CS 294-73 (CCN 27241)
Software Engineering for Scientific Computing

http://www.cs.berkeley.edu/~colella/CS294

Supplementary Information on Homework 2

Homework 2

We expand on three topics in the implementation of homework #2:

- Implementation of the `FEPoissonOperator` class.
- Point Jacobi
- Reinsertion

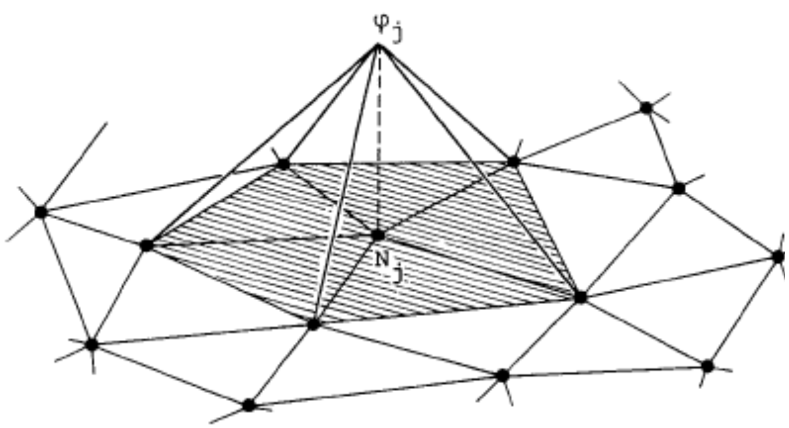

Fig 1.9 The basis function φ_j.

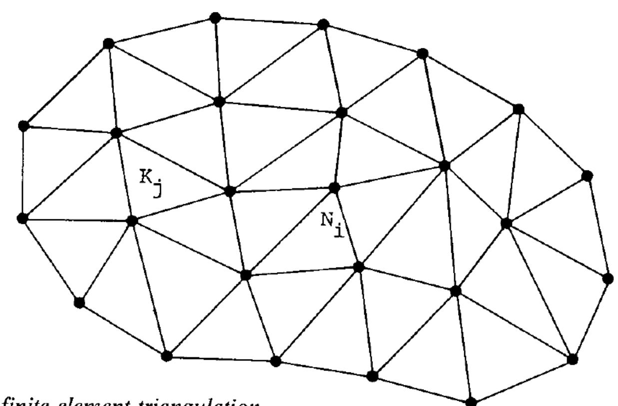

Fig 1.8 A finite element triangulation

FEPoissonOperator

`m_FEPoissonOperator(const FEGrid& a_FEGrid)` copies the input argument onto `m_grid`, and computes the sparse matrix (denoted here by *L*, represented in `FEGrid` as `m_matrix`).

`m_matrix = SparseMatrix(Ni,Ni), Ni= FEGrid.getNumInteriorNodes().`

Initialize $L = 0$

`E = m_grid.getNumElts();`

for $e = 0 \ldots E-1$

`const Element& K = m_grid.Element();`
Then access the nodes in K using `Element::vertices` and `FEGrid::getNode` .

for $(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \in K_e : (n, n') \in \mathcal{N}_I$

Determine whether both nodes are interior (`Node::isInterior`). If so, get the gradients using `FEGrid::gradient` , compute their dot product and multiply by the element area (`FEGrid::elementArea`). Increment the corresponding entry of the matrix `m_matrix[...] += ...`

$$L_{n,n'} += \int_{K_e} \nabla \Psi^h_{n'} \cdot \nabla \Psi^h_n d\boldsymbol{x}$$

endfor

endfor

Important: there are three indexing schemes for the nodes: one that is local to the element, one that indexes over the interior elements, and one that indexes over all the elements. The one that is appropriate to use in any of the `FEGrid, Node, or Element` member functions is documented by the function or argument name.

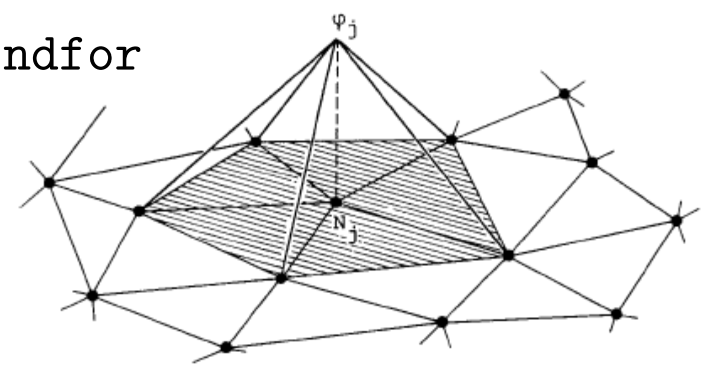

Fig 1.9 The basis function φ_j.

FEPoissonOperator::makeRHS(...);

$$\int_{\Omega} \Psi_n^h f d\boldsymbol{x} = \sum_{K_e} \int_{K_e} \Psi_n^h f d\boldsymbol{x}$$

$$\int_{K_e} \Psi_n^h f d\boldsymbol{x} \approx Area(K_e) f(\boldsymbol{x}_e^{centroid}) \Psi_n^h(\boldsymbol{x}_e^{centroid})$$

`b=a_rhsAtNodes` is a `vector<float>` of length equal to `m_grid.getNumInteriorNodes()`, while the function values at centroids are held in a `vector<float>` of length `m_grid.numElts()`.

Initialize $b = 0$

for $e = 0 \ldots E-1$

for $\boldsymbol{x}_n \in K_e : n \in \mathcal{N}_I$

$b_n += Area(K_e) f(\boldsymbol{x}_e^{centroid}) \Psi_n^h(\boldsymbol{x}_e^{centroid})$

endfor

endfor

The process for computing b is very similar to the process for computing L. The relevant member functions of `FEGrid` are `Element()`, `Node()`, `getNode(…)`, `centroid(…)`, `elementArea(…)`.

Reinsertion

The unknowns in the linear solve *La* = *b* are computed on interior nodes. However, for the purpose of plotting, we want to represent the solution on all of the nodes, including the boundary nodes, which are set to zero. The pseudocode for this is given as follows.

The important thing to remember is that a^{tot} and a are indexed by different numbering systems. `FEGrid::Node(int)` accesses all of the nodes (interior and boundary). From the Node, you can get the interior Node number (`Node::getInteriorNodeID`)

$a^{tot} : \mathcal{N} \rightarrow \mathbb{R}$

$\texttt{for}\ n \in \mathcal{N}\ \texttt{do}$
$\quad\texttt{if}\ \ n \in \mathcal{N}_I$
$$a_n^{tot} = a_n$$
$\quad\texttt{else}$
$$a_n^{tot} = 0$$
$\quad\texttt{endif}$
$\texttt{endfor}$