Latin American Journal of
Solids and Structures
www.lajss.org

An Object-Oriented Class Organization for Global-Local Generalized Finite Element Method

Abstract

This paper shows and discusses a generic implementation of the global-local analysis toward generalized finite element method (GFEM^{gl}). This implementation, performed into an academic computational platform, follows the object-oriented approach presented by the authors in a previous work for the standard version of GFEM in which the shape functions of finite elements are hierarchically enriched by analytical functions, according to the problem behavior. In global-local GFEM, however, the enrichment functions are constructed numerically from the solution of a local problem. This strategy allows the use of a coarse mesh even when the problem produces complex stress distributions. On the other hand, a local problem is defined where the stress field presents high gradients and it is discretized using a large number of elements. The results of the local problem are used to enrich the global problem which improves the approximate solution. The great advantage is allowing a well-refined description of the local problem, when necessary, avoiding an overburden for the computation of the global solution. Details of the implementation are presented and important aspects of using this strategy are highlighted in the numerical examples.

Keywords

Generalized finite element method, eXtended finite element method, Object-oriented programming, Global-local, Enrichment function

Mohammad Malekan [a]
Felício B. Barros [b]
Roque L. S. Pitangueira [c]
Phillipe D. Alves [d]

[a] Graduate Program in Structural Engineering (PROPEEs), School of Engineering, Federal University of Minas Gerais (UFMG), Belo Horizonte, MG, Brazil, mmalekan1986@gmail.com
[b] felicio@dees.ufmg.br
[c] roque@dees.ufmg.br
[d] Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, 205 North Mathews Avenue, Urbana, IL, USA, phpalves@yahoo.com.br

http://dx.doi.org/10.1590/1679-78252832

Received 02.02.2016
In revised form 06.07.2016
Accepted 07.07.2016
Available online 03.08.2016

1 INTRODUCTION

Finite element method (FEM) is a powerful method to solve various physical problems. This method is robust and has been thoroughly developed to solve engineering and applied science problems such as linear and non-linear stress analysis of solid and structures, electromagnetism, fluid flows and similar fields. In generalized finite element method (GFEM), Melenk and Babuška (1996), Strouboulis et al. (2000a), Duarte et al. (2000), Strouboulis et al. (2001), as in FEM, the approximation is built over a mesh of elements using interpolation functions. However, the approximation is associated with nodal points as in the Meshless methods, and it is enriched in the same fashion as in the *hp*-cloud method, Duarte et el. (1995), Duarte (1996), Liszka et al. (1996), Oden et al. (1998). Special functions multiply the original FEM functions and smooth and non-smooth solutions can be modeled independently of the mesh. The enrichment strategy of GFEM is similar to the one used in the extended finite element method (XFEM), introduced in Belytschko and Black (1999).

Global-local FEM was proposed by Noor (1986) in order to solve non-linear problems. This method was presented after proposing zooming method by Hirai et al. (1985). A local problem is defined where a local phenomenon happens. The global-local FEM approach has two steps. The first step is done with a coarse FEM mesh that ignores the effect of the local phenomena. This is followed by the second step which includes an analysis of the local region using refined finite element meshes. The key parameter for the local analyses is the application of field state variables as boundary conditions on the local boundaries. Once the solution of local problem is obtained, the global-local FEM analysis can be finished. In global-local GFEM (GFEM^{gl}), Duarte et al. (2007), a variation of the standard GFEM, enrichment functions are constructed numerically from the solution of a local problem. GFEM^{gl} approach has three steps, its first and second steps are the same as the global-local FEM. In the third step, the results of the local problem are used to enrich the global problem which improves the approximate solution. In the case of fracture mechanic problems, the stress field around the crack tip presents high gradients and it is discretized using a large number of elements. The great advantage is providing a well-refined description of the local problem. More information on global-local GFEM, considering its various aspects can be found in Duarte and Kim (2008), Kim et al. (2008), Kim et al. (2009), Kim et al. (2010), Pereira et al. (2011), Gupta et al. (2012), Kim et al. (2012), Gupta et al. (2013), Evangelista et al. (2013), Plews and Duarte (2015).

According to Alves et al. (2013), the similarity between GFEM/XFEM and FEM can allow a straightforward migration for these methods and also the reuse of the FEM structure. This migration and code reuse depend on how the FEM code is implemented, and a successful strategy is the object-oriented approach. Following such strategy several implementations have been proposed, Strouboulis et al. (2000a), Strouboulis et al. (2001), Bordas et al. (2007), Dunant et al. (2007), Nistor et al. (2008), Chamrová and Patzák (2010), Neto et al. (2013). In Alves et al. (2013), the FEM programming environment proposed by Fonseca and Pitangueira (2007) is expanded to enclose the standard version of GFEM. This environment, so called INSANE (Interactive Structural Analysis Environment) is an open source software available at *http://www.insane.dees.ufmg.br* and written in Java language. In the present paper, a new expansion that includes the GFEM^{gl} is presented. The implementation, conducted through the development of comprehensive object-oriented design, allows generalization of the global-local approach in such way that any types of partition of unity methods, analysis model and enrichment strategy can be combined. It also allows that stress and strain field can be applied as boundary conditions for local problems in addition to the capability of applying displacement boundary condition. Details of the implementation are discussed and important aspects of using these strategies are highlighted in numerical examples. This work is organized as follows: A brief explanation of GFEM is presented in section 2. In section 3, main ideas and formulations of global-local GFEM is presented. The INSANE class organization and its expansion to include GFEM^{gl} method are discussed in section 4. In the section 5, numerical examples are presented, and final section is devoted to concluding remarks and discussion.

2 THE GENERALIZED FINITE ELEMENT METHOD

A brief explanation of GFEM is described in this section. Further details can be found in Melenk and Babuška (1995), Babuška and Melenk (1997), Oden et al. (1998), Duarte et al. (2000), Strouboulis et al. (2001), Pereira et al. (2011). The GFEM was developed for modeling structural problems with discontinuities. The origins of this method can be summarized as:

- Research done by Babuška and co-workers – initially named as special finite element method by Babuška et al. (1994) and later as partition of unity method (PUM) by Melenk and Babuška (1996).
- Then, as a meshless formulation in the hp-cloud method, Duarte et al. (1995), and later as a hybrid approach with the FEM, Oden et al. (1998).

Furthermore, it can be considered an instance of the PUM, in the sense that it employs a set of PU functions to guarantee interelement continuity. Such strategy creates conforming approximations which are improved by a nodal enrichment scheme. This basic idea shares the same characteristics from XFEM proposed by Belytschko and Black (1999), as it is observed in Fries and Belytschko (2010).

Here this procedure is summarized following Barros et al. (2004, 2013) and considering two-dimensional space. A conventional finite elements mesh can be considered for which $\{K_e\}_{e=1}^{NE}$ is a set of NE elements, defined by N nodes, $\{\boldsymbol{x}_j\}_{j=1}^{N}$.

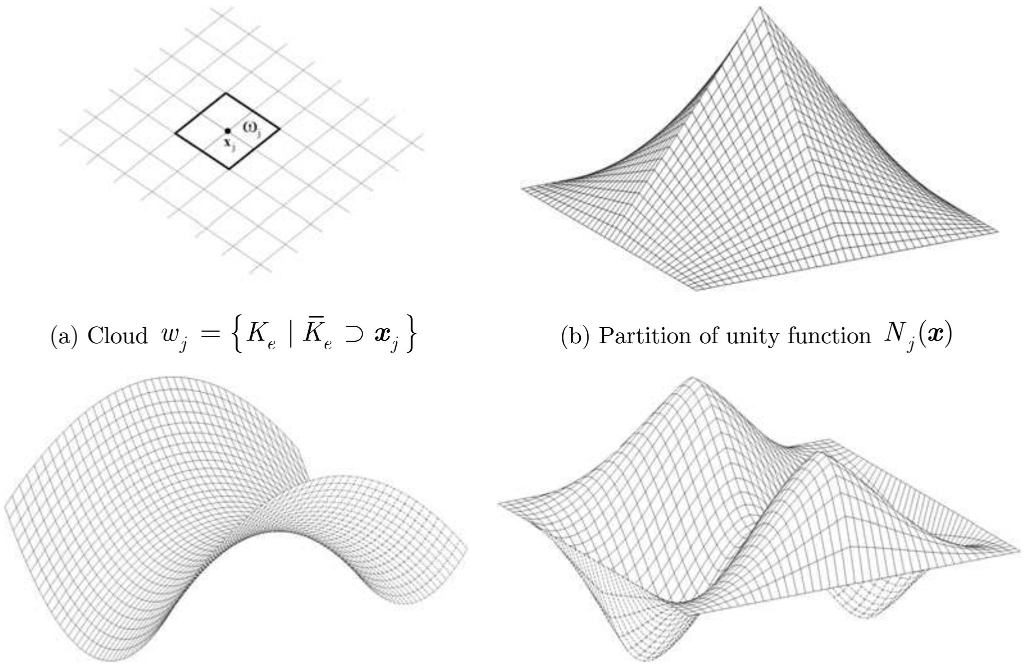
A generic patch of elements or cloud $w_j \in \bar{\Omega}$ is obtained by the union of finite elements sharing the vertex node $\boldsymbol{x}_j$, Fig. 1(a). The assemblage of the interpolation functions, built at each element $K_e \in w_j$ and associated with node $\boldsymbol{x}_j$, composes the function $N_j(\boldsymbol{x})$ defined over the support cloud w_j. As $\sum_{j=1}^{N} N_j(\boldsymbol{x}) = 1$ at every point $\mathbf{x}$ in the domain $\bar{\Omega}$, the set of functions $\{N_j(\boldsymbol{x})\}_{j=1}^{N}$ constitutes a partition of unity (PU). A set of q linearly independent functions is defined at each cloud w_j as:

$$I_j = \{L_{j1}(\boldsymbol{x}), L_{j2}(\boldsymbol{x}), ..., L_{jq}(\boldsymbol{x})\} = \{L_{ji}(\boldsymbol{x})\}_{i=1}^{q} \qquad L_{j1}(\boldsymbol{x}) = 1 \tag{1}$$

The generalized finite element shape functions are determined by the enrichment of the PU functions, which is obtained by the product of such functions by each one of the components of the set I_j at the generic cloud w_j:

$$\{\phi_{ji}\}_{i=1}^{q} = N_j(\boldsymbol{x}) \times \{L_{ji}(\boldsymbol{x})\}_{i=1}^{q} \tag{2}$$

The enrichment scheme is obtained by multiplying a PU function of C^0 type with compact support w_j, Fig. 1(a), by the function $L_{ji}(\boldsymbol{x})$, Fig. 1(c), named in Strouboulis et al. (2000b), as a local approximation (also called enrichment function). The resulting shape function, $\phi_{ji}(\boldsymbol{x})$, Fig. 1(d), inherits characteristics of both functions, i.e., the compact support and continuity of the PU and the approximate character of the local function.



(a) Cloud $w_j = \left\{ K_e \mid \bar{K}_e \supset \boldsymbol{x}_j \right\}$

(b) Partition of unity function $N_j(\boldsymbol{x})$

(c) Enrichment Function $L_{ji}(\boldsymbol{x})$

(d) $\phi_{ji}(\boldsymbol{x}) = N_j(\boldsymbol{x}) \times L_{ji}(\boldsymbol{x})$

Figure 1: Enrichment scheme for a cloud w_j, from Barros et al. (2007).

As a consequence, the generalized global approximation, denoted by $\tilde{u}(\boldsymbol{x})$, can be described as a linear combination of the shape functions associated with each node:

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}) = \sum\nolimits_{j=1}^{N} N_j(\boldsymbol{x}) \left[\boldsymbol{u}_j + \sum\nolimits_{i=2}^{q} L_{ij}(\boldsymbol{x}) \boldsymbol{b}_{ji} \right] \tag{3}$$

where $\boldsymbol{u}_j$ and $\boldsymbol{b}_{ji}$ are nodal parameters associated with standard – $N_j(\boldsymbol{x})$ – and GFEM – $N_j(\boldsymbol{x}) \times L_{ji}(\boldsymbol{x})$ – shape functions, respectively. Aiming to minimize round-off errors, Duarte et al. (2000) suggest that a transformation should be performed over the $L_{ji}(\boldsymbol{x})$ functions, if they are of polynomial type. In such case, the coordinate **x** is replaced as follows:

$$\boldsymbol{x} \rightarrow \frac{\boldsymbol{x} - \boldsymbol{x}_j}{h_j} \tag{4}$$

in which h_j is the diameter of the largest finite element sharing the node j, the cloud w_j.

3 GLOBAL-LOCAL GENERALIZED FINITE ELEMENT METHOD

This method, originally proposed by Duarte and Babuška (2005), combines the standard GFEM with the global-local strategy proposed by Noor (1986). GFEM$^{\mathrm{gl}}$ is suitable for problems with local phenomena, such as stress field next to the crack tip. The analysis is divided in three steps: initial global problem (step 1), local problem (step 2) and final global problem (step 3). These steps are described in the following sub-sections.

3.1 Initial Global Problem (Step 1)

Following Kim et al. (2010), consider a domain $\bar{\Omega}_G = \Omega_G \cup \partial\Omega_G$ of an elastic problem in R^n . The boundary is decomposed in $\partial\Omega_G = \partial\Omega_G^u \cup \partial\Omega_G^\sigma$ with $\partial\Omega_G^u \cap \partial\Omega_G^\sigma = \varnothing$, where indices u and σ refer to the Dirichlet and Neumann boundary conditions. $\boldsymbol{u}_G^0 \in \boldsymbol{X}_G^0(\Omega_G)$ represents the solution of the approximate space $\boldsymbol{X}_G^0(\Omega_G)$ (built by FEM or GFEM functions) for the initial global problem in its weak form, shown in:

$$\int_{\Omega_G} \boldsymbol{\sigma}(\boldsymbol{u}_G^0) : \boldsymbol{\varepsilon}(\boldsymbol{v}_G^0) d\boldsymbol{x} = \int_{\partial\Omega_G^\sigma} \bar{\boldsymbol{t}}.\boldsymbol{v}_G^0 d\boldsymbol{s} \tag{5}$$

where $\boldsymbol{\sigma}, \boldsymbol{\varepsilon}, \boldsymbol{v}_G^0 \in \boldsymbol{X}_G^0(\Omega_G)$, and $\bar{\boldsymbol{t}}$ are stress tensor, strain tensor, test functions, and prescribed traction vector, respectively.

3.2 Local Problem (Step 2)

Ω_L is a sub-domain from Ω_G. This sub-domain may contain cracks, holes or other special features, as suggested by Duarte and Kim (2008). The corresponding local solution $\boldsymbol{u}_L \in \boldsymbol{X}_L^E(\Omega_L)$ is obtained from:

$$\begin{gathered}\int_{\Omega_L} \boldsymbol{\sigma}(\boldsymbol{u}_L) : \boldsymbol{\varepsilon}(\boldsymbol{v}_L) d\boldsymbol{x} + \eta \int_{\partial\Omega_L \cap \partial\Omega_G^u} \boldsymbol{u}_L.\boldsymbol{v}_L d\boldsymbol{s} + \kappa \int_{\partial\Omega_L \setminus (\partial\Omega_L \cap \partial\Omega_G^u)} \boldsymbol{u}_L.\boldsymbol{v}_L d\boldsymbol{s} = \int_{\partial\Omega_L \cap \partial\Omega_G^\sigma} \bar{\boldsymbol{t}}.\boldsymbol{v}_L d\mathrm{s} + \\ \eta \int_{\partial\Omega_L \cap \partial\Omega_G^u} \bar{\boldsymbol{u}}.\boldsymbol{v}_L d\mathrm{s} + \int_{\partial\Omega_L \setminus (\partial\Omega_L \cap \partial\Omega_G)} (\boldsymbol{t}(\boldsymbol{u}_G^0) + \kappa \boldsymbol{u}_G^0).\boldsymbol{v}_L d\mathrm{s}\end{gathered} \tag{6}$$

where $\boldsymbol{v}_L \in \boldsymbol{X}_L^E(\Omega_L)$ represents the test functions, $\boldsymbol{X}_L^E(\Omega_L)$ is the space generated by FEM or GFEM functions, η is the penalty parameter and κ is the stiffness parameter both used to impose Dirichlet and Cauchy boundary conditions, respectively. Additionally, $\bar{\boldsymbol{t}}$ is the prescribed traction, $\bar{\boldsymbol{u}}$ is the prescribed displacement, $\boldsymbol{t}(\boldsymbol{u}_G^0)$ is the traction vector obtained from the first step. $\partial\Omega_L \setminus (\partial\Omega_L \cap \partial\Omega_G)$ is the part of the local problem boundary that doesn't coincide with the global problem boundary and on which the numerical solution that comes from the step 1 is imposed as boundary condition. The last integrals on the left and right side of Eq. (6) correspond to the Cauchy or spring boundary condition, imposing the displacements as well as the stress calculated in the global problem. The $\boldsymbol{X}_L^E(\Omega_L)$ space can be defined as:

$$\boldsymbol{X}_L^E(\Omega_L) = \left\{ \tilde{\boldsymbol{u}}(\boldsymbol{x}) = \sum_{j=1}^{N_L} N_j(\boldsymbol{x}) \left[\hat{\boldsymbol{u}}_j^E(\boldsymbol{x}) + H\tilde{\boldsymbol{u}}_j^E(\boldsymbol{x}) + \breve{\boldsymbol{u}}_j^E(\boldsymbol{x}) \right] \right\} \tag{7}$$

functions $\hat{\boldsymbol{u}}_j^E(\boldsymbol{x}), H\tilde{\boldsymbol{u}}_j^E(\boldsymbol{x})$ and $\breve{\boldsymbol{u}}_j^E(\boldsymbol{x})$ are continuous, discontinuous and singular components of the approximate solution and N_j is the partition of unity function used in global problem. The number of nodes of the local domain is given by N_L.

3.3 Final Global Problem (Step 3)

The global problem is enriched by solution $\boldsymbol{u}_L$ (from step 2). The new solution $\boldsymbol{u}_G^E \in \boldsymbol{X}_G^E(\Omega_G)$ is obtained from:

$$\int_{\Omega_G} \boldsymbol{\sigma}(\boldsymbol{u}_G^E) : \boldsymbol{\varepsilon}(\boldsymbol{v}_G^E) d\boldsymbol{x} = \int_{\partial\Omega_G^\sigma} \overline{\boldsymbol{t}}.\boldsymbol{v}_G^E d\boldsymbol{s} \tag{8}$$

where $\boldsymbol{v}_G^E$ represents the test functions and $\boldsymbol{X}_G^E(\Omega_G)$ is the initial space increased by $\boldsymbol{u}_k^{gl}(\boldsymbol{x})$ from local problem $\boldsymbol{u}_L$:

$$\boldsymbol{X}_G^E(\Omega_G) = \left\{ \tilde{\boldsymbol{u}}(\boldsymbol{x}) = \sum\nolimits_{j=1}^{N} N_j(\boldsymbol{x}) \hat{\boldsymbol{u}}_j^E(\boldsymbol{x}) + \sum_{k \in I_{gl}} N_k(\boldsymbol{x}) \boldsymbol{u}_k^{gl}(\boldsymbol{x}) \right\} \tag{9}$$

and $k \in I_{gl}$ represent the set of nodes enriched by the local solution $\boldsymbol{u}_L$.

Figure 2 illustrates a simplified global-local strategy. First the initial global problem is solved using a coarse mesh. After defining the local domain, the results of the initial global problem is transferred as a boundary condition to the local problem. Finally, the solutions of the local problem are transferred to the final global problem as a numeric enrichment function in order to enrich the predefined nodes of the global problem.

4 INTERACTIVE STRUCTURAL ANALYSIS ENVIRONMENT

The INSANE environment, Fonseca and Pitangueira (2007), Alves et al. (2013), is an open source software implemented in Java, an OOP language. The INSANE computational environment is composed by three great applications: pre-processor, processor and postprocessor. The persistence of data among these three segments of INSANE is performed by data files written in extensible markup language format. Here, a summary of several modules of the numerical core application (the processor) are presented aiming to introduce the INSANE system, corresponding to the standard FEM/GFEM approach, and also to show the generalization performed here to enclose the GFEM$^{\mathrm{gl}}$ formulation. More information can be found in Alves et al. (2013).

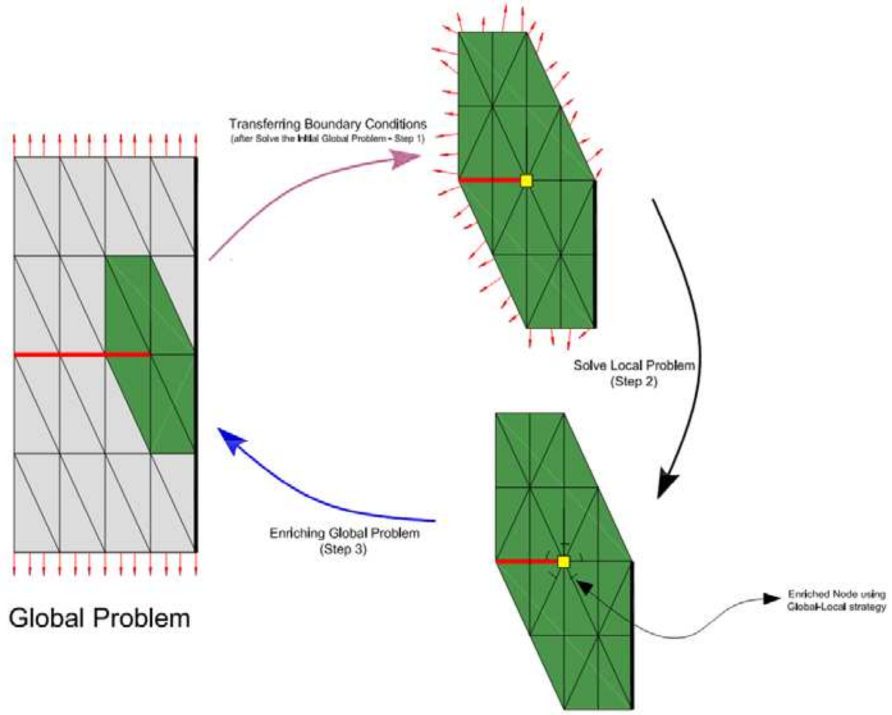


Figure 2: Three steps of the Global-Local strategy.

The INSANE numerical core is composed by the interfaces *Assembler*, *Model* and *Persistence* and the abstract class *Solution*. Figure 3 shows the unified modeling language (UML) diagram from numerical core of INSANE.

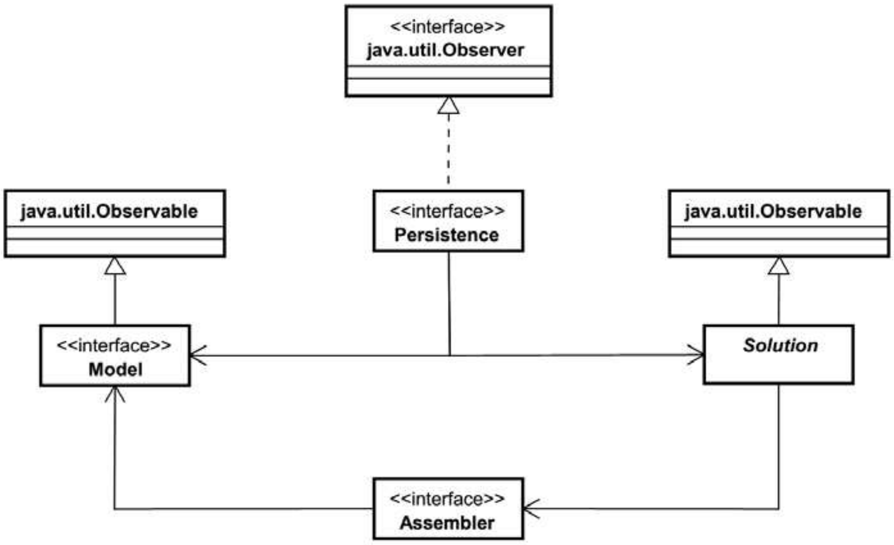


Figure 3: Organization of numerical core of INSANE.

The observation strategy is determined by the *observer-observable* design pattern, which is a change propagation mechanism. When an object of type *observer* (which implements the *interface java.util.observer*) is instantiated, it is added to a list of observers of other objects of type observables (which extends the class *java.util.observable*). Any modification in the state of an observed object notifies the corresponding observer object that updated itself.

Detailed of GFEM implementation can be found in Alves et al. (2013), section 4. Here, the focus is on the main modifications in order to expand the computational system to comprise GFEM$^{\mathrm{gl}}$. Following sub-sections explain various INSANE classes and approaches in order to perform GFEM analysis as well as GFEM$^{\mathrm{gl}}$ analysis.

4.1 Persistence Interface

Persistence treats the input data and persists the output data. For GFEM$^{\mathrm{gl}}$, this class was extended to deal with more than one model. In that case, the data is separated in a global model and several local models, corresponding to the global and the several local problems respectively.

4.2 Assembler Interface

Assembler interface is responsible for assembling the linear equation system provided by the discretization of the initial value problem. This class is implemented following the generic representation:

$$\boldsymbol{A}\ddot{\boldsymbol{X}} + \boldsymbol{B}\dot{\boldsymbol{X}} + \boldsymbol{C}\boldsymbol{X} = \boldsymbol{D} \tag{10}$$

where **X** is the solution vector; the single dot represent its first time derivative and the double dots its second time derivative; **A**, **B** and **C** are matrices with the properties of the problem and **D** is a vector that represents the system excitation.

The *Assembler* interface has the necessary methods for assembling the matrix system. It is implemented by an object of the class *GFemAssembler* that was modified in order to transfer boundary condition from the global model to the several local models. A Cauchy boundary condition was implemented following Kim et al. (2010) (here represented by Eq. (6)).

In static analysis, Eq. (10) is simplified by eliminating the two first terms. The resulting matrix system is:

$$\begin{bmatrix} \boldsymbol{C}_{uu} & \boldsymbol{C}_{up} \\ \boldsymbol{C}_{pu} & \boldsymbol{C}_{pp} \end{bmatrix} \begin{Bmatrix} \boldsymbol{X}_{u} \\ \boldsymbol{X}_{p} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{D}_{p} \\ \boldsymbol{D}_{u} \end{Bmatrix} \tag{11}$$

where Eq. (11) is used to represent each step of the global-local solution given by Eqs. (5), (6), and (8). In these equations, the matrix **C** is the model stiffness matrix, **X** is the vector of nodal displacements, **D** is the vector of forces.

4.3 Solution

Solution abstract class starts the solution process and has the necessary resources for solving the matrix system. GFEM^{gl} process is implemented by *GlobalLocal* class, as shown in Fig. 4. Since GFEM^{gl} is composed by more than one model (one global model and several local models), the problem has more than one assembler, each one responsible to build the corresponding equation in the form of (11). This is the main difference from standard GFEM solution class. Figure 4 shows in details the UML diagram for this class.

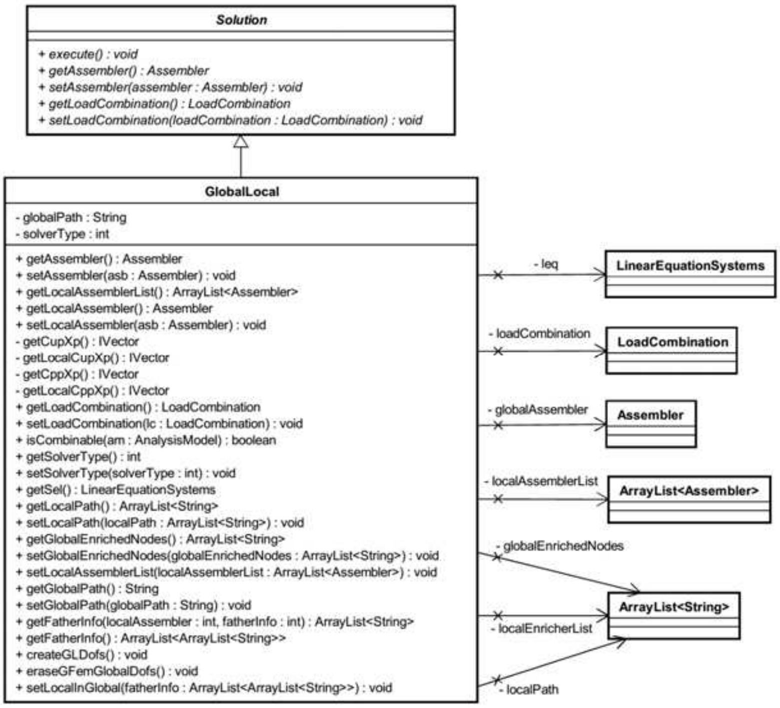

Solution
+ execute() : void
+ getAssembler() : Assembler
+ setAssembler(assembler : Assembler) : void
+ getLoadCombination() : LoadCombination
+ setLoadCombination(loadCombination : LoadCombination) : void
GlobalLocal
- globalPath : String
- solverType : int
+ getAssembler() : Assembler
+ setAssembler(asb : Assembler) : void
+ getLocalAssemblerList() : ArrayList<Assembler>
+ getLocalAssembler() : Assembler
+ setLocalAssembler(asb : Assembler) : void
- getCupXp() : IVector
- getLocalCupXp() : IVector
- getCppXp() : IVector
- getLocalCppXp() : IVector
+ getLoadCombination() : LoadCombination
+ setLoadCombination(lc : LoadCombination) : void
+ isCombinable(am : AnalysisModel) : boolean
+ getSolverType() : int
+ setSolverType(solverType : int) : void
+ getSel() : LinearEquationSystems
+ getLocalPath() : ArrayList<String>
+ setLocalPath(localPath : ArrayList<String>) : void
+ getGlobalEnrichedNodes() : ArrayList<String>
+ setGlobalEnrichedNodes(globalEnrichedNodes : ArrayList<String>) : void
+ setLocalAssemblerList(localAssemblerList : ArrayList<Assembler>) : void
+ getGlobalPath() : String
+ setGlobalPath(globalPath : String) : void
+ getFatherInfo(localAssembler : int, fatherInfo : int) : ArrayList<String>
+ getFatherInfo() : ArrayList<ArrayList<String>>
+ createGLDofs() : void
+ eraseGFemGlobalDofs() : void
+ setLocalInGlobal(fatherInfo : ArrayList<ArrayList<String>>) : void
- leq
LinearEquationSystems
- loadCombination
LoadCombination
- globalAssembler
Assembler
- localAssemblerList
ArrayList<Assembler>
- globalEnrichedNodes
ArrayList<String>
- localEnricherList
- localPath

Figure 4: UML diagram for Solution class.

The main attributes of this class are:

- *globalPath*: contains the path of the file responsible for storing the global model description.
- *solverType:* contains information about the type of solver of the linear equation system.
- *leq*: *LinearEquationSystem* object which is responsible to solve the linear equation system.
- *loadCombination*: load combination of the problem.
- *globalAssembler*: handles information about the global model.
- *localAssemblerList*: list of Assembler objects that handles information about the several local models.
- *globalEnrichedNodes*: handles the pointer to the global nodes that will be enriched by global-local enrichment.

- *localEnrichedList*: list that relates each global node, that will be enriched by the global-local enrichment strategy, with the corresponding local model. This is a very important information because it is in the local model that the function $\boldsymbol{u}_k^{gl}(\boldsymbol{x})$ will be built and used to enrich the PU of the global node.
- *localPath*: contains the list of paths of each file responsible for storing each local model description.

4.4 Model Interface

The *Model* interface contains the data of the discrete model and provides to *Assembler* information to assemble the final matrix system. Both *Model* and *Solution* communicate with the *Persistence* interface, which treats the input data and persists the output data to the other applications, whenever it observes a modification of the discrete model state. This interface consisted of the following classes:

4.4.1 Node

This class is designed to manage the geometric representation of a node entity as well as the information from the discrete model. In addition to information about the node coordinates, this class holds lists of identifiers such as type of degrees of freedom, state variables and type of nodal constraint. This list of identifiers are based on *HashMap* strategy provided by JAVA API Collection Horstmann and Cornell (2008). This type of Java storing strategy allows an indirect communication between variables associated to the Node, named keys, and the variables stored in the *Model*. In spite of degrading performance of the framework, such strategy creates a generic and friendly implementation because it simplifies how new information is added to the class. Furthermore, methods from JAVA API Collection allows using the keys like conventional attributes which preserves the readability of the code. *“LOCAL_ ENRICHER”* is added in order to indicate which local domain will provide the global-local enrichment function, as shown in Fig. 5. This new attribute is defined in the *Node* class as an instance of GFEM using the enrichment strategy.

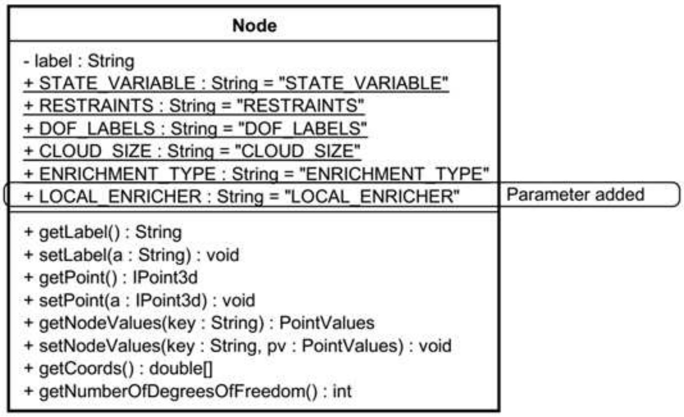

Node
- label : String
+ STATE_VARIABLE : String = "STATE_VARIABLE"
+ RESTRAINTS : String = "RESTRAINTS"
+ DOF_LABELS : String = "DOF_LABELS"
+ CLOUD_SIZE : String = "CLOUD_SIZE"
+ ENRICHMENT_TYPE : String = "ENRICHMENT_TYPE"
+ LOCAL_ENRICHER : String = "LOCAL_ENRICHER"
Parameter added
+ getLabel() : String
+ setLabel(a : String) : void
+ getPoint() : IPoint3d
+ setPoint(a : IPoint3d) : void
+ getNodeValues(key : String) : PointValues
+ setNodeValues(key : String, pv : PointValues) : void
+ getCoords() : double[]
+ getNumberOfDegreesOfFreedom() : int

Figure 5: UML diagram for Node.

4.4.2 Element

In Alves et al. (2013), this class was not modified for standard GFEM. On the other hand, the special characteristics of GFEM^{gl}, such as the relationship between the global and the local elements, requires a new extension of the Element class. Note that this new *GFemElement* class is not only derived from the superclass Element but also contains on object of Element type, Fig. 6.

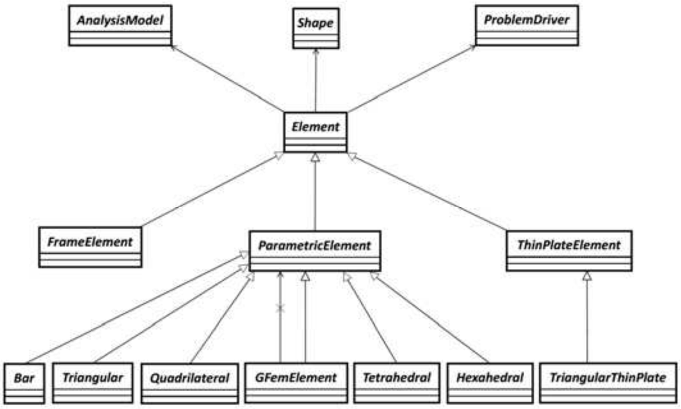


Figure 6: UML diagram for *Element* class.

Indeed this strategy takes advantage of parametric element library, with bar, triangular, quadrilateral, tetrahedral and hexahedral elements, without having to use simple inheritance from each one of those types of elements. Figure 7 shows the new parameters added in *GFemElement* class in order to add the ability to solve problems using the global-local scheme. Similar to the Node class, the *HashMap* strategy is used to include new parameters. The parameters of this class are:

- *LOCAL_NAME*: if the element belongs to a local problem, this parameter indicates in which local model it is inserted.
- *GLOBAL_ELEMENT*: if the element belongs to a local problem, this parameter indicates which element from the global model contains the current element. This relationship allows to obtain from the global element the necessary data to apply the boundary conditions on the edge, if it is the case, of the current element.
- *LOCAL_ELEMENTS*: if the element belongs to a global problem, this parameter indicates the elements of the local problem that discretize the domain of this global element. In this implementation the local mesh is nested in the global mesh. As a consequence each global element exactly contains a set of local elements.
- *BOUNDARY_INFORMATION*: informs which kind of boundary conditions is transferred from step 1 to step 2.
- STEP_GL: informs in which step of the problem the solution is being processed.
- *LOCAL_TWINS*: if the element belongs to a local problem, this parameter indicates which element, if there is, from another local problem coincides with it.

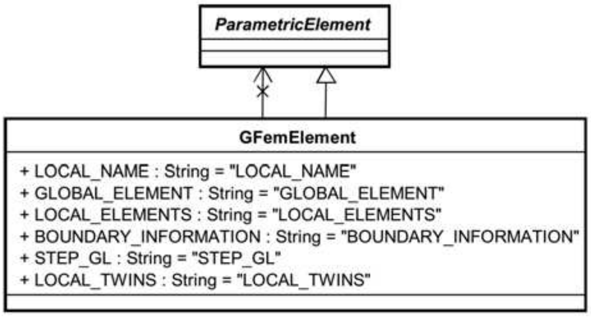

ParametricElement
GFemElement
+ LOCAL_NAME : String = "LOCAL_NAME"
+ GLOBAL_ELEMENT : String = "GLOBAL_ELEMENT"
+ LOCAL_ELEMENTS : String = "LOCAL_ELEMENTS"
+ BOUNDARY_INFORMATION : String = "BOUNDARY_INFORMATION"
+ STEP_GL : String = "STEP_GL"
+ LOCAL_TWINS : String = "LOCAL_TWINS"

Figure 7: UML diagram for *GFemElement* class.

4.4.3 EnrichmentType

GlobalLocalEnrichment extends *EnrichmentType* and provides specific methods to build the enriched functions from the solution of the local problem and applied in the third step of the global-local problem. Figure 8 shows the class diagram of this class. *EnrichmentType* is an abstract class and its methods are abstract. Thus, *GlobalLocalEnrichment* class contains the same method as the *EnrichmentType* class.

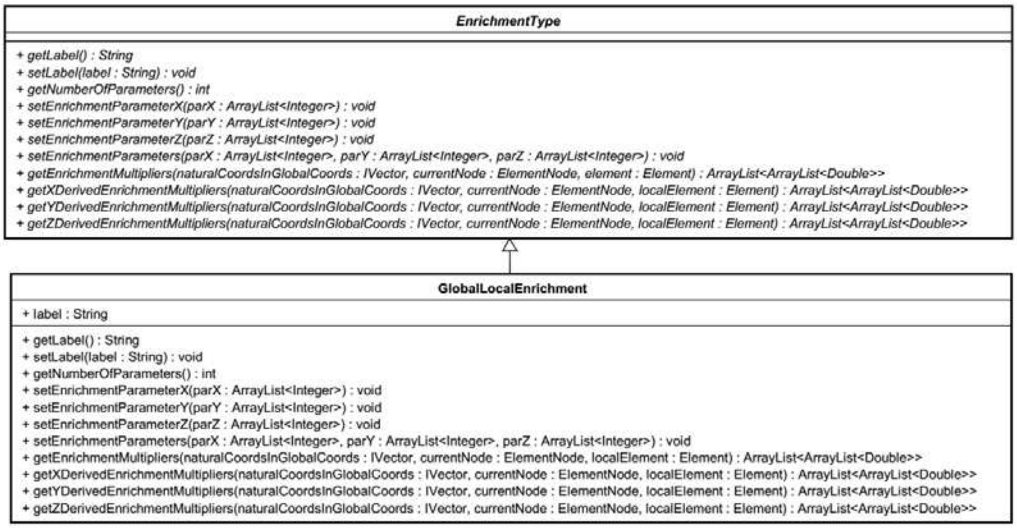

EnrichmentType
+ getLabel() : String
+ setLabel(label : String) : void
+ getNumberOfParameters() : int
+ setEnrichmentParameterX(parX : ArrayList<Integer>) : void
+ setEnrichmentParameterY(parY : ArrayList<Integer>) : void
+ setEnrichmentParameterZ(parZ : ArrayList<Integer>) : void
+ setEnrichmentParameters(parX : ArrayList<Integer>, parY : ArrayList<Integer>, parZ : ArrayList<Integer>) : void
+ getEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, element : Element) : ArrayList<ArrayList<Double>>
+ getXDerivedEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, localElement : Element) : ArrayList<ArrayList<Double>>
+ getYDerivedEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, localElement : Element) : ArrayList<ArrayList<Double>>
+ getZDerivedEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, localElement : Element) : ArrayList<ArrayList<Double>>
GlobalLocalEnrichment
+ label : String
+ getLabel() : String
+ setLabel(label : String) : void
+ getNumberOfParameters() : int
+ setEnrichmentParameterX(parX : ArrayList<Integer>) : void
+ setEnrichmentParameterY(parY : ArrayList<Integer>) : void
+ setEnrichmentParameterZ(parZ : ArrayList<Integer>) : void
+ setEnrichmentParameters(parX : ArrayList<Integer>, parY : ArrayList<Integer>, parZ : ArrayList<Integer>) : void
+ getEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, localElement : Element) : ArrayList<ArrayList<Double>>
+ getXDerivedEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, localElement : Element) : ArrayList<ArrayList<Double>>
+ getYDerivedEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, localElement : Element) : ArrayList<ArrayList<Double>>
+ getZDerivedEnrichmentMultipliers(naturalCoordsInGlobalCoords : IVector, currentNode : ElementNode, localElement : Element) : ArrayList<ArrayList<Double>>

Figure 8: UML diagram for *GlobalLocalEnrichment*.

4.4.4 Problem Driver

Informs to *Assembler* all the necessary data for assembling the final system of the model, Eq. (11). For GFEM^{gl} case, an additional loop is performed using descendants of each global element, since they play the role of cells of integration. Actually, information of each descendants will be used to enriching corresponding global element based on relationship between step 2 and step 3 of global-local strategy. Figure 9 describes stiffness matrix calculation using sequence diagram.

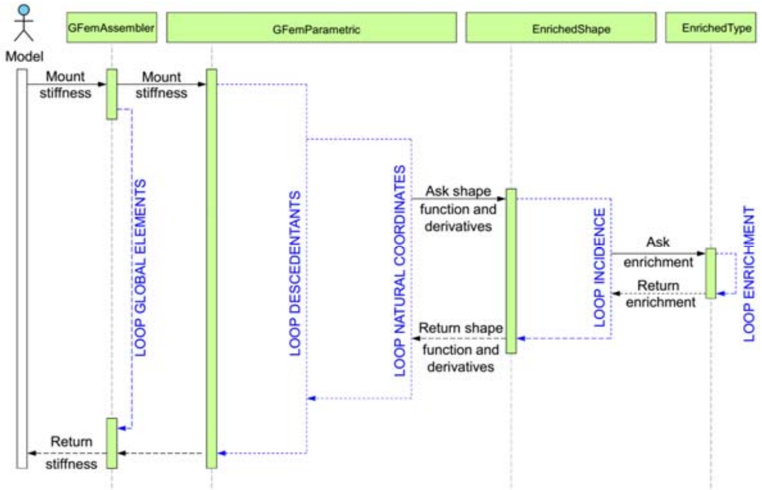


Figure 9: Sequence diagram for stiffness calculation.

4.5 Generic Boundary Condition

Figure 10 shows the relationship between the types of objects found in the package *Value*, which is used to define a generic boundary condition of the problem (e.g., conditions of Dirichlet, Neumann or Cauchy required by Eq. (6)). The definition of these boundary conditions in different geometric entities is performed through an *ElementValue* object, which is derived for the specific types *ElementVolumeValue* (boundary conditions applied in a volume), *ElementAreaValue* (boundary conditions applied to a area) and *ElementLineValue* (boundary conditions imposed on a line). *ElementValue* consists of the following attributes:

- An array of objects of type *PointValue*, responsible for storing the coordinates of a point and information that defines the boundary condition at this point.
- An object of type Shape, responsible for the interpolation of *PointValue*.

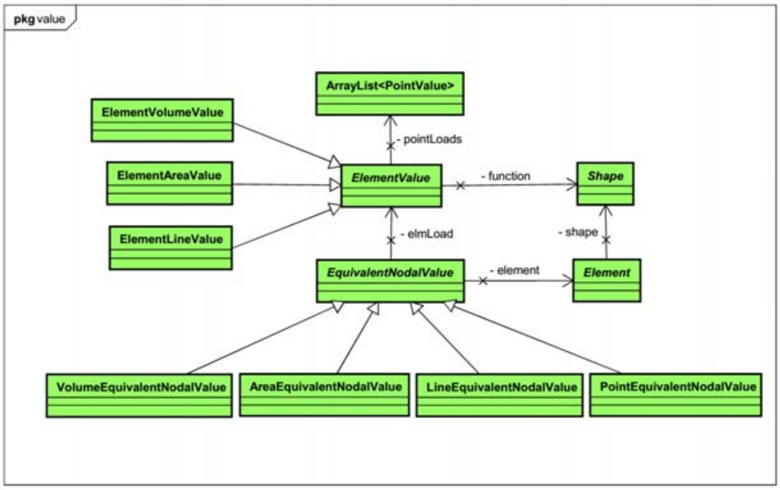


Figure 10: UML diagram of the Value package.

The class *Shape* is responsible for providing the approximation function that interpolates the different *PointValue* applied in a region of the element that can be a line, area or a volume, depending on the element type. The responsible for combining this information and providing the equivalent nodal value is *EquivalentNodalValue* class, whose attributes are objects of *Element* and *ElementValue* classes.

4.6 Information Transferring from the Global to the Local Problem

One of the actions performed during the process of generalization of the imposition of boundary conditions on INSANE environment is to add the ability to obtain boundary conditions of an element of the local problem from an element of the global problem. This is required in the global-local analysis process in which, it is necessary to impose boundary conditions for the local problem provided by the global mesh solution. There are basically three types of boundary conditions that can be used:

- Dirichlet boundary conditions
- Neumann boundary conditions
- Cauchy boundary conditions

The class responsible for the transfer of boundary conditions between domains is the *EquivalentNodalValue* class that is part of the *Value* package. The *EquivalentNodalValue* class has methods that fall into those capable of modifying the matrix **C** (stiffness) and those able to modify the vector **D** (force) of the system in Eq. (10), as can be seen in Fig. 11.

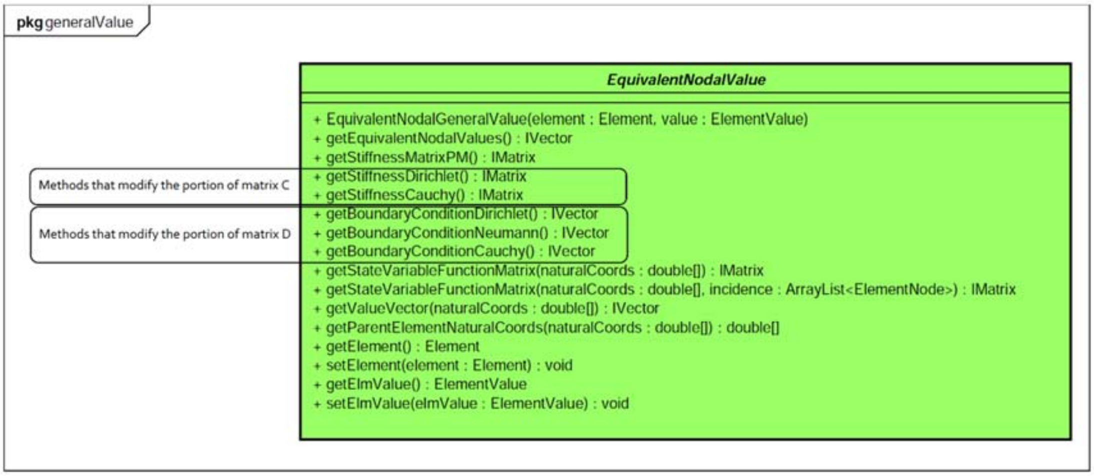

pkg generalValue
EquivalentNodalValue
+ EquivalentNodalGeneralValue(element : Element, value : ElementValue)
+ getEquivalentNodalValues() : IVector
+ getStiffnessMatrixPM() : IMatrix
Methods that modify the portion of matrix C
+ getStiffnessDirichlet() : IMatrix
+ getStiffnessCauchy() : IMatrix
Methods that modify the portion of matrix D
+ getBoundaryConditionDirichlet() : IVector
+ getBoundaryConditionNeumann() : IVector
+ getBoundaryConditionCauchy() : IVector
+ getStateVariableFunctionMatrix(naturalCoords : double[]) : IMatrix
+ getStateVariableFunctionMatrix(naturalCoords : double[], incidence : ArrayList<ElementNode>) : IMatrix
+ getValueVector(naturalCoords : double[]) : IVector
+ getParentElementNaturalCoords(naturalCoords : double[]) : double[]
+ getElement() : Element
+ setElement(element : Element) : void
+ getElmValue() : ElementValue
+ setElmValue(elmValue : ElementValue) : void

Figure 11: EquivalentNodalValue class.

5 NUMERICAL EXAMPLES

This section presents two linear-elastic problems in R^2. Section 5.1 presents a linear bending problem, and section 5.2 presents a fracture mechanic problem. The geometry and boundary conditions are very simple, since the goal is not demonstrate the capabilities of the methods, but the general approach proposed to enclose the GFEM^{gl} formulation in INSANE environment.

Among the three aforementioned boundary conditions in section 4.6, according to Kim et al. (2010), the Dirichlet boundary conditions (a limiting case of Cauchy boundary condition) lead to worse results than Cauchy boundary conditions. Thus, for all two examples, the Dirichlet boundary conditions are applied on the local problem boundaries in order to demonstrate the robustness of the methodology in the worst case scenario.

Numerical integration for the first and second steps of the global-local analysis is done based on standard Gaussian quadrature procedure. In the third step, the numerical integration for those global elements that contain local elements is done over the Gauss points of local elements, as proposed by Kim et al. (2010). Consider that a global element contains n^{Le} local elements and number of Gauss points for each local elements be equal to GP. Thus, the number of integration points for this global element is obtained by: $\sum_{i=1}^{n^{L_e}} GP_i$. In other words, the global numerical integration is done over Gauss points of each local element that represent a part of the global element.

5.1 Linear Bending-Moment Problem

Figure 12 illustrates a beam subjected to a set of loads that produce a linear-bending moment. In Lee and Bathe (1993), this problem is used to evaluate numerically the effects of element distortions in FEM. Here, this problem exemplifies and validates the GFEM^{gl} implementation within INSANE environment.

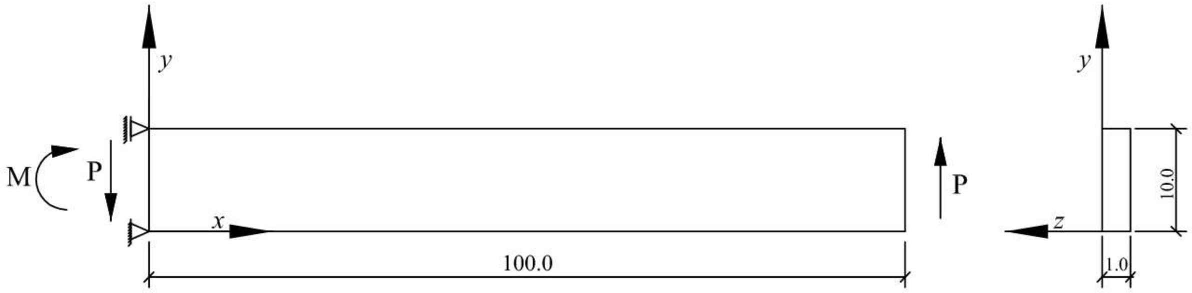


Figure 12: Linear bending-moment problem - geometry and loading.

The data for solving this two-dimensional plane stress problem are (using consistent units):

- Modulus of elasticity E = 1.0×10^7;
- Poisson ratio $\nu = 0.3$;
- Load P = 20 distributed as the load function $f_y = 0.12y^2 + 1.2y$.
- Moment M = 2000 distributed as the load function $f_x = 24y–120$.

The analytical solution is given for the displacements u and v, in direction x and y respectively and described in Cartesian coordinates, as:

$$u = \frac{1}{E}(0.12x^2 - 0.092\,\mathrm{y}^3 - 0.6x^2 - 24xy + 1.38y^2 + 120x - 4.6y) \tag{12}$$

$$v = \frac{1}{E}(-0.04x^3 - 0.036x\,\mathrm{y}^2 + 12x^2 + 0.36xy + 3.6y^2 + 4.6x - 36y) \tag{13}$$

Using the GFEM$^{\mathrm{gl}}$, the solution of this problem can be divided in three steps:

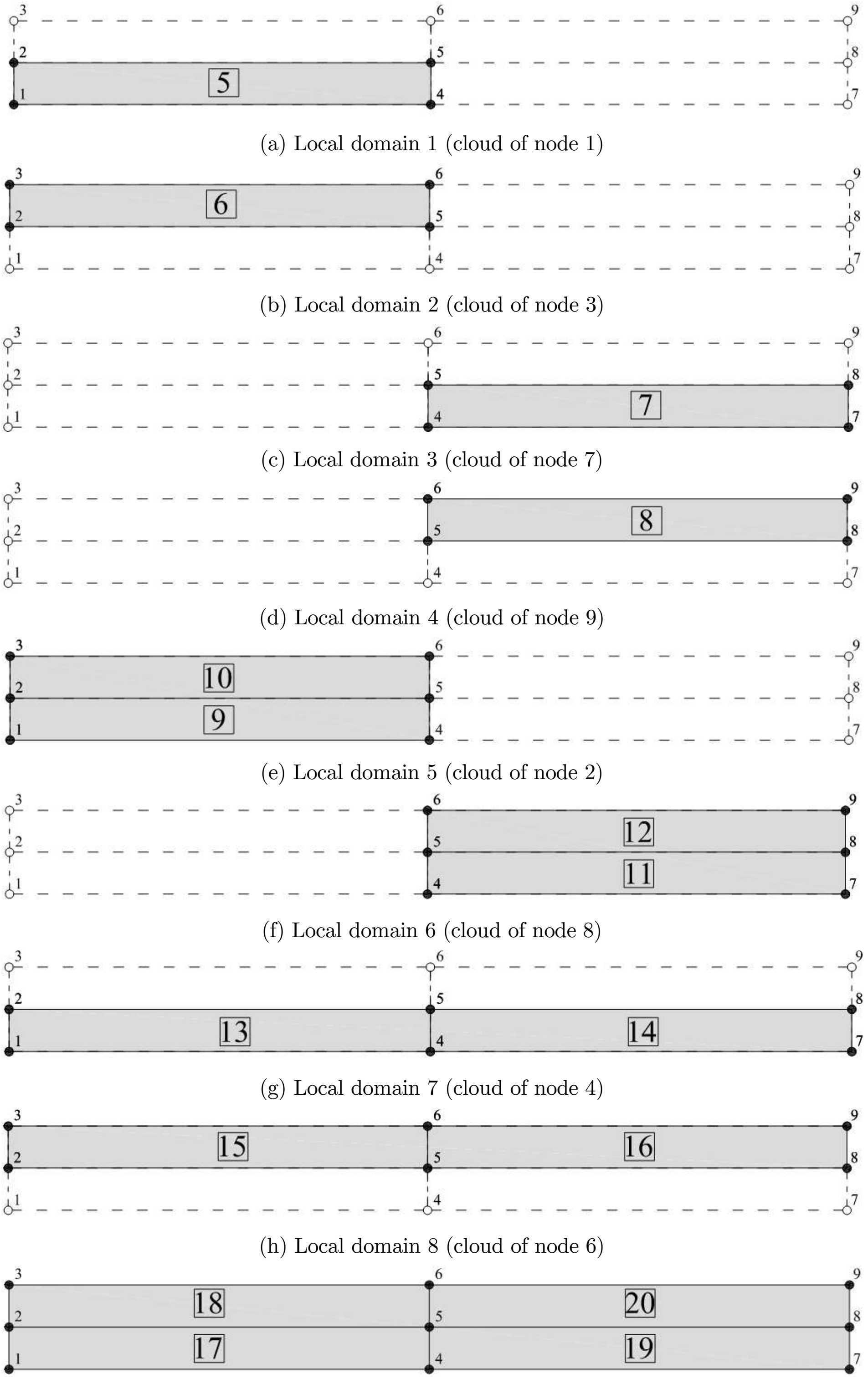
- *Step 1:* All the nodes of the problem are enriched by P2 functions (quadratic enrichment). This strategy guarantees a cubic order of approximation through the beam, reproducing the analytical solution of the problem (see Eqs. (12) and (13)). The P2 enrichment function for x direction is as follows:

$$\phi_j^T(\boldsymbol{x}) = \begin{bmatrix} N_j(\boldsymbol{x}) & 0 & \left(\frac{x - x_j}{h_j}\right)^2 N_j(\boldsymbol{x}) & 0 & \left(\frac{y - y_j}{h_j}\right)^2 N_j(\boldsymbol{x}) & 0 \\ 0 & N_j(\boldsymbol{x}) & 0 & \left(\frac{x - x_j}{h_j}\right)^2 N_j(\boldsymbol{x}) & 0 & \left(\frac{y - y_j}{h_j}\right)^2 N_j(\boldsymbol{x}) \end{bmatrix} \tag{14}$$

- *Step 2:* Global problem is split in 9 local problems, as presented in Fig. 13. Each local problem is enriched using P2 functions in order to exactly represent the analytical solution of the problem (this is only possible if exact boundary conditions are provided from step 1). At the end of step 2, there is a numerical solution that can exactly reproduce this problem.
- *Step 3:* Returning to the global problem, the enrichment P2 is replaced by global-local enrichment obtained from step 2. For this example, the numerical solution from local problems can exactly reproduce the analytical solution for this problem.

For this analysis, a Gauss-Legendre quadrature rule with 4 × 4 points is employed for both global and local problems.

Figure 14 shows stress distribution of stress component σ_{xx} for each local domain of the beam. This figure depicts a generic application of the GFEM$^{\mathrm{gl}}$ implementation, in which any region, or the whole mesh can be defined as local domains and used to solve the problem. This is an important issue in the case of not having a predefined local phenomenon and can take advantage of the independent characteristic of each local problem. As an efficient way and with the help of parallel processing, each local problem can be solved separately using multi-core machines. The P2 enrichment is used for the global problem aiming to avoid any error in the boundary condition of the local problem. The idea is to having an "*exact*" local solution to enrich the global mesh. Then, the implementation is validate if the global-local solution exactly reproduce the analytic solution.



(a) Local domain 1 (cloud of node 1)

(b) Local domain 2 (cloud of node 3)

(c) Local domain 3 (cloud of node 7)

(d) Local domain 4 (cloud of node 9)

(e) Local domain 5 (cloud of node 2)

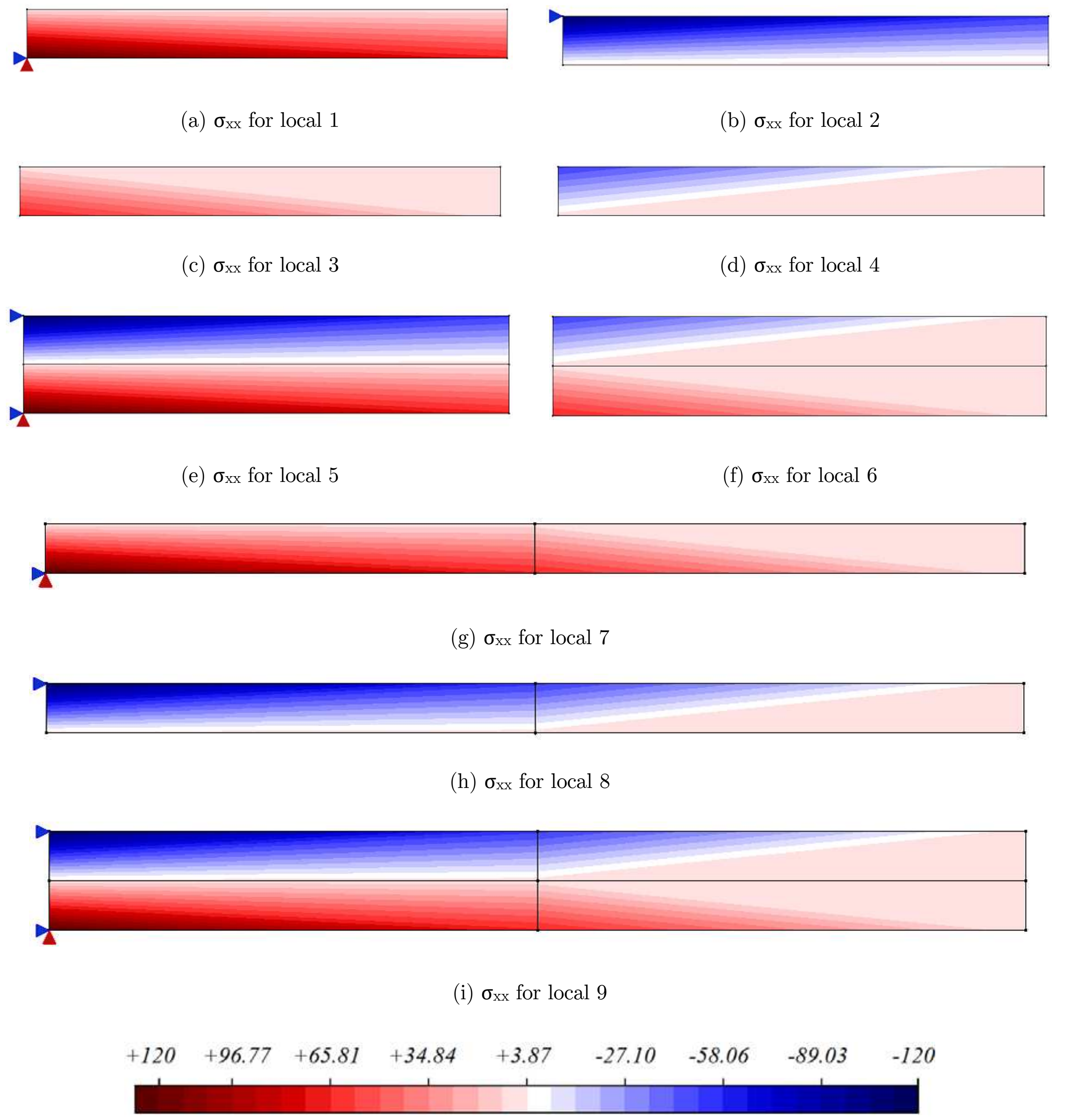
(f) Local domain 6 (cloud of node 8)

(g) Local domain 7 (cloud of node 4)

(h) Local domain 8 (cloud of node 6)

(i) Local domain 9 (cloud of node 5)

Figure 13: Description of the 9 local domains of the problem.


(a) σ_{xx} for local 1

(b) σ_{xx} for local 2

(c) σ_{xx} for local 3

(d) σ_{xx} for local 4

(e) σ_{xx} for local 5

(f) σ_{xx} for local 6

(g) σ_{xx} for local 7

(h) σ_{xx} for local 8

(i) σ_{xx} for local 9

Figure 14: Results of stress component σ_{xx} for each local domain.

5.2 Fracture Mechanic Problem

This example considers a cracked plate submitted to an axial stress, as shown in Fig. 15. The cracked zone produces singular stress field near the crack tip. The objective of this problem is to illustrate the use of this version of GFEM^{gl} for fracture mechanic problems. This problem analyzed under plane stress state, has the following parameters (in consistent units):

- Modulus of elasticity E = 1.0;
- Poisson ratio $\nu = 0.3$;
- Traction $\sigma = 1.0$.

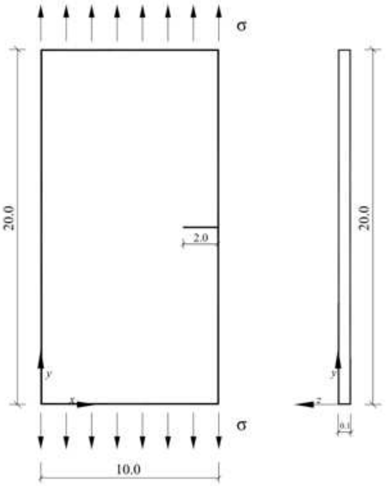


Figure 15: Geometry and loading of a cracked plate submitted to an axial stress.

The reference solution of this problem is obtained using a mesh of 12087 p–quadrilateral elements in ANSYS®. For GFEM^{gl} analysis, however, a smaller number of finite elements as well as of degrees of freedom (DOFs) is used. The reason for using a smaller number of DOFs is explained by the use of global-local enrichment function, which is suitable for high stress concentration. This strategy is used to describe the singular stress near the crack tip.

A geometric mesh is used to describe the local domain, Fig. 16 and it is shown in Fig. 17. This mesh is graded so that the elements are decreased in geometric progression toward the crack tip, in four levels (L1, L2, L3 and L4) with a common reduction factor of 10%. Results are shown in Fig. 18. Also, Table 1 presents a comparison between the strain energy values for the reference and approximate solutions.

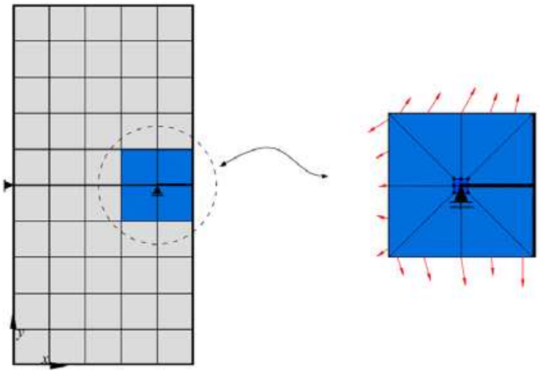


Figure 16: Global and local domain of the problem.

Analysis	DOFs	Strain energy
Reference result	24648	10.98326
Global-local result	134	10.80963

Table 1: DOFs and strain energy for geometric enrichment.

It can be observed from the results shown in Fig. 18 and Table 1 the capability of the global-local enrichment strategy to represent the singular stress field close to the crack tip.

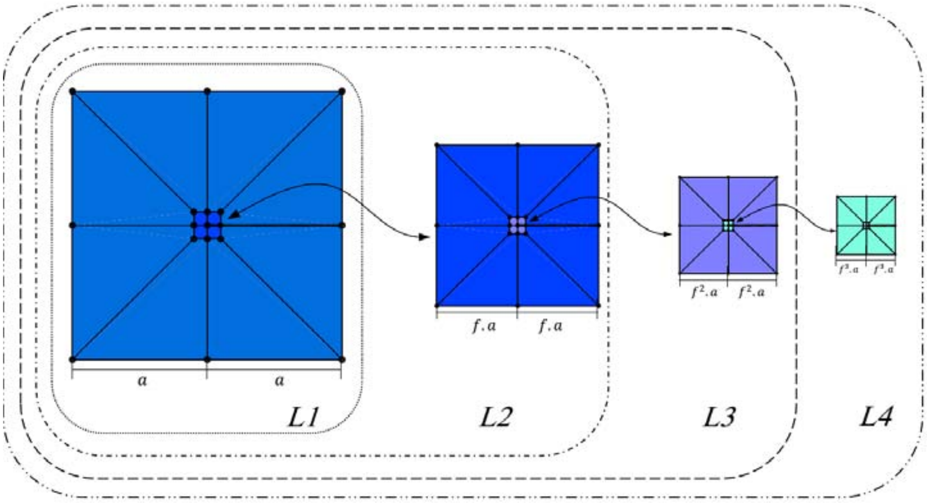


Figure 17: Geometric mesh with reduction rate f = 10%.

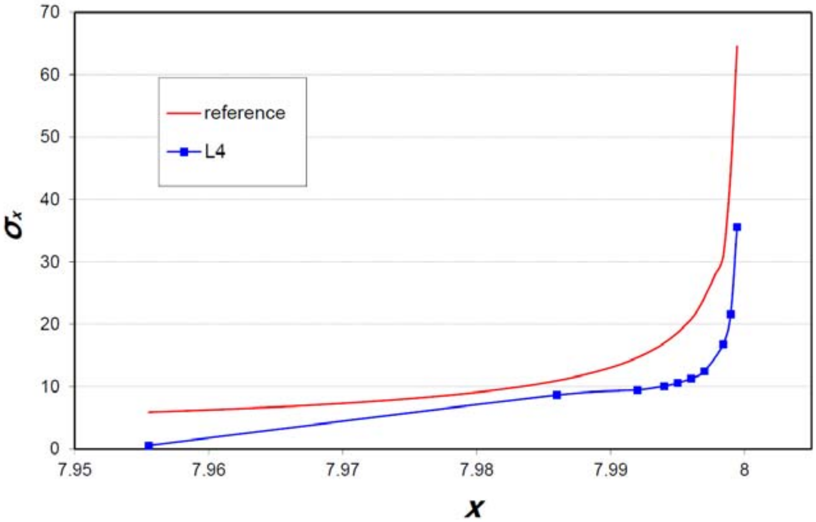


Figure 18: Results for stress component σxx (step 3) in different positions given by the coordinate x and the same coordinate y of the crack, for L4 in the global-local analysis.

6 CONCLUSION

An overview of the GFEM^{gl} is given with emphasis on implementation aspects under OOP approach. The focus of this work is to present a GFEM^{gl} extension of a FEM programming environment called INSANE. The proposed approach allows to combine any types of partition of unity methods, analysis model and enrichment strategy. Also, an expansion of the numerical core INSANE, in order to adapt it to impose boundary conditions using the penalty method is included. The OOP approach is fully exploited, reinforcing the premise segmentation and encapsulation of the INSANE environment. The validation of this extension and some additional conclusions about the GFEM global-local strategy are presented by numerical examples for Solid Mechanics.

Acknowledgments

The authors would like to gratefully acknowledge the important support of the Brazilian research agencies CNPq (National Council for Scientific and Technological Developments - Grants 309005/2013-2, 486959/2013-9 and 308785/2014-2), FAPEMIG (Foundation for Research Support of the State of Minas Gerais - Grant PPM-00669-15) and CAPES (Coordination for the Improvement of Higher Education Personnel).

References

Alves, P.D., Barros, F.B., Pitangueira, R.L.S. (2013). An object oriented approach to the generalized finite element method, Advances in Engineering Software 59: 1–18.

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

Babuška, I., Melenk, J.M. (1997). The partition of unity method, International Journal of Numerical Methods in Engineering 40: 727–58.

Barros, F.B., de Barcellos, C.S., Duarte, C.A. (2007). p-Adaptive C^k generalized finite element method for arbitrary polygonal clouds, Computational Mechanics 41: 175–87.

Barros, F.B., de Barcellos, C.S., Duarte, C.A., Torres D.F. (2013). Sub-domain based error techniques for generalized finite element approximations of problems with singular stress fields, Computational Mechanics 52: 1395–415.

Barros, F.B., Proença, S.P.B., de Barcellos, C.S. (2004). On error estimator and p-adaptivity in the generalized finite element method, International Journal of Numerical Methods in Engineering 60(14): 2373–98.

Belytschko, T., Black, T. (1999). Elastic crack growth in finite elements with minimal remeshing, International Journal of Numerical Methods in Engineering 45: 601–20.

Bordas, S., Nguyen, P.V., Dunant, C., Guidoum, A., Nguyen-Dang, H. (2007). An extended finite element library, International Journal of Numerical Methods in Engineering 71: 703–32.

Chamrová, R., Patzák, B. (2010). Object-oriented programming and the extended finite-element method, Proc. Inst. Civ. Eng. - Eng. Computational Mechanics 163: 271–78.

Duarte, C. A. (1996). The hp-cloud method, PhD dissertation, University of Texas at Austin, USA.

Duarte, C.A., Babuška, I. (2005). A global-local approach for the construction of enrichment functions for the gfem and its application to propagating three-dimensional cracks, ECCOMAS Thematic Conference on Meshless Methods.

Duarte, C.A., Babuška, I., Oden, J.T. (1995). Hp clouds - A meshless method to solve boundary-value problem, Tech. rep., TICAM, The University of Texas at Austin, technical Report.

Duarte, C.A., Babuška, I., Oden, J.T. (2000). Generalized finite element methods for three-dimensional structural mechanics problems, Computers and Structures 77: 215–32.

Duarte, C.A., Kim, D.J. (2008). Analysis and applications of a generalized finite element method with global-local enrichment functions, Computer Methods in Applied Mechanics and Engineering 197: 487–504.

Duarte, C.A., Kim, D.J., Babuška, I. (2007). A global-local approach for the construction of enrichment functions for the generalized fem and its application to three-dimensional cracks, in: V. L. Ao, C. Alves, C. A. Duarte (eds.), Adv. Mesh. Tech., 1–26.

Dunant, C., Vinh, P.N., Belgasmia, M., Bordas, S., Guidoum, A. (2007). Architecture tradeoffs of integrating a mesh generator to partition of unity enriched object-oriented finite element software, Revue Europ. de Mécanica Numérico 16(2): 237–58.

Evangelista, F., Roesler, J.R., Duarte, C.A. (2013). Two-scale approach to predict multi-site cracking potential in 3-d structures using the generalized finite element method, International Journal of Solids and Structures 50: 1991–2002.

Fonseca, F.T., Pitangueira, R.L.S. (2007). An object oriented class organization for dynamic geometrically nonlinear, Tech. rep., CMNE (Congress on Numerical Methods in Engineering)/CILAMCE 2007, Porto, 13-15 June, Portugal.

Fries, T.P., Belytschko, T. (2010). The extended/generalized finite element method: An overview of the method and its applications, International Journal of Numerical Methods in Engineering 84(3): 253–304.

Gupta, V., Kim, D.J., Duarte, C.A. (2012). Analysis and improvements of global-local enrichments for the generalized finite element method, Computer Methods in Applied Mechanics and Engineering 245-246: 47–62.

Gupta, V., Kim, D.J., Duarte, C.A. (2013). Extensions of the two-scale generalized finite element method to nonlinear fracture problems, International Journal for Multiscale Computational Engineering 11(6): 581–96.

Hirai, I., Uchiyama, Y., Mizuta, Y., Pilkey, W.D. (1985). An exact zooming method, Finite Elements in Analysis and Design 1: 61–9.

Horstmann, C.S., Cornell, G. (2008). Core Java - Volume I - Fundamentals, vol. 1, 8th ed., Sun Microsystems Press.

Kim, D.J., Duarte, C.A., Pereira, J.P. (2008). Analysis of interacting cracks using the generalized finite element method with global-local enrichment functions, ASME Journal of Applied Mechanics 75: 763–813.

Kim, D.J., Duarte, C.A., Proença, S.P.B. (2009). Generalized finite element method with global-local enrichments for nonlinear fracture analysis, International Symposium on Mechanics of Solids in Brazil, H.S.C. Mattos and M. Alves (Editors), 317–30.

Kim, D.J., Duarte, C.A., Proença, S.P.B. (2012). A generalized finite element method with global-local enrichment functions for confined plasticity problems, Computational Mechanics 50: 1–16.

Kim, D.J., Pereira, J.P., Duarte, C.A. (2010). Analysis of three-dimensional fracture mechanics problems: A two-scale approach using coarse generalized fem meshes, International Journal of Numerical Methods in Engineering 81: 335–65.

Lee, N.S., Bathe, K.J. (1993). Effects of element distortions on the performance of isoparametric elements, International Journal of Numerical Methods in Engineering 36: 3553–76.

Liszka, T.J., Duarte, C.A., Tworzydlo, W.W. (1996). hp-meshless cloud method, Computer Methods in Applied Mechanics and Engineering 139: 263–88.

Melenk, J.M., Babuška, I. (1995). The partition of unity finite element method, Technical Report BN-1185, Inst. for Phys. Sci. Tech., University of Maryland, USA.

Melenk, J.M., Babuška, I. (1996). The partition of unity finite element method: Basic theory and applications, Computer Methods in Applied Mechanics and Engineering 39: 289–314.

Melenk, J.M., On generalized finite element methods, PhD dissertation, University of Maryland, College Park (1995).

Neto, D.P., Ferreira, M.D.C., Proença, S.P.B. (2013). An object-oriented class design for the generalized finite element method programming, Latin American Journal of Solids and Structures 10(6): 1267–91.

Nistor, I., Pantale, O., Caperaa, S. (2008). Numerical implementation of the extended finite element method for dynamic crack analysis, Advances in Engineering Software 39(7): 573–87.

Noor, A.K. (1986). Global-local methodologies and their application to nonlinear analysis, Elements in Analysis and Design 2: 333–46.

Oden, J.T., Duarte, C.A., Zienkiewicz, O.C. (1998). A new cloud-based hp finite element method, Computer Methods in Applied Mechanics and Engineering 153:117–26.

Pereira, J.P., Kim, D.J., Duarte, C.A. (2011). A two-scale approach for the analysis of propagating three-dimensional fractures, Computational Mechanics 49(1): 99–121.

Plews, J.A., Duarte, C.A. (2015). Bridging multiple structural scales with a generalized finite element method, International Journal of Numerical Methods in Engineering 102(3-4): 180–201.

Strouboulis, T., Babuška, I., Copps, K. (2000a). The design and analysis of the generalized finite element method, Computer Methods in Applied Mechanics and Engineering 181(1-3): 43–69.

Strouboulis, T., Copps, K., Babuška, I. (2000b). The generalized finite element method: an example of its implementation and illustration of its performance, International Journal of Numerical Methods in Engineering 47: 1401–17.

Strouboulis, T., Copps, K., Babuška, I. (2001). The generalized finite element method, Computer Methods in Applied Mechanics and Engineering 190: 4081–193.