces. Hence, the shape functions employed in the PPUM are the products $\varphi_i \vartheta_i^m$ of a PU function $\varphi_i$ and a local basis function $\vartheta_i^m$.

This abstract approach is the basis of any partition of unity method [2,3] such as e.g. the generalized/extended finite element method (GFEM/XFEM) [15,25–27]. The key difference between our PPUM and the GFEM/XFEM is that the PU in GFEM/XFEM is usually constructed by classical FE shape functions $\phi_i$ based on some kind of mesh which may also encode the (discrete) computational domain $\Omega$ or may simply cover $\Omega$ In the PPUM, however, the PU is constructed from independent point data only; i.e. it is always independent of the representation of the computational domain $\Omega$.

The fundamental construction principle employed in [9] for the construction of the PU $\{\varphi_i\}$ is a $d$-binary tree. Based on the given point data $P = \{x_i \,|\, i = 1, \ldots, \hat{N}\}$, we sub-divide a bounding-box $\mathcal{C}_\Omega \supset \Omega$ of the domain $\Omega$ until each cell

$$\mathcal{C}_i = \prod_{l=1}^{d} (c_i^l - h_i^l, c_i^l + h_i^l)$$

associated with a leaf of the tree contains at most a single point $x_i \in P$, see Figure 1.1. We obtain an overlapping cover $C_\Omega := \{\omega_i\}$ from this tree by defining the cover patches $\omega_i$ by

$$\omega_i := \prod_{l=1}^{d} (c_i^l - \alpha h_i^l, c_i^l + \alpha h_i^l), \quad \text{with } \alpha > 1. \tag{1.3}$$

Note that we define a cover patch $\omega_i$ for leaf-cells $\mathcal{C}_i$ that contain a point $x_i \in P$ as well as for *empty* cells that do not contain any point from $P$.

To obtain a PU on a cover $C_\Omega$ with $N := \mathrm{card}(C_\Omega)$ we define a weight function $W_i : \Omega \to \mathbb{R}$ with $\mathrm{supp}(W_i) = \omega_i$ for each cover patch $\omega_i$ by

$$W_i(x) = \begin{cases} \mathcal{W} \circ T_i(x) & x \in \omega_i \\ 0 & \text{else} \end{cases} \tag{1.4}$$

with the affine transforms $T_i : \overline{\omega}_i \to [-1,1]^d$ and $\mathcal{W} : [-1,1]^d \to \mathbb{R}$ the reference $d$-linear B-spline. By simple averaging of these weight functions we obtain the Shepard functions

$$\varphi_i(x) := \frac{W_i(x)}{S(x)}, \quad \text{with} \quad S(x) := \sum_{l=1}^{N} W_l(x). \tag{1.5}$$

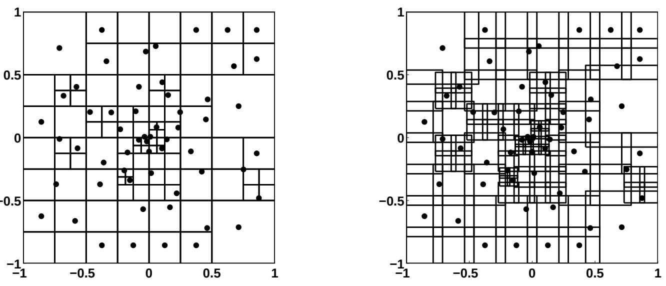

**Fig. 1.1.** Point cloud and induced tree decomposition (left) and the resulting overlapping patches (1.3) with $\alpha = 1.3$ (right).

We refer to the collection $\{\varphi_i\}$ with $i = 1, \dots, N$ as an admissible partition of unity since there hold the relations

$$\begin{aligned} 0 \ \leq \ \varphi_i(x) \leq 1, \qquad\qquad & \sum_{i=1}^{N} \varphi_i \equiv 1 \text{ on } \overline{\Omega}, \\ \|\varphi_i\|_{L^\infty(\mathbb{R}^d)} \leq C_\infty, \qquad & \|\nabla \varphi_i\|_{L^\infty(\mathbb{R}^d)} \leq \frac{C_\nabla}{\operatorname{diam}(\omega_i)} \end{aligned} \tag{1.6}$$

with constants $0 < C_\infty < 1$ and $C_\nabla > 0$ so that the assumptions on the PU for the error analysis given in [3] are satisfied by our PPUM construction.

Furthermore, the PU (1.5) based on the cover $C_\Omega$ obtained from the scaling of a tree decomposition with (a particular choice of) $\alpha \in (1, 2)$ satisfies

$$\mu(\{x \in \omega_i \,|\, \varphi_i(x) = 1\}) \approx \mu(\omega_i),$$

i.e., the PU has the flat-top property, see [12, 23]. This ensures that the product functions $\varphi_i \vartheta_i^n$ are linearly independent and stable, provided that the employed local approximation functions $\vartheta_i^n$ are stable with respect to $\{x \in \omega_i \,|\, \varphi_i(x) = 1\}$ (and $\omega_i$) [22].

With the help of the shape functions $\varphi_i \vartheta_i^n$ we then discretize a PDE in weak form

$$a(u, v) = \langle f, v \rangle$$

via the classical Galerkin method to obtain a discrete linear system of equations $A\tilde{u} = \hat{f}$. Since our general PPUM shape functions do not satisfy essential boundary conditions by construction we employ Nitsche's method for their implementation.[3] This approach, for instance, yields the bilinear form

$$a_\beta(u, v) := \int_\Omega \nabla u \cdot \nabla v \; dx - \int_{\partial\Omega} \big((\partial_n u)v + u(\partial_n v)\big) \; ds + \beta \int_{\partial\Omega} uv \; ds \tag{1.7}$$

and the associated linear form

[3] Note, however, that there is also a conforming treatment of boundary conditions for the PPUM [24].

$$\langle l_\beta, v\rangle := \int_\Omega fv \; dx - \int_{\partial\Omega} g(\partial_n v) \; ds + \beta \int_{\partial\Omega} gv \; ds \tag{1.8}$$

for the Poisson model problem

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega \subset \mathbb{R}^D, \\ u &= g \text{ on } \partial\Omega, \end{aligned} \tag{1.9}$$

where the regularization parameter $\beta$ is used to enforce the definiteness of the bilinear form (1.7) with respect to the employed finite dimensional discretization space and can be computed automatically, see [11, 20, 21] for details.

### 1.2.1 Numerical Integration

Note that the PU functions (1.5) in the PPUM are in general piecewise rational functions only. Therefore, the use of an appropriate numerical integration scheme is indispensable in the PPUM as in most meshfree Galerkin approaches [1, 4, 5, 8, 10].

In the FEM the (numerical) integration of the weak form of the considered PDE is simpler than in meshfree methods due to the availability of a mesh. This fact often leads to the perception that a mesh is required for numerical integration also in meshfree methods and that therefore meshfree Galerkin methods are not truly meshfree. However, a mesh is not required for the reliable and stable numerical integration of the weak form. We only need an appropriate decomposition of the integration domain into cells with pairwise disjoint interiors (a far more general partitioning of the domain than a mesh). Observe for instance that we can in principle allow for hanging nodes of arbitrary order in the union of these cells—a property that is usually not acceptable for FE meshes to ensure inter-element continuity of the shape functions. Thus, our construction is a much simpler task than full-blown mesh-generation.

Recall that the global regularity of our PPUM shape functions $\varphi_i \vartheta_i^m \in V^{\mathrm{PU}}$ is dominated by the regularity of the PU functions $\varphi_i$ of (1.5), i.e.

$$\varphi_i(x) := \frac{W_i(x)}{S(x)}, \qquad \text{with} \qquad S(x) := \sum_{l=1}^{N} W_l(x).$$

Thus, let us first focus on the PU functions $\varphi_i$ and how we can construct a decomposition $\{T_\alpha\}$ of its support $\omega_i$ such that $\varphi_j|_{T_\alpha}$ is smooth (i.e. of arbitrary regularity) for all $\omega_j \in C_i := \{\omega_l \in C_\Omega \,|\, \omega_i \cap \omega_l \neq \emptyset\}$. To this end, we consider a patch $\omega_i \cap \partial\Omega = \emptyset$ and carry out the differentiation in (1.7) and (1.8) and introduce the shorthand notation $G_i := \nabla W_i S - W_i \nabla S$, $\Omega_i := \Omega \cap \omega_i$, $\Omega_{i,j} := \Omega_i \cap \omega_j$, and $\Gamma_{i,j} := \partial\Omega \cap \overline{\omega}_i \cap \overline{\omega}_j$ to obtain the integrals

$$a(\varphi_j\vartheta_j^m,\varphi_i\vartheta_i^n) = \int_{\Omega_{i,j}} S^{-4}G_iG_j\ \vartheta_i^n\vartheta_j^m\ dx + \int_{\Omega_{i,j}} S^{-2}W_iW_j\ \nabla\vartheta_i^n\nabla\vartheta_j^m\ dx + \int_{\Omega_{i,j}} S^{-3}\Big(G_iW_j\ \vartheta_i^n\nabla\vartheta_j^m + W_iG_j\ \nabla\vartheta_i^n\vartheta_j^m\Big)\ dx \tag{1.10}$$

for the stiffness matrix and the integrals

$$\langle f,\varphi_i\vartheta_i^n\rangle_{L^2} = \int_{\Omega_i} S^{-1}W_i\vartheta_i^n f\ dx \tag{1.11}$$

for the right-hand side. Thus, we need to assess the regularity of the functions $S$, $W_j$ and $G_j$ for all $\omega_j \in C_i$ to construct a decomposition $\{T_\alpha\}$ of the domain, i.e. of each patch $\omega_i$, such that their restrictions to the interiors of the cells $T_\alpha$ are smooth functions. In essence this means that all weight functions $W_j$ must be smooth on the cells $T_\alpha$. Our weight functions $W_j$ however are (tensor products of) splines and therefore only piecewise smooth functions. Hence, our decomposition $\{T_\alpha\}$ must resolve the overlapping supports of weight functions as well as the piecewise character of the weight functions.

Recall that we obtained the patches $\omega_i$ of our cover $C_\Omega$ from a tree-decomposition of a bounding box of the domain $\Omega$. Thus, there is an initial pairwise disjoint decomposition $\{\mathcal{C}_i\}_{i=1}^N$ which covers the domain, i.e. $\overline{\Omega} \subset \bigcup_{i=1}^N \overline{\mathcal{C}_i}$, available. Hence, we must only be concerned with the (minimal) refinement of the cells $\mathcal{C}_i$ such that the restrictions of all weight functions $W_j$ (and local approximation functions $\vartheta_j^n$) are arbitrarily smooth on the refined cells $T_{i,\alpha}$. Observe that this refined decomposition is easily computable on the fly since we must consider the intersections of axis-aligned rectangles only due to our construction. Thus, we obtain a decomposition into axis-aligned rectangular cells.

First we split a cell $\mathcal{C}_i$ into disjoint rectangular sub-cells according to its intersections with $\omega_j \in C_i$, then we consider the piecewise character of the respective spline weight functions $W_j$ to define the decomposition $\{T_{i,\alpha}\}$, see Figures 1.2 and 1.3.[4] Thus, all PU functions $\varphi_j$ satisfy $\varphi_j|_{T_{i,\alpha}} \in C^\infty(T_{i,\alpha})$.

*Remark 1.* This decomposition is sufficient if we employ polynomial local approximation space $V_j = \mathcal{P}^{p_j}$ in our PPUM only. It only remains to select an integration rule on the cells $T_{i,\alpha}$ considering the bilinear form and the maximal polynomial degree $p_i + p_j$ of the integrands.

In the case of a non-polynomial enrichment, i.e. $V_j = \mathcal{P}^{p_j} + \mathcal{E}_j$, we must also consider the discontinuities and singularities of the enrichment functions either by the choice of appropriate integration rules or additional refinement of the decomposition $\{T_{i,\alpha}\}$.

*Remark 2.* Note that the construction above is suitable for all domain integrals that involve the assembled shape functions $\varphi_i\vartheta_i^m$ for a specific choice of

[4] Observe however that for the flat-top region of our PU functions it is sufficient to construct a single cell $T_{i,\alpha}$.

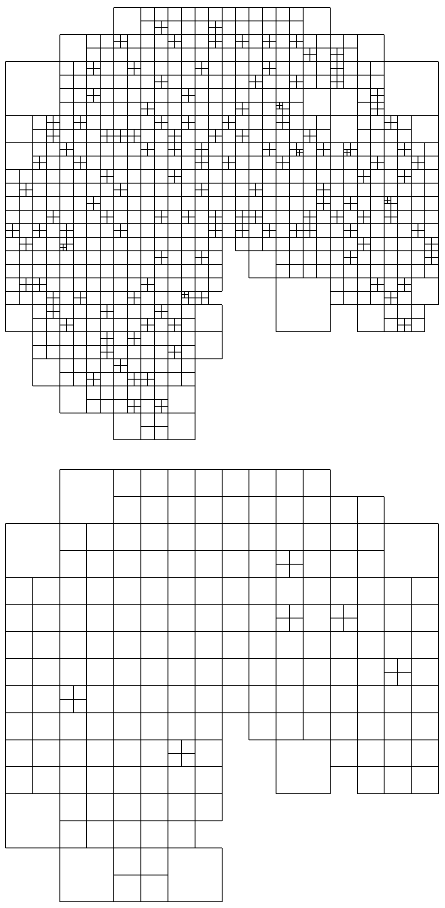

**Fig. 1.2.** Initial decomposition based on the tree-cells $\mathcal{C}_j$ only (top: level 7, bottom: level 5). The respective tree decomposition was generated by sampling with Halton points.

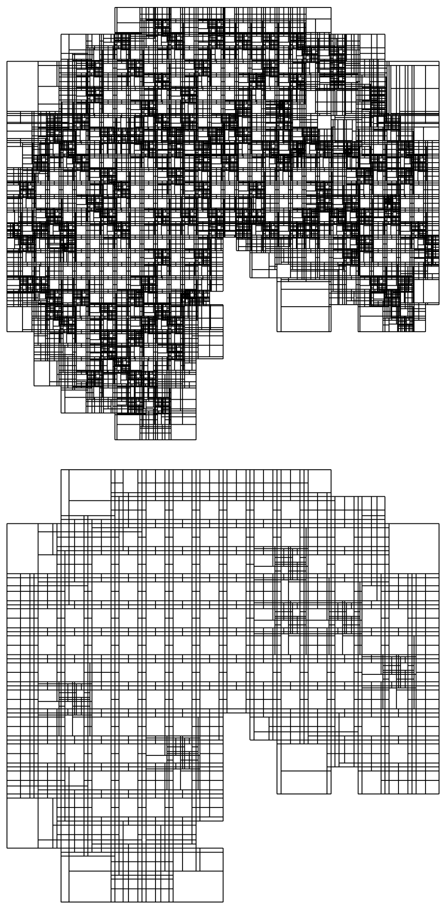

**Fig. 1.3.** Refined decomposition (top: level 7, bottom: level 5).

$\alpha$ in (1.3) and weight function $\mathcal{W}$. In the multilevel PPUM however we may also need to compute certain operators locally; i.e. just using the local approximation functions $\vartheta_i^m$ on $\omega_i$ or on the subset $\omega_{\mathrm{FT},i} := \{x \in \omega_i \,|\, \varphi_i(x) = 1\}$, see [10, 22]. These local operators can be integrated with much less integration cells since they are independent of the PU functions $\varphi_i$. An appropriate decomposition for these local integrals can be obtained easily with a variant of the above construction where we consider only the overlapping patches but not the weight functions.

The rectangular cells $T_{i,\alpha}$ obviously cover the complete domain $\Omega$ and its boundary $\partial\Omega$, however, they are not aligned with $\partial\Omega$ since our PPUM construction is completely independent of the geometry. For the definition of a refined decomposition $\{T_{i,\alpha,\partial\Omega}\}$ with boundary aligned integration cells $T_{i,\alpha,\partial\Omega}$ from the above decomposition $\{T_{i,\alpha}\}$ we must now be concerned with the geometry of the domain and its representation in our meshfree PPUM implementation.

## 1.3 Realization on General Domains

In this section we are concerned with the efficient application of the PPUM approach on general multiply connected domains $\Omega$ in two space dimensions. To this end, we will assume that the domain $\Omega$ is given as a CAD object described by a collection of IGES entities and our PPUM discretization process will essentially operate directly on this description of $\Omega$. The fundamental geometric operation needed for this approach is the efficient clipping of the domain against an axis-aligned rectangle. Recall however that we will compute the integration cells on the fly thus this operation must be rather fast. Therefore, we employ a two-step procedure: First we perform a setup step where we decompose the general multiply connected domain $\Omega$ into a collection of convex quadrilateral NURBS patches $\{\mathcal{P}\}$. Note that this setup step is independent of the discretization process and can be pre-computed. Based on this collection of NURBS patches $\{\mathcal{P}\}$ we can now compute the intersection of an arbitrary axis-aligned rectangle $\mathcal{R}$ with the domain $\Omega$ simply via the intersections of $\mathcal{R}$ with the NURBS patches $\{\mathcal{P}\}$; an operation which is substantially faster than directly computing the intersection $\mathcal{R} \cap \Omega$.

### 1.3.1 Domain Representation

Let us first summarize the employed geometry representation in two dimensions and its CAD digitization using the IGES format. Here, the fundamental objects are B-spline and NURBS curves. In order to introduce the B-spline basis, we consider any constant integer $k \geq 2$ which controls the smoothness of the spline and a knot sequence $\zeta_0, ..., \zeta_{n+k}$ such that $\zeta_{i+k} \geq \zeta_i$. The usual definition of B-spline basis functions [14,17] with respect to the knot sequence $(\zeta_i)_i$ is

$$N_i^k(t) = (\zeta_{i+k} - \zeta_i)[\zeta_i, ..., \zeta_{i+k}](\cdot - t)_+^{k-1} \tag{1.1}$$

where$[\zeta_i, ..., \zeta_{i+k}]$ denotes the forward divided difference and $(\cdot - t)_+^k$ is the truncated power function

$$(x-t)_+^k := \begin{cases} (x-t)^k & \text{if} \quad x \geq t, \\ 0 & \text{if} \quad x < t. \end{cases} \tag{1.2}$$

By induction, one can show that the above definition is equivalent to

$$N_i^1(t) := \begin{cases} 1 & \text{if} \quad t \in [\zeta_i, \zeta_{i+1}), \\ 0 & \text{otherwise}, \end{cases} \tag{1.3}$$

$$N_i^k(t) := \left( \frac{t - \zeta_i}{\zeta_{i+k-1} - \zeta_i} \right) N_i^{k-1}(t) + \left( \frac{\zeta_{i+k} - t}{\zeta_{i+k} - \zeta_{i+1}} \right) N_{i+1}^{k-1}(t). \tag{1.4}$$

A B-spline curve $f$ with control points $\mathbf{d}_i \in \mathbb{R}^2$ with respect to the above knot sequence is defined as

$$f(t) = \sum_{i=0}^{n} \mathbf{d}_i N_i^k(t) \text{ for all } t \in [\zeta_0, \zeta_{n+k}]. \tag{1.5}$$

To ensure that the B-spline curve $f$ is *open*, we assume that the knot sequence is *clamped*; i.e. there holds

$$\zeta_0 = \cdots = \zeta_{k-1} < \zeta_k \leq \zeta_{k+1} \leq \cdots \leq \zeta_n < \zeta_{n+1} = \cdots = \zeta_{n+k}. \tag{1.6}$$

The above assumption (1.6) ensures that the initial and final control points are interpolated such that the curve begins and ends at the control points. The case of rational splines or NURBS is given as

$$f(t) = \frac{\sum_{i=0}^{n} w_i \mathbf{d}_i N_i^k(t)}{\sum_{i=0}^{n} w_i N_i^k(t)} \text{ for all } t \in [\zeta_0, \zeta_{n+k}] \tag{1.7}$$

and the weights $\{w_i\}$ are assumed to be in $]0, 1]$.

As a CAD input for the PPUM implementation, we accept a multiply connected domain $\Omega$. To this end, let us assume that there are univariate smooth functions $\boldsymbol{\kappa}_i^j$ defined on $[e_i^j, f_i^j] \subset \mathbb{R}$ with $\mathcal{B}_j^i = \boldsymbol{\kappa}_i^j([e_i^j, f_i^j])$ which encode the boundary $\partial\Omega$; i.e. there holds

$$\partial\Omega = \bigcup_{i=0}^{N} \bigcup_{j=0}^{n_i} \mathcal{B}_j^i. \tag{1.8}$$

Moreover, we make the convention that the external boundary

$$\Gamma_0 := \bigcup_{j=0}^{n_0} \mathcal{B}_j^0$$

is traversed in counter clockwise direction and the internal boundaries $\Gamma_p := \bigcup_{j=0}^{n_p} \mathcal{B}_j^p$ for $p = 1, ..., N$ are traversed in clockwise direction, compare Figure 4(a). A realistic example of the above description is shown in Figure 4(b) where the control polygons are identified by the dashed lines. The above representation are practically realized with the help of the IGES format. It is a CAD standard written in structured records specified as IGES entities which are stored in five sections. Note that we have restricted ourselves to IGES 144 where the most important geometric items are summarized in Table 1.1 since a complete IGES implementation is rather cumbersome and usually not

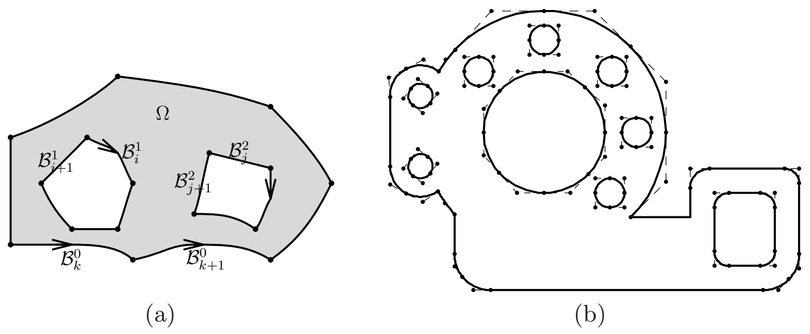

**Fig. 1.4.** (a) The boundary of $\Omega \subset \mathbb{R}^2$ is composed of the images $\mathcal{B}^i_j$ of several curves $\boldsymbol{\kappa}^j_i$. (b) A realistic CAD model where the control points are identified by large dots.

| IGES Entities | ID numbers | IGES-codes |
|---|---|---|
| Line | 110 | `LINE` |
| Circular arc | 100 | `ARC` |
| Polynomial/rational B-spline curve | 126 | `B_SPLINE` |
| Composite curve | 102 | `CCURVE` |
| Polynomial/rational B-spline surface | 128 | `SPLSURF` |
| Transformation matrix | 124 | `XFORM` |

**Table 1.1.** Appropriate IGES entities for 2D curved multiply connected domains.

necessary. Moreover, we will describe our approach assuming that all $\boldsymbol{\kappa}^j_i$ are B-spline or NURBS curves, since all practical curves including circular arcs and lines can be represented as such.

Our geometric objective consists of the following two tasks when we are given a rectangle $\mathcal{R}$. First, we need to find the intersection $\mathcal{I}$ of $\mathcal{R}$ with the multiply connected domain $\Omega$. Second, if that intersection $\mathcal{I}$ is not empty, then we decompose it into several four-sided patches $\pi_i$ and we find a mapping from the unit square to each $\pi_i$. In addition to those two points, we need that those operations are very efficient and robust because they need to be applied very often for the PPUM application. Since clipping a rectangle against a NURBS patch is easier than clipping against the whole domain $\Omega$, we adopt the following two-stage approach. First, we determine the intersection of the domain $\Omega$ with a a coarse subdivision $G = \cup_i R_i$ of a bounding box of $\Omega$ as illustrated in Figure 1.6 (compare section 1.2). That is, we intersect each rectangle $R_i$ of $G$ against the domain $\Omega$. We represent that intersection as a union of NURBS patches $\mathcal{P}^i_j$ such that $\mathcal{D}_i := R_i \cap \Omega = \cup_j \mathcal{P}^i_j$ and $\Omega = \cup_i \mathcal{D}_i = \cup_{i,j} \mathcal{P}^i_j$. Second, upon availability of the results of this setup phase using $G$, the clipping of any rectangle $\mathcal{R}$ against $\Omega$ amounts to the clipping of $\mathcal{R}$ against $R_i \cap \Omega$, thus against the relevant $\mathcal{P}^i_j$. Performing the setup phase has several

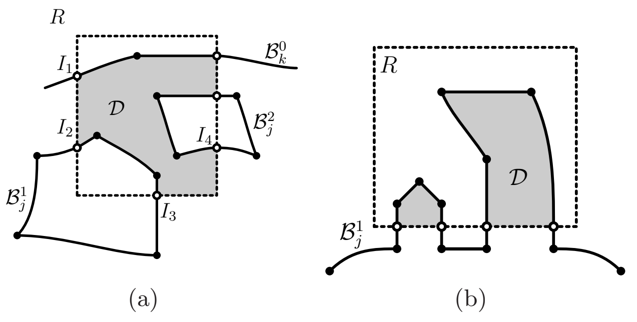

**Fig. 1.5.** Clipped region: (a) $\mathcal{D}$ is simply connected, (b) $\mathcal{D}$ is disconnected and has several connected components, (c) $\mathcal{D}$ is multiply connected.

advantages. First, it serves as a coarsest level in the PPUM method. Second, it gives a fast location to determine which rectangles $R_i$ are relevant to make each clipping fast.

### 1.3.2 Clipping a curved multiply connected domain

Let us suppose that we have a rectangle $R \in G$ which intersects the domain $\Omega$. We want to briefly discuss how to identify the boundary of the intersection $\mathcal{D} = R \cap \Omega$. Several situations regarding the connectivity of $\mathcal{D}$ may be encountered. First, $\mathcal{D}$ can be simply connected as illustrated in Figure 5(a) where the shaded region defines the clipped domain $\mathcal{D}$. On the other hand, it may be disconnected. Note that this case can occur even if the original domain $\Omega$ is simply connected as displayed in Figure 5(b). Moreover, the clipped region $\mathcal{D}$ may contain some holes and is therefore multiply connected. Combinations of these situations can also be encountered. That is, $\mathcal{D}$ has several connected components and some of them are multiply connected.

The determination of the clipped region $\mathcal{D}$ is as follows. The first step consists of finding the boundary curves $\Gamma_{s_1},...,\Gamma_{s_N}$ which intersect $R$. Then, we must identify the corresponding intersection points $I_p$ as illustrated in Figure 1.5 to which we assign an additional marker indicating whether the respective curve is *entering* or *leaving* the clipping rectangle $R \in G$ through the point $I_p$. With this information at hand, we start from any intersection point e.g. $I_1$ and we distinguish two cases. First, if that intersection point is of type *leaving*, we traverse the boundary of the rectangle $R \in G$ counter clockwise until another intersection point e.g. $I_2$ is met. In the case that $I_1$ is of *entering* type in which we suppose that $I_1$ is the intersection of $\Gamma_{s_1}$ and $R$, we traverse $\Gamma_{s_1}$ according to its original orientation. That is to say, the traversal is counter clockwise if $\Gamma_{s_1}$ encodes an external boundary whereas it is in clockwise direction if $\Gamma_{s_1}$ is an internal boundary of $\Omega$. Again, this traversal is done until we meet another intersection point. We repeat this process until

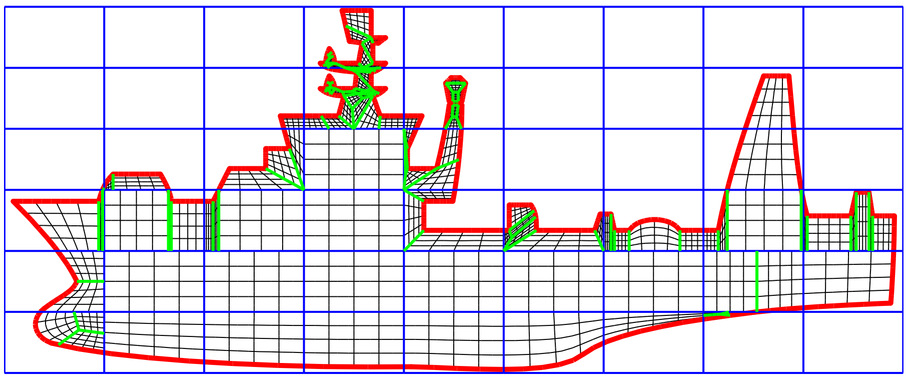

**Fig. 1.6.** Setup phase on the coarsest level: intersection against a coarse decomposition $G$ in form of several NURBS patches.

we return to the initial intersection point $I_1$. At this stage, we have generated one connected component of $\mathcal{D}$.

If all intersection points have been traversed already, the intersection is completely determined and we terminate. Otherwise, we remove those intersection points which have been traversed and we repeat the same procedure based on the remaining intersection points in order to find the other connected components of $\mathcal{D}$. After all intersection points have been dealt with, we have constructed a collection of simply connected components of $\mathcal{D}$. If the original domain $\Omega$ contains some holes, we need to perform a few additional steps. For each internal curve $\Gamma_p$ of $\Omega$, we test if it is completely located inside the rectangle $R$. If so, we test further whether $\Gamma_p$ is inside one connected component of $\mathcal{D}$ and we insert it there in the positive case. After those steps, we obtain the correct intersection by the union of several possibly multiply connected components of $\mathcal{D}$.

The above description requires the process of intersection between a NURBS curve $\mathcal{C}$ and a rectangle (see Figure 7(a)) which we briefly summarize now. Obviously, this task can be reduced to intersecting an infinite line $\mathcal{L}$ and a curve $\mathcal{C}$. Without loss of generality, we suppose that the line $\mathcal{L}$ is horizontal. We denote by $\mathbb{H}^+$ (resp. $\mathbb{H}^-$) the half-plane having positive (resp. negative) ordinates. The search for the intersections consists in examining the position of the control points with respect to $\mathbb{H}^+$ and $\mathbb{H}^-$. If the first and the last control points of $\mathcal{C}$ are located on different half-planes, then there is surely an intersection. If all control points are in one half-plane, no intersection point exists. Note that it is possible that there are some control points on both half-planes while the curve is completely inside one half-plane. To treat that ambiguous case, we apply a subdivision which is the process of splitting a NURBS curve $\mathcal{C}$ at a parameter value $t_0$ so as to obtain two curves which are again described in NURBS representation. One way of doing this is by means of discrete B-splines [6]. If the knot sequence of the original curve is defined

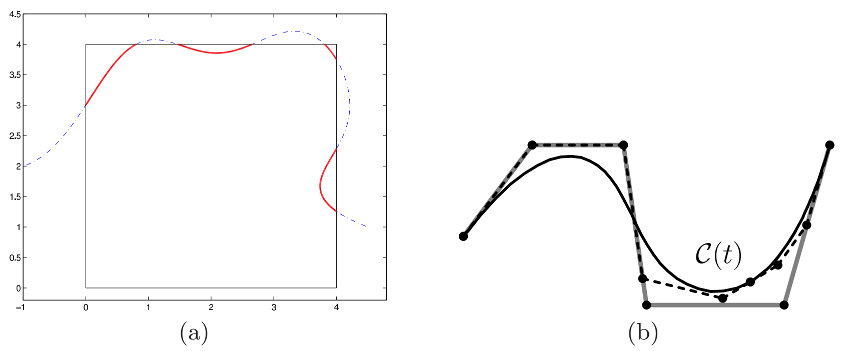

**Fig. 1.7.** (a) Fast clipping operation which works even in the case where only a little part is inside or outside the rectangle. (b) A B-spline curve split into two B-spline curves.

in $[a, b]$, then those of the resulting curves are respectively in $[a, t_0]$ and $[t_0, b]$. An illustration is shown in Figure 7(b). A repeated application of subdivisions then yields the coordinates of the intersections.

### 1.3.3 Decomposition and parametrization

Now, we assume that we have a multiply connected domain $\mathcal{D} = \Omega \cap R$ as in Figure 1.5 and we would like to briefly describe the way of obtaining its decomposition into four-sided patches $\mathcal{P}_j$. It is beyond the scope of this paper to completely describe that decomposition. We will summarize only the most important steps and refer the reader to [18] for details. First, we take a coarse polygonal approximation $\mathbb{P}$ of the domain $\mathcal{D}$. For the case of a simply connected polygon $\mathbb{P}$, we have shown that it is always possible to chop off one quadrilateral (which is not necessarily convex) by inserting at most one internal node. By recursively applying this approach, one can generate a quadrangulation of $\mathbb{P}$. In the case of a multiply connected polygon $\mathbb{P}$, we need to insert cuts. That is, we join two vertices which are located on an interior curve and on an exterior one respectively. Note that in most cases, several possible cuts can be inserted. We have devised an algorithm [18] for choosing the optimal direction and the position of cuts to be inserted automatically. A drawback of this approach is that we may obtain some quadrilaterals which are non-convex so that we must employ some additional steps to convert the nonconvex quadrilaterals into convex ones. To obtain the decomposition of $\mathcal{D}$ from $\mathbb{P}$, we simply replace every straight boundary edge of the quadrilaterals by the corresponding curvilinear part from $\mathcal{D}$. Note however that we must be concerned with issues like corner smoothing or boundary interference [18]. The number of the our-sided patches $\mathcal{P}_j$ such that $\mathcal{D} = \cup_j \mathcal{P}_j$ constructed by this approach is not minimal but small.

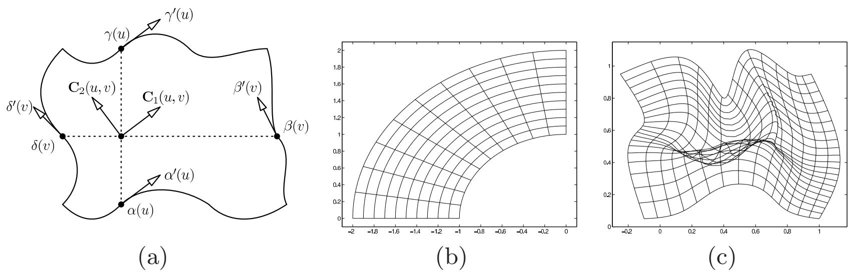

**Fig. 1.8.** (a) Tangents on a four sided domain for Coons patch. (b) Diffeomorphic Coons patches. (c) Undesired overspill phenomena.

Now, we want to generate a mapping onto the four-sided subdomains $\mathcal{P}_j$ which result from the above process. Let $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$ : $[0,1] \longrightarrow \mathbf{R}^2$ be four $\mathcal{C}^1[0,1]$ curves that satisfy the compatibility conditions at the corners such as $\boldsymbol{\alpha}(0) = \boldsymbol{\delta}(0)$, $\boldsymbol{\alpha}(1) = \boldsymbol{\beta}(0)$, $\boldsymbol{\gamma}(0) = \boldsymbol{\delta}(1)$, $\boldsymbol{\gamma}(1) = \boldsymbol{\beta}(1)$. We assume that besides those common points, there are no further intersection points. Since our method of generating a map from the unit square to the four-sided domain $S$ bounded by $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$ is based on transfinite interpolation, we briefly recall some basic facts about this technique.

We are interested in generating a parametric surface $\mathbf{x}(u,v)$ defined on the unit square $[0,1]^2$ such that the boundary of the image of $\mathbf{x}$ coincides with the given four curves:

$$\begin{array}{lll} \mathbf{x}(u,0) = \boldsymbol{\alpha}(u) & \mathbf{x}(u,1) = \boldsymbol{\gamma}(u) & \forall\, u \in [0,1] \\ \mathbf{x}(0,v) = \boldsymbol{\delta}(v) & \mathbf{x}(1,v) = \boldsymbol{\beta}(v) & \forall\, v \in [0,1]\ . \end{array} \tag{1.9}$$

This transfinite interpolation problem can be solved by a first order Coons patch whose construction involves the operators

$$(\mathcal{Q}_1\mathbf{x})(u,v) := F_0(v)\mathbf{x}(u,0) + F_1(v)\mathbf{x}(u,1) \tag{1.10}$$

$$(\mathcal{Q}_2\mathbf{x})(u,v) := F_0(u)\mathbf{x}(0,v) + F_1(u)\mathbf{x}(1,v) \tag{1.11}$$

where the so-called blending functions $F_0$ and $F_1$ denote two arbitrary smooth functions satisfying

$$F_i(j) = \delta_{ij} \quad i,j = 0,1 \quad \text{and} \qquad F_0(t) + F_1(t) = 1 \quad \forall\, t \in [0,1], \tag{1.12}$$

i.e. form a univariate PU. Obviously, there is much freedom in the choice of $F_0$ and $F_1$, throughout this paper we employ a linear blending. Now, a Coons patch $\mathbf{x}$ can be defined [7] by the relation

$$\mathcal{Q}_1 \oplus \mathcal{Q}_2(\mathbf{x}) = \mathbf{x}, \qquad \text{where} \qquad \mathcal{Q}_1 \oplus \mathcal{Q}_2 := \mathcal{Q}_1 + \mathcal{Q}_2 - \mathcal{Q}_1\mathcal{Q}_2. \tag{1.13}$$

It follows that $\mathbf{x}$ is of the form

$$\mathbf{x}(u,v) = -\begin{bmatrix} -1 \\ F_0(u) \\ F_1(u) \end{bmatrix}^T \begin{bmatrix} \mathbf{0} & \mathbf{x}(u,0) & \mathbf{x}(u,1) \\ \mathbf{x}(0,v) & \mathbf{x}(0,0) & \mathbf{x}(0,1) \\ \mathbf{x}(1,v) & \mathbf{x}(1,0) & \mathbf{x}(1,1) \end{bmatrix} \begin{bmatrix} -1 \\ F_0(v) \\ F_1(v) \end{bmatrix}. \tag{1.14}$$

The above Coons representation can be converted into B-spline or NURBS form provided that the four boundary curves are B-spline or NURBS curves. In Figure 1.8, we illustrate that for simple cases a Coons patch is already diffeomorphic. However, when the boundary curves become too wavy, like in Figure 8(c), we observe overlapping isolines indicating that the mapping is not invertible. We will need the notion of discrete B-splines to formulate some of our subsequent results. If $\mathbf{t} = (t_i)$ is a subknot of $\boldsymbol{\tau} = (\tau_i)$, then $N_j^{k,\mathbf{t}} = \sum_i b_{j,k}^{\boldsymbol{\tau},\mathbf{t}}(i) N_i^{k,\boldsymbol{\tau}}$ where $b_{j,k}^{\boldsymbol{\tau},\mathbf{t}}$ are the discrete B-splines given by the recurrence relations:

$$\begin{aligned} &b_{j,1}^{\boldsymbol{\tau},\mathbf{t}}(i) := N_j^{1,\mathbf{t}}(t_i) \\ &b_{j,k}^{\boldsymbol{\tau},\mathbf{t}}(i) := \omega_{i,k,\mathbf{t}}(\tau_{i+k-1}) b_{j,k-1}^{\boldsymbol{\tau},\mathbf{t}}(i) + (1 - \omega_{j+1,k,\mathbf{t}}(\tau_{i+k-1})) b_{j+1,k-1}^{\boldsymbol{\tau},\mathbf{t}}(i) \end{aligned} \tag{1.15}$$

where $\omega_{i,k,\mathbf{t}}(u) := (u - t_i)/(t_{i+k-1} - t_i)$. Below, we present some conditions on the boundary curves that guarantee the regularity of the Coons map. The linear independence of tangents on opposite curves (see Figure 8(a)) in conjunction with a second condition that controls the curvatures of the boundary curves, are sufficient for the regularity of $\mathbf{x}$. We suppose first that the boundary curves $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$ are B-spline curves with control points $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$, $\boldsymbol{\delta}_i$. The opposite curves $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are supposed to be defined on the knot sequence $\mathbf{t}^u = (t_i^u)$ while $\boldsymbol{\beta}$ and $\boldsymbol{\delta}$ on $\mathbf{t}^v = (t_i^v)$ :

$$\boldsymbol{\alpha}(t) = \sum_{i=0}^{n_u} \boldsymbol{\alpha}_i N_i^{k_u}(t), \qquad \boldsymbol{\beta}(t) = \sum_{i=0}^{n_v} \boldsymbol{\beta}_i N_i^{k_v}(t), \tag{1.16}$$

$$\boldsymbol{\gamma}(t) = \sum_{i=0}^{n_u} \boldsymbol{\gamma}_i N_i^{k_u}(t), \qquad \boldsymbol{\delta}(t) = \sum_{i=0}^{n_v} \boldsymbol{\delta}_i N_i^{k_v}(t). \tag{1.17}$$

Since the orders of opposite curves are different in general, we use the discrete B-spline techniques in (1.15) to obtain equal order representations. To ensure that the first and the last control points are interpolated, we assume that the knot sequences $\mathbf{t}^u = (t_i^u)$ and $\mathbf{t}^v = (t_i^v)$ are clamped as in (1.6). Moreover, let us assume that the blending function $F_1$ is expressed in Bézier form such that $F_1(t) = \sum_{i=0}^{n} \phi_i B_i^n(t) = 1 - F_0(t)$ and introduce $F := \max\{S^1, S^2\}$ where

$$S^1 := \max_{i=0,\cdots,n_v} \{\rho \|\boldsymbol{\beta}_i - \boldsymbol{\delta}_i\|\} \qquad \text{and} \qquad S^2 := \max_{i=0,\cdots,n_u} \{\rho \|\boldsymbol{\gamma}_i - \boldsymbol{\alpha}_i\|\}. \tag{1.18}$$

Furthermore, we define

$$\lambda_i := (k_u - 1)/(t_{i+k_u}^u - t_{i+1}^u) \quad \text{and} \quad \mu_j := (k_v - 1)/(t_{j+k_v}^v - t_{j+1}^v) \tag{1.19}$$

for all $i = 1, \cdots, n_u$, $j = 1, \cdots, n_v$ and introduce the expressions

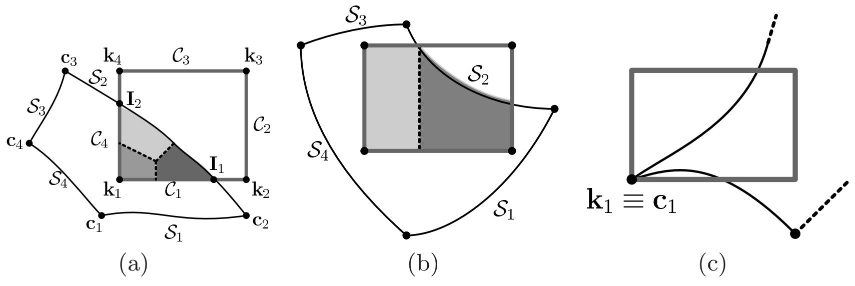

**Fig. 1.9.** (a) Special case where $\mathcal{S}_2$ intersects $\mathcal{C}_1$ and $\mathcal{C}_4$ and $\mathbf{k}_1$ in $\mathcal{P}$. (b) Special case where $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_4$ are inside $\mathcal{P}$ while $\mathcal{C}_2 \cap \mathcal{S}_2 \neq \emptyset$ and $\mathcal{C}_3 \cap \mathcal{S}_2 \neq \emptyset$. (c) Nodal coincidence.

$$\begin{aligned} A_{ij} &:= \lambda_i \mu_j \det[\boldsymbol{\alpha}_{i+1} - \boldsymbol{\alpha}_i, \boldsymbol{\delta}_{j+1} - \boldsymbol{\delta}_j]\,, \quad & B_{ij} &:= \lambda_i \mu_j \det[\boldsymbol{\alpha}_{i+1} - \boldsymbol{\alpha}_i, \boldsymbol{\beta}_{j+1} - \boldsymbol{\beta}_j]\,, \\ C_{ij} &:= \lambda_i \mu_j \det[\boldsymbol{\gamma}_{i+1} - \boldsymbol{\gamma}_i, \boldsymbol{\delta}_{j+1} - \boldsymbol{\delta}_j]\,, \quad & D_{ij} &:= \lambda_i \mu_j \det[\boldsymbol{\gamma}_{i+1} - \boldsymbol{\gamma}_i, \boldsymbol{\beta}_{j+1} - \boldsymbol{\beta}_j]\,, \end{aligned}$$

and

$$\tau := \min_{i,j} \{A_{ij}, B_{ij}, C_{ij}, D_{ij}\}. \tag{1.20}$$

Let $M$ be a constant such that

$$\begin{aligned} \lambda_i \|(1-\phi_j)(\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_{i-1}) + \phi_j(\boldsymbol{\gamma}_i - \boldsymbol{\gamma}_{i-1})\| &\leq M \\ \mu_l \|(1-\phi_j)(\boldsymbol{\delta}_l - \boldsymbol{\delta}_{l-1}) + \phi_j(\boldsymbol{\beta}_l - \boldsymbol{\beta}_{l-1})\| &\leq M, \end{aligned} \tag{1.21}$$

for all $i = 1, \cdots, n_u$; $l = 1, \cdots, n_v$ and $j = 0, \cdots, n$. Suppose that $A_{ij}$, $B_{ij}$, $C_{ij}$, $D_{ij}$ are all positive for all $i = 0, \cdots, n_u - 1$ and $j = 0, \cdots, n_v - 1$. Then the condition $2MF + F^2 < \tau$ is sufficient [13] for $\mathbf{x}$ to be a diffeomorphism. More efficient results for checking regularity are detailed in [13, 18] by using adaptive subdivisions. We used a method [13] for treating curves which are not necessarily in the form (1.16) and (1.17).

### 1.3.4 Rectangle-NURBS clipping

This section will discuss the fast process of NURBS-decomposition of the intersection between a rectangle $\mathcal{R}$ and a NURBS patch $\mathcal{P}$ which does not present an overspill phenomenon as in Figure 8(c). Note that the process here is different from the one in Section 1.3.2 and Section 1.3.3. Of course, one can apply the method there but our main objective here, apart from finding a result, is to make that intersection process fast because it has to be applied very often in PPUM simulation. Let the four curve sides of $\mathcal{P}$ be $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_4$ and its corners be $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, $\mathbf{c}_4$. Similarly, the sides and the corners of $\mathcal{R}$ are respectively $\mathcal{C}_i$ and $\mathbf{k}_i$. The process consists in distinguishing many special cases depending on several factors: (1) intersection of $\mathcal{S}_i$ with $\mathcal{C}_j$, (2) position of the corner $\mathbf{c}_i$ with respect to $\mathcal{R}$, (3) position of the corner $\mathbf{k}_i$ with respect to $\mathcal{P}$. We need to implement a program where each special case is individually

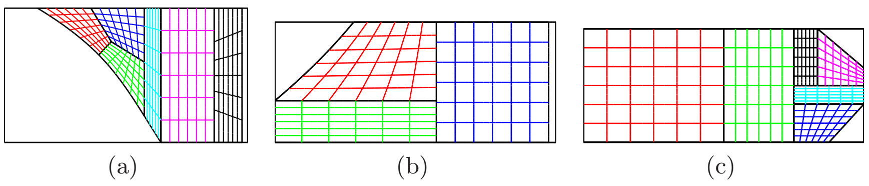
Fig. 1.10. Recursively applying some special cases.

treated. It is beyond the scope of this paper to describe all possible special cases. In Figure 9(a) and Figure 9(b), we display two special cases. For the first situation, the patch side $\mathcal{S}_2$ intersects rectangle sides $\mathcal{C}_1$ and $\mathcal{C}_4$ while no corners $\mathbf{c}_i$ are included in $\mathcal{R}$ and the corner $\mathbf{k}_1 \in \mathcal{P}$. For the second situation, three corners $\mathbf{k}_i$ are inside the patch while $\mathcal{S}_2$ intersects $\mathcal{C}_2$ and $\mathcal{C}_3$. In practice, about 15 cases are sufficient if none of the corners $\mathbf{c}_i$, $\mathbf{k}_j$ coincide as in Figure 9(c) and if we have $\mathbf{c}_i \notin \mathcal{C}_j$ and $\mathbf{k}_i \notin \mathcal{S}_j$ for all $i, j = 1, ..., 4$. More cases must be implemented to treat those latter cases which are not a rare situation for a simulation on practical CAD models. The practical difficulty is to come up to a fast and efficient point location method inside a NURBS patch and curve-curve intersections. If the rectangle $\mathcal{R}$ is too large then we split it into two rectangles $\mathcal{R}_1$ and $\mathcal{R}_2$ and apply the same method to each subrectangle $\mathcal{R}_i$. One chooses between vertical or horizontal splitting whichever gives the better shape (closer to a square) for the sub-rectangles. Some results of such recursive splitting are displayed in Figure 1.10. Note that the resulting NURBS patches are not globally continuous [19] but that does not create any problem for the PPUM approach. Problems related to curvature may occur in those special cases if the curves are too wavy. In such a situation, one has to apply NURBS subdivisions.

## 1.4 Numerical Experiments

The former geometric processing has been implemented in C/C++ and was integrated in our PPUM implementation. As a reference example of a CAD model, we use the exterior domain of an aircraft, see Figure 1.11. Let us now show some numerical results about the clipping of a NURBS patch $\mathcal{P}$ by a rectangle $\mathcal{R}$ as described in Section 1.3.4. To quantify the distortion of the bounding curves from being straight, we used the following distortion gauge $\mathcal{G}(\mathcal{P})$. For a NURBS curve $\mathcal{S}$ which has control points $\mathbf{d}_i$ for $i = 0, ..., n$ and which starts at $\mathbf{A}$ and terminates at $\mathbf{B}$, we define

$$\mathcal{G}(\mathcal{S}) := \Big|\ell(\mathcal{S}) - \|\mathbf{A} - \mathbf{B}\|\Big| \sum_{i=0}^{n} \mathrm{proj}[\mathbf{d}_i, \mathcal{L}(\mathbf{A}, \mathbf{B})] \tag{1.1}$$

where $\ell(\mathcal{S})$ designates the chord length of the control polygon while proj $[\mathbf{x}, \mathcal{L}(\mathbf{A}, \mathbf{B})]$ denotes the projection of a point $\mathbf{x} \in \mathbf{R}^2$ onto the line $\mathcal{L}(\mathbf{A}, \mathbf{B})$

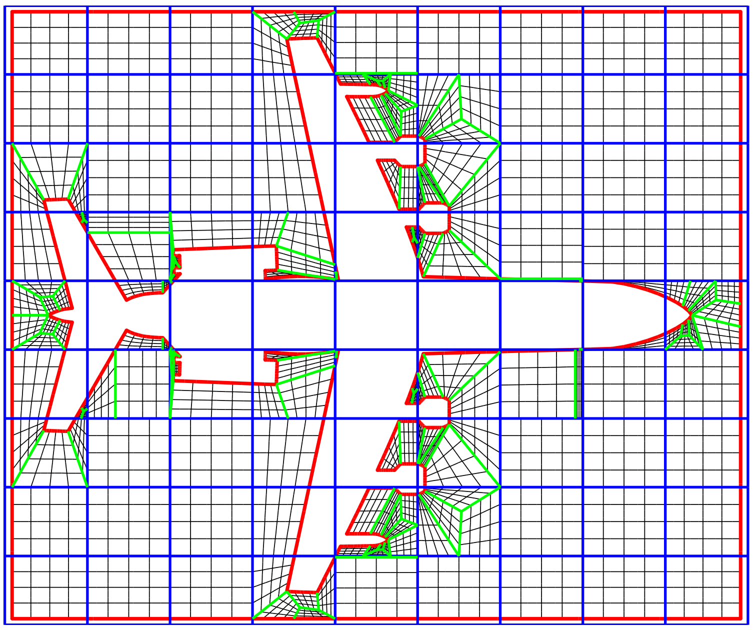

**Fig. 1.11.** Setup phase for an exterior domain.

**Table 1.2.** Performance of clipping operations for 2000 intersections with respect to average number of patches *np* (Fixed distortion gauge=7.693015).

| Size of rectangles | *np* |
|---|---|
| 0.00-0.10 | 1.009 |
| 0.10-0.20 | 1.041 |
| 0.20-0.30 | 1.103 |
| 0.30-0.40 | 1.146 |
| 0.40-0.50 | 1.179 |
| 0.50-0.60 | 1.246 |
| 0.60-0.70 | 1.295 |
| 0.70-0.80 | 1.338 |
| 0.80-0.90 | 1.390 |
| 0.90-1.00 | 1.441 |

**Table 1.3.** Performance of clipping operations for 2000 intersections with respect to average number of patches *np*.

| Distortion gauge | *np* |
|---|---|
| 0.000000 | 1.000000 |
| 0.095194 | 1.329500 |
| 0.734081 | 1.571786 |
| 2.347471 | 1.945000 |
| 5.220395 | 2.069104 |
| 9.528208 | 2.331331 |
| 15.386027 | 2.413327 |

passing through **A** and **B**. For a NURBS patch $\mathcal{P}$ having four boundary curves $\mathcal{S}_1$,...,$\mathcal{S}_4$, we define the distortion gauge to be $\mathcal{G}(\mathcal{P}) := \sum_{i=1}^{4} \mathcal{G}(\mathcal{S}_i)$. That is, for a NURBS patch $\mathcal{P}$ which is a perfect convex quadrilateral, the distortion gauge $\mathcal{G}(\mathcal{P})$ is zero.

First, we would like to examine the number of resulting NURBS patches. Table 1.2 gathers some numerical results from 2000 clipping operations. The first column presents the ratio of the area of the rectangle $\mathcal{R}$ with respect to the area of the original NURBS patch $\mathcal{P}$. The rectangles are chosen randomly

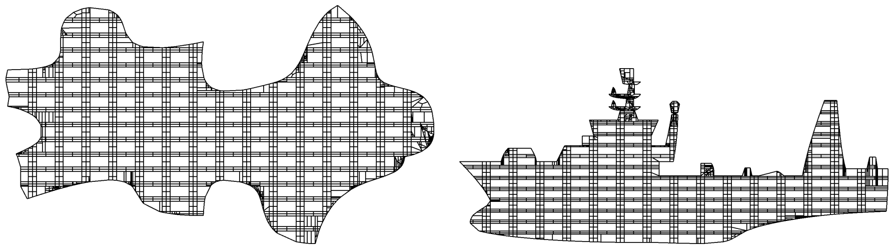
**Fig. 1.12.** Wireframe representation of integration cells for two different reference domains on level 4. All interior cells are affine.

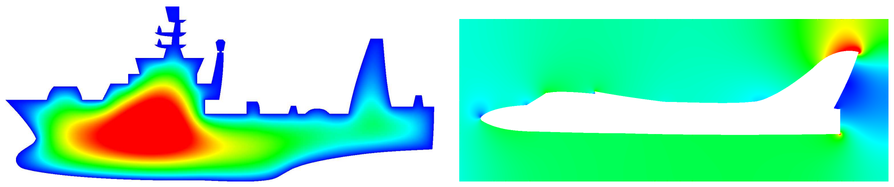
**Fig. 1.13.** Contour plot of approximation to problem (1.2) with homogeneous Dirichlet boundary conditions and $f = 1$ on level 8 (left). Contour plot of computed pressure for a potential flow problem (1.2) with inflow boundary conditions at the left boundary on level 8.

using the condition that the intersections are not empty. We observe that the average number of resulting patches are significantly small.

As a second test, we generate a NURBS patch $\mathcal{P}$ whose distortion coefficient $\mathcal{G}(\mathcal{P})$ can be changed. We investigate the average number of patches in clipping operations in terms of the distortion coefficient. The NURBS patch is chosen so that when $\mathcal{G}(\mathcal{P})$ vanishes, $\mathcal{P}$ coincides to a rectangle. In Tab. 1.3, we display the results of such tests. We observe that the number of patches for the intersections is reasonably small even when the distortion gauge is already practically large.

Finally, we present some approximation results with the PPUM on general domains. To this end, we consider a simple diffusion problem

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ u &= g_D \text{ on } \Gamma_D \subset \partial\Omega, \\ \partial_n u &= g_N \text{ on } \Gamma_N := \partial\Omega \setminus \Gamma_D \end{aligned} \tag{1.2}$$

on three different realistic domains in two space dimensions, see Figures 1.13 and 1.14, and a linear elasticity model problem

$$\begin{aligned} -\,\mathbf{div}\,\boldsymbol{\sigma}(u) &= f && \text{in } \Omega, \\ u &= g_D && \text{on } \Gamma_D \subset \partial\Omega, \\ \boldsymbol{\sigma}(u)\cdot n &= g_N && \text{on } \Gamma_N := \partial\Omega \setminus \Gamma_D, \end{aligned} \tag{1.3}$$

see Figure 1.15.

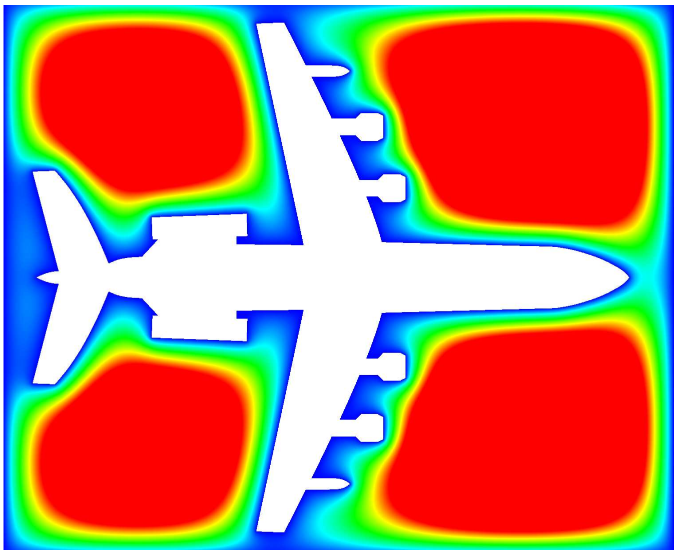

**Fig. 1.14.** Contour plot of approximation to problem (1.2) with homogeneous Dirichlet boundary conditions and $f = 1$ on level 9.

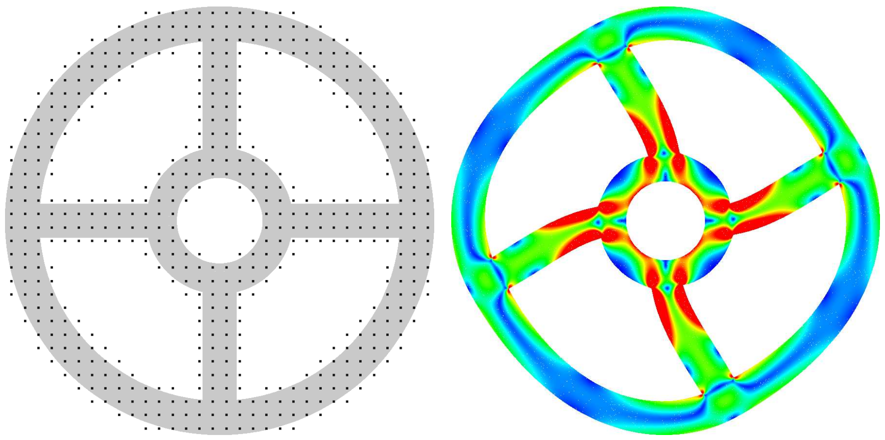

**Fig. 1.15.** Computational domain and particle distribution on level 5 considered in (1.3) (left). Here, we apply tangential tractions on the outer ring and homogeneous Dirichlet boundary conditions along the inner ring. Contour plot of the computed von Mises stress on the deformed configuration on level 9 (right).

We consider a sequence of uniformly refined covers $C_\Omega^k$ with $\alpha = 1.3$ in (1.3) and local polynomial spaces $\mathcal{P}^{p_{i,k}}$ on all levels $k = 1, \ldots, J$ in this paper. From the plots depicted in Figure 1.12 we see that only a small number of integration cells must be intersected with the boundary and that the total number of integration cells is increased only slightly. Thus, the compute time spent in the assembly of the linear system is almost unaffected by the geometric complexity of the domain. However, the compute time spent in the processing of the domain, i.e. in the computation of the intersections, is currently the most time consuming step; it takes about 70% of the total compute time — which is comparable with the situation in FEM.

Recall that we construct a cover patch $\omega_i$, i.e. a PU function $\varphi_i$, for each tree-cell $\mathcal{C}_i$ which satisfies $\mathcal{C}_i \cap \Omega \neq \emptyset$. Thus, close to the boundary we may have to deal with PU functions $\varphi_i$ whose support $\omega_i$ intersects the domain $\omega_i \cap \Omega$ barely. Due to this issue we cannot ensure that all the PU functions have the flat-top property and we may experience a deterioration of the condition number of the stiffness matrix. How to overcome this issue is the subject of current work and will be discussed in a forthcoming paper. Here, we simply employ our multilevel solver [9] as a preconditioner for a conjugate gradient solver applied to the possibly ill-conditioned arising linear system.

The measured asymptotic convergence rate of a $V(1,1)$-preconditioned CG solver in our experiments varies between 0.25 and 0.80 depending e.g. on the number of patches $\omega_i$ with very small intersections $\mathcal{C}_i \cap \Omega$. The respective rate using a $V(5,5)$-cycle as a preconditioner however was already very stable at roughly 0.1 up to level 9 with about 500.000 degrees of freedom.

## 1.5 Concluding Remarks

We presented a general approach to the treatment of arbitrary domains in two space dimensions with meshfree Galerkin methods in this paper. We have implemented the proposed scheme in the PPUM and presented some first numerical results which clearly demonstrate the viability of our approach.

There are two main challenges which are currently being investigated. The compute time for the processing of the geometry must be further reduced to allow for an on the fly use of the presented approach which is essential for a direct coupling of the simulation engine to a CAD system. Moreover, the impact of very small intersections on the conditioning of the basis and stiffness matrix must be analyzed in detail.

## References

1. I. Babuška, U. Banerjee, and J. E. Osborn, *Survey of Meshless and Generalized Finite Element Methods: A Unified Approach*, Acta Numerica, (2003), pp. 1–125.
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. S. Beissel and T. Belytschko, *Nodal Integration of the Element-Free Galerkin Method*, Comput. Meth. Appl. Mech. Engrg., 139 (1996), pp. 49–74.
5. J. S. Chen, C. T. Wu, S. Yoon, and Y. You, *A Stabilized Conforming Nodal Integration for Galerkin Mesh-free Methods*, Int. J. Numer. Meth. Engrg., 50 (2001), pp. 435–466.
6. E. Cohen, T. Lyche, and R. Riesenfeld, *Discrete B-Splines and Subdivision Techniques in Computer Aided Geometric Design and Computer Graphics*, Computer Graphics and Image Processing, 14 (1980), pp. 87–111.
7. S. Coons, *Surfaces for Computer Aided Design of Space Forms*, tech. report, Department of Mechanical Engineering in MIT, 1967.
8. J. Dolbow and T. Belytschko, *Numerical Integration of the Galerkin Weak Form in Meshfree Methods*, Comput. Mech., 23 (1999), pp. 219–230.
9. 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 (2002), pp. 1655–1682.
10. ———, *A Particle-Partition of Unity Method—Part III: A Multilevel Solver*, SIAM J. Sci. Comput., 24 (2002), pp. 377–409.
11. ———, *A Particle-Partition of Unity Method—Part V: Boundary Conditions*, in Geometric Analysis and Nonlinear Partial Differential Equations, S. Hildebrandt and H. Karcher, eds., Springer, 2002, pp. 517–540.
12. ———, *A Particle-Partition of Unity Method—Part VII: Adaptivity*, in Meshfree Methods for Partial Differential Equations III, M. Griebel and M. A. Schweitzer, eds., vol. 57 of Lecture Notes in Computational Science and Engineering, Springer, 2006, pp. 121–148.
13. H. Harbrecht and M. Randrianarivony, *From Computer Aided Design to Wavelet BEM*, Journal of Computing and Visualization in Science, 13 (2010), pp. 69–82.
14. J. Hoschek and D. Lasser, *Grundlagen der geometrischen Datenverarbeitung*, Teubner, 1989.
15. N. Moës, J. Dolbow, and T. Belytschko, *A Finite Element Method for Crack Growth without Remeshing*, Int. J. Numer. Meth. Engrg., 46 (1999), pp. 131–150.
16. 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 (1970–1971), pp. 9–15.
17. H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline Techniques*, Springer, 2002.
18. M. Randrianarivony, *Geometric Processing of CAD Data and Meshes as Input of Integral Equation Solvers*, ph.d. thesis, Technische Universität Chemnitz, 2006.
19. ———, *On Global Continuity of Coons Mappings in Patching CAD Surfaces*, Computer Aided Design, 41 (2009), pp. 782–791.
20. M. A. Schweitzer, *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*, vol. 29 of Lecture Notes in Computational Science and Engineering, Springer, 2003.
21. ———, *Meshfree and Generalized Finite Element Methods*, Habilitationsschrift, Institut für Numerische Simulation, Universität Bonn, 2008.
22. ———, *Stable Enrichment and Local Preconditioning in the Particle–Partition of Unity Method*, tech. report, Sonderforschungsbereich 611, Rheinische Friedrich-Wilhelms-Univeristät Bonn, 2008.

23. ———, *An Adaptive hp-Version of the Multilevel Particle–Partition of Unity Method*, Comput. Meth. Appl. Mech. Engrg., 198 (2009), pp. 1260–1272.
24. ———, *An Algebraic Treatment of Essential Boundary Conditions in the Particle–Partition of Unity Method*, SIAM J. Sci. Comput., 31 (2009), pp. 1581–1602.
25. T. Strouboulis, I. Babuška, and K. Copps, *The Design and Analysis of the Generalized Finite Element Method*, Comput. Meth. Appl. Mech. Engrg., 181 (2000), pp. 43–69.
26. T. Strouboulis, K. Copps, and I. Babuška, *The Generalized Finite Element Method*, Comput. Meth. Appl. Mech. Engrg., 190 (2001), pp. 4081–4193.
27. T. Strouboulis, L. Zhang, and I. Babuška, *Generalized Finite Element Method using mesh-based Handbooks: Application to Problems in Domains with many Voids*, Comput. Meth. Appl. Mech. Engrg., 192 (2003), pp. 3109–3161.