1D Model Problem: Element Assembly:

In the last lecture, we discussed well-posedness and convergence of Bubnov-Galerkin finite element approximations of our 1D model problem. Today, we begin to discuss efficient computer implementation. In particular, we will discuss how we can construct the global stiffness matrix and force vector for a particular finite element approximation by assembling element stiffness matrices and forcing vectors.

To begin, recall that the Bubnov-Galerkin approximation of the variational form of our 1D model problem takes the form:

$$(G)\quad \begin{cases} \text{Find } u^h \in \mathcal{V}^h + g^h \text{ such that} \\ \qquad \int_0^L K u^h_{,x} w^h_{,x}\, dx = \int_0^L f w^h\, dx \\ \text{for all } w^h \in \mathcal{V}^h. \end{cases}$$

As discussed previously, the Galerkin solution of the above problem is:

$$u^h(x) = \sum_{B=1}^{n} d_B N_B(x) + g^h(x)$$

where the unknown degrees of freedom $\{d_B\}_{B=1}^n$ may be attained by solving the matrix problem:

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

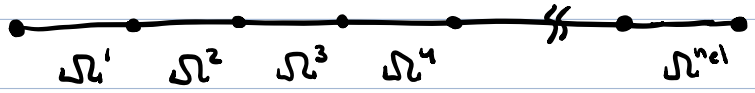
where:

$$K_{AB} = \int_0^L k\, N_{A,x}\, N_{B,x}\, dx$$

$$F_A = \int_0^L f\, N_A\, dx - \int_0^L k\, N_{A,x}\, g^h_{,x}\, dx$$

denote the components of the stiffness matrix $\underline{\underline{K}}$ and force vector $\underline{F}$, respectively. Thus, a key aspect of any efficient computer implementation is efficient evaluation of the entries of the stiffness matrix and forcing vector. The most efficient finite element computer implementations exploit the local support of finite element basis functions by first computing local element-wise contributions to the stiffness matrix and force vector before assembling these into the global system.

Recall that we decompose the domain $\Omega_L = (0, L)$ into n_{el} non-overlapping elements $\{\Omega^e\}_{e=1}^{n_{el}}$ with a finite element method:



We can thus express integrals appearing in the entries of the stiffness matrix and force vector as sums of integrals over elements:

$$K_{AB} = \sum_{e=1}^{nel} \int_{\Omega^e} K N_{A,x} N_{B,x} \, dx$$

$$F_A = \sum_{e=1}^{nel} \left(\int_{\Omega^e} f N_A \, dx - \int_{\Omega^e} K N_{A,x} g^h_{,x} \, dx \right)$$

Define:

$$K^e_{AB} = \int_{\Omega^e} K N_{A,x} N_{B,x} \, dx$$

$$F^e_A = \int_{\Omega^e} f N_A \, dx - \int_{\Omega^e} K N_{A,x} g^h_{,x} \, dx$$

Then:

$$K_{AB} = \sum_{e=1}^{nel} K^e_{AB} \quad \text{and} \quad F_{AB} = \sum_{e=1}^{nel} F^e_A$$

This inspires the following two-step procedure for constructing the stiffness matrix and force vector:

Step 1 (Formation): First, form $\underline{\underline{K}}^e$ and $\underline{F}^e$ for $e = 1, \ldots, n_{el}$.

Step 2 (Assembly): Second, assemble $\underline{\underline{K}}$ and $\underline{F}$ by summing $\underline{\underline{K}}^e$ and $\underline{F}^e$ over $e = 1, \ldots, n_{el}$.

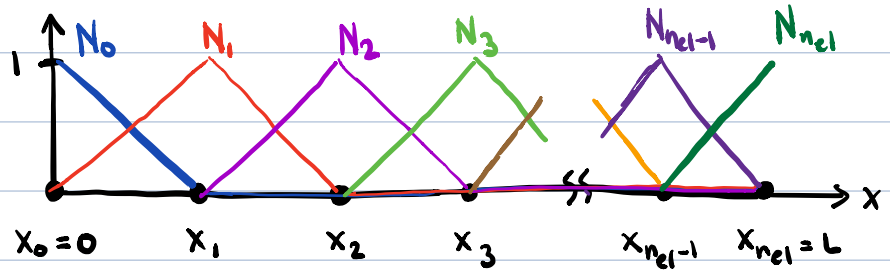
The particular appeal of this two-step approach is:

(i) The integrals appearing in the definition of $\underline{\underline{K}}^e$ and $\underline{F}^e$ are easier to evaluate than the integrals appearing in the definition of $\underline{\underline{K}}$ and $\underline{F}$. This is because finite element functions are simply polynomials over individual elements.

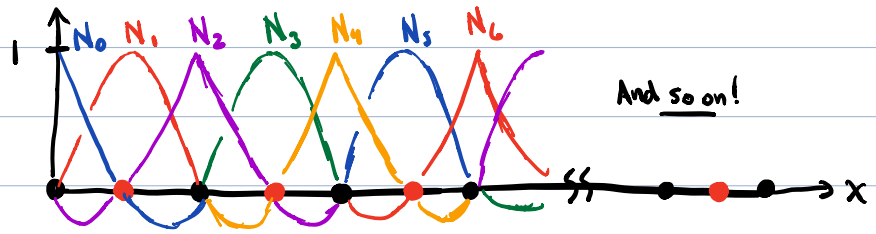
(ii) The matrices $\underline{\underline{K}}^e$ and $\underline{F}^e$ are even sparser than $\underline{\underline{K}}$ and $\underline{F}$. In particular, they have a fixed number of nonzeros regardless of mesh resolution.

To illustrate the second point, recall that only k+1 finite element basis functions are nonzero over a given element:

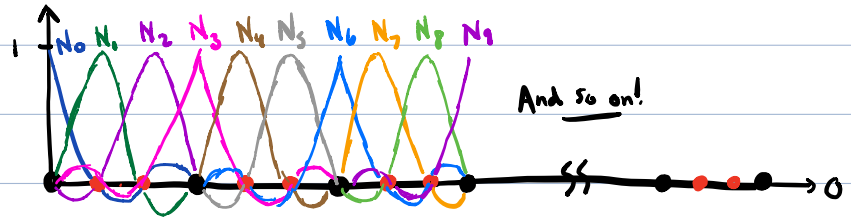
k=1: Two Nonzero Basis Functions Per Element



k=2: Three Nonzero Basis Functions Per Element



k=3: Four Nonzero Basis Functions Per Element



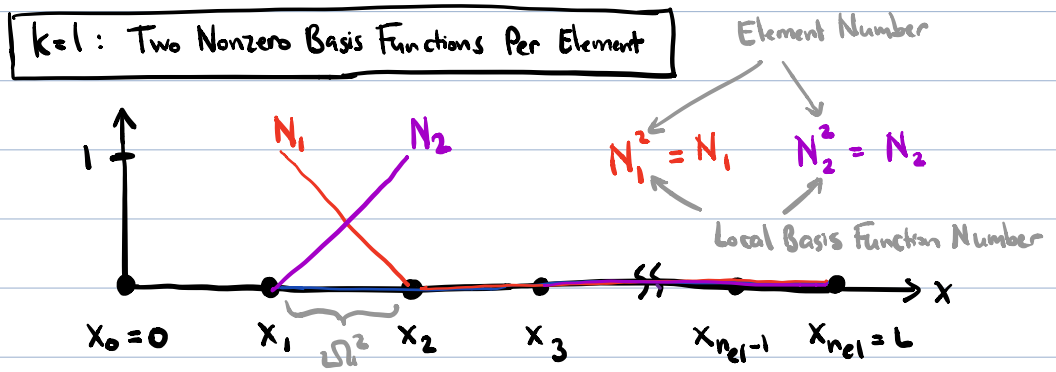
It holds that $K^e_{AB} = 0$ if either $N_A \equiv 0$ or $N_B \equiv 0$ over element e, so $\underline{\underline{K}}^e$ has <u>at most</u> $(k+1)\times(k+1)$ nonzero entries. In fact, $\underline{\underline{K}}^e$ has precisely $(k+1)\times(k+1)$ nonzero entries except when $e=1$ or $e=n_{el}$ in which case it has only $k\times k$ nonzero entries. Similarly, $N^e_A = 0$ if $N_A \equiv 0$ over element e, so $\underline{F}^e$ has <u>at most</u> $k+1$ nonzero entries. Not too surprisingly, $\underline{F}^e$ has exactly $k+1$ nonzero entries when $1<e<n_{el}$ and k nonzero entries when either $e=1$ and $e=n_{el}$.

As $\underline{\underline{K}}^e$ and $\underline{F}^e$ are so sparse, it is far more economical to store their nonzero entries rather than all their entries. In practice, we store alternative representations of these objects, the so-called <u>element stiffness matrix</u> $\underline{\underline{k}}^e$ and <u>force vector</u> $\underline{f}^e$, along with a so-called <u>element connectivity</u>. The element stiffness matrices are $(k+1)\times(k+1)$ and the element forcing vectors are $k\times 1$, so they are easily stored in low-level cache, making for both a memory and operation efficient computer implementation.

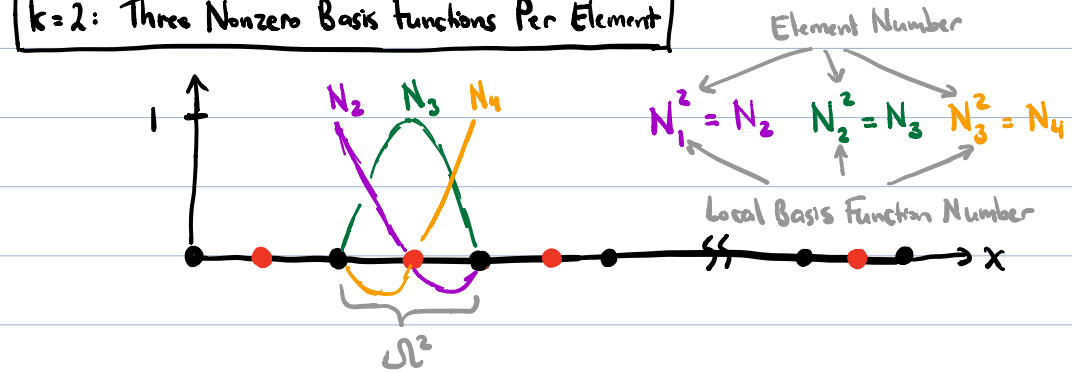
To arrive at a precise definition of the element stiffness matrix and

and force vector, we first need to establish a local numbering of nonzero basis functions for each element. In particular, we number the basis functions from 1 to k+1, starting with the leftmost basis function:

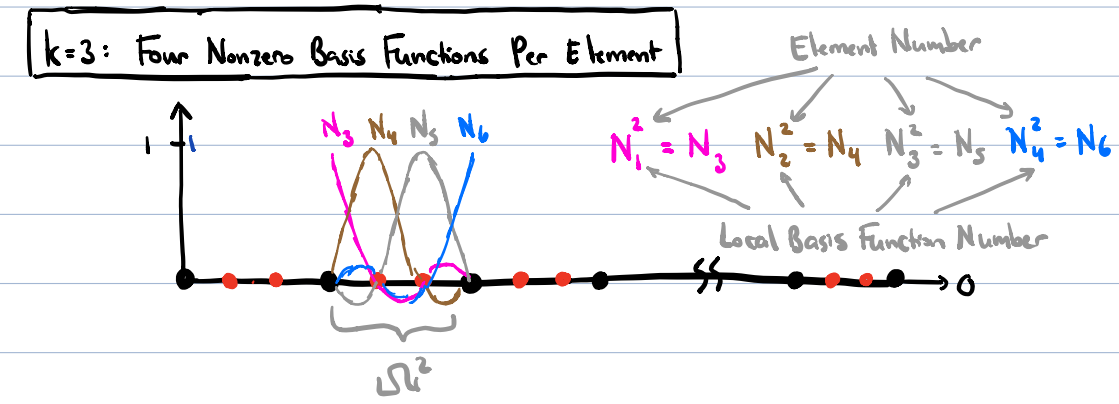
k=1: Two Nonzero Basis Functions Per Element



k=2: Three Nonzero Basis Functions Per Element



k=3: Four Nonzero Basis Functions Per Element



Mathematically:

$$N_a^e(x) = N_A(x)$$

for $a = 1, \ldots, k+1$ and $e = 1, \ldots, n_{el}$ where:

$$A = k * (e-1) + (a-1)$$

Note in the above formula, capital A denotes the global basis function number while lower case a denotes the local basis function number. We utilize this convention throughout the rest of this class to distinguish between local and global quantities. We encode the relation between local and global basis function numbers using an element connectivity:

$$A = IEN(a, e) = k * (e-1) + (a-1)$$

Element Number

Global Basis Function Number/ Degree of Freedom

Local Basis Function Number/ Degree of Freedom

With a local numbering in hand, we are now ready to define the element stiffness matrices and forcing vectors. The element stiffness matrix $\underline{\underline{k}}^e$ is the $(k+1) \times (k+1)$ matrix with entries:

$$k_{ab}^e = \int_{\Omega^e} K N_{a,x}^e N_{b,x}^e \, dx$$

and the element force vector $\underline{f}^e$ is the $(k+1) \times 1$ vector with entries:

$$f_a^e = \int_{\Omega^e} f N_a^e \, dx$$

for $e = 1, \ldots, n_{el}$. Again note the element stiffness matrices and force vectors are denoted with lower case letters as opposed to upper case letters.

Since:

$$g^h(x) = N_0(x)\, g_0 + N_{n+1}(x)\, g_L$$

for Lagrange finite element basis functions, it can be easily shown that for $e = 2, \ldots, n_{el} - 1$:

$$K_{AB}^e = \begin{cases} k_{ab}^e & \text{if there is an } a \text{ and } b \text{ such that} \\ & A = IEN(a,e) \text{ and } B = IEN(b,e) \\ 0 & \text{otherwise} \end{cases}$$

$$F_A^e = \begin{cases} f_a^e & \text{if there is an } a \text{ such that } A = IEN(a,e) \\ 0 & \text{otherwise} \end{cases}$$

that for $e = 1$:

$$K_{AB}^1 = \begin{cases} k_{ab}^1 & \text{if there is an } a \text{ and } b \text{ such that} \\ & A = IEN(a,1) > 0 \text{ and } B = IEN(b,1) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$F_A^1 = \begin{cases} f_a^1 - k_{a1}^1 g_0 & \text{if there is an } a \text{ such that} \\ & A = IEN(a,1) > 0 \\ 0 & \end{cases}$$

and that for $e = n_{el}$:

$$K_{AB}^{n_{el}} = \begin{cases} k_{ab}^{n_{el}} & \text{if there is an } a \text{ and } b \text{ such that} \\ & A = IEN(a, k+1) < n+1 \text{ and } B = IEN(b, k+1) < n+1 \\ 0 & \text{otherwise} \end{cases}$$

$$F_A^{n_{el}} = \begin{cases} f_a^{n_{el}} - k_{a,k+1}^{n_{el}} g_L & \text{if there is an } a \text{ such that} \\ & A = IEN(a, k+1) < n+1 \\ 0 & \end{cases}$$

Since:

$$K_{AB} = \sum_{e=1}^{n_{el}} K_{AB}^{e} \quad \text{and} \quad F_A = \sum_{e=1}^{n_{el}} F_A^{e}$$

We can build the global stiffness matrix and force vector directly from the element stiffness matrices and force vectors. Pseudocode for this construction is included below.

Element Assembly Pseudocode

- Initialize System

Set $\underline{\underline{K}} = \underline{\underline{0}}$ and $\underline{F} = \underline{0}$

- Loop Over Elements

for $e = 1 : n_{el}$

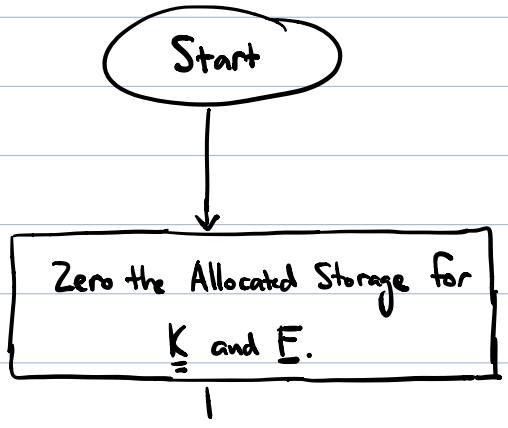
- Loop Over Rows of $\underline{\underline{k}}^e$

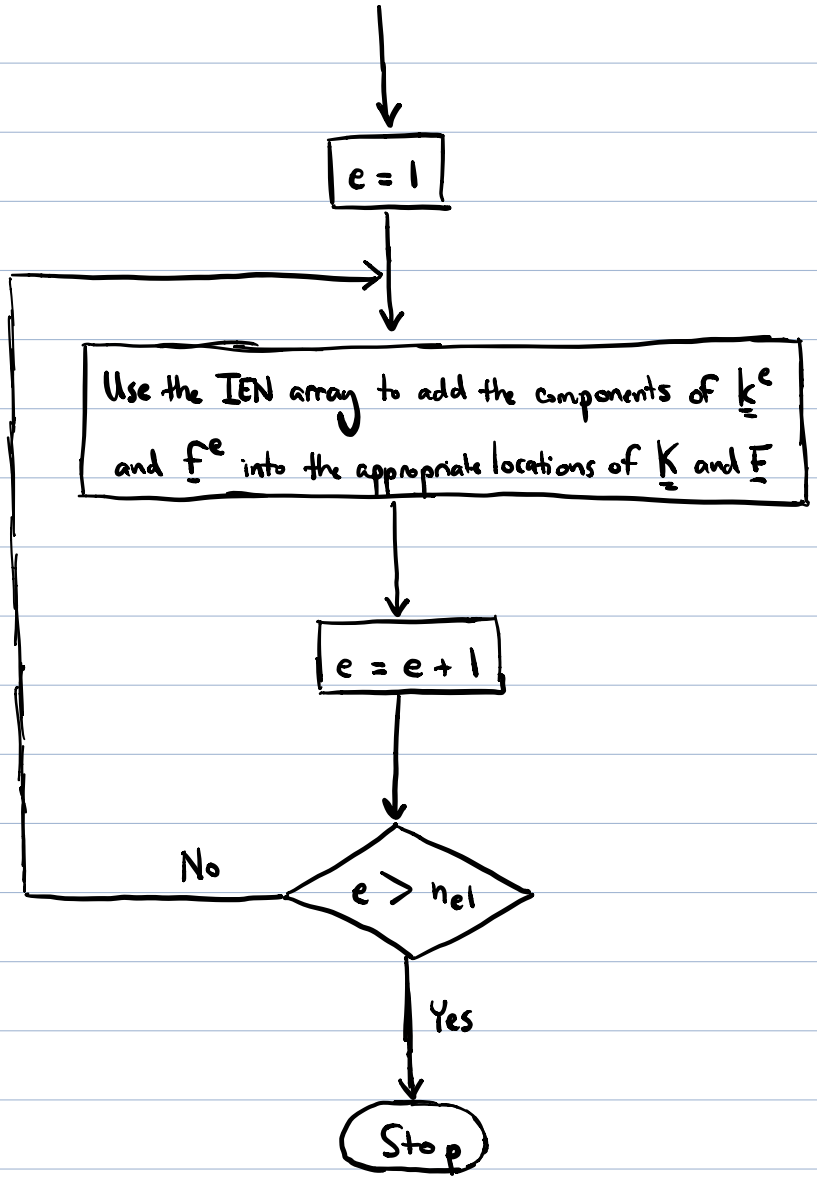
for $a = 1 : k+1$

- Set Global Row
- A = DOF?
- Loop Over Columns of $\underline{\underline{k}}^e$
- Set Global Column
- B = DOF?
- Update $\underline{\underline{K}}^e$
- B = Left BC?
- Update $\underline{F}^e$
- B = Right BC?
- Update $\underline{F}^e$
- Update $\underline{F}^e$

```
Set A = IEN(a,e)
if 1 ≤ A ≤ n
    for b = 1:k+1
        Set B = IEN(b,e)
        if 1 ≤ B ≤ n
            K_AB = K_AB + k^e_ab
        else if B = 0
            F_A = F_A - k^e_ab g_0
        else if B = n+1
            F_A = F_A - k^e_ab g_L
        endif
    endfor
    F_A = F_A + f^e_a
endif
end for
endfor
```

In a flowchart:





The above algorithm is referred to as element assembly, and it is typically denoted by $\mathbb{A}$:

$$\underline{\underline{K}} = \mathop{\mathbb{A}}_{e=1}^{n_{el}} (\underline{\underline{k}}^e) \quad \text{and} \quad \underline{F} = \mathop{\mathbb{A}}_{e=1}^{n_{el}} (\underline{f}^e)$$

To be truly efficient, a computer implementation should also use a sparse

matrix representation to store the stiffness matrix such as a compressed sparse row (CSR) or compressed sparse column (CSC) representation. This is especially important in the three-dimensional setting where the number of elements may be staggering. Note that for some applications, a so-called matrix-free representation is preferred. This is a topic for a more advanced finite element class.

Now that we know how to build the global stiffness matrix and force vector from element stiffness matrices and force vectors, all that remains is to determine how to compute the entries of the element stiffness matrices and force vectors. This is the focus of the next lecture.