# Object oriented approach to consistent implementation of Meshless and classical FEM

**Albert Seidl and Thomas Schmidt**
University of Applied Sciences Magdeburg-Stendal
Breitscheidstraße 2
39114 Magdeburg

**ABSTRACT**

Numerical experiments show that the full potential of the Finite Element Method (FEM) can be exploited by combination of classical with meshless FEM. A class structure for flexible consistent implementation of both methods is presented.

Fully automatized 3D mesh-generation still constitutes a serious problem in software development concerning FEM. In the recent years various methods of meshless FEM have been developed as an alternative to overcome this problem. In this work meshless and classical FEM have been implemented.

A further objective of this work is to implement different classical and meshless methods together with an appropriate mesh/point-set generation method. An appropriate class structure for realizing this in a consistent manner with classical FEM is developed and implemented in C++. The performance of the discussed methods was tested with problems relevant in electrical and civil engineering i.e. static electrical field calculations (Poison equation) and elasticity problems.

## 1. MESHLESS METHODS ARE NOT COMPLETELY „MESHLESS“

Some of the mesh objects necessary in classical FEM are also necessary for the implementation of meshless methods:

(1) In both, meshless and classical FEM the solution has to be represented on a point-set (nodes).
(2) The functionals containing the shape-functions and their partial derivatives (e.g. the bilinear form) have to be integrated over the entire domain. One way to do this is to use Gaussian Integration. For this purpose the domain has to be divided into elements. An alternative would be to use the so-called nodal integration as described in [3]. This is an integration method which only uses the nodes for integration not requiring additional Gauss-Points. In this case more recent papers [7,8] have pointed out the necessity to use Voronoi cells in order to accurately determine weight factors for nodal integration. The necessity of Voronoi cells implies the existence of appropriate neighborhood information. If this neighborhood information is interpreted as an explicit connection between the respective nodes in form of an edge we again end up with a regular mesh – in this case the Delauny triangulation.

Independent from the numerical Method the concept of an „Element" or "Cell" as a set of nodes was maintained throughout this work for data management purposes.

## 2. MESHLESS METHODS IMPLEMENTED

**Definitions and Notation**
In this paper vectors will be indicated by one and matrices by two underlines, e.g. $\underline{u}$ denotes a vector and $\underline{\underline{A}}$ denotes a matrix. The following index-definitions will be used

*I=1...n* - number of node
*j=1...m* - number of polynomial coefficient used

**Approach to Solve Partial Differential Equations**
In this work electrical field problems and elasticity problems have been treated. For a better understanding the governing equations will be presented using the Poison Equation in two dimensions as an example. This equation reads in its strong form:

$$\nabla(k\nabla u) = f \quad (1)$$

where $k$ and $f$ are given functions of $x,y$. The solution to be computed $u\ (x,y)$ denotes the electrostatic potential. Correspondingly the displacement would be computed in case of an elasticity problem. For all FEM addressed in this paper the solution will be based on the weak of Eq. (1):

$$I_B + I_P + I_r = \min \quad \textbf{and} \quad I_d = 0$$

with

$$I_B = \frac{1}{2}\int\int_\Omega k(x,y)(u_x^2 + u_y^2)dx,dy \quad \text{bilinear Form}$$

$$I_P = \int_\Omega f(x,y)udxdy$$

$$I_r = -\int_{\Gamma u}[\lambda u]ds \quad \text{and} \quad I_d = -\int_{\Gamma u}\left|u - \bar{u}\right|ds$$

The EFG Method (Belytschko) [1] and PIM (Liu) [2] have been implemented together with classical Finite Elements. The concept of approximating the solution by a weighted set of shape-functions is common to these methods. The set of spatial coordinates is written „$\underline{x}$‘ in the following and can be a vector with 1-3 components.
$u\ (\underline{x})$ denotes the solution. Physically the solution may be potentials in an electrical field problem or displacements in an elasticity problem. For the numerical solution approximated values of $u$ at discrete points (nodes) are considered. The vector $\underline{u}$ denotes the function-values on these points. The discretized approximation of the solution is obtained by:

$$u^h(\underline{x}) = \Phi^T(\underline{x})\underline{u} \tag{2}$$

The component $\Phi_I$ of the Vector $\underline{\Phi}$ is called the shape-function attributed to node $I$. Usually these shape-functions take non-zero values in the influence domain of the respective node only. In classical FEM the shape-function $\Phi_I(x)$ is confined to the element that contains the node I. In meshless methods such rules apply in a more general way introducing the concept of „domain of influence“ [2].

**Outline of Element-free Galerkin (EFG)**
EFG-shape-functions are constructed by interpolating the solution by polynomial functions using moving least squares. EFG uses a weight function attributed to node I $w_I := w_I\ (\underline{x})$ which is non-zero in the environment of node I. If a point $\underline{x}$ is close enough to node I such that $w_I(\underline{x}) \neq 0$ then the node will be used to construct shape-function-values at point $\underline{x}$, otherwise not.

The regression polynomial can be written as a scalar product:

$$u(\underline{x}) = p^T(\underline{x})\underline{a} \tag{3}$$

with $\underline{a}=(a_1,a_2,...,a_m)^T$ being coefficient vector and $\underline{p}$ a vector containing a set of m basis-functions
$p_1(x), p_2(x),..., p_m(x)$.
Example: with $p^T(x) = (1, x, x^2)$ the interpolation polynomial becomes: $u(x) = a_1 + a_2 x + a_3 x^2$
The following weight matrices are introduced:

$$\underline{\underline{V}} = \begin{pmatrix} \sqrt{w_1(x)} & \dots & 0 \\ \dots & \dots & \\ 0 & \dots & \sqrt{w_n(x)} \end{pmatrix}; \qquad \underline{\underline{W}} := \underline{\underline{V}}^T \underline{\underline{V}} \tag{4}$$

$\underline{\underline{W}}$ and $\underline{\underline{V}}$ are square diagonal matrices; their elements depend on $\underline{x}$. We now regard the lease square fit expressed by the over-determined equation system:

$$\underline{\underline{V}}\,\underline{\underline{H}}\,\underline{a} = \underline{\underline{V}}\,\underline{u} \tag{5}$$

where:

$$\underline{\underline{H}} = \begin{pmatrix} \underline{p}^T(\underline{x}_1) \\ \underline{p}^T(\underline{x}_2) \\ \dots \\ \underline{p}^T(\underline{x}_n) \end{pmatrix}, \quad \underline{u} = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{pmatrix}$$

note that:

- $\underline{\underline{H}}$ does not depend on $\underline{x}$
- At least m+1 elements of the vector $\underline{\underline{V}}\underline{u}$ must be non-zero for Eq.(5) to become a least square fit.

An over-determined equation-system has no solution, but the error squares can be minimized. This is done by multiplying the transposed of the coefficient-matrix from the left side.

$$\left(\underline{\underline{V}}\,\underline{\underline{H}}\right)^T \underline{\underline{V}}\,\underline{\underline{H}}\,\underline{a} = \left(\underline{\underline{V}}\,\underline{\underline{H}}\right)^T \underline{\underline{V}}\,\underline{u}$$

Using elementary matrix operations this expression can be transformed as follows:

$$\underline{\underline{H}}^T \underline{\underline{V}}^T \underline{\underline{V}}\,\underline{\underline{H}}\,\underline{a} = \underline{\underline{H}}^T \underline{\underline{V}}^T \underline{\underline{V}}\,\underline{u}$$

or using the definition of $\underline{\underline{W}}$ in Eq.(4):

$$\underline{\underline{H}}^T \underline{\underline{W}}\,\underline{\underline{H}}\,\underline{a} = \underline{\underline{H}}^T \underline{\underline{W}}\,\underline{u}$$

which leads to the interpolation relationship

$$\underline{a} = \underline{\underline{Q}}\,\underline{u} \quad \text{with} \quad \underline{\underline{Q}} = \left(\underline{\underline{H}}^T \underline{\underline{W}}\,\underline{\underline{H}}\right)^{-1} \left(\underline{\underline{H}}^T \underline{\underline{W}}\right)$$

In the following we use: the vector $\underline{Q}_I$, the $I^{th}$ column out of $\underline{\underline{Q}}$, which can be computed by

$$\underline{Q}_I = \left[\left(\underline{\underline{H}}^T \underline{\underline{W}}\,\underline{\underline{H}}\right)^{-1} \left(\underline{\underline{H}}^T \underline{\underline{W}}\right)\right]_I = \left(\underline{\underline{H}}^T \underline{\underline{W}}\,\underline{\underline{H}}\right)^{-1} \left(\underline{\underline{H}}^T \underline{\underline{W}}\right)_I$$

$$\underline{Q}_I = \left(\underline{\underline{H}}^T \underline{\underline{W}}\,\underline{\underline{H}}\right)^{-1} \underline{p}(\underline{x}_I) w_I$$

where $\underline{x}_I$ denotes the coordinate-set attributed to the node $I$. The shape-function attributed to node $I$ finally becomes

$$\Phi_I(\underline{x}) = \underline{p}^T(\underline{x})\underline{Q}_I(\underline{x})$$

This shape-function is not a polynomial but rather a rational function. Like in the case of classical FEM the shape-functions of nodes neighboring to any point in the plane add up to one (partition of unity). Due to usage of appropriate weight functions $w_I(\underline{x})$ conformity is guaranteed. In contrast to classical FEM an element where a particular shape-function becomes non-zero is not defined.

For details of the implementation, especially the computation of partial derivatives of the shape-functions the reader is referred to [1]. Please note that the computational effort for evaluation of the shape-functions is considerably high.

**Outline of Point Interpolation Method (PIM)**

Let $\underline{x}$ be a point where a set of shape-functions is to be evaluated. A set of neighbor-nodes is chosen for interpolation. A variety of different possible criteria for choice of neighbor nodes are given in [2]. The main difference to EFG is the fact that the number of neighbor-nodes is equal to the number of basis functions. Therefore a weighting of nodes is not possible. The regression polynomial of Eq. (3) becomes an interpolation polynomial. Instead of Eq.(5) we get the following interpolation relationship:

$$\underline{\underline{H}}\,\underline{a} = \underline{u} \tag{6}$$

Since this equation system is not over-determined (as is EFG) the construction of shape-functions is straightforward:

$$\underline{\underline{Q}} = \underline{\underline{H}}^{-1}$$

$$\underline{\Phi}(\underline{x}) = \underline{p}^T(\underline{x})\underline{\underline{Q}}$$

$$\Phi_I(\underline{x}) = \underline{p}^T(\underline{x})\underline{Q_I}$$

PIM is based on polynomial interpolation like classical FEM, however the choice of neighbor nodes of a point in the plane to be used for interpolation is not defined by any element. As a consequence neighbor-nodes and the degree of interpolation polynomial can be chosen freely without any changes in the mesh. The cost of this advantage is the loss of conformity or the additional measures (i.e. Penalty Methods [2]) to maintain conformity.

In this work PIM were found to be as efficient as classical FEM, however EFG were found to be particularly robust against local degeneracies. Usage of low (linear) or high (quadratic) order basis functions are flexibly possible on any mesh. However computation times of EFG were found to be considerably larger than in the case of classical FEM. In this context it should be mentioned that EFG can be accelerated considerably by nodal integration at the expense of flexibility [3,4].

**Implementation of classical FEM**

Due to the parallels between PIM and classical FEM the latter was implemented as a special case of PIM. An alternative neighborhood-relationship was implemented such that the corners/mid-side-nodes of the element itself are used for interpolation only.

**Combination elements**

As will be shown in section 5 the full potential of meshless FEM can only be exploited by combination of multiple methods. As an alternative to a coupling of different domains over internal boundaries, the introduction of transition elements allows for flexible coupling of different methods within one domain. In a domain where mixed approaches have to be used, every meshpoint is attributed a shape-function type. If different type-indices are found within one element, shape-functions of all types in question have to be evaluated within the element. Consequently inside the element the shape-functions

$$\Phi_{i,j}(x,y)$$

have to be computed where i denotes the local node-number and j the shape-function type index. For superposition of the shape-function components weight-functions are used which constitute a partition of unity:

$$\sum_{j=1}^{3} w_j(x,y) = 1$$

$w_j(x,y)$ here denotes the weight-function attributed to shape-function-type of node with local number j. The resulting combined shape-function can then be computed by:

$$\Phi_i(x,y) = \sum_{j=1}^{3} \Phi_{i,j} \cdot w_j(x,y)$$

All of the above described shape-function types can be combined using this concept thus allowing flexible combination of all different methods including classical FEM. In contrast to coupling over domain boundaries the weight-function strategy realizes a smooth transition between regions where different shape-functions are used.

## 3. CLASS STRUCTURE

Before the class structure is described very brievly the several computational steps for FEM / Meshless FEM are summarized in a flow chart (*Fig.1).*

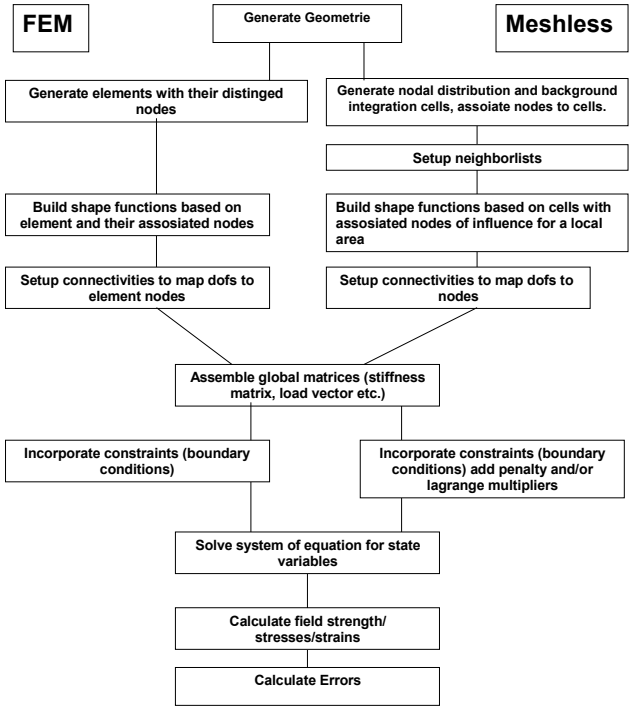


*Fig.1.: flow chart for FEM and meshless FEM computational calculations*

The class structure proposed in this paper is shown in *Fig. 2* as UML-class diagram. The diagram includes only the most important classes that reflect the classical „three phases"-model -Pre-Processing, Processing, Post-Processing - for compuational FEM calculations. Utility classes i.e. for handling the FEM/meshless data are not shown.

All three phases are included in the CProcessor class. This wrapps the hole computational process.

The classes for the Pre-Processing phase are CMesh2D which generates rectangular elements/cells (class CElement) and CGeoTree which generates arbitrary meshes. For meshless FEM special methods in class CMesh2D or CGeoTree are necessary to setup neighborlists and the connectivities between the nodes and the corresponding degrees of freedom.

The Processing phase is modelled by the classes CAssembler, CIntegrator, CShapeFunc, CLinSystem.
The CAssembler class performs the assembly of the global stiffness matrix and the constraints. The integration of the element/nodal stiffness is done by the CIntegrator class. The integrator class was developed to perform Gaussian integration of the weak form. The integrator in turn uses the CShapeFunc class to calculate the shape functions, the function values and derivates at the nodes and/or the integration points. The assembler also builds the „global load vector“ with help of the CIntegrator.

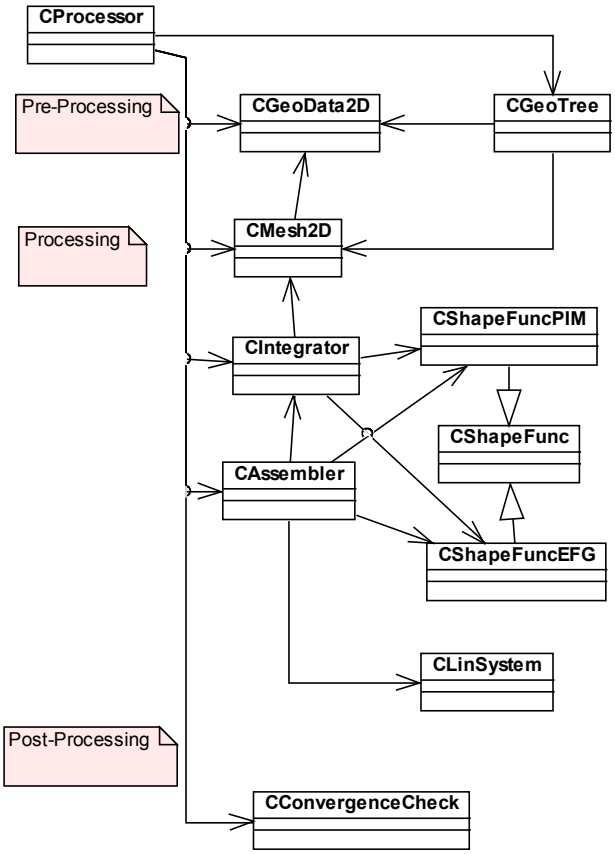


*Fig.2.: UML-Class-Diagram of FEM and meshless FEM Design*

After the global matrices are built the CLinSystem class is used to solve the linear system of equations. The so obtained nodal state variables are then used to calculate the field strength/strains/stress for each integration point. If the EFG shape functions are used then a special step is necessary to build the final nodal values of the state variables.
The last phase is performed within the CProcessor class by the method CheckSolution. This will be moved later to a special Post-Processor class.

In classical Finite Elements the element (class CElement) has a defined mathematical meaning. In meshless Finite Elements (class CElement) it makes sense to maintain the abstract concept of an element/cell as a set of points for data-management reasons.

The class-structure was derived straightforward from the mathematical representation of the methods. The formulation of the weak form is the same for all methods implemented in this work.

The main difference between FEM and meshless FEM is represented by the shape-function builder. Each method uses its special derived shape function class which inherits from CShapeFunc. This general interface for the shape-function builder was defined in form of a purely virtual class. Any particular shape-function builder is derived from this class and is called by the integrator (and other modules) independent on the method realized. An internal status management avoids the class to be called in the wrong way. A test-class for the shape-function checks correctness of function evaluation and its partial derivatives, partition of unity and consistency. Thus errors are eliminated to the widest possible extent before including the shape-function-builder into the FEM software.

The combined-shape-function (see chapter 2, section on combination elements) was realized completely within the concept of the shape-function-builder class as outlined in this section. Internally the combined shape-function builder sets up an array of instances of the individual type-classes needed. Thus any new shape-function builder added to our software can immediately be included into the combination elements.

## 4. MESH/NODE-SET GENERATION

A quadtree strategy has been realized to generate the node-set for arbitrary geometries. In this work a 2D node-generation was implemented. Within the scope of this paper the node-set generation was implemented in 2D to investigate the general applicability of meshless methods (EFG) to problems arising from engineering science. To extend the developed program for 3D is trivial because of the class structure proposed above.

Since a Voronoi triangulation based on arbitrary node-sets is trivial, any mesh generated by the described method can be readily used for classical FEM with triangular elements. However no special treatment of the mesh to avoid degenerate elements was implemented because it was the scope of this paper to overcome this problem by local usage of EFG.

The Mesh (consisting of nodes and elements, CMesh2D) together with all mutual connectivity lists constitutes an additional class, which is accessed by shape-function builder and assembler. All neighbor-search operations in the mesh are of complexity *O(n)* provided that the different neighbor-lists are set up in the appropriate sequence. The quadtree (CGeoTree) constitutes an additional class and is discarded after mesh-generation.

## 5. TEST RESULTS

Our tests of the implemented methods can be summarized as follows: EFG shows good robustness against irregular node-positioning (in contrast to - for example - classical FEM which suffers from poorly shaped triangles in a delauny-mesh). The shortcomings of EFG are its high computational cost for shape-function computation. Point Interpolation Methods (PIM) are just as efficient as classical Finite Elements, however in our experiments they were found to be sensitive against poor node-positioning.

The mesh was varied between a completely regular mesh and a strongly degenerate one. The degree of degeneracy could be varied using a parameter between 0 (regular) and 1.2 (degenerated) as shown in *Fig.3*. For test purposes a Dirichlet Problem was solved with a known solution:

$$u(x, y) = e^{-y} \cdot \sin x$$

In order to test the sensitivity of the respective method against the quality of the node-set used, different patterns were generated. The structure of these patterns is characterized as follows:

| Abreviation | Description |
|---|---|
| eq. | equidistant, rectangular |
| neq. | nonequidistant rectangular |
| Rnd | random-disturbed node-set |
| Quad | quadratically distorted node-set |

*Tab. 1: characterization of node-sets used for evaluation of meshless methods*

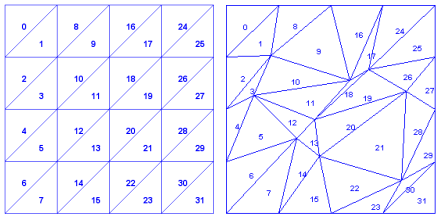

*Fig.3.: regular and degenerate mesh for sensitivity test of classical FEM versus EFG. The degenerate-Parameter is defined to be 0 for the regular and 1.2 for the degenerate mesh.*

EFG was tested against classical FEM in order to investigate the sensitivity against degenerate node-sets. The errors on the nodes were evaluated; their arithmetic mean and maximum norm were computed. The results for the mean error norm are shown in *Fig.4*. It can be seen that EFG method exhibits stable behavior for degenerate meshes.

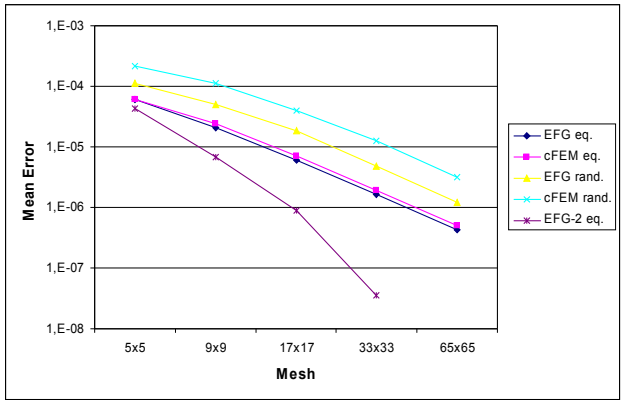


*Fig. 4: mean error of the test-problem versus mesh size for classical FEM (cFEM) and EFG. EFG-2 refers to shape-functions obtained with a 2nd degree basis function*

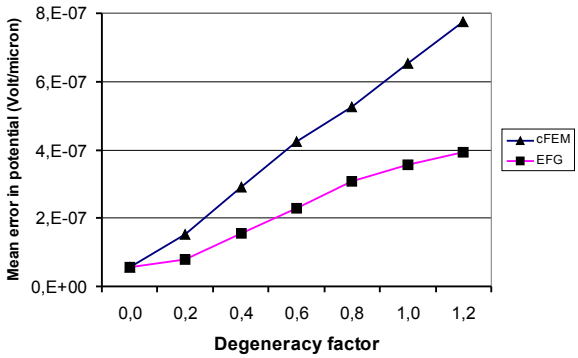


*Fig. 5: mean error of the test-problem versus mesh-degeneracy for classical FEM (cFEM) and EFG.*

The same test was repeated by varying a degeneracy-parameter. The definition of the degeneracy-parameter is shown in Fig.3, the results are shown in Fig.5. It can be seen that the advantage of EFG becomes particularly significant (reduction of error by a factor of 2 compared with classical FEM) in the case of a strongly degenerate node-set.

The test of PIM was done in a similar way. Fig. 6 summarizes the test-results. Results obtained with classical FEM on different node-sets as characterized according to Tab.1 are shown for comparison. Nonconforming PIM were found to exhibit poor solution quality. Only on rectangular meshes the accuracy could be obtained as to be expected for the high order shape-functions used. Our experiments have shown that in this case PIM becomes conforming for symmetry reasons, even if no measures are taken to enforce conformity. For meshes lacking these symmetry properties a penalty function strategy has been proposed by Liu [2] (CPIM). Consequently, results obtained using CPIM also are superior to those obtained with classical FEM. However, the penalty strategy requires the usage of an additional parameter, the proper choice of which also turned out to be difficult. PIM therefore where not used within our applications outlined in section 6.

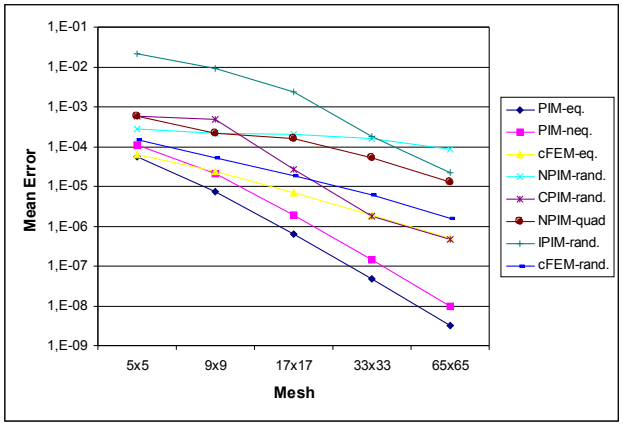


*Fig. 6: Test of different variants of PIM depending on the mesh in comparison with cFEM*

Based on the test-results we can summarize that EFG has been shown to be particularly resistant against the effects of odd geometries and extreme local mesh-refinement such as the appearance of single degenerate elements. However, EFG that has been tested to be the most robust is less efficient than classical Finite Elements. Therefore a combination of both methods with local use of EFG is proposed as a result of this work.

## 6. APPLICATION IN ELECTRONICS CAD

Extraction of maximum current densities from the layout is becoming increasingly important for reliability issues. For this purpose electric field calculations are necessary. In contrast to field calculations used for capacitance and conductance computation, careful geometry modeling and extreme fine-grained local mesh refinement is crucial to achieve acceptable results. If simply the layout geometries with their idealized

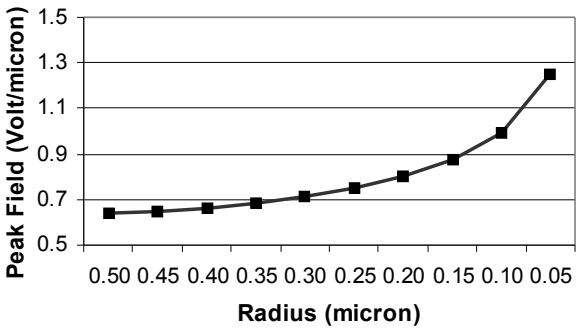


*Fig. 7. Dependency of computed peak field-strength on radius of curvature of a critical corner for a given structure.*

corners are used, the computed maximum current density merely reflects the mesh size at the corner. In order to obtain reliable data

- corner rounding must be modeled based on technology data, and
- local mesh refinement must provide accurate description of the potential gradient in proximity to the corner.

These requirements in combination with the need for efficient simulation software lead to extreme local variations in the element size of the mesh used for the simulations.

A detailed discussion of accuracy problems in combination with extreme local mesh refinement can be found in [9]. Results of these investigations have shown that a minimum of 10 mesh-points per is unit are necessary to achieve reasonable accuracy. Fig. 7 shows that computed field strength can vary by a factor of more than two, depending on the radius of curvature of the structure in question. If extreme local mesh-refinement (up to four orders of magnitude) is required it cannot be completely avoided that locally degenerated elements arise if classical FEM would be used. EFG was applied locally to overcome this problem.

The CAD Program PARIS [9] was equipped with a solver for extraction of maximum current densities of interconnects. Due to efficiency considerations classical FEM was used in the main part of the domain. EFG were employed in the critical regions only. The combined elements as outlined in section 3 were used to implement both element-types simultaneously. Fig 8 shows the entire domain for an application example (power line) together with local refinement (zoom in Fig. 9).

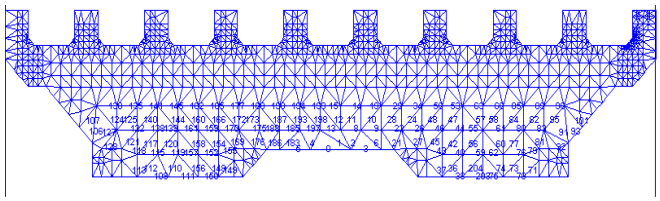

*Fig.8: Geometry of a ground-line together with triangular mesh used for numerical field calculation*

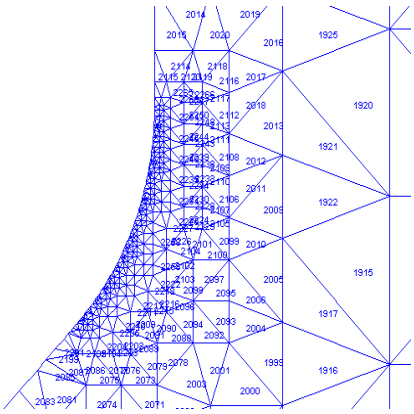

*Fig.9: magnified region of a critical 45° corner with local mesh-refinement*

## 7. CONCLUSION

Meshless methods have been implemented in combination with classical FEM. By empirical tests they have been shown to be particularly resistant against the effects of odd geometries and the effects of extreme local mesh-refinement such as the appearance of single degenerate elements. However EFG that have been tested to be the most robust are less efficient than classical Finite Elements. Therefore a combination of both methods with local use of EFG is proposed as a result of this work. An object oriented flexible class structure to meet this requirement is proposed.

## 8. ACKNOWLEDGEMENT

This research was sponsored by the German Ministry of Research under grant no. 1705503.

## 9. REFERENCES

[1] Belytschko, T.; Lu, Y.Y. and Gu, L.: „Element-Free Galerkin Methods“, International Journal for Numerical Methods in Engineering, vol. 37, 1994, pp.229-256

[2] G.R. Liu: *Meshfree Methods - Moving beyond the Finite Element Method,* 712 pages, 2002, CRC Press. ISBN: 0849312388

[3] Beissel, S.; Belytschko T.: „Nodal Integration of the element-free Galerkin Method“, Computer Methods in Applied Mechanics and Engineering, 1996, 139: 49-74

[4] Martin Larcher private communication

[5] Zienkiewitz, O.C.: „Methode der finiten Elemente“, Carl-Hanser Verlag, München 1984

[6] Chen, J.S.; Wu, C.T.; Yoon, S.; Yu, J.: „A stabilized conforming nodal integration for Galerkin mesh-free methods“, Int. J. Numer. Meth. Engng. 2001; 50:435-466

[7] Schmidt, Th.; Abschlussbericht: Effektive Implementierung gitterloser Finite Elemente Methoden, Forschungssemester, Abschlußbericht

[8] Seidl, A; Schmidt, Th.: "Gitterlose FEM", Abschlußbericht des Forschungsvorhabens FKZ 1705503

[9] Seidl, A. ; Schnattinger, Th. ; Erdmann, A. ; Hartmann, H; Petrashenko, A.: "Accurate extraction of maximum current densities from the layout", presented at IWCE 11, Vienna, 2006