`FEPack`

implementation notes

1 Essential conditions

In order to handle essential conditions, I used the approach proposed by `XLiFE++`. Let $\vec{U}$ denote the vector of components of a function u in a vector space $\mathcal{V} = \mathrm{Span}\{w_1, w_2, \ldots, w_n\}$. Then, any essential condition can be expressed under the generic form

$$\mathbb{E}\,\vec{U} = \vec{\varphi}, \tag{1.1}$$

where $\mathbb{E}$ is a $m \times n$ matrix, and $\vec{\varphi}$ a m–vector. One could impose similar constraints on the associated test function v

$$\mathbb{F}\,\vec{V} = 0, \tag{1.2}$$

where $\mathbb{F}$ is a $m \times n$ matrix that needs not to be equal to $\mathbb{E}$, so that the discrete problem we are interested in solving is

$$\text{Find } u \in \mathcal{V},\ \ \mathbb{E}\,\vec{U} = \vec{\varphi}, \text{ such that} \qquad a(u,v) = \ell(v), \qquad \forall\, v \in \mathcal{V},\ \ \mathbb{E}\,\vec{V} = 0. \tag{1.3}$$

The goal is to *compute the projection matrices* corresponding to the constrained spaces $\{u \in \mathcal{V},\ \mathbb{E}\,\vec{U} = 0\}$ and $\{v \in \mathcal{V},\ \mathbb{F}\,\vec{V} = 0\}$, and to rewrite the system (1.3).

1.1 Reducing the constraints

Under the general form (1.1), the essential conditions might admit redundant or contradictory constraints. Therefore, they need to be reduced to a minimal system. To do so, we use a QR decomposition with permutation. In `Matlab`, given the $m \times n$ matrix $\mathbb{E}$, the command

```
[Q, R, P] = qr(E)
```

returns an $m \times m$ unitary matrix $\mathbb{Q}$, an $m \times n$ upper triangular matrix $\mathbb{R}$ as well as an $m \times n$ permutation matrix $\mathbb{P}$ such that $\mathbb{E}\,\mathbb{P} = \mathbb{Q}\,\mathbb{R}$. Additionaly, $\mathbb{E}$ is chosen so that the components of $\mathbb{R}$ are in decreasing order.