Czech Technical University in Prague

Faculty of Civil Engineering

Department of Mechanics

Daniel Rypl, Bořek Patzák

From the Finite Element Method toward the Isogeometric Analysis in an Object Oriented Computing Environment

Presentation Outline

- **Motivation**
- **B-spline basis**
- **T-splines = NURBS + PB-splines**
- **Principles of OO design**
- **OOFEM**
- **OO design of IGA module**
- **Numerical example**
- **Summary**

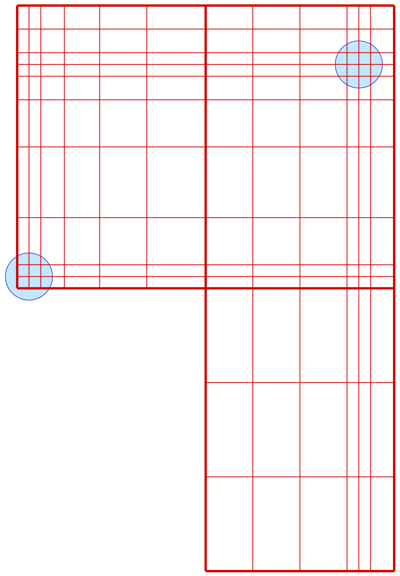
Motivation

Isogeometric Analysis

- recently introduced alternative to the FEM
- employs the same functions for the description of geometry and for the approximation of the solution on that geometry
  - eliminates costly FE mesh generation
  - geometric preprocessing still required
- outperforms classical FEM in various aspects
- still many open issues

  (trimmed geometry, boundary conditions, integration, efficiency issues, implementation, performance . . .

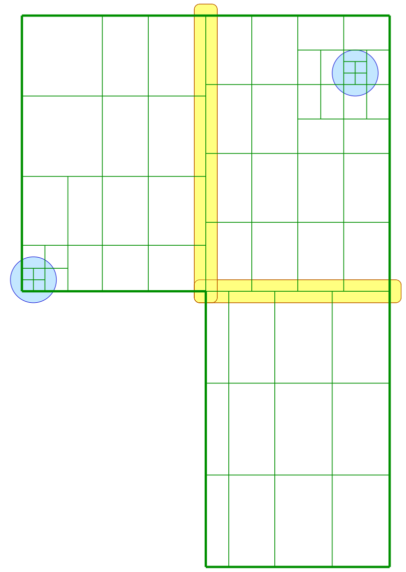
Motivation



- **IGA originally developed for NURBS**
  - **convenient for free-form surface modelling**
  - **exact representation of quadric surfaces**
  - **stable and efficient algorithms available**
  - **present in most CAD systems**

  - **gaps and overlaps cannot be avoided**
  - **trimmed NURBS not handled by IGA**
  - **generally only C^0 continuity across patch boundaries**
  - **tensor product structure of NURBS not efficient for representation of local features and for connection of adjacent surfaces**
  - **most shapes cannot be represented as a single watter-tight NURBS**

Motivation



T-spline Based Isogeometric Analysis

- **generalization of NURBS technology**
  - **inherits geometrical flexibility of NURBS**
  - **allows efficient local refinement**
  - **allows watter-tight merging of adjacent NURBS**
  - **T-splines are forward and backward compatible with NURBS**
  - **trimmed NURBS can be represented as T-spline**

  - **non-straightforward refinement around extraordinary points**
  - **non-trivial representation of solids preserving exactly boundary surface geometry**
  - **limited availability in commercial CAD (Maya, Rhino, SolidWorks)**

Motivation

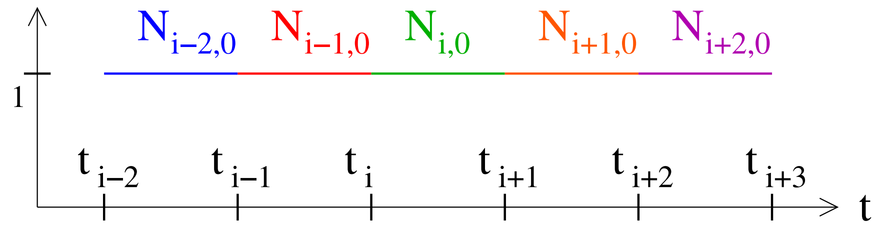
Implementation

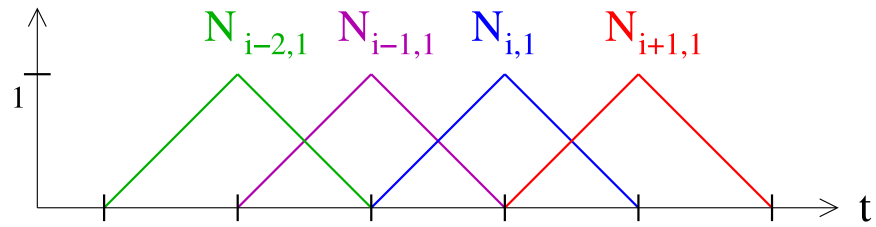
- many similar features between FEM and IGA
- no need to start implementation from scratch
- most of the FE codes can be reused
- object oriented design recognized as very appropriate
  - proved to be a viable concept significantly enhancing modularity, extensibility, maintainability, and robustness of the code without sacrificing its performance
  - supports team work, allows further developments without participation of original authors

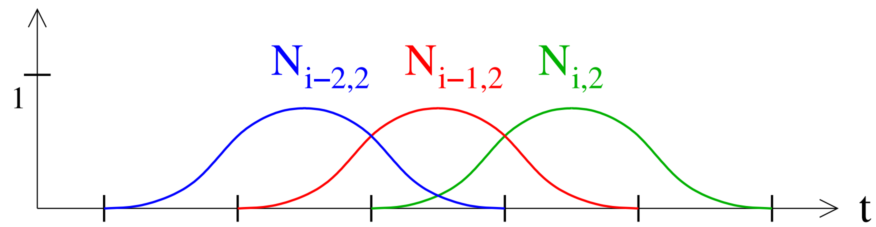
Univariate B-spline basis functions

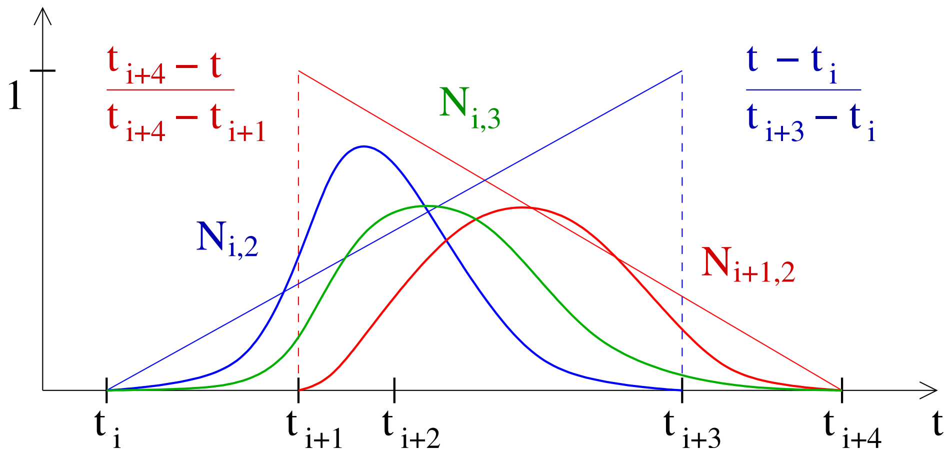
$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t) \qquad \text{for } p > 0$$

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



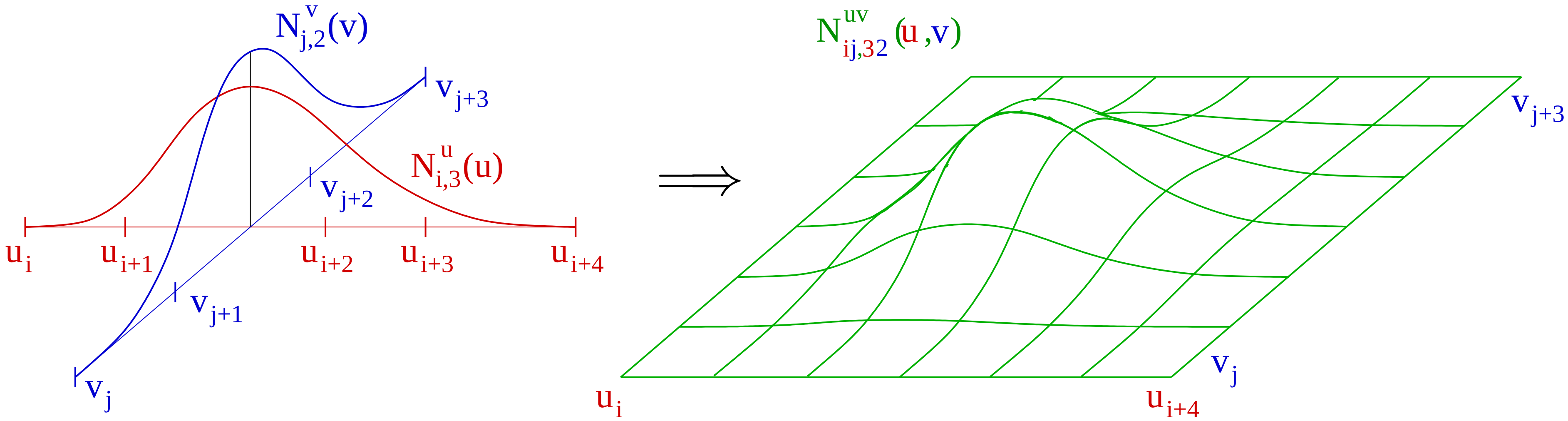






Bivariate B-spline basis functions

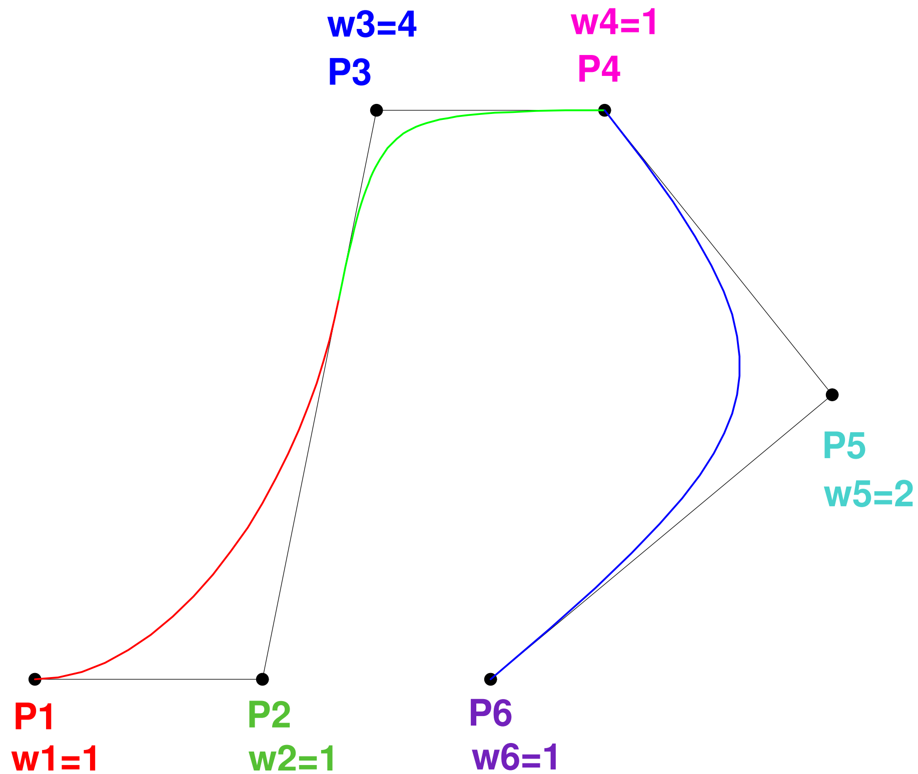
$$N^{uv}_{ij,pq}(u,v) = N^{u}_{i,p}(u)N^{v}_{j,q}(v) = N_k(u,v)$$



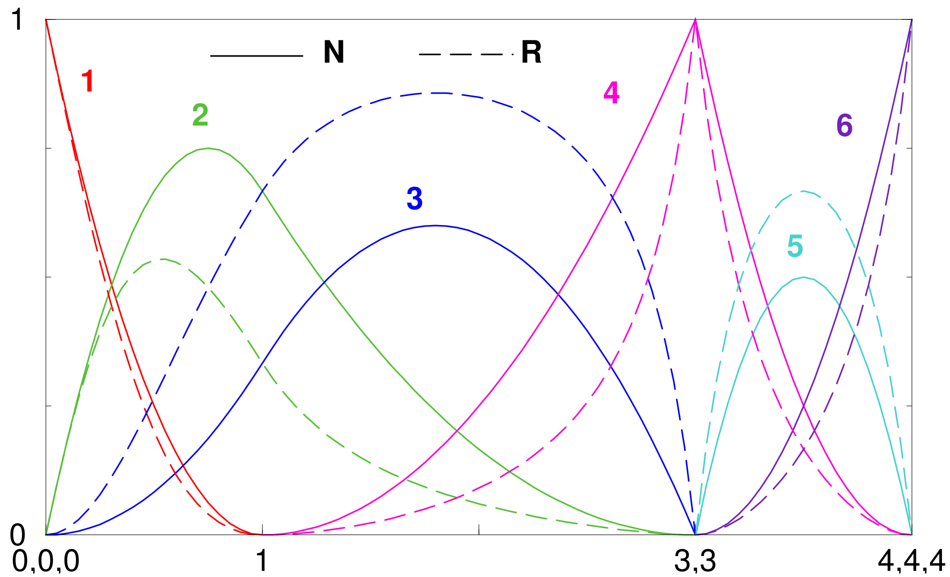
Rational bivariate B-spline basis functions

$$R_k(t) = \frac{N_k(u,v)w_k}{\sum_{m=1}^{n} N_m(u,v)w_m} \qquad k = 1, 2, \ldots, n \qquad w_k > 0$$

Quadratic NURBS curve



$$\boldsymbol{t} = \{0, 0, 0, 1, 3, 3, 4, 4, 4\}$$

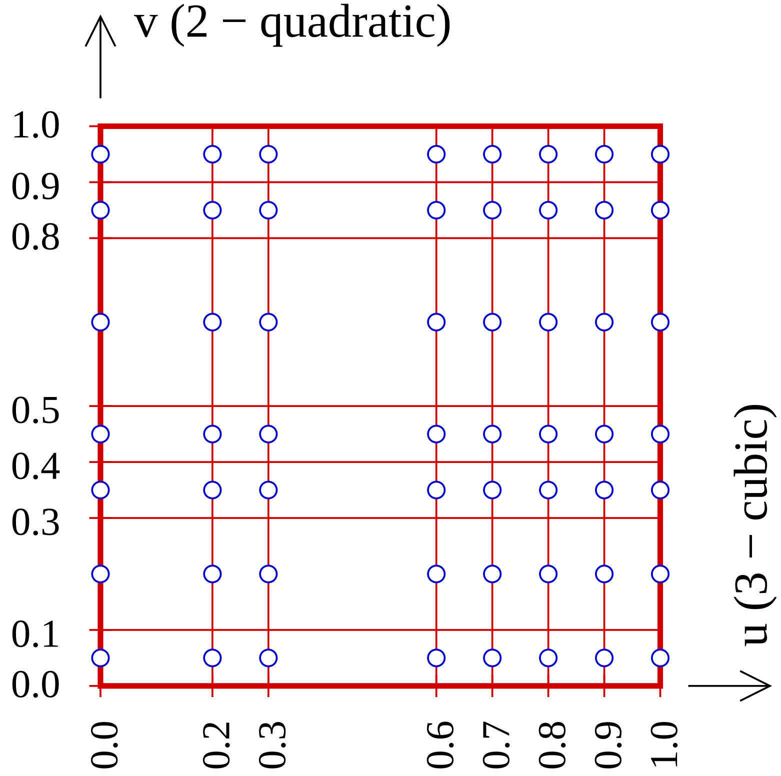


$$\boldsymbol{r}(t) = \sum_{j=1}^{6} R_i(t)\boldsymbol{P}_i$$

$$R_i(t) = \frac{N_i(t)w_i}{\sum_{j=1}^{6} N_j(t)w_j}$$

NURBS – Nonuniform Rational B-splines

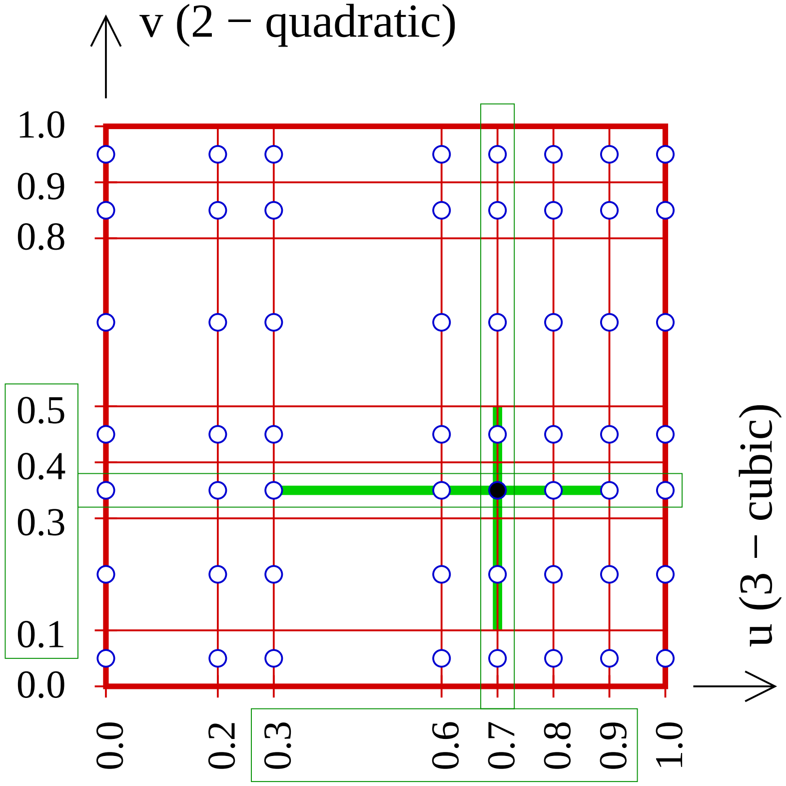
- **a NURBS patch is defined by**
  - **set of control points (coordinates and weights) topologically forming regular grid**
  - **global degrees of B-spline basis functions for each parametric direction of the patch**
  - **global knot vectors for each parametric direction of the patch**



NURBS – Nonuniform Rational B-splines

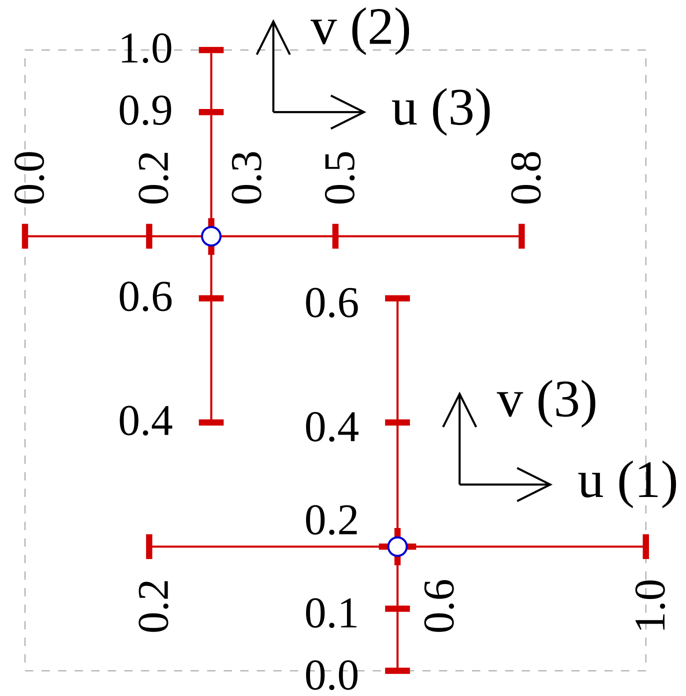
- **a NURBS patch is defined by**
  - **set of control points (coordinates and weights) topologically forming regular grid**
  - **global degrees of B-spline basis functions for each parametric direction of the patch**
  - **global knot vectors for each parametric direction of the patch**

⟹ **NURBS is fully structured**



PB-splines – Point-based B-splines

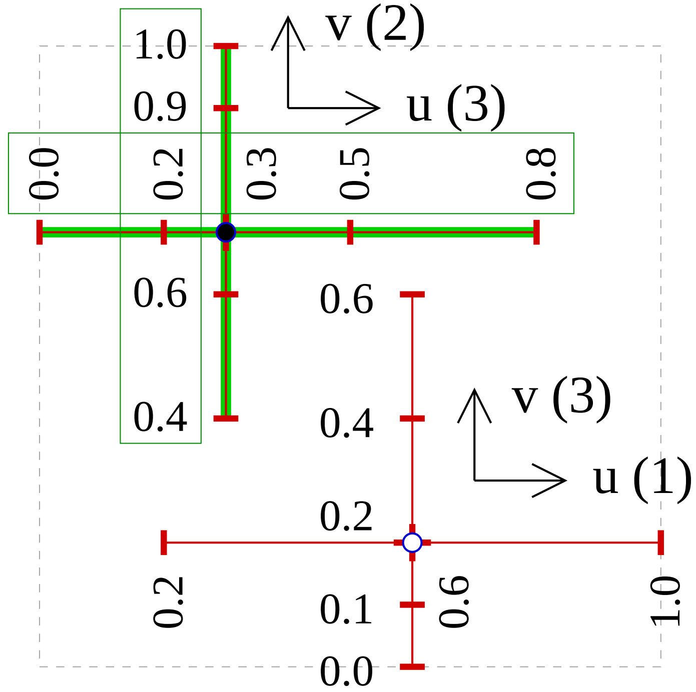
- **a PB-spline patch is defined by**
  - set of control points (coordinates and weights) topologically irregular
  - local degrees of B-spline basis functions for each parametric direction of each control point
  - local knot vectors for each parametric direction of each control point



PB-splines – Point-based B-splines

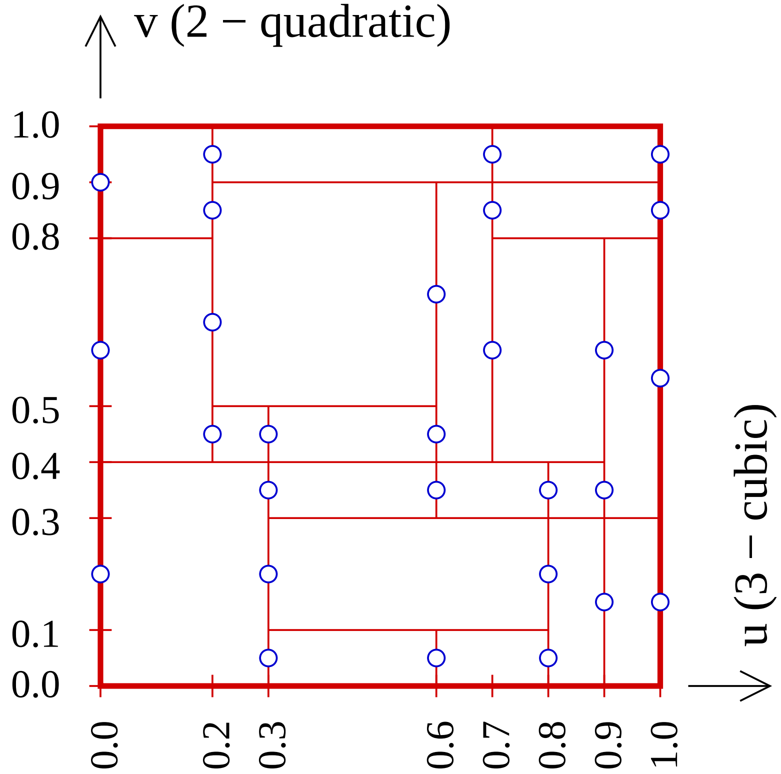
- **a PB-spline patch is defined by**
  - **set of control points (coordinates and weights) topologically irregular**
  - **local degrees of B-spline basis functions for each parametric direction of each control point**
  - **local knot vectors for each parametric direction of each control point**

$\Longrightarrow$ **PB-spline is fully unstructured**



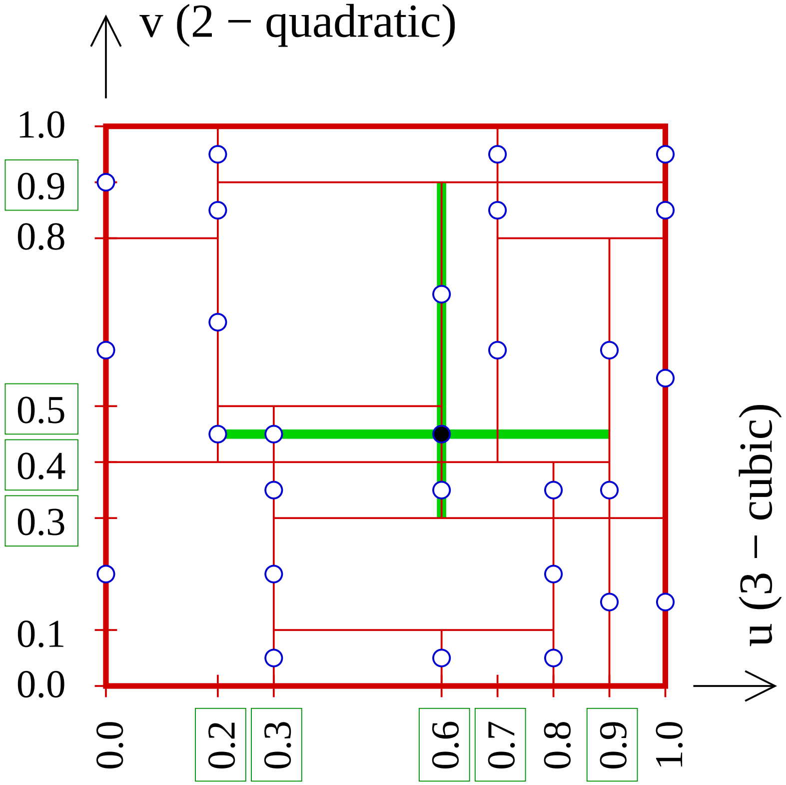
T-splines

- designed as compromise between NURBS and PB-splines
- a T-spline patch is defined by
  - set of control points (coordinates and weights) topologically consistent with a T-mesh
  - global degrees of B-spline basis functions for each parametric direction of the patch
  - global knot vectors for each parametric direction of the patch



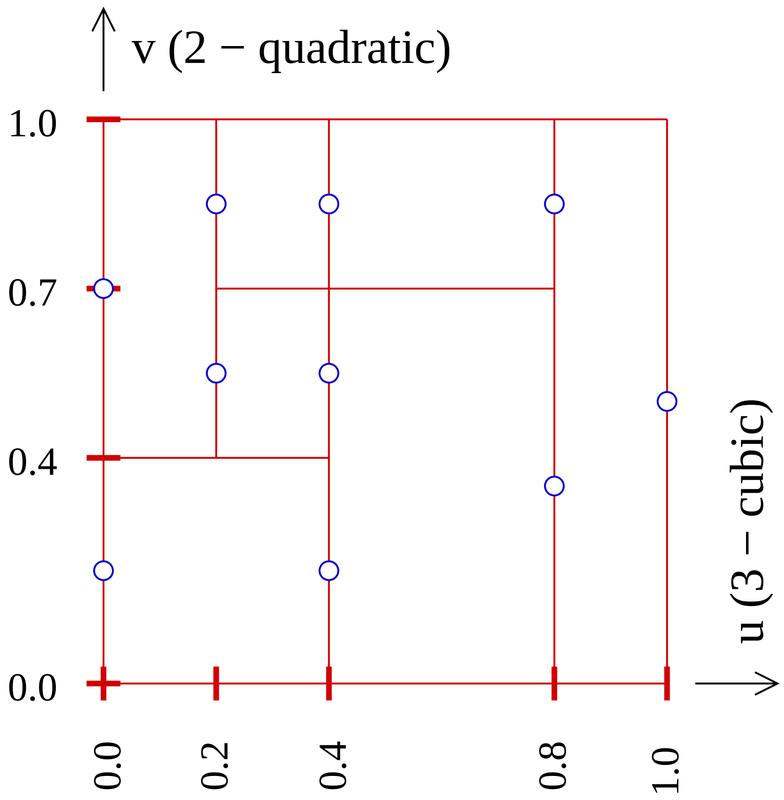
T-splines

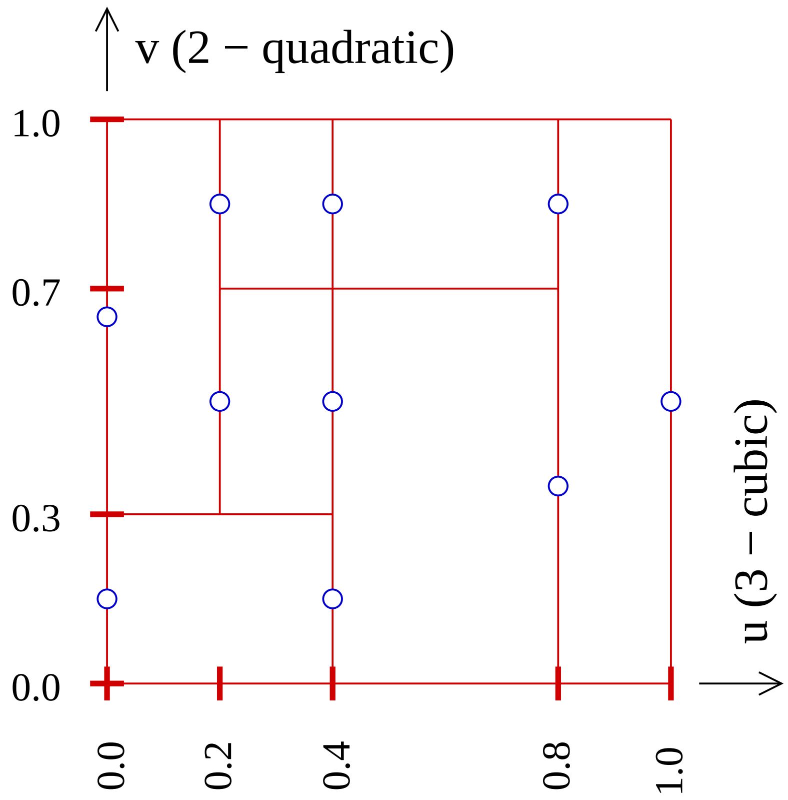
- designed as compromise between NURBS and PB-splines
- a T-spline patch is defined by
  - set of control points (coordinates and weights) topologically consistent with a T-mesh
  - global degrees of B-spline basis functions for each parametric direction of the patch
  - global knot vectors for each parametric direction of the patch



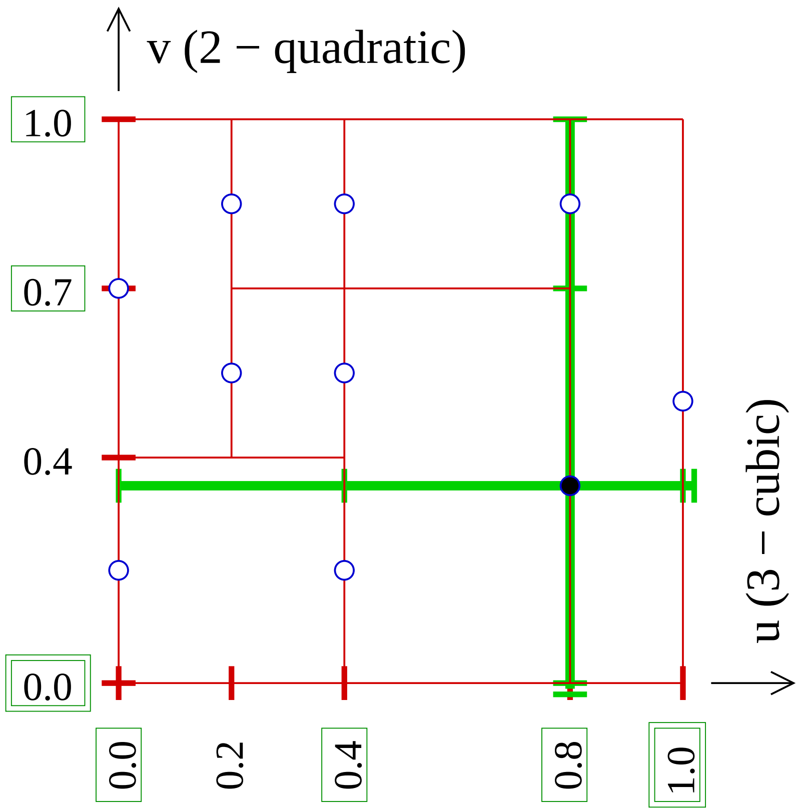
$\Longrightarrow$ T-spline is quasi-structured

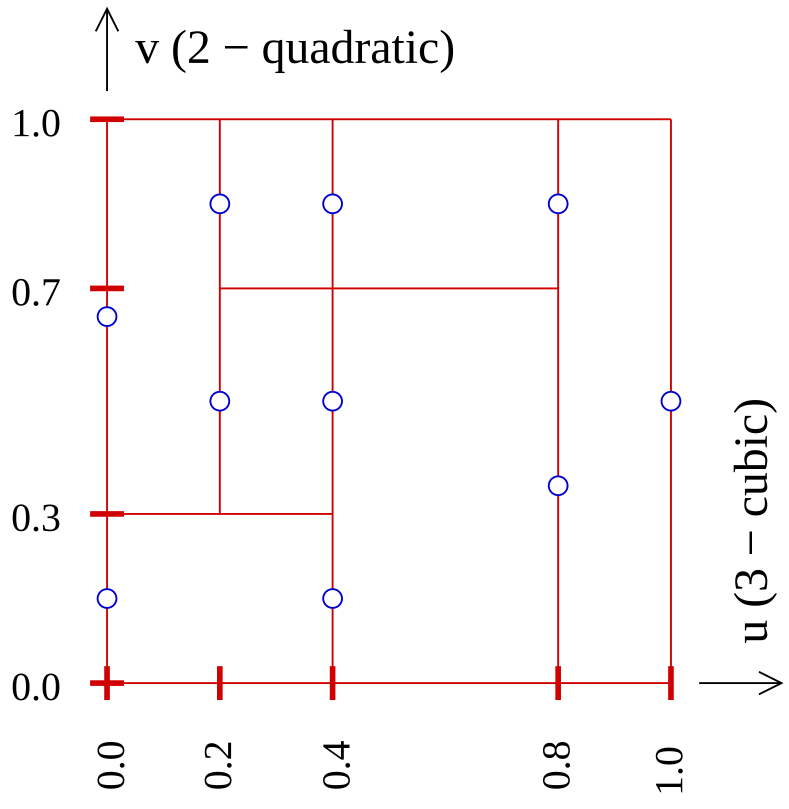
T-splines – local knot vector in parametric space



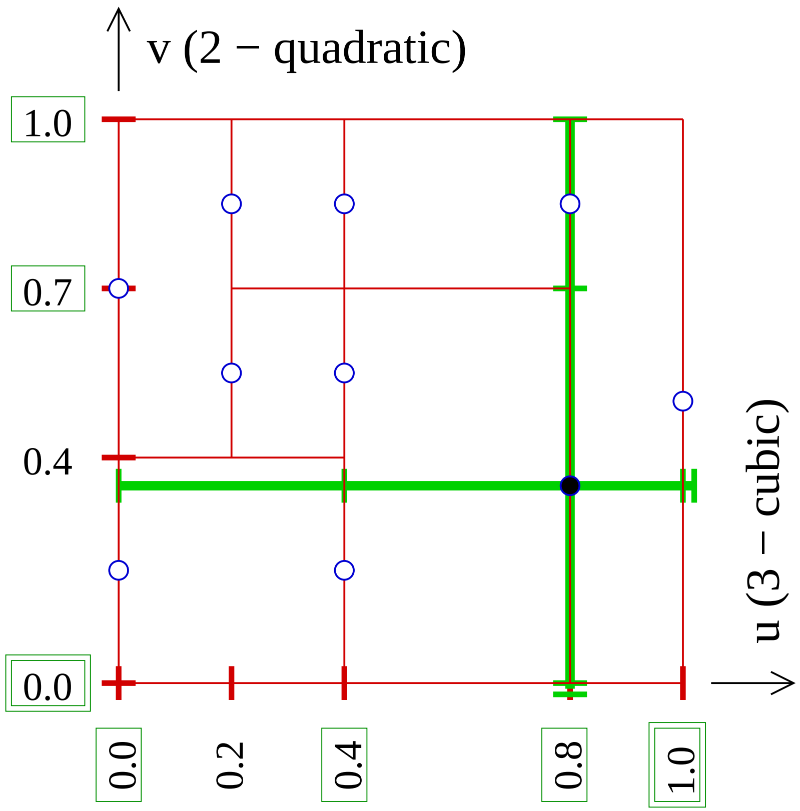


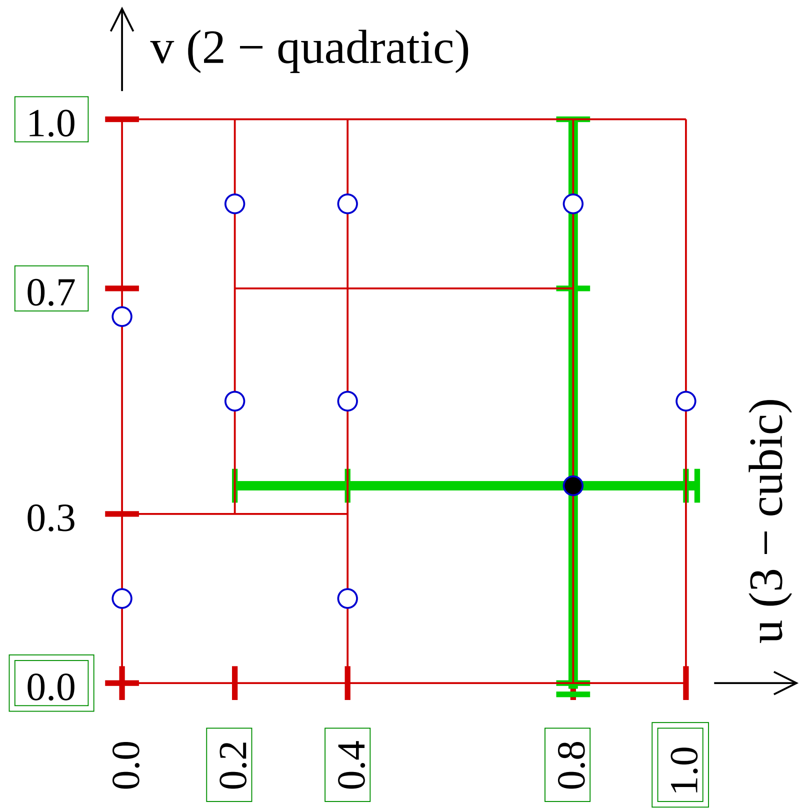
T-splines – local knot vector in parametric space



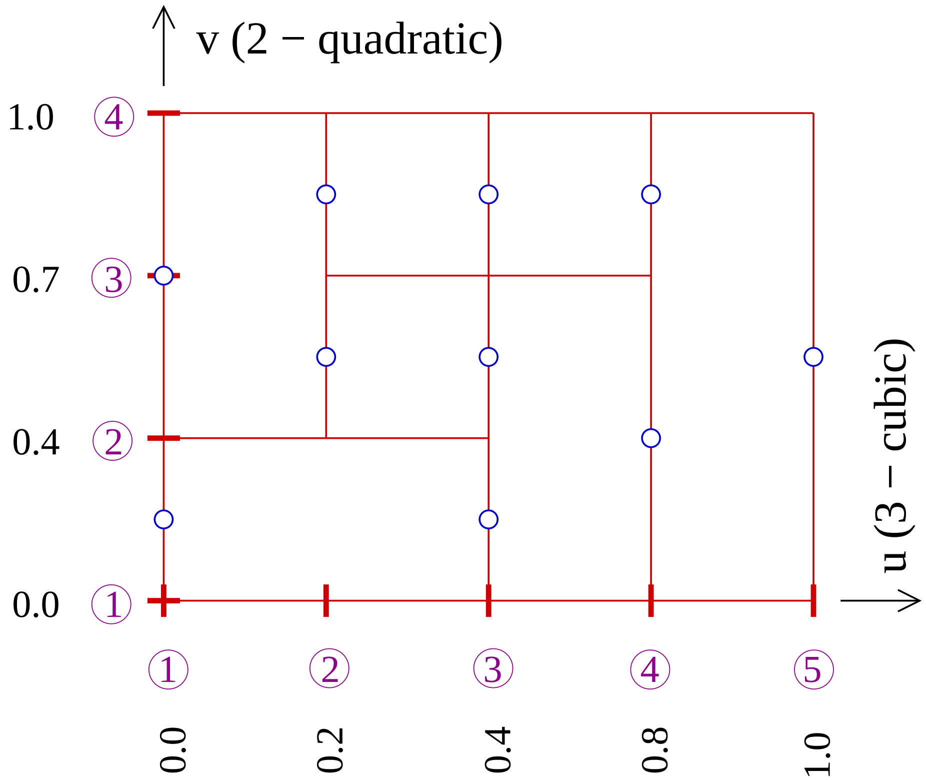


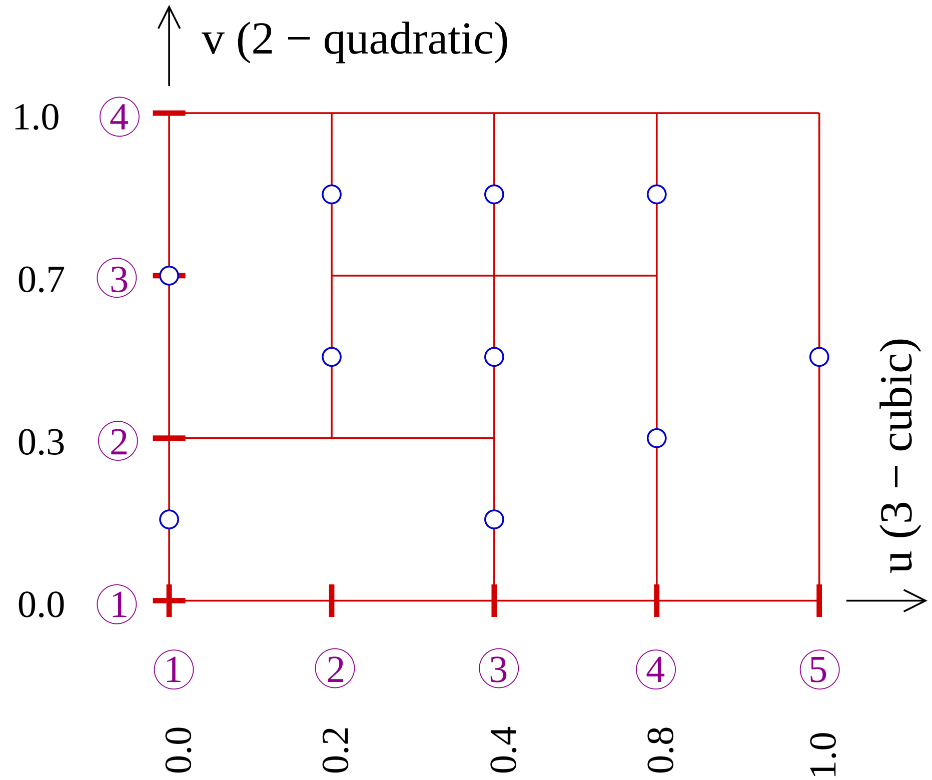
T-splines – local knot vector in parametric space



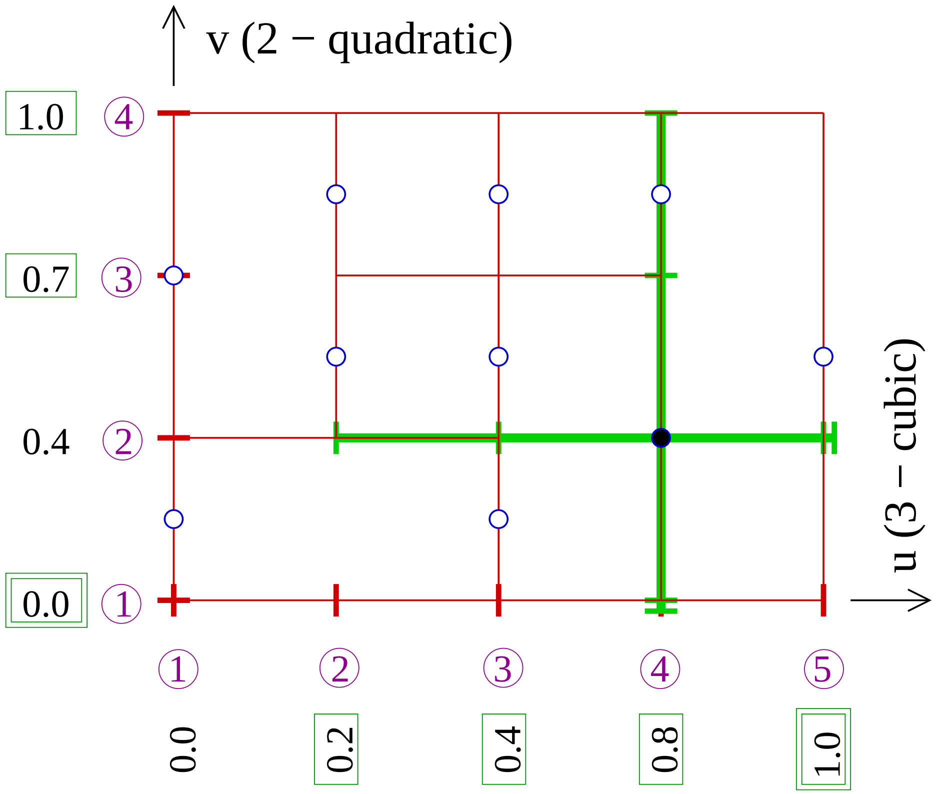


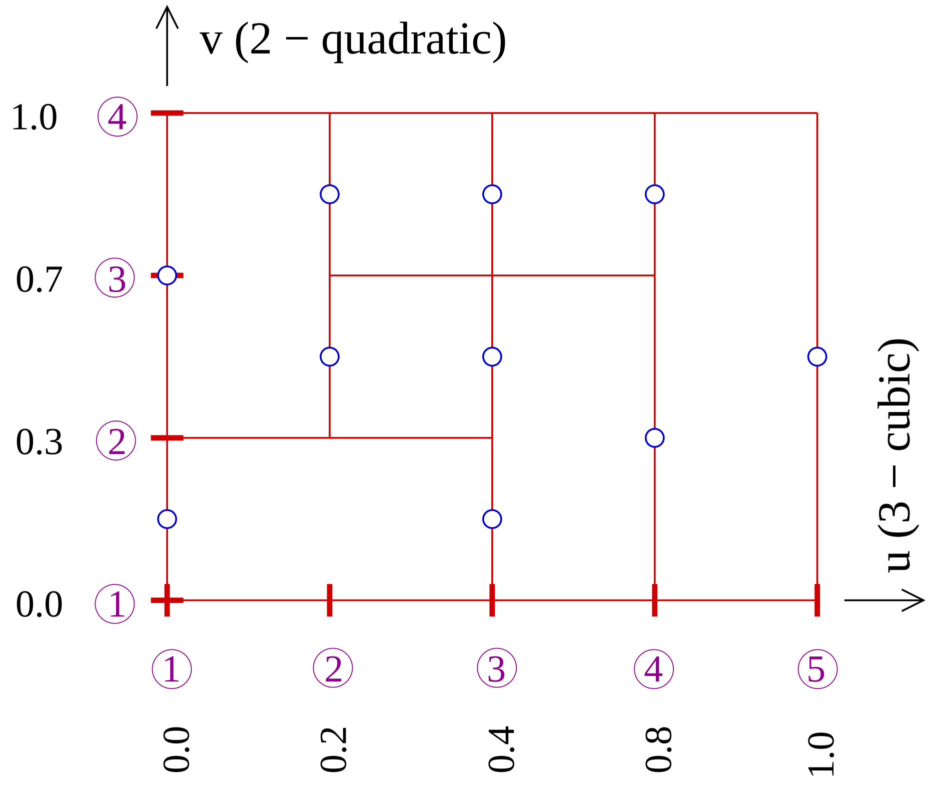
T-splines – local knot vector in index space



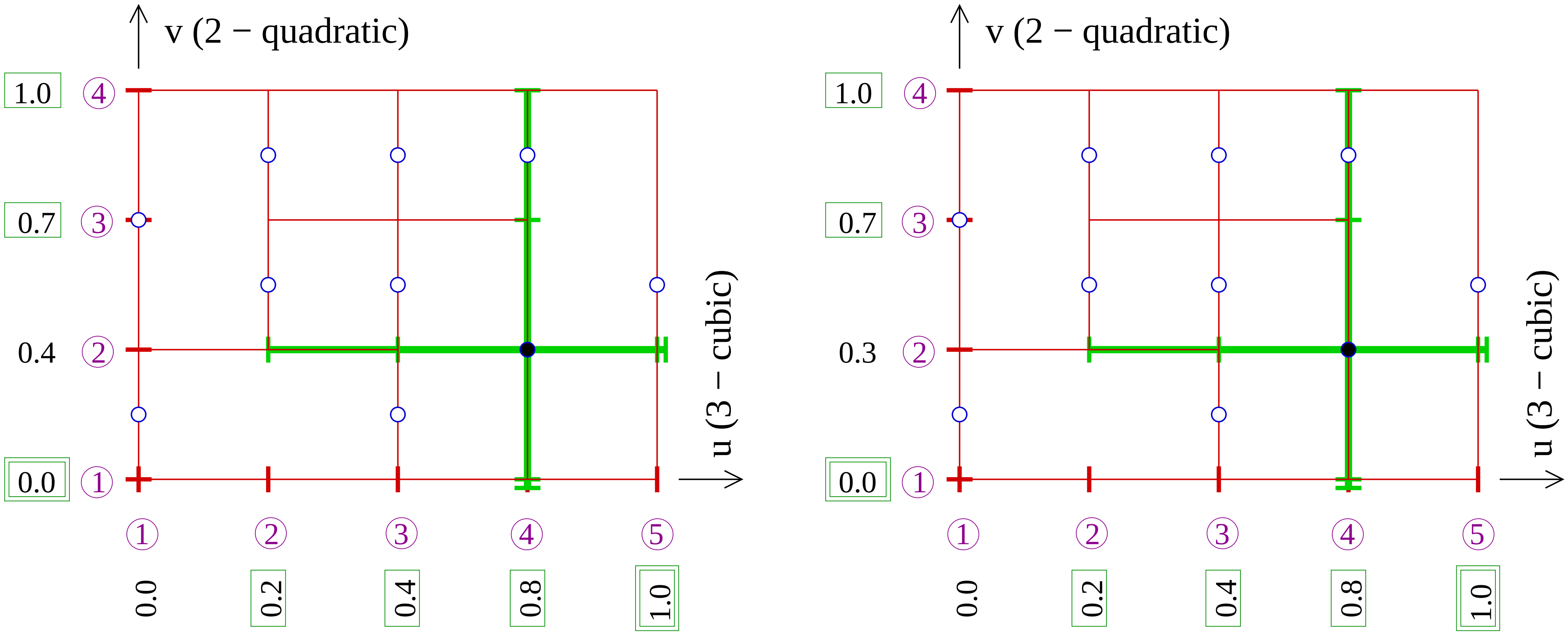


T-splines – local knot vector in index space





T-splines – local knot vector in index space



Object Oriented Design – Fundamental principles

- **encapsulation**
  (clustering together data and functionality)
- **inheritance**
  (reuse of existing code by derived classes)
- **abstraction / polymorphism**
  (transparent use of derived classes)
- **communication using messages**
  (general interface, safe data handling)

A good design is a trade-off between the level of implementation of object oriented principles and efficiency !

OOFEM

- **Object Oriented Finite Element Method computing environment**
- **open source distributed under the GNU Public License**
- **being continuously developed since 1997**
- **inspired by FEM_Object code** (EPFL Lausanne, 1993)
- **written in C++** (≈ 185.000 lines of code, ≈ 550 classes)
- **Ohloh analytics - 48 PersonYears**
- **modules for**
  - **structural mechanics**
  - **heat and mass transfer**
  - **fluid dynamics**

OOFEM – Features

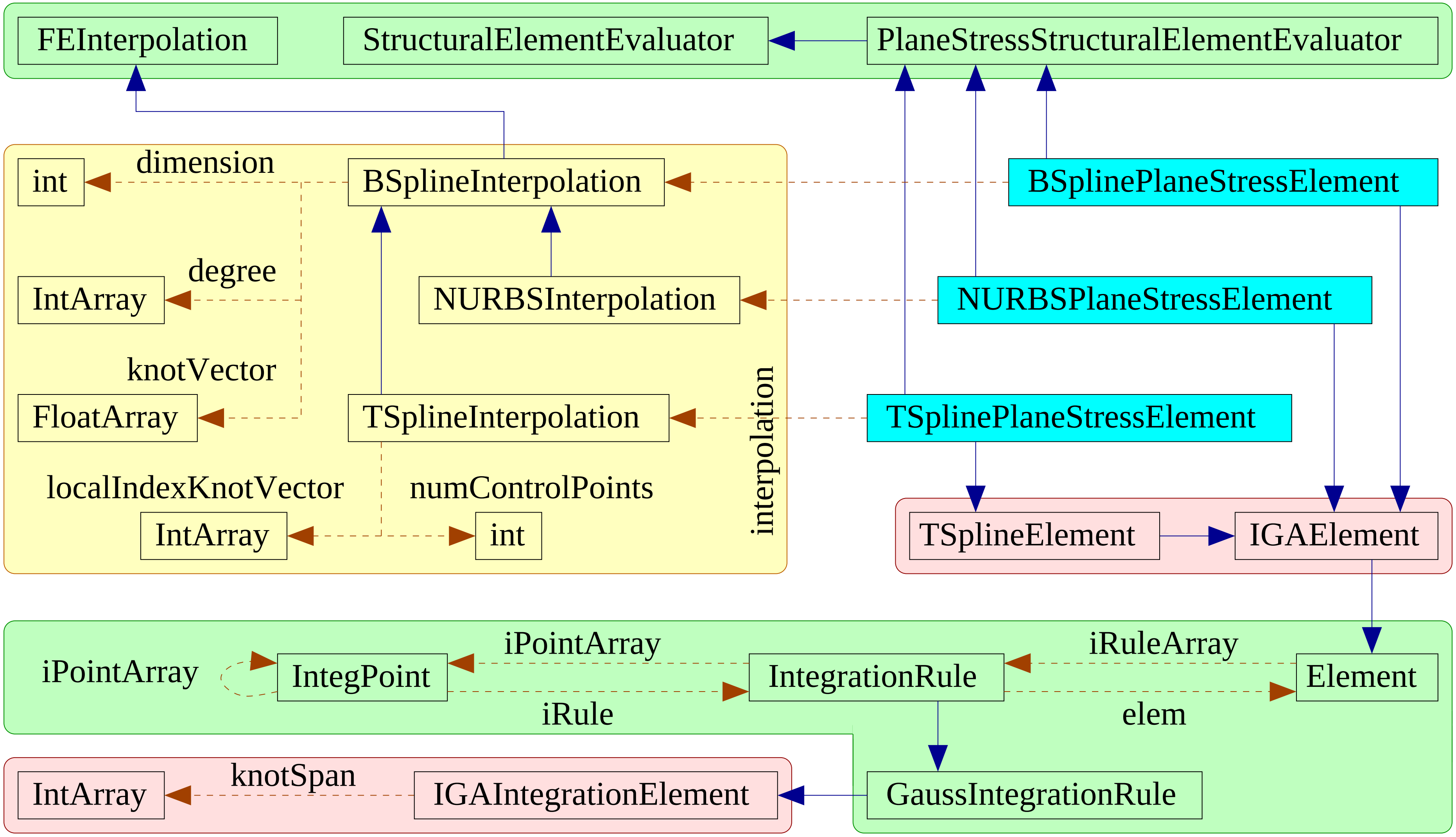
- **fully extensible** - a new element type, material model (with any internal history), BC, numerical algorithm, analysis module, ...
- **independent problem formulation, numerical solution and data storage**
- **full restart support**
- **staggered analysis support**
- **parallel processing support** - based on domain decomposition, message passing paradigms and dynamic load balancing
- **adaptive analysis support**
- **eXtended FEM support**
- **efficient sparse solvers** - interface to third party packages available

OO Design of IGA Module

- **strict separation of**
  - **interpolation**
  - **integration**
  - **analysis-specific functionality**

- **implementation of general IGA element**
- **implementation of integration on IGA element**
- **implementation of interpolation on IGA element**
- **implementation of analysis-specific IGA element**

OO Design of IGA Module



OO Design of IGA Module

```
StructuralElementEvaluator::computeStiffnessMatrix(FloatMatrix answer) {
   element = this->giveElement();
   ndofs = element->giveNumberOfDofs();

   answer.resize(ndofs, ndofs);
   answer.zero();

   loop over all integration rules (iRule) on the element {
      loop over all Gauss points (gp) of the iRule {
         B = this->computeStrainDisplacementMatrix(gp);
         D = this->computeConstitutiveMatrix(gp);
         dV = this->computeVolumeAround(gp);
         answer->add(product of B^T_D_B_dV);
      }
   }
}
```

OO Design of IGA Module

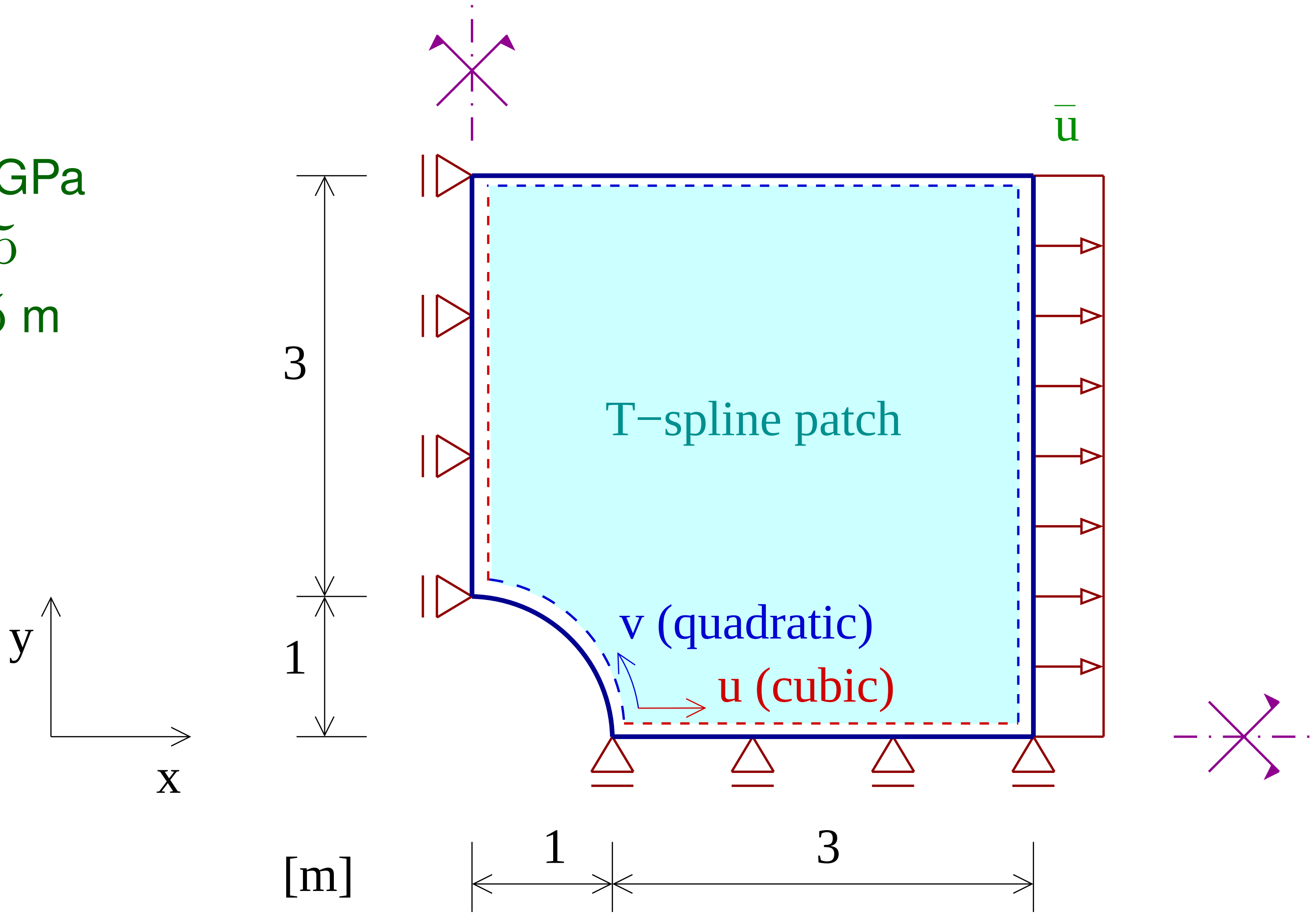
```
PlaneStressStructuralElementEvaluator::
      computeStrainDisplacementMatrix(FloatMatrix answer, IntegPoint gp) {
   FEInterpolation interp = gp->giveElement()->giveInterpolation();
   interp->evalShapeFunctDerivatives(der, gp);
   nnodes = gp->giveElement()->giveNumberOfNodes();

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

   for i=1:nnodes{
      answer.at(1, i*2-1) = der.at(i, 1);   // dN(i)/dx
      answer.at(2, i*2)   = der.at(i, 2);   // dN(i)/dy
      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
   }
}
```

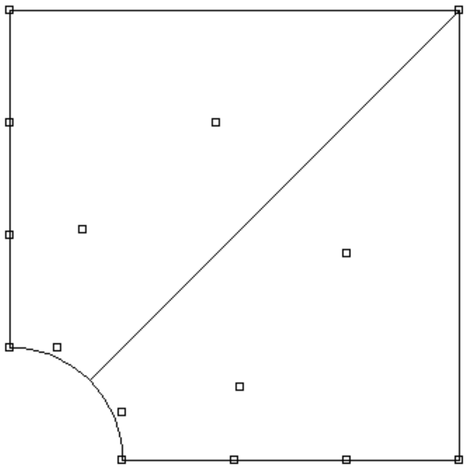
Numerical Example

$E = 15$ GPa
$\nu = 0.25$
$t = 0.15$ m
$\bar{u} = 1$ m

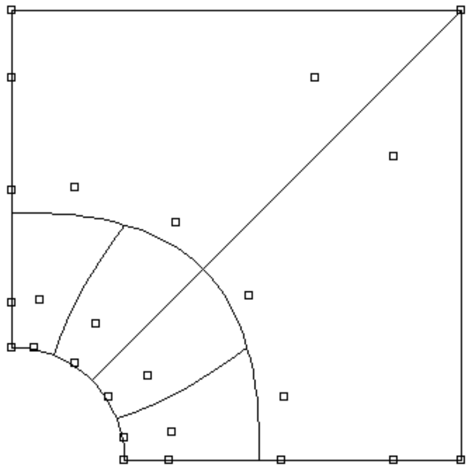


Numerical Example – IGA

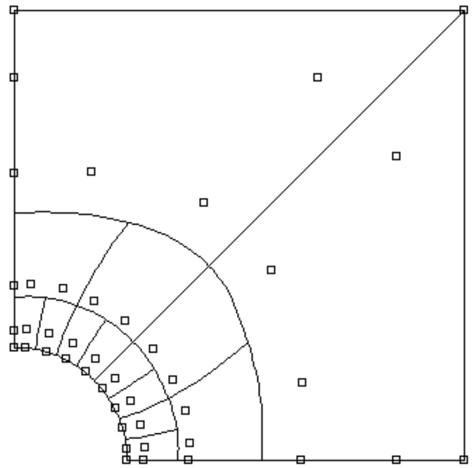
profile ε_{xx}



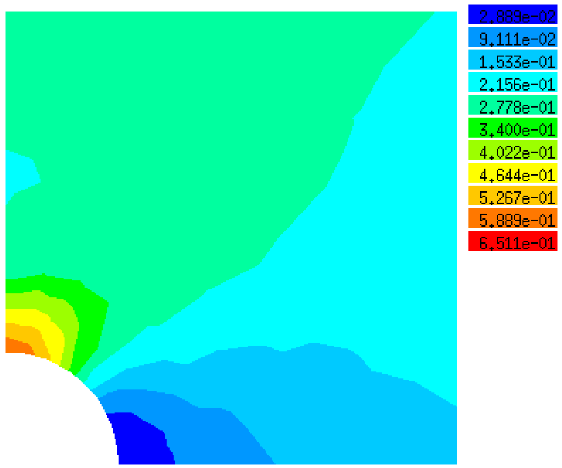
16 control points

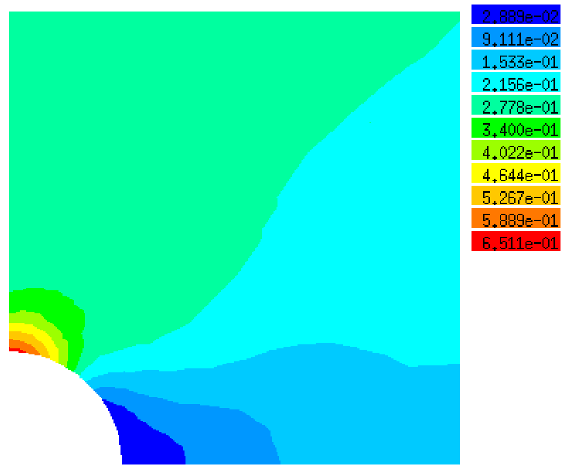


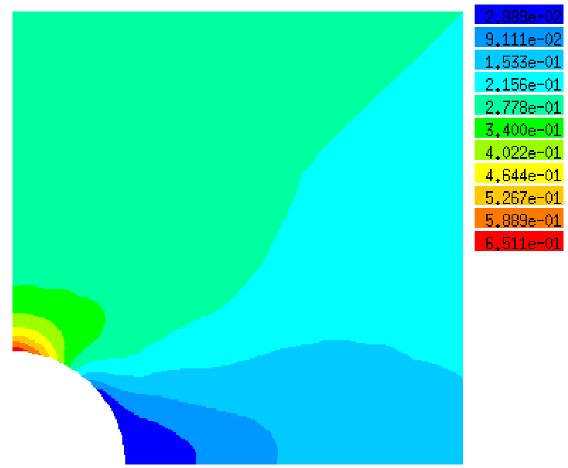
26 control points



44 control points

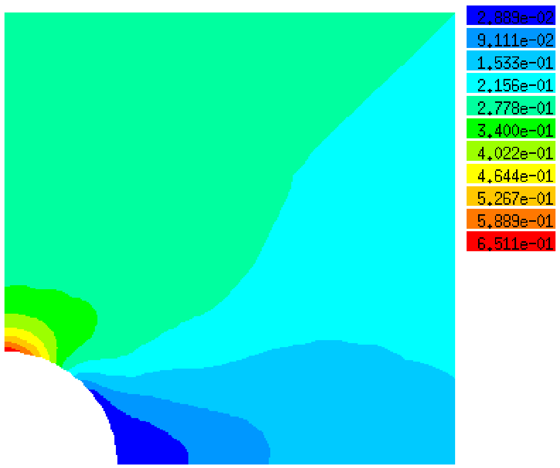


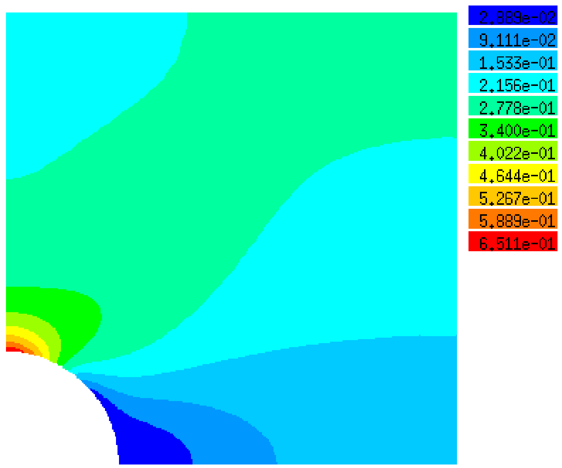


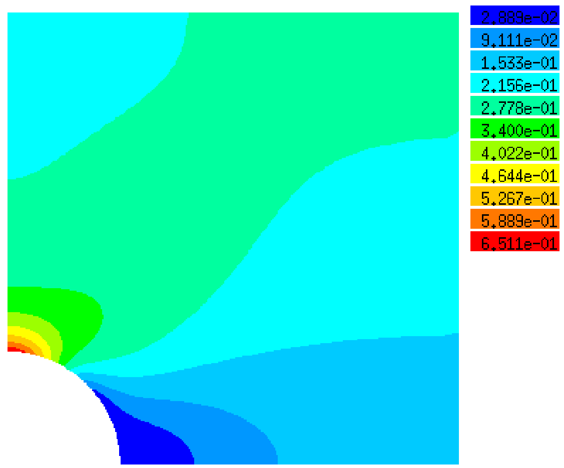


Numerical Example – IGA $\times$ FEA

profile ε_{xx}







IGA

3x2 T-spline

44 control points

IGA

5x5 NURBS

294 control points

FEA

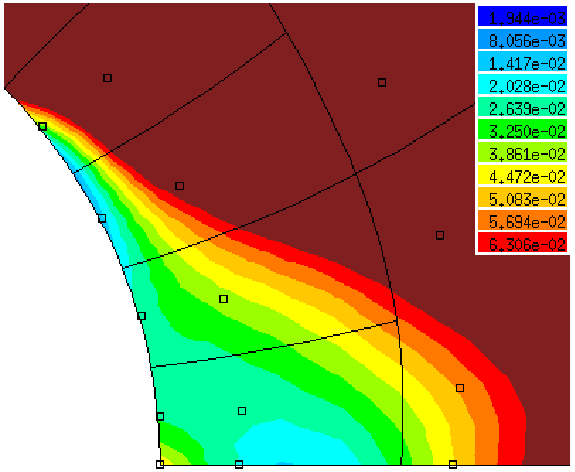
bilinear quads

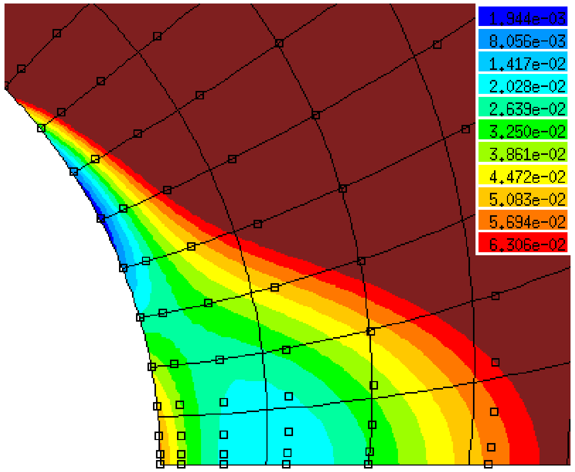
7345 nodes

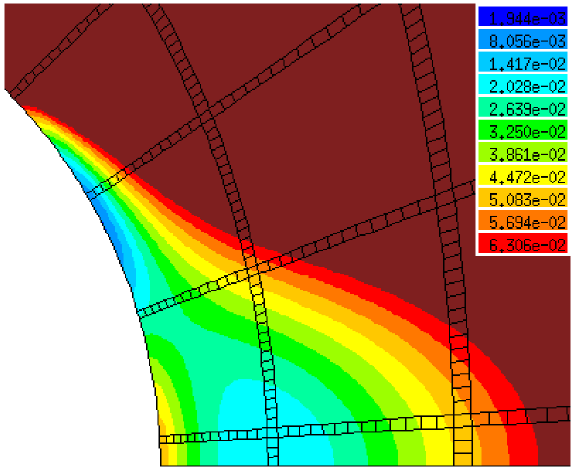
7168 elements

Numerical Example – IGA × FEA – detail

profile ε_{xx}







IGA	IGA	FEA
3x2 T-spline	5x5 NURBS	bilinear quads
44 control points	294 control points	7345 nodes
		7168 elements

Summary

- **implementation of an IGA module into an existing object oriented finite element code was presented**
- **emphasis was given on proper OO design**
  - **most of the functionality of the existing code reused**
  - **modularity and extensibility of the code preserved**
- **amount of modified and/or added code is rather limited mostly related to handling basis functions**
- **functionality of implementation was verified on numerical example**
- **T-spline based IGA proved to be a promising technology**

Acknowledgments

- **implemented into open source FEM package OOFEM**
- **the support by the Grant Agency of the Czech Republic (Project No. 103/09/2009) is gratefully acknowledged**