Boundary Value Problems: Assembling the Matrix System

It is important to note that $\underline{\underline{K}}$ is a sparse matrix as each NURBS basis function is highly localized. Thus, for many combinations of i and j, $K_{ij} = 0$. We can take advantage of this fact in order to reduce the amount of work in building and solving the matrix system.

Perhaps the easiest and most efficient means of building the matrix system is by a process called element assembly, or simply just assembly. Assembly relies on the fact that we can write integrals over the entire domain as integrals over only the elements. For example:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} K \vec{\nabla} w \cdot \vec{\nabla} u \, d\Omega^e = \int_{\Omega} K \vec{\nabla} w \cdot \vec{\nabla} u \, d\Omega$$

As opposed to computing integral quantities over the entire domain Ω, we only compute integral quantities over the elements $\{\Omega^e\}_{e=1}^{n_{el}}$. For each element, there are only $n_{loc} = (p_1+1)(p_2+1)$ nonzero basis functions, and as discussed previously, we may localize them as $\{N_a^e\}_{a=1}^{n_{loc}}$. Consequently, for each element, we may build a small matrix $\underline{\underline{k}}^e$, referred to as the element or local stiffness matrix, associated with these functions. For the heat equation:

$$[\underline{\underline{k}}^e]_{ab} := \int_{\Omega^e} K \vec{\nabla} N_a^e \cdot \vec{\nabla} N_b^e \, d\Omega^e + \int_{\Gamma_R \cap \partial\Omega^e} \beta N_a^e N_b^e \, d\Gamma^e$$

We may also build a small vector $\underline{f}^e$, referred to as the element or local forcing vector:

$$[f^e]_a := \int_{\Omega^e} N_a^e \mathcal{F} \, d\Omega^e + \int_{\Gamma_N \cap \partial\Omega^e} N_a^e h \, d\Gamma^e + \int_{\Gamma_R \cap \partial\Omega^e} \beta N_a^e u_R \, d\Gamma^e$$

Once we compute these element entries, we can update the appropriate entries in the global stiffness and forcing vectors using the IEN array:

$$\left.\begin{aligned} K_{ij} &\leftarrow K_{ij} + k^e_{ab} \\ F_i &\leftarrow F_i + f^e_a \end{aligned}\right\} \quad a,b = 1,\dots,n_{loc}, \quad i = IEN(a,e),\ j = IEN(b,e)$$

where the arrow ($\leftarrow$) is read "is replaced by". The one exception to the above update is when either i or j corresponds to a basis function N_i or N_j which is nonzero over the Dirichlet boundary. To handle this exception, we employ the BC array:

$$BC(i) = \begin{cases} 0 & \text{if } N_i|_{\Gamma_D} = 0 \\ 1 & \text{otherwise} \end{cases}$$

If $BC(i) = 1$, we set:

$$K_{ii} = 1 \qquad K_{ij} = 0 \quad i \neq j$$

$$F_i = g_i$$

which corresponds to the Dirichlet condition $d_i = g_i$.

If during the assembly process we come across i, j such that $BC(i) = 0$ and $BC(j) = 1$, then we do not update K_{ij} with k^e_{ab} but rather F_i to account for the Dirichlet boundary condition:

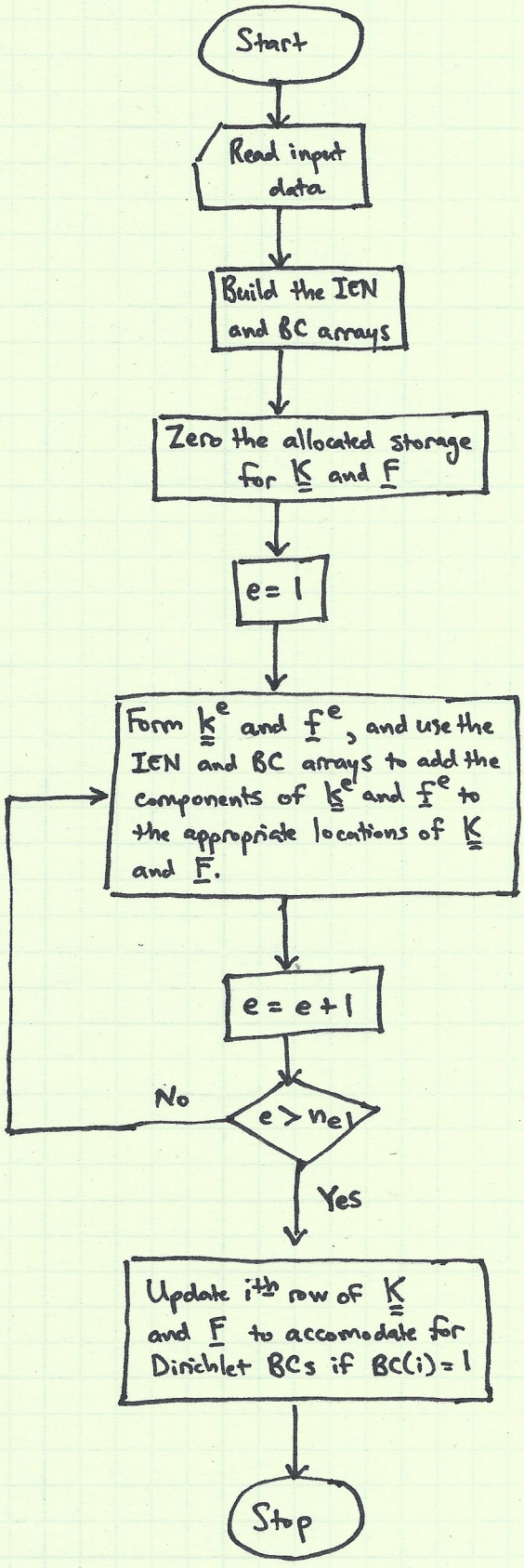
$$F_i \leftarrow F_i - k^e_{ab} g_j \quad \text{if } BC(i) = 0 \text{ \& } BC(j) = 1$$

Algorithmically, the assembly process for element e looks as follows:

Assembly for Element e:

```
for a = 1,...,n_loc
    Set i = IEN(a,e)
    if BC(i) = 0
        for b = 1,...,n_loc
            Set j = IEN(b,e)
            if BC(j) = 0
                K_ij ← K_ij + k^e_ab
            else
                F_i ← F_i - k^e_ab g_j
            endif
        endloop
        F_i ← F_i + f^e_a
    endif
endloop
```

An overall algorithm for the ~~construction~~ assembly of the global stiffness matrix $\underline{\underline{K}}$ and forcing vector $\underline{F}$ then looks as follows:



The action of the assembly algorithm is denoted throughout as $\mathbb{A}$, the assembly operator, and we write:

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