Finite Element Method (FEM)

- With finite differences, we approximate the *equation*, e.g.,

$$-\frac{d^2\tilde{u}}{dx^2} \;=\; f(x) \;\longrightarrow\; -\frac{\delta^2 u_j}{\delta x^2} \;=\; f_j, \quad + \text{BCs.} \tag{1}$$

- With the FEM, we approximate the *solution*, e.g.,

$$u(x) := \sum_{j=1}^{n} \hat{u}_j \phi_j(x), \qquad X_0^N \;:=\; \text{span}\{\phi_1, \ldots, \phi_n\}, \tag{2}$$

  where the ϕ_js satisfy homogeneous BCs (say).

- We try to make the error, $e(x) := \tilde{u}(x) - u(x)$, *small*.

Collocation

- One way to try to make the error small is to force the *residual* to be zero at certain collocation points, x_i:

$$r(x) \;:=\; -\frac{d^2u}{dx^2} \;-\; f(x) \;=\; 0 \;\text{ at } x_i,\; i = 1, \ldots, n. \tag{3}$$

$$:=\; -\frac{d^2u}{dx^2} \;+\; \frac{d^2\tilde{u}}{dx^2} \tag{4}$$

$$:=\; -\frac{d^2e}{dx^2}. \tag{5}$$

- Clearly, $r(x) \equiv 0$ if $u(x) = \tilde{u}(x)$.
- The *residual* is computable and is the only available measure of the error.
- Implementation of the collocation scheme is

$$-\begin{bmatrix} \phi_1''(x_1) & \phi_2''(x_1) & \cdots & \phi_n''(x_1) \\ \phi_1''(x_2) & \phi_2''(x_2) & \cdots & \phi_n''(x_2) \\ \vdots & & & \vdots \\ \phi_1''(x_n) & \phi_2''(x_n) & \cdots & \phi_n''(x_n) \end{bmatrix} \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \tag{6}$$

- Q: If you were going to implement collocation, what would be a good set of points?

Weighted Residual Techniques

- For several reasons, its better to make the residual small in a *weighted* sense, rather than just enforcing $r(x) = 0$ at a few isolated points.

- A disadvantage of collocation is that it requires $\phi_j \in C^1$, i.e., twice differentiable, which precludes piecewise-linear (FEM) basis functions. :(

- Another disadvantage is that it does not guarantee a *best-fit* approximation.

- Also, it does not yield a symmetric "stiffness" matrix ($A := -D^2$).

- Moreover, Neumann and Robin BCs are not easy to implement (many many papers on this very topic).

- For these reasons, the *Weighted Residual Technique* is strongly preferred.

Weighted Residual Method

- Here, rather than enforcing $r(x) = 0$ pointwise, we seek a solution $u(x) \in X_0^N$ such that $r(x) \perp Y^N$ for a suitably chosen Y^N.

- We call X_0^N the *trial space* and Y_0^N the *test space.*

- Here, we define "$\perp$" in the following sense:

$$\text{Let } (f, g) \;:=\; \int_\Omega f(x)\, g(x)\, dx. \tag{7}$$

$$\text{We say } f \perp g \quad \text{if} \quad (f, g) \;=\; 0. \tag{8}$$

- Q: Why would orthogonality imply a small residual ??

The WRT is essentially a method of undetermined coefficients.

- Consider the 1D Helmholtz equation with $\beta > 0$,

$$-\frac{d^2\tilde{u}}{dx^2} + \beta\tilde{u} = f(x), \qquad \tilde{u}(0) = \tilde{u}(1) = 0. \tag{9}$$

- Seek an approximate solution u in a finite-dimensional *trial space* X_0^N,

$$u \in X_0^N := \text{span}\{\phi_1(x), \phi_2(x), \ldots, \phi_n(x)\}, \ \phi_j(0) = \phi_j(1) = 0. \tag{10}$$

  (We use the subscript 0 on X_0^N to indicate that functions in this space satisfy the homogeneous Dirichlet boundary conditions.)

- The trial solution is a linear combination of the *basis functions* $\phi_j(x)$ with *basis coefficients* $\hat{u}_j$,

$$u(x) \;=\; \sum_{j=1}^{n} \phi_j(x)\hat{u}_j. \tag{11}$$

- The orthogonality condition is based on the standard L^2 inner product. Specifically, we require

$$0 \;=\; \int_0^1 v(x)\, r(x)\, dx \;=\; \int_0^1 v(x) \left(f + \frac{d^2u}{dx^2} \;-\; \beta u \right) dx \;\; \forall v \in Y_0^N \tag{12}$$

or,

$$\int_0^1 v(x) \left(-\frac{d^2u}{dx^2} \;+\; \beta u \right) dx \quad = \quad \int_0^1 v\, f\, dx \;\; \forall v \in Y_0^N. \tag{13}$$

$$\tag{14}$$

- Note that if Y_0^N=span$\{\psi_i(x)\}$, $i = 1, \ldots, n$ and $\psi_i(x) = \delta(x - x_i)$, the Dirac delta function, then we **recover collocation**.
  - That is, we are enforcing $r(x_i)$=0.

Galerkin Method

- For the Poisson and Helmholtz equations, the optimal choice is $Y_0^N = X_0^N$, which is the **Galerkin method**.
- It appears that u must be twice differentiable.
  We can avoid this requirement through integration by parts.
- Let $\mathcal{I}$ denote the left-hand side of the preceding equation.

$$\mathcal{I} = \int_0^1 \left(-v(x)\frac{d^2u}{dx^2} + \beta vu \right) dx \tag{15}$$

$$= \int_0^1 \left(\frac{dv}{dx}\frac{du}{dx} + \beta vu \right) dx - vu'|_0^1 \tag{16}$$

$$= \int_0^1 \left(\frac{dv}{dx}\frac{du}{dx} + \beta vu \right) dx. \tag{17}$$

- The boundary terms vanish because $v = 0$ at $x = 0$ and 1.
- We see that the number of derivatives on the trial (u) and test (v) functions is now the same.
- They thus have the same (low) continuity requirements, which is feasible because they are in the same space ($Y_0^N \equiv X_0^N$).

- We denote the integral $\mathcal{I}$ as the energy (or "a") inner-product,

$$a(v,u) \quad := \quad \int_0^1 \left(\frac{dv}{dx}\frac{du}{dx} \; + \; \beta v u \right) dx. \tag{18}$$

- $a(\cdot,\cdot)$ is symmetric, $a(v,u) = a(u,v)$, and positive definite:

$$a(u,u) > 0 \, \forall u \neq 0. \tag{19}$$

  For $\beta \equiv 0$, $a(u,u) = 0$ only if $u =$ constant, but the only constant in X_0^N is $u = 0$.

- Our discrete problem can be stated as,

$$\begin{aligned} &\textit{Find } u \in X_0^N \textit{ such that} \\ &a(v,u) \; = \; (v,f) \; \forall v \; \in X_0^N. \end{aligned} \tag{20}$$

- Note that this statement is an identity for $\tilde{u}$ (generally $\tilde{u} \notin X_0^N$):

$$a(v,\tilde{u}) \; \equiv \; (v,f) \tag{21}$$

  which holds for all v for which the integrand is computable.

- Specifically, for the continuous problem, we refer to the formulation,
  Find $\tilde{u} \in \mathcal{H}_0^1$ *such that*

$$a(v,\tilde{u}) \; = \; (v,f) \; \forall v \; \in \mathcal{H}_0^1, \tag{22}$$

  as the *weak form*, which means we are allowed to look for solutions that are in a space that is larger than $C^1[\Omega]$.

- Set of spaces commonly used for analysis of PDEs are **Sobolev spaces**.
- The most important ones for our purpose:

$$\mathcal{L}^2 = \left\{ v | \int_\Omega v^2 \, d\mathbf{x} < \infty \right\} \tag{23}$$

$$\mathcal{H}^1 = \left\{ v | \, v \in \mathcal{L}^2, \, \int_\Omega \nabla v \cdot \nabla v \, d\mathbf{x} < \infty \right\} \tag{24}$$

$$\mathcal{H}_0^1 = \left\{ v | \, v \in \mathcal{H}^1, \; v(\mathbf{x})|_{\partial\Omega_D} = 0 \right\} \tag{25}$$

$$\mathcal{H}_b^1 = \left\{ v | \, v \in \mathcal{H}^1, \; v(\mathbf{x})|_{\partial\Omega_D} = v_b(\mathbf{x}) \right\} \tag{26}$$

$$X_0^N = \left\{ v | \, v \in \mathcal{H}_0^1 \cap \; span\{\phi_1 \, \phi_2 \, \ldots \phi_n\} \right\} \tag{27}$$

$$\tag{28}$$

- Note that $\mathcal{H}_b^1$ is not closed.
  Functions in this space cannot be represented as arbitrary linear combinations of elements in this space. Usually, we pick one element from $\mathcal{H}_b^1$, say $u_b(\mathbf{x})$ and then seek our solution as $u(\mathbf{x}) = u_b + u_0(\mathbf{x})$, where $u_0 \in \mathcal{H}_0^1$.

- Associated with these spaces we have the following inner products and induced norms

$$\begin{array}{llllll} \mathcal{L}^2: & (f,g)_0 = \int_\Omega f g \, d\mathbf{x}, & & \|f\| = [(f,g)]^{\frac{1}{2}} \\ \mathcal{H}^1: & (f,g)_1 = \int_\Omega f g + \nabla f \cdot \nabla g \, d\mathbf{x}, & & \|f\|_1 = [(f,g)_1]^{\frac{1}{2}} \end{array} \tag{29}$$

- Returning to our differential equation, we have, for $f \in \mathcal{L}^2$,

$$\textit{(a) Find } \tilde{u} \in \mathcal{H}_0^1 \textit{ such that} \quad a(v, \tilde{u}) \;=\; (v, f)\; \forall v \;\in \mathcal{H}_0^1, \tag{30}$$

$$\textit{(b) Find } u \in X_0^N \subset \mathcal{H}_0^1 \textit{ such that} \quad a(v, u) \;=\; (v, f)\; \forall v \;\in X_0^N. \tag{31}$$

- Nominally, *(b)* completes our discretization.
- Once the finite-dimensional subspace $X_0^N \subset \mathcal{H}_0^1$ is identified there are no more choices.

- Let's summarize the results so far:

$$u(\mathbf{x}) \;=\; \sum_{j=0}^{n} u_j \phi_j(\mathbf{x}) \;\in\; X_0^N \tag{32}$$

$$v(\mathbf{x}) \;=\; \sum_{i=0}^{n} v_i \phi_i(\mathbf{x}) \;\in\; X_0^N. \tag{33}$$

$$\tag{34}$$

- For *any* $v \in X_0^N$,

$$a(v,u) \;=\; \int_\Omega \nabla v \cdot \nabla u \, dV \;=\; (v,f) \;:=\; \int_\Omega v\, f \, dV. \tag{35}$$

- Using the preceding expansions,

$$a(v,u) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{n} v_i \left[\int_\Omega \nabla \phi_i \cdot \nabla \phi_i \, dV \right] u_j \tag{36}$$

$$= \; \sum_{i=1}^{n}\sum_{j=1}^{n} v_i a_{ij} u_j \tag{37}$$

$$= \; \underline{v}^T A \underline{u}, \tag{38}$$

where $A = [a_{ij}]$ and

$$a_{ij} \;:=\; \int_\Omega \nabla \phi_i \cdot \nabla \phi_i \, dV. \tag{39}$$

- On the rhs, we have

$$(v, f) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{n} v_i \int_\Omega \phi_i f \, dV \;=:\; \sum_{ij} v_i b_i \;=\; \underline{v}^T \underline{b}. \tag{40}$$

- Combining these, we have, for all $\underline{v} \in \mathbb{R}^n$,

$$\underline{v}^T A \underline{u} \;=\; \underline{v}^T \underline{b}, \tag{41}$$

or simply,

$$A\underline{u} \;=\; \underline{b}. \tag{42}$$

$$\tag{43}$$

- Note that A is SPD.

- There are of course many details...
- And, there is an issue that we generally cannot exactly integrate the product of the data and the test functions, (v, f).

- We turn to some of those details momentarily, but first discuss some optimality properties.

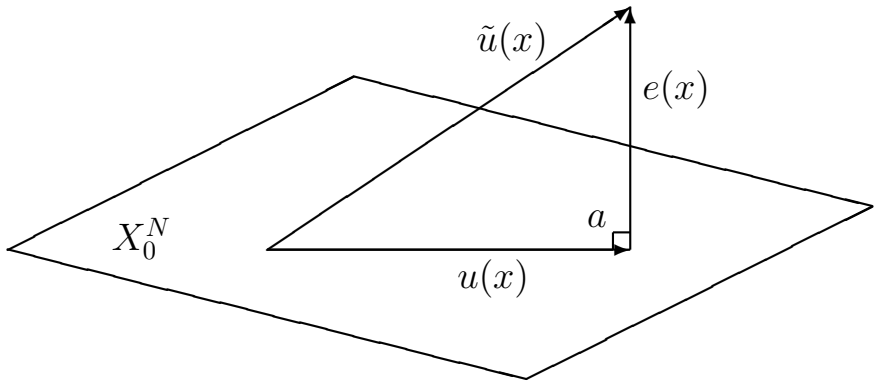
- Because $X_0^N \subset \mathcal{H}_0^1$, we can derive the following error relationship
  Find $u \in X_0^N \subset \mathcal{H}_0^1$ *such that, for all* $v \in X_0^N$,

$$a(v,u) = a(v,\tilde{u}) \tag{44}$$

$$a(v,\tilde{u}) - a(v,u) = 0 \tag{45}$$

$$a(v,\tilde{u}-u) = 0. \tag{46}$$

- Which implies that $e := \tilde{u} - u$ is a-orthogonal to X_0^N: $e(x) \perp_a X_0^N$:



- Therefore, $u(x)$ is the *closest* function in X_0^N to $\tilde{u}(x)$ in the $\|\cdot\|_a$ norm.
  - For any function w satisfying the homogeneous Dirichlet conditions, $w(0) = w(1) = 0$, we define the "a-norm"

$$\|w\|_a := \sqrt{a(w,w)}.$$

Finite Element Methods

- Finite elements (FEM) offer significant advantages over finite differences regarding complex geometry, general boundary conditions, and guaranteed SPD properties (when the PDE is "SPD").
- A key idea of the weighted residual technique is to express the numerical solution as a linear combination of basis functions, $\phi_j(\mathbf{x})$

$$u(\mathbf{x}) \;=\; \sum_{j=1}^{\bar{n}} u_j \, \phi_j(\mathbf{x}), \tag{47}$$

  and to then find the unknown basis coefficients, u_j, such that $u(\mathbf{x})$ approximately solves the PDE.
- In (47), $\bar{n}$ is the number of "global" basis functions, including ones that are nonzero on the domain boundary, $\partial\Omega$.

- For the *finite* element method, these basis functions will

  (i) have compact support (i.e., vanish almost everywhere), and

  (ii) be expressed in terms of expansions on individual patches (*elements*).

- These local expansions take the form

$$u(\mathbf{x})|_{\Omega^e} = \sum_{j=1}^{n_v} u_j^e \, l_j(\mathbf{r}), \quad e = 1, \ldots, E \tag{48}$$

$$\mathbf{x}|_{\Omega^e} = \sum_{j=1}^{n_v} \mathbf{x}_j^e \, l_j(\mathbf{r}), \quad e = 1, \ldots, E, \tag{49}$$

  where n_v is the number of vertices associated with each element and E is the number of finite elements.

- Each element Ω^e is the image of $\hat{\Omega}$ under the transformation (49).

- Here, it is understood that $\mathbf{r} \in \hat{\Omega} \subset \mathbb{R}^d$, $d = 1$, 2, or 3 being the number of spatial dimensions for the PDE.

- In our applications, $\phi_j(\mathbf{x})$ and $l_j(\mathbf{r})$ are *Lagrangian interpolants* that satisfy $\phi_i(\mathbf{x}_j) = \delta_{ij}$ and $l_i(\mathbf{r}_j) = \delta_{ij}$, which implies that $u(\mathbf{x}_j) = u_j$ and $u_i^e = u(\mathbf{x}_j^e)$.

- That is, the basis coefficients are also *grid point values.*

- This choice makes it easy to enforce boundary conditions and function continuity.

Global/Local Finite Element Bases

- The FEM is predicated on a representation of $u(\mathbf{x})$ for all $\mathbf{x} \in \Omega$ that is generically of the form

$$u(\mathbf{x}) \;=\; \sum_{j=1}^{\bar{n}} u_j \, \phi_j(\mathbf{x}). \tag{50}$$

- A characteristic of the *finite* element method is that the region of support (i.e., the region where $\phi_j \neq 0$) is *compact* or *finite.*

- That is, it is typically a small subset of Ω.

- A single global basis function, $\phi_j(\mathbf{x})$ is illustrated in the accompanying figure.

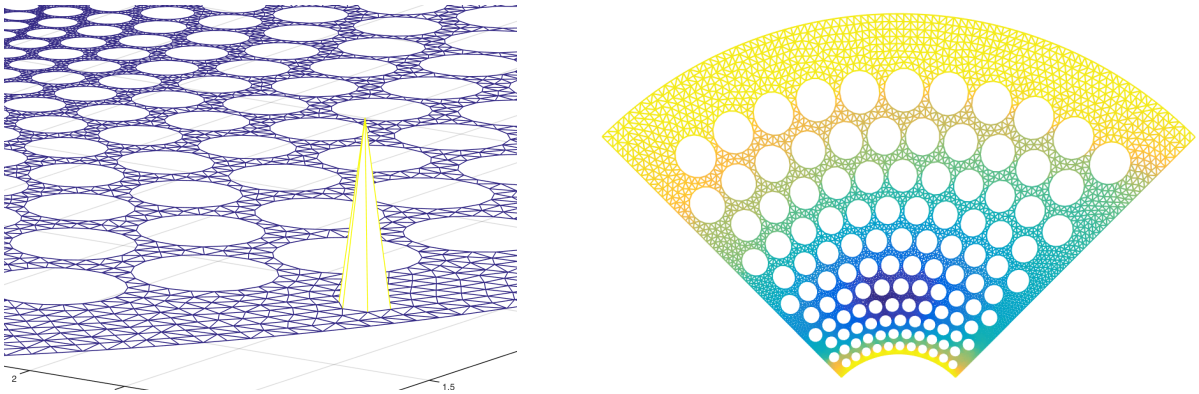


Figure 1: A global finite-element basis function, $\phi_j(\mathbf{x})$ in a complex domain. The basis function shown here is piecewise linear on triangular elements.

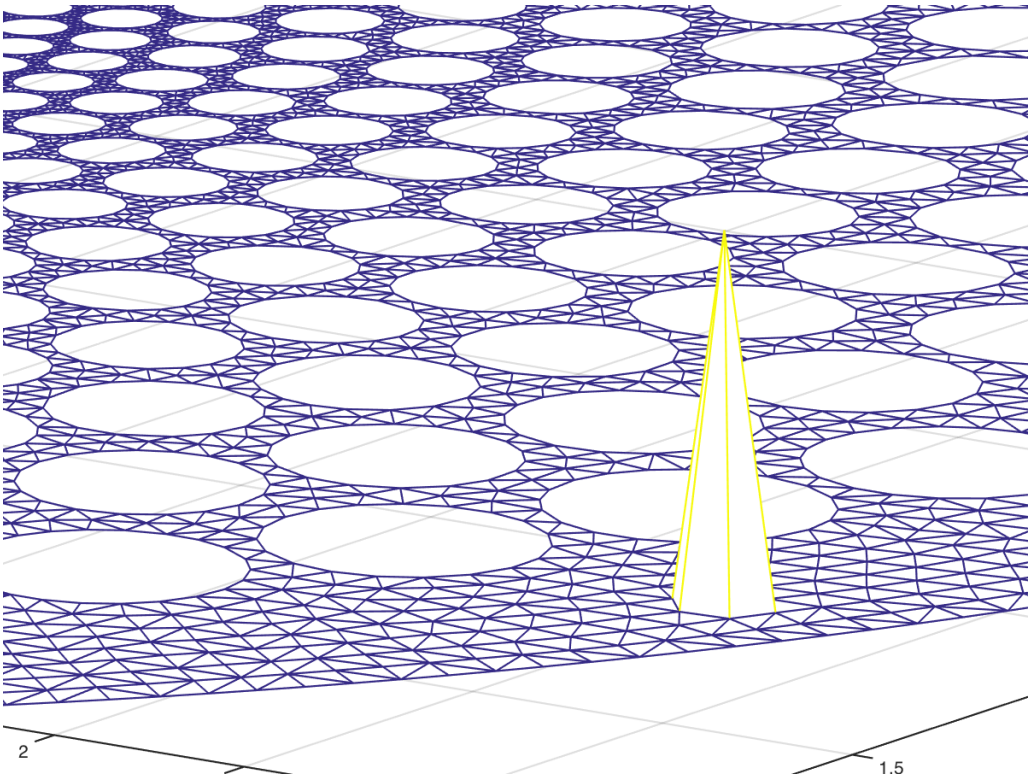
- Note that $\phi_j(\mathbf{x}) \equiv 0$ outside the basis of support, which we denote as the region in the neighborhood of $\mathbf{x}_j$.

- Specifically, we define each *element*, Ω^e as one of the triangular patches seen in the figure and we have that $\phi_j(\mathbf{x})|_{\Omega^e} \equiv 0$ unless $\mathbf{x}_j \in \Omega^e$. (Here, we allow Ω^e to include its boundary.)

- This compact support feature leads to sparse operators because most of the basis functions are mutually orthogonal.

- If there is no element Ω^e, $e = 1, \ldots, E$ containing both $\mathbf{x}_i$ and $\mathbf{x}_j$, then

$$\int_\Omega \phi_i \, \phi_j \, dV \;\; = \;\; 0 \tag{51}$$



- Note that another advantage of the FEM is that integrals of the type (51)
  are readily computed as the sum of integrals over each element,

$$\int_\Omega \phi_i \, \phi_j \, dV \;\; = \;\; \sum_{e=1}^{E} \int_{\Omega^e} \phi_i \, \phi_j \, dV \;\; = \;\; \sum_{e=1}^{E} \int_{\hat{\Omega}} \phi_i \, \phi_j \, \mathcal{J}^e \, dV \tag{52}$$

- These local integrals are readily evaluated by working in a canonical reference element, $\hat{\Omega}$, which is defined as the unit square or unit triangle, depending on the underlying local expansions.

- As indicated in (52), integration on Ω^e is effected by integrating over $\hat{\Omega}$ and using an appropriate Jacobian, $\mathcal{J}^e$, associated with the map from $\hat{\Omega}$ to Ω^e.

- To compute integrals (derivatives, etc.) on $\hat{\Omega}$, we need local *basis functions.*

- These are simply interpolation functions that allow one to accurately interpolate grid point values onto any point in Ω^e (here, $\hat{\Omega}$).

- For all of the FEM bases, the local representations are given by (48), which is repeated here,

$$u(\mathbf{x})|_{\Omega^e} = \sum_{j=1}^{n_v} u_j^e \, l_j(\mathbf{r}), \quad e = 1, \ldots, E \tag{53}$$

  where n_v is the number of vertices in the reference element.

- For linear triangles, $n_v = 3$. (Here, *linear* refers to the polynomial order of the basis functions, $l_j(r, s)$, rather than whether the triangle has straight or "curved" sides.)

- If we take $\hat{\Omega}$ to be the right triangle with vertices $(r, s) = (0,0)$, $(0,1)$, and $(1,0)$, then the basis functions are

$$l_1(r, s) = 1 - r - s \tag{54}$$
$$l_2(r, s) = r \tag{55}$$
$$l_3(r, s) = s, \tag{56}$$

  which are illustrated in the figure below.

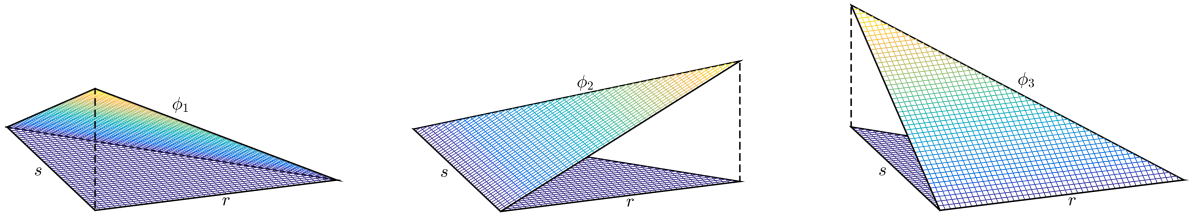
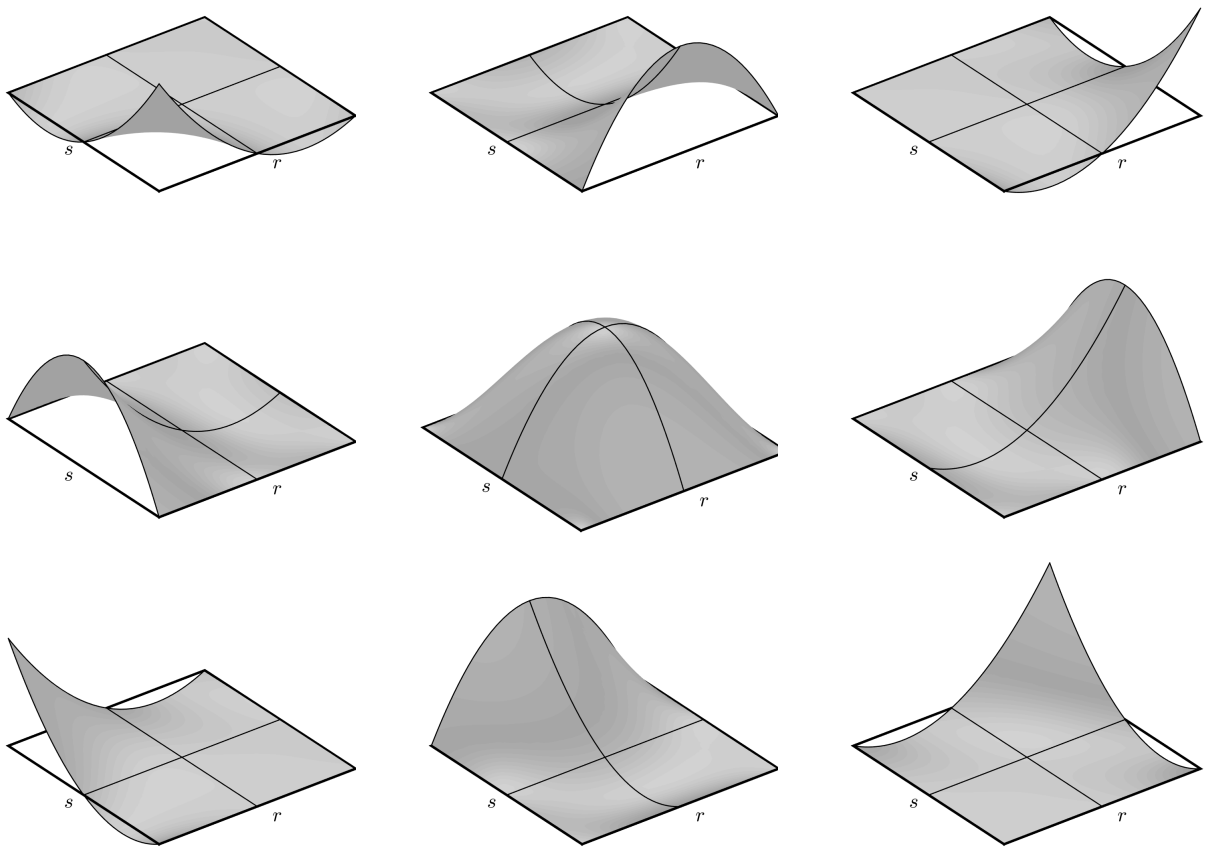


Figure 2: Linear basis functions on the unit-triangle.

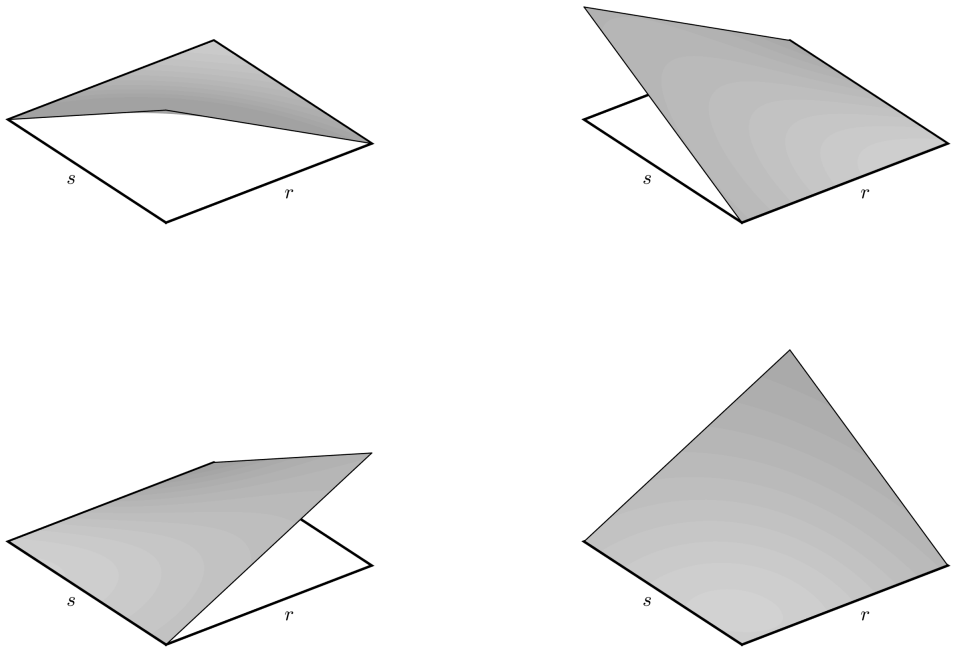
Biquadratic and Bilinear Elements

- We can also have *quadrilateral* elements, such as bilinear, biquadratic, or, in general, of order N, which can be arbitrarily high as is the case for the spectral element method (SEM) [Patera, 84].

- The figure below shows the nine biquadratic basis functions for (r, s) in the unit square, $\hat{\Omega} := [-1, 1]^2$.



- The *bilinear* counterparts to the biquadratic elements are illustrated in the figure below.



- If we enumerate the local dofs lexicographically on $\hat{\Omega} := [-1,1]^2$ as $[u_{00}\ u_{10}\ u_{01}\ u_{11}]$, then the bilinear interpolant takes the form,

$$u(r,s) \;=\; \sum_{j=0}^{1}\sum_{i=0}^{1} l_i(r)\, l_j(s)\, u_{ij}, \tag{57}$$

with the linear 1D interpolants defined as,

$$l_0(r) \;=\; \frac{1-r}{2}, \quad l_1(r) \;=\; \frac{1+r}{2}. \tag{58}$$

Enforcing Function Continuity in the FEM

- Before returning to the FEM derivation, we remark that the *global basis* functions have n dofs, where typically $n \ll En_v$.

- One of the major considerations when implementing the FEM is to ensure that functions are *continuous*, which puts constraints on the *local basis* coefficients, u_i^e.

- Function continuity is ensured if, for every (e, e') (i, i') pair where

$$\mathbf{x}_i^e \;=\; \mathbf{x}_{i'}^{e'}, \tag{59}$$

  we insist that

$$u_i^e \;=\; u_{i'}^{e'}. \tag{60}$$

- This condition is typically enforced through matrix and vector *assembly*, which we'll discuss later.

- We note, however, that it derives from an element-to-vertex map that comes with most FEM meshes.

- For each element, Ω^e, we associate n_v integers. These integers in turn point to unique *global* vertices, $\mathbf{x}_j$.

- For the case of triangles, $j = t(e, i)$, $i = 1, \ldots, 3$, would point into an array, $\mathbf{x}(j)$. Both arrays, $\mathbf{x}(:)$ and $t(:, :)$ are provided by *mesh generators*.

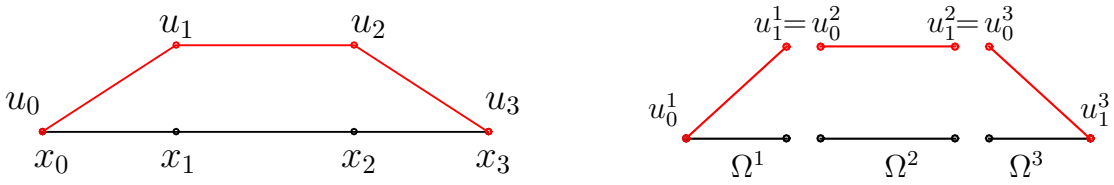
- It will be convenient (in theory and practice) to define a matrix operation that maps from the global index form of a scalar field, $\underline{\bar{u}} = [u_1\ u_2\ \ldots\ u_{\bar{n}}]^T$ to its local counter part, $\underline{u}_L := [u_1^1\ \ldots u_{n_v}^1 \ldots u_i^e \ldots u_{n_v}^E]^T$.

- We'll denote this map by a sparse Boolean matrix (comprising only 1s and 0s), Q, such that

$$\underline{u}_L \;=\; Q\underline{\bar{u}}. \tag{61}$$

- Note that this map includes the boundary dofs—we typically distinguish between nodal values on $\partial\Omega_D$ and those in $\Omega\backslash\partial\Omega_D$ only at the end of the problem setup in order to support a variety of boundary conditions for different fields.

- An illustration of the global-local representations of $u(x)$ for linear 1D elements with $E = 3$ is given in the accompanying figure.



- For this example, there are four inputs for $\underline{u}$ and six outputs for $\underline{u}_L$.

- The explicit form of the matrix vector product $\underline{u}_L = Q\underline{u}$ is

$$\underline{u}_L = \begin{pmatrix} u_0^1 \\ u_1^1 \\ u_0^2 \\ u_1^2 \\ u_0^3 \\ u_1^3 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & 1 & & \\ & & 1 & \\ & & 1 & \\ & & & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix} = Q\underline{u}. \qquad (62)$$

- Notice that the role of the columns of Q is to *copy* data from elements of the global vector to their local counterparts.

- Conversely the corresponding role of Q^T is to *sum* local counterparts.

- Given the "$t()$" matrix introduced earlier, it is possible to construct the matrix Q in matlab with the statement:

```
Q  = sparse(1:nl,reshape(t',nl,1),1);
```

- In matlab it is probably faster to evaluate $\underline{u}_L = Q\underline{u}$ than to execute a `for` loop, which is what would be preferred in Fortran or C.

FEM Formulation: Galerkin/Variational Projection

- The FEM postulates solutions of the type (47), which is equivalent to (48), modulo constraints to be defined, and seeks to find the vector of basis coefficients $\underline{u} = [u_1\ u_2\ \ldots\ u_{\bar{n}}]^T$ such that

$$\|\tilde{u}\ -\ u\|_* \tag{63}$$

  is minimized, where $\tilde{u}(\mathbf{x})$ is the unknown exact solution to the PDE and $\|\cdot\|_*$ is an appropriate norm.

- To illustrate the ideas, we'll start with the Poisson problem,

$$-\nabla^2\tilde{u}\ =\ f \text{ in } \Omega,\ \ \tilde{u} = u_b \text{ on } \partial\Omega_D,\ \ \nabla\tilde{u}\cdot\hat{\mathbf{n}} = 0 \text{ on } \partial\Omega_N. \tag{64}$$

- We'll minimize the error in the a-norm,

$$a(v,u)\ :=\ \int_\Omega \nabla v\cdot\nabla u\, dV \tag{65}$$

$$\|u\|_a\ :=\ \left[\int_\Omega \nabla u\cdot\nabla u\, dV\right]^{\frac{1}{2}}\ =\ \sqrt{a(u,u)}. \tag{66}$$

- Assume that

$$u,\,\tilde{u}\ \in\ \mathcal{H}^1_b\ =\ \left\{v\ |\, v = u_b \text{ on } \partial\Omega_D,\ \int_\Omega \nabla v\cdot\nabla v\, dV\ <\ \infty,\ \ \int_\Omega v^2\, dV\ <\ \infty\right\}.$$

- Let's also introduce the space $\mathcal{H}^1_0$,

$$v\ \in\ \mathcal{H}^1_0\ =\ \left\{v\ |\, v = 0 \text{ on } \partial\Omega_D,\ \int_\Omega \nabla v\cdot\nabla v\, dV\ <\ \infty,\ \ \int_\Omega v^2\, dV\ <\ \infty\right\}, \tag{}$$

  and $\mathcal{H}^1$,

$$v\ \in\ \mathcal{H}^1\ =\ \left\{v\ |\ \int_\Omega \nabla v\cdot\nabla v\, dV\ <\ \infty,\ \ \int_\Omega v^2\, dV\ <\ \infty\right\}. \tag{69}$$

- Note that the *error* $e := \tilde{u} - u$, is in $\mathcal{H}_0^1$ even though $\tilde{u}$ and u are in $\mathcal{H}_b^1$.

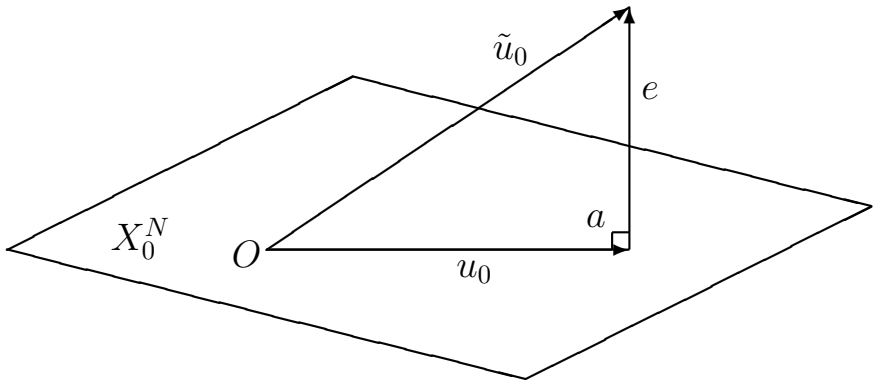
- The defining property of u is that, out of all functions in $X_b^N \subset \mathcal{H}_b^1$, it minimizes the error.

- Specifically, let $w \in X_b^N$ and $v := u - w \in X_0^N$, then

$$\begin{aligned} \|e\|_a^2 \;=\; \|\tilde{u} - u\|_a^2 \;&\leq\; \|\tilde{u} - w\|_a^2 && (70)\\ &=\; \|\tilde{u} - (u - v)\|_a^2 && (71)\\ &=\; \|e + v\|_a^2 && (72)\\ &=\; a(e,e) + 2a(v,e) + (v,v). && (73) \end{aligned}$$

- Clearly, we need $a(v, e) = 0$ if the inequality is to hold for any possible choice of $v \in X_0^N$.

- Once again we have equivalence between minimization of $\|e\|_a$ and orthogonality of e to the search space, X_0^N, as illustrated in the accompanying figure.



- From the orthogonality relationship we can derive an explicit formula for u,

$$0 \;=\; a(v,e) \;=\; a(v, \tilde{u} - u) \;\Longrightarrow\; a(v,u) \;=\; a(v,\tilde{u}) \;\; \forall\, v \in X_0^N. \qquad (74)$$

- The statement on the right in (74) is our standard projection form.

- It says that we are seeking an n-dimensional object in an n-dimensional space, X_0^N, and that we have n test conditions (e.g., take $v = \phi_i$, $i = 1, \ldots, n$).

- This n-dimensional object, u, will match the infinite-dimensional object, $\tilde{u}$, under these n test conditions or any linear combination of them.

- To generate a *computable* solution, we modify both the right- and left-hand sides of (74).
- Starting with the right, we integrate by parts as follows.

$$a(v,u) \;=\; a(v,\tilde{u}) \;=\; \int_\Omega \nabla v \cdot \nabla \tilde{u}\, dV \tag{75}$$

$$= \; -\int_\Omega v\, \nabla^2 \tilde{u}\, dV \;+\; \int_{\partial\Omega} v \nabla \tilde{u} \cdot \hat{\mathbf{n}}\, dS \tag{76}$$

$$= \; \int_\Omega v\, f\, dV \;+\; \int_{\partial\Omega_D} v \nabla \tilde{u} \cdot \hat{\mathbf{n}}\, dS \;+\; \int_{\partial\Omega_N} v \nabla \tilde{u} \cdot \hat{\mathbf{n}}\, dS \tag{77}$$

$$= \; \int_\Omega v\, f\, dV \tag{78}$$

- In (77), the surface integral on $\partial\Omega_D$ vanishes because v vanishes there ($v \in X_0^N$).
- The surface integral on $\partial\Omega_N$ vanishes because of the homogeneous Neumann condition on $\tilde{u}$ given in (64).
- The steps (75)–(78) allow us to eliminate the unknown $\tilde{u}$ in exchange for things we do know, namely, $f(\mathbf{x})$ and the boundary conditions.
- The complete statement of the variational/Galerkin problem statement is,

Find $u \in X_b^N \subset \mathcal{H}_b^1$ such that, for all $v \in X_0^N$,

$$a(v,u) \;=\; (v,f) \tag{79}$$

- The next steps will lead us to a linear system for the unknown basis coefficients.

- Let

$$\phi_1, \ldots, \phi_n \;=\; 0 \;\text{ on } \partial\Omega_D \tag{80}$$

$$\phi_{n+1}, \ldots, \phi_{\bar{n}} \;\neq\; 0 \;\text{ on } \partial\Omega_D. \tag{81}$$

  Recall that since we are working with Lagrangian interpolants the above classification amounts to enumerating grid points in the interior *and* on the Neumann boundary, $\partial\Omega_N$, *first* and those on $\partial\Omega_D$ *last*.

- In practice, this renumbering is not necessary and typically not done—but
  it simplifies the derivation below.

- At the risk of overloading the notation, we will define

$$u_b(\mathbf{x}) \;:=\; \sum_{j=n+1}^{\bar{n}} u_b(\mathbf{x}_j)\phi_j(x) \;\in\; X_b^N \;\subset\; \mathcal{H}_b^1. \tag{82}$$

- Note that (82) is, in effect a lifting function that extends the known trace (boundary) data, $u_b(\mathbf{x} \in \partial\Omega_D)$ to a function defined in X_b^N.

- We further define the decomposition,

$$u(\mathbf{x}) \;=\; u_0(\mathbf{x}) \;+\; u_b(\mathbf{x}), \tag{83}$$

  which we insert into the bilinear form,

$$a(v, u) \;=\; a(v, u_0) \;+\; a(v, u_b). \tag{84}$$

- We next use the global expansions to express the continuous functions in terms of a set of discrete values.

$$u_0(x) \;=\; \sum_{j=1}^{n} u_j\phi_j(\mathbf{x}) \tag{85}$$

$$u_b(x) \;=\; \sum_{j=n+1}^{\bar{n}} u_j\phi_j(\mathbf{x}) \tag{86}$$

$$v(x) \;=\; \sum_{i=1}^{n} v_i\phi_i(\mathbf{x}). \tag{87}$$

- Note that the *unknowns* are $\underline{u} = [u_1\ u_2\ \ldots\ u_n]^T$, as the boundary coefficients, u_j, $j > n$ are *known*.

- We now derive the system matrix.

- Consider the bilinear form,

$$a(\bar{v}, u) = a\left(\sum_{i=1}^{\bar{n}} \phi_i(\mathbf{x})\bar{v}_i, \sum_{j=1}^{\bar{n}} \phi_j(\mathbf{x})u_j\right) \tag{88}$$

$$= \sum_{i=1}^{\bar{n}}\sum_{j=1}^{\bar{n}} \bar{v}_i\, a(\phi_i, \phi_j)\, u_j \tag{89}$$

$$= \sum_{i=1}^{\bar{n}}\sum_{j=1}^{\bar{n}} \bar{v}_i\, \bar{a}_{ij}\, u_j = \underline{\bar{v}}^T \bar{A}\, \underline{u}. \tag{90}$$

- Here, we have defined the temporary function, $\bar{v}$, which is allowed to be nonzero on $\partial\Omega_D$.

- Likewise, the system matrix $\bar{A}$ has an index range that spans $i, j \in \{1, \ldots, \bar{n}\}^2$.

- As it accounts for dofs on *all* of $\partial\Omega$ we refer to it as the *Neumann* operator.

- $\bar{A}$ is singular because $\bar{A}\underline{1} = 0$. (The nullspace comprises the constant vector.)

- We recover an *invertible* operator as follows.

- Recall that the solution we seek, u_0, and the test functions, v, are in X_0^N.

- Let $\underline{u}_0 \in \mathbb{R}^n$ represent the unknowns in $\Omega\backslash\partial\Omega_D$, and define a *prolongation matrix*, R^T that extends $\underline{u}_0$ to have 0 values for all basis functions associated with $\partial\Omega_D$. (In our case, that implies zeros in rows $j > n$.)

- We write this extended vector as

$$\bar{\underline{u}}_0 \;=\; R^T \underline{u}_0. \tag{91}$$

- With this definition, we can assert that

$$\bar{u}_0(x) \;=\; \sum_{j=1}^{\bar{n}} \bar{u}_{0,j}\, \phi_j(\mathbf{x}) \;\in\; X_0^N. \tag{92}$$

- Similarly, for any $\underline{v} \in \mathbb{R}^n$, $\bar{\underline{v}} = R^T \underline{v}$ represents a function in $v \in X_0^N$.

- Thus, our variational statement reads, *Find* $u_0 \in X_0^N \subset \mathcal{H}_0^1$ *such that, for all* $v \in X_0^N$,

$$a(v, u_0) \;=\; (v, f) \;-\; a(v, u_b). \tag{93}$$

- From our definitions, we have

$$a(v, u_0) \;=\; \bar{\underline{v}}^T \bar{A} \bar{\underline{u}}_0 \;=\; (R^T \underline{v})^T \bar{A} (R^T \underline{u}_0) \;=\; \underline{v}^T (R \bar{A} R^T) \underline{u}_0 \;=\; \underline{v}^T A \underline{u}_0, \tag{94}$$

  where $A = R\bar{A}R^T$ is our invertible (SPD) system matrix.

- Following through in a similar way for the terms on the right of (93) leads
  to the linear system

$$\underline{v}^T A \underline{u}_0 \;=\; \underline{v}^T R\, (\bar{B} \bar{\underline{f}} \;-\; \bar{A} \underline{u}_b), \tag{95}$$

  which holds for all $\underline{v} \in \mathbb{R}^n$. Since A is invertible, this implies simply,

$$A \underline{u}_0 \;=\; R\, (\bar{B} \bar{\underline{f}} \;-\; \bar{A} \underline{u}_b). \tag{96}$$

- In the preceding equations, we have introduced the *mass matrix*,

$$\bar{B}_{ij} \;=\; \int_\Omega \phi_i\, \phi_j\, dV. \tag{97}$$

FEM: Element-Based Implementation

- We see that the FEM relies on integration.

- Fortunately, integration is relatively easy.

- Let's ignore the boundary conditions for the moment and just consider functions v and u in $X^N :=\text{span}\{\phi_1 \ \ldots \ \phi_{\bar{n}}\}$.

- We have, for all $v,\, u \ \in \ X^N$,

$$
\begin{aligned}
(v,u) \ &= \ \int_\Omega v\, u\, dV && (98)\\
&= \ \sum_{e=1}^{E} \int_{\Omega^e} v\, u\, dV && (99)\\
&= \ \sum_{e=1}^{E} \int_{\hat{\Omega}} \left(\sum_{i=1}^{n_v} v_i^e\, l_i(\mathbf{r}) \right) \left(\sum_{j=1}^{n_v} u_j^e\, l_j(\mathbf{r}) \right) \mathcal{J}^e(\mathbf{r}) d\mathbf{r} && (100)\\
&= \ \sum_{e=1}^{E} \sum_{i=1}^{n_v} \sum_{j=1}^{n_v} v_j^e \left(\int_{\hat{\Omega}} \ l_i(\mathbf{r})\, l_j(\mathbf{r})\, \mathcal{J}^e(\mathbf{r}) d\mathbf{r} \right) u_i^e && (101)\\
&= \ \sum_{e=1}^{E} \sum_{i=1}^{n_v} \sum_{j=1}^{n_v} v_j^e B_{ij}^e u_i^e && (102)\\
&= \ \sum_{e=1}^{E} (\underline{v}^e)^T B^e \underline{u}^e. && (103)
\end{aligned}
$$

- Here, $B_{ij}^e := \int_{\hat{\Omega}} \ l_i(\mathbf{r})\, l_j(\mathbf{r})\, \mathcal{J}^e(\mathbf{r}) d\mathbf{r}$ is the *local* mass matrix, and $\underline{u}^e$ is the local vector of unknown basis coefficients.

- Recall that $\underline{u}_L = [\underline{u}^1 \ \underline{u}^2 \ \ldots \ \underline{u}^E]$ is the vector containing vectors of *local* basis coefficents,

$$\underline{u}^e \ = \ \left[u_1^e \ u_2^e \ \ldots \ u_{n_v}^e \right]^T. \qquad (104)$$

- Since we require $u(x) \in X^N \subset \mathcal{H}^1$ to be continuous, we recall from (61) that a vector $\underline{u}_L = Q\bar{\underline{u}}$ will have the correct continuity requirements.

- Therefore, with $B_L :=$block-diagonal(B^e), we have the following equivalence, *For any* $v,\ u \in X^N$,

$$(v,u) \ = \ (Q\bar{\underline{v}})^T B_L (Q\bar{\underline{u}}) \ = \ \bar{\underline{v}}^T Q^T B_L Q \bar{\underline{u}} \ = \ \bar{\underline{v}}^T \bar{B} \bar{\underline{u}}, \tag{105}$$

  where $\bar{B} := Q^T B_L Q$ is the *assembled* mass matrix.

- We likewise have the assembled stiffness (system) matrix,

$$\bar{A} := Q^T A_L Q, \tag{106}$$

  with $A_L :=$block-diagonal(A^e), and $A^e_{ij} := \displaystyle\int_{\hat{\Omega}} \sum_{k=1}^{d} \frac{\partial l_i}{\partial x_k} \frac{\partial l_j}{\partial x_k} \mathcal{J}^e(\mathbf{r}) d\mathbf{r}$.

- Here, we have cheated slightly by using $\frac{\partial l_i}{\partial x_k}$, by which we mean that one must apply the chain rule to express the derivatives in (r,s) coordinates.

- Fortunately, for linear triangles, there are relatively simple expressions (still about 30 lines in matlab) to generate A^e and B^e, and these, coupled with Q and R are all we need to set up the FEM system.

- Combining all of these tools together, our FEM system reads,

$$A\underline{u}_0 \ = \ R\,(\bar{B}\underline{\bar{f}} \ - \ \bar{A}\underline{u}_b) \tag{107}$$

$$\underline{u} \ = \ R^T \underline{u}_0 \ + \ \underline{u}_b, \tag{108}$$

  with

$$A \ = \ R(Q^T A_L Q) R^T \tag{109}$$

$$\bar{B} \ = \ Q^T B_L Q. \tag{110}$$

- Since $\bar{A}$, $\bar{B}$, and Q depend only on the mesh geometry and topology, it is convenient to have a utility that computes those for any set of triangles. Such code will be provided.

- Finally, we remark that all of this notation extends to the 3D case. One only needs new utilities for computing (A^e, B^e).