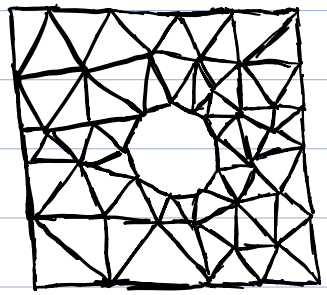
2D Heat Conduction: Affine Triangular Finite Element Approximations:

Now that we have presented the general form of a Galerkin approximation for the weak form of steady two-dimensional heat conduction, we turn to the construction of Galerkin finite element approximations. We begin with a discussion of affine triangular finite element approximations.

The cornerstone of the construction of an affine triangular finite element approximation is the creation of a suitable triangular mesh. Generally speaking, a two-dimensional mesh is a set of non-overlapping open subsets of $\mathbb{R}^2$ referred to as elements. A triangular mesh then simply is a two-dimensional mesh comprised of triangular elements, e.g.:



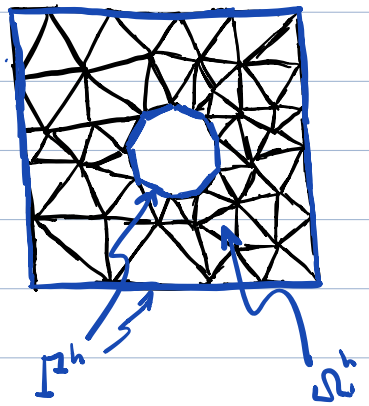
Assuming there are n_{el} elements in the mesh, each denoted Ω_h^e for $e = 1, \ldots, n_{el}$, we denote a triangular mesh as:

$$\mathcal{M}^h := \{\Omega_h^e\}_{e=1}^{n_{el}}$$

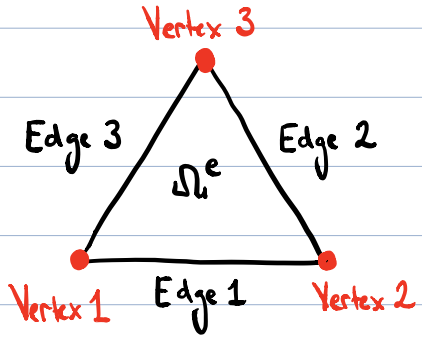
There is a domain implicitly defined by the mesh:

$$\Omega_h^h := \text{int}\left(\overline{\bigcup_{e=1}^{n_{el}} \Omega_h^e}\right)$$

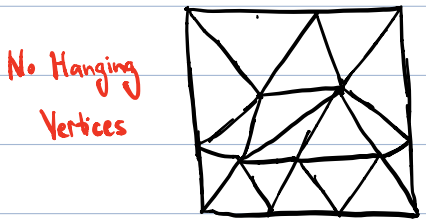
with boundary Γ_h, e.g.:



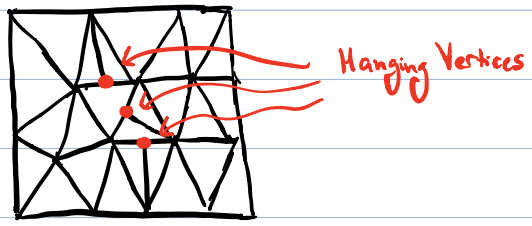
Each element has three edges and three vertices:



We take the edges to be open in the sense that they not contain their endpoints. A mesh is said to be conforming if no element vertices lie on the edge of another element. Visually:



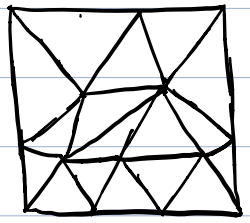
Example of a Conforming Mesh



Example of a Non-Conforming Mesh

As illustrated above, element vertices that lie on another element's edge are called hanging vertices. Thus a conforming mesh is a mesh free of hanging vertices. We are exclusively interested in conforming meshes in this class.

For a conforming mesh, each element edge either belongs to the mesh boundary or is also an edge of a neighboring element. We denote the total number of unique edges as n_{edge}, and we denote the total number of unique edges on the mesh boundary as n_{elb}. We also denote the total number of unique vertices as n_{vertex}. For the below mesh, $n_{el} = 17$, $n_{edge} = 30$, $n_{vertex} = 14$, and $n_{elb} = 9$:



For a conforming mesh, Euler's formula holds:

$$n_{el} - n_{edge} + n_{vertex} = \text{Number of Connected Components}$$

We collect the boundary edges, denoted Γ^e for $e = 1, \ldots, n_{elb}$, into a boundary mesh:

$$\mathcal{E}_\Gamma^h := \{\Gamma^e\}_{e=1}^{n_{elb}}$$

Note that:

$$\Gamma^h = \overline{\bigcup_{f=1}^{n_{elb}} \Gamma^e}$$

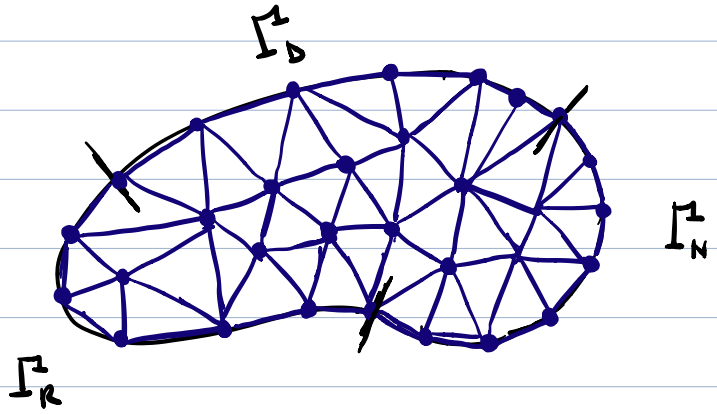
so the boundary of the mesh may be thought of as the domain associated with the boundary mesh.

In an affine triangular finite element approximation of the steady two-dimensional heat conduction problem, a triangular mesh is employed to approximate the domain of interest. In this context, the mesh is referred to as the finite element mesh, and the domain associated with the mesh is referred to as the finite element domain. The following criteria must be satisfied for a given mesh to be considered a finite element mesh:

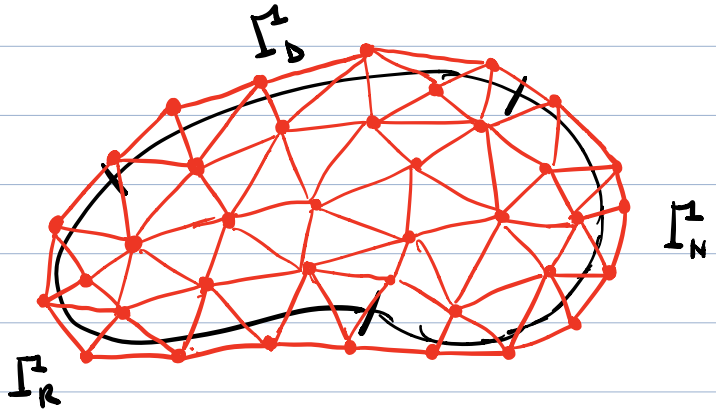
Criterion 1: Each vertex must lie either inside the domain of interest or on the boundary of the domain of interest.

Criterion 2: Each vertex on the mesh boundary must lie on the boundary of the domain of interest. Moreover, the ends of each boundary edge must lie either in the closure of the Dirichlet boundary, the closure of the Neumann boundary, or the closure of the Robin boundary.

Visually:

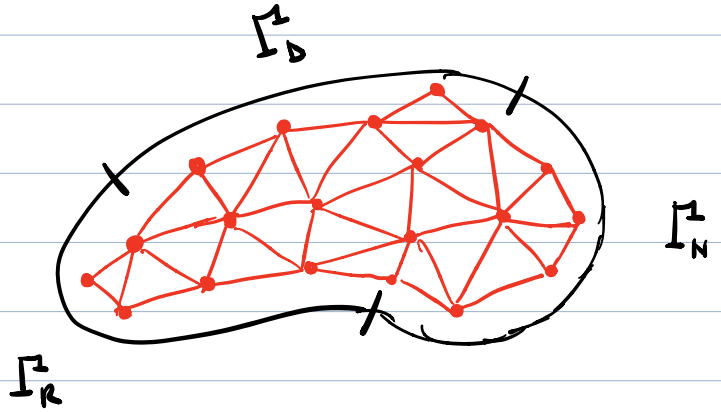


A mesh satisfying both Criterion 1 and Criterion 2



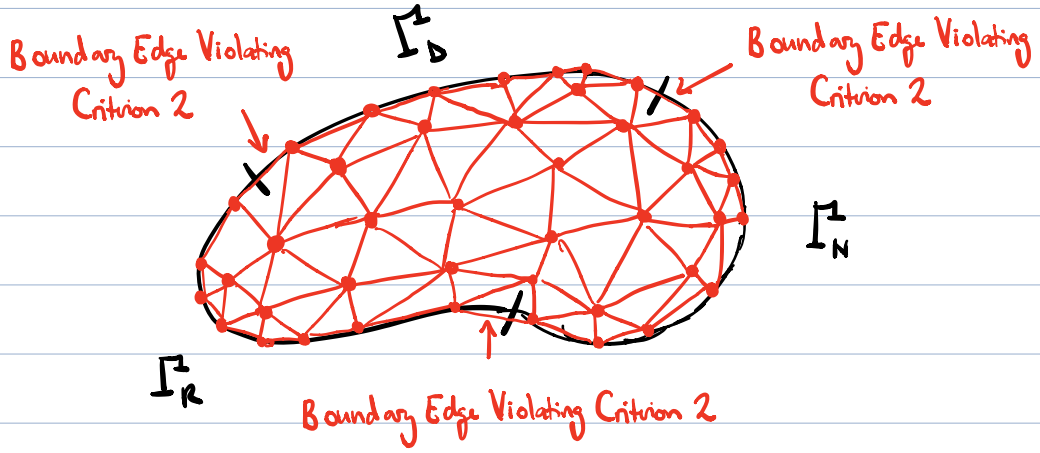
A mesh satisfying neither Criterion 1 nor Criterion 2

Here, some mesh vertices lie completely outside the domain of interest, so Criterion 1 is violated. Vertices on the mesh boundary also do not lie on the domain boundary, so Criterion 2 is violated.



A mesh satisfying Criterion 1 but not Criterion 2

Here, vertices on the mesh boundary do not lie on the domain boundary, so Criterion 2 is violated.



A mesh satisfying Criterion 1 but not Criterion 2

Here, vertices on the mesh boundary do lie on the domain boundary. However, some boundary edges violate the second requirement of Criterion 2.

If Condition 2 is satisfied, then each boundary edge $\Gamma^e \in \mathcal{E}^h_\Gamma$ satisfies either $\partial\Gamma^e \subset \overline{\Gamma}_D$, $\partial\Gamma^e \subset \overline{\Gamma}_N$, or $\partial\Gamma^e \subset \overline{\Gamma}_R$.
Thus we can create a Dirichlet boundary mesh:

$$\mathcal{E}^h_D := \left\{ \Gamma^e \in \mathcal{E}^h_\Gamma : \partial\Gamma^e \subset \overline{\Gamma}_D \right\}$$

a Neumann boundary mesh:

$$\mathcal{E}^h_N := \left\{ \Gamma^e \in \mathcal{E}^h_\Gamma : \partial\Gamma^e \subset \overline{\Gamma}_N \right\}$$

and a Robin boundary mesh:

$$\mathcal{E}^h_R := \left\{ \Gamma^e \in \mathcal{E}^h_\Gamma : \partial\Gamma^e \subset \overline{\Gamma}_R \right\}$$

such that:

$$\mathcal{E}^h_\Gamma = \mathcal{E}^h_D \cup \mathcal{E}^h_N \cup \mathcal{E}^h_R$$

and:

$$\mathcal{E}^h_D \cap \mathcal{E}^h_N = \emptyset, \quad \mathcal{E}^h_N \cap \mathcal{E}^h_R = \emptyset, \quad \mathcal{E}^h_R \cap \mathcal{E}^h_D = \emptyset$$

That is, each boundary edge belongs to either the Dirichlet boundary mesh, the Neumann boundary mesh, or the Robin boundary mesh.
We can also create a Dirichlet finite element boundary:

$$\Gamma^h_D := \operatorname{int}\left(\overline{\bigcup_{\Gamma^e \in \mathcal{E}^h_D} \Gamma^e} \right)$$

a Neumann finite element boundary:

$$\Gamma^h_N := \operatorname{int}\left(\overline{\bigcup_{\Gamma^e \in \mathcal{E}^h_N} \Gamma^e} \right)$$

and a Robin finite element boundary:

$$\Gamma_R^h := \text{int}\left(\overline{\bigcup_{\Gamma^e \in \mathcal{E}_R^h} \Gamma^e}\right)$$

such that:

$$\Gamma^h := \overline{\Gamma_D^h \cup \Gamma_N^h \cup \Gamma_R^h}$$

and:

$$\Gamma_D^h \cap \Gamma_N^h = \emptyset, \quad \Gamma_N^h \cap \Gamma_R^h = \emptyset, \quad \Gamma_R^h \cap \Gamma_D^h = \emptyset$$

Note that, in general, it is impossible to exactly represent the domain of interest using a triangular mesh, so generally:

$$\Omega^h \approx \Omega$$

$$\Gamma^h \approx \Gamma$$

$$\Gamma_D^h \approx \Gamma_D$$

$$\Gamma_N^h \approx \Gamma_N$$

$$\Gamma_R^h \approx \Gamma_R$$

Thus, with an affine triangular finite element approximation, we not only approximate the set of trial solutions and space of test functions using finite-dimensional approximation spaces, we also approximate the domain geometry. As a consequence, an affine triangular finite element

approximation is not a true Galerkin approximation unless $\Omega_h^h = \Omega_h$, and a priori error estimates which hold for Galerkin approximations do not necessarily hold for an affine triangular finite element approximation. Instead, the finite element solution is polluted by the geometry approximation. Fortunately, as we refine the mesh, we expect that $\Omega_h^h \to \Omega$, and thus we expect the finite element solution to converge to the exact solution under mesh refinement. As it turns out, we can characterize the finite element error for any given finite element mesh using Strang's lemma. Strang's lemma accounts for both the error induced by approximating the set of trial solutions and space of test functions using finite element approximation spaces as well as the error induced by approximating the domain geometry. Strang's lemma actually reveals that geometry error actually limits the convergence rate of affine triangular finite element approximation to second-order in the H^1-norm, which will inspire us later to turn to so-called isoparametric finite element approximations.

The approximation of the domain geometry by a finite element mesh is an example of a so-called "variational crime". Variational crimes are approximations made that result in a finite element approximation not being a true Galerkin approximation. Other variational crimes include the use of numerical quadrature for approximating integrals, approximation of boundary conditions by surrogate boundary conditions, and even finite precision arithmetic. It turns out Strang's lemma can be used to characterize errors due to each of these variational crimes.

The process of creating a finite element mesh is called mesh generation, and the process of creating a finite element mesh in an automated manner given the domain geometry is called automatic mesh generation. Automated generation of triangular finite element meshes is a technical but solved problem. Most automatic triangular mesh generation algorithms are composed of two steps:

Step 1: Generation of a polygonal mesh of the boundary of the domain.

Step 2: Generation of a triangular finite element mesh whose boundary coincides with the polygonal mesh generated in Step 1.

Both Steps 1 and 2 are typically guided using mesh size functions that dictate how small elements should be in different portions of the domain. Step 2 is typically executed by first populating the domain with vertices and then using constrained Delaunay triangulation to construct a suitable finite element mesh. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation. Thus, they tend to avoid sliver triangles, which are often undesirable in finite element analysis. However, in certain applications, such as finite element analysis of high Reynolds flows, sliver triangles are desired in regions where the solution field has an

anisotropic character, such as in boundary layers. In this class, we will employ the open-source mesh generator gmsh to construct triangular meshes.

With the notion of a triangular finite element mesh in hand, we define finite element functions on such a mesh as piecewise polynomial functions with respect to the mesh. As we require trial solutions and test functions that are square integrable with square integrable first derivatives, we are specifically interested in finite element functions that are also square integrable with square integrable with square integrable first derivatives. The following result dictates the required continuity between adjacent elements to meet such a requirement:

Theorem: A piecewise infinitely differentiable function $v : \bar{\Omega}^h \to \mathbb{R}$ over the mesh $\mathcal{M}^h$ belongs to $H^k(\Omega^h)$ if and only if $v \in C^{k-1}(\bar{\Omega}^h)$.

That is, a finite element function of polynomial degree k is in $H^1(\Omega)$ if and only if it is a member of the following finite element approximation space:

$$\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h) := \left\{ v^h \in C^0(\bar{\Omega}^h) : v^h|_{\Omega^e} \in \mathcal{P}^k(\Omega^e),\ e = 1, \dots, nel \right\}$$

where $\mathcal{P}^k(\Omega^e)$ is the space of polynomials of degree k over the e^{th} element in the finite element mesh. Just as in the 1D setting, $\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$ consists only of globally constant functions when $k=0$, so we are interested in the case when $k \geq 1$.

Given a proper space of finite element functions to work with, we are now in a position to define a corresponding space of test functions:

$$\mathcal{V}^h := \left\{ v^h \in \mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h) : v^h|_{\Gamma_D^h} = 0 \right\}$$

Defining a corresponding set of trial solutions, however, is more challenging. One might be inclined to define:

$$\mathcal{S}^h := \left\{ v^h \in \mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h) : v^h|_{\Gamma_D^h} = g \right\}$$

However, when $\Gamma_D^h \neq \Gamma_D$, the above set is not well-defined since g is defined on Γ_D, not Γ_D^h. More concerning is the fact that g is, in general, not piecewise polynomial. Thus, even when $\Gamma_D^h = \Gamma_D$, the above set is empty for general specifications of g. To deal with this issue, we need to replace the exact Dirichlet boundary condition with a surrogate Dirichlet finite element boundary condition defined on the Dirichlet finite element boundary. This procedure, as we will discover, relies on the use of Lagrange basis functions and nodal degrees of freedom.

To construct a basis for $\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$, and hence a basis for the space of test functions $\mathcal{V}^h$, we first need to compute the dimension of $\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$. To do this computation, we begin by computing the dimension of $\mathcal{P}^k(\Omega^e)$ for an arbitrary triangular element Ω^e.

For $k=1$, an arbitrary function of $\mathcal{P}^k(\Omega^e)$ takes the form:

$$v^h(\vec{x}) = c_1 + c_2 x_1 + c_3 x_2$$

where c_1, c_2, c_3 are arbitrary constants. Thus:

$$\dim\left(\mathcal{P}^1(\Omega^e)\right) = 3$$

For $k=2$, an arbitrary function of $\mathcal{P}^k(\Omega^e)$ takes the form:

$$v^h(\vec{x}) = c_1 + c_2 x_1 + c_3 x_2 + c_4 x_1^2 + c_5 x_1 x_2 + c_6 x_2^2$$

where $\{c_a\}_{a=1}^{6}$ are arbitrary constants. Thus:

$$\dim\left(\mathcal{P}^2(\Omega^e)\right) = 6$$

For $k=3$, an arbitrary function of $\mathcal{P}^k(\Omega^e)$ takes the form:

$$v^h(\vec{x}) = c_1 + c_2 x_1 + c_3 x_2 + c_4 x_1^2 + c_5 x_1 x_2 + c_6 x_2^2 + c_7 x_1^3 + c_8 x_1^2 x_2 + c_9 x_1 x_2^2 + c_{10} x_2^3$$

where $\{c_a\}_{a=1}^{10}$ are arbitrary constants. Thus:

$$\dim\left(\mathcal{P}^3(\Omega^e)\right) = 10$$

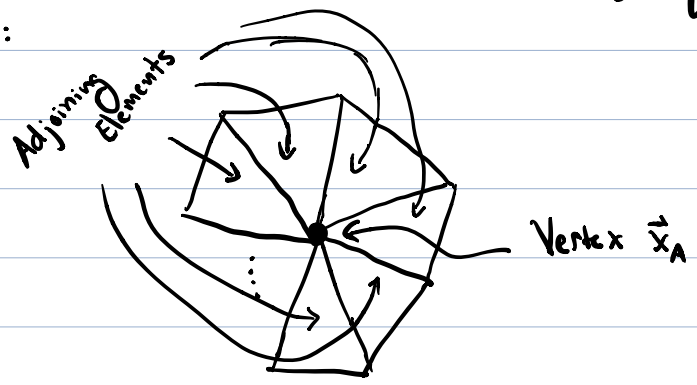
For arbitrary k, it can be shown that:

$$\dim\left(\mathcal{P}^k(\Omega^e)\right) = \frac{(k+2)(k+1)}{2}$$

Thus, the dimension of the space of discontinuous piecewise polynomials of degree k over a triangular finite element mesh with n_{el} elements is:

$$n_{el} * \frac{(k+2)(k+1)}{2}$$

To compute the dimension of the space of continuous piecewise polynomials of degree k over the same mesh, we can simply subtract the number of constraints required for enforcing continuity. First, at each vertex, we require that the finite element function in each adjoining element be equal:



Suppose there are n_{adj} adjoining elements and the finite element functions over these elements are $v_1^h, \ldots, v_{adj}^h$. Then we have the $(n_{adj}-1)$ constraints:

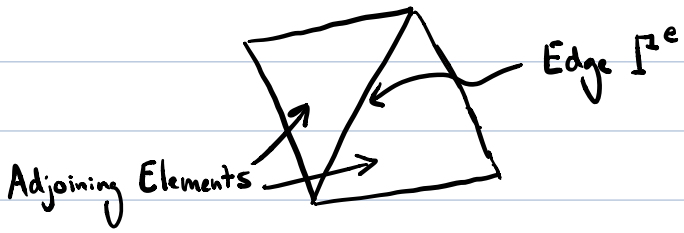
$$v_1^h(\vec{x}_A) = v_2^h(\vec{x}_A) = \cdots = v_{adj}^h(\vec{x}_A)$$

Each element has three vertices, so it is an adjoining element for three different vertices. Thus if we sum the number of constraints

associated with continuity at vertices, we attain:

$$\text{Number of Vertex Constraints} = 3 * n_{el} - n_{vertex}$$

Along each interior edge, we also require that the finite element solution in each adjoining element be equal:



Suppose the finite element functions over the two adjoining elements are v_1^h and v_2^h. Then we require:

$$v_1^h|_{\Gamma^e} = v_2^h|_{\Gamma^e}$$

Both v_1^h and v_2^h are univariate polynomials of degree k over Γ^e, so they each have k+1 degrees of freedom over the edge. Thus the $v_1^h|_{\Gamma^e} = v_2^h|_{\Gamma^e}$ if the degrees of freedom are set to be equal, so we have k+1 constraints. However, two of these constraints are associated with the endpoints of the edge which are vertices, so we only have k-1 constraints after applying vertex constraints. Thus if we sum the number of constraints associated with continuity along interior edges, we attain:

$$\text{Number of Edge Constraints} = (k-1) * (\text{Number of Interior Edges})$$

Now, each element has three edges, and each interior edge is shared by two elements while each boundary edge is associated with a single element. It follows then that:

$$\begin{aligned} 3 * n_{el} &= 2 * (\text{Number of Interior Edges}) \\ &\quad + 1 * (\text{Number of Boundary Edges}) \\ &= n_{edge} + \text{Number of Interior Edges} \end{aligned}$$

So:

$$\text{Number of Edge Constraints} = 3 * (k-1) * n_{el} - (k-1) * n_{edge}$$

Therefore, we have:

$$\begin{aligned} \dim\left(P^k_{cont.}(\mathcal{M}^h)\right) &= n_{el} * \frac{(k+2)(k+1)}{2} \\ &\quad - \text{Number of Vertex Constraints} \\ &\quad - \text{Number of Edge Constraints} \\ &= n_{el} * \frac{(k+2)(k+1)}{2} \\ &\quad - (3 * n_{el} - n_{vertex}) \\ &\quad - (3 * (k-1) * n_{el} - (k-1) * n_{edge}) \end{aligned}$$

or after re-arranging:

$$\dim\left(\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)\right) = n_{el} * \frac{(k-1)(k-2)}{2} + n_{edge} * (k-1) + n_{vertex}$$

Now that we know the dimension of $\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$, we can construct a basis for $\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$. We follow the same procedure as we did in the 1D setting, namely, we will specify degrees of freedom then infer the corresponding basis. In this direction, note each finite element function $v^h \in \mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$ takes the form:

$$v^h(\vec{x}) = \sum_{A=1}^{n_{nod}} v^h_A \, N_A(\vec{x})$$

v^h_A: Degree of Freedom; $N_A(\vec{x})$: Basis Function

Where:

$$n_{nod} := \dim\left(\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)\right)$$

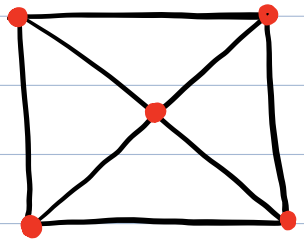
We choose the degrees of freedom as the values of v^h at chosen nodes in our finite element domain:

$$v^h_A = v^h(\vec{x}_A)$$

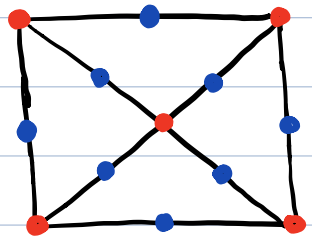
Following our dimension formula for $\mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$, we place a node at each vertex, $k-1$ nodes along each edge, and $\frac{(k-1)(k-2)}{2}$ nodes

within each element:

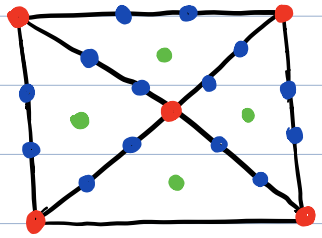
k=1



k=2



k=3



● = Vertex Node

● = Edge Node

● = Element Node

Many choices of node locations are valid, but it is most common to place edge nodes and element nodes so all of the nodes on a given edge or element are equi-spaced. We will assume this placement for the remainder of this class.

Now, with degrees of freedom in hand, we can infer the basis. By our selection of degrees of freedom, we have:

$$N_A(\vec{x}_B) = \begin{cases} 1 & \text{if } A = B \\ 0 & \text{otherwise} \end{cases}$$

for $A, B = 1, \ldots, n_{nod}$. If we isolate our view to a single element, each basis function becomes a polynomial subject to constraints applied at the vertex, edge, and element nodes. Since:

$$\dim\left(\mathcal{P}^k(\Omega^e)\right) = \frac{(k+2)(k+1)}{2}$$

and:

$$\text{Number of Equations} = \underbrace{3}_{\#\text{ of Vertex Constraints}} + \underbrace{3*(k-1)}_{\#\text{ of Edge Constraints}} + \underbrace{\frac{(k-1)(k-2)}{2}}_{\#\text{ of Element Constraints}}$$

$$= \frac{(k+2)(k+1)}{2}$$

We have an equal number of unknown degrees of freedom and equations for each basis function. Thus we can exactly solve for the form of each basis function over each element in the finite element mesh. The results of these calculations are rather ugly, so they are not displayed here. However, it turns out we can arrive at greatly simplified expressions for the basis functions if we express them in terms of shape functions defined over a parent triangular element. We will return to this discussion momentarily.

The basis functions $\{N_A\}_{A=1}^{n_{nod}}$ corresponding to the degrees of freedom $\{v^h(\vec{x}_A)\}_{A=1}^{n_{nod}}$ are referred to as Lagrange basis functions

while the degrees of freedom are referred to as nodal degrees of freedom. Lagrange basis functions over a triangular finite element mesh satisfy the same properties over a one-dimensional finite element mesh:

1. They are interpolatory. That is:

$$N_A(\vec{x}_B) = \begin{cases} 1 & \text{if } A = B \\ 0 & \text{if } A \neq B \end{cases}$$

2. They are locally supported. In particular:

$$N_A(\vec{x}) = 0 \quad \text{if } \vec{x} \in \bar{\Omega}^e \text{ and } \vec{x}_A \notin \bar{\Omega}^e$$

3. They form a partition of unity. That is:

$$\sum_{A=1}^{n_{nod}} N_A(\vec{x}) = 1 \quad \text{for } \vec{x} \in \Omega$$

4. They are non-negative for $k=1$. That is:

$$N_A(\vec{x}) \geq 0 \quad \text{for } \vec{x} \in \Omega$$

if $k=1$.

Lagrange basis functions are not the only choice of basis functions for a triangular finite element mesh. Other choices include hierarchical

and Bernstein basis functions. However, Lagrange basis functions are arguably the most convenient from an implementation point of view, so we will stick with them here.

Now let:

$$\eta := \{1, 2, \ldots, n_{nod}\}$$

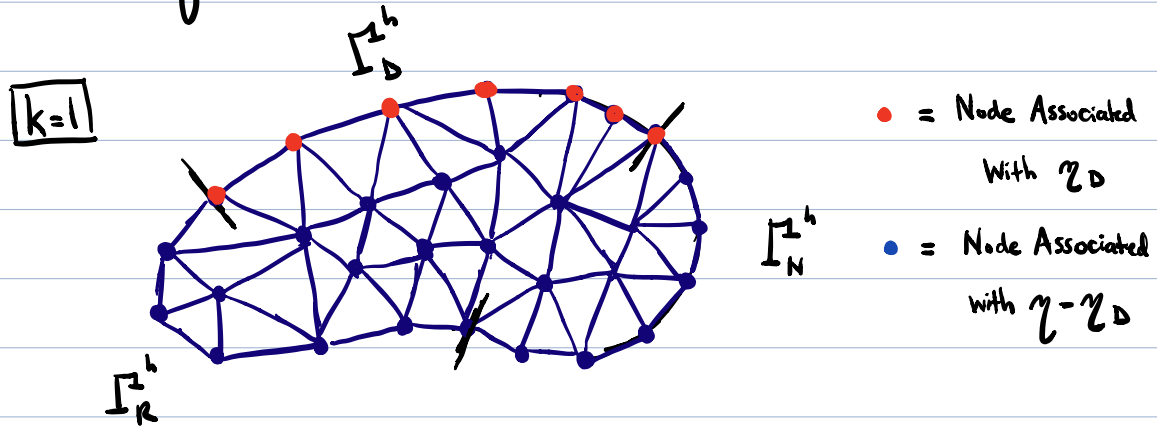
be the set of node numbers and:

$$\eta_D := \{A \in \eta : \vec{x}_A \in \overline{\Gamma}_D^h\}$$

be the subset of node numbers associated with nodes located on the closure of the Dirichlet finite element boundary. The subset of node numbers not associated with nodes located on the closure of the Dirichlet finite element boundary is then:

$$\eta - \eta_D = \{A \in \eta : A \notin \eta_D\}$$

Visually:



A finite element function $v^h \in \mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h)$ is then in $\mathcal{V}^h$ if and only if:

$$v^h(\vec{x}_A) = 0 \qquad \text{for } A \in \eta_D$$

Thus an equivalent definition of the space of finite element test functions is:

$$\mathcal{V}^h := \left\{ v^h \in \mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h) : v^h(\vec{x}_A) = 0 \quad \text{for } A \in \eta_D \right\}$$

The above definition inspires the following definition of the set of finite element trial solutions, provided the domain of definition of g may be expanded to include all nodes on Γ_D^h (note g is already defined at vertex nodes since vertex nodes are on Γ_D):

$$\mathcal{S}^h := \left\{ v^h \in \mathcal{P}^k_{\text{cont.}}(\mathcal{M}^h) : v^h(\vec{x}_A) = g(\vec{x}_A) \text{ for } A \in \eta_D \right\}$$

As opposed to the previously proposed definition for $\mathcal{S}^h$, the above definition is well-posed and leads to a nonempty set of finite element trial solutions.

Note that any function $v^h \in \mathcal{V}^h$ takes the form:

$$v^h(\vec{x}) = \sum_{A \in \eta} v^h(\vec{x}_A) N_A(\vec{x})$$

$$= \sum_{A \in \eta - \eta_D} v^h(\vec{x}_A) N_A(\vec{x}) + \sum_{A \in \eta_D} v^h(\vec{x}_A) N_A(\vec{x})$$

Free

0 by BC

Thus $\{N_A\}_{A \in \eta - \eta_D}$ is a basis for $\mathcal{V}^h$. Moreover, any function $v^h \in \mathcal{S}^h$ takes the form:

$$v^h(\vec{x}) = \sum_{A \in \eta}{}' v^h(\vec{x}_A) N_A(\vec{x})$$

$$= \underbrace{\sum_{A \in \eta - \eta_D}{}' v^h(\vec{x}_A) N_A(\vec{x})}_{\in \mathcal{V}^h} + \underbrace{\sum_{A \in \eta_D}{}' v^h(\vec{x}_A) N_A(\vec{x})}_{\in \mathcal{S}^h}$$

($v^h(\vec{x}_A)$: Free; $g(\vec{x}_A)$: by BC)

Defining:

$$g^h(\vec{x}) = \sum_{A \in \eta_D}{}' g(\vec{x}_A) N_A(\vec{x})$$

we thus have $\mathcal{S}^h = \mathcal{V}^h + g^h$. Note this equivalence makes it clear that the original boundary condition:

$$T = g \quad \text{on } \Gamma_D^2$$

has been replaced by the surrogate finite element boundary condition:

$$T^h = g^h = \sum_{A \in \eta_D}{}' g(\vec{x}_A) N_A(\vec{x}) \quad \text{on } \Gamma_D^{2h}$$

With finite element trial solutions and test functions in hand, we are now in a position to state the form of a Galerkin affine triangular finite element

approximation of steady two-dimensional heat conduction. Provided the domains of definition of f, h, β, and T_R can be extended to Ω^h, Γ_N^h, Γ_R^h, and Γ_R^h respectively, we simply replace the space of test functions and set of trial solutions appearing in the Galerkin approximation presented in the last lecture with their finite element counterparts presented in this lecture and also replace Ω with Ω^h, Γ_N with Γ_N^h, and Γ_R with Γ_R^h:

(G)

Find $T^h \in \mathcal{S}^h$ such that:

$$b^h(T^h, w^h) = l^h(w^h) \qquad \text{for all } w^h \in \mathcal{V}^h$$

where:

$$b^h(T^h, w^h) = \int_{\Omega^h} K \vec{\nabla} T^h \cdot \vec{\nabla} w^h \, d\Omega + \int_{\Gamma_R^h} \beta T^h w^h \, d\Gamma$$

$$l^h(w^h) = \int_{\Omega^h} f w^h \, d\Omega + \int_{\Gamma_N^h} h w^h \, d\Gamma + \int_{\Gamma_R^h} \beta T_R w^h \, d\Gamma$$

Note that since we replaced Ω with Ω^h, Γ_N with Γ_N^h, and Γ_R with Γ_R^h, we have used a superscript "h" in denoting the bilinear and linear forms in the above problem. This emphasizes that the bilinear and linear forms depend on the finite element mesh since the finite element mesh is generally an approximation of the actual domain geometry.

We refer to the solution of the above problem as the finite element solution.

It takes the form:

$$T^h(\vec{x}) = \sum_{B \in \eta - \eta_D} T^h(\vec{x}_B) N_B(\vec{x}) + g^h(\vec{x})$$

Unknown DOF

Known

Just as was the case for the 1D model problem studied earlier, the unknown degrees of freedom associated with the Galerkin solution may be found via the solution of a linear system of equations. Arguing as we did before for the 1D model problem studied earlier, we find:

$$\sum_{B \in \eta - \eta_D} b^h(N_B, N_A) T^h(\vec{x}_B) = \ell^h(N_A) - b^h(g^h, N_A) \quad \text{for } A \in \eta - \eta_D$$

Now since we have an equation for each $A \in \eta - \eta_D$, we assign each $A \in \eta - \eta_D$ a unique equation number between 1 and n_{eq} where $n_{eq} = |\eta - \eta_D|$ is the total number of equations. We assign each $A \in \eta_D$ an equation number of zero since there are no equations associated with the node numbers in η_D. We store these assignments using a destination array:

$$ID(A) = \begin{cases} P & \text{if } A \in \eta - \eta_D \\ 0 & \text{if } A \in \eta_D \end{cases}$$

Note that for each $P = 1, \ldots, n_{eq}$ there is a unique $A \in \eta - \eta_D$ such that $ID(A) = P$. If we then define $\underline{\underline{K}}$ to be the $n_{eq} \times n_{eq}$ stiffness

matrix with entries:

$$K_{PQ} = b^h(N_B, N_A) \qquad \text{where} \quad ID(A) = P,\ ID(B) = Q$$

$\underline{F}$ to be the $n_{eq} \times 1$ force vector with entries:

$$F_P = \ell^h(N_A) - b^h(g^h, N_A) \qquad \text{where} \quad ID(A) = P$$

and $\underline{d}$ to be the $n_{eq} \times 1$ displacement vector with entries:

$$d_Q = T^h(x_B) \qquad \text{where} \quad ID(B) = Q$$

then we have arrived at the matrix problem:

$$(L) \begin{cases} \text{Find } \underline{d} \in \mathbb{R}^{n_{eq}} \text{ such that:} \\ \underline{\underline{K}}\,\underline{d} = \underline{F} \end{cases}$$

Just as was the case with the 1D model problem studied earlier, once we solve the matrix problem above for the displacement vector $\underline{d} \in \mathbb{R}^{n_{eq}}$, we know the finite element solution:

$$T^h(\vec{x}) = \sum_{B \in \eta - \eta_D} d_{ID(B)} N_B(\vec{x}) + g^h(\vec{x})$$

Of course, before one can solve the matrix problem stated above, one must first compute the entries of the stiffness matrix and force vector. We will again turn to element formation and assembly for this task. This is the focus

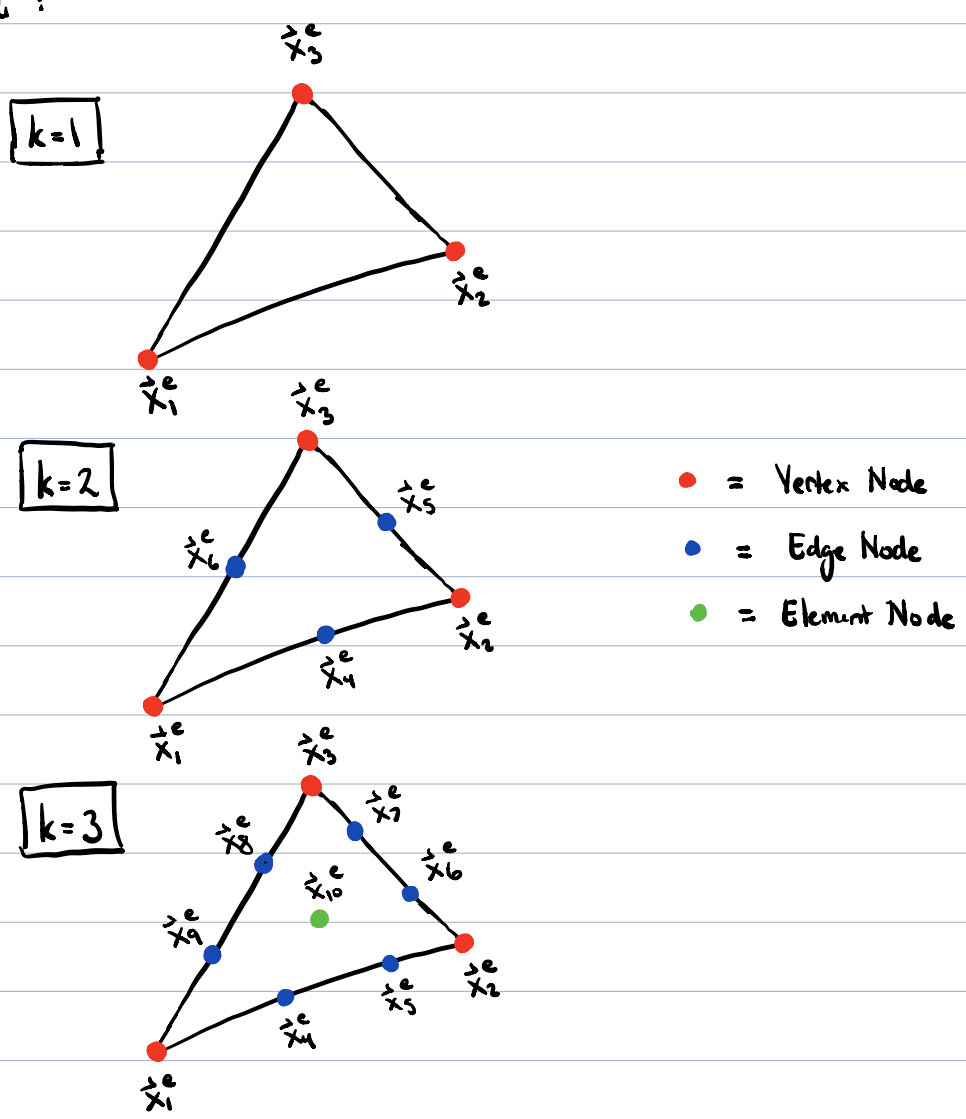
of a future lecture. Just as was the case for the 1D model problem studied earlier, the stiffness matrix here is symmetric and positive-definite. This can be taken advantage of during both system formation and system solution to save computational cost.

Before proceeding, it should be remarked that the destination array is not unique. It can be constructed in many different ways. Certain constructions will result in a stiffness matrix with a smaller bandwidth than other constructions, which is advantageous when sparse direct solvers are employed for system solution. A rather simple algorithm for constructing the connectivity array given the sets η and η_D is provided below:

```
Set P = 0
for A = 1 : n_nod
    if A ∈ η - η_D
        Set P = P+1
        Set ID(A) = P
    else
        Set ID(A) = 0
    endif
endfor
```

Smarter algorithms take into account the finite element mesh connectivity.

To conclude, recall earlier it was mentioned earlier that simplified expressions can be attained for the basis functions associated with an affine triangular finite element approximation if we express them in terms of shape functions defined over a parent triangular finite element. To demonstrate this, let us restrict our view to a single element, Ω^e:



The only nonzero basis functions are those associated with the

n_{en} nodes attached to the element vertices, the element edges, and the element itself, so we create a local numbering associated with these nodes as illustrated above according to the following rules:

Rule 1: The first three nodes, $\{\vec{x}_a^e\}_{a=1}^{3}$, are the vertex nodes, ordered in counter-clockwise fashion.

Rule 2: The next $3*(k-1)$ nodes, $\{\vec{x}_a^e\}_{a=4}^{3k}$, are the edge nodes, also ordered in counter-clockwise fashion. The first of these nodes, $\vec{x}_4^e$, is taken to be the first to the right of $\vec{x}_1^e$.

Rule 3: The remaining nodes, $\{\vec{x}_a^e\}_{a=3k+1}^{n_{en}}$, are the element nodes. These nodes are organized into concentric triangles. The nodes in the outermost triangle are numbered first, using Rules 1 and 2 unless there is only one node, followed by the next outermost triangle and so on. The first node in every concentric triangle is taken to be that closest to $\vec{x}_1^e$.

To link the local and global numberings of nodes, we use an element connectivity, just as we did in the 1D setting:

$$A = IEN(a, e)$$

A: Global Node Number; a: Local Node Number; e: Element Number

Element connectivities are typically provided by mesh generators. For instance, gmsh provides element connectivities for both triangular and quadrilateral finite element meshes that it produces for degrees $k=1,2,3$. We can also locally renumber the nonzero basis functions using the same numbering scheme as the local nodes:

$$N_a^e(\vec{x}) = N_A(\vec{x}) \quad \text{where} \quad A = \text{IEN}(a,e)$$

for $a = 1, \ldots, n_{en}$. Note that:

$$N_a^e(\vec{x}_b^e) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}$$

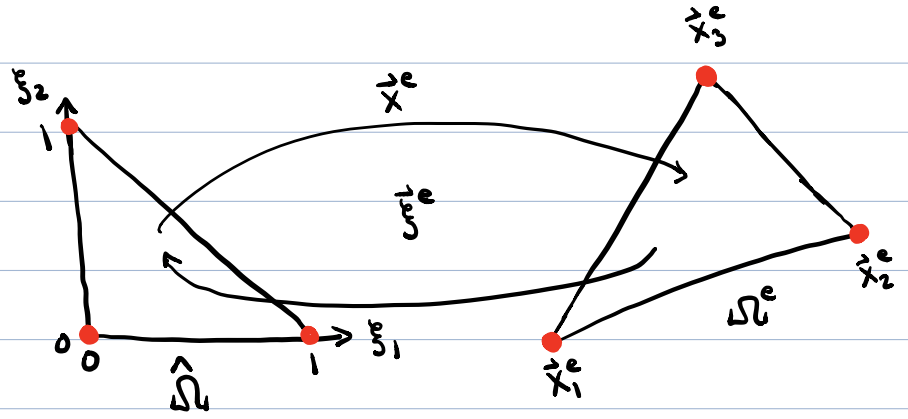
for $a, b = 1, \ldots, n_{en}$. The above interpolation property can be used to find the exact form of each of $\{N_a^e\}_{a=1}^{n_{en}}$. However, we take a different route. Let:

$$\hat{\Omega} = \{(\xi_1, \xi_2) : \xi_1, \xi_2, 1 - \xi_1 - \xi_2 > 0\}$$

be the unit right triangle, and let $\vec{x}^e : \hat{\Omega} \to \Omega^e$ be the mapping:

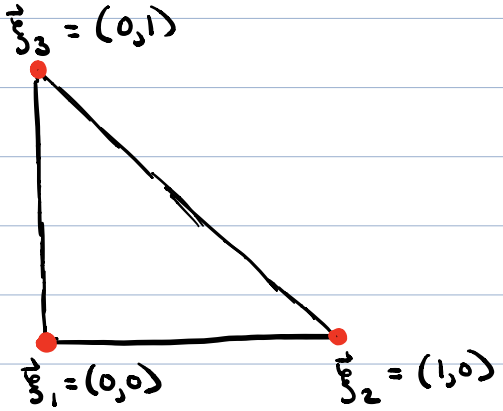
$$\vec{x}^e(\vec{\xi}) = \vec{x}_1^e (1 - \xi_1 - \xi_2) + \vec{x}_2^e \xi_1 + \vec{x}_3^e \xi_2$$

If Ω^e is non-degenerate, that is, if it is non-empty, then the above mapping is invertible with inverse $\vec{\xi}^e : \Omega^e \to \hat{\Omega}$. Visually:

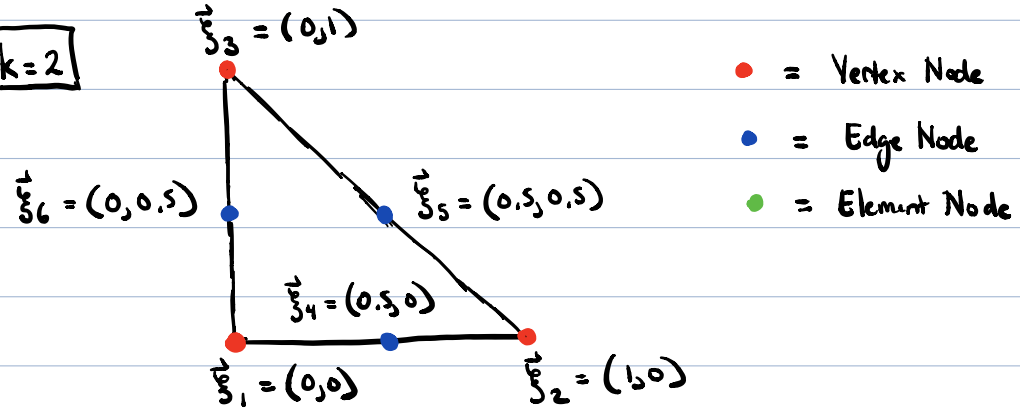


Just as in the 1D setting, $\hat{\Omega}$ is referred to as the parent or reference element here. We can place n_{en} equi-spaced nodes $\{\vec{\xi}_a\}_{a=1}^{n_{en}}$ on the parent element, just like the physical element, ordering them according to Rules 1-3 just like the physical element with $\vec{\xi}_1 = (0,0)$:

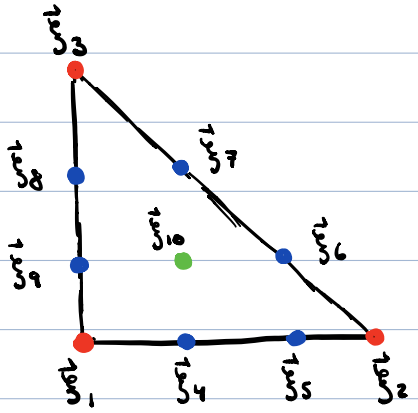
k=1



k=2



k = 3



$\vec{\xi}_1 = (0,0)$, $\vec{\xi}_2 = (1,0)$,
$\vec{\xi}_3 = (0,1)$, $\vec{\xi}_4 = (\frac{1}{3},0)$,
$\vec{\xi}_5 = (\frac{2}{3},0)$, $\vec{\xi}_6 = (\frac{2}{3},\frac{1}{3})$,
$\vec{\xi}_7 = (\frac{1}{3},\frac{2}{3})$, $\vec{\xi}_8 = (0,\frac{2}{3})$,
$\vec{\xi}_9 = (0,\frac{1}{3})$, $\vec{\xi}_{10} = (\frac{1}{3},\frac{1}{3})$

A simple calculation reveals that:

$$\vec{x}_a^e = \vec{x}^e(\vec{\xi}_a)$$

Thus it holds that:

$$N_a^e(\vec{x}^e(\vec{\xi}_b)) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}$$

Since the points $\{\vec{\xi}_a\}_{a=1}^{n_{en}}$ are independent of element, it follows then that:

$$N_a^e(\vec{x}^e(\vec{\xi})) = \hat{N}_a(\vec{\xi})$$

where $\{\hat{N}_a\}_{a=1}^{n_{en}}$ are generic shape functions defined by the conditions:

$$\hat{N}_a(\vec{\xi}_b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}$$

for $a, b = 1, \ldots, n_{en}$. As opposed to the basis functions $\{N_a^e\}_{a=1}^{n_{en}}$,

the shape functions $\{\hat{N}_a\}_{a=1}^{nen}$ do not depend on element and thus they can be expressed in a single closed form independent of element:

k=1

$$\hat{N}_1(\vec{\xi}) = 1 - \xi_1 - \xi_2$$
$$\hat{N}_2(\vec{\xi}) = \xi_1$$
$$\hat{N}_3(\vec{\xi}) = \xi_2$$

k=2

$$\hat{N}_1(\vec{\xi}) = (1-\xi_1-\xi_2)(1-2\xi_1-2\xi_2)$$
$$\hat{N}_2(\vec{\xi}) = \xi_1(2\xi_1-1)$$
$$\hat{N}_3(\vec{\xi}) = \xi_2(2\xi_2-1)$$
$$\hat{N}_4(\vec{\xi}) = 4(1-\xi_1-\xi_2)\xi_1$$
$$\hat{N}_5(\vec{\xi}) = 4\xi_1\xi_2$$
$$\hat{N}_6(\vec{\xi}) = 4\xi_2(1-\xi_1-\xi_2)$$

k=3

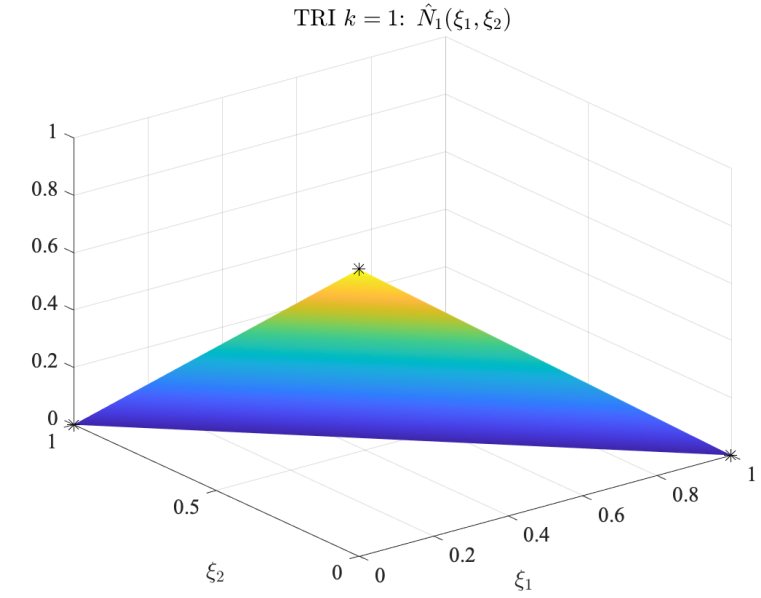
$$\hat{N}_1(\vec{\xi}) = \frac{1}{2}(1-\xi_1-\xi_2)(2-3\xi_1-3\xi_2)(1-3\xi_1-3\xi_2)$$
$$\hat{N}_2(\vec{\xi}) = \frac{1}{2}\xi_1(3\xi_1-1)(3\xi_1-2)$$
$$\hat{N}_3(\vec{\xi}) = \frac{1}{2}\xi_2(3\xi_2-1)(3\xi_2-2)$$
$$\hat{N}_4(\vec{\xi}) = \frac{9}{2}(1-\xi_1-\xi_2)\xi_1(2-3\xi_1-3\xi_2)$$
$$\hat{N}_5(\vec{\xi}) = \frac{9}{2}(1-\xi_1-\xi_2)\xi_1(3\xi_1-1)$$
$$\hat{N}_6(\vec{\xi}) = \frac{9}{2}\xi_1\xi_2(3\xi_1-1)$$
$$\hat{N}_7(\vec{\xi}) = \frac{9}{2}\xi_1\xi_2(3\xi_2-1)$$
$$\hat{N}_8(\vec{\xi}) = \frac{9}{2}\xi_2(1-\xi_1-\xi_2)(3\xi_2-1)$$
$$\hat{N}_9(\vec{\xi}) = \frac{9}{2}\xi_2(1-\xi_1-\xi_2)(2-3\xi_1-3\xi_2)$$
$$\hat{N}_{10}(\vec{\xi}) = 27(1-\xi_1-\xi_2)\xi_1\xi_2$$

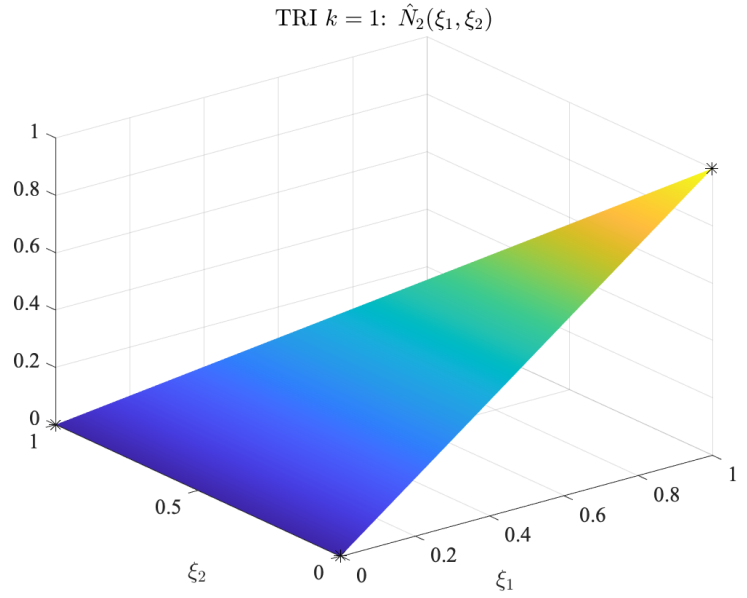
For easy reference, surface plots of each of the above shape functions is included at the end of these notes.

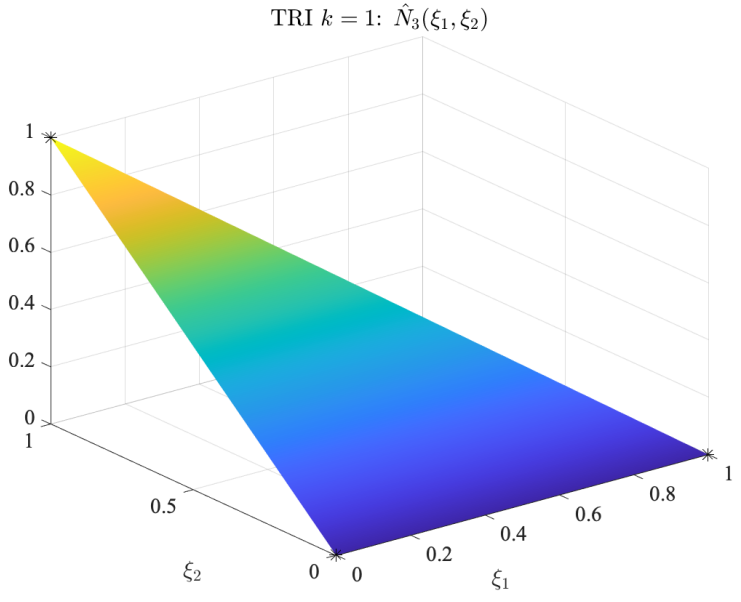
As a final remark, note we can evaluate every basis function at every point on a physical element Ω_e^e given the connectivity array and shape functions:

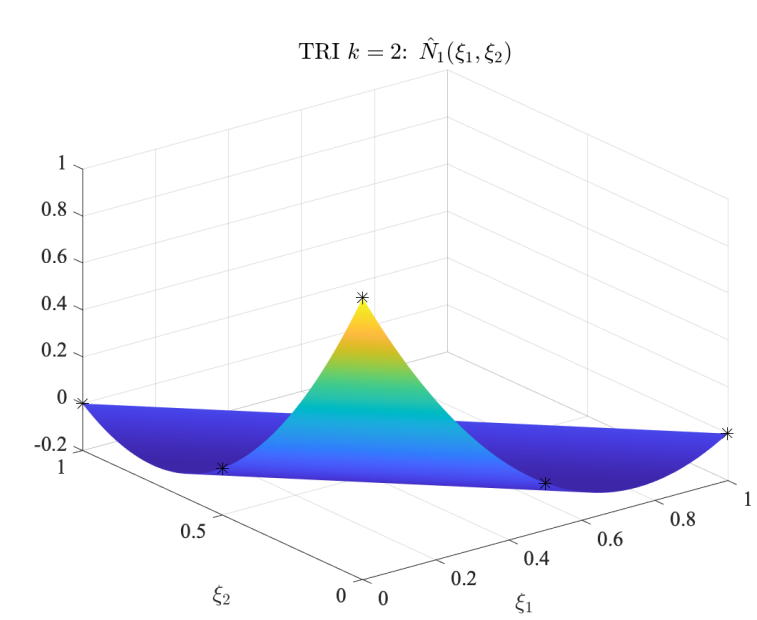
$$N_A(\vec{x}) = \begin{cases} \hat{N}_a(\vec{\xi}^e(\vec{x})) & \text{if there is an } a \text{ such that } A = IEN(a,e) \\ 0 & \text{otherwise} \end{cases}$$

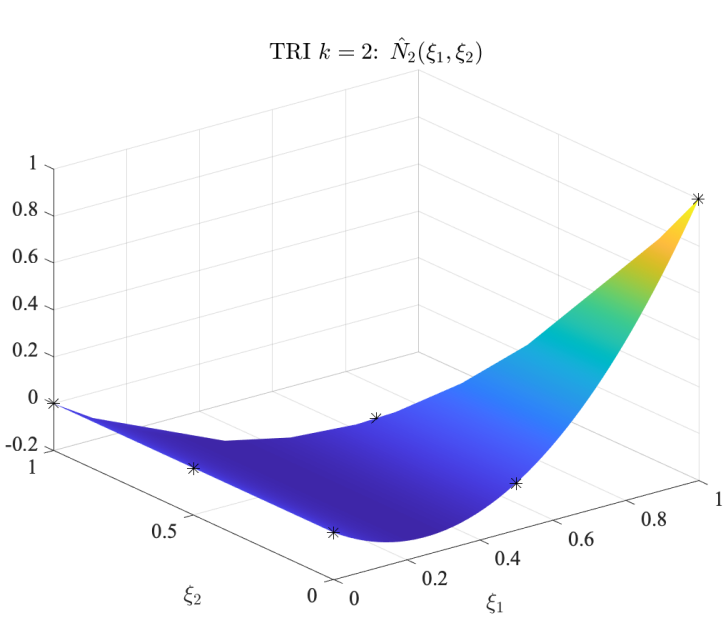
However, it is far more economical to only compute nonzero basis functions on a given element. We will take this approach in element formation and assembly.

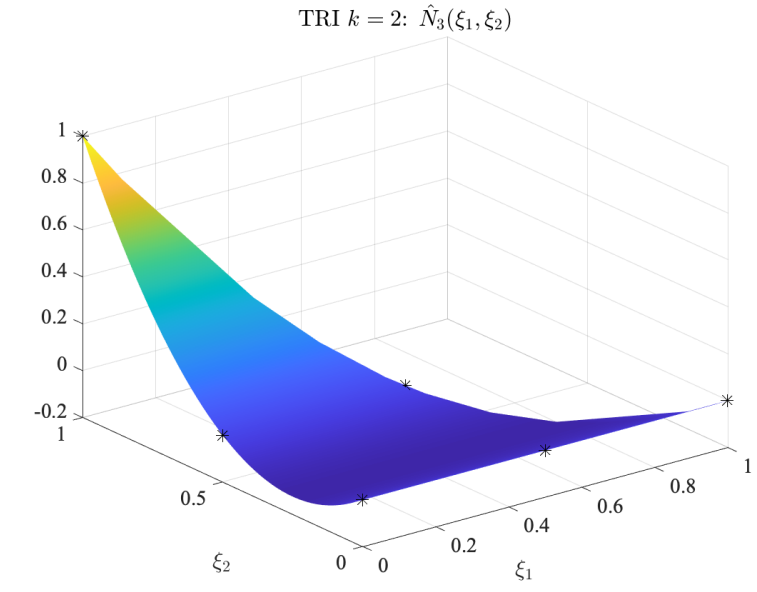

TRI $k = 1$: $\hat{N}_1(\xi_1, \xi_2)$
1
0.8
0.6
0.4
0.2
0
1
0.5
0
ξ_2
0
0.2
0.4
0.6
0.8
1
ξ_1

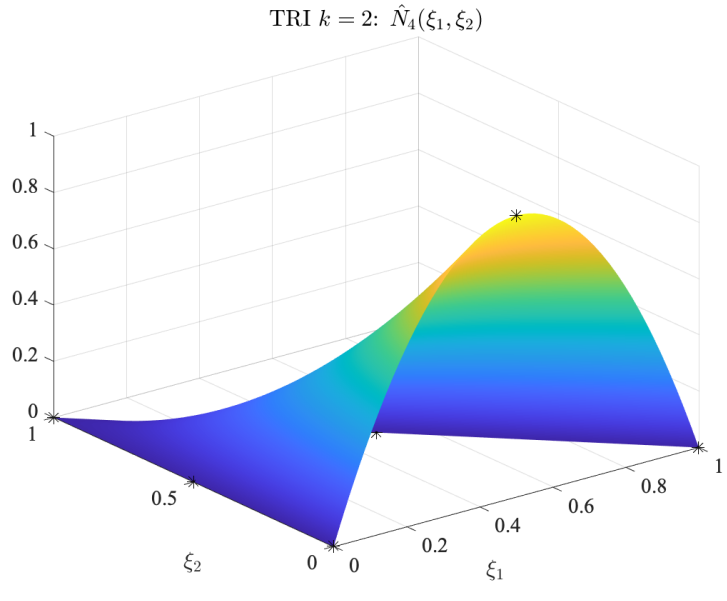

TRI $k = 1$: $\hat{N}_2(\xi_1, \xi_2)$
1
0.8
0.6
0.4
0.2
0
1
0.5
0
ξ_2
0
0.2
0.4
0.6
0.8
1
ξ_1

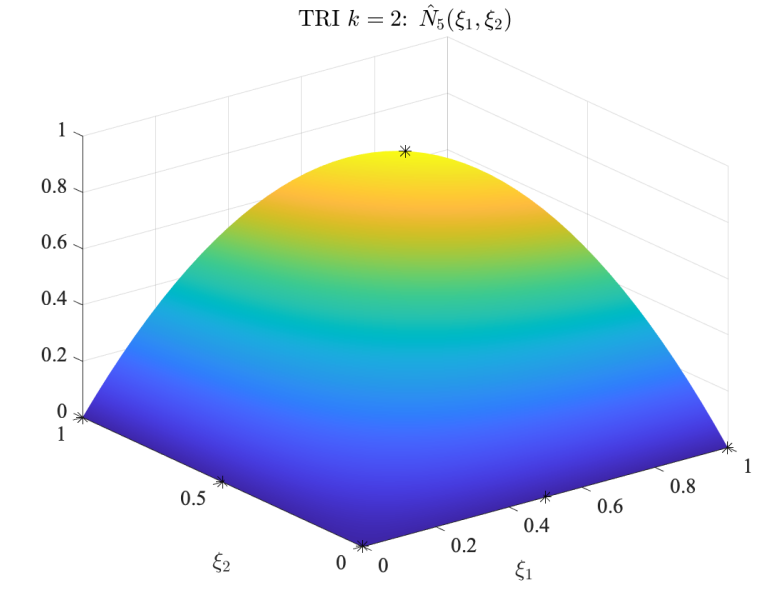

TRI $k = 1$: $\hat{N}_3(\xi_1, \xi_2)$
1
0.8
0.6
0.4
0.2
0
1
0.5
0
ξ_2
0
0.2
0.4
0.6
0.8
1
ξ_1

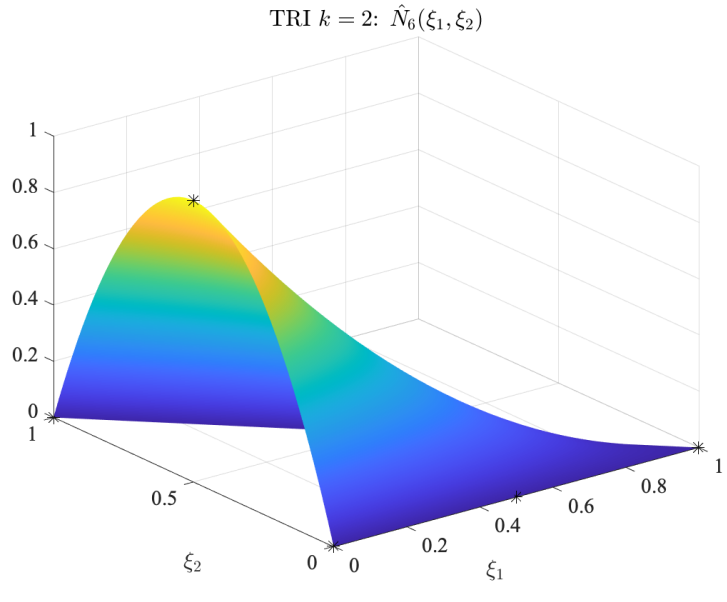

TRI $k = 2$: $\hat{N}_1(\xi_1, \xi_2)$
ξ_1
ξ_2


TRI $k = 2$: $\hat{N}_2(\xi_1, \xi_2)$
ξ_1
ξ_2

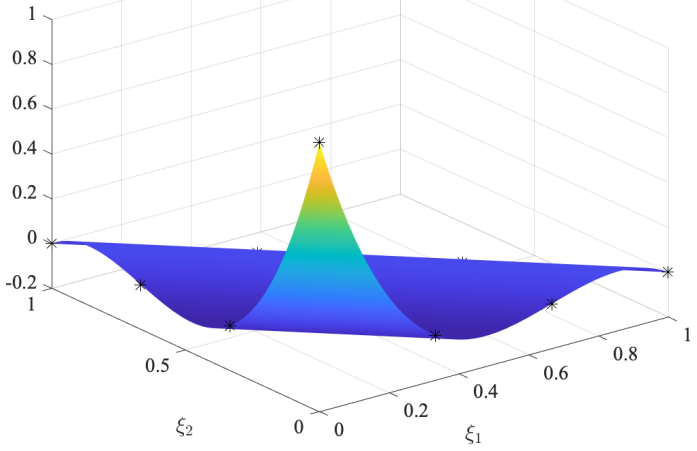

TRI $k = 2$: $\hat{N}_3(\xi_1, \xi_2)$
ξ_1
ξ_2


TRI $k = 2$: $\hat{N}_4(\xi_1, \xi_2)$
ξ_1
ξ_2

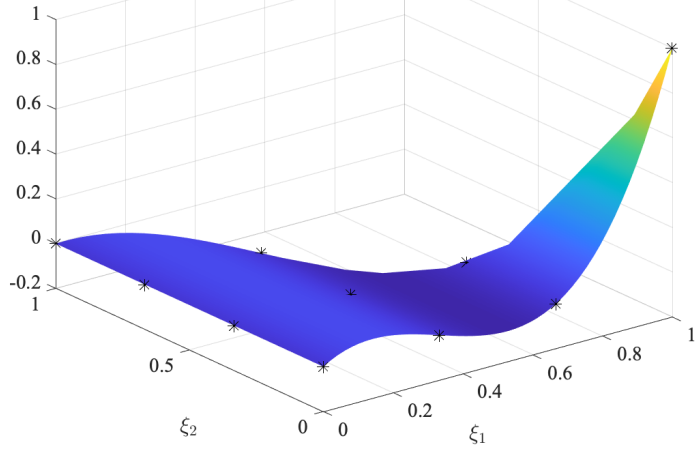

TRI $k = 2$: $\hat{N}_5(\xi_1, \xi_2)$
ξ_1
ξ_2


TRI $k = 2$: $\hat{N}_6(\xi_1, \xi_2)$
ξ_1
ξ_2

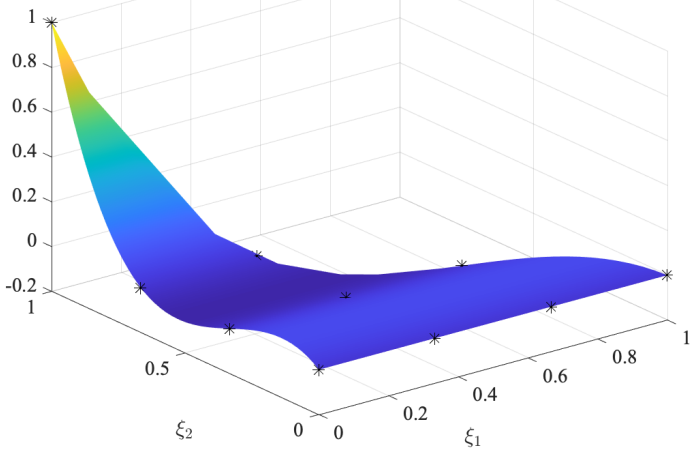
TRI $k = 3$: $\hat{N}_1(\xi_1, \xi_2)$



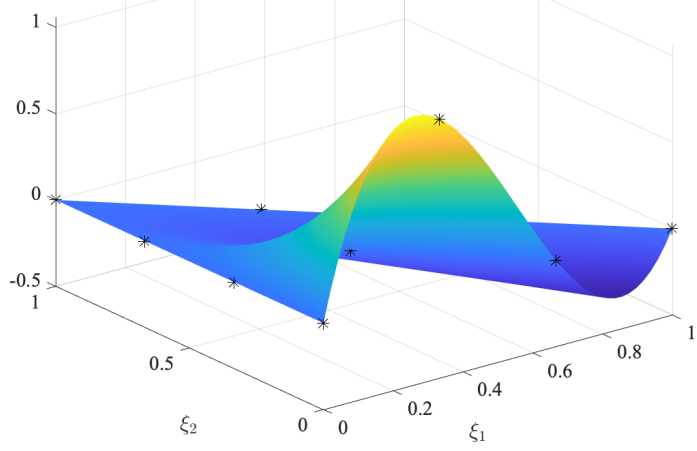
TRI $k = 3$: $\hat{N}_2(\xi_1, \xi_2)$



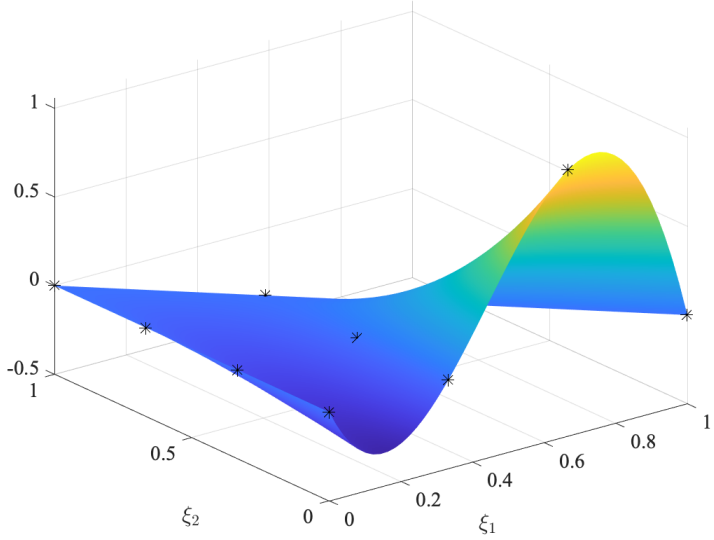
TRI $k = 3$: $\hat{N}_3(\xi_1, \xi_2)$



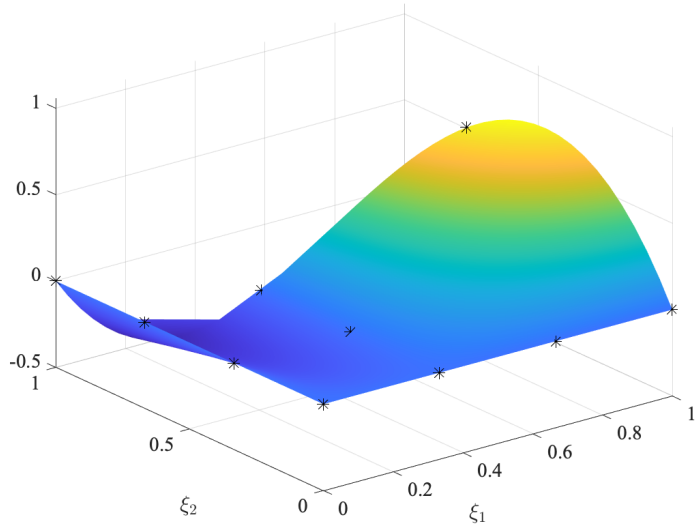
TRI $k = 3$: $\hat{N}_4(\xi_1, \xi_2)$



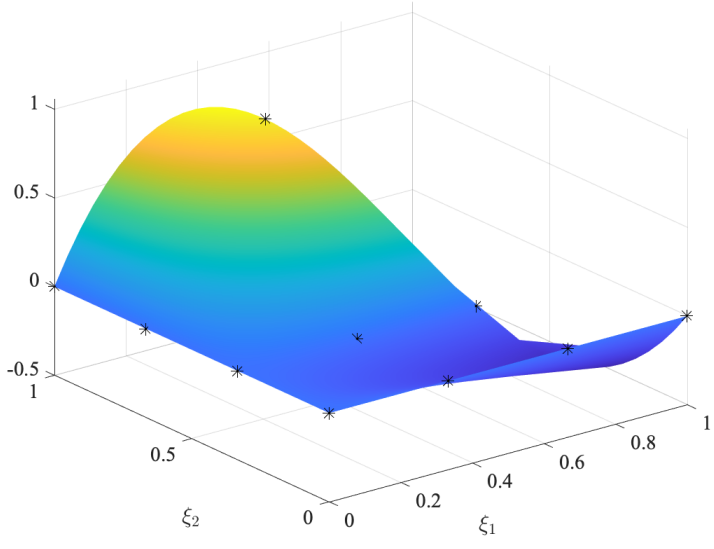
TRI $k = 3$: $\hat{N}_5(\xi_1, \xi_2)$



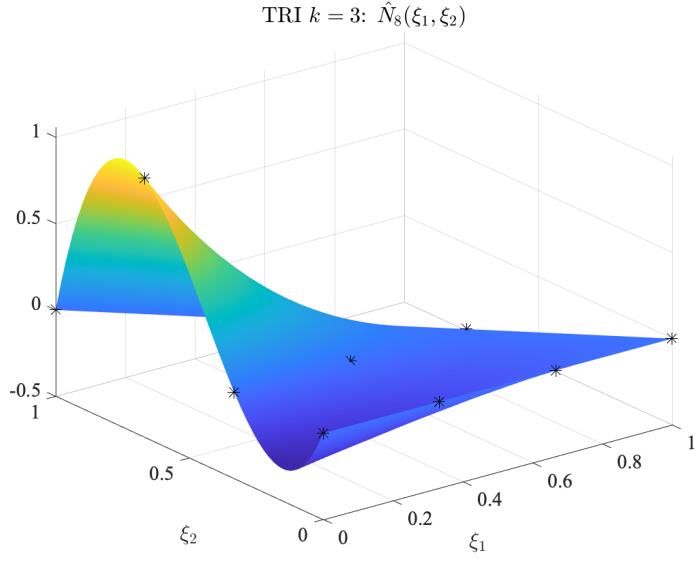
TRI $k = 3$: $\hat{N}_6(\xi_1, \xi_2)$



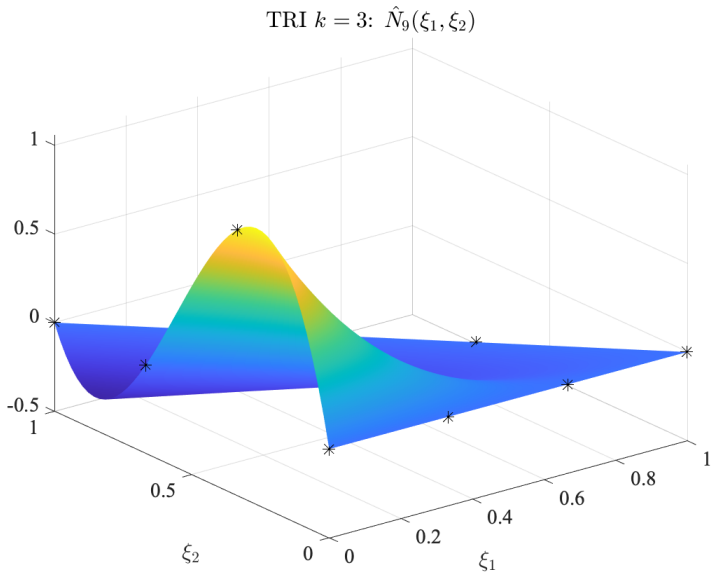
TRI $k = 3$: $\hat{N}_7(\xi_1, \xi_2)$



TRI $k = 3$: $\hat{N}_8(\xi_1, \xi_2)$



TRI $k = 3$: $\hat{N}_9(\xi_1, \xi_2)$



TRI $k = 3$: $\hat{N}_{10}(\xi_1, \xi_2)$