International Conference
ENGINEERING MECHANICS 2010
Svratka, Czech Republic, May 10 – 13, 2010

EXPERIENCES WITH THE ISOGEOMETRIC ANALYSIS IN STRUCTURAL MECHANICS

D. Rypl, B. Patzák [1]

Summary: *The aim of this paper is to share with its readers the early experiences of its authors related to the implementation, use, and performance of the isogeometric analysis applied to two-dimensional linearly elastic problems. Isogeometric analysis has been recently introduced as a viable alternative to the standard, polynomial-based finite element analysis. Although the isogeometric analysis exhibits several interesting features and it has been shown that it can outperform the classical finite element method in many aspects, its way to the practical engineering is still at the beginning. Not only there are still many open issues that have to be solved but also rather different methodology has to get into awareness of engineering community currently tied up mostly to the finite element method.*

1. Introduction

The concept of isogeometric analysis (IGA) (Hughes et al., 2005; Bazilevs et al., 2006; Cottrell et al., 2006, 2007, 2009), initially motivated by the gap between the computer aided design (CAD) and the finite element method (FEM), builds upon the concept of isoparametric elements, in which the same shape functions are used to approximate the geometry and the solution on a single finite element. The IGA, as its name suggests, goes one step further as it employs the same functions for the description of the geometry and for the approximation of the solution space on that geometry. This implies that the isogeometric mesh (discretization for computational purposes) of the CAD geometry encapsulates the exact geometry no matter how coarse the mesh actually is. As a consequence, the need to have a separate representation for the original CAD model and another one for the actual computational geometry is completely eliminated. The preprocessing is therefore reduced to the construction (i) of the so-called analysis suitable geometry (ASG) by fixing various ambiguities (gaps and overlaps) and removing inappropriate details (small features) from the initial CAD geometry and (ii) of the computational isogeometric mesh which is just the tessellation of the underlying parametric space of the CAD geometry along each of its directions and has nothing common with the (often) costly finite element mesh generation. Note that the isogeometric mesh can be refined to any level without altering the geometry in any way and without accessing the CAD geometry. This makes the isogeometric analysis perfectly suitable for the application to adaptively solved problems because the missing link between the finite element analysis (FEA) and CAD as well as the interaction with external mesh generation software is completely obviated.

[1] Doc. Dr. Ing. Daniel Rypl, Doc. Dr. Ing. Bořek Patzák, Czech Technical University in Prague, Faculty of Civil Engineering, Department of Mechanics, Thákurova 7, 166 29 Prague 6, tel. +420 224 354 369, e-mail drypl@fsv.cvut.cz

This paper presents how the IGA approach may be implemented into an existing object oriented finite element environment. The concept of the IGA is briefly recalled in Section 2. The actual structure and design of class hierarchy for the IGA is described in Section 3. The performance of the developed implementations is demonstrated on a simple example in Section 4. And finally, the paper ends by summarizing the experiences with the implementation and performance issues of the IGA in Section 5.

2. Isogeometric Analysis

The isogeometric approach has been originally developed for the non-uniform rational B-splines (NURBS)[2] (Rogers, 2000; Piegl and Tiller, 1997; Farin, 1995) patches which are the basic building blocks in most CAD systems and which allow precise representation of a wide class of objects. A NURBS patch is described by a set of control points (topologically forming a regular grid of the dimension corresponding to the spacial dimension of the underlying parametric space of the NURBS patch), their weights, degree of the B-spline basis functions in each direction of the parametric space, and a so-called knot vector represented by a nondecreasing sequence of parametric coordinates for each direction defining the support for individual B-spline basis functions (in other words parameterization) in that particular direction. The data at the control points (for example the coordinates when the geometry is concerned, or the primary unknowns when solution space is handled) are interpolated over the NURBS patch using the shape functions which are defined as weighted normalized tensor product of univariate B-spline basis functions in each of the parametric directions. The univariate B-spline basis functions for a particular parametric direction are defined recursively for degree $p > 0$ as

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t), \tag{1}$$

in which the piecewise constant basis functions of zero degree are defined by

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

and where t_i (for $i = 1, 2, \ldots, n + p + 1$) stands for entries of the knot vector and n denotes the number of control points in the given direction. For details concerning the definition of the B-spline basis functions and their properties the reader is referred to Piegl and Tiller (1997).

An example of a quadratic ($p = 2$) NURBS curve (i.e. one-dimensional NURBS patch) defined by six ($n = 6$) control points and their weights (w_i) and parameterized over the knot vector $\{0, 0, 0, 1, 3, 3, 4, 4, 4\}$ is depicted in Figure 1a. The colors of individual parts of the curve correspond to the individual non-zero knot spans (red: $0 - 1$, green: $1 - 3$, blue: $3 - 4$). The B-spline basis functions used to construct the NURBS shape functions R_i (for $i = 1, 2, \ldots, n$)

$$R_i(t) = \frac{N_i(t) w_i}{\sum_{j=1}^{n} N_j(t) w_j} \tag{3}$$

are shown in Figure 1b over the entire span of the knot vector. The curve interpolates those control points for which the corresponding basis function attains value one (knot value at which

[2] Nowadays, the research is rather focused on the use of T-splines patches that are a powerful generalization of NURBS patches (Sederberg et al., 2003, 2004) capable to overcome many problems related to using pure NURBS representation.

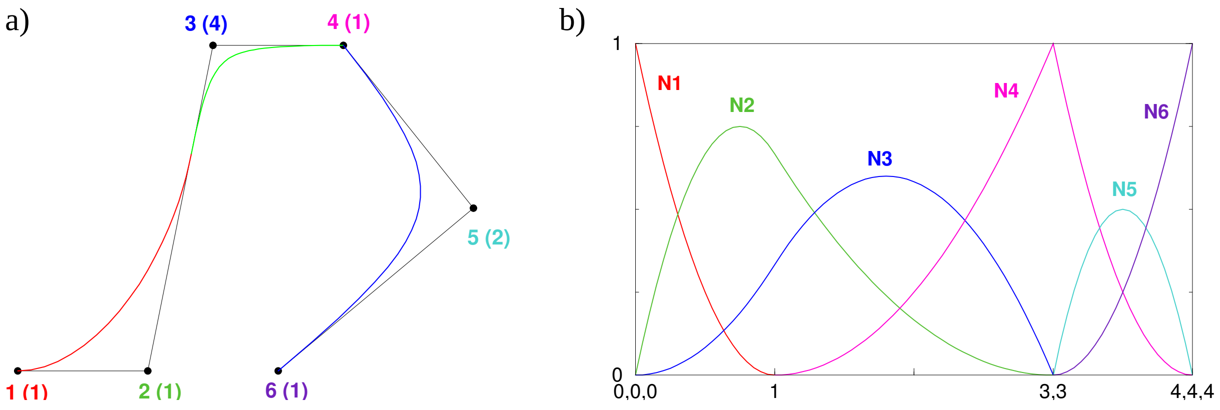

Figure 1: Quadratic NURBS curve: (a) control polygon in black; numbers of individual control points and their weights (in parenthesis) in color corresponding to associated basis function; segments of the curve in red/green/blue corresponding to non-zero knot spans 0-1/1-3/3-4, (b) B-spline basis functions corresponding to individual control points plotted over the entire span of the knot vector $\{0, 0, 0, 1, 3, 3, 4, 4, 4\}$.

this occurs defines the parameter corresponding to that control point), the rest of the control polygon is only approximated. The curve is C^1 continuous everywhere except for the point corresponding to parameter 3 at which the continuity has been weakened by repeating that particular value in the knot vector twice.[3] Note the C^0 continuity of the B-spline basis function N_4 in Figure 1b at parameter 3. The coincidence of the interface between the first (red) and the second (green) knot span on the curve with the intersection of the curve with its control polygon is a rule for quadratic curve only. Note that the red part of the curve (corresponding to the first knot span of size 1) is significantly larger than the green part (corresponding to the second knot span of size 2) despite the fact that the control polygon between control points 1 and 4 is symmetric with respect to the middle of its second segment. This is the consequence of the weight 4 applied at the third control point which results in the attraction of the curve toward the third control point.

The computational isogeometric mesh within the single NURBS patch is formed by partitioning the parametric space to the non-zero knot spans in each direction. The C^0 continuity of the geometry formed by more NURBS patches is attained either in strong sense by using the same NURBS representation along the interface on both patches sharing that interface or by using different NURBS representations along the interface accompanied by appropriate constraints applied to control points, or in the weak sense by using the discontinuous Galerkin formulation with different NURBS representations along the interface. Note that while higher order continuity could be achieved for the geometry representation across adjacent patches, generally only C^0 continuity can be achieved for the solution. In the context of structural mechanics, the NURBS shape functions are complete with respect to affine transformations (Piegl and Tiller, 1997), which means that all rigid body motions and constant strain states are represented exactly. This implies that the convergence to the exact solution is guaranteed.

The IGA has many features in common with the FEM (the basis functions form a partition of unity, they have the compact support, they exhibit affine invariance, numerical integration is employed, Neumann boundary conditions are satisfied naturally etc.) but there are some more

[3] Generally, multiplicity $k \leq p$ of a particular inner knot decreases the continuity at that knot to C^{p-k}.

Table 1: Differences between the IGA and FEA.

Feature	**Finite element analysis**	**Isogeometric analysis**
geometry	not available	represented by control points
mesh	defined by nodal points	defined by knot spans
mesh	approximates geometry	represents exact geometry
solution	defined by nodal variables	defined by control variables
basis	formed by polynomials	formed by NURBS functions
basis	interpolates nodal points and variables	does not generally interpolate control points and variables
basis	satisfies Kronecker delta property	does not generally satisfy Kronecker delta property
basis support	over patch of elements sharing a common node	over a rectangular array of knot spans size of which depends on continuity of the basis
Dirichlet BC	straightforward	approximated within NURBS space

or less significant differences which are summarized in Table 1. Note, that in the traditional FEM, the individual nodes are part of the computational domain, and corresponding degrees of freedom (DOFs) have the direct physical meaning (e.g. displacement in particular direction at the node), which is the direct consequence of Kronecker delta property of the finite element shape functions. Note that in the context of the IGA, the control points of NURBS patches are generally not part of the physical computational domain. This results in lost of clear physical meaning of individual DOFs. Moreover, the physical meaning of DOF in the IGA is made further intricate by the fact that generally not only a single shape function attains non-zero value for the particular node (i.e. physical unknown at such a point has to be evaluated as sum of contributions from nodes with non-zero shape functions at that point). There exist analogues of h-, p-, and hp-refinement strategies in the IGA based on knot insertion and degree elevation algorithms. However, the isogeometric concept offers also a higher order methodology, known as k-refinement (Hughes et al. (2005)), which has no analogue in the standard FEA and which is based on the fact that knot insertion and degree elevation algorithms do not commute. Using the k-refinement, it is possible to increase the continuity across element boundaries (within a single NURBS patch) while limiting the growth of the number of control variables. The various refinement strategies available in the IGA are nicely summarized in Cottrell et al. (2007) using the hpk-space.

3. Design and Implementation of IGA Concept

One of the important issues related to a faster expansion of IGA into research as well as engineering community is the implementation of the isogeometric concept without the necessity to start the coding from scratch. In this section, an approach based on the integration of the IGA

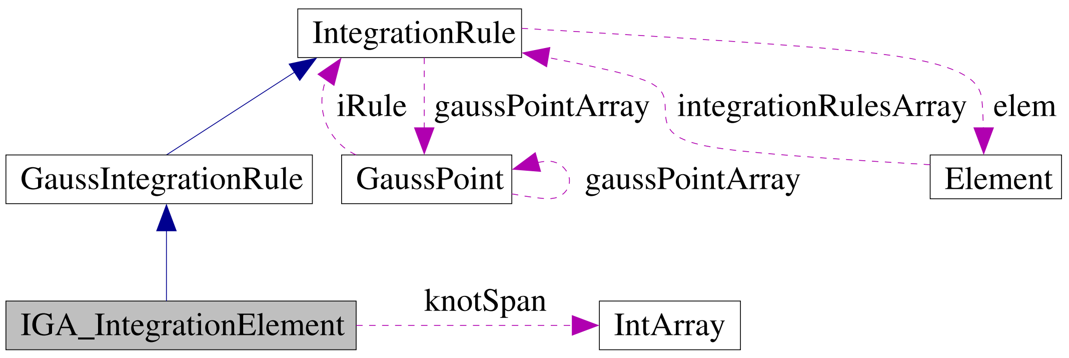

Figure 2: Collaboration diagram for *IGA_IntegrationElement* class.

within an object oriented finite element environment OOFEM (Patzák, 2010) is presented. The emphasis is given on proper design of the hierarchy of classes, their attributes and methods in order to reuse most of the existing functionality of the finite element code while still preserving object oriented features, such as modularity, extensibility, maintainability, and robustness, of the whole environment.

As has been already mentioned, within the IGA terminology the individual isogeometric elements on a particular patch are formed by rectangular parts of the underlying parametric space of the patch corresponding to the non-zero knot spans in each parametric direction. However such a concept is generally not in correspondence with the arrangement in standard finite element codes. In the proposed approach, the whole patch is treated as a single computational element. This is quite natural because in a general case, DOFs at all control points of a single patch may interact with each other (analogy to FEA where DOFs at all nodes of the single element interact with each other). Thus in terms of the object oriented terminology, each patch is represented by instance of class derived from the *IGAElement* class which is in turn derived from base *Element* class which is the parent class for finite elements maintaining list of nodes, boundary conditions (loads), integration rules, keeping links to its interpolation and associated material model and providing general services for accessing its attributes, methods for giving back its code numbers, and abstract services for the evaluation of characteristic components (e.g. stiffness matrix, load vector, etc.).

One of the fundamental issues, that has to be addressed at the computational element level, is the integration. This is important especially for the (very common) case when the support of basis functions is limited only to several consecutive knot spans. In such a case it is highly desirable to limit the integration only to relevant knot spans where the basis functions attain non-zero value. This is achieved by the application of the concept of multiple integration rules per element (analogy to the selective integration in the FEA). In this approach, an integration rule is setuped for each element (thereafter called integration element) of the isogeometric mesh (on the computational element) corresponding to non-zero knot spans in each parametric direction. These integration rules are represented by instances of *IGA_IntegrationElement* class (see Figure 2), which is derived from class *GaussIntegrationRule*, which is in turn inherited from base *IntegrationRule* class, grouping together individual integration points of a particular integration rule and providing general services for their setup. The reason for creating a new class (*IGA_IntegrationElement*) is to introduce the new attribute *knotSpan*, in which the corresponding knot spans are stored and which are used in the interpolation class to evaluate non-zero

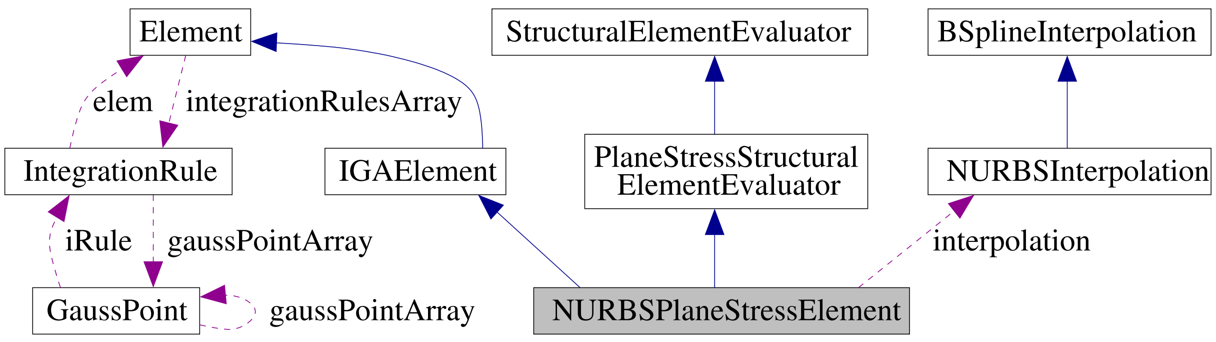

Figure 3: Collaboration diagram for *NURBSPlaneStressElement* class.

basis functions and their derivatives at a given integration point. Note that the knot spans can generally be determined for each integration point on the fly whenever it is needed, but this may introduce some overhead especially for high order integration schemes over patches with large (in terms of number of entries) knot vectors. The integration over the computational element then consists of an outer loop over individual integration elements and an inner loop over the individual integration points setuped and stored on the integration element.

Since the NURBS patch with control points described by coordinates x, y, z and weight w can be represented as the projection of non-rational B-spline patch with control points described by homogeneous coordinates xw, yw, zw, w, it is natural to implement the interpolation within the isogeometric concept for B-splines at first and then to extend it to NURBS. The B-spline interpolation on IGA element is encapsulated into *BSplineInterpolation* class derived from base *FEInterpolation* class which defines the abstract interface in terms of services that evaluate shape functions, their derivatives, jacobian matrix, etc. at given point. Each element (no matter whether it is a standard finite element or IGA element) has to set up its interpolation and to provide access to it. This is enforced by general *Element* interface which requires to define the method for accessing element interpolation. The abstract *FEInterpolation* class interface is essential, as it allows to implement problem specific element methods (for example evaluation of the strain-displacement matrix) already at the top level. The attributes of *BSplineInterpolation*

Table 2: Symbolic code for the evaluation of the stiffness matrix (keyword “this” indicates that the called method is provided by the class itself).

```
StructuralElementEvaluator::computeStiffnessMatrix() {
   loop over all integration rules of the element {
      loop over all Gauss points of the IntegrationRule {
         B = this->computeStrainDisplacementMatrix(gp);
         D = this->computeConstitutiveMatrix(gp);
         dV = this->computeVolumeAround(gp);
         stiffnessMatrix->add(product of BˆT_D_B_dV);
      }
   }
}
```

Table 3: Symbolic code for the evaluation of the strain-displacement matrix.

```
PlaneStressStructuralElementEvaluator::
computeStrainDisplacementMatrix(IntegrationPoint gp) {
   FEInterpolation interp = gp->giveElement()->giveInterpolation();
   interp->evalShapeFunctDerivatives (der, gp);

   answer.resize(3, nnode*2);             // 2 DOFs per each node
   answer.zero();

   for i=1:nnode{
      // normal strain in x direction
      answer.at(1, i*2-1) = der.at(i, 1);   // dN(i)/dx
      // normal strain in y direction
      answer.at(2, i*2)   = der.at(i, 2);   // dN(i)/dy
      // shear strain
      answer.at(3, i*2-1) = der.at(i, 2);   // dN(i)/dy
      answer.at(3, i*2)   = der.at(i, 1);   // dN(i)/dx
   }
}
```

class describe the individual components of the B-spline interpolation, namely the knot vector, degree and number of control points in each parametric direction. The efficient implementation of *BSplineInterpolation* class should profit from the locality of individual interpolation functions which have limited support over several consecutive knot spans. Therefore methods declared by *FEInterpolation* class evaluating values of interpolation functions or their derivatives return the values only for those functions that are nonzero in actual knot span. This enables to compute characteristic element contributions on a knot span basis (via integration elements) efficiently. For each integration element, the contributions are computed only for generally nonzero shape functions and then they are localized into the computational element contribution. The mask of nonzero shape functions for individual knot spans can be evaluated using *giveKnotBasisFuncMask* service declared by *FEInterpolation* and really provided by *BSplineInterpolation*. Once the B-spline interpolation is available, the NURBS interpolation is established in terms of a new class *NURBSInterpolation* inherited from *BSplineInterpolation* class. Class *NURBSInterpolation* overloads only few methods of its parent *BSplineInterpolation* class namely those which require the projection (conversion from the space of homogeneous coordinates to the space of real coordinates), such as the evaluation of the shape functions, their derivatives and jacobian matrix. Note that no new attributes need to be introduced to *NURBSInterpolation* class because the weights needed for the projection are part of the geometry of the NURBS patch.

Specific IGA computational elements are derived from the *IGAElement* class which delivers the generic element functionality (via the inheritance from the base *Element* class) and from one or more classes implementing problem-specific functionality (see Figure 3). The purpose of *IGAElement* class is to provide general method for the initialization of integration rules on individual integration elements (represented by *IGA_IntegrationElement* class). In the present approach, *StructuralElementEvaluator* class is an abstract base class that defines the interface for structural analysis which includes methods for the evaluation of mass and stiffness matrices,

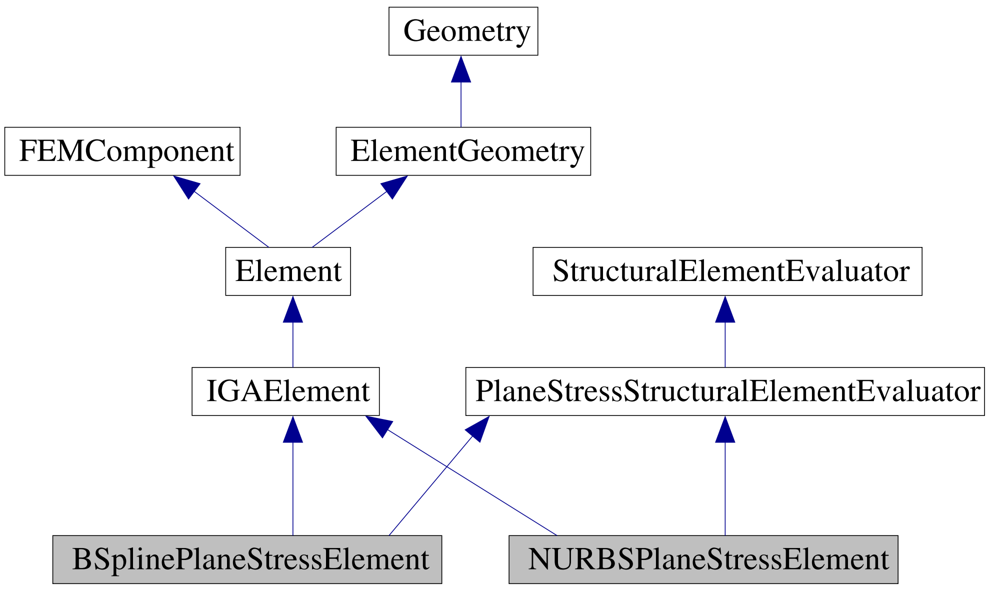
Geometry
FEMComponent
ElementGeometry
Element
StructuralElementEvaluator
IGAElement
PlaneStressStructuralElementEvaluator
BSplinePlaneStressElement
NURBSPlaneStressElement

Figure 4: Inheritance diagram for *BSplinePlaneStressElement* and *NURBSPlaneStressElement* classes.

load vectors, etc. Some of the methods are already implemented at this level, such as stiffness matrix evaluation, based on declared abstract services (evaluation of strain-displacement matrix, etc.), which have to be implemented by derived classes. The example of the evaluation of the element stiffness matrix, which can be used by both classical and isogeometric computational elements, is presented in Table 2 using symbolic code. In the structural analysis context, classes derived from *StructuralElementEvaluator* implement desired functionality for specific types of structural analyzes (plane-stress, plane-strain, full 3D, etc). Provided that the computational element defines its interpolation (and this is enforced by general *Element* interface) it is possible to evaluate remaining abstract methods from *StructuralElementEvaluator* interface (without the need to implement them on the derived specific elements) as it is illustrated in Table 3 on the example of the evaluation of the strain-displacement matrix for the case of plane stress analysis. This is the direct consequence of the design based on decoupled representation of element interpolation and problem specific evaluators, which enables to define problem specific methods only once for various elements with different geometry and interpolation and allows straightforward implementation of elements for multi-physics simulations. Thus, when a new computational element is to be defined, it only has to create its own interpolation and it should be derived from *Element* class (in case of classical finite element) or *IGAElement* class (in case of isogeometric element), which delivers the general basic element functionality and initialization methods, and from one or more evaluators, implementing analysis-specific functionality (see Figure 4).

4. Numerical Example

The functionality of the developed implementation and the performance of the IGA has been verified on a simple two-dimensional example of a mechanical part in the plane stress regime. Geometry with dimensions, boundary conditions, and relevant parameters of the problem is shown in Figure 5a. The computational domain is exactly modelled by a single(!) NURBS

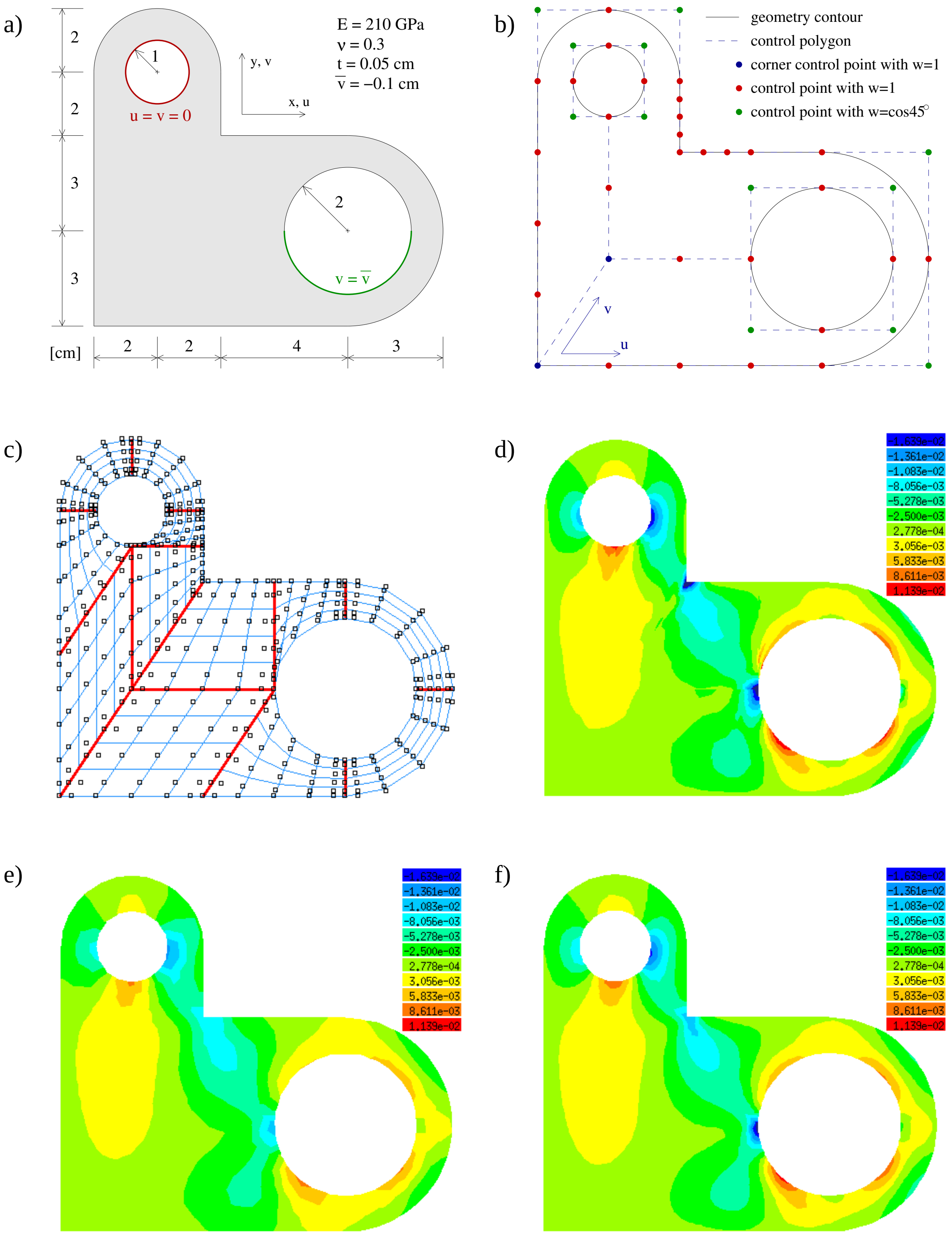

Figure 5: Mechanical part: (a) geometry, boundary conditions, and parameters, (b) initial representation by single NURBS patch, (c) refined representation by single NURBS patch, (d) IGA - contour plot of shear strain on refined discretization, (e) FEA - contour plot of shear strain on coarse mesh, (f) FEA - contour plot of shear strain on fine mesh.

patch quadratic in u-direction and linear in v-direction. The position of individual control points of the patch (together 25 $\times$ 2), their weights and orientation of parametric directions are indicated in Figure 5b. The knot vector describing the parameterization of the patch in u-direction is defined as $\{0, 0, 0, 1, 1, 2, 2, \ldots, 10, 10, 11, 11, 12, 12, 12\}$, the knot vector for v-direction as $\{0, 0, 1, 1\}$. Multiplicity 2 of all inner knots of knot vector in u-directions reveals that the patch is composed in u-direction of 12 individual subpatches with C^0 continuity at the joints. From Figure 5b it is also apparent that the NURBS patch is connected to itself along the 3 legs of the control polygon which are emanating from the corner control point inside of the domain. Again, these connections are of C^0 continuity only. The discretization resulting from this initial representation of the patch is too coarse, however. It has been therefore refined using the concept of k-refinement. The degree of the patch has been firstly elevated by one in each of the parametric directions and then the knot space was uniformly refined to have 48 non-zero knot spans in u-direction and 4 non-zero knot spans in v-direction. This has resulted in a new representation of the NURBS patch with cubic degree and 73 control points in u-direction and quadratic degree and 6 control points in v-direction. The corresponding isogeometric mesh containing 192 integration elements and 419 independent control points[4] is displayed in Figure 5c, in which the red lines indicate the C^0 continuity joints (inside the patch and where the patch is connected with itself). The contour plot of shear deformation computed on the refined discretization is shown in Figure 5d. The deformation is drawn over individual integration elements by subdividing each into 16 subelements (four in each direction) and by using bilinear interpolation on each of them. Note that the figure displays raw data without any postprocessing (smoothing). This is apparent from discontinuities (observable by detailed inspection) of the plotted field along the joints of C^0 continuity (marked in Figure 5c).

A comparative analysis was performed by the standard FEM using linear isoparametric triangular elements. Two uniform discretizations have been considered - a coarse one with 426 nodes and 725 elements (having similar number of DOFs as the discretization used for the IGA) and a fine one with 1714 nodes and 4882 elements (taken as the reference discretization). The profiles of the shear deformation computed by the FEA are drawn in Figures 5e and 5f for the coarse and fine discretization, respectively. Note that these figures display the smoothed values of the shear strain.

The examination of contour plots in Figure 5, reveals that the response obtained by the IGA tends to capture slightly better (despite the discontinuities along the joints) the profile (including the concentrations) of the shear strain, compared to the FEA on the coarse mesh having similar number of DOFs, which is the consequence of the 2nd (in u-direction) and 1st (in v-direction) order continuity of the basis functions inside of individual subpatches compared to the zero order continuity between the finite elements. This effect would be even more pronounced if the degree of the patch interpolation would be further raised.

5. Conclusions

The present work describes an implementation of the IGA concept into an existing object oriented finite element code (Patzák, 2010). The emphasis has been given on the proper design of the hierarchy of classes, their attributes and methods in order to reuse most of the functionality of the finite element code while still preserving modularity and easy extensibility of the whole environment. The experience reveals that the amount of the added code is rather limited (mostly

[4] Note that control points along the legs of the control polygon where the surface is connected to itself are coupled.

related to handling B-spline and NURBS basis functions). Note, however, that this conclusion is tightly adherent to the adopted OOFEM package and may not be generally applicable to other software. It is also worth to mention that the less transparent physical meaning of primary variables in the IGA may complicate debugging of the code.

From the point of view of utilization and performance of the IGA, the analyst is facing several challenges. The first one is related to the order of numerical quadrature that has to be applied to integrate rational terms possibly of high degree. While too low order causes underintegration and consequently significant degradation of the obtained solution, too high order is computationally inefficient. There is now initiated a new research effort (Hughes et al., 2010) focusing on the development of integration schemes that would be more suitable for application within the IGA compared to the traditionally used Gaussian numerical integration. Another issue is related to the enrichment of the discretization in the IGA. While the degree elevation in the IGA is quite similar to the p-refinement in the FEA (with the only difference that with increasing degree also the support of the basis functions increases), the knot refinement in the IGA is far less intuitive than the h-refinement in the FEA. The problem is that the knots are inserted in the parametric space and that the position of control points is related to physical computational domain via the mapping (in other words, control point may be far from the actual location on the real geometry that is mostly influenced by it). Thus without a reasonable knowledge of the mapping between the parametric space and the real space of the patch it may easily happen that the refinement is fairly not as efficient as expected. Also application of boundary conditions in the IGA may be more problematic compared to classical FEM. Homogeneous or constant Dirichlet boundary condition can be easily prescribed either for the whole side of the patch or for its part independent from the rest of that side. Application of variable Dirichlet boundary condition is generally only approximate within the available NURBS space. For the application of point load or point support it is necessary to solve the point inversion problem (to find parametric coordinates corresponding to that point), which cannot be done in a closed form for degree larger than 4 and which may be non-trivial even for cubic or quartic degree. The great advantage of the IGA analysis over the FEA is the simplicity with which higher continuity of the solution inside of the patch (it is just accomplished by elevating the degree of the patch) can be achieved. This is also the reason why the isogeometric solution may (and it usually does) outperform the finite element solution for the same number of unknowns. On the other hand, with increasing degree the computational costs of the IGA are also increasing. This is due to the more demanding evaluation of basis functions and their derivatives, due to the need to enrich the integration scheme and, if an elimination based equation solver is adopted, also due to the larger bandwidth of the the system of equations, which is the consequence of the increase of the support of basis functions over more knot spans. However, the elevated degree does not improve the continuity of the solution of the IGA between patches (and also inside the patch along joints of independent subpatches), thus there is still need to postprocess the solution (at least in the problematic regions) to get smooth contours of quantities depending on the gradient of the primary unknowns. Moreover, higher continuity of the solution may propagate undesirable phenomena (for example due to singularities) deeper in the computational domain (Cottrell et al., 2007). In such a case, it is desirable to decrease the degree locally. If the degree has been previously raised, then it is no problem to decrease it again. In other cases, the degree reduction could lead to change of geometry (of course within a given tolerance, otherwise the degree reduction is rejected). Another nice feature of the IGA (however significant mostly for CFD) is its capability to reproduce sharp fronts in the solution without the occurrence of undesirable oscillations known as Gibbs phenomenon (Hughes et al., 2005).

Clearly, despite few not so favourable aspects of the IGA mentioned above, the IGA is definitely worth consideration as a viable alternative to the FEM because its gains, especially the elimination of the finite element mesh generation and the fact that the exact geometry is handled no matter how coarse the discretization is, are significant.

6. Acknowledgment

This work was supported by the Grant Agency of the Czech Republic - Project No. 103/09/2009. Its financial assistance is gratefully acknowledged.

7. References

Hughes T.J.R. & Cottrell J.A. & Bazilevs Y. 2005: Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement, *Computer Methods in Applied Mechanics and Engineering* 194, 4135–4195.

Bazilevs Y. & Beirao de Veiga L. & Cottrell J.A. & Hughes T.J.R. & Sangalli G. 2006: Isogeometric Analysis: Approximation, Stability and Error Estimates for h-refined Meshes, *Mathematical Models and Methods in Applied Sciences* 16, 1031–1090.

Cottrell J.A. & Reali A. & Bazilevs Y. & Hughes T.J.R. 2006: Isogeometric Analysis of Structural Vibrations, *Computer Methods in Applied Mechanics and Engineering*, 195, 5257–5296.

Cottrell J.A. & Hughes T.J.R. & Reali A. 2007: Studies of Refinement and Continuity in Isogeometric Structural Analysis, *Computer Methods in Applied Mechanics and Engineering*, 196, 4160–4183.

Cottrell J.A. & Hughes T.J.R. & Bazilevs Y. 2009: *Isogeometric Analysis: Toward Integration of CAD and FEA*, John Wiley & Sons.

Hughes T.J.R. & Reali A. & Sangalli G. 2010: Efficient Quadrature for NURBS-based Isogeometric Analysis, *Computer Methods in Applied Mechanics and Engineering* 199, 301–313.

Sederberg T.W. & Zheng J. & Bakenov A. & Nasri A. 2003: T-splines and T-NURCCs, *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3), 477–484.

Sederberg T.W. & Gardon D. & Finnigan G. & North N. & Zheng J. & Lyche T. 2004: T-spline Simplification and Local Refinement, *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3), 276–283.

Rogers D.F. 2000: *An Introduction to NURBS: With Historical Perspective*, Morgan Kaufmann.

Piegl L. & Tiller W. 1997: *The NURBS Book*, Springer-Verlag.

Farin G.E. 1995: *NURBS Curves and Surfaces: From Projective Geometry to Practical Use*, A.K. Peters.

Patzák B. 2010: *OOFEM project home page*, http://www.oofem.org.