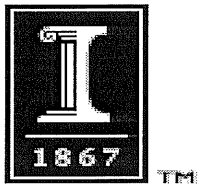
CIVIL ENGINEERING STUDIES
Structural Research Series No. 639

UILU-ENG-2005-2005



ISSN-0197-9191

The Generalized Finite Element Method – Improving Finite Elements Through Meshless Technology

by

C.A. Duarte

Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign

T.J. Liszka, W.W. Tworzydlo, T.A. Westermann

Altair Engineering, Inc.
Austin, Texas

May 2005

The Generalized Finite Element Method - Improving Finite Elements Through Meshless Technology

C.A. Duarte

Department of Civil and Environmental Engr.

University of Illinois at Urbana-Champaign

2122 Newmark Laboratory, 205 North Mathews Avenue

Urbana, Illinois 61801, USA

T.J. Liszka* and W.W. Tworzydlo and T.A. Westermann

Altair Engineering, Inc., 7800 Shoal Creek Blvd. Suite 290E

Austin, Texas, 78757, USA

*Corresponding author: liszka@tx.altair.com

Abstract

The Generalized Finite Element Method (GFEM) presented in this paper combines and extends the best features of the finite element method with the help of meshless formulations based on the Partition of Unity Method. Although an input finite element mesh is used by the proposed method, the requirements on the quality of this mesh are significantly relaxed. The main technique presented in this paper, "element clustering", allows the combination of neighboring elements, and generate mathematically correct and smooth solution approximation for such clusters. The paper shows how this can be used to effectively hide mesh defects internal to each cluster, and also allows for effective coarsening of the solution to reduce the computational cost and memory requirements in exchange for the solution accuracy.

In particular, the proposed GFEM can correctly and efficiently deal with: (i) severely distorted or elements with large aspect ratio; (ii) elements with negative Jacobian (inverted inside–out); (iii) large number of small elements; (iv) meshes consisting of several sub-domains with mismatched interfaces. Under such relaxed requirements for an acceptable mesh, and for correctly defined geometries, today's automated tetrahedral mesh generators can practically guarantee successful volume meshing that can be entirely hidden from the user. In addition, the method is fully applicable to all existing finite element algorithms (e.g. non–linear, time–dependent) and is also fully hp-adaptive.

Keywords: Meshless methods; Generalized finite element method; Partition of unity method; Hp-cloud method; Adaptivity.

1 Introduction

Computational simulation on complex three-dimensional (3D) domains has become a common task in recent years in many research laboratories and industries. However, the construction of an appropriate finite element discretization on complex 3D domains is still a difficult and time demanding task. The finite element method, which is widely used in industry and research, requires the generation of a mesh satisfying several quality criteria that are not easily matched when the geometry of the domain is complex. All elements in a mesh must, for example, be properly connected to their neighbors and accurately represent the geometry of the domain. In addition, the aspect ratio of the elements must be within acceptable bounds and elements with small or negative Jacobians are not acceptable.

Automatic mesh generators are now widely used in computational mechanics. However, the current state of the art in automatic mesh generation suffers from several limitations. Automatic mesh generators for hexahedral elements are still a subject of intense research and those currently available require considerable user intervention and tuning of parameters in order to produce acceptable meshes. This limitation has lead to increasing use of automatic tetrahedral mesh generators, although the solutions from these elements are in general of lesser quality than comparable hexahedral models. Tetrahedral mesh generators are much more robust then their hexahedral counterparts, but they also suffer from several limitations common to any automatic mesh generator. They tend, for example, to produce elements of poor quality near curved boundaries that need to be manually fixed by the user, leading to an overall very time consuming process. These difficulties are compounded when quadratic or higher order elements are used. Automatic mesh generators also often create an excessive number of elements in order to keep the aspect ratio of the elements within reasonable bounds. This is especially pronounced when the domain has transition zones between bulky and slender parts. Another drawback of automatic mesh generators, especially when tetrahedral elements are used, is that they inhibit the optimal use of p-anisotropic approximations, that is, approximations that have different polynomial orders associated with each direction. Problems where boundary layers occur, such as in the analysis of orthotropic materials or high speed flow or where one of the dimensions of the structural part is much smaller than the others, are examples in which p-orthotropic approximations may lead to considerable savings in the number of degrees of freedom needed to achieve acceptable accuracy. Similarly, when modeling element parts with different leading dimensions (e.g. thick plates), considerable savings can be obtained if consistent orthotropic approximation of different p-orders can be used.

It is common practice, especially in the well established FEM analysis centers in the industry, that the solution process is split into two (very) separate stages: meshing and analysis. As the main effort measurable in human time expenditure is used on creating acceptable FE meshes, the meshing itself became the goal, although actually it has absolutely no value to the engineer, who is interested in the numerical solution (e.g. stress distribution and its implications for the design process). The first attempts to combine the meshing and solution process into a single, fully automated process, employed error estimation techniques and mesh adaptivity. This resolved one aspect of the meshing: that of attaining suitable mesh density to reduce solution approximation errors without solving overkill large problems with very fine mesh. Unfortunately, adaptive finite element programs did not help in automatic mesh generation, and in many cases actually made mesh generation more difficult. Traditional finite element codes will very often accept meshes with topological errors (e.g duplicate elements, mismatched edges) or bad elements, and may produce “acceptable” results, while automatic adaptivity will usually outright refuse to accept such a mesh, or at least expose these errors by producing local solution singularities.

The difficulties associated with the generation of quality meshes for the finite element method has lead to the investigation of alternative methods for solving boundary value problems. In particular this lead

to huge popularity during last decade of all numerical techniques which could claim the "meshless method" name. Among these alternatives is the the generalized finite element method (GFEM) [3, 12, 13, 15, 28, 31, 32, 35, 36]. This method is a special case of the so-called partition of unity method [2, 3, 16, 18, 19, 28] and is closely related to the classical finite element method (FEM) while providing a much higher level of flexibility than the later. The partition of unity framework used in the GFEM can be exploited to create a very robust and flexible method capable of using meshes that are unacceptable for the finite element method, while retaining its accuracy and computational efficiency. The GFEM method presented in this paper, although focused on the solver technology, is aimed at solving meshing problems by reducing the quality requirements for the initial mesh, and thus allowing the analyst to concentrate on the solution process as a whole, in particular on the quality of final solution results, without artificial emphasis on the meshing process. Hopefully, this may allow creation of fully automatic analysis packages, where automatic mesh generators are a hidden part of the whole process, and the output solution is guaranteed good accuracy, even if the automatically generated mesh is not perfect.

This paper investigates possible extensions to the partition of unity framework used in the GFEM so that the resulting method has the following unique features:

1. It can perform effective unrefinement of finite element meshes composed of any type of elements in two- or three-dimensional spaces. This is accomplished not by changing the mesh, but by "clustering" a set of nodes into a single node or "cluster" with the aid of a modified finite element partition of unity, as described in Section 2.2. We refer to the GFEM presented here as a *GFEM with clustering* (GFEMC) whenever we want to emphasize its distinction from existing generalized finite element methods.

2. It accepts mismatched finite element meshes. That is, meshes composed of subparts that were meshed independently of each other. Here, the partition of unity framework is used to "glue" the subparts together in such way that the solution is continuous along the entire interface between the subparts. This technique is presented in Section 2.4.

3. It accepts meshes containing elements with bad aspect ratios and even elements with negative Jacobians, while retaining its accuracy and computational efficiency. This again is handled by clustering all of the nodes of an unacceptable element into a single node as described in Section 2.5.

In addition, the method presented herein retains all of the attractive features of classical finite element methods. In particular:

1. The shape functions are polynomials and the integration of the matrices is done with the aid of the so-called master element exactly as in the classical finite element. This is in contrast with most meshless methods [4, 17, 18, 25, 26].

2. The computational performance is essentially the same as a finite element method when the same mesh is used. This is mainly due to the previous property.

3. It can be applied to the solve the same classes of problems solvable by the finite element method (elasto-statics, heat transfer, fluid mechanics, acoustics, etc.). In addition, the GFEM discretization can be mixed with classical finite elements if such a need arises.

A detailed technical discussion of the GFEM with clustering is presented in Section 2. Here, it is important to note that the technique can have a great impact in computer engineering and scientific simulations by providing the following benefits:

- *A virtually* 100 *percent success rate in the automatic discretization of complex CAD geometries without user intervention*. Given the relaxed requirements for an acceptable GFEM mesh (high element distortions, negative Jacobians, collapsed elements, element mis-matches), today's automated tetrahedral mesh generators can practically guarantee successful volume meshing of geometries that are correctly defined, i.e., without gaps, overlaps, etc. The automation of the meshing generation process will make it feasible to investigate a much broader range of alternative designs, thus leading to more optimal designs in a shorter period of time.

- Reduction of mesh size through "clustering". *This guarantees, within reasonable bounds, the solution of large models using only the computational resources available to the analyst.* This automatic model reduction capability may also be used to perform convergence analysis using several levels of mesh resolution with little or no user intervention.

- The capability of mesh unrefinement considerably expands the scope of mesh adaptivity. With today's technology, a mesh can not be unrefined (coarsened) past the initial mesh even if it is too fine for the requested accuracy. In addition, with the proposed mesh reduction, the representation of the geometry of the domain will be preserved, i.e., there will be *no* loss in the approximation of the geometry of the domain. This is in contrast to traditional mesh coarsening and it is unlikely that such a feature will be matched in the near feature using existing meshing technology.

In summary, the GFEM with clustering combines and extends the best features of the finite element method while allowing for easier and automatic model preparation. It may virtually guarantee that an acceptable computational model can be created even for the most complex domains with little or no user intervention, and that such a model can be solved using the computer facilities currently available to the analyst.

Importantly, presently there are very few techniques available in finite element analysis that can be used to handle mismatched meshes, mesh coarsening and, especially, element of poor quality. Some existing methods, while practically usable, have difficulties in retaining theoretical correctness and consistency. Below, we present a brief overview of techniques in this class.

1.1 Mismatched Meshes

Several methods to handle non-conforming or mismatched finite element meshes have been proposed in the literature. Among the widely used are the mortar element or Lagrange Multiplier method and rigid elements or point collocation in solid mechanics. The mortar/Lagrange Multiplier method suffers from Babuška-Brezzi [5, 30] type instability problems, especially when the interfaces between the subparts have corners or other types of singularities. The case of interfaces between different materials is also not trivial. In addition, this approach leads to non-positive definite matrices.

The use of rigid elements or point collocation does not lead to a saddle point problem like in the mortar/Lagrange Multiplier method. However, the continuity of the solution between subparts is imposed only at a discrete set of points. Therefore the behavior of the whole assembly depends on the number and

configuration of rigid elements/collocation points used. Moreover, the solution in the neighborhood of the interfaces exhibits artificial stress singularities that do not disappear with mesh refinement.

Another approach to connected dissimilar finite element meshes is to modify the formulation of the elements at the mismatched interface such that first-order patch tests are passed [7, 8]. This method is recommended only for the case of linear elements since it leads to sub-optimal convergence rates in the case of higher order elements.

1.2 Mesh Unrefinement

The existing technology for finite element mesh unrefinement is limited to two-dimensional meshes [6, 20]. In the case of three-dimensional meshes, the unrefinement can, in general, only be done on meshes obtained by successive refinements [33]. That is, the unrefinement algorithms cannot be used to obtain a mesh that is coarser than the initial mesh.

1.3 Elements of Unacceptable Quality

The approach described in Section 2.5 to handle elements of unacceptable quality in a finite element mesh enjoys the following features:

- Can be used with any type of finite element in two- and three-dimensional meshes;
- does not require any modification of the mesh;
- optimal convergence rates are achieved if the mesh is refined or enriched.

To our knowledge, there is no technique available in the literature that can deliver all the features above.

2 A GFEM with Clustering

2.1 Background – Generalized Finite Element Methods

In this section, we review the basic ideas behind the construction of generalized finite element approximations in a one-dimensional setting using a 1D linear finite element partition of unity. The two- and three-dimensional cases are based on exactly the same ideas and are discussed in Section 2.1.1. For a more detailed discussion on the theoretical aspects of GFE approximations, see, for example, [3, 13, 17, 28, 32] and the references therein. The eXtended Finite Element Method [29, 38] is another method closed related to the generalized finite element method.

Let u be a function defined on a domain $\Omega \subset \mathbb{R}$. Suppose that we build an open covering

$$\mathcal{T}_N = \{\omega_\alpha\}_{\alpha=1}^N \qquad \bar{\Omega} \subset \bigcup_{\alpha=1}^N \omega_\alpha$$

of Ω consisting of N supports ω_α (often called *clouds*) with centers at $x_\alpha,\ \alpha = 1, \ldots, N$.

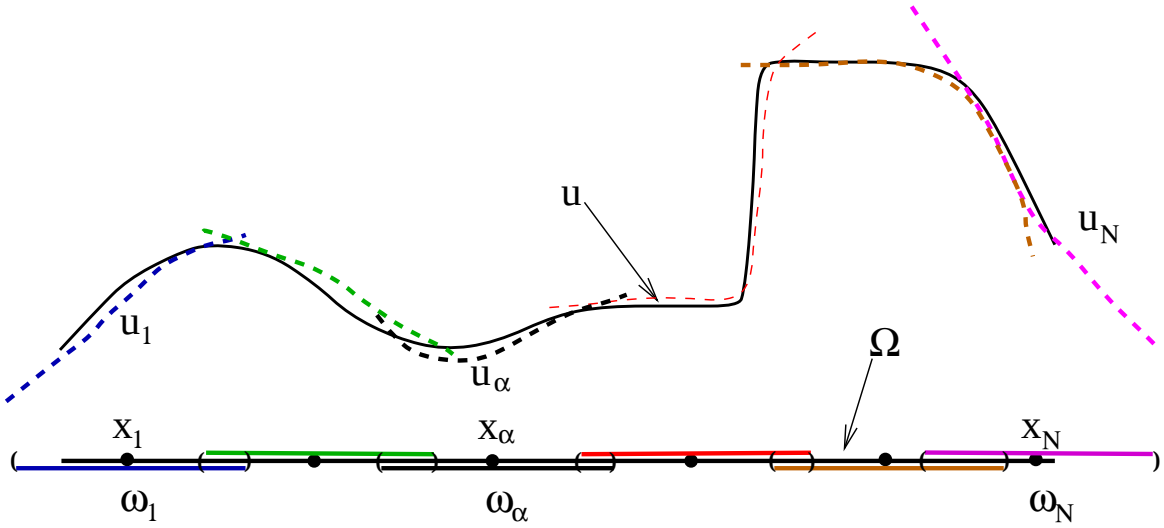


Figure 1: Local approximations defined on the supports ω_α.

Let u_α be a local approximation of u that belongs to a local space $\chi_\alpha(\omega_\alpha)$ defined on the support ω_α. It is presumed that each space $\chi_\alpha(\omega_\alpha),\ \alpha = 1, \ldots, N$, can be chosen such that there exists a $u_\alpha \in \chi_\alpha(\omega_\alpha)$ that can approximate well $u|_{\omega_\alpha}$ in some sense. Here, $u|_{\omega_\alpha}$ denotes the restriction of u to ω_α. Figure 1 illustrates the definitions given above. In this case, the supports ω_α are open intervals with centers x_α.

The local approximations $u_\alpha,\ \alpha = 1, \ldots, N$, have to be somehow combined together to give a global approximation u_{hp} of u. This global approximation has to be built such that the difference between u_{hp} and u, in a given norm, be bounded by the local errors $u - u_\alpha$. In partition of unity methods, this is accomplished using functions φ_α defined on the supports $\omega_\alpha,\ \alpha = 1, \ldots, N$, and having the following property

$$\varphi_\alpha \in C_0^s(\omega_\alpha),\ s \geq 0, \qquad 1 \leq \alpha \leq N \tag{1}$$

$$\sum_\alpha \varphi_\alpha(x) = 1 \qquad \forall x \in \Omega \tag{2}$$

The functions φ_α are called a *partition of unity (POU) subordinate to the open covering* $\mathcal{T}_N$. Examples of partitions of unity are Lagrangian finite elements, the various "reproducing" functions generated by moving least squares methods and Shepard functions [18, 22].

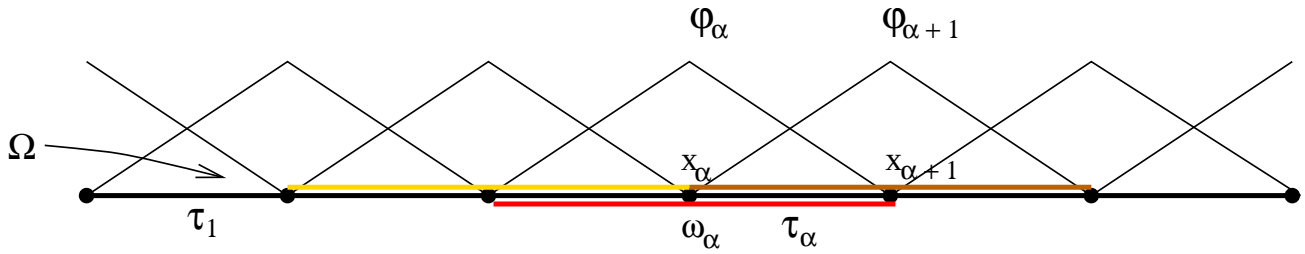


Figure 2: One-dimensional finite element partition of unity.

In the case of finite element partitions of unity, the supports (clouds) ω_α are simply the union of the finite elements sharing a vertex node x_α (see, for example, [28, 32]). In this case, the implementation

of the method is essentially the same as in standard finite element codes, the main difference being the definition of the shape functions as explained below. This choice of partition of unity avoids the problem of integration associated with the use of moving least squares methods or Shepard partitions of unity used in several meshless methods. Here, the integrations are performed with the aid of the so-called master elements, as in classical finite elements. Therefore, the GFEM can use existing infrastructure and algorithms developed for the classical finite element method.

Figure 2 shows a one-dimensional finite element discretization. The partition of unity functions φ_α are the usual global finite element shape functions, the classical "hat-functions", associated with node x_α. The support ω_α is thus the union of the elements $\tau_{\alpha-1}$ and τ_α.

Consider now the element τ_α with nodes x_α and $x_{\alpha+1}$ as depicted in Fig. 2. Suppose that the following shape functions are used on this element

$$S_\alpha = \{\varphi_\alpha, \varphi_{\alpha+1}\} \times \{1, u_\alpha, u_{\alpha+1}\} = \{\varphi_\alpha, \varphi_{\alpha+1}, \varphi_\alpha u_\alpha, \varphi_\alpha u_{\alpha+1}, \varphi_{\alpha+1} u_\alpha, \varphi_{\alpha+1} u_{\alpha+1}\}$$

That is, the element τ_α has a total of six shape functions (three at each node) built from the product of the standard Lagrangian finite element shape functions (a partition of unity), and the local approximations $u_\alpha, u_{\alpha+1}$ that, by assumption, can approximate well the function u over the finite element τ_α. Of course, we can further generalize this idea by increasing the number of functions u_α and $u_{\alpha+1}$, resulting in a space S_α of still larger dimension. This approach is discussed in the next section.

Thanks to the partition of unity property of the finite element shape functions, we can easily show that linear combinations of the shape functions defined above can reproduce the local approximations $u_\alpha, u_{\alpha+1}$, that is,

$$\begin{aligned} &\varphi_\alpha u_\alpha + \varphi_{\alpha+1} u_\alpha = u_\alpha(\varphi_\alpha + \varphi_{\alpha+1}) = u_\alpha \qquad \text{(no sum on } \alpha) \\ &\varphi_\alpha u_{\alpha+1} + \varphi_{\alpha+1} u_{\alpha+1} = u_{\alpha+1}(\varphi_\alpha + \varphi_{\alpha+1}) = u_{\alpha+1} \end{aligned}$$

In other words,

$$u_\alpha, u_{\alpha+1} \in \text{ span}\{S_\alpha\}.$$

The basic idea in partition of unity methods and, in particular, in the GFE method, is to use the partition of unity to "paste" together local approximation spaces. The shape functions are built such that they can reproduce, through linear combinations, the local approximations defined on each cloud ω_α. The approximation properties of such functions are discussed in the next section.

2.1.1 Generalized Finite Element Shape Functions: The Family of Functions $\mathcal{F}_N^p$

In this section, we define generalized finite element shape functions in an n-dimensional setting using the ideas outlined in the previous section.

Let the functions $\varphi_\alpha,\ \alpha = 1, \ldots, N$, denote a finite element partition of unity subordinate to the open covering $\mathcal{T}_N = \{\omega_\alpha\}_{\alpha=1}^N$ of a domain $\Omega \subset \mathbb{R}^n,\ n = 1, 2, 3$. Here, N is the number of vertex nodes in the finite element mesh. The cloud ω_α is the union of the finite elements sharing the vertex node $\boldsymbol{x}_\alpha$.

Let $\chi_\alpha(\omega_\alpha) = \text{span}\{L_{i\alpha}\}_{i \in \mathcal{I}(\alpha)}$ denote local spaces defined on $\omega_\alpha,\ \alpha = 1, \ldots, N$, where $\mathcal{I}(\alpha), \alpha = 1, \ldots, N$, are index sets and $L_{i\alpha}$ denotes local approximation or enrichment functions analogous with the

functions u_α mentioned in the previous section. Possible choices for these functions are discussed below.

Suppose that the finite element shape functions, φ_α, are linear functions and that

$$\mathcal{P}_{p-1}(\omega_\alpha) \subset \chi_\alpha(\omega_\alpha) \qquad \alpha = 1, \ldots, N,$$

where $\mathcal{P}_{p-1}$ denotes the space of polynomials of degree less or equal to $p-1$. The generalized finite element shape functions of degree p are defined by [12, 13, 28, 32, 35–37]

$$\mathcal{F}_N^p = \{\phi_i^\alpha = \varphi_\alpha L_{i\alpha},\ \alpha = 1, \ldots, N,\ i \in \mathcal{I}(\alpha)\} \tag{3}$$

Note that there is considerable freedom in the choice of the local spaces χ_α. The most obvious choice for a basis of χ_α is polynomial functions which can approximate smooth functions well. In this case, the GFEM is essentially identical to the classical FEM. The approximations can also be non-isotropic (i.e., different polynomial orders in different directions), regardless of the choice of finite element partition of unity (hexahedral, tetrahedral, etc.) [9–11].

There are many situations in which the solution of a boundary value problem is not a smooth function. In these situations, the use of polynomials to build the approximation space, as in the FEM, may be far from optimal and may lead to poor approximations of the solution u unless carefully designed meshes are used. In the GFEM, we can use any a-priori knowledge about the solution to make better choices for the local spaces χ_α. Then we can use the local spaces χ_α to build generalized finite element shape functions that represent these singularities much more effectively than polynomial functions.

2.1.2 Adaptivity in the GFEM

Generalized FEM is so similar to the "traditional" Finite Element Method, that all well known adaptation methods can be almost immediately applied to it. In particular, h-adaptation (refinement of elements), and r-adaptation (adjustment of nodal coordinates), being independent of shape functions do not require any modifications to be immediately applicable to the GFEM. p-adaptivity (modification of local order of approximation) is accomplished by varying the number α and type of local approximation functions $L_{i\alpha}$ in Eq. (3). Note that these enrichment functions are not limited by local element coordinate system (actually they can't be, because they usually span neighboring elements, as opposed to traditional FE enrichment functions, which are confined to each element). In particular, families χ_α can be selected as:

- traditional monomials of p-order defined in the global coordinate system (1, x, y, x^2, xy, y^2, ...) or any global coordinate system suitable to the problem,
- the same monomials but rotated in local coordinate system suitable for each node. Orientation of such coordinate system may be given as a part of the problem definition or determined dynamically using some kind of "directional error indicator". Note that this technique may allow substantial convergence improvement in boundary layers or for thin solid models, because the orthotropic p-approximation can be optimally oriented independently of local element orientation in the mesh (structured or unstructured) [9, 11].
- special function families selected based on a-priori knowledge of the solution (e.g. bi-harmonic functions to solve linear elasticity problems, singular expansion around the crack tip in fracture mechanics [14, 15], etc.)

The implementation of h and p adaptivity is, however, greatly simplified by the partition of unity (POU) framework. Since each basis function, $\{L_{i\alpha}\}_{i\in\mathcal{I}(\alpha)},\ \alpha = 1,\ldots,N$, can have a different polynomial order for each α, we can have different polynomial orders associated with each vertex node of the finite element mesh [32]. The concept of edge and middle nodes, which is used in conventional p FEMs, is not needed in the framework of the GFEM. Therefore, the implementation of h adaptivity for high-order approximations (hp-adaptivity) is the same as for linear approximations (there is no need, for example, of using high-order constraints and very complex consistency rules as is done in hp-adaptive finite element methods.

The concept of clustering presented in this paper introduces a new type of adaptivity (c-adaptivity) discussed in Section 3.1.3.

Detailed convergence analysis of the generalized finite element method can be found in [18, 19, 27, 28]. Several a-priori error estimates along with numerical experiments demonstrating their accuracy can also be found in those works.

2.2 Generalized Mesh Unrefinement Technique

The definition of a partition of unity given by (1) and (2) imposes only mild requirements on the partition of unity functions $\varphi_\alpha,\ \alpha = 1,\ldots,N$. In particular, given any partition of unity

$$\mathsf{POU}_N = \{\varphi_\alpha\}_{\alpha=1}^N$$

another partition of unity, $\mathsf{POU}_{N'}$ with $N' < N$, can be created by adding elements $\varphi_\beta,\ \beta \in \mathcal{I}$, belonging to the original set, where $\mathcal{I}$ is an index set. The resulting set has less elements then the original set if $\mathcal{I} \neq \emptyset$. This property of a partition of unity is the cornerstone of the generalized mesh unrefinement technique presented here.

An *unrefined or clustered partition of unity* is defined as follows:

Let $\mathsf{POU}_N = \{\varphi_\alpha\}_{\alpha=1}^N$ be a partition of unity composed of N elements. A *clustered or unrefined partition of unity* is then given by

$$\mathsf{POU}_{N'} = \{\varphi_\alpha = \sum_{\beta\in\mathcal{I}(\alpha)} \varphi_\beta \ :\ \alpha = 1,\ldots,N,\ \alpha \notin \mathcal{I}_{clustered}\} \tag{4}$$

Here, $\mathcal{I}(\alpha)$ are the indexes of the functions that are added to build the clustered function φ_α and $\mathcal{I}_{clustered}$ are the indexes of the functions in the the original set that are *not* in the clustered or unrefined set $\mathsf{POU}_{N'}$. The dimension of this set relative to the original number of nodes is an indication of the degree of clustering performed.

Let us consider some examples of clustered partitions of unity in the case of a finite element partition of unity in one- and two-dimensional spaces. From now on, we will refer to "partition of unity function associated with a node" simply by "node".

Figure 3 shows a one-dimensional finite element partition of unity composed of seven elements ($N = 7$). Figures 4, 5 and 6 show examples of unrefinements performed on the original partition of unity of Figure 3. In the figures, the boxes indicate the regions where the partition of unity is identically equal to one. In the

unrefined partition of unity shown in Figure 4 we have

$$\begin{aligned} N' &= 6 \\ \mathcal{I}_{clustered} &= \{5\} \\ \mathcal{I}(4) &= \{4,5\} \\ \mathcal{I}(\alpha) &= \alpha:\ \alpha = 1,2,3,6,7 \end{aligned}$$

If $\mathcal{I}(\alpha) = \alpha$, the partition of unity function φ_α is not modified. In the case of Figure 5 we have

$$\begin{aligned} N' &= 5 \\ \mathcal{I}_{clustered} &= \{4,5\} \\ \mathcal{I}(3) &= \{3,4,5\} \\ \mathcal{I}(\alpha) &= \alpha:\ \alpha = 1,2,6,7 \end{aligned}$$

and in the case of Figure 6 we have

$$\begin{aligned} N' &= 5 \\ \mathcal{I}_{clustered} &= \{3,6\} \\ \mathcal{I}(2) &= \{2,3\} \\ \mathcal{I}(5) &= \{5,6\} \\ \mathcal{I}(\alpha) &= \alpha:\ \alpha = 1,4,7 \end{aligned}$$

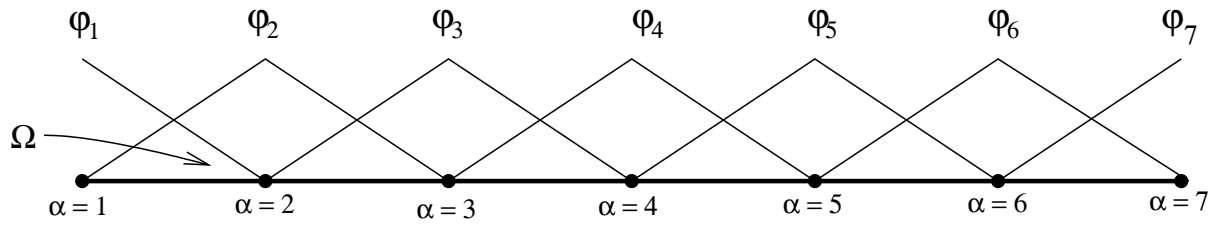


Figure 3: One-dimensional finite element partition of unity composed of seven elements ($N = 7$).

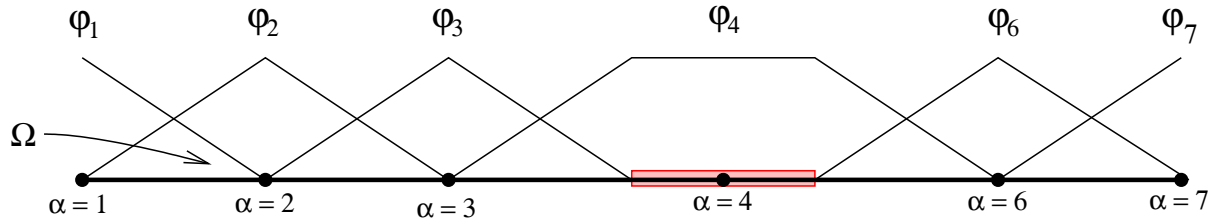


Figure 4: Unrefined finite element partition of unity. Functions φ_4 and φ_5 of Figure 3 were clustered into a single partition of unity function.

Figures 7 and 8 show examples of unrefinements of a uniform 3×3 mesh of quadrilateral finite elements. In the figures, the boxes indicate the regions where the partition of unity is identically equal to one. In the case of Figure 7, all partition of unity functions of the element in the center of the mesh were

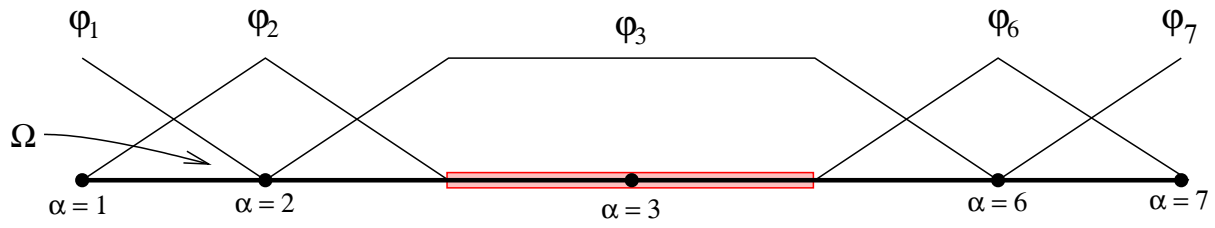


Figure 5: Resulting partition of unity from the clustering of functions φ_3, φ_4 and φ_5 of Figure 3.

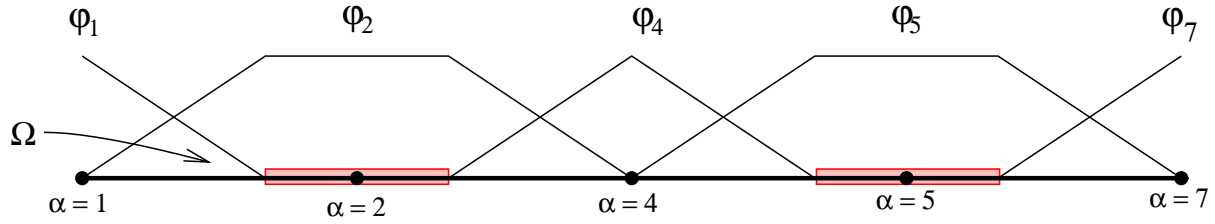


Figure 6: Unrefinement of the partition of unity shown in Figure 3.

clustered into a single function reducing the number of functions by three. For this example we have

$$\begin{aligned} N &= 16 \\ N' &= 13 \\ \mathcal{I}_{clustered} &= \{7, 10, 11\} \\ \mathcal{I}(6) &= \{6, 7, 10, 11\} \\ \mathcal{I}(\alpha) &= \alpha : \ \alpha \notin \mathcal{I}_{clustered} \text{ and } \alpha \neq 6 \end{aligned}$$

In the case of Figure 8 we have

$$\begin{aligned} N &= 16 \\ N' &= 11 \\ \mathcal{I}_{clustered} &= \{2, 5, 12, 15, 16\} \\ \mathcal{I}(1) &= \{1, 2, 5\} \\ \mathcal{I}(11) &= \{11, 12, 15, 16\} \\ \mathcal{I}(\alpha) &= \alpha : \ \alpha \notin \mathcal{I}_{clustered} \text{ and } \alpha \neq 1, 11 \end{aligned}$$

An unrefined or clustered partition of unity built as described above can be used in the construction of generalized finite element shape functions exactly as described in Section 2.1.1 for the case of a standard finite element partition of unity. This unrefinement technique although conceptually simple is very generic and can be used with any type of finite element and in any spatial dimension. Note also that there is no loss in the information about the geometry of the domain since all the elements of the original (not clustered) mesh are used in the clustered mesh. The reduction in the number of degrees of freedom comes from the clustering of partition of unity functions associated with finite element nodes in the mesh.

Besides being used to reduce the size of a computational model, the unrefinement technique described above can also be used in a local sense, for example, to handle inconsistent finite element meshes, meshes

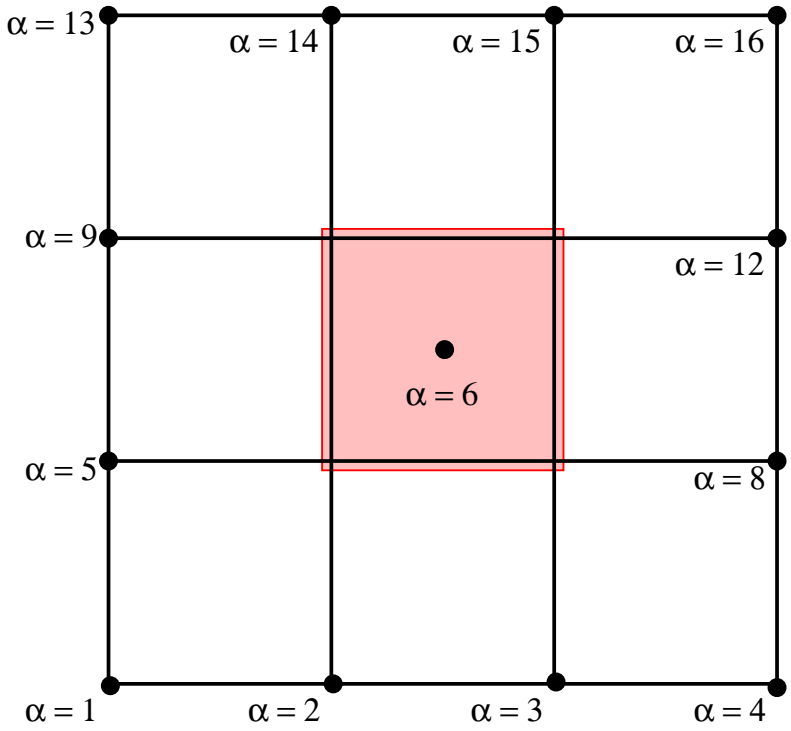


Figure 7: Representation of the unrefinement of a two dimensional finite element mesh. The partition of unity functions (nodes) of the element in the center of the domain were unrefined (clustered) into a single function (node). As a result, the partition of unity is identically equal to one over that element.

with very distorted elements, and even elements with negative Jacobians. These two topics are discussed in Sections 2.4 and 2.5, respectively.

2.3 Limitations of the Generalized Mesh Unrefinement Technique

In this section we discuss the limitations of the mesh unrefinement technique presented in the previous section.

2.3.1 Cluster Quality

The unrefinement procedure described in the previous section can be applied to as many nodes in a finite element mesh as needed. In addition, any set of nodes can be clustered into a single node. Here, the term *node* means *the partition of unity function associated with a node*. We also use the terms *cluster* to denote a set of partition of unity functions (nodes) that have been clustered (unrefined) into a single function (node) and *clustered element* to denote an element with all nodes clustered into a single node.

The approximation properties of the resulting GFE shape functions strongly depends on the size and shape of underlying partition of unity functions and care must be taken in the selection of the nodes used in each cluster. The optimal selection of these nodes is indeed not trivial. The automatic selection of a group of nodes to be clustered into a single node is denoted by a *clustering technique*. The techniques we have

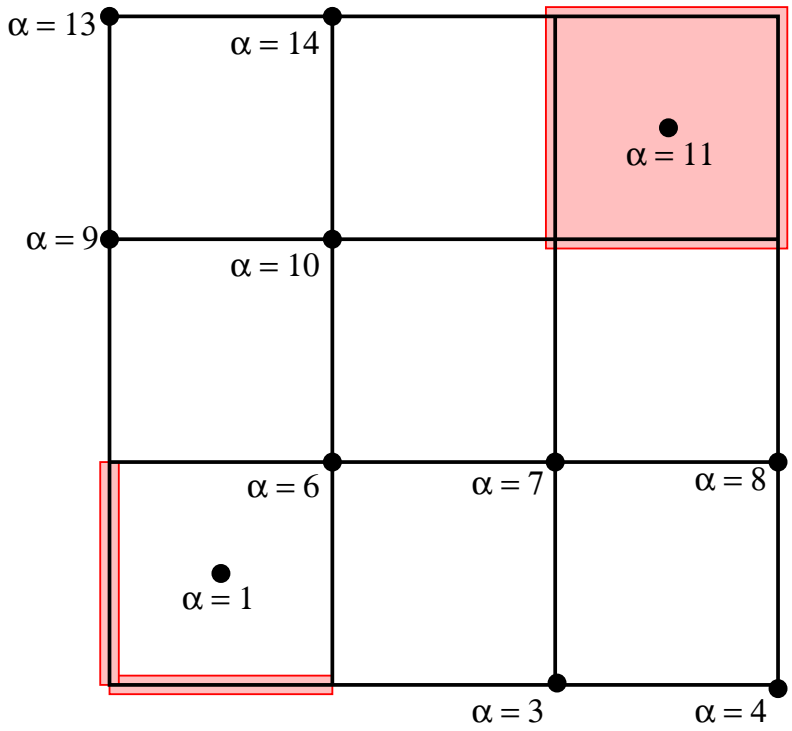


Figure 8: Unrefinement of a two dimensional finite element mesh. The clustering can be done quite arbitrarily. In this example, three nodes (i.e., partition of unity functions) of one element have been clustered into a single node.

investigated and how the effectiveness of each approach can be measured are described in this section.

Three clustering methods have been implemented: Hilbert-Peano space filing curve [34], Metis (based on the theory of graphs) [21] and Voronoi cells. Combinations of these techniques are also possible. The first two techniques have extensively been used in domain decomposition for parallel finite element solvers. The third one is actually implemented as a "correction" step to improve the shape of clusters.

All these methods led to reasonable results in the case of a simple, bulky domain, but are not sufficient in the case of multiply connected domains. The convergence of solutions was best when each cluster was roughly spherical in shape, and as "full" as possible.

More general clustering algorithm should should also take into account:

- The smoothness of the computed stresses. Good clustering algorithms should be able to deliver smooth stresses on clustered meshes if the solution of the problem is smooth and does not introduce artificial stress concentrations (hot spots) or oscillatory stresses.
- Convexity of the clusters. Non-convex clusters tend to produce stiff solutions and therefore must be avoided.
- Continuity of the clusters. The clustering algorithm must guarantee that each cluster is contiguous and possibly simply connected.

Increasing the size of the clusters in a GFEM discretization reduces the total number of degrees of freedom and has the same effect as using a coarser (unrefined) standard finite element mesh. Therefore, the discretization error increases with the size of the clusters. Ideally, the rate of change of the discretization error with respect to the number of degrees of freedom should be close to the one observed if the mesh was manually unrefined by using larger (standard) finite elements. This suggests the following approach to measure the performance of a clustering technique:

Starting from the original mesh, progressively cluster it and compute the corresponding discretization error measured in the energy norm. Different approaches for mesh clustering can then be compared based on:

- The algebraic rate of change of the error with respect to number of degrees of freedom, size of the clusters, CPU time for solution of the system of equations or total CPU time. Optimally, the rate should be as close as possible to the one obtained with traditional/manual mesh unrefinement.
- The monotonic or oscillatory behavior of the error relative to cluster size.

2.3.2 Global Enrichment of the GFEM Approximation

Generalized finite element shape functions, as defined in (3), are built from the product of a partition function, φ_α, by a local enrichment function, $L_{i\alpha}$. If the partition of unity functions are linear finite element shape functions and the enrichment functions are of degree $p-1$, the resulting shape function, ϕ_i^α, is of degree p.

Let us consider now the case where the partition of unity is a clustered partition of unity as defined in (4). In this case, the functions φ_α may be constant over an edge, face, element, or set of elements. Therefore

the GFE shape functions built using this partition of unity do not have a well defined p order. It is of degree p over edges, faces, and elements with partition of unity functions not clustered and of degree $p-1$ elsewhere. This must be taken into account when globally enriching a mesh.

At least three approaches can be used to globally enrich a GFE approximation to degree p when clustered partitions of unity are used:

1. Use local enrichment functions of degree less than or equal to p at every node. This is the most conservative approach and it considers that linear combinations of a clustered partition of unity can reproduce only a constant.

2. Use local enrichment functions of degree less than or equal to p only on the nodes of elements that have at least one partition of unity function that has been clustered. This guarantees that linear combinations of the shape functions of any element can reproduce polynomials of degree less or equal to p, which is the case of standard finite elements of degree p.

3. Use local enrichment functions of degree less then or equal to p only at nodes associated with partition of unity functions that has been clustered. This will lead to the smallest number of degree of freedom among the proposed enrichment strategies.

Each enrichment strategy leads, in general, to a different number of degrees of freedom and to a different approximation.

2.3.3 Stiffness and Mass Matrix Calculation

The mesh unrefinement technique preserves the representation of the geometry of the domain since all the elements of the original (not clustered) mesh are used in the clustered mesh. This however implies that the computational work to compute the global stiffness and mass matrices are the same in the original and unrefined mesh. In fact, the cost to compute the matrices in the unrefined mesh can be larger than in the original mesh if the same global polynomial order is used. This is due to the fact that, as described in the previous section, the polynomial order of the transition elements between clustered and non-clustered elements is one degree higher than the polynomial order in the clustered elements. Therefore, these transition elements have more degrees of freedom and also require more integration points than the elements in the original mesh.

The cost of computing the matrices relative to the cost of solving the system of equations decreases as the number of degrees of freedom increases. In addition, the matrix computations can be fully parallelized almost trivially as in most finite element methods. Nonetheless, we believe that this does not justify neglecting the cost of computing the matrices and we propose remedies to minimize this cost that take advantage of the special features of the GFEM shape functions over clustered elements.

Local Assembly of Clustered Elements

All clustered elements belonging to a given cluster have the same set of shape functions and, therefore, the same set of degrees of freedom. This suggests that it is more efficient to locally assemble the stiffness/mass matrices of these elements into a local matrix and only then assemble this single matrix into the global matrix. This minimizes the number of direct accesses to the global matrix, which involves searching operations in the case of, for example, sparse matrix storages.

Use of Functions that Satisfy the Underlying PDE

The local approximation functions $L_{i\alpha}$ used in the construction of the GFEM as defined in (3) can be chosen with great freedom. They can, in the case of some linear problems, be functions that satisfy the partial differential equations being solved. Consider now the case of the GFEM shape functions of an element τ in which all partition of unity functions (nodes) have been clustered into a single function (node) φ_β as in the case of the hatched element shown in Figure 7. Such an element is said to be clustered. The resulting partition of unity function over element τ is identically equal to one and the GFEM shape functions are given by

$$\phi_i^\beta = \varphi_\beta L_{i\beta} = L_{i\beta} \qquad i \in \mathcal{I}(\beta)$$

according to (3). Therefore, if $L_{i\alpha}$ satisfies the partial differential equations being solved, ϕ_i^β also does. This can be exploited during the matrix calculations to transform domain integrals into boundary integrals. That is, instead of integrating over all the elements in a given cluster, the integration can be done over the boundary of the cluster. For clusters composed of a large number of elements this can lead to substantial savings in the computation of the global matrices. This technique however, requires the knowledge of local enrichment functions $L_{i\alpha}$ that satisfy the partial differential equations being solved.

2.4 Mismatched Meshes in the GFEM

In this section, we describe how the mesh unrefinement technique can be used to handle finite element meshes that have localized mesh inconsistencies. This kind of mesh can arise in several situations. It is not uncommon, for example, to have complex mechanical parts divided in subparts that are simpler and meshed independently of each other. The problem then is how to "tie" the subparts together in a way such that the behavior of the whole assembly can be accurately represented.

The technique presented in this section is basically a local application of the mesh unrefinement method presented previously. As such, it does not suffer from Babuška-Brezzi type instability issues while rendering a solution that is continuous along the entire interface between inconsistent meshes.

The technique is better described with the aid of a few examples. Let us consider as a first example the mesh shown in Figure 9. This mesh can not be used, as it is, in the classical finite element method since the global shape functions associated with nodes 3, 4 and 5 are discontinuous along the boundary between elements τ_3 and τ_1, τ_2. These global shape functions are given by

$$\begin{aligned} \varphi_3 &= N_3^{\tau_1} \cup N_3^{\tau_3} \\ \varphi_4 &= N_4^{\tau_1} \cup N_4^{\tau_2} \\ \varphi_5 &= N_5^{\tau_2} \cup N_5^{\tau_3} \end{aligned}$$

where $N_\alpha^{\tau_i}$ represents the bilinear shape function of element τ_i associated with global node α.

These discontinuous global functions can be clustered into a single function using equation (4) with

$$\begin{aligned} N' &= 6 \\ \mathcal{I}_{clustered} &= \{4, 5\} \\ \mathcal{I}(3) &= \{3, 4, 5\} \\ \mathcal{I}(\alpha) &= \alpha :\ \alpha = 1, 2, 6, 7, 8 \end{aligned}$$

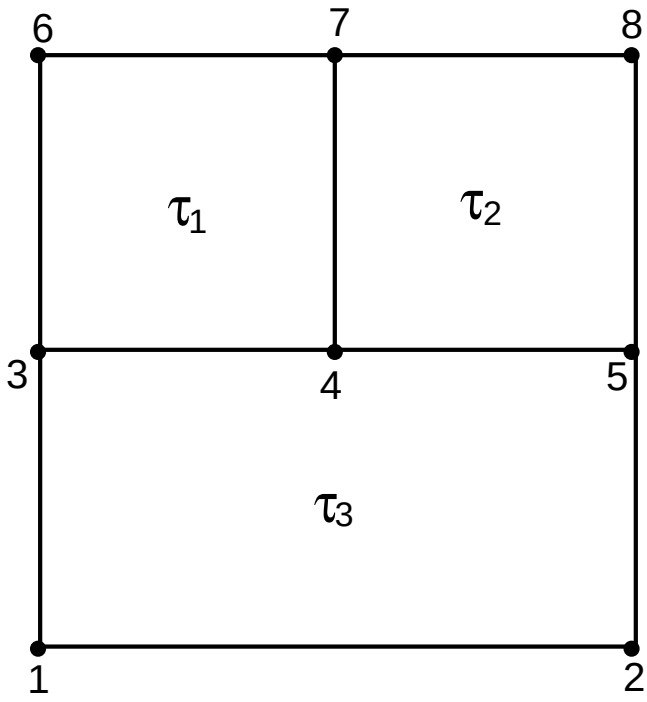


Figure 9: Mismatched mesh between element τ_3 and elements τ_1 and τ_2.

This gives

$$\varphi_3 \doteq \varphi_3 + \varphi_4 + \varphi_5$$

A representation of the clustered partition of unity is shown in Figure 10. Let us now show that this partition of unity is continuous. Let $\boldsymbol{t} \in \bar{\tau}_1 \cap \bar{\tau}_3$, as shown in Figure 10. If this function is computed from element τ_1 we have

$$\varphi_3|_{\tau_1}(\boldsymbol{t}) = N_3^{\tau_1}(\boldsymbol{t}) + N_4^{\tau_1}(\boldsymbol{t}) = 1$$

since the only non-zero shape functions of element τ_1 along the edge $\bar{\tau}_1 \cap \bar{\tau}_3$ are $N_3^{\tau_1}$ and $N_4^{\tau_1}$ and the shape functions of a Lagrangian finite element constitute a partition of unity. Similarly, if function φ_3 is computed from element τ_3 we have

$$\varphi_3|_{\tau_3}(\boldsymbol{t}) = N_3^{\tau_3}(\boldsymbol{t}) + N_5^{\tau_3}(\boldsymbol{t}) = 1$$

Therefore $\varphi_3|_{\tau_1}(\boldsymbol{t}) = \varphi_3|_{\tau_3}(\boldsymbol{t})$ and the function φ_3 is continuous at $\boldsymbol{t}$. The same argument can be used at any other point along the interface between element τ_3 and τ_1, τ_2.

As a second example, let us consider the finite element mesh shown in Figure 11. The global shape functions associated with nodes $1, \ldots, 7$ are discontinuous.

The discontinuous global functions in the mesh of Figure 11 can be clustered into a single function using equation (4) with

$$\begin{aligned} \mathcal{I}_{clustered} &= \{2, \ldots, 7\} \\ \mathcal{I}(1) &= \{1, \ldots, 7\} \\ \mathcal{I}(\alpha) &= \alpha : \ \alpha \notin \mathcal{I}_{clustered} \text{ and } \alpha \neq 1 \end{aligned}$$

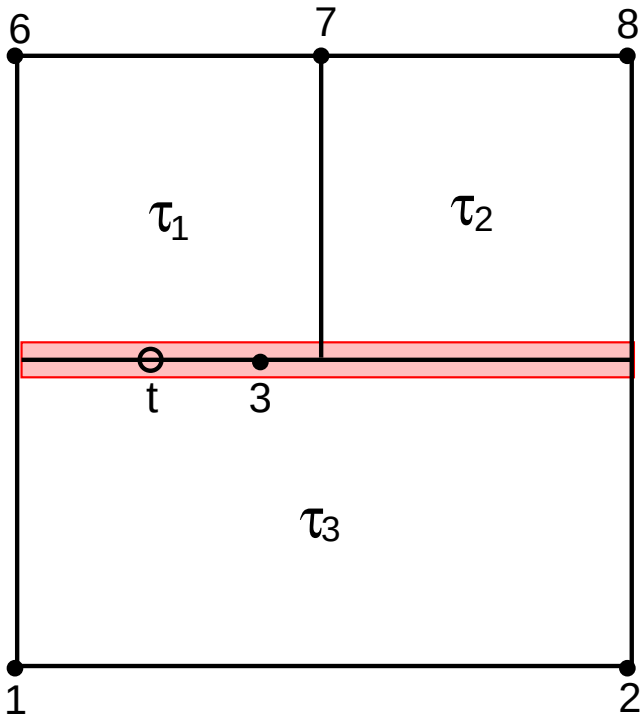


Figure 10: Representation of the clustered partition of unity used to handle the mismatched mesh of Figure 9. The partition of unity is identically equal to one along the mismatched interface and it is therefore continuous.

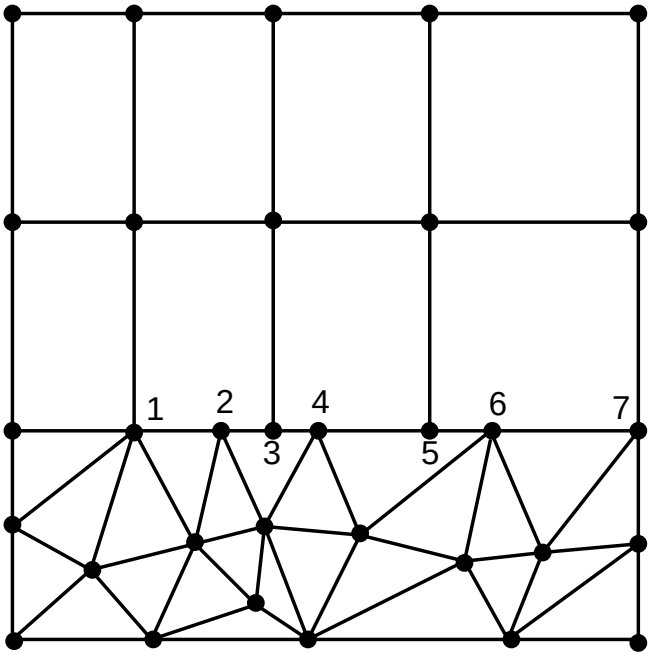


Figure 11: Mismatched mesh composed of quadrilateral and triangular elements.

This gives

$$\varphi_1 \doteq \sum_{\beta=1}^{7} \varphi_\beta$$

which can be shown to be continuous using the same arguments as in the previous example. A representation of the resulting clustered partition of unity is shown in Figure 12. This continuous partition of unity can then be used to build GFEM shape functions as described in Section 2.1.1.

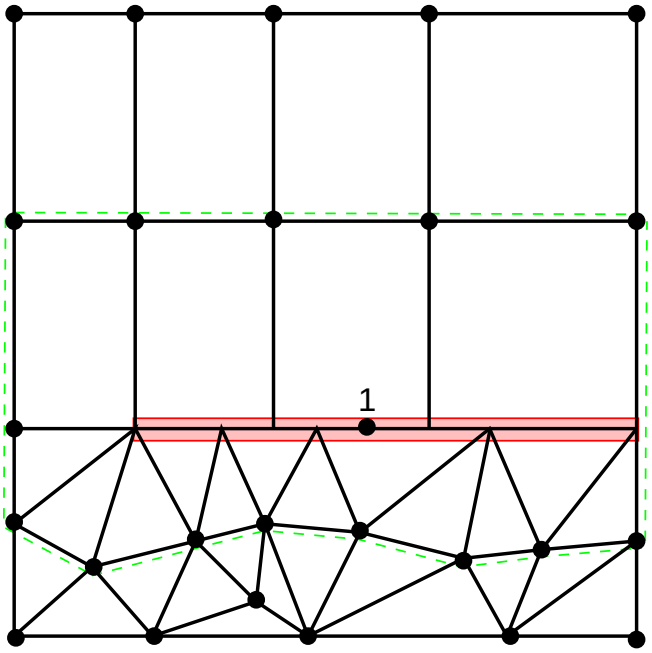


Figure 12: Representation of the clustered partition of unity used to handle the mismatched mesh of Figure 11. The dashed lines indicate the boundary of the support of the clustered function φ_1.

From the two previous examples, we can see that an algorithm to handle a mismatched mesh can be described as follows:

Let $\Gamma_m \subset \Omega$ be a mismatched interface. This is a curve in two-dimensional domains and a surface in the case of three-dimensional domains. Let

$$\mathcal{I}_m = \{\alpha \,:\, \varphi_\alpha(\boldsymbol{x}) \neq 0,\ \boldsymbol{x} \in \Gamma_m\}$$

denote the index set of partition of unity functions that are discontinuous due to the mismatched interface Γ_m. Let $\beta \in \mathcal{I}_m$. Then a continuous clustered partition of unity can be built using (4) with

$$\begin{aligned} \mathcal{I}_{clustered} &= \mathcal{I}_m - \beta \\ \mathcal{I}(\beta) &= \mathcal{I}_m \\ \mathcal{I}(\alpha) &= \alpha :\ \alpha \notin \mathcal{I}_{clustered} \text{ and } \alpha \neq \beta \end{aligned}$$

The technique described above is quite general and can be used with any type of element that can be unrefined using the method of Section 2.2. The technique however has some limitations that must be addressed before it can find broader use. The support of the clustered partition of unity function at a mis-

matched interface is equal to the union of the support of all originally discontinuous functions along the interface. In case of the clustered POU represented in Figure 12, for example, the support of the partition of unity function φ_1 is equal to the union of all elements sharing nodes $1, \ldots, 7$. This support is represented by the dashed lines in the figure. The GFE shape functions associated with φ_1 have the same support. Therefore, the approximation at a mismatched interface is analogous to a single finite element with size equal to the union of all elements with nodes on the interface. The approximation properties of the shape functions at the interface therefore deteriorates as the size of the mismatched interface grows. One approach to compensate this loss of accuracy is to increase the polynomial order of the shape functions at the interface. However this approach, in general, does not give satisfactory results if the solution of the problem has a singularity at the interface since polynomial functions cannot approximate singularities well. In the next section we discuss some approaches to handling this limitation.

2.4.1 Techniques to handle large mismatched interfaces

As discussed above, the clustering of nodes along and near a mismatched interface may in general give a poor representation of the solution in this region. This effect can be diminished by clustering only the nodes that are strictly necessary to make the approximation continuous across the mismatched interface. These are the nodes with index in the set $\mathcal{I}_m$ defined above. However, this does not guarantee that the supports of the shape functions at the interface are small enough to give acceptable results, and this is certainly not enough if the mismatched interface has points or lines of singularity.

Let φ_β be the clustered partition of unity function at a mismatched interface Γ_m and let ω_β denote its support. We can guarantee the quality of the GFEM approximation at a clustered mismatched mesh if we can somehow perform h-refinement of ω_β in addition to p-enrichment. This can be done with the aid of a mesh built on ω_β which is *independent* of the mesh used in Ω. This mesh can be built without much difficulty since *it has no relation with the mesh used in* Ω. In addition, ω_β is a small domain with, in general, a simple geometry. This approach is illustrated in Figure 13 (b). The support ω_1 of the clustered partition of unity function φ_1 is represented by the dashed lines. The triangularization used on ω_1 is shown on the right hand side of the figure. This triangularization can then be used to create shape functions defined on ω_1 and that are zero on $\partial\omega_1 \cap \Omega$. This last requirement can be achieved by not using the nodes on $\overline{\partial\omega_1 \cap \Omega}$.

The presence of different material properties across the mismatched interface surprisingly does not complicate the situation. This discontinuity can be handled by simply forcing the local mesh to take into account the material interface. One such example is illustrated in Figure 13 (c).

As a second example, let us consider the mismatched mesh of Figure 14. In this case, there are two point singularities at the mismatched interface and p-enrichment of the clustered partition of unity function φ_β can not resolve these features. A triangularization of ω_β is shown on the right hand side of the figure. The local mesh used does not match the mesh on Ω. The nodes not on $\overline{\partial\omega_\beta \cap \Omega}$ are indicate by solid dots. Note that the nodes located on the singular points can be used. This is in general the case. If the lines or points of singularities happen to be on $\overline{\partial\omega_\beta \cap \Omega}$, then ω_β can always be extended (by clumping more nodes) such that they belong to ω_β.

The examples above illustrate that the accuracy of the GFEM solution at the mismatched mesh can be controlled using the proposed technique. This demonstrates the unique features of this technique, namely stability and accuracy.

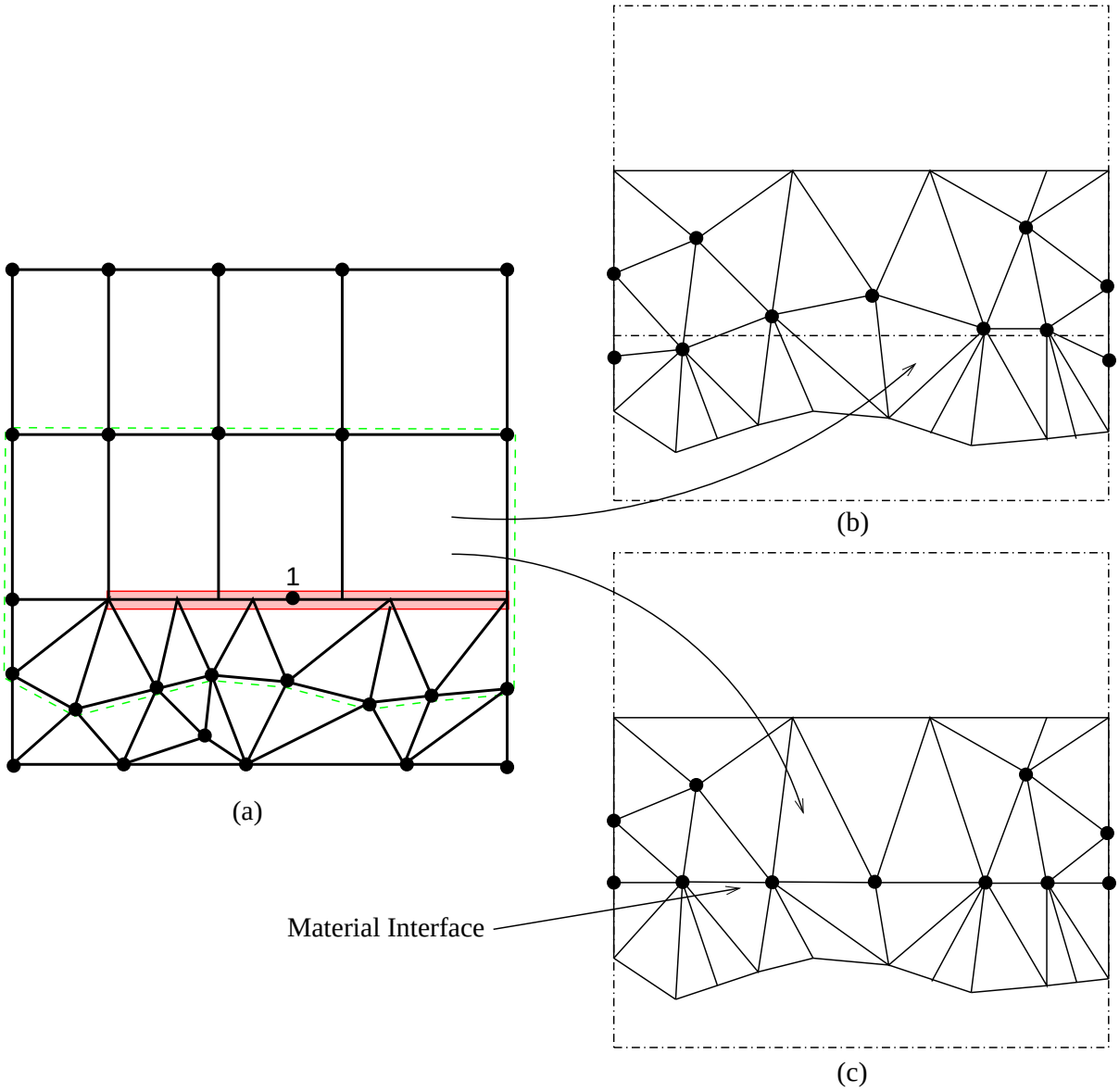


Figure 13: Local meshes used to provide global functions with small support across a mismatched interface in the case of continuous (b) and discontinuous (c) material properties across the interface. The vertex nodes with degrees of freedom are indicated by small solid circles. The meshes are independent of the one used in the domain Ω. They only need to cover the support ω_1 of the clustered partition of unity φ_1.

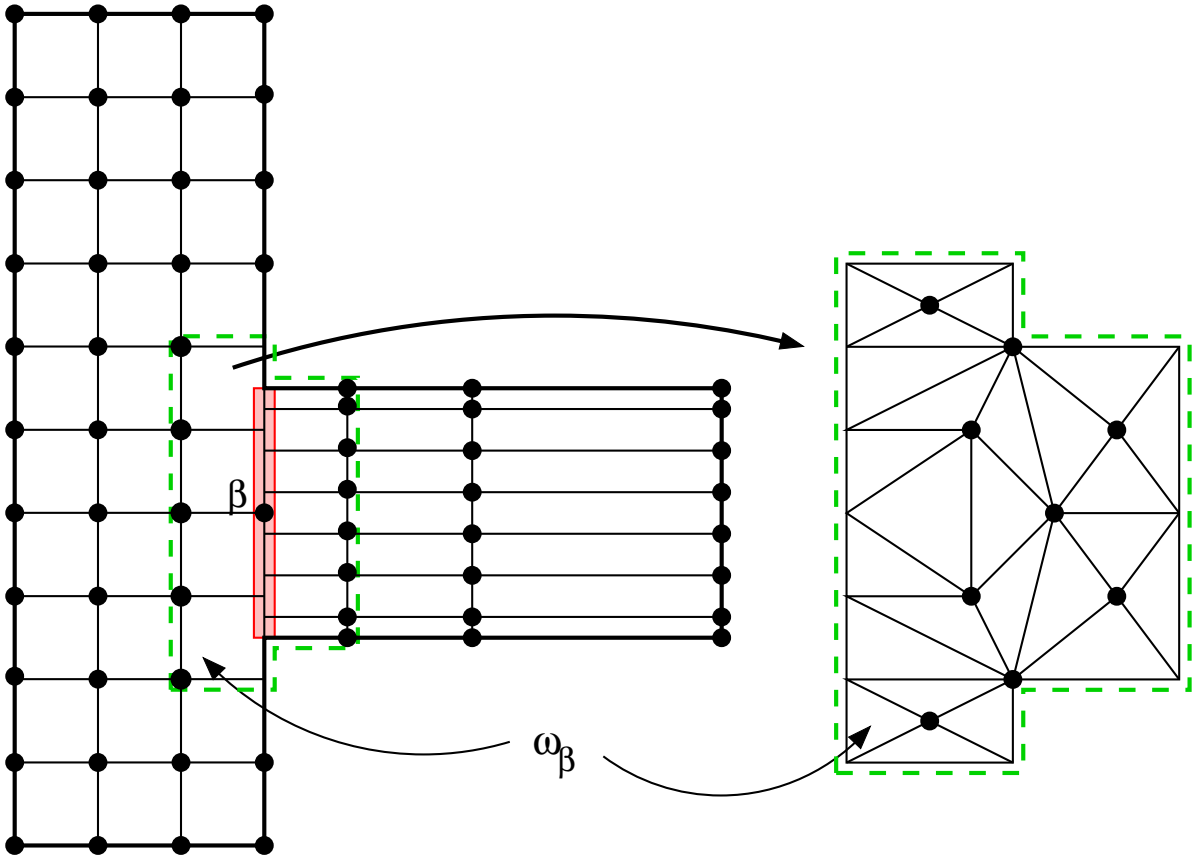


Figure 14: Local mesh used to provide global functions with small support across a mismatched interface. The vertex nodes with degrees of freedom are indicated by small solid circles. The local mesh is independent of the existing meshes at the subparts. This is in contrast with existing local remeshing approaches.

2.5 Elements of Unacceptable Quality

It is well known by finite element practitioners and from the theoretical point of view that the quality of a finite element approximation to the solution of boundary or initial value problems depends directly on the aspect ratio of the elements in the mesh. The aspect ratio of an element is defined as the ratio between the diameter of the smallest circle (or sphere) that circumscribes the element and the diameter of the largest circle (or sphere) that can be inscribed in the element.

Elements of unacceptable aspect ratio can appear in a finite element mesh for several reasons. During the mesh generation, for example, the transition zones between small and large elements, in general, contain elements with large aspect ratios. This can only be avoided, in most cases, by substantially increasing the number of elements in the mesh. Even meshes initially containing only elements of good quality can become unacceptable in the course of a simulation involving nodal movement. This happens, for example, during the shape optimization of mechanical parts or in the analysis of problems involving finite deformations using a Lagrangian formulation. A common approach to handle this situation is to remesh the entire domain whenever elements of unacceptable quality appear. This cannot always be done without the intervention of the user and in general leads to a loss of accuracy caused by mapping the solution between consecutive meshes.

In this section we describe how elements of unacceptable quality for the finite element method can be handled in the GFEM using a local version of the mesh unrefinement technique proposed in Section 2.2. We concentrate here on the case of elements with a Jacobian equal or close to zero at some point(s) in the element. This causes the transformation from the reference or master coordinate system to the physical or global coordinate system to be non-invertible. This inverse mapping is used by most finite element solvers to compute derivatives of the shape functions of the element in the physical directions. Elements with this type of problem arise quite often during automatic mesh generation around curved boundaries as illustrated in Figure 15. Most mesh generators for elements with quadratic geometry, first generate a mesh with straight edges or faces and then move the center nodes of the edges or faces to the actual geometry. This last step is very prone to deteriorating the quality of elements. An example of a mesh generated using this approach is shown on Figure 15. The edge of the hatched element that is on the curved boundary was moved from the dashed line to the position shown. The resulting element has a negative or zero Jacobian at several points and is therefore unacceptable.

Let us now consider more closely how the shape functions and their derivatives are computed in an element τ, in which all partition of unity functions (nodes) have been clustered into a single function (node) φ_β as in the case of the hatched element shown in Figure 7. Such an element is said to be clustered. The resulting partition of unity function over element τ is identically equal to one and the GFEM shape functions are given by

$$\phi_i^\beta = \varphi_\beta L_{i\beta} = L_{i\beta} \qquad i \in \mathcal{I}(\beta)$$

according to (3). The derivatives of these functions with respect to the global directions (x_1, x_2, x_3) are therefore given by

$$\frac{\partial \phi_i^\beta}{\partial x_a} = \frac{\partial L_{i\beta}}{\partial x_a} \qquad i \in \mathcal{I}(\beta), \;\; a = 1, 2, 3$$

The computation of the derivatives of enrichment functions $L_{i\beta}$ does not use the inverse of the Jacobian matrix since these functions are independent of the partition of unity. Therefore, the calculation of the GFEM shape functions and their derivatives over clustered elements is not affected by the quality of the elements. This property of the GFEM shape functions suggests the following strategy to handle elements of

unacceptable quality:

- Check the element Jacobians at sampling points and cluster those with near zero or negative values. Here, the sampling points can be, for example, the vertices of elements, integration points, etc.
- Check the quality of elements in a given mesh by computing the aspect ratio and/or any other measure of element quality and cluster the elements of unacceptable quality.
- Enrich the approximation at the nodes to the desired order using any one of the techniques described in Section 2.3.2.

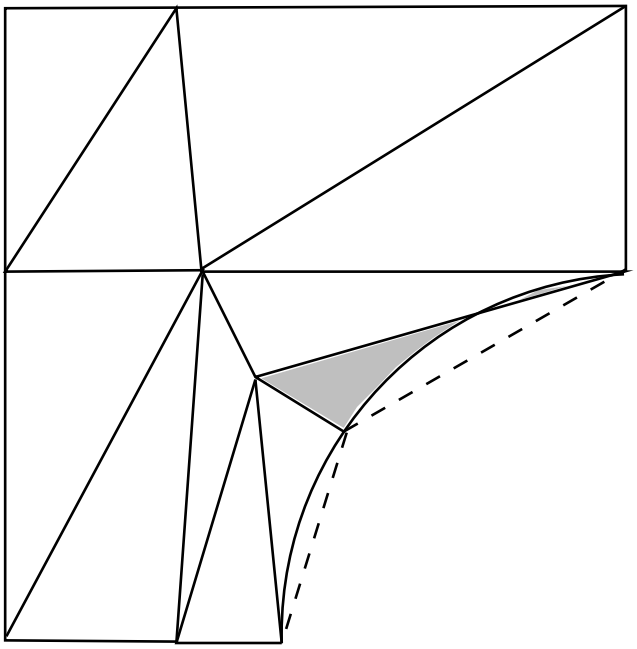


Figure 15: Element of unacceptable quality near a curved boundary. The mapping from the reference element to the hatched element cannot be inverted. If the element is clustered, however, the inverse mapping is not needed.

3 Computer implementation

The GFEMC algorithms described above, were implemented using ProPHLEX hp-adaptive finite element environment [23]. ProPHLEX is a customizable software library aimed at development of hp-adaptive finite element solvers. It consists of several ready ("black box") components needed for the solution of various problems that can be formulated as second order partial differential equations:

- hp-adaptive finite elements, easily customizable for arbitrary PDE's,
- error estimation and adaptivity algorithms,

- linear, nonlinear, and time-dependent algebraic solvers,
- generic discrete (not automatically adaptive) elements,
- object oriented, dynamic data structures,
- customizable input deck reader (proprietary PHLEX format and Nastran compatible format),
- customizable command line language, and
- built–in interactive 3d post-processor.

The GFEM with clustering was implemented in ProPHLEX using generic, discrete, finite element concepts, as a result only p-adaptivity and clustering are currently available (not traditional h-adaptivity).

The element formulation (numerical integration, shape functions, and PDE formulation) were implemented mimicking that of the hp-adaptive finite elements already present in ProPHLEX. As a result, most of the applications built on PHLEX can currently choose GFEMC as an alternative discretization method, and many structural mechanics examples can be solved using classical FE's, hp-adaptive technique and GFEMC using almost identical input data files. It is also possible to mix and match these three approaches, i.e. solve a single model containing seamless combination of all three types of elements.

3.1 Implementation of adaptivity algorithms

ProPHLEX code is not written using C++ or any other well known object oriented computer programming language. Its object oriented technology is implemented in C and Fortran with the help of custom preprocessor [1]. Thus the implementation details for GFEMC database are presented below as C–style `struct` code.

Two main OBJECTs defining the database in ProPHLEX relevant for the GFEM are `Gnode` and `GenEl`. `Gnode` (geometry node) represents a vertex of each element and stores nodal coordinates, solution components (Lagrange style DOF's) and some components of h-adaptive connectivities (Fig 16 a). `GenEl` OBJECT (generic discrete finite element - Fig 17 a) represents traditional finite element, with data members containing connectivities, physical/material properties, boundary condition, and some possible extensions for adaptivity (`family` points to an OBJECT linking to parent and children, in the refinement element tree). `Class` member points to a `GenClass` OBJECT, which is used mainly to reference methods needed for element assembly, physics equations, reader and post-processing methods, but it can also have methods `refine_elem`, `unrefine_elem`, and `enrich_elem`) which could be optionally coded to implement h- and p-adaptivity.

3.1.1 p-adaptivity

The above OBJECTs were expanded to handle p-adaptivity in the GFEM:

- `Gnode` was augmented with `Mnode` ("meshless node") (Fig. 16 b) to contain array of DOF's needed for all enrichment functions, and some additional flags defining family φ_α. `CBasis` and `IsAtCrackFront` are used for the special, non–polynomial enrichment functions, which will not be discussed in this paper (see, for example, [15, 24]).

```
OBJECT Gnode {
   int                    objname ;
   int                    NodeID ;
   int                    map ;
   double                 xyz[NDIM]; /* node coordinates */
   OBJECT Gconst *        gconst ;   /* h-adaptive constraint */
   OBJECT DOF *           dof ;      /* solution */
   OBJECT CIO *           coor_system ;
};

OBJECT Mnode {
   int                    objname ;
   OBJECT Gnode *         gnode ;
   OBJECT DOF **          dofs ;     /* array of DOFs */
   OBJECT Any_Object *    CBasis;    /* special functions */
   double                 size ;
   UBYTE                  ndofs;     /* number of unknowns */
   UBYTE                  weight_type ;
   UBYTE                  basis;     /* type of PoU basis */
   UBYTE                  IsAtCrackFront;
   UBYTE                  order[NDIM] ;  /* p_order */
   UBYTE                  shared ;
};
```

Figure 16: OBJECT for finite element's node, and its' expansion to handle GFEMC meshless nodal data

```
OBJECT GenEl {
   int               objname ;
   int               ElemID ;
   OBJECT Gnode **   gnodes ;          /* geometry vertices */
   OBJECT GenEl *    neigh [ NFACE ] ;    /* neighbors */
   OBJECT CIO *      material;
   OBJECT UserData * userElem;
   OBJECT GenClass * Class;            /* incl. object method's */
   OBJECT BCelem *   BCinfo ;
   OBJECT ErrEst *   ElemErr ;         /* error estimator data */
   OBJECT Any_Object * appData;
   OBJECT Genealogy * family;          /* for h-adaptivity */
   UBYTE             is_boundary ;
   UBYTE             geom_type;
   UBYTE             num_nodes;
   UBYTE             solver_flags;
};

OBJECT Cloud {
  int             objname;
  int             nMnodes;
  OBJECT Mnode **  m_nodes ;          /* solution nodes */
  OBJECT Any_Object * appData;
  UBYTE*          localMasterIndex ;
  int             NumICinClump;
  UBYTE           nMsSlvNodes;
  UBYTE           IntOrder[NDIM];     /* p_order */
  UBYTE           ICCrackness ;
};
```

Figure 17: OBJECT for generic element data in ProPHLEX, and its expansion to handle GFEMC element data.

- `GenEl` OBJECT was expanded with `Cloud`, (Fig. 17 b), which stores mainly connectivities to `Mnodes`, needed to evaluate shape functions during assembly and post-processing.

In addition the OBJECT method `enrich_elem` was coded in such a way, that a request to enrich an element is transferred to all connected `Mnodes`, which in turn are modified to contain at least all monomials needed for a specified p-order. After modifications of all `Mnodes` in an element, local member `order[]` is adjusted to contain the maximum p–order of all `Mnodes`- this value is needed e.g. to properly choose integration formula used during element assembly.

3.1.2 h-adaptivity

h-adaptivity can be implemented via `refine_elem` method in `GenClass` OBJECT. It is currently not implemented in the traditional element refinement sense – instead it is used during element clustering (see below).

3.1.3 c-adaptivity

All mesh modifications needed for mesh clustering (c-adaptivity) are implemented in the same way, as it does not really matter whether clustering was triggered by misaligned connectivities, negative Jacobian elements, or if it is used to coarsen the solution. The database modifications implemented for p-adaptivity in GFEM (`Mnode` and `Cloud`) are sufficient for the clustering, because they allow for full separation of:

- geometry definition, where element (i.e. integration cell) is defined by geometry nodes, and
- solution approximation, where solution patch (i.e. element or cluster of elements) is defined by mesh-less nodes (representing families of functions φ_α).

Elements which are fully "clustered–out" point to a single `Mnode`. For a cluster consisting of multiple elements, all `Clouds` point to the same `Mnode`. The position XYZ of this `Mnode` is arbitrarily adjusted to be close to the geometrical center of the cluster, thus minimizing ill–conditioning when higher order polynomials are used. All `Cloud`'s which point to a single `Mnode` (member `NumICinClump`$= 1$) use reduced number of integration points (lower order Gauss formulas) because their PoU is constant (does not contain linear x, y, z terms) and they are also shown during post-processing with lowered p–order.

Some basic refinement/unrefinement operations are as follows:

- `refine_elem` method, when applied to a cluster of multiple elements, deletes the single `Mnode`, replaces it with two or more `Mnodes`, arbitrarily located. All elements of the cluster are then reattached to new `Mnodes`.
- `refine_elem` method, when applied to a single element (cluster with one element) regenerates `Mnodes` associated with element vertices, thus creating "regular" GFEM element.
- `unrefine_elem` method performs actions opposite to `refine_elem`. In particular it can:
  - convert single "regular" GFEM element to a single element cluster (element with "flat" PoU),

– replace two or more elements or clusters with a single `Mnode`/`Cloud` pair.

Note that refinement in c-adaptivity (and also h-adaptivity for GFEM, when implemented) in general cannot be performed in such a way as to preserve exactly existing solution. This is in contrast to hp-adaptivity implemented in PHLEX, in which solution space before refinement is contained in the solution space after refinement. This property is quite important in time dependent solution sequences. In the GFEM, like in almost all meshless techniques, the solution needs to be recomputed after each adaptation, using L^2–projection or similar technique. The errors introduced in this step, and their possible effects on solution quality and stability need to be further investigated.

3.2 Voronoi clustering technique

Initial clustering in the mesh is based on any mesh partitioning technique appropriate for parallel FEM solver. Two such algorithms have been implemented: using space filling curves [34], and public domain Metis partitioner [21]. In both cases either nodes or elements can be send to partitioner to create cluster data. Metis partitioner produced usually more compact clusters, and node-based partitioning produced somewhat better clusters, and it was also algorithmically simpler, so this was selected as the default method in the code.

Voronoi based clustering is actually a smoothing algorithm, i.e. it starts with existing partitioning (e.g from Metis) and rearranges clusters by:

- identifying all `Mnodes` associates with clusters (one `Mnode` per cluster),
- adjusting all `Mnodes` to locations at the geometric center of each cluster, and
- disregarding all cluster connectivities, and reassigning each element to the "best" `Mnode`, based primarily on the distance between the `Mnode` and the element center.

This process is conceptually identical to generating Voronoi cells for all `Mnodes` and assigning to each cluster all elements falling inside Voronoi cells.

Voronoi smoothing can be applied to the initial (global) clustering, but it can also be used during the adaptivity, to readjust clusters after each adaptation step, which results in smoother mesh (more uniform cluster sizes).

3.2.1 Additional considerations during adaptivity in the GFEM with clustering

To ensure continuity of the solution and avoid unexpected local mis–behaving of the solution, clustering (as described above) needs to be adjusted to account for:

- presence of point boundary conditions,
- surface-based boundary conditions,
- interfaces with non-GFEM elements or pseudo–elements (RBE, MPC, etc)

- non–convex parts of the domain (see example 4.1.2)

Current implementation over conservatively prohibits clustering and limits p-adaptivity in all elements connected to boundary conditions and non–GFEM elements, although it is possible to relax this restriction for surface-based boundary conditions (e.g. pressures). This can be seen in the examples in the following section, when low number of requested clusters still uses relatively large number of equations.

The algorithms to properly handle non-convex domains also need to be further investigated.

3.2.2 Automatic handling of incorrect meshes

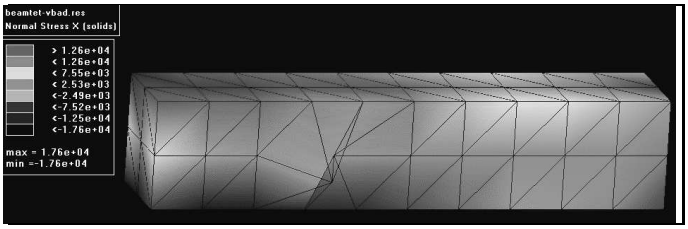


Figure 18: Beam model with severely misplaced node, resulting in elements with negative volume.

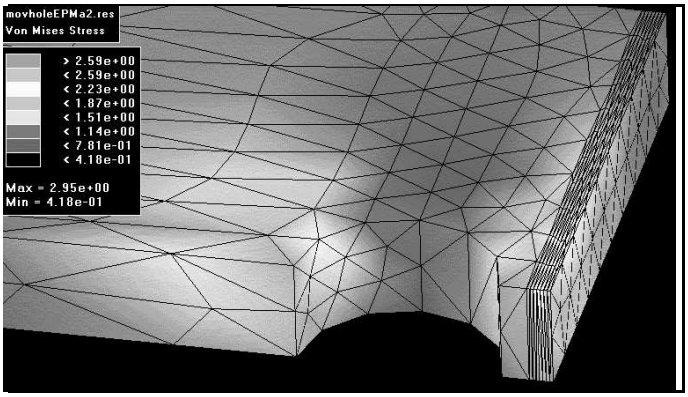


Figure 19: Elements distorted due to misplaced mid edge nodes. Global mesh seems to be correct because edges are customarily not drawn as curved.

The GFEM with clustering can handle several types of incorrect meshes, assuming that the mesh has properly defined connectivities and that the boundary geometry is approximated correctly. Additionally, with manual intervention, mesh with incorrect connectivities along the surface (misaligned mesh) can also be handled.

Most common examples of incorrect meshes result in negative Jacobian elements:

- Incorrectly placed node (Fig. 18) results in mesh “folded” on itself. Elements on two sides of the “fold” overlap, and the element(s) in between have negative Jacobian (negative volume). As a result, total volume may be computed correctly.
- Incorrectly placed mid-node for second order elements (Fig. 19 - 21). Jacobian changes sign within a single element, and parts of the element “fold” onto the neighboring elements.

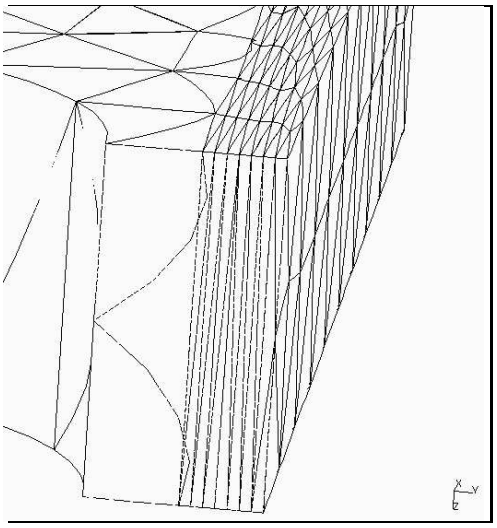


Figure 20: Elements distorted due to misplaced mid edge nodes. Zoom on the area with distorted elements.

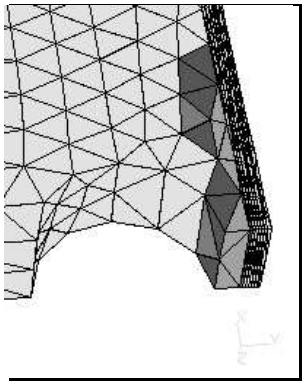


Figure 21: Elements distorted due to misplaced mid edge nodes. Element clusters encapsulating overlapping zones.

- Ill-conditioned elements (see 4.3.2), mostly due to bad aspect ratio, or large internal element angles. Some of these elements do not need special handling as the GFEM is less sensitive to bad aspect ratio elements, others, especially ones with large internal angles, have to be handled the same way as negative Jacobian elements.

In both cases the challenge is to detect the bad elements, and also properly determine the extent of overlapping regions. During initial mesh checks most finite element codes verify value of element Jacobian, the GFEM code tests this value at multiple points inside each element (using Gauss points and/or element vertices). All elements with bad Jacobian are automatically clustered with neighboring elements. The code properly handles "flipped" elements (clusters sufficient number of neighbors). Current code is limited to separated incorrect elements, i.e. cannot connect two negative element Jacobian elements sharing common node. In such case special command `element_patch` is used in the input deck, which defines list of nodes to be clustered, or bounding box surrounding the problem area.

Inconsistent mesh (see e.g. fig. 30) in general cannot be efficiently handled fully automatically. The brute force search for elements which are neighbors but do not share common vertices is possible, albeit costly, but it will of course generate incorrect results when mesh contains intentional discontinu-

ities (contact zones, or fracture mechanics problems). Such meshes are handled semi-automatically, using `element_patch` command.

Techniques to handle large mismatched interface presented in 2.4.1 have not been implemented in present version of the code.

4 Numerical Examples

Several examples were solved using the GFEM with clustering presented in previous sections. The CPU times, where reported, are in seconds for Athlon 2400 Linux computer. Most of examples presented are small enough such that the time spent in the element assembly is large compared with the linear solver time. The routines used for assembly were not optimized, thus CPU times are relative only, and in particular should not be used to judge the effectiveness of the method.

4.1 Clustering techniques

4.1.1 Bulky 3–d model

Simple three–dimensional model (Fig 22) was used to investigate the effects of mesh clustering. For the convergence tests several meshes with different number of tetrahedrals were used, ranging from 15527 up to 825996 elements. Large models could not be solved on 32-bit computer without clustering, also large number of clusters, especially for $p = 2$, required more memory than available. Note also that elements with prescribed displacements on their faces were excluded from clustering, thus even for very small number of clusters the number of linear equations was significant, especially for the large models. This can be improved with better handling of boundary conditions, and better clustering algorithms near the boundary. Figs. 23 and 24 show the effects of partitioning of the smallest mesh into 120 clusters, using Metis partitioning and Metis with Voronoi cells, respectively.

Tables 1, and 2 summarizes convergence of the solution with varying number of clusters and different approximation order p. Tables 3 and 4 show selected results for the eigenvalue problem. All times reported in this example are for Athlon 2000 computer. Since the maximum stress for this problem is not well defined due to the presence of singularities, total strain energy was used for comparison.

Figure 25 shows the unexpected stress distribution, when computed directly from derivatives of the displacements. By comparison with Fig. 23 one can observe that single layer of “full” GFEM elements, i.e., with all their nodes, interfacing clusters of elements acts as relatively soft “glue” and absorbs most of deformations resulting in thin bands of large stress. This should be resolved by smoothing stresses across larger patches, via appropriate stress extraction technique.

Figure 26 shows error distribution computed for the full GFEM model (approximation order $p = 2$). While actual value for the error is computed with unknown constant and cannot be used to estimate global error in the solution, error distribution is suitable for adaptation process to select candidate elements for (un)refinement or enrichment. For derivation of error estimator see [24].

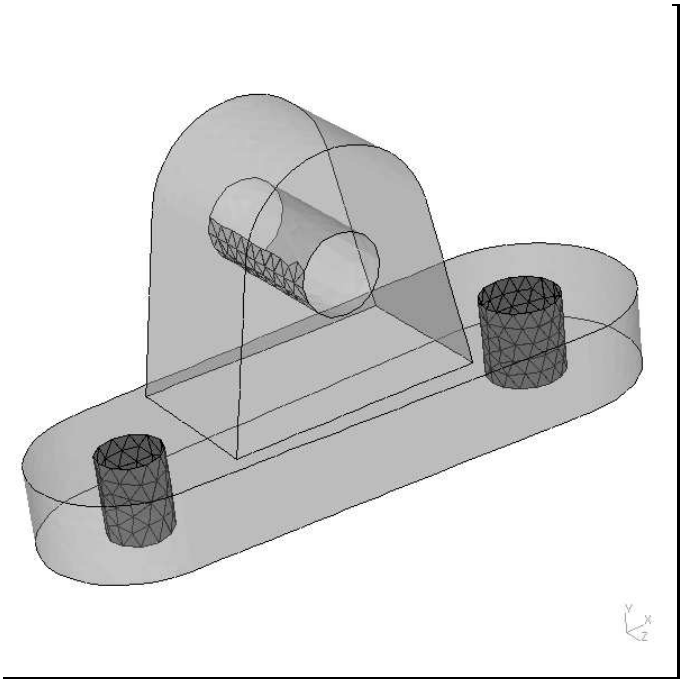


Figure 22: Bulky three–dimensional model. It is fully supported at perimeter of vertical openings, loaded with uniform normal pressure on a part of horizontal, upper openings

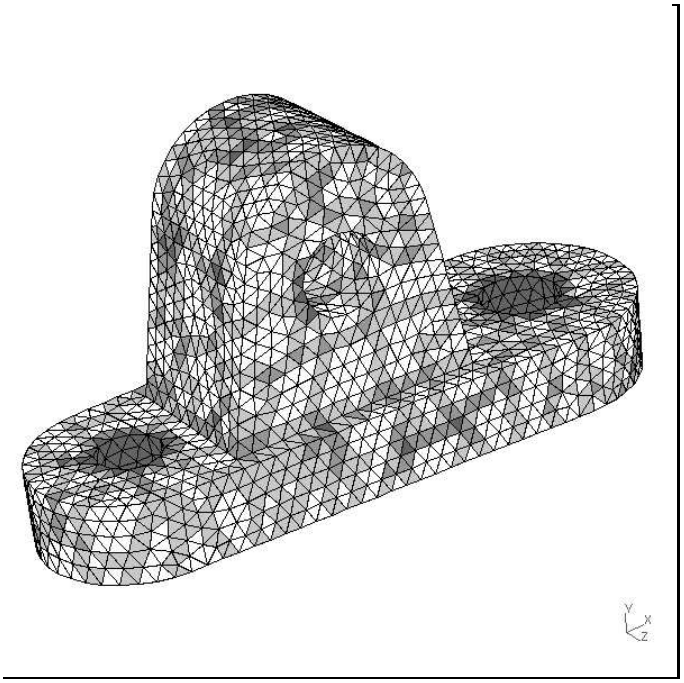


Figure 23: Metis based partitioning. White elements use “flat” PoU, i.e., all `Mnodes` clustered into a single `Mnode`, darkest ones have 4 distinct `Mnodes`, similarly to standard FEM.

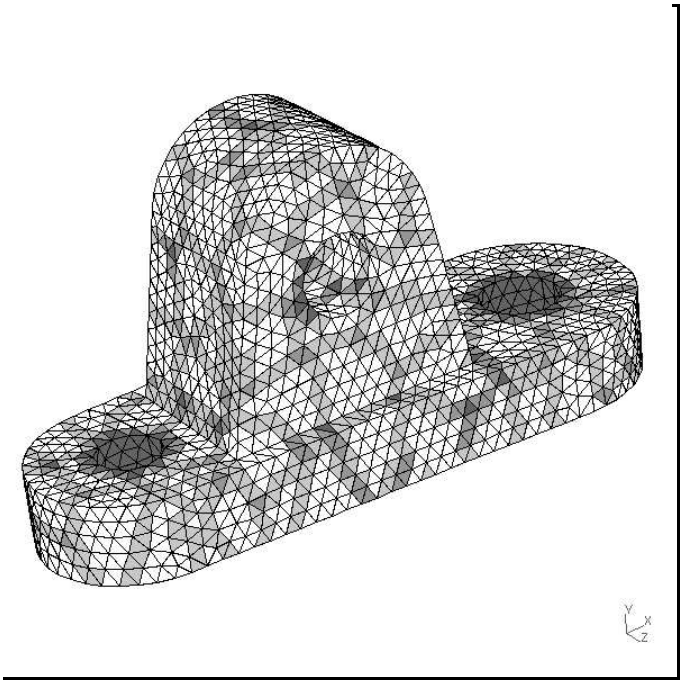


Figure 24: Metis based partitioning with Voronoi algorithm

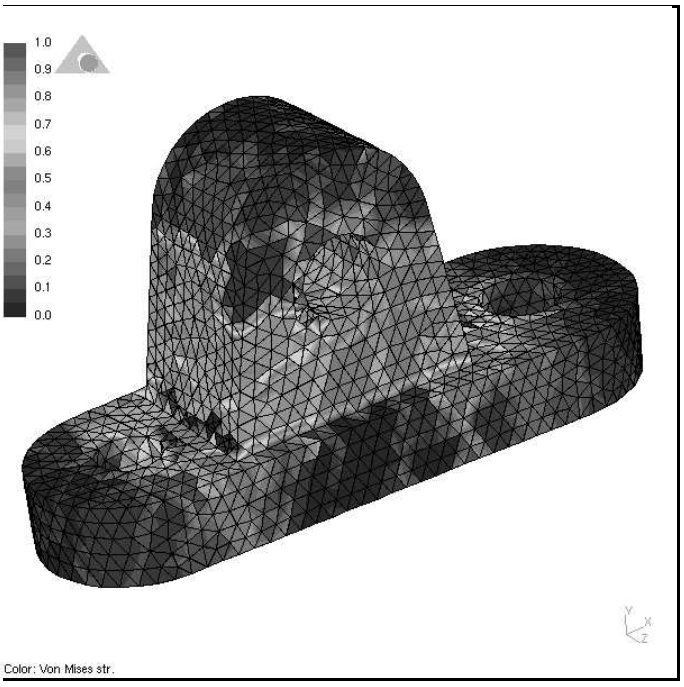


Figure 25: Stress distribution for partitioning in Fig 23 computed directly from pointwise derivatives of displacements.

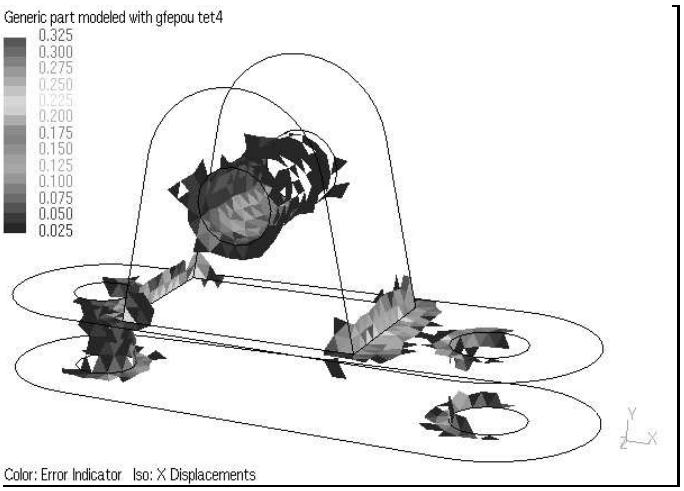


Figure 26: Error distribution for model without clustering.

# of clusters	# eqns	CPU time	RAM(MB)	Tot energy
p-order$= 1$				
20	18888	23.91	99	0.466014
200	21072	28.76	116	0.881289
2000	42312	89.98	105	1.188349
p-order$= 2$				
20	34296	82.78	137	0.910704
200	39738	109.32	169	1.202554
2000	92838			
no clustering				
$p = 1$	61041	60.05	70	1.289970
$p = 2$	244164	2018.01	549	1.351846

Table 1: Convergence of the solution for bulky three–dimensional model. Mesh with 92149 elements.

# of clusters	# eqns	CPU time	RAM(MB)	Tot energy
p-order$= 1$				
20	106116	332.57	328	0.397022
200	108300	350.08	334	0.806180
2000	129936	468.53	387	1.219049
p-order$= 2$				
20	191058	1133.50	808	0.877485
200	196518	927.34	843	1.258240
2000	250608			
no clustering				
$p = 1$	505209			
$p = 2$	2020836			

Table 2: Convergence of the solution for bulky three–dimensional model. Mesh with 820142 elements.

# of clusters	# eqns	CPU time	RAM(MB)	Eig 1 (Hz)	Eig 10 (Hz)
p-order$= 1$					
20	16104	43.13	128	68.1	418.8
200	18264	56.65	151	45.3	331.5
2000	39360	181.01	138	36.4	287.3
p-order$= 2$					
20	27318	193.30	160	41.1	322.7
200	32718	277.22	201	35.0	284.9
no clustering					
$p = 1$	241089	3810.55	957	33.5	267.1

Table 3: Convergence of eigenvalue solution for generic bulky model, 91124 tetrahedral elements. Full model could not be solved on 32-bit computer at $p = 2$.

# of clusters	# eqns	CPU time	RAM(MB)	Eig 1 (Hz)	Eig 10 (Hz)
p-order$= 1$					
20	91920	530.18	389	92.1	491.1
200	94068	582.44	397	54.8	370.2
2000	115836	806.50	473	38.9	296.9
p-order$= 2$					
20	155514	2242.82	970	47.8	331.9
200	160884	2016.43	1016	36.4	289.9

Table 4: Convergence of eigenvalue solution for generic bulky model, 825996 tetrahedral elements. Full model could not be solved on 32-bit computer even at $p = 1$.

4.1.2 Plate with T–section ribs

The following example tests clustering algorithm for non–convex 3–dimensional model: plate with T–shaped ribs (Fig. 27). The model has 720 linear brick elements (HEX8) and with low number of requested clusters it will create non–convex clusters, as shown in Fig. 28.

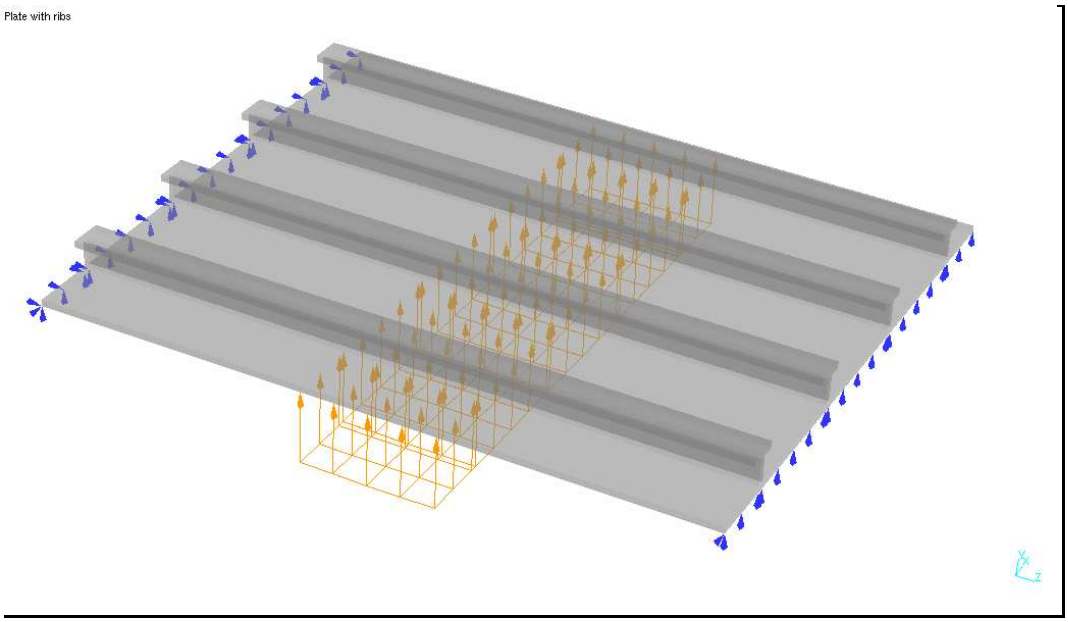


Figure 27: Plate model with T shaped ribs.

Although true adaptive solution would eventually detect and resolve singularities along all the T joints, the model is coarse enough that this behavior does not interfere with the results listed in the table. The reference solution is obtained using PHLEXsolid (hp-adaptive FE method) with the same mesh; all elements with edge support were excluded from adaptation. Maximum stress appears on the top surface of ribs (Fig. 29), and it is relatively smooth; this explains good convergence for stress in the table.

Table 5 shows convergence results for this example, and Table 6 shows the convergence of eigen-problem for the same model. It may be noted that results for low number of clusters with p-order equal 1 are very stiff – this is caused by clusters including void space in their span (e.g. two front–most clusters in Fig. 28). Such clusters behave as if the void had some fractional non–zero stiffness, because the shape function within the cluster are continuous in the void and in the elements next to the void. For such examples, the partitioning algorithm should detect non–convex clusters and automatically refine them.

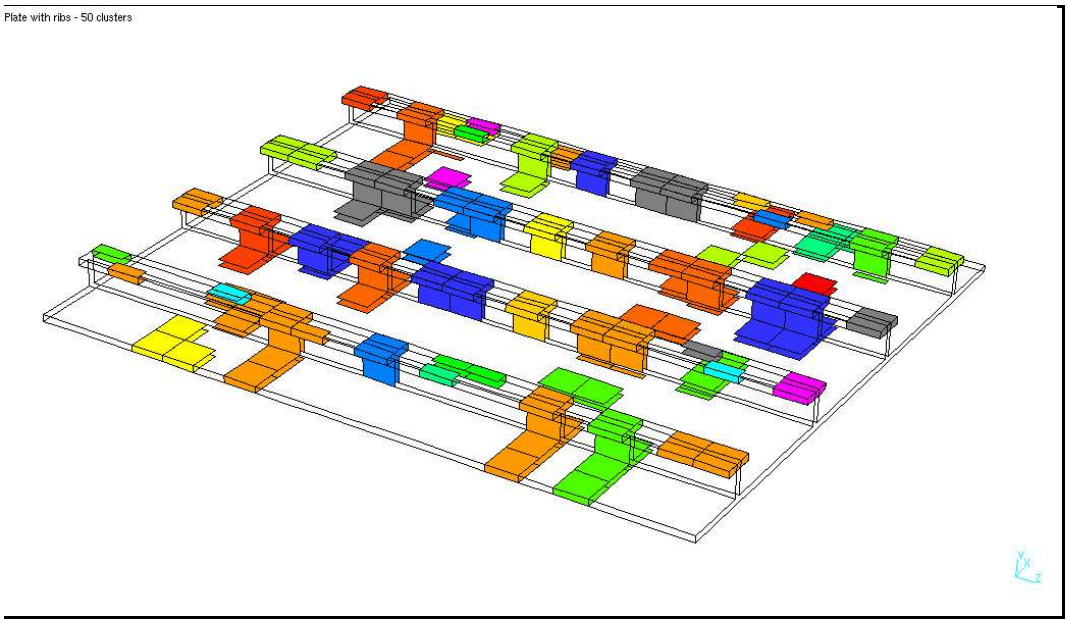


Figure 28: Plate with ribs. Cluster distribution for 50 requested clusters - The image shows only elements with single `Mnode`, actual clusters contain additional `Gnodes`.

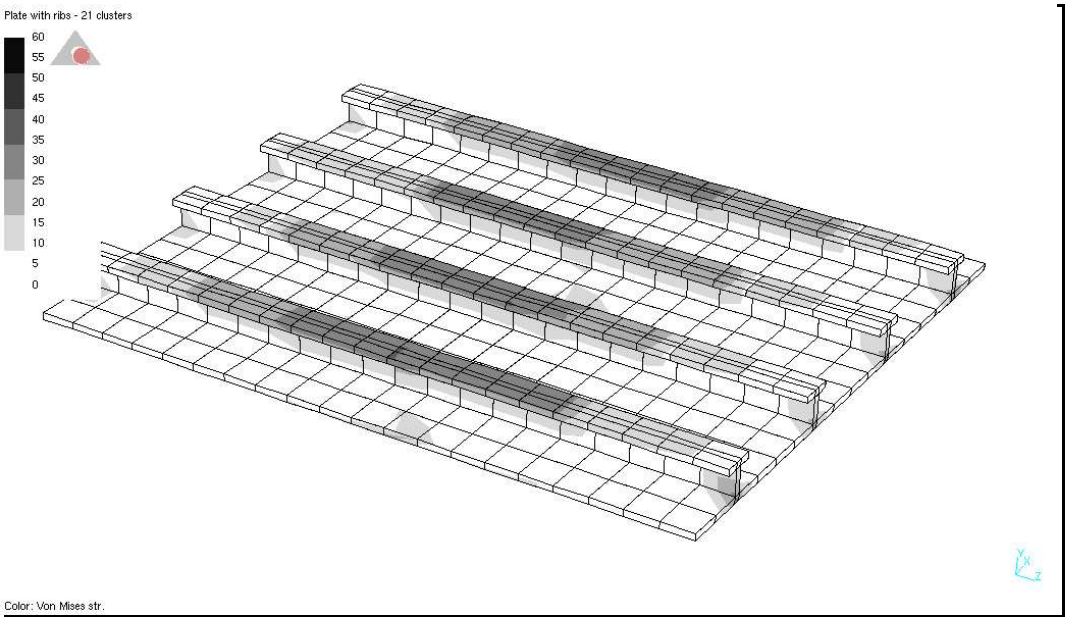


Figure 29: Plate with ribs. Stress distribution for 20 requested clusters.

# of clusters	# eqns	CPU time	Tot energy	Max displ	Max VM stress
Reference					
hp-FEM	86394		3.29949	0.0492413	36.0042
p-order$= 1$					
20	3540	7.37	0.37220	0.0052948	20.0547
50	3948	8.05	0.89328	0.0115606	20.9156
200	5820	9.83	2.35183	0.0314791	29.1563
500	8616	11.70	2.95466	0.0410756	31.7996
p-order$= 2$					
20	7626	25.39	2.90423	0.0376752	31.9445
50	8646	29.07	3.06821	0.0407663	33.5929
200	13326	41.19	3.21160	0.0453042	35.4000
500	20316	67.94	3.28508	0.0484317	35.0655
no clustering					
$p = 1$	4662	6.56	2.57123	0.0339121	27.8050
$p = 2$	18648	24.78	3.28089	0.0485858	35.0892

Table 5: Convergence of the solution for plate with ribs model

# of clusters	# eqns	CPU time	RAM(MB)	Eig 1 (kHz)	Eig 10 (kHz)
p-order$= 1$					
20	3540	6.12	17	3.639	19.628
50	3948	6.43	19	2.820	12.977
200	5820	8.79	31	1.763	10.446
500	8616	11.74	52	1.588	7.773
p-order$= 2$					
20	7626	24.20	75	1.575	10.678
50	8646	28.75	92	1.545	8.845
200	13326	42.73	155	1.523	5.964
500	20316	90.69	135	1.512	5.492
no clustering					
$p = 1$	4662	5.07	13	1.697	9.147
$p = 2$	18648	34.89	177	1.513	5.498

Table 6: Convergence of eigenvalue solution for plate with ribs model

4.2 Mismatched meshes

4.2.1 2.5–d plate with the hole

A three-dimensional model of a plate with the hole shown in Fig. 30 includes both hexahedral and very long tetrahedral elements, mismatched along a relatively large section of the domain. In spite of this discrepancy, the displacements and stresses calculated with the GFEM (see Fig. 31 and Fig32) exhibit very good symmetry. Also, the GFEM results compare favorably with an analytical solution: the calculated maximum longitudinal stress component σ_{xx} was equal 3.44, while the analytical value for two-dimensional problem is 3.25.

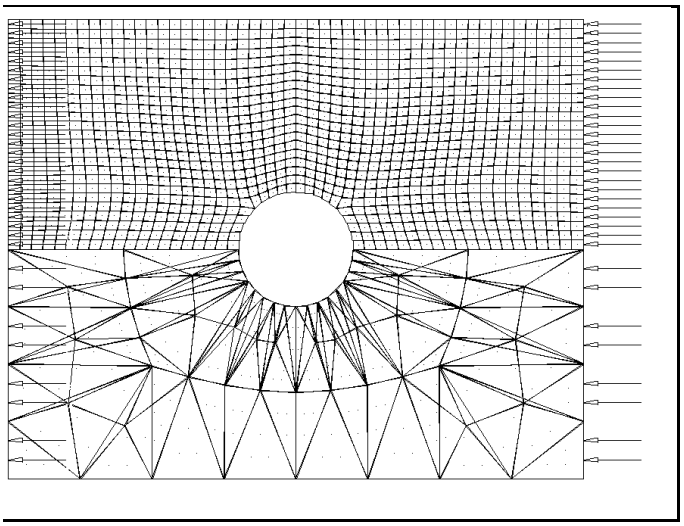


Figure 30: Plate with the hole with inconsistent mesh.

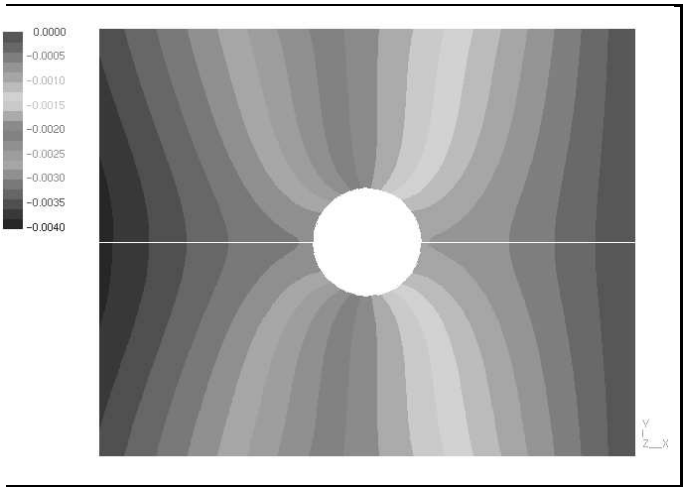


Figure 31: Plate with the hole with inconsistent mesh.Displacement distribution.

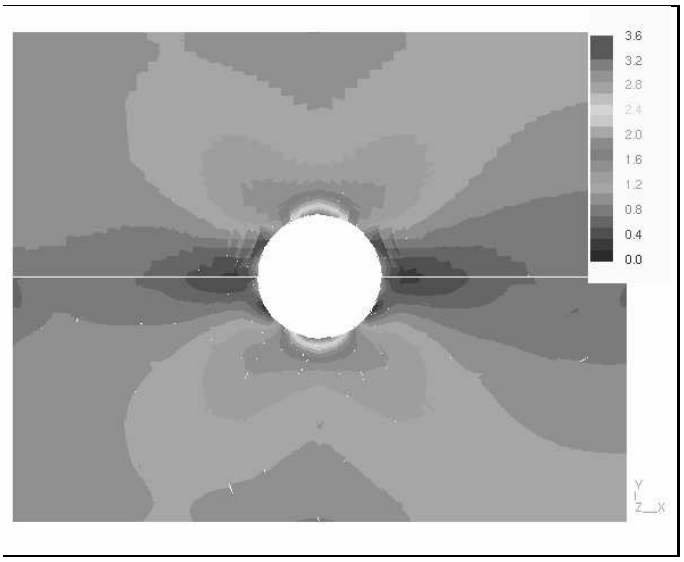


Figure 32: Plate with the hole with inconsistent mesh. Stress distribution.

4.2.2 T connected beams

The model of a T–shaped beam (Fig. 33) subjected to a bending load, with mesh inconsistencies present along three surfaces (see Fig.34, note that the mesh on the crossbar does not even place nodes on the inner T-corner). After user-defined identification of the mismatched surfaces (a task which can be automated) the domain was covered with GFEM discretization nodes (`Mnodes`) as shown in Fig. 35. The GFEM results (von Mises stress distribution) are shown in Fig. 36. The solution satisfies all the necessary continuity and consistency requirements. Moreover, it captures the stress concentration on or near the inner T-corner.

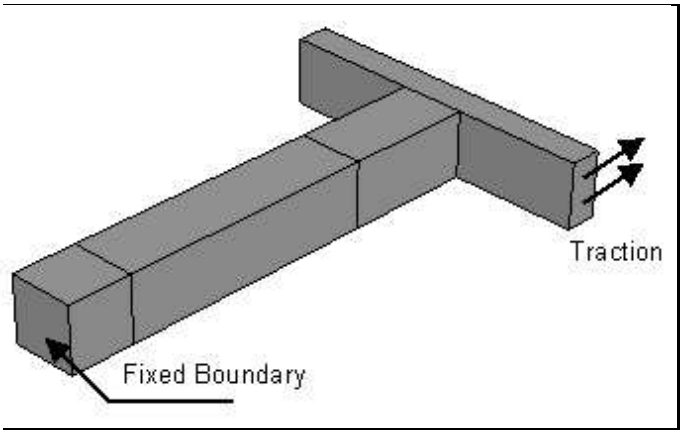


Figure 33: T-shaped domain under bending load.

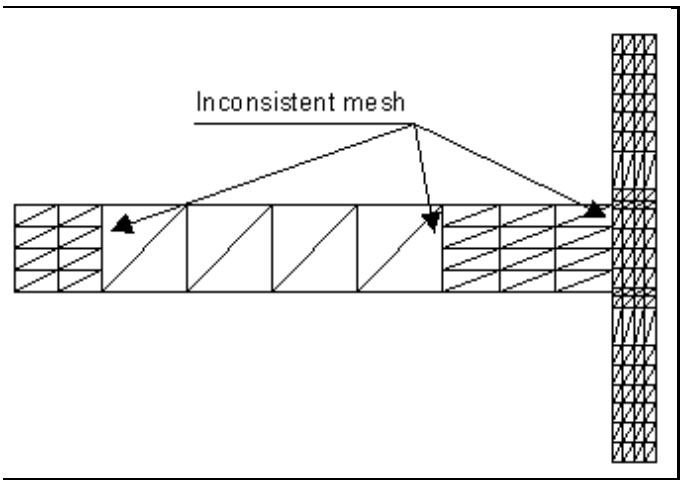


Figure 34: T-shaped domain under bending load: inconsistent tetrahedral mesh.

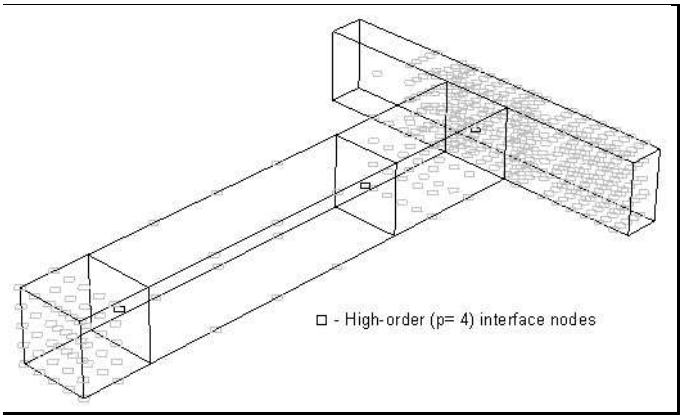


Figure 35: T-shaped domain under bending load: GFEM-`Mnodes`.

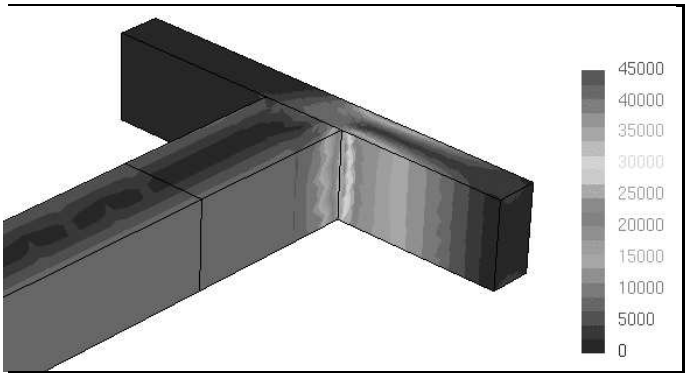


Figure 36: T-shaped domain under bending load: stress distribution.

4.3 Elements of unacceptable quality

All examples in this section can be solved and compared with various commercial FEA codes. Such comparisons were performed but are not described here. Some commercial codes do not detect such meshing errors, some others do and either refuse to solve or require special override command, while warning that results may be incorrect. Our GFEM code does not require such warning, because, with proper clustering, the formulation is mathematically correct, even with negative Jacobians.

4.3.1 Very long beam

A simple cantilever beam was solved using both finite element and GFE methods (Fig. 37). Table 7 summarizes the results for order of approximation $p = 3$; the stress σ_{xx} is reported at the center of top surface (away from singularities). It can be observed that, while the finite element solution, even with third order approximation, is too stiff, the GFEM method provides very accurate results, with and without clustering of degrees of freedom.

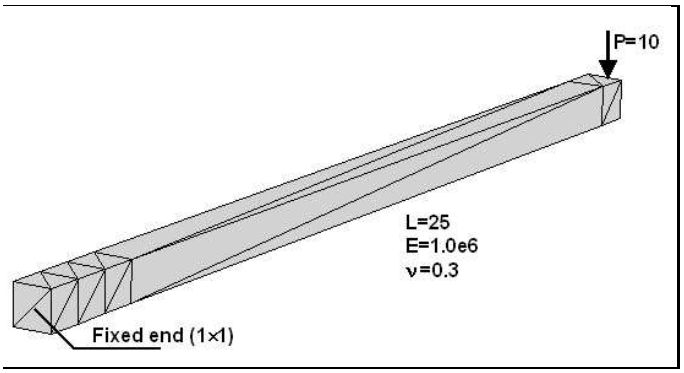


Figure 37: Simple cantilever beam, using tetrahedral elements with 1:20 aspect ratio.

Method:	FEM	GFEM full	GFEM w/ clustering	Exact
# eqns.	522	720	345	(beam theory)
Max. displ.	0.168	0.624	0.618	0.625
Strain energy	0.824	3.120	3.090	3.125
σ_{xx} at(12.5,0.5,1)	0.27	745	735	750

Table 7: Comparison of solutions for the cantilever beam.

4.3.2 Very long tetrahedral elements

In this example, a plate with the circular hole was solved. The mesh, shown in Fig. 38. used second order tetrahedral elements with straight edges. The aspect ratio of the worst element is $1/12.8$. The solution (tip displacement) obtained with GFEM is 1.006 of exact two-dimensional theoretical value, while the solution with standard TET10 elements was 0.942. GFEM elements allowed also selective enrichment of approximation; by using $p = 4$ in plane and $p = 2$ out–of–plane, the normalized stress solution was 0.999.

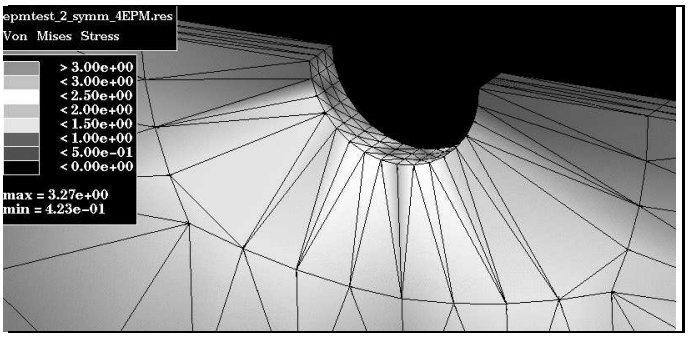


Figure 38: Plate with a hole: Elements with large aspect ratio.

4.3.3 Negative Jacobian elements

The thick shell model (hoop problem) was solved using second order tetrahedral elements with curved edges. The model shown in the Fig. 39. contains a quarter of the shell, with bottom edge supported and inner edge loaded with vertical tension opening the shell. The reference solution was obtained using PHLEXsolid hp-adaptive code, with hexahedral mesh. Elements on the left side of the Figure were longer than on the right side, and as a result of curved edges, 4 elements (shown in dark grey color) had negative value of Jacobian at some vertices. This problem was solved using the GFEM with clustering in the negative Jacobian area. With second order approximation, normalized peak stress along left plane of symmetry was 0.87 and 0.96 on the right side. By using third order approximation, the solution was 0.98 and 0.99 respectively.

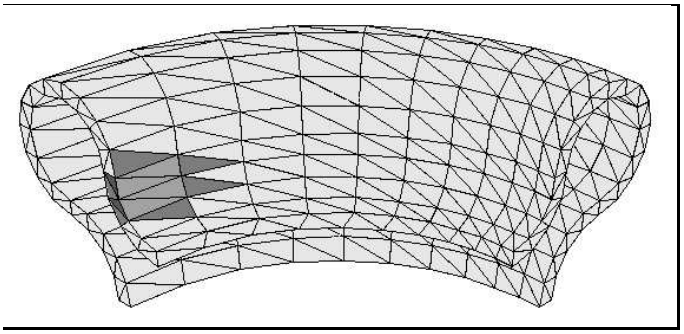


Figure 39: Thick shell model: curved elements with negative Jacobian. Clustered elements showed in dark grey.

4.3.4 Elements with zero Jacobian

The wheel rim model shown in Fig. 40. contains elements on the surface that had exactly zero volume. Such elements are often generated by an automatic mesh generators for highly curved non-convex domains. In this example five such elements were present. While most FE codes require manual “pruning” of bad elements, the GFEM produced correct results, without any difference between the results on the original and corrected mesh.

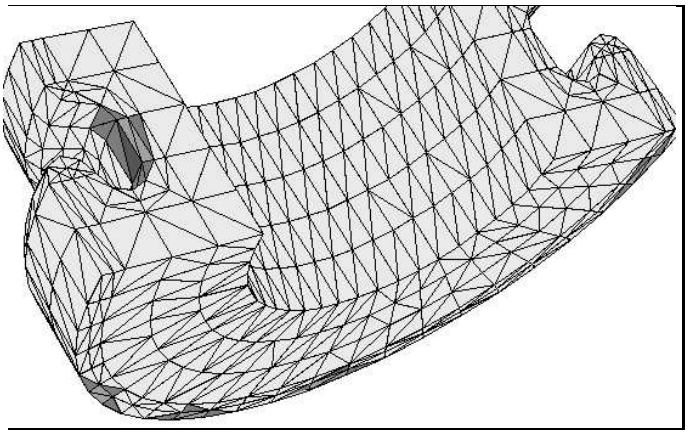


Figure 40: Wheel rim model: Zero Jacobian elements. 5 clusters around the faulty elements shown in dark grey.

5 Summary and Conclusions

The techniques presented in this paper can be immediately applied to any truly continuum finite element algorithms [1]:

- linear and non–linear,
- static, time–dependent, eigenvalue problems,
- two– and three– dimensional problems, even time–space elements, and higher dimensional problems.

Examples presented in this paper show that the GFEM with clustering can handle correctly most of finite element meshes, even those which, when used in standard FEA codes, could lead to erroneous results, or would require very time–consuming manual corrections. This capability makes the proposed method a very promising tool in the development of truly "meshless" application (i.e. FEA code with fully automatic meshing process hidden from the user).

CPU times and memory requirement presented in section 4.1, show that for the large mesh it is possible to reduce problem size and CPU time, in exchange for the accuracy of the solution. Thus eventually it should be possible to implement user controllable "slider", to select how much accuracy or how much computer resources should be used to solve a given problem.

The clustering technique applied for the coarsening of very large meshes showed to produce expected results, although actual computer implementation needs further development. In particular, the following aspects of the technique should be further investigated:

- *Improve clustering algorithms to detect and avoid non–convex clusters*: The quality of the numerical solution on clustered models is directly related to size and shape of the element clusters. In this regard new clustering algorithms must be identified which produce optimal element clusters. A discussion on this issue is presented in Section 2.3.1.

[1] as with any hp-adaptive FE code, it will require some modifi cations for underintegrated formulation

- *Implement stress extraction algorithms to avoid artificially high stresses in the zones between clusters*: The numerical solution for the primary variables (displacement field) on clustered meshes is reasonably smooth and continuous. However, the derivatives of the solution are poor. In the context of the linear statics test code these derivatives lead to unacceptable stress values.

- *Improve clustering algorithms to handle various types of boundary conditions*: Algorithms for special handling of Dirichlet and possibly non-homogeneous Neumann boundary conditions during clustering need to be devised. Without these algorithms the utility and performance of the method could be greatly affected.

- *Optimize generation of stiffness matrices for each cluster, to reduce unnecessary CPU time during assembly*: The overall performance of the solver on clustered meshes was found to scale very poorly in regard to the final model size. In fact, for clustered meshes that contained relatively small element clusters the overall performance was generally worse than solving the entire model. We believe this scaling can be significantly improved using specialized assembly techniques, optimized integration rules, special functions that satisfy the partial differential equations, and/or possibly through parallel processing as discussed in Section 2.3.3.

For models that have mis-matched meshes, "gluing" the sub-parts together using the partition of unity framework is not a trivial task. The primary open issues associated with this technique are related to handling the deterioration of the approximation properties of the GFEM shape functions at the interface as the size of the mis-match grows. Possible solutions for this problem are discussed in Section 2.4.1.

Finally, regarding the acceptance of poor quality meshes and even bad meshes, the open issues here are primarily related to local mesh clustering (as opposed to global mesh clustering for solver performance) and the quality/stability of the numerical approximation that can be expected in clustered regions.

References

[1] Altair Engineering, Inc. Database management for computational mechanics. `http://www.tx.altair.com/technology/technology.html#DSMAKE`.

[2] I. Babuška, G. Caloz, and Osborn J. E. Special finite element methods for a class of second order elliptic problems with rough coefficients. *SIAM J. Numerical Analysis*, 31(4):745–981, 1994.

[3] I. Babuška and J. M. Melenk. The partition of unity finite element method. *International Journal for Numerical Methods in Engineering*, 40:727–758, 1997.

[4] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[5] G. F. Carey and J. T. Oden. *Texas Finite Element Series Volume II–A Second Course*. Prentice-Hall, New Jersey, 1983.

[6] A. Plaza de la Hoz. The fractal behaviour of triangular refined/derefined meshes. *Communications in Numerical Methods in Engineering*, 12(5):295–302, 1996.

[7] C. R. Dohrmann, S. W. Key, and M. W. Heinstein. A method for connecting dissimilar finite element meshes in two dimensions. *International Journal for Numerical Methods in Engineering*, 48:655–678, 2000.

[8] C. R. Dohrmann, S. W. Key, and M. W. Heinstein. Methods for connecting dissimilar three-dimensional finite element meshes. *International Journal for Numerical Methods in Engineering*, 47:1057–1080, 2000.

[9] C. A. Duarte and I. Babuška. Mesh-independent directional *p*-enrichment using the generalized finite element method. In K. J. Bathe, editor, *Computational Fluid and Solid Mechanics*, volume 2, pages 1555–1558. Elsevier, June 2001. Proceedings of First MIT Conference on Computational Fluid and Solid Mechanics, June 12-15, 2001.

[10] C. A. Duarte and I. Babuška. Mesh-independent directional *p*-enrichment using the generalized finite element method. In Gregory M. Hulbert, editor, *Sixth U.S. National Congress on Computational Mechanics*, page 48, Dearborn, Michigan, August 2001.

[11] C. A. Duarte and I. Babuška. Mesh-independent directional *p*-enrichment using the generalized finite element method. *International Journal for Numerical Methods in Engineering*, 55(12):1477–1492, 2002.

[12] C. A. Duarte, I. Babuška, and J. T. Oden. Generalized finite element methods for three dimensional structural mechanics problems. In S. N. Atluri and P. E. O'Donoghue, editors, *Modeling and Simulation Based Engineering*, volume I, pages 53–58. Tech Science Press, October 1998. Proceedings of the International Conference on Computational Engineering Science, Atlanta, GA, October 5-9, 1998.

[13] C. A. Duarte, I. Babuška, and J. T. Oden. Generalized finite element methods for three dimensional structural mechanics problems. *Computers and Structures*, 77:215–232, 2000.

[14] C. A. Duarte, O. N. Hamzeh, and T. J. Liszka. A generalized finite element method for the simulation of three-dimensional dynamic crack propagation. In Z. Waszczyszyn and J. Pamin, editors, *Second European Conference on Computational Mechanics*, volume 1, pages 208–209, Cracow, Poland, June 2001. Fundacja Zdrowia Publicznego.

[15] C. A. Duarte, O. N. Hamzeh, T. J. Liszka, and W. W. Tworzydlo. A generalized finite element method for the simulation of three-dimensional dynamic crack propagation. *Computer Methods in Applied Mechanics and Engineering*, 190:2227–2262, 2001.

[16] C. A. M. Duarte and J. T. Oden. Hp clouds–a meshless method to solve boundary-value problems. Technical Report 95-05, TICAM, The University of Texas at Austin, May 1995.

[17] C. A. M. Duarte and J. T. Oden. An *hp* adaptive method using clouds. *Computer Methods in Applied Mechanics and Engineering*, 139:237–262, 1996.

[18] C. A. M. Duarte and J. T. Oden. *Hp* clouds—an *hp* meshless method. *Numerical Methods for Partial Differential Equations*, 12:673–705, 1996.

[19] C. Armando Duarte. *The hp Cloud Method*. PhD dissertation, The University of Texas at Austin, December 1996. Austin, TX, USA.

[20] N. V. Hattangady. Coarsening of mesh models for representation of rigid objects in finite element analysis. *International Journal for Numerical Methods in Engineering*, 44:313–326, 1999.

[21] George Karypis. `http://www-users.cs.umn.edu/~karypis/metis/index.html`.

[22] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981.

[23] T.J. Liszka, W.W. Tworzydlo, J.M. Bass, S.K. Sharma, T.A. Westermann, and B.B. Yavari. Prophlex – an hp-adaptive finite element kernel for solving coupled systems of partial differential equations in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 150:251–271, 1997.

[24] T.J. Liszka (PI). Hp-meshless cloud method for dynamic fracture in fluid structure interaction. Technical Report N00014-95-C-0373 (Office of Naval Research), The Computational Mechanics Company, Inc., Austin, TX, 2000.

[25] W. K. Liu, Y. Chen, S. Jun, T. Belytschko, C. Pan, R. A. Uras, and C. T. Chang. *Overview and Applications of the Reproducing Kernel Particle Methods*. CIMNE, 1996.

[26] W. K. Liu, S. Jun, and Y. F. Zhang. Reproducing kernel particle methods. *International Journal for Numerical Methods in Engineering*, 20:1081–1106, 1995.

[27] J. M. Melenk. *On Generalized Finite Element Methods*. PhD thesis, The University of Maryland, 1995.

[28] J. M. Melenk and I. Babuška. The partition of unity finite element method: Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139:289–314, 1996.

[29] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46:131–150, 1999.

[30] J. T. Oden and G. F. Carey. *Texas Finite Element Series Volume IV–Mathematical Aspects*. Prentice-Hall, New Jersey, 1983.

[31] J. T. Oden, C. A. Duarte, and O. C. Zienkiewicz. A new cloud-based *hp* finite element method. Technical report, TICAM Report 96-55, The University of Texas at Austin, Austin, Texas, USA, December 1996.

[32] J. T. Oden, C. A. Duarte, and O. C. Zienkiewicz. A new cloud-based *hp* finite element method. *Computer Methods in Applied Mechanics and Engineering*, 153:117–126, 1998.

[33] Angel Plaza, Miguel A. Padron, and Graham F. Carey. A 3d derefinement algorithm for tetrahedral grids. In *Trends in Unstructured Mesh Generation*, AMD-Vol. 220, pages 17–23. ASME, 1997.

[34] Hans Sagan. *Space Filing Curves*. Springer-Verlag, Berlin, Heidelberg, New York, 1994.

[35] T. Strouboulis, I. Babuška, and K. Copps. The design and analysis of the generalized finite element mehtod. *Computer Methods in Applied Mechanics and Engineering*, 81(1–3):43–69, 2000.

[36] T. Strouboulis, K. Copps, and Babuška I. The generalized finite element method: An example of its implementation and illustration of its performance. *International Journal for Numerical Methods in Engineering*, 47(8):1401–1417, 2000.

[37] T. Strouboulis, K. Copps, and Babuška I. The generalized finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190:4081–4193, 2001.

[38] N. Sukumar, N. Moës, B. Moran, and T. Belytschko. Extended finite element method for three-dimensional crack modelling. *International Journal for Numerical Methods in Engineering*, 48(11):1549–1570, 2000.