Chapter 5

Curvilinear Elements

This chapter is devoted to a special module of our code which implements elements with curved edges. Curved boundaries of the computional domain or curved material interfaces may require hundreds of linear elements only to describe their shape, but these elements are usually not needed by the solution itself. By using curvilinear elements we can avoid forced refinements and hence save a significant amount of degrees of freedom. In our code, curved parts of elements are described by NURBS curves, which are covered in the first section. Certain finite element procedures have to be changed in order to include curvilinear elements, such as the reference mapping, element refinement and numerical quadrature. These are described in the following sections. Finally, a numerical example is presented.

5.1 NURBS curves

NURBS, non-uniform rational B-splines, are standard tools for the representation of geometry. There are several reasons why we chose NURBS curves:

- NURBS can accurately represent both standard analytical shapes, such as lines, circles, ellipses, and free-form shapes like car bodies and human bodies.

- The NURBS evaluation rule can be implemented by a numerically stable and accurate algorithm.
- They can represent very complex curves using a small amount of data.

Every NURBS curve is defined by its **degree**, its **control points** with **weights** and the **knot vector**. The NURBS curve is a parametric curve and is defined as a vector-valued piecewise rational function by the following evaluation rule:

$$C(t) = \frac{\sum_{i=0}^{n} w_i \; P_i \; N_{i,k}(t)}{\sum_{i=0}^{n} w_i \; N_{i,k}(t)}, \tag{5.1}$$

where $t \in [0,1]$ represents time, w_i are weights, P_i are control points (x and y coordinates), n is the number of control points and $N_{i,k}$ are normalized B-spline basis functions of degree k. These basis functions are defined recursively as

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} N_{i,k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(t) \tag{5.2}$$

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

where t_i are knots that form the knot vector of length $m + 1$

$$V = \{t_0, t_1, \ldots, t_m\}.$$

The **degree** k is a positive integer, usually 1,2,3 or 5. Lines and polylines are of degree 1, circles have degree 2 and free-form curves are of degree 3 or 5. Sometimes the *order* of the NURBS curve is stated, *order = degree + 1*.

The **control points** $P_i,\ i = 0 \ldots n$, are the main tool for changing the shape of the curve. A curve of degree k must have at least $k + 1$ control points, ie., $n \geq k$. All control points have an associated weight $w_i \geq 0$ which influences the shape of

the curve near the corresponding control point. If $w_i = 0$ then P_i has no effect on the shape. As w_i increases, the curve is pulled towards P_i. If all control points have the same weight then the curve is called non-rational, otherwise it is rational (circles, ellipses, ...).

The **knot vector** is a sequence of $m + 1$ values that determines how much and where the control points influence the shape. The relation $m = n + k + 1$ must hold. The sequence is nondecreasing, $t_i \leq t_{i+1}$, and divides the whole interval $[0, 1]$ into smaller intervals which determine the area of influence of the control points. This relative parametric intervals (knot spans) need not be the same, i.e., the parametrization is non-uniform. Since the curved boundaries in our applications always start at the first control point and end at the last, the knot vector has a special form. It starts with $k + 1$ zeros and ends with $k + 1$ ones.

Example: Circular arc

Probably the most common shape in modeling is the circular arc, let us therefore construct a part of the circle with central angle α using NURBS. See Figure 5.1 for the notation.

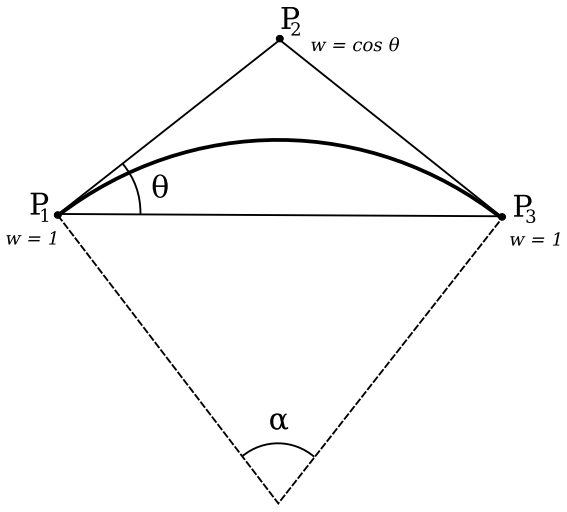

Figure 5.1: Circular arc

- The circle is a quadratic NURBS curve, i.e., $k = 2$.
- The curve is defined by three control points. The first and the last are the endpoints of the arc.
- The coordinates of the inner control point can be seen from the picture, where $\theta = \frac{\alpha}{2}$

$$\begin{aligned} x &= x_A + \frac{x_B - x_A}{2} - \frac{y_B - y_A}{2} \tan\theta \\ y &= y_A + \frac{y_B - y_A}{2} + \frac{x_B - x_A}{2} \tan\theta \end{aligned}$$

- The weights of the first and last control points are 1.0.
- The weight of the inner control point is given as $\cos\theta$.
- The knot vector is $\{0, 0, 0, 1, 1, 1\}$.

5.2 Reference mapping

In this section we construct the reference mapping from the master element to the curved element. Assume we have an element (triangle or quad) with curved edges and we have the parametrization of these edges obtained by the NURBS algorithm. From the previous section it follows that edges are parametrized by nonpolynomial functions and thus also the reference mapping is nonpolynomial. In order to store it and use it in calculations we need a suitable approximation - *isoparametric* approximation, which is a polynomial mapping defined in terms of master element shape functions.

Mapping from $\hat{K}_q$ onto curved quadrilaterals

We are looking for a mapping $\boldsymbol{X}_K(\boldsymbol{\xi}) : \hat{K}_q \to K$, where $\hat{K}_q$ is the reference domain and $K \in \mathcal{T}_{h,p}$ is a quadrilateral with edges parametrized by continuous functions

$\boldsymbol{C}^{e_1}(t), \ldots, \boldsymbol{C}^{e_4}(t) \subset R^2,\ t \in [-1, 1]$. These functions have to coincide at the vertices $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4$ of the physical element K.

$$\begin{aligned}
\boldsymbol{C}^{e_4}(1) = \boldsymbol{C}^{e_1}(-1) &= \boldsymbol{x}_1 \\
\boldsymbol{C}^{e_1}(1) = \boldsymbol{C}^{e_2}(-1) &= \boldsymbol{x}_2 \\
\boldsymbol{C}^{e_2}(1) = \boldsymbol{C}^{e_3}(-1) &= \boldsymbol{x}_3 \\
\boldsymbol{C}^{e_3}(1) = \boldsymbol{C}^{e_4}(-1) &= \boldsymbol{x}_4
\end{aligned}$$

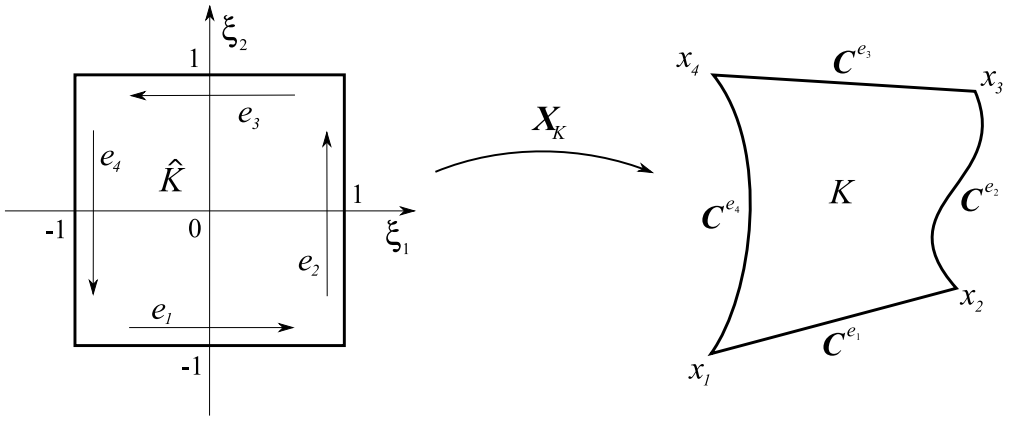

Figure 5.2: Reference mapping

Now, the mapping $\boldsymbol{X}_K$ is defined by the following formula

$$\begin{aligned}
\boldsymbol{X}_K(\boldsymbol{\xi}) &= \frac{1-\xi_2}{2}\boldsymbol{C}^{e_1}(\xi_1) + \frac{1+\xi_1}{2}\boldsymbol{C}^{e_2}(\xi_2) + \frac{1+\xi_2}{2}\boldsymbol{C}^{e_3}(-\xi_1) \\
&+ \frac{1-\xi_1}{2}\boldsymbol{C}^{e_4}(-\xi_2) - \frac{(1-\xi_1)(1-\xi_2)}{4}\boldsymbol{x}_1 - \frac{(1+\xi_1)(1-\xi_2)}{4}\boldsymbol{x}_2 \\
&- \frac{(1+\xi_1)(1+\xi_2)}{4}\boldsymbol{x}_3 - \frac{(1-\xi_1)(1+\xi_2)}{4}\boldsymbol{x}_4
\end{aligned} \tag{5.3}$$

The formula is justified by the so-called transfinite interpolation. For details, see [26].

Mapping from $\hat{K}_t$ onto curved triangles

Assume that the triangle K has edges parametrized by continuous functions $\boldsymbol{C}^{e_1}(t), \boldsymbol{C}^{e_2}(t), \boldsymbol{C}^{e_3}(t), t \in [-1, 1]$ and it holds

$$\begin{aligned} \boldsymbol{C}^{e_3}(1) = \boldsymbol{C}^{e_1}(-1) &= \boldsymbol{x}_1 \\ \boldsymbol{C}^{e_1}(1) = \boldsymbol{C}^{e_2}(-1) &= \boldsymbol{x}_2 \\ \boldsymbol{C}^{e_2}(1) = \boldsymbol{C}^{e_3}(-1) &= \boldsymbol{x}_3 \end{aligned}$$

Again without going into details we are using the following transfinite interpolation scheme

$$\boldsymbol{X}_K(\boldsymbol{\xi}) = \boldsymbol{X}_K^v(\boldsymbol{\xi}) + \boldsymbol{X}_K^e(\boldsymbol{\xi}),\ \boldsymbol{\xi} \in \hat{K}_t,$$

where $\boldsymbol{X}_K^v$ is the vertex part and $\boldsymbol{X}_K^e$ is the higher-order part. The vertex interpolant is defined as

$$\boldsymbol{X}_K^v(\boldsymbol{\xi}) = \sum_{i=1}^{3} \boldsymbol{x}_i \lambda_i(\boldsymbol{\xi}),$$

where $\boldsymbol{x}_i$ are vertices and λ_i are scalar vertex functions (pyramids), and the higher-order part as

$$\boldsymbol{X}_K^e(\boldsymbol{\xi}) = \sum_{i=1}^{3} \bar{\boldsymbol{C}}^{e_i}(\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}))\ \lambda_A(\boldsymbol{\xi})\ \lambda_B(\boldsymbol{\xi}),$$

where λ_A and λ_B are vertex functions corresponding to the endpoints of the edge e_i and

$$\bar{\boldsymbol{C}}^{e_i}(t) = \boldsymbol{C}^{e_i}(t) - \boldsymbol{C}^{e_i}(-1)\frac{1-t}{2} - \boldsymbol{C}^{e_i}(1)\frac{1+t}{2}.$$

Geometrically, the latter means subtracting the straight part from the curved edge.

Isoparametric approximation of reference maps

As we said at the begining of this section, reference mappings $\boldsymbol{X}_K(\boldsymbol{\xi}) : \hat{K} \to K$ are generally nonpolynomial. In order to work with reference maps in the finite

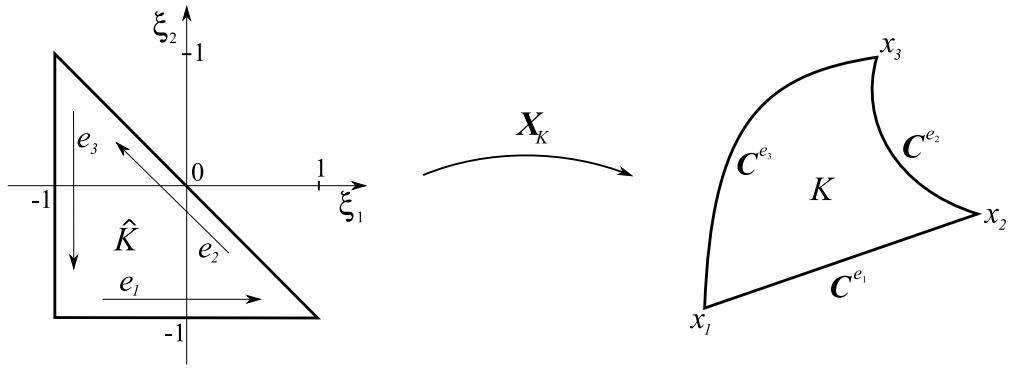

Figure 5.3: Reference mapping

element method, we need a polynomial approximation $\boldsymbol{x}_K(\boldsymbol{\xi}) \approx \boldsymbol{X}_K(\boldsymbol{\xi})$, defined for each element as a linear combination of master element shape functions (scalar H^1 functions) with vector-valued coefficients. This isoparametric approximation is easy to integrate. Values, derivatives and inverse derivatives can be obtained easily: we actually store only the coefficients of the linear combination. These vector-valued coefficients are constructed by projection-based interpolation. In this technique, the interpolant is constructed as a sum of vertex, edge and bubble interpolants:

$$\boldsymbol{x}_K = \boldsymbol{u}^v + \boldsymbol{u}^e + \boldsymbol{u}^b,$$

where interpolants $\boldsymbol{u}^v$, $\boldsymbol{u}^e$ and $\boldsymbol{u}^b$ are linear combinations of vertex, edge and bubble shape functions, respectively. The accuracy of the isoparametric approximation is controled by the polynomial order p of the shape functions.

Vertex interpolant matches the nonpolynomial map $\boldsymbol{X}_K$ at all element vertices, therefore

$$\boldsymbol{u}^v = \sum_{i=1}^{n} \boldsymbol{a}_i^v \varphi_i^v(\boldsymbol{\xi}), \tag{5.4}$$

where n is the number of element edges, φ_i^v are scalar vertex functions and the vector-valued coefficients $\boldsymbol{a}_i^v$ are equal to the coordinates of the vertices of the physical element.

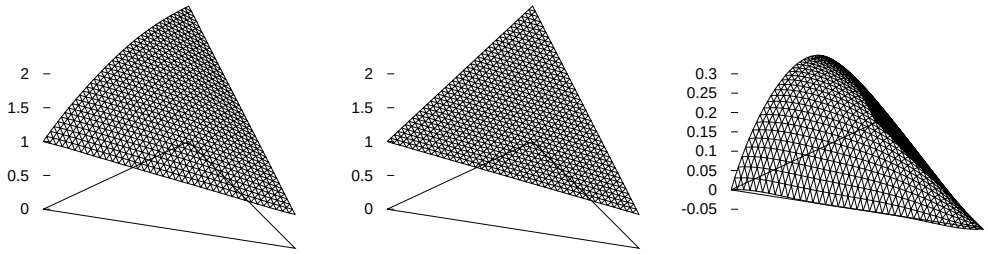

Figure 5.4: One component of the nonpolynomial mapping $\boldsymbol{X}_K$, its vertex interpolant $\boldsymbol{u}^v$ and the residual $\boldsymbol{X}_k - \boldsymbol{u}^v$.

Edge interpolant is constructed as a sum of edge interpolants for each element edge.

$$\boldsymbol{u}^e = \sum_{j=1}^{n} \boldsymbol{u}^{e_j}.$$

Approximation theory suggests that the distance between the residual $\boldsymbol{X}_k - \boldsymbol{u}^v$ and the edge interpolant $\boldsymbol{u}^e$ should be minimized on edges in the norm $H_{00}^{\frac{1}{2}}(e_j)$. This is a nontrivial norm and in calculations it is usually substituted by the H_0^1 norm. Therefore, we solve the following minimization problem

$$||(\boldsymbol{X}_k - \boldsymbol{u}^v) - \boldsymbol{u}^{e_j}||_{H_0^1(e_j)} \to \min, \quad j = 1, \ldots, n. \tag{5.5}$$

This problem can be transformed into a system of $p-1$ linear equations for unknown coefficients $\boldsymbol{a}_i^{e_j},\ i = 2, \ldots, p$,

$$\sum_{i=2}^{p} \alpha_i^{e_j} (\varphi_i^{e_j}, \varphi_k^{e_j})_{H_0^1(e_j)} = ((\boldsymbol{X}_k - \boldsymbol{u}^v), \varphi_k^{e_j})_{H_0^1(e_j)} \quad k = 2, 3, \ldots, p \tag{5.6}$$

where the edge interpolant $\boldsymbol{u}^{e_j}$ is written as a linear combination of shape functions

$$\boldsymbol{u}^{e_j} = \sum_{i=2}^{p} \alpha_i^{e_j} \varphi_i^{e_j}(\boldsymbol{\xi})$$

Remark 5.1 *Notice that in the Figure 5.5 on the right the residual $\boldsymbol{X}_k - \boldsymbol{u}^v - \boldsymbol{u}^e$ vanishes in all vertices, but it does not vanish on edges. The mapping $\boldsymbol{X}_k$*

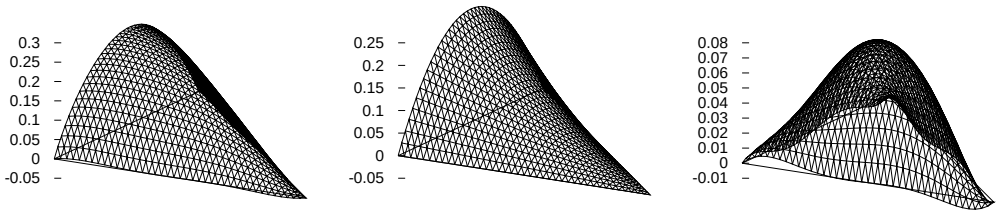

Figure 5.5: Residual $\boldsymbol{X}_k - \boldsymbol{u}^v$, edge interpolant $\boldsymbol{u}^e$ and residual $\boldsymbol{X}_k - \boldsymbol{u}^v - \boldsymbol{u}^e$.

which we interpolate by polynomial functions is originally a nonpolynomial function and therefore it cannot be approximated exactly. It is obvious that the higher the polynomial degree p, the smaller the error on edges.

Bubble interpolant is obtained by projecting the residual $\boldsymbol{X}_k - \boldsymbol{u}^v - \boldsymbol{u}^e$ in H^1-seminorm onto the polynomial space generated by the bubble shape functions $\varphi_k^b,\ k = 1, \ldots, n_p$, where n_p denotes the number of bubble functions for polynomial degree p. The corresponding minimization problem reads

$$|(\boldsymbol{X}_k - \boldsymbol{u}^v - \boldsymbol{u}^e) - \boldsymbol{u}^b|_{H^1} \to \min, \tag{5.7}$$

where the bubble interpolant is written as a linear combination

$$\boldsymbol{u}^b = \sum_{i=1}^{n_p} \alpha_i^b \varphi_i^b(\boldsymbol{\xi}).$$

Therefore, we can rewrite the minimization problem (5.7) as a system of algebraic equations

$$\int \nabla((\boldsymbol{X}_k - \boldsymbol{u}^v - \boldsymbol{u}^e) - \boldsymbol{u}^b)\ \nabla\varphi_k^b\ \mathrm{d}\boldsymbol{x} = 0 \tag{5.8}$$

for unknown coefficients $\boldsymbol{a}_k^b$. This is illustrated in Figure 5.6, where the final error of our interpolation is displayed on the right.

Remark 5.2 *Projection-based interpolation can be done in the same way on the quadrilateral reference domain $K_q = (-1, 1)^2$ with the corresponding shape functions.*

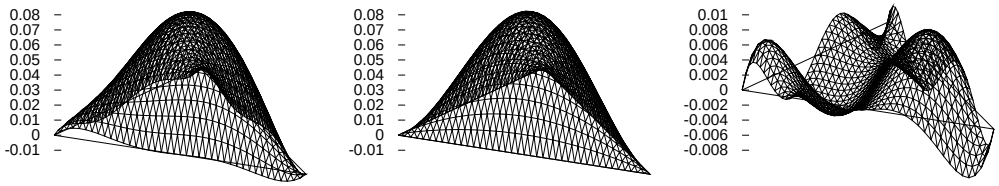

Figure 5.6: Residual $\boldsymbol{X}_k - \boldsymbol{u}^v - \boldsymbol{u}^e$, bubble interpolant $\boldsymbol{u}^b$ and residual $\boldsymbol{X}_k - \boldsymbol{u}^v - \boldsymbol{u}^e - \boldsymbol{u}^b$.

5.3 Implementation

This section is devoted to the implementation of the curved element module. Necessary mesh structures for curvilinear elements are described.

Every element has a member `cm` which is a pointer to `CurvMap`. CurvMap is a structure storing information about curved boundary of the element.

```
struct CurvMap
{
  bool toplevel;
  union
  {
    Nurbs* nurbs[4];
    struct
    {
      Element* parent;
      unsigned part;
    };
  };

  int order;
  int nc;
  double2* coefs;
};
```

The structure has two variants, depending on the value of `toplevel`. If `toplevel` is `true`, it means that the element is a base element (defined by the user using

NURBS curves). The structure stores up to four pointers to NURBS curved edges. The structure `Nurbs` stores curve data, such as control points, weights and the knot vector. If `toplevel` is `false`, it means that the structure belongs to a refined element. Then `parent` points to the base element which stores all data about the curved boundary and `part` describes what is the location of the refined element inside the base element. The other members pertain to the reference mapping approximation. The variable `order` represents the polynomial order of the approximation, `nc` is the number of coefficients and the array `coefs` is used to store the calculated coefficients of the linear combination of shape functions approximating the reference mapping.

The following function can be called:

```
update_refmap_coefs(Element* e)
```

It performs the projection-based interpolation to determine the array `coefs`. Projection-based interpolation is done in 3 steps. Vertex interpolant is easy to obtain, but to find the coefficients for edge and bubble interpolant one must solve the systems of algebraic equations (5.6) and (5.8). In order to do that, we first precalculate matrices for the highest polynomial degree. For example, the projection matrix for edges

$$A_{ij} = (\varphi_i^{e_k}, \varphi_j^{e_k})_{H_0^1(e_k)} \quad \text{for } i, j = 2, \ldots, p$$

is calculated for $p = 10$ (maximal degree). The same is done for matrices constructed from bubble shape functions on both triangles and quadrilaterals. These matrices are symmetric and positive definite, we can thus calculate their Cholesky decompositions

$$A = L \cdot L^T,$$

where L is a lower triangular matrix. The system of algebraic equations of the form $A\boldsymbol{x} = \boldsymbol{b}$ can then be solved as two systems with triangular matrices:

$$\begin{aligned} L\boldsymbol{y} &= \boldsymbol{b} \\ L^T\boldsymbol{x} &= \boldsymbol{y} \end{aligned}$$

Now let us consider the matrix A for a smaller polynomial degree p. The advantage of Cholesky decomposition lies in the fact that Cholesky decomposition of the smaller matrix is a submatrix of L. Therefore, we can precalculate the matrix L for the maximal order and then just use its submatrices to solve systems for all orders $p \leq 10$. This optimizes the whole projection-based interpolation, because the work of constructing the matrix A and decomposing it is done only once.

We also have to adjust refining of elements in order to allow splits of curved elements. When refining linear elements we just find a center of each edge (and possibly center of an element) and we create element sons using these centers as vertices of new elements. For curvilinear elements we have to adjust coordinates of centers using the original nonpolynomial map from reference element onto curved element. Once we have corrected coordinates of vertices we can create new elements. They do not store information about their curved boundary, they contain only pointer to the `parent` base element and `part` where they are located within it. First time the new element is used, the reference mapping is calculated using `Nurbs` and `update_refmap_coefs(Element* e)` with the same order of approximation. Therefore, curved boundary of new elements is approximated more accurately than the boundary of their parents.

5.4 Numerical quadrature

The choice of a numerical quadrature rule in our calculations is an important decision. Quadrature rule which is too small can negatively affect final solution. When assembling stiffness matrix we are, for example, evaluating the following integral

$$\int_K \varphi(\boldsymbol{x})\psi(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \left(\frac{\mathrm{d}\boldsymbol{X}_K}{\mathrm{d}\boldsymbol{\xi}}\right)^{-1} \hat{\varphi}(\boldsymbol{\xi}) \left(\frac{\mathrm{d}\boldsymbol{X}_K}{\mathrm{d}\boldsymbol{\xi}}\right)^{-1} \hat{\psi}(\boldsymbol{\xi})(J_{\boldsymbol{X}_K})\ \mathrm{d}\boldsymbol{\xi}. \tag{5.9}$$

To evaluate this integral on reference domain $\hat{K}$ accurately we need quadrature rule of sufficiently high order. Polynomial orders of shape functions $\hat{\varphi}$ and $\hat{\psi}$ are known, but the order of the inverse of reference map may not be. Reference mapping of

linear triangles is constant and so is the inverse. On the other hand the inverse of the reference mapping of quadrilateral and curved elements can be generally nonpolynomial. Therefore for each element we precalculate sufficient order needed to integrate the inverse exactly.

In order to find it we are computing for each element an artificial integral, which is similar to (5.9) but without shape functions.

$$\int_{\hat{K}} \left(\left(\frac{\mathrm{d}\boldsymbol{X}_K}{\mathrm{d}\boldsymbol{\xi}} \right)^{-1} \right)^2 (J_{\boldsymbol{X}_K}) \, \mathrm{d}\boldsymbol{\xi}. \tag{5.10}$$

Using the quadrature rules of higher and higher orders on reference domain until we reach the one which is sufficient. The polynomial order of the integrand in (5.10) is the order which has to be added to the polynomial orders of shape functions to get sufficient integration rule for (5.9). If the order is too high, the curved element is too distorted and there may not be enough integration rules to calculate it accurately. In this case the adaptive quadrature should be used.

The algorithm of adaptive quadrature for the integral (5.9) is described below. For simplicity we denote the integrand in (5.9) by $f(\boldsymbol{x})$.

1. compute the integral over the whole domain $\hat{K}$ to get

$$I = \int_{\hat{K}} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

2. compute the integral as a sum of four integrals on subdomains $\hat{K}_i$ (the same as sons of the element)

$$\tilde{I} = \sum_{i=1}^{4} I_i \quad \text{where} \quad I_i = \int_{\hat{K}_i} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

3. if $\dfrac{|I - \tilde{I}|}{|\tilde{I}|} < EPS$ then algorithm returns more accurate value $\tilde{I}$

4. if $\dfrac{|I-\tilde{I}|}{|\tilde{I}|} \geq EPS$ call the adaptive quadrature for the subdomains $\hat{K}_i$

The constant EPS, accuracy of the integral, is given by the user.

Remark 5.3 *The same stategy is used for the second common integral in electromagnetic problems*

$$\int_K \mathrm{curl}\varphi(\boldsymbol{x})\ \mathrm{curl}\psi(\boldsymbol{x})\ d\boldsymbol{x} = \int_{\hat{K}} (J_{\boldsymbol{X}_K})^{-1}\hat{\varphi}(\boldsymbol{\xi})\ (J_{\boldsymbol{X}_K})^{-1}\hat{\psi}(\boldsymbol{\xi})(J_{\boldsymbol{X}_K})\ d\boldsymbol{\xi}, \quad (5.11)$$

where $J_{\boldsymbol{X}_K}$ is the Jacobian of the reference mapping. Finally, we take the maximum of these two orders as the order of inverse reference map to obtain sufficient order to integrate both integrals accurately.

5.5 Numerical example

Let us solve the following problem

$$\nabla \times (\mu_r^{-1}\nabla \times \boldsymbol{E}) - \kappa^2\epsilon_r \boldsymbol{E} = \boldsymbol{F},$$

where $\mu_r = 1, \kappa = \omega/c = 12\pi$, and $\epsilon_r = 1$. It is a simplified model problem of microwave heating, taking place in a square waveguide $\Omega = (-0.125, 0.125)^2$ filled with air, containing a spherical load of radius $r = 0.015625$m and relative permittivity $\epsilon_r = 5.5$ (permittivity of porcelain). The situation is depicted in Fig. 5.7.

In the waveguide, a horizontal wave is generated by time-harmonic current along the edge DA, using the Neumann boundary condition

$$\boldsymbol{n} \times \left(\mu_r^{-1}\nabla \times \boldsymbol{E}\right) = j\omega \boldsymbol{J}_s.$$

We use time-harmonic exciting current $\boldsymbol{J}_s = 10^{-7}$ A. The rest of the boundary is equipped with perfect conductor boundary conditions $\boldsymbol{E} \cdot \boldsymbol{t} = 0$.

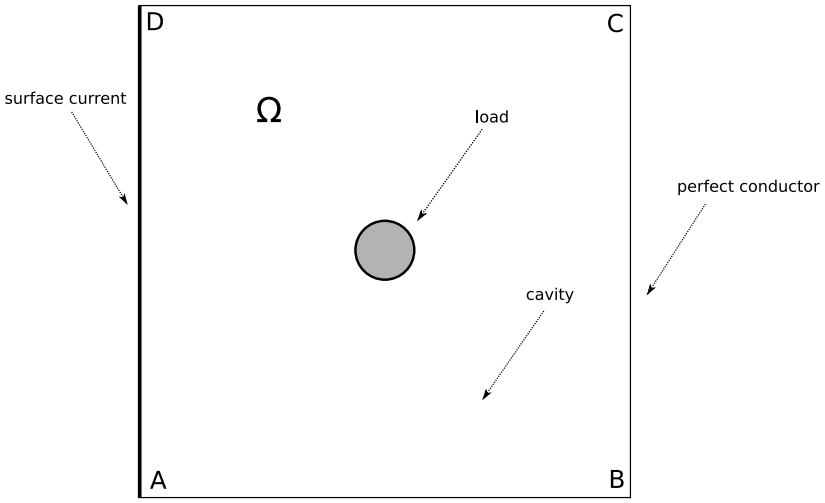

Figure 5.7: Computational domain.

We present a comparison of meshes with curved edges and standard meshes with linear elements. As you can see in Figure 6.14 the load has curved boundary, therefore we would need a lot of linear elements to catch the shape. But the solution is quite smooth there since the edge elements allow discontinuities between elements and therefore the dense mesh is not needed be the solution. The vector field of the solution along with the magnitude is shown in Figure 5.8. In Figure 5.9 we illustrate two triangulations of the domain, one with linear elements and second with curvilinear elements. Linear mesh consists of 732 elements and the curvilinear contains only 88 elements.

Now, we demonstrate that the sparse curvilinear mesh is sufficient partition of the domain to get desired accuracy. All elements in both meshes are equipped with suitable polynomial order and the problem is solved by finite element method. The *hp*-meshes are depicted in Figures 5.10 and 5.11, where the numbers mean the polynomial orders on corresponding elements. The total number of degrees of freedom in the linear mesh is 8569 and obtained relative error of approximation in $H(\mathrm{curl})$-norm is 0.09%. The curvilinear mesh has only 2536 degrees of freedom and resulting relative error is even smaller, 0.05%.

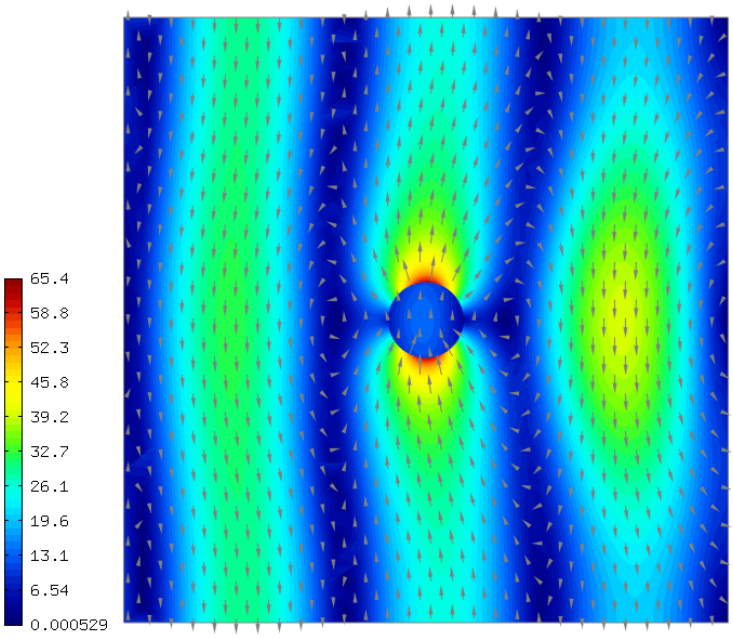

Figure 5.8: Approximation of $\boldsymbol{E}$.

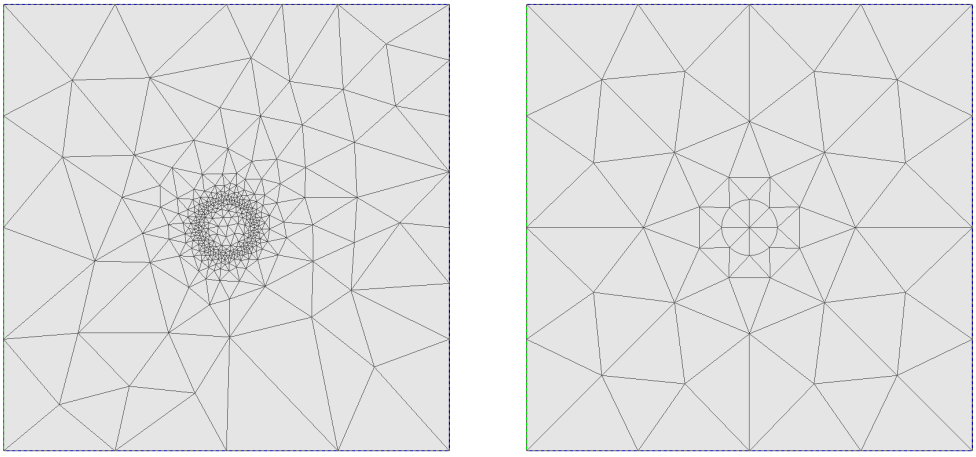

Figure 5.9: Linear and curvilinear mesh.

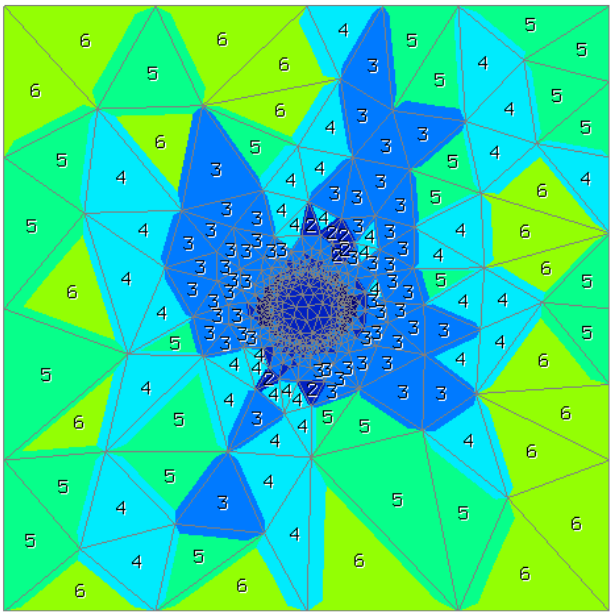

Figure 5.10: Linear mesh together with polynomial orders.

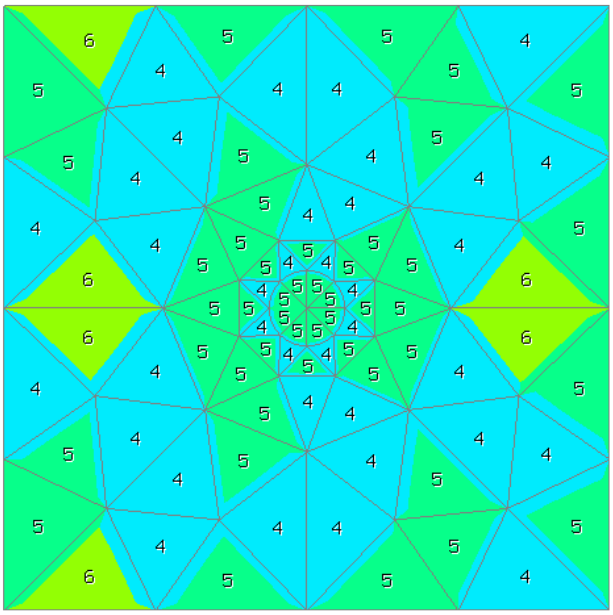

Figure 5.11: Curvilinear mesh together with polynomial orders.