
Programming in C^{++}:
Use the following points and weights for the quadrature rule on the reference tetraeder:

```
xi = [0.127647, 0.293999, 0.544152], weight = 0.00916943
xi = [0.245713, 0.565933, 0.122515], weight = 0.021157
xi = [0.303773, 0.0706797, 0.544152], weight = 0.016027
xi = [0.584748, 0.136055, 0.122515], weight = 0.0369799
xi = [0.0342028, 0.293999, 0.544152], weight = 0.00916943
xi = [0.0658387, 0.565933, 0.122515], weight = 0.021157
xi = [0.0813957, 0.0706797, 0.544152], weight = 0.016027
xi = [0.156683, 0.136055, 0.122515], weight = 0.0369799
```

You can use `sparsematrix.hh`, `sparsematrix.cc` (a sparse matrix class) and `vector.hh` (a vector class for vectors of dynamic length) downloadable from the website.

28 Write a function

```
void calcSourceElVec (const ElTrans& elTrans, const NedelecTet& fe,
                      VectorField f, Vec<6>& elVec);
```

that approximates the element load vector `elVec` associated to a tet T given by

$$\int_T \boldsymbol{f}(x) \cdot \boldsymbol{\varphi}_k(x)\, dx = \int_{\hat{T}} \boldsymbol{f}(x(\xi)) \cdot (F_T^{-T} \hat{\boldsymbol{\varphi}}_k(\xi))\, J_T\, d\xi,$$

where $x(\xi) = x_0 + F_T\, \xi$, using quadrature rule.

29 30 Complete and implement the following class modelling the mesh:

```
class Mesh{
public:
   void read(...);
   int numVertices();
   int numEdges();
   int numTets();
   Vec<3> vertexCoord (int i);
   void getTetVertexIndices(int t, Vec<4, int>& tetVertexInd);
   void getTetEdgeIndices(int t, Vec<6, int>& tetEdgeInd);
   ...
private:
   Vector vertexCoord_;
   Vector tetVertexInd_;
   ...
};
```

Hint: Choose a minimal format for reading the mesh. Discuss in preparation of programming the need of additional functions or private variables.

31 Write a function

```
void getFE (const Mesh& mesh, const int& t, NedelecTet& fe,
            ElTrans& elTrans, Vec<6, int>& dnums);
```

that returns the finite element `fe` together with `elTrans` and the global edge numbers of the local degrees of freedom.

Hint: Take care of the order of the edge numbers in `elTrans`.

32 Write a function

```
void assembleStiffnessMatrix(const Mesh& mesh, SparseMatrix& K);
```

that assembles the stiffness matrix `K` corresponding to the bilinear form

$$a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \mathbf{curl}\, \boldsymbol{u} \cdot \mathbf{curl}\, \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{v}\, dx$$

for `mesh` being the triangulation of Ω.

33 Write a function

```
void assembleLoadVector(const Mesh& mesh, VectorField f, Vector& b);
```

that assembles the load vector `b` corresponding to the functional

$$\langle F, \boldsymbol{v} \rangle = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}\, dx$$

for `mesh` being the triangulation of Ω.