Object Oriented Finite Element Analysis for Structural Optimization using p-Elements

Matthias Baitsch, University of Bochum (matthias.baitsch@ruhr-uni-bochum.de)
Dietrich Hartmann, University of Bochum (hartus@inf.bi.rub.de)

Summary

The optimization of continuous structures requires careful attention to discretization errors. Compared to ordinary low order formulation (h-elements) in conjunction with an adaptive mesh refinement in each optimization step, the use of high order finite elements (so called p-elements) has several advantages. However, compared to the h-method a higher order finite element analysis program poses higher demands from a software engineering point of view. In this article the basics of an object oriented higher order finite element system especially tailored to the use in structural optimization is presented. Besides the design of the system, aspects related to the employed implementation language Java are discussed.

1 Introduction

Structural optimization is a tool which supports the engineer in designing good structural systems. Hereby, the design task is formulated as a mathematical optimization problem by means of an optimization model which represents a virtual structure. Often the scalar valued continuous mathematical optimization problem

$$\min_{\mathbf{x}} f_0(\mathbf{x}) \quad \text{with} \quad \mathbf{x} \in S \subseteq \mathbb{R}^n \tag{4}$$

where $\mathbf{x}$ is the n-dimensional design vector and f_0 the objective function is employed. The set of feasible designs S is defined by

$$S = \{\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n,\ f_i(\mathbf{x}) \le 0,\ i = 1, \dots, m\} \tag{5}$$

where the m functions $f_1, \dots, f_m$ are called constraint functions and reflect the requirements the structural design has to fulfill. The problem (4) is then iteratively solved by an adequate numerical optimization technique. Supposing an adequate optimization model is used, the solution $\mathbf{x}^\star$ of (4) also represents a solution of the original design task (Figure 1).

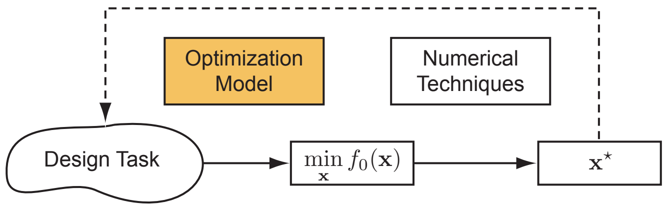
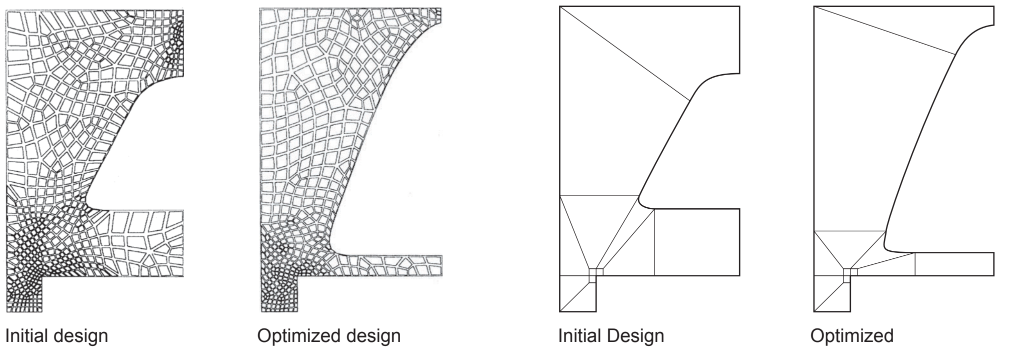


Figure 1: Structural optimization

The core of the optimization model is a finite element model of the system used to predict the behavior of the structural design. This requires the structural analysis to yield reliable results for each possible design vector. When considering – as in this paper – continuous structures, especially discretization errors must be thoroughly controlled. Otherwise, it is likely that the optimization algorithm systematically exploits defects of the model instead of generating a good design with respect to the original design task.

First attempts which used finite element nodes directly as optimization variables (Zienkiewicz and Campbell 1973) or later parametrically meshed design elements (Bletzinger 1990) proved not to be sufficient for complex structures. Nowadays, mainly two approaches are in use: i) A design model which is coupled to the optimization variables describes the structure to be analyzed. An mesh generator automatically generates a discretization with low order elements (h-elements) according to the design model. Based upon an error estimate the mesh is then subsequently adapted until a prescribed error criterion is fulfilled. In general, this procedure has to be repeated in each iteration step of the optimization (see Figure 2(a) from (Wieghardt 1995)). ii) The structure is initially discretized with a coarse mesh consisting of high order elements (p-Elements). Optimization variables are directly linked to entities of the finite element model. Because p-elements are robust with respect to distorted element geometries, the initial discretization does not have not to be adapted during optimization. In order to stay within some reasonable error bounds, the polynomial degree can be increased during optimization (see Figure 2(b) from (Klein 2000)).



(a) Adaptive discretization with h-elements (b) Discretization with p-elements kept during optimization

Figure 2: Approaches for shape optimization of continuous structures

The advantages of the p-method for structural optimization are at hand. There is no need for the time consuming procedure of h-adaptivity. Even if a single p-analysis took longer, the need for repeated mesh generation - analysis cycles in h-adaptivity outweighs this disadvantage. Moreover, the amount of complexity introduced to the software system by the optimization is much smaller. Since the techniques of geometry description employed in the p-method are sufficient for optimization purposes no additional design model and the associated hassles (error estimation, adaptivity and automatic transfer of boundary conditions) has to be introduced.

Within this contribution, an object oriented finite element analysis system is presented. Since the main purpose of the software is structural optimization, a simple yet efficient description of geometry is of high interest. Therefore, Non Uniform Rational B-Splines (NURBS) are used to describe the model by few but substantial key points. From a software engineering point of view, the p-method is more complicated than the h-version of finite elements. The presented software uses object oriented paradigms to model the mathematical concept of basis functions as well as concepts directly related to the finite element method (Node, Edge, Element, . . .). Java as an implementation language has many advantages but execution speed is considered as a major drawback for numerical intense applications. Therefore, a hybrid approach is used, which combines the advantages of Java with the efficiency of native numerical libraries.

2 The *p*-version of the finite element method

In the p-version of the finite element method, convergence to the exact solution of the continuous problem is obtained by increasing the polynomial degree of the basis functions while the mesh is not changed. The presented implementation uses hierarchical basis function according to (Szabó and Babuška 1991).

2.1 One-dimensional hierarchical basis

Other than Lagrangian basis functions (see Figure 3(a)) widely used for low order elements, hierarchical basis functions are constructed such that all lower order basis functions are contained in a higher order basis (see Figure 3(b)).

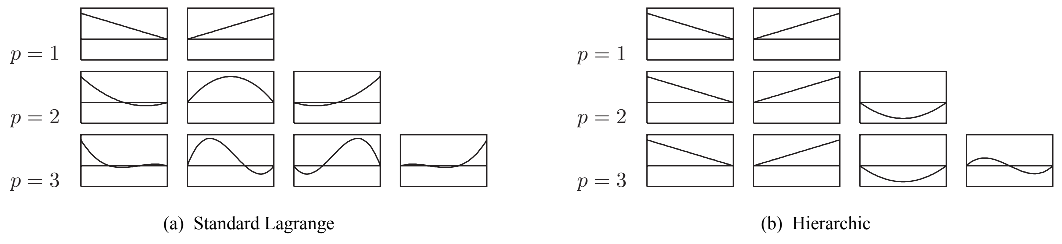


Figure 3: One-dimensional shape functions

Given the Legendre polynomials

$$P_n(r) = \frac{1}{2^n n!}\frac{d^n}{dr^n}(r^2-1)^n, \quad n = 0, 1, \ldots, \tag{6}$$

the one dimensional hierarchic basis functions

$$\begin{aligned} N_1(r) &= 0.5(1-r), & (7)\\ N_2(r) &= 0.5(1+r), & (8)\\ N_i(r) &= \sqrt{\frac{2i-3}{2}}\int_{-1}^{r} P_{i-1}(\xi)d\xi, \quad i = 3, 4, \ldots, p+1, & (9) \end{aligned}$$

are defined on the interval $[-1, 1]$ for any polynomial degree p. The basis functions (9) can be easily be implemented by employing the code generation feature of programs like Maple or Mathematica.

2.2 Two-dimensional hierarchic basis for quadrilaterals

The two-dimensional hierarchic basis functions, defined on $[-1, 1] \times [-1, 1]$, are constructed by multiplying the one-dimensional basis functions (7–9):

$$N_{o(i,j)}(r, s) = N_i(r)N_j(s), \quad (i, j) \in A \subset \mathbb{N}^+ \times \mathbb{N}^+. \tag{10}$$

Hereby, A is the set of index pairs taken into account and $o : A \to 1, 2, \ldots, d$ an indexing function which is used to arrange the individual basis functions in a one dimensional array. The number of elements contained in A is called d.

By choosing the set of index pairs A, different ansatz spaces can be constructed. Currently, only the isotropic tensor product space with

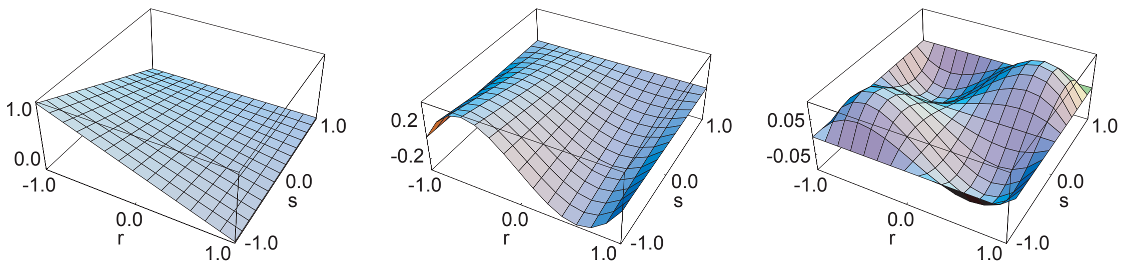
$$A_{itp} = \{1, \ldots, p+1\} \times \{1, \ldots, p+1\} \tag{11}$$

is used. For the extension to anisotropic ansatz spaces, which use different polynomial degrees for each local coordinate direction, the reader is referred to (Düster 2001).

Independently from the choice of an ansatz space, the two-dimensional basis functions can be collected in three groups (Figure 4):

1. The four **nodal modes**, which are be obtained for $\{(i, j) \in A \mid i = 1, 2 \text{ and } j = 1, 2\}$ are identical to the standard bilinear shape functions. The nodal modes have indices 1 to 4 which correspond to the local node indices in Figure 5.

2. **Edge modes** vanish at all edges except one. The corresponding index pairs are contained in $\{(i,j) \in A \mid i \leq 2,\ j \geq 3 \text{ or } i \geq 3,\ j \leq 2\}$.
3. The set of index pairs $\{(i,j) \in A \mid i \geq 3,\ j \geq 3\}$ yields the **internal modes** which vanish at all edges.



(a) Nodal mode (b) Edge mode (c) Internal mode

Figure 4: Modes of hierarchical basis functions

2.3 Blending function method

Because of the large element size typically used in a discretization with p-elements, the element boundaries must exactly match the boundaries of the meshed domain. For this purpose, the blending function method is employed. Here, the geometry of the element is described by means of the four nodes $\mathbf{X}_i, i = 1, \cdots, 4$ and four parametrical curves

$$\mathbf{E}_i(t) = [E_{ix}(t), E_{iy}(t)]^T, \quad t \in [-1,1], \quad i = 1, \cdots, 4 \tag{12}$$

connecting the nodes. The blending function method fits very well in the context of shape optimization, when the edge functions are represented adequately (see Section 3.1).

The mapping from natural coordinates $\mathbf{r} = [r,s]^T$ to physical coordinates $\mathbf{x} = [x,y]^T$ is performed by the function $\mathbf{Q}^e :\ [-1,1] \times [-1,1] \to \mathbb{R}^2$. A graphical representation of this relation shows Figure 5.

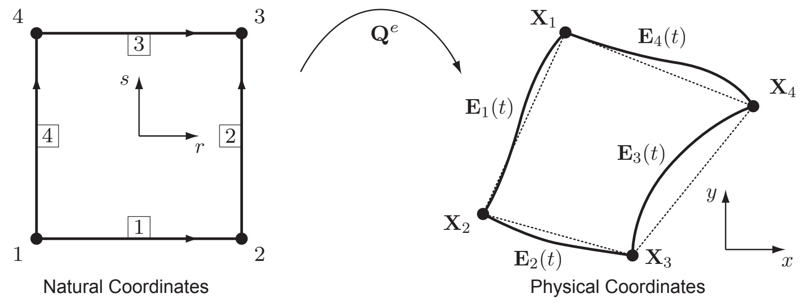


Figure 5: Blending function method for quadrilaterals

By using the previously defined basis functions, the mapping can be written as follows:

$$\mathbf{Q}^e(r,s) = N_1(s)\mathbf{E}_1(r) + N_2(r)\mathbf{E}_2(s) + N_2(s)\mathbf{E}_3(r) + N_1(r)\mathbf{E}_4(s) - \sum_{i=1}^{4} N_i(r,s)\mathbf{X}_i. \tag{13}$$

Knowing the tangents on the edges, the Jacobian of $\mathbf{Q}^e$, needed for the computation of derivatives with respect to global coordinates as well as for the numerical integration of element matrices can easily be evaluated.

3 Geometry description

In the following sections, some employed concepts related to geometrical modeling are presented.

3.1 NURBS

Today, Non Uniform Rational B-Splines (NURBS) are a de facto standard in computer aided geometric design (CAGD). Using NURBS, it is possible to represent free form geometries as well as regular shapes (conics) in a formal uniform manner. Because of the blending function method, NURBS can be elegantly integrated in a p-version finite element system.

The B-Spline basis functions are the basis for the definition of NURBS. Let the vector of parameter knots $\mathbf{t} = [t_0, \ldots, t_m]$ be a non falling sequence of real numbers. Starting from the B-Spline basis functions of degree zero

$$N_{i,0}(t) = \begin{cases} 1 & \text{for } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

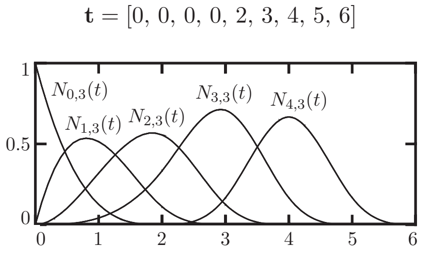
the basis functions of higher degrees

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t), \tag{15}$$

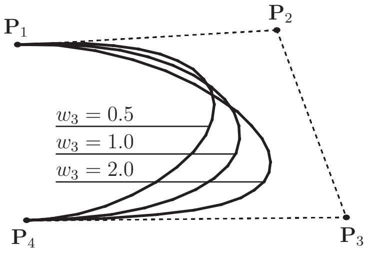
are defined recursively. An example for B-Spline basis functions shows Figure 6(a). A one-dimensional NURBS in $\mathbb{R}^m$ is at first defined in homogenous coordinates by the linear combination

$$\mathbf{r}^w(t) = \left[r_1^w(t), \ldots, r_{m+1}^w(t)\right]^T = \sum_{i=1}^{n} N_{i,p}(t)\mathbf{P}_i^w \tag{16}$$

of the basis functions $N_{i,p}(t)$ with the weighted control nodes $\mathbf{P}_i^w = [w_i\, p_{i,1}, \ldots,\ w_i\, p_{i,m}, w_i]^T$ where $p_{i,j}$ are the Cartesian coordinates and w_i is the weight of the control node.



(a) B-Spline basis functions



(b) Influence of a varying weight

Figure 6: B-Spline basis functions and example of a NURBS

The perspective projection of the curve (16) into the $\mathbb{R}^m$ yields the NURBS

$$\mathbf{r}(t) = \frac{1}{r_{m+1}^w(t)} \left[r_1^w(t), \ldots, r_m^w(t)\right]^T \tag{17}$$

itself. Figure 6(b) shows three instances of a NURBS with varied weight of one control node. For the computation of the tangent vector and the construction of conics using NURBS, the reader is referred to (Piegl and Tiller 1995).

3.2 Segmented edges

Often it is necessary to ensure $C1$ continuity of two edges which belong to adjacent elements at a common node. For optimization purposes, it would be more convenient to use one single parametric curve to describe the shape of two or more edges than to establish an additional relation between control points in order to ensure the required continuity.

The considered situation is illustrated in Figure 7. Let $\mathbf{E}_s(t)$ be a curve which is to be shared amongst n elements and a $\mathbf{t} = [t_0 = -1, t_1, \ldots, t_n = 1]$ be an increasing sequence of parameters at which the edge should be segmented. The curve for edge i of the individual element is obtained by projecting the standard interval $[-1, 1]$ on the corresponding interval $[t_{i-1}, t_i]$. By conveniently employing the one-dimensional basis functions $N_1(t)$ and $N_2(t)$, the relation

$$\mathbf{E}_i(t) = \mathbf{E}_s(t_{i-1}N_1(t) + t_iN_2(t)), \quad i = 1, \ldots, n \tag{18}$$

holds for $t \in [-1, 1]$. The location of the intermediate finite element nodes are hereby implicitly located on the curve.

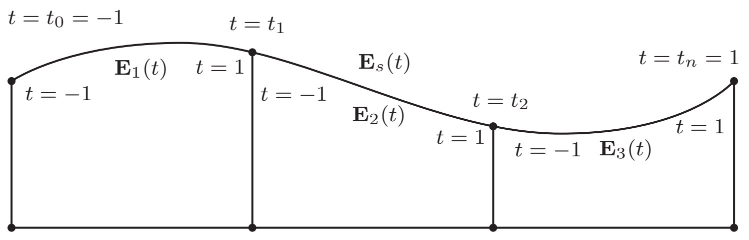


Figure 7: Edge shared amongst three elements

3.3 Non-constant thickness

In various situations, eg. the example in Section 5.1, it may be necessary to analyze two-dimensional continua which have a varying thickness. A linearly varying thickness can be represented by

$$t = t(r, s) = \sum_{i=1}^{4} t_iN_i(r, s) \tag{19}$$

where the t_i are the thicknesses at the four nodes and $N_i(r, s)$ standard bilinear shape functions. This approach has two major disadvantages: first, it may be hard to manage in the case of more than a few elements (thicknesses have to be stored for each element individually) and second, when using p-elements only simple geometries can be adequately described. Therefore, an analytical description

$$t = t(x, y) \tag{20}$$

which depends on global rather than local coordinates is also possible in the presented software.

4 Object oriented software system

In the finite element method, the solution (eg. a displacement field) is element wise approximated by a linear combination of basis functions. This concept is not coupled to the physical background of a particular element formulation. The classes which provide these functionalities are therefore collected in a self contained framework (package fct). The classes needed to establish a finite element model (nodes, elements, ...) are located in the package fe. The individual element classes hereby use of the classes contained in the package fct.

4.1 Modeling of basic mathematical concepts

Functions and functions as a linear combinations of basis functions play a fundamental role in the finite element method. The interfaces and classes which represent these concepts are illustrated in Figure 8.

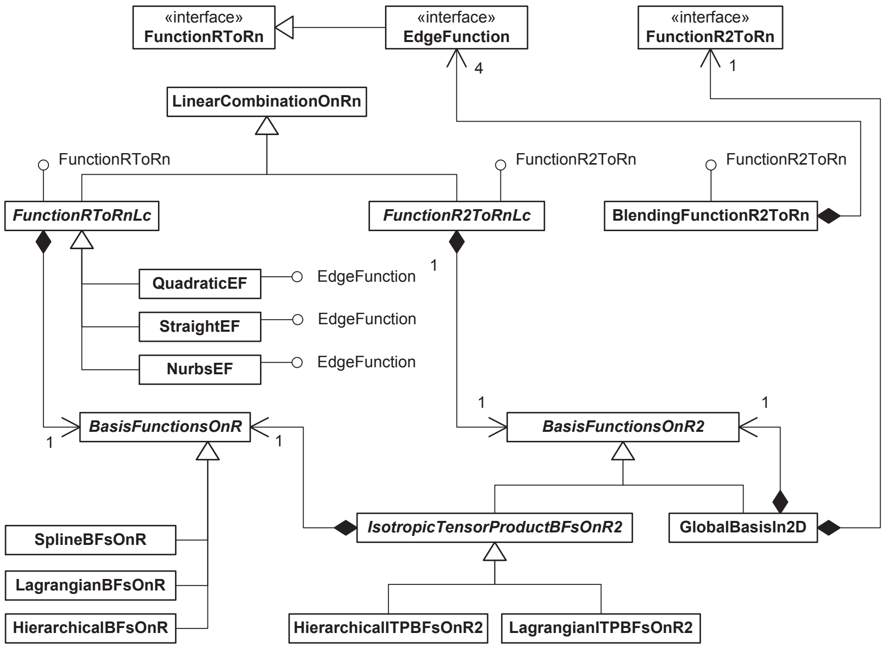


Figure 8: UNL class diagram of the function package

The concept of a function $f : \mathbb{R}^m \to \mathbb{R}^n$ for which function values and derivatives can be computed is formulated in the interfaces FunctionRToRn for $m = 1$ and FunctionR2ToRn for $m = 2$ respectively. A function $f : \mathbb{R}^m \to \mathbb{R}^n$ can be constructed by the linear combination

$$f = \sum_{i=1}^{d} \mathbf{k}_i N_i \tag{21}$$

where $\mathbf{k}_i \in \mathbb{R}^n$ are coefficients, $N_i : \mathbb{R}^m \to \mathbb{R}$ are linearly independent basis functions and d is the dimension of the basis. The abstract class LinearCombinationOnRn serves as base class for such linear combinations. Its main purpose is to store the coefficients $\mathbf{k}_i$.

The linear combination (21) for the case $m = 1$ is represented by the class FunctionRToRnLc. It therefore aggregates an instance of the abstract class BasisFunctionsOnR which has methods to compute function values as well as derivatives for all basis functions. There exist derived classes for Lagrangian, hierarchical and B-Spline basis functions.

The classes FunctionR2ToRnLc and BasisFunctionsOnR2 represent the same concept in two dimension ($m = 2$). Currently, only the isotropic tensor product basis functions are implemented. The corresponding class IsotropicTensorProductBFsOnR2 uses one-dimensional basis functions according to Equation (10). In order to compute derivatives of basis functions with respect to physical coordinates $\mathbf{x}$ defined by $\mathbf{Q}^e$ (Figure 5), the class GlobalBasisIn2D is introduced. Thus, it consists of basis functions, for which these derivatives should be evaluated and a function $\mathbb{R}^2 \to \mathbb{R}^2$ which represents $\mathbf{Q}^e$.

The class BlendingFunctionR2ToRn implements the blending function method as described in Section 2.3. Four instances of the interface EdgeFunction describe the individual edges. Hereby, an edge function is a curve, for which the start point and the end point can be prescribed.

4.2 The finite element package

The classes necessary to establish a finite element model are shown in Figure 9. The organization of the classes is straightforward, thus in the following only some special aspects will be discussed.

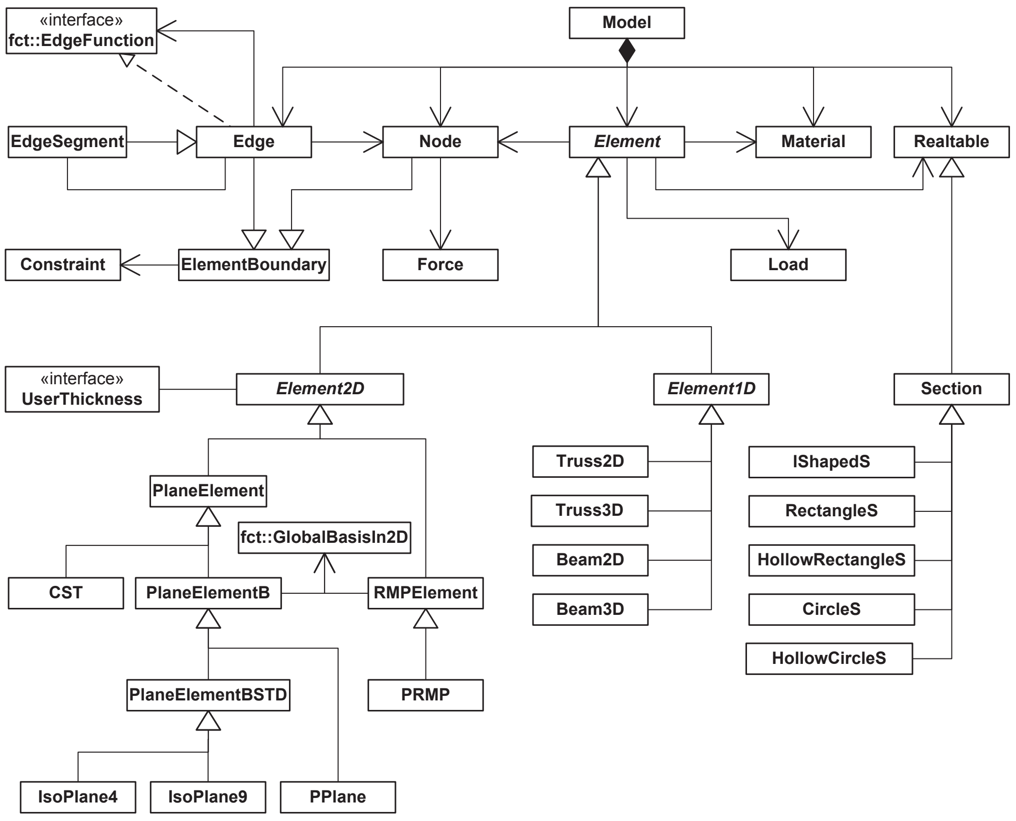


Figure 9: UML class diagram of the finite element package

In the p-method, degrees of freedom are not only related to nodes but also to edges. The functionality needed for the enumeration of degrees of freedom and the handling of constraints is therefore collected in a class ElementBoundary which serves as base class for Node and Edge. Furthermore, Edge implements the interface EdgeFunction (from the package fct, see Section 4.1) such that it can be used for the blending function method. For flexibility, a delegate mechanism is employed.

The class Element2D is the base class for all two-dimensional element formulations. Element thicknesses can either be stored in an instance of Realtable or analytically described by implementing the interface UserThickness. It should be noted that in Java classes can be loaded at runtime. The implementation of user defined thickness function does therefore not involve a compilation of the whole program.

Currently, two two-dimensional element families are implemented: plane elements (base class: PlaneElementB, the constant strain triangle element CST is only listed for completeness) and Reissner-Mindlin plate elements (base class: RMPElement). These two base classes use an instance of fct::GlobalBasisIn2D for the computation of the element stiffness matrix and the element load vector. The concrete element classes consist only of a few lines of code in which the actual set of basis functions is chosen.

4.3 Implementation notes

The above object oriented model is implemented in the programming language Java. Many features of Java as a modern object oriented language allow a more efficient implementation of more reliable software compared to, for instance, C++. Therefore, Java has also been recognized in the world of scientific computing (Boisvert, Moreira, Philippsen, and Pozo 2001). The price to pay is execution efficiency which is, although improved by just in time compilation (JIT) still below that of procedural languages like FORTRAN or C (Nikishkov, Nikishkov, and Savchenko 2003). Also in the near future, Java can not be expected to outperform highly tuned, processor specific numerical libraries provided by CPU-manufacturers (Intel corporation).

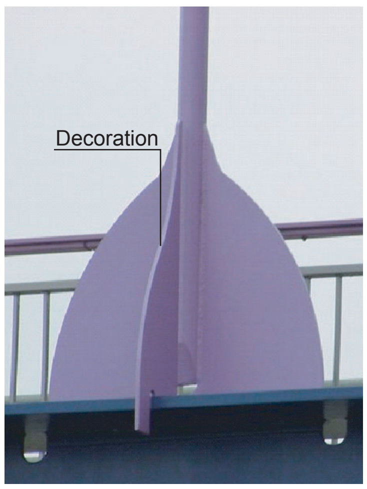
For these reasons a hybrid approach is taken. Java is used for the implementation of the overall system while numerical intense computations are delegated to native routines incorporated through the Java native interface (JNI). Currently, this affects primarily the employed equation solver for banded positive definite matrices taken from LAPACK (Linear Algebra PACKage). The implementation used is contained in the Intel Math Kernel Library (MKL) which is available for Windows and Linux operating systems. For the future, it is planned to use routines from the LAPACK and BLAS (Basic Linear Algebra Subprograms, also contained in MKL) for all linear algebra purposes.

5 Application examples

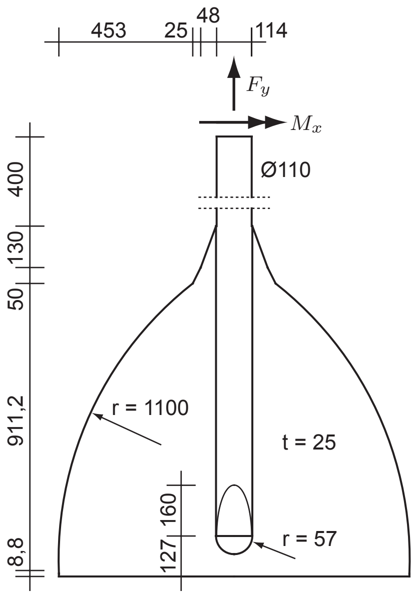
The following examples illustrate that the presented software can be used both for the analysis of complex structures as well as for structural optimization.

5.1 Analysis of a hanger connection

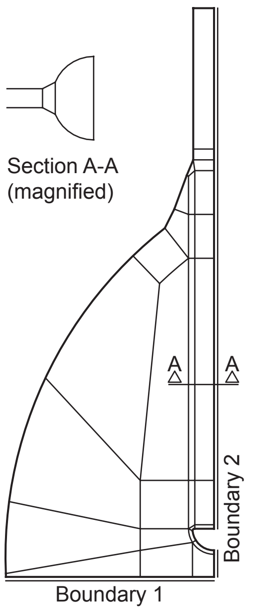
In the first example, a hanger connection which is a part of an arch bridge is analyzed. The considered component (see Figure 10(a)) is a reference example for lifetime-oriented design in the DFG-Collaborative Research Center 389 involves a moderately complex geometry with varying thickness. Dimensions and applied unit loads can be taken from Figure 10(b).



(a) Photograph of the system



(b) Geometry and loading



(c) Discretization with 28 p-elements

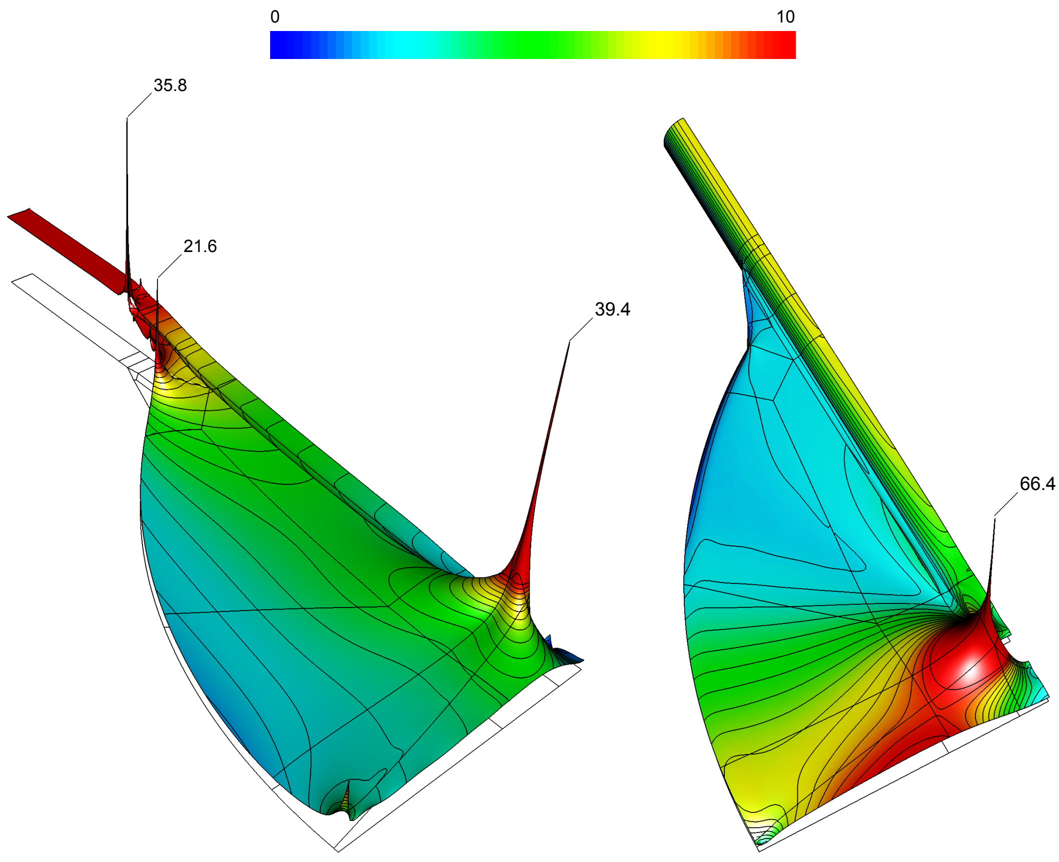
Figure 10: Hanger connection

Since the component as well as the applied loads are symmetric, only one half of the domain is modeled. The employed discretization uses 28 p-elements and is refined at critical points (Figure 10(c)). The considered loading cases together with the employed element types and boundary conditions are listed in Table 1. For the material, linear elastic behavior with Young's modulus $E = 210000$ and Poisson's ration $\nu = 0.3$ is assumed. The varying thicknesses at the rod and the welding are described by two individual instances of the interface UserThickness (see Section 4.2).

Table 1: Loading cases

			Boundary conditions	
Loading case	Applied load	Element type	Boundary 1	Boundary 2
1	$F_y = 100$ kN	Plane	$u_x = u_y = 0$	$u_x = 0$
2	$M_x = 1$ kNm	Bending plate	$u_z = \beta_x = \beta_y = 0$	$\beta_x = 0$

For the analysis of both loading cases a polynomial degree of 9 is used. The resulting number of degrees of freedom is 4790 for the plane problem and 7185 for the plate problem. Figure 5.1 shows the distribution of equivalent stress for both loading cases. It can clearly be seen that the chosen design of the hanger has three critical regions: first at the point where the plate ends, second and less critical at the sharp bend of the plate just below and third and most serious where the rod ends. Regions one and two hereby are not critical for loading case 2.



Loading case 1 ($F_y = 100$ kN)   Loading case 2 ($M_x = 1$ kNm)

Figure 11: Equivalent stress

5.2 Optimization of a Connecting Rod

In the second application example the shape of a connecting rod is optimized. This optimization problem is a widely used test problem (Kimmich 1990; Sienz and Hinton 1997; Klein 2000). The objective in this optimization problem is to minimize the volume with respect to an allowable equivalent stress of $\sigma_{all} = 1\,200\,\mathrm{N/mm}^2$. The initial structure is depicted in Figure 5.2. One of the edges belongs to two elements and connects nodes eight, nine and ten (see Section 3.2). The shape of this edge is described by a NURBS of degree three having five control points. The coupling of control points and nodes to the optimization variables is shown also in the picture.

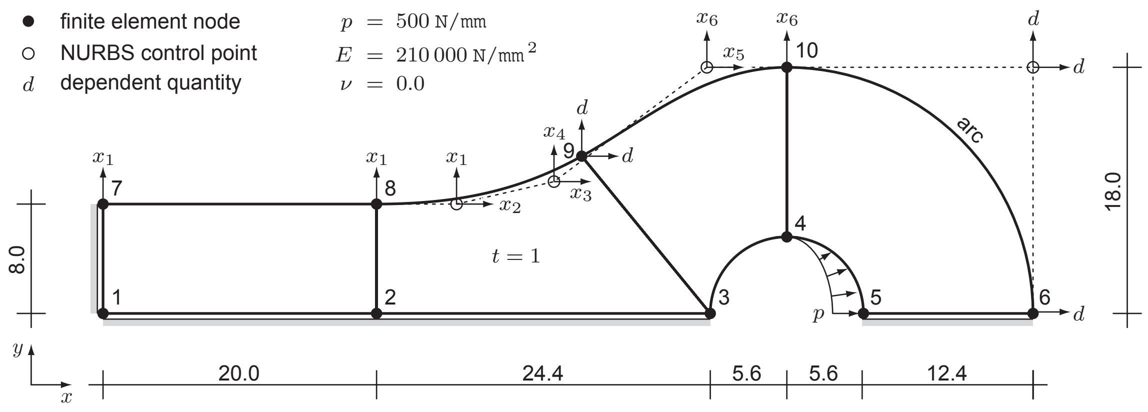
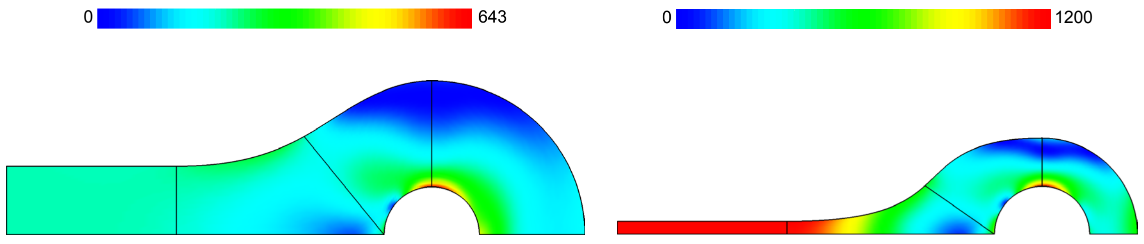


Figure 12: Initial Geometry, Design Variables, Loading and Boundary Conditions (dimensions are [mm])

The optimization is performed by an FSQP-Algorithm (Lawrence, Zhou, and Tits 1997) which is integrated in a CORBA based optimization component providing various optimization methods (Baitsch, Lehner, and Hartmann 1999). During optimization, the initial polynomial degree of $p = 6$ is not changed. The optimized shape (see Figure 5.2) has a volume of $V = 254.3\mathrm{mm}^3$. This value is slightly better than in references above but not directly comparable because this geometry model allows for greater variability.



Initial shape ($V = 734.1\mathrm{mm}^3$) Optimized shape ($V = 254.3\mathrm{mm}^3$)

Figure 13: Distribution of equivalent stress for initial and optimized shape

6 Conclusions

In this contribution an object oriented finite element system has been presented. Because of the use of p-elements in conjunction with NURBS it is especially well suited for the shape optimization of continuous structures. The implementation incorporates the advantages of the programming language Java as well as the high efficiency of native libraries for linear algebra.

7 References

The Intel Math Kernel Library (Intel MKL). `http://www.intel.com/software/products/mkl/overview.htm`.

Baitsch, M., K. Lehner, and D. Hartmann (1999). A CORBA based universal optimization service. In *Proceedings of the 1. ASMO UK/ISMO Conf. on Engineering Design Optimization*, Ilkley, GB, pp. 233–240.

Bletzinger, K.-U. (1990). *Formoptimierung von Flächentragwerken*. Dissertation, Institut für Baustatik, Universität Stuttgart.

Boisvert, R., J. Moreira, M. Philippsen, and R. Pozo (2001). Numerical computing in Java. *Computing in Science and Engineering 3*(2), 18–24.

Düster, A. (2001). *High order finite elements for three-dimensional, thin-walled nonlinear continua*. Dissertation, Technische Universität München.

Kimmich, S. (1990). *Strukturoptimierung und Sensitivitätsanalyse mit finiten Elementen*. Dissertation, Institut für Baustatik, Universität Stuttgart.

Klein, M. (2000). *Strukturoptimierung mit der p-Version der finiten Elemente*. Dissertation, Institut für Konstruktiven Ingenieurbau, Ruhr-Universität Bochum.

Lawrence, C., J. Zhou, and A. Tits (1997). *User's Guide for CFSQP Version 2.5*. University of Maryland.

Nikishkov, G. P., Y. G. Nikishkov, and V. V. Savchenko (2003). Comparison of C and Java performance in finite element computations. *Computers & Structures 81*(24–25), 2401–2408.

Piegl, L. and W. Tiller (1995). *The NURBS book*. Springer, Berlin.

Sienz, J. and E. Hinton (1997). Reliable structural optimization with error estimation, adaptivity and robust sensitivity analysis. *Computers & Structures 64*(1–4), 31–63.

Szabó, B. and I. Babuška (1991). *Finite Element Analysis*. Chichester, England: J. Wiley & Sons.

Wieghardt, K. (1995). *Konzept zur interaktiven Formoptimierung kontinuierlicher Strukturen*. Dissertation, Institut für Konstruktiven Ingenieurbau, Ruhr-Universität Bochum.

Zienkiewicz, O. and J. Campbell (1973). *Shape optimization and sequential linear programming*, Chapter 7. Chichester, England: J. Wiley & Sons.