Chapter 3

Higher-order FEM

In Section 2.4 we constructed the Galerkin sequence $V_1 \subset V_2 \ldots$ in the Sobolev space V by refining the mesh of linear elements (h-refinement). Sometimes, much faster convergence can be achieved by increasing the polynomial degree of the elements instead (p-refinement). Such approach is usually more e□cient for elements where the solution is smooth.

3.1 Higher-order finite element space

Let the domain Ω_h be covered with a mesh $\mathcal{T}_{h,p} = \{K_1, K_2, \ldots, K_M\}$ where the elements K_m carry arbitrary polynomial degrees $1 \leq p_m$, $m = 1, 2, \ldots, M$. The space $V(\Omega)$ is now approximated by a picewise-polynomial space $V_{h,p}(\Omega_h)$,

$$V_{h,p} = \{v \in C(\Omega_h);\ v|_{K_m} \in P^{p_m}(K_m) \text{ for all } 1 \leq m \leq M\}$$

where P^p is defined as

$$P^p = \text{span}\{1$$

3.2 Nodal shape functions

The nodal basis is a generalization of the classical linear basis (see Section 2.4), where the expansion coe□cients of the basis or shape functions are obtained by taking the value of the function to be approximated at the vertices of an element. A more general definition is via linear forms:

Definition 3.1 (Nodal finite element) Nodal finite element *is a triad* $\mathcal{K} = (K, P, \Sigma)$, *where*

- K *is a bounded domain in* R^d *with a Lipschitz-continuous boundary,*
- P *is a space of polynomials on* K *of the dimension* $dim(P) = N_P$,
- $\Sigma = \{L_1, L_2, \ldots, L_{N_P}\}$ *is a set of linear forms*

$$L_i : P \to R, \quad i = 1, 2, \ldots, N_P.$$

The elements of Σ *are called* degrees of freedom *(DOF).*

Any polynomial $g \in P^p(K)$ must be uniquely identified by a set of N_P given numbers. This property is called unisolvency:

Definition 3.2 (Unisolvency) *A nodal finite element* (K, P, Σ) *is said to be* unisolvent *if for every polynomial* $g \in P$ *it holds*

$$L_1(g) = L_2(g) = \ldots = L_{N_P}(g) = 0 \quad \Rightarrow \quad g = 0.$$

Lemma 3.1 *Let* (K, P, Σ) *be a unisolvent nodal element. Given any set of numbers* $\{g_1, g_2, \ldots, g_{N_P}\} \in R^{N_P}$, *where* $N_P = dim(P)$, *there exists a unique polynomial* $g \in P$ *such that*

$$L_1(g) = g_1, \; L_2(g) = g_2, \ldots, L_{N_P}(g) = g_{N_P}.$$

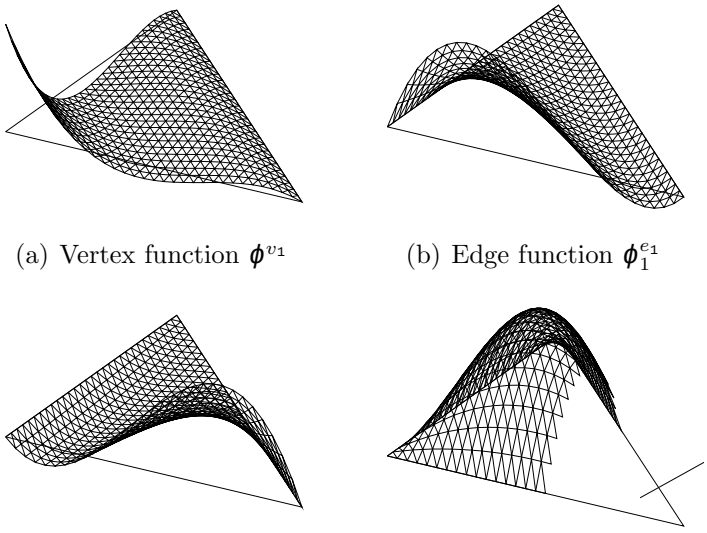
Definition 3.3 (Nodal basis) *Let* (K, P, Σ), $dim(P) = N_P$, *be a nodal finite element. We say that a set of functions* $\mathcal{B} = \{\theta_1, \theta_2, \ldots, \theta_{N_P}\} \subset P$ *is a* nodal basis *of* P *if it satisfies*

$$L_i(\theta_j) = \delta_{ij} \quad \textit{for all } 1 \leq i, j \leq N_P.$$

The functions θ_i *are called* nodal shape functions.

Theorem 3.1 (Characterization of unisolvency) *Consider a nodal finite element* (K, P, Σ), $dim(P) = N_P$. *The finite element is unisolvent if and only if there exists a unique nodal basis* $\mathcal{B} = \{\theta_1, \theta_2, \ldots, \theta_{N_P}\} \subset P$.

The proof of Theorem 3.1 (see [23]) provides an algorithm for the construction of the nodal shape functions. Figure 3.1 is an example of the nodal basis for a cubic element. Note that the functions ϕ^{v_2}, ϕ^{v_3}, $\phi_1^{e_2}$, $\phi_1^{e_3}$, $\phi_2^{e_2}$, $\phi_2^{e_3}$ are not shown and can be obtained from ϕ^{v_1}, $\phi_1^{e_1}$ and $\phi_2^{e_1}$ by rotation.



(a) Vertex function ϕ^{v_1} (b) Edge function $\phi_1^{e_1}$

(c) Edge function $\phi_2^{e_1}$ (d) Bubble function ϕ_1^b

Figure 3.1: Nodal basis of the P^3 element (only 4 of total 10 shape functions are depicted).

Nodal shape functions are advantageous in the sense that the corresponding unknown coe□cients obtained from the solution of the discrete problem directly represent the value of the approximate solution $u_{h,p}$ [illegible]

3.3 Hierarchic shape functions

Hierarchic shape functions are constructed in such a way that the basis $\mathcal{B}^{p+1}$ of the polynomial space $P^{p+1}(K)$ is obtained from the basis $\mathcal{B}^p$ of the polynomial space $P^p(K)$ by adding new shape functions only. This is essential for *p*- and *hp*-adaptive finite element codes since one does not have to change his shape functions completely when increasing the order of polynomial approxi-

The set of Legendre polynomials forms an orthonormal basis of the space $L^2(-1,1)$.

Definition 3.6 (Lobatto functions and kernel functions) *Let us define the functions*

$$\begin{aligned} l_0(x) &= \frac{1-x}{2} \\ l_1(x) &= \frac{x+1}{2} \\ l_k(x) &= \frac{1}{||L_{k-1}||_2}\int_{-1}^{x} L_{k-1}(\xi)\,d\xi, \quad k \geq 2 \end{aligned} \tag{3.2}$$

Since all functions $l_k(x)$, $k \geq 2$ vanish at ± 1, we can define the kernel functions φ_j, $j = 0,1,2,\ldots$, *by decomposing l_k into*

$$l_k(x) = l_0(x)l_1(x)\varphi_{k-2}(x), \quad k \geq 2$$

The edge functions $\phi_j^{e_i}$ can now be written in the form

$$\begin{aligned} \phi_j^{e_1} &= \lambda_2\lambda_3\varphi_{j-2}(\lambda_3-\lambda_2), \quad 2 \leq j \leq p^{e_1} \\ \phi_j^{e_2} &= \lambda_3\lambda_1\varphi_{j-2}(\lambda_1-\lambda_3), \quad 2 \leq j \leq p^{e_2} \\ \phi_j^{e_3} &= \lambda_1\lambda_2\varphi_{j-2}(\lambda_2-\lambda_1), \quad 2 \leq j \leq p^{e_3} \end{aligned} \tag{3.3}$$

Figure 3.2 is an example of a quadratic, cubic and a fourth-order edge function on the edge e_1.

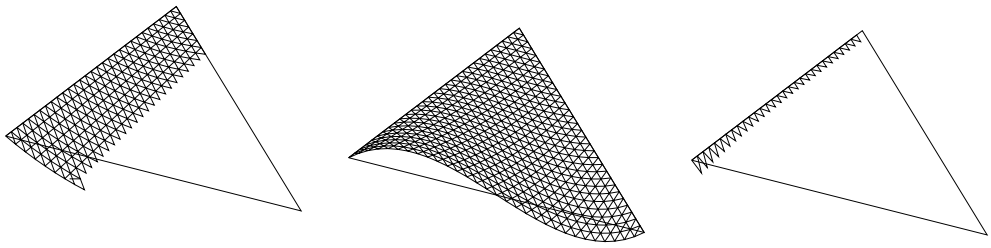


Figure 3.2: Edge shape functions $\phi_2^{e_1}$, $\phi_3^{e_1}$, $\phi_4^{e_1}$.

The numbers $p^{e_1},\ldots,p^{e_3}$ in (3.3) are edge polynomial degrees, as determined by the *minimum rule*:

Definition 3.7 (Minimum rule) *Let K_i, K_j, $1 \leq i,j \leq M$ be two elements sharing a common edge e_k and let p_i, p_j be their associated polynomial degrees. The polynomial degree p^{e_k} of the edge e_k is $p^{e_k} = \min\{p_i, p_j\}$.*

The minimum rule guarantees that the resulting piece-wise polynomial space $V_{h,p}$ will be independent of a concrete choice of the shape functions.

The hierarchic basis will be completed by defining the bubble functions. A standard approach is to combine a□ne coordinates with varying powers,

$$\phi^b_{n_1,n_2} = \lambda_1(\lambda_2)^{n_1}(\lambda_3)^{n_2}, \quad 1 \leq n_1, n_2;\ n_1 + n_2 \leq p_m - 1 \tag{3.4}$$

However, conditioning properties of these functions are relatively bad. This motivates us to define a di□erent set of bubble functions by incorporating the kernel functions:

$$\phi^b_{n_1,n_2} = \lambda_1\lambda_2\lambda_3\varphi_{n_1-1}(\lambda_3 - \lambda_2)\varphi_{n_2-1}(\lambda_2 - \lambda_1), \tag{3.5}$$

where n_1 and n_2 satisfy the same conditions as in (3.4). Figures 3.3 and 3.4 are examples of such shape functions.

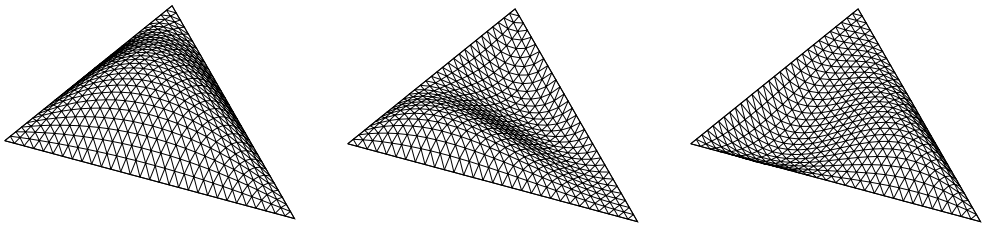


Figure 3.3: Cubic bubble function $\phi^b_{1,1}$ and fourth-order functions $\phi^b_{1,2}$, $\phi^b_{2,1}$.

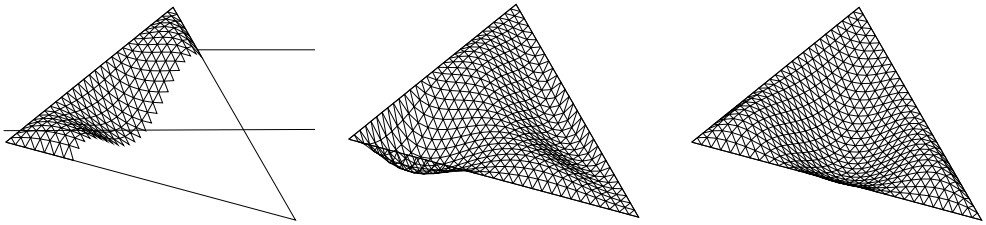


Figure 3.4: Fifth-order bubble functions $\phi^b_{1,3}$, $\phi^b_{3,1}$ and $\phi^b_{2,2}$.

Proposition 3.1 *The shape functions (3.1), (3.3) and (3.5) constitute a hierarchic basis of the space* $P^{p_m}(K_t)$

Proof The complete proof can be found in [25]. Briefly, the following must be verified:

1. Are all the shape functions linearly independent?
2. Do they all belong to the space $P^{p_m}(K_t)$?
3. Matches their number the dimension of the space $P^{p_m}(K_t)$?

As has been mentioned, the choice of the shape functions has a strong influence on the conditioning of the stiffness matrix associated with the discrete problem. The described bubble functions can still be improved, e.g., by their orthogonalization. Even though the orthogonality is not preserved after the shape functions are transformed from the reference domain to the actual element, experiments show that the condition number of the stiffness matrix is significantly lower nevertheless (see [27]). In our software we have been using these orthonormal bubble functions.

Let us conclude this section with Table 3.1, showing the numbers of shape functions depending on the polynomial degree p^m of the element.

Shape type	Existence	Number of shape functions
Vertex	always	3
Edge	$p^{e_i} \geq 2$	$\sum_{i=1}^{3}(p^{e_i}-1)$
Bubble	$p^m \geq 3$	$(p^m-1)(p^m-2)/2$

Table 3.1: Numbers of shape functions.

3.4 Nodes and connectivity arrays

In standard linear FEM, mesh vertices are simply numbered, ie. assigned vertex basis function indices. Due to the existence of edge and bubble basis functions in higher-order FEM, a more elaborate bookkeeping is required. Data structures that link the shape functions $\phi_1, \phi_2, \phi_3, \ldots$ on an element to the basis functions $v_i, v_j, v_k, \ldots \subset V_{h,p}$ are called *connectivity arrays*. These arrays ensure that:

- vertex shape functions are joined properly to form vertex basis functions, ie. that the appropriate vertex shape functions on elements sharing a common vertex are assigned the same vertex basis function index;
- edge shape functions are joined properly to form edge basis functions, ie. that the appropriate edge shape functions on elements sharing a common edge are assigned the same edge basis function index.

This is usually achieved by data structures called $nodes$. Three types of nodes are defined: $vertex\ nodes$, associated with vertices, $edge\ nodes$, associated with edges and $bubble\ nodes$, associated with element interiors. Each node holds one or more basis function indices. Each element is linked with seven nodes corresponding to its vertices, edges and interior during initialization.

3.5 Higher-order numerical quadrature

Most commonly, the integrals in (2.12), (2.13) are evaluated numerically by the $Gaussian\ quadrature$. The k-point Gaussian quadrature rule on the domain K_t has the form

$$\int_{K_t} g(\xi)\mathrm{d}\xi \approx \sum_{i=1}^{k} w_{k,i} g(\xi_{k,i}) \tag{3.6}$$

where g is a bounded continuous function, $\xi_{k,i} \in K_t, i = 1, 2, \ldots, k$ are the integration points and $w_{k,i} \in R$ are the integration weights. The sum of the integration weights must be equal to the area of K_t, so that the rule (3.6) is exact for constants. If the points and weights are chosen carefully, the formula (3.6) can be exact for polynomials up to a certain degree $q > 0$.

In 1D the integration points are roots of the Legendre polynomials. Also in 2D it is quite easy to find the integration points and weights for low-order polynomials ocurring in traditional linear FEM. For higher-order polynomials, however, the task of finding optimal Gaussian quadrature rules presents a complex non-linear optimization problem with many open questions left. Optimal integration points and weights are known on K_t for polynomials up to degree 10. Suboptimal (with more points than necessary) rules have been found for polynomials up to degree 20. Complete integration tables along with more information on this subject can be found in [25].

3.6 Assembling procedure

Recall from Chapter 2 that after converting the PDE (2.1) to the weak form (2.7), we can use the Galerkin method to obtain the discrete problem

$$\boldsymbol{S}\boldsymbol{Y} = \boldsymbol{F},$$

where $\boldsymbol{Y}$ is the vector of uknowns, and the stiffness matrix $\boldsymbol{S}$ and the load vector $\boldsymbol{F}$ are defined as

$$\boldsymbol{S} = \{\ a(v_j, v_i)\ \}_{i,j=1}^{N},$$

$$\boldsymbol{F} = \{\ l(v_i)\ \}_{i=1}^{N}.$$

Here $v_1, v_2, \ldots, v_N$ are the basis function of the finite-dimensional space $V(\Omega)$, $\dim(V(\Omega)) = N$ and $a(u, v)$, $l(v)$ are the bilinear and linear forms, respectively. The construction of $\boldsymbol{S}$ and $\boldsymbol{F}$ is called *assembling*.

A naive implementation of the assembling procedure might read as follows:

```
S = 0_{N×N}
for (i = 1...N)
  for (j = 1...N)
    for (k = 1...M)
      S_ij = S_ij + a(v_j, v_i)|K_k
```

The innermost statement is summing contributions of $a(v_j, v_i)$ restricted to each element K_k, M is the total number of elements. The first flaw of this algorithm is that it ignores the *sparsity* of $\boldsymbol{S}$, since $\boldsymbol{S}$ only contains $O(N)$ nonzero elements. This is due to the fact that the supports of v_i, $1 \leq i \leq N$,

$$\mathrm{supp}(v_i) = \overline{\{\boldsymbol{x} \in \Omega;\ v_i(\boldsymbol{x}) \neq 0\}} \subset \Omega$$

are small and mostly disjoint. The element $\boldsymbol{S}_{ij} = a(v_j, v_i)$ is nonzero if and only if

$$\mathrm{meas}_{2D}(\mathrm{supp}(v_i) \cap \mathrm{supp}(v_j)) > 0.$$

This is easy to see, since the bilinear form $a(u, v)$ is zero whenever u or v is zero. The zero elements of $\boldsymbol{S}$ should not be stored at all to avoid $O(N^2)$ memory complexity and only the nonzero elements should be evaluated:

```
for (i = 1...N)
  for (j = 1...N)
```

```
for (k = 1...M)
  if (meas_2D(supp(v_i) ∩ K_k) > 0 and meas_2D(supp(v_j) ∩ K_k) > 0)
    S_ij = S_ij + a(v_j, v_i)|K_k
```

However, this algorithm still has about $O(MN^2)$ time complexity. This is easily remedied by rearranging the cycles and utilizing the element connectivity arrays (Section 3.4):

```
for (k = 1...M)
  L = list of basis function indices whose supp(v_i) ∩ K_k ≠ ∅
  for (all i ∈ L)
    for (all j ∈ L)
      S_ij = S_ij + a(v_j, v_i)|K_k
```

The size of the list L is $O(p^2)$, where p is the polynomial degree of the elements (see Table 3.1), therefore the total time required to assemble $\boldsymbol{S}$ is $O(Mp^4)$. The last algorithm is called the *element-by-element assembling procedure*.