1D Model Problem: Element Formation:

In the last lecture, we began to discuss computer implementation of Bubnov-Galerkin approximations of the variational form of our 1D model problem. In particular, we discussed how to build the global stiffness matrix and force vector by assembling element stiffness matrices and force vectors. Today, we discuss how to compute the entries of these matrices and vectors in a computer program.

To begin, recall the entries of the element stiffness matrices are:

$$k^e_{ab} = \int_{\Omega^e} K N^e_{a,x} N^e_{b,x} \, dx \qquad a,b = 1,\dots,k+1 \text{ and } e = 1,\dots,n_{el}$$

and the entries of the element force vectors are:

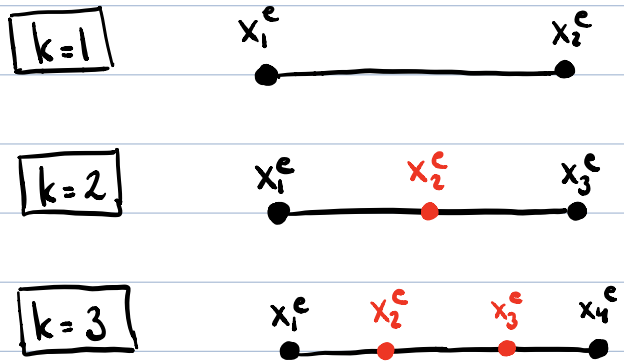
$$f^e_a = \int_{\Omega^e} f N^e_a \, dx \qquad a = 1,\dots,k+1 \text{ and } e = 1,\dots,n_{el}$$

The elements may be of arbitrary size and location, and in the multi-dimensional setting, the elements may be of arbitrary shape too. Thus, to simplify evaluation of element integrals, it is common to pull back these integrals to a canonical parent or reference element. To explain this idea, let us isolate our view to a single element Ω^e. There are $k+1$ nodes associated with this element, and just like we did with the basis functions, we can associated a local numbering with

these nodes:

$$x_a^e = x_A \qquad \text{where} \quad A = \text{IEN}(a,e) \quad \text{and} \quad a = 1, \dots, k+1$$

Visually:



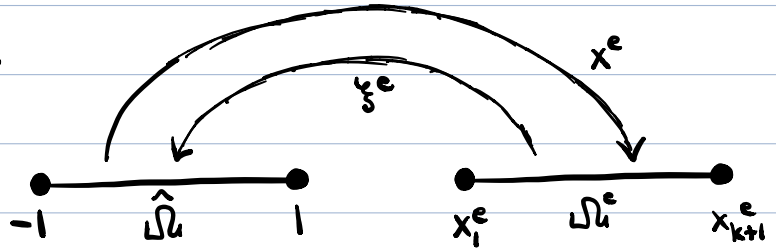
Now define the map $\xi^e : \Omega^e \to \hat{\Omega}$ by:

$$\xi^e(x) = -1 + 2\left(\frac{x - x_1^e}{x_{k+1}^e - x_1^e}\right)$$

where $\hat{\Omega} = (-1,1)$. The map above map is invertible with inverse $x^e : \hat{\Omega} \to \Omega^e$ defined by:

$$x^e(\xi) = x_1^e + (x_{k+1}^e - x_1^e)\left(\frac{\xi + 1}{2}\right)$$

Visually:



Thus we can express any integral over Ω_h^e as an integral over $\hat{\Omega}_h$ or vice versa using a change of variables. In particular, we can write out the element stiffness matrices and force vectors in terms of integrals over $\hat{\Omega}_h$:

$$k_{ab}^e = \int_{\hat{\Omega}_h} K(x^e(\xi))\, N_{a,x}^e(x^e(\xi))\, N_{b,x}^e(x^e(\xi))\, x_{,\xi}^e \, d\xi$$

$$f_a^e = \int_{\hat{\Omega}_h} f(x^e(\xi))\, N_a^e(x^e(\xi))\, x_{,\xi}^e \, d\xi$$

As previously alluded to, $\hat{\Omega}_h$ is referred to as the parent or reference element. We can also pull back our basis functions to the parent element. Remarkably, the local a^{th} basis function over each element is always pulled back to the same function:

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

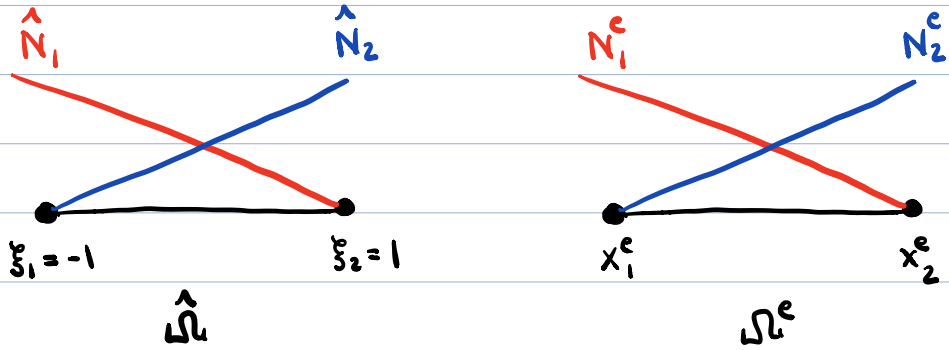
where:

$$\hat{N}_a(\xi) = \prod_{\substack{b=1 \\ b \neq a}}^{k+1} \left(\frac{\xi_b - \xi}{\xi_b - \xi_a} \right)$$

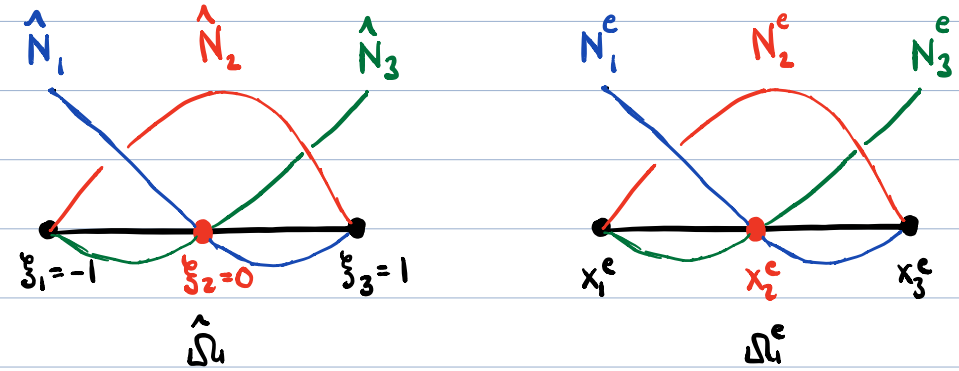
and $\xi_a = -1 + 2\left(\frac{a-1}{k}\right)$ are equispaced nodes on the parent element. It holds that $\xi_a = \xi^e(x_a^e)$, that is, ξ_a is the pullback of node x_a^e to the parent element.

Visually:

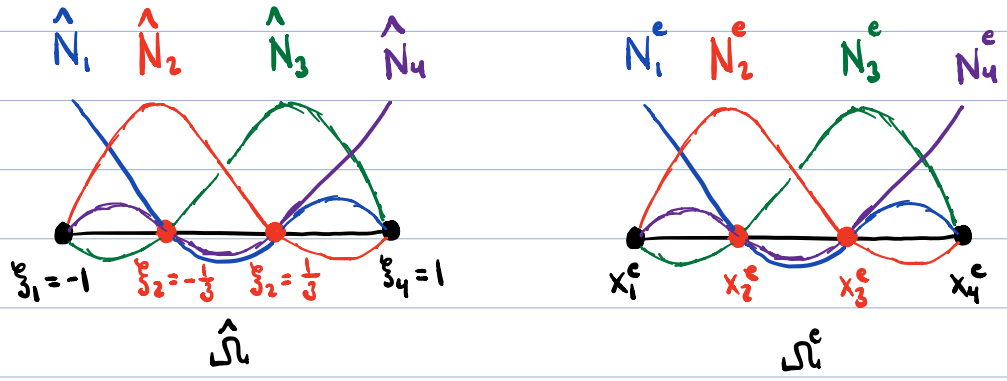
k=1



k=2



k=3



The functions $\{\hat{N}_a\}_{a=1}^{k+1}$ are commonly referred to as shape functions.

Note that, by the chain rule, we can also write the derivatives of the basis functions in physical space $\{N_a^e\}_{a=1}^{k+1}$ with respect to x in terms of derivatives of shape functions with respect to ξ.

In particular:

$$\hat{N}_{a,\xi}(\xi) = N^e_{a,x}(x^e(\xi))\, x^e_{,\xi}(\xi)$$

So:

$$N^e_{a,x}(x^e(\xi)) = \hat{N}_{a,\xi}(\xi) \,/\, x^e_{,\xi}(\xi)$$

Thus, noting:

$$x^e_{,\xi}(\xi) = \frac{x^e_{k+1} - x^e_1}{2} = \frac{h^e}{2}$$

we can write the entries of the element stiffness matrices and force vectors as:

$$k^e_{ab} = \frac{2}{h^e} \int_{\hat{\Omega}} K(x^e(\xi))\, \hat{N}_{a,\xi}(\xi)\, \hat{N}_{b,\xi}(\xi)\, d\xi$$

Element Specific — Generic

$$f^e_a = \frac{h^e}{2} \int_{\hat{\Omega}} f(x^e(\xi))\, \hat{N}_a(\xi)\, d\xi$$

Note we have removed all element-specific information associated with computation of the basis functions or their derivatives. However, we are still left with the task of evaluating integrals over the parent element. In general, K and f can be quite nonlinear, rendering integration a difficult task. To overcome this issue, we turn to numerical integration of the integrals appearing above. In particular, we turn to Gaussian quadrature.

The n_q-point Gaussian quadrature approximation of the integral of a function $g: (-1,1) \to \mathbb{R}$ takes the form:

$$\int_{-1}^{1} g(\xi) d\xi \approx \sum_{q=1}^{n_q} g(\tilde{\xi}_q) \tilde{w}_q$$

where $\tilde{\xi}_q$ is the q^{th} root of the n_q^{th} Legendre polynomial:

$$P_{n_q}(\xi) = \frac{1}{2^{n_q} n_q!} \frac{d^{n_q}}{d\xi^{n_q}} (\xi^2 - 1)^{n_q}$$

and:

$$\tilde{w}_q = \frac{2}{(1 - \tilde{\xi}_q^2)\left(\frac{d}{d\xi} P_{n_q}(\tilde{\xi}_q)\right)^2}$$

The n_q-point Gaussian quadrature rule is exact for polynomials of degree $2n_q - 1$, and it converges exponentially fast for smooth integrands. The quadrature points can only be attained in general

using numerical root-finding. Fortunately, however, they are tabulated for a wide range of values of n_q. For instance, for $1 \le n_q \le 4$:

Number of Points, n_q	Quadrature Points, $\tilde{\xi}_q$	Quadrature Weights, $\tilde{w}_q$
1	0	2
2	$\pm \frac{1}{\sqrt{3}}$	1
3	0	$\frac{8}{9}$
	$\pm \sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm \sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm \sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$

More rules can be found in Abramowitz and Stegun's "Handbook of Mathematical Functions", up to $n_q = 256$.

With an n_q-point Gauss quadrature rule applied, we arrive at the following approximations of the entries of the element stiffness matrices and force vectors:

$$k_{ab}^e \approx \frac{2}{h^e} \sum_{q=1}^{n_q} K(x^e(\tilde{\xi}_q)) \hat{N}_{a,\xi}(\tilde{\xi}_q) \hat{N}_{b,\xi}(\tilde{\xi}_q) \tilde{w}_q$$

$$f_a^e \approx \frac{h^e}{2} \sum_{q=1}^{n_q} f(x^e(\tilde{\xi}_q)) \hat{N}_a(\tilde{\xi}_q) \tilde{w}_q$$

and the above approximations are easy to implement in a computer program even for complicated K and f. See, for example, the below pseudocode.

Element Formation Pseudocode

- Initialize Quantities

Set $\underline{\underline{k}}^e = \underline{\underline{0}}$ and $\underline{f}^e = \underline{0}$

- Loop Over Quad Points

for $q = 1 : n_q$

- Evaluate Local Quantities

Evaluate $x^e(\tilde{\xi}_q)$, $K(x^e(\tilde{\xi}_q))$, and $f(x^e(\tilde{\xi}_q))$

- Loop Over Rows

for $a = 1 : k+1$

- Loop Over Columns

for $b = 1 : k+1$

- Update $\underline{\underline{k}}^e$

$$k_{ab}^e = k_{ab}^e + K(x^e(\tilde{\xi}_q)) * \hat{N}_{a,\xi}(\tilde{\xi}_q) * \hat{N}_{b,\xi}(\tilde{\xi}_q) * \tilde{w}_q * \frac{2}{h^e}$$

endfor

- Update $\underline{f}^e$

$$f_a^e = f_a^e + f(x^e(\tilde{\xi}_q)) * \hat{N}_a(\tilde{\xi}_q) * \tilde{w}_q * \frac{h^e}{2}$$

endfor

endfor

Provided the same quadrature rule is applied over each element, the shape function values and derivatives can be pre-computed, stored, and accessed in the above element formation subroutine rather than re-computed for each individual element. This is one pronounced advantage of the finite element basis construction. The quadrature

points and weights can also be determined from tables, stored, and accessed as required.

There is no hard and fast rule for how many quadrature points should be employed for a particular finite element approximation, provided enough points are employed so that the global stiffness matrix is of full rank (so that Problem (L) is well-posed). A general rule of thumb is to employ $n_q = k+1$ quadrature points per element.

With an element formation subroutine in hand, we can now present the overall control flow associated with a finite element program exploiting element formation and assembly: