ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

AUTOMATIC CONFORMAL DECOMPOSITION OF ELEMENTS CUT BY NURBS

Jakob W. Steidl[1] **and Thomas-Peter Fries**[1]

[1]Institute of Structural Analysis, Graz University of Technology
Lessingstrae 25/II, 8010 Graz, Austria
e-mail: {steidl, fries}@tugraz.at

Keywords: conformal decomposition, fictitious domain method, NURBS, signed distance, level set method

Abstract. *In fictitious domain and extended finite element methods, elements cut by boundaries or interfaces are frequently present. They pose challenges with respect to the application of boundary and interface conditions, numerical integration, and the conditioning of the system of equations. A different path is the automatic decomposition of cut elements into smaller, conforming finite elements. Herein, we propose a concept for the automatic decomposition of elements cut by (curved) NURBS which automatically generates higher-order Lagrange elements on the two sides of the interfaces. The NURBS data, defining boundaries and interfaces in two dimensions, is converted into implicit level-set data by evaluating signed distance computations at all nodes in the background mesh. Based on the findings in [1] this data is then used for the decomposition into sub-elements. First numerical results are achieved as a proof-of-concept.*

1 INTRODUCTION

Within the scientific community of numerical engineering, recently efforts are made to incooperate exact geometry definitions delivered from designers. The reasons for including the exact geometry definition are due to the tighter integration of simulation tools within design software. However, also problems sensitive to the geometry, such as buckling of shells or in fluid mechanics can benefit.

An effort almost as old as the finite element method itself is the automatic generation of meshes suitable for simulations. While proven meshing approaches are available for some time now (see [2] for an historic overview), in practice, generated meshes are often manually reworked by the user. This intervention, however, is often a problem because the link to the original geometry description is lost, which makes automatic mesh re-generation as in refinement or in moving interface problems difficult.

Also classic mesh triangulation is time consuming and yields a high number of nodes if complex boundaries or interfaces have to be discretized with high accuracy. These two problems are tackled by a number of methods that do not capture boundaries explicitly in terms of element edges. This allows the use of a structured *background mesh* with a wealth of potential speedups in computation time (e.g. constant Jacobians). The trade-off is that elements from the background mesh intersected by a boundary or interface require special treatments.

The kind of treatment of cut elements is what the methods differ in. For example, in the extended Finite Element Method (XFEM) [3], the approximation space is enriched by additional shape functions tailored to capture discontinuities across interfaces. Thereby cut elements yield additional degrees of freedom and customized integration schemes are required.

Another class of methods which naturally involve elements cut by interfaces are immersed boundary methods [4, 5]. The 'interface' is then rather the boundary of the domain and, herein, we do no longer distinguish between interfaces and boundaries. These methods embed the original domain into a much simpler, usually rectangular, domain spanned by the background mesh. The methods of this class differ in the treatment of boundary conditions, numerical integration and conditioning issues in the resulting system of equations.

A different approach to handle these intersected elements is to decompose them into subelements that *conform* to the boundary or interface and is referred to as the Conformal Decomposition FEM (CDFEM) in [6]. The advantage of their approach is that new nodes are being introduced to the mesh alongside the interface, which makes the application of boundary conditions trivial. However it also becomes necessary to modify the mesh and to (at least partially) give up the advantages of the structured background mesh. So far conformal decomposition has only been reported using linear elements for reconstruction.

Herein, we are concerned with the extension to higher-order accuracy. The interfaces are defined by NURBS in the beginning. This information is converted into implicit level-set data by performing signed-distance computations. Then, based on the approach in [1] elements cut by the zero-level set are automatically decomposed into sub-elements.

2 TOWARDS A HIGHER-ORDER CONFORMAL DECOMPOSITION METHOD

The aim is to develop a higher-order accurate CDFEM, which uses NURBS for the definition of interfaces. NURBS are able to represent all surface types common in engineering exactly.

The main part of the paper deals with converting the NURBS into level-set data by evaluating distances of nodes in the background mesh to the NURBS. Then, the approach in [1] is used to generate the sub-elements in cut background elements.

2.1 Level Set Method

For implicit interface descriptions, the level set method [7] is frequently used. In this framework interfaces Γ can be described as the zero-isoline of a scalar function $\phi(\mathbf{x})$, i.e.

$$\Gamma = \{\mathbf{x} \mid \phi(\mathbf{x}) = 0\}, \tag{1}$$

where $\mathbf{x}$ describes arbitrary points in space and ϕ typically is the shortest distance from $\mathbf{x}$ to Γ multiplied by a sign to denote the side of Γ where $\mathbf{x}$ lies. In its discrete form, used for numerical analysis, values for ϕ are usually stored at the mesh nodes, that is, for every node in the mesh the distance to Γ needs to be computed. In between the nodes, level set values are interpolated based on the shape function of the background mesh.

2.2 Non-Uniform Rational B-Splines

Non-Uniform Rational B-Splines (NURBS) can be seen as B-Splines projected onto a projective plane. This is what allows them to represent arcs and conic sections exactly. B-Splines are a generalization of Beziér curves, which again are a linear combination of Bernstein polynomials. A NURBS curve, often also denoted as univariate NURBS is defined in [8] as

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad u_0 < u < u_m \qquad u \in \mathbb{R}$$

where P_i are the control points, w_i the associated weights and $N_{i,p}(u)$ B-Spline basis functions, which can be defined in their recursive form:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

$$N_{i,0}(u) = \begin{cases} 1 & u_i < u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

Associated with this is the non-decreasing sequence of $U = \{u_0, \ldots, u_m\}$ called the *knot vector* with $m = n + p + 1$. It is common to repeat u_0 and u_m with a multiplicity of $p + 1$ to ensure the endpoints of the curve are interpolatory with the corresponding control points. The linear connection of the ordered $\mathbf{P}_i$ is referred to as the *control polygon*. Properties of NURBS and Beziér curves include:

- endpoints are interpolatory, ie $\mathbf{C}(u_0) = \mathbf{P}_0$ and $\mathbf{C}(u_m) = \mathbf{P}_n$
- strong convex hull property: for $u \in [u_i, u_{i+p})$ will be inside the convex hull $\mathcal{H}$ spanned by the control points $\mathbf{P}_{i-1} \ldots \mathbf{P}_i$. For Beziér curves this holds for the whole curve.

3 SIGNED MINIMAL DISTANCE TO NURBS CURVES

The signed, minimal distance function defines the shortest Euclidean distance from a *test point* $\mathbf{p}_t$ to a given curve $\mathbf{C}(u)$ and can be written as

$$\phi(\mathbf{p}) = \min(\|\mathbf{C}(u) - \mathbf{p}\|) \cdot s \tag{2}$$

with

$$s = \text{sign}(\langle \mathbf{C}(u) - \mathbf{p}_t, \mathbf{n}_{\mathbf{C}(u)} \rangle. \tag{3}$$

where $\|\cdot\|$ denotes the Euclidean norm, $\langle\cdot,\cdot\rangle$ the dot product, $\mathbf{n}_{\mathbf{C}(u)}$ the normal vector of $\mathbf{C}(u)$ and

$$\min(\|\mathbf{C}(u) - \mathbf{p}_t\|) = \|\mathbf{C}(u^*) - \mathbf{p}_t\| \tag{4}$$

with the criterion for the *projection point* $\mathbf{p}_p = \mathbf{C}(u^*)$ to $\mathbf{p}_t$

$$\langle \mathbf{C}(u^*) - \mathbf{p}_t, \mathbf{C}_{,u}(u^*)\rangle = 0 \quad \Longleftrightarrow \quad \|\mathbf{C}(u) - \mathbf{p}_t\|_{,u} = 0. \tag{5}$$

Note that for the criteria in (5) a unique solution is not guaranteed. If no solution exists, $\mathbf{p}_p$ must be one of the curve's endpoints ($\mathbf{C}(u_0)$ or $\mathbf{C}(u_m)$). This case can be handled quite easily. In order to find all u^* that can lead to the solution of (4), tailored root finding algorithms have been developed by the authors and others.

3.1 Root finding algorithms

3.1.1 Literature review

For arbitrary polynomial orders p, there is no closed form solution to (5). A common approach is to solve the equation iteratively via a Newton-Raphson iteration scheme as in

$$u^*_{k+1} = u^*_k \cdot \frac{\|\mathbf{C}(u^*_k) - \mathbf{p}_t\|_{,u}}{\|\mathbf{C}(u^*_k) - \mathbf{p}_t\|_{,uu}}. \tag{6}$$

It is well known, that the iteration will only converge for start values close to the actual solution of (5). Therefore a major task is to find start values $U_k = \{u^*_0\}$ such that ultimately the shortest distance to the NURBS is found. It is noted that depending on the NURBS, several local minimal of (5) may exist. Hence, different start values may lead to different converged solutions of (5). It is, therefore, not a trivial task to define suitable start values. In the CADG community the following procedure has emerged: Decompose the NURBS into a set of (projective) Beziér segments, drop segments that can not contain the projection point, refine remaining segments and repeat with the refinement and dropping procedure until a certain exit criterion is reached.

Refinement: Apart from splitting the Beziér curves, in [9] the Beziér curves are further divided into *simple* (i.e. without a self crossing control polygon) and convex curves.

Culling: In order to reduce the number of Beziér curves, in [10] and [11] simplifications of the Beziér curves' convex hull property and interpolation property are employed: Segments with a distance $\min(\|\mathbf{p}_t - \mathcal{H}\|) > \min(\|\mathbf{p}_t - \mathbf{C}(\{u_0, u_m\})\|$ can be dropped because they can not contain the solution.

Stopping criterion: The stopping criterion proposed in [9] checks if the remaining segments are 'flat' enough to be approximated as a straight line. If so, the test point will be projected on that line to get a start value u^*_k. This, however, is not a sufficient criterion to guarantee only one root of (5) and so far only one sufficient criterion has been reported in [12]. However, it is not trivial to implement and only covers curves in $\mathbb{R}^2$.

3.1.2 Implementation

Herein an algorithm is proposed that is tailored towards the reliable computation of $\phi(\mathbf{p}_t)$ for a larger number of test points and which can be extended easily to surfaces in $\mathbb{R}^3$. The algorithm is outlined as follows:

1. Evaluate NURBS 'often enough' at parameters u_j, so that there is not more than one solution of (5) in between each interval.

2. Determine $d_j = d(u_j) = \|\mathbf{C}(u_j) - \mathbf{p}_t\|$ for each u_j
3. Using piecewise linear interpolation of the discrete distance function d_j find its roots. This yields starting points u_j^* for the NR-iteration.
4. Using the NR-Iteration scheme (6), improve all u_j^* to get all u^* that fulfill (5).
5. Evaluate distance for all $\mathbf{C}(u^*)$ and the endpoints of $\mathbf{C}(u)$ and finally,
6. return $\phi(\mathbf{p}_t)$.

The question of 'how often is often enough' to capture all roots for (5) extensively using randomly generated curves and test points has been studied. The solutions were compared against a brute force algorithm, that iteratively subdivided the NURBS into a polygon until the projection point was determined up to a defined precision.

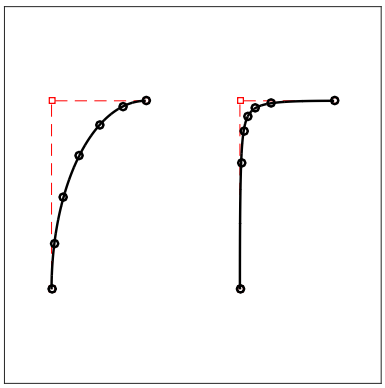

Figure 1: On the right hand curve the middle weight ($w_1/w_0 = 5$) is decreasing the uniformity of the otherwise equally (alongside u) spaced dots.

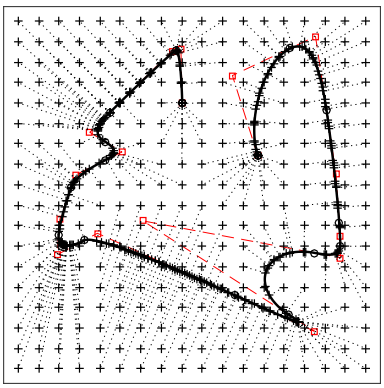

Figure 2: A curve used to validate our algorithm and showing shortest distance from gridded test points.

To arrive at the start values u_k for the bisection we subdivide the interval spanned by the knot vector by a factor $\psi = \psi_p \cdot \psi_{w,i}$ that is different for each interval of the knot vector U. It is based on a base subdivision

$$\psi_p = 2p + 1 \tag{7}$$

and a factor

$$\psi_{w,i} = \frac{w_i}{\min(\{w\})} \tag{8}$$

to account for the 'distortion effect' (see Fig. 1) caused by the weights w_i. Note that $\psi_{w,i}$ only affects the knot spans in $[i, \ldots, i + p - 1]$.

Although this approach leads to a high number of evaluation points on the NURBS, it pays off when ϕ needs to be computed for a high number of test points (i.e. nodes in the mesh). Because in that case the computationally expensive NURBS evaluations is independent from $\mathbf{p}_t$ and hence needs to be done only once for each curve. There is even a further potential of avoiding NURBS evaluations. If one is only interested in ϕ (and not for example $\mathbf{p}_p$ or derivatives of ϕ) then the result of the bisection part can be evaluated directly without the need of a Newton-Raphson iteration. That is because the point found by the bisection is already very close to the exact solution and the curve's tangent is (almost) normal to the vector between $\mathbf{p}_t$ and $\mathbf{p}_P$.

3.2 Extension to Multiple Curves

Interface descriptions may consist of multiple NURBS. It is therefore mandatory to handle (2) for more than one NURBS. Let Φ_i denote the set of $\{\phi(\mathbf{C}_i(u), \mathbf{p}_t)\}$ to a set of multiple curves, then the the sought solution of the minimal signed distance extends to

$$\phi(\mathbf{C}_i(u), \mathbf{p}_t) = \min(|\Phi_i|). \tag{9}$$

It is important to note that there might exist multiple minima for (9). From a practical point of view this is not a problem as long as all minima are of the same sign. In sets of curves which define domains, only for test points that project to endpoints of curves (i.e. in corners made of multiple curves) the found minima might be of different signs.

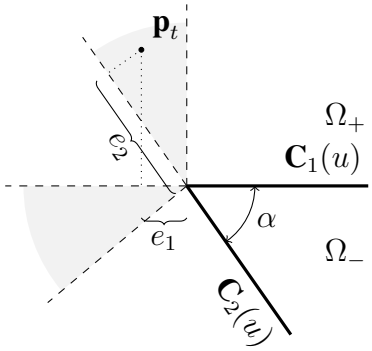

Figure 3: Situation for corners with $0 < \alpha \leqslant 90°$

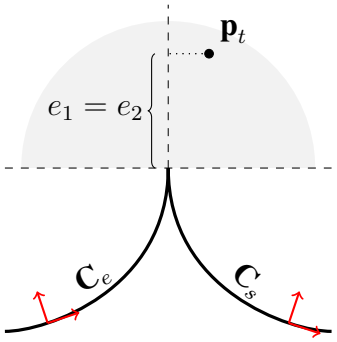

Figure 4: Situation for corners with $\alpha = 0$. The ↳ denote the tangent and normal vectors of each curve.

As seen in Fig. 3 in corners there are two zones where the sign computation (3) would return ambiguous signs for each curve. It is also seen that this zone vanishes for $\alpha > 90°$.

The curve to deliver the correct sign can be determined by

$$\begin{aligned} e_i &= \langle \mathbf{p}_t - \mathbf{p}_{corner}, \mathbf{C}_{i,u}(u) \rangle \qquad i = 1, 2 \\ s &= \text{sign}(\min(e_1, e_2)). \end{aligned} \tag{10}$$

In Fig. 3 it can also be seen, that for $\alpha = 0 \iff e_1 = e_2$. In such cases, it is useful to compare the curvatures of both curves:

$$s = \begin{cases} 1 & \text{for } \kappa_{C,e} > -\kappa_{C,s} \\ -1 & \text{otherwise} \end{cases}, \tag{11}$$

where $\kappa_{C,e}$ and $\kappa_{C,s}$ are the curvatures of the curves whose end-point (respectively, start-point) define the corner. The curvature is defined so that $\kappa > 0$ if the curve bends towards its normal vector. As long as interface lines do not overlap there exists a unique solution to (11).

4 DECOMPOSITION OF CUT ELEMENTS

Once, the signed distance to the interface has been computed for all nodes, this information is used to determine which elements are cut and and then to automatically decompose these elements into conforming sub-elements. The procedure follows ideas presented in the context of integration in [1].

The decomposition algorithm can be outlined as follows: Based on a sample grid in a reference background element, it is determined whether an element is cut or not. Using the element shape functions, the zero-level set in the reference element is detected and meshed by higher-order interface elements (blue line in Fig. 4). Based on how these elements cut the reference background element, topologically different situations result, leading to different sub-cells, see e.g. the gray and yellow elements in Fig. 4. Higher-order elements are then mapped to these sub-cells with one higher-order side so that they conform with the interface.

For a successful reconstruction, a few things need to be considered: To achieve a valid cutting pattern, recursive refinement of the cut elements might be required. To reconstruct the zero-isoline inside the element, a Newton-Raphson scheme with prescribed search paths is used. The suitability of the sub-elements for numerical analysis heavily depends on the search paths used and also on the mappings to the sub-cells.

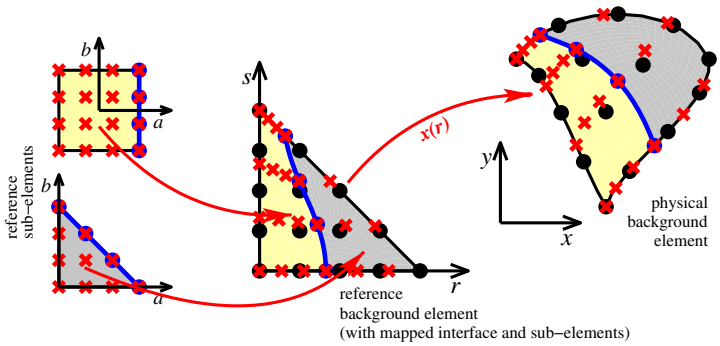

Figure 5: An example of a cut triangular element decomposed into a triangluar and a quadrilateral sub-element. Note the interface element (blue) and the additional mapping $\mathbf{r}(\mathbf{a})$.

5 NUMERICAL STUDY

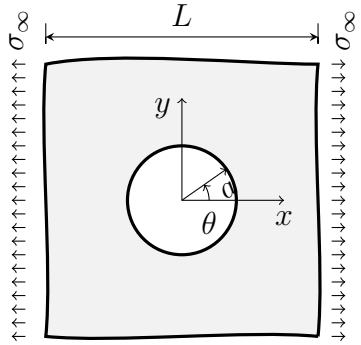

Figure 6: Infinite plate with hole under uniaxial load

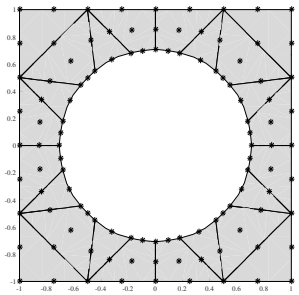

Figure 7: Coarsest mesh used in study with triangular elements in background mesh.

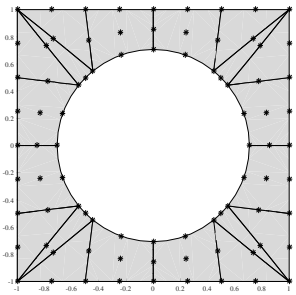

Figure 8: Coarsest mesh used in study with quadrilateral elements in background mesh.

The numerical study validates the conversion from NURBS to level-set data and demonstrates the applicability of higher order conformal decompositions. The study is carried out on an infinite plate under uniaxial load (plane strain) as shown in Fig. 6. In order to reduce the size of the computational domain, exact tractions were prescribed at the boundary. A closed

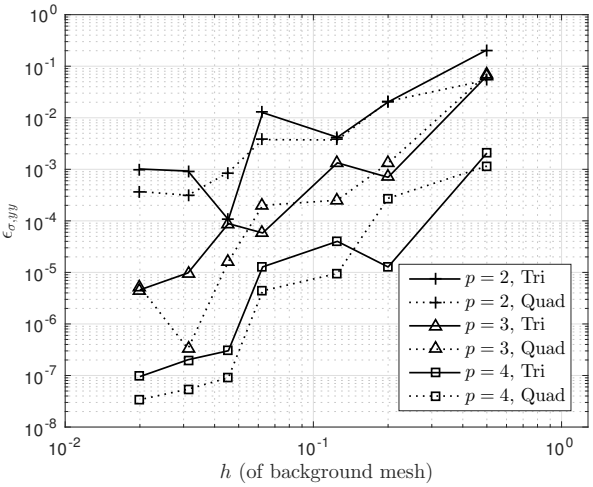

Figure 9: Error in stresses σ_{yy} on left side of hole

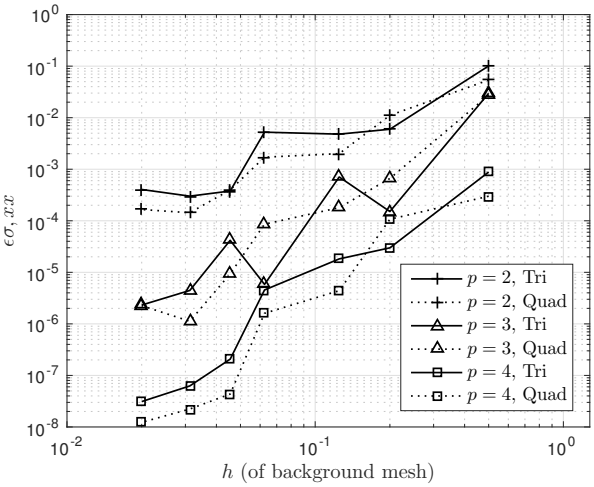

Figure 10: Error in stresses σ_{xx} on top side of hole

form solution for the stresses and displacements within the domain is found in [13]. As material parameters, a Young's modulus of $E = 10000$ and a Poisson ratio of $\nu = 0.3$ are chosen. For the hole a ratio of $L/a = 2\sqrt{2}$ is used. The rate of convergence is studied with triangular and quadrilateral background elements and element orders of $p = \{2, 3, 4\}$ and with different element sizes ($L/h = n = \{4, 10, 16, 32, 44, 64, 100\}$).

In order to avoid an ill-conditioning of the system matrix, the ratio of element areas was limited by moving nodes from the background mesh away from the interface if they were too close.

One measure is the relative error of the stresses on the left and on the top side of the hole

$$\epsilon_{\sigma_{ii}} = \left| \frac{\sigma_{ii}^{h} - \sigma_{ii}^{ex}}{\sigma_{ii}^{ex}} \right| \tag{12}$$

where σ_{xx} is evaluated at $(r = a, \theta = 90°)$ and σ_{yy} at $(r = a, \theta = 180°)$. To get an impression of the overall approximation quality, the total error

$$\epsilon_{\Omega} = \|\mathbf{u}^{ex} - \mathbf{u}^{h}\|_{L2} \tag{13}$$

is also studied.

It can be seen in Figures 9, 10, 11 that the achieved convergence rates are very good, they are optimal in the L_2-norm (13). It is interesting but not surprising that the error for the stresses at the selected points is less straight. This can most likely be attributed to the different element shapes around the nodes in question.

6 CONCLUSIONS

In this paper a 'proof of concept' for a higher-order accurate CDFEM was outlined. It promises to combine fully automated efficient meshing with trivial application of boundary conditions, optimal convergence rates and NURBS-based interface and boundary descriptions. A foundation for this method is the integration scheme proposed in [1].

Herein, the focus was on an algorithm to compute the signed distance to NURBS. Therefore a substantial effort is needed to define start values for the Newton-Raphson iteration required to obtain the minimum distance. In a numerical example, optimal convergence rates suggest that higher order convergence and conformal decomposition do work together.

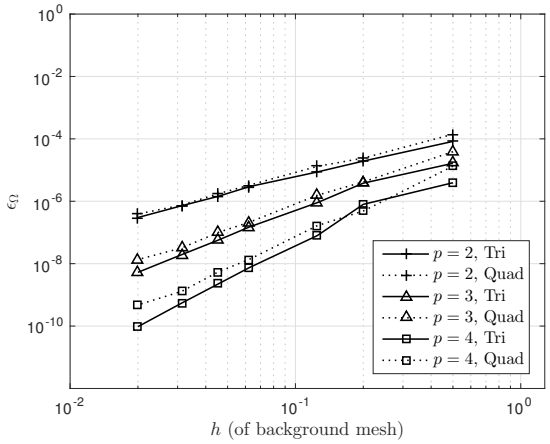

Figure 11: Overall approximation error ϵ_Ω

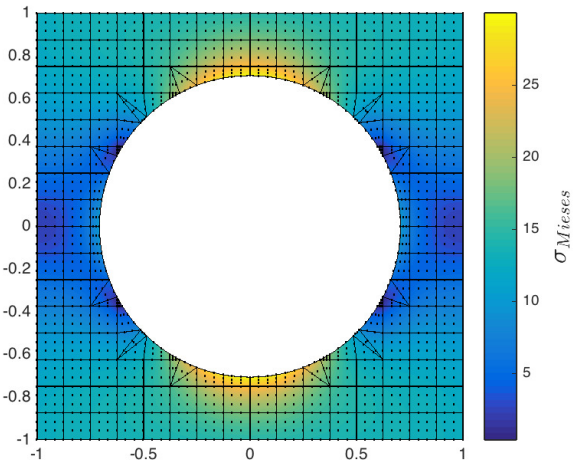

Figure 12: Distribution of von Mieses Stress based on the mesh plotted in the foreground.

The new method is, however, still in an early stage. Further steps are planned towards adaptivity and the extension to three dimensions.

REFERENCES

[1] T. Fries, S. Omerović, Higher-order accurate integration of implicit geometries. *International Journal for Numerical Methods in Engineering*, 2015. early view.

[2] K. Ho-Le, Finite element mesh generation methods: a review and classification. *Computer-aided design*, **20**(1), 27–38, 1988.

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

[4] R. Mittal, G. Iaccarino, Immersed boundary methods. *Annual Review of Fluid Mechanics*, **37**, 239 – 261, 2005.

[5] J. Parvizian, A. Düster, E. Rank, Finite cell method. *Computational Mechanics*, **41**(1), 121–133, 2007.

[6] D. R. Noble, E. P. Newren, J. B. Lechman, A conformal decomposition finite element method for modeling stationary fluid interface problems. *International Journal for Numerical Methods in Fluids*, **63**(6), 725–742, 2010.

[7] S. Osher, R. P. Fedkiw, Level set methods: An overview and some recent results. *Journal of Computational Physics*, **169**(2), 463–502, 2001.

[8] L. Piegl, W. Tiller, *The NURBS book*. Springer, 1997.

[9] Y. L. Ma, W. Hewitt. Point inversion and projection for NURBS curve: control polygon approach. In *Proceedings of Theory and Practice of Computer Graphics, 2003.*, 113–120. Institute of Electrical & Electronics Engineers (IEEE), 2003.

[10] I. Selimovic, Improved algorithms for the projection of points on NURBS curves and surfaces. *Computer Aided Geometric Design*, **23**(5), 439–445, 2006.

[11] Y.-T. Oh, Y.-J. Kim, J. Lee, M.-S. Kim, G. Elber, Efficient point-projection to freeform curves and surfaces. *Computer Aided Geometric Design*, **29**(5), 242–254, 2012.

[12] X.-D. Chen, J.-H. Yong, G. Wang, J.-C. Paul, G. Xu, Computing the minimum distance between a point and a NURBS curve. *Computer-Aided Design*, **40**(10), 1051–1054, 2008.

[13] B. Szabó, I. Babuška, *Introduction to finite element analysis: Formulation, verification and validation*. John Wiley & Sons, 2011.