
Programming.
In this tutorial we will calculate the stiffness matrix and load vector corresponding to the pure Neumann problem with homogeneous boundary conditions (see Exercise 16) and an arbitrary mesh of (0, 1):

$$\mathcal{T}_h = \{T_1, \ldots, T_{n_h}\} \qquad \text{where} \quad T_k = (x_{k-1}, x_k),$$
$$0 = x_0 < x_1 < \cdots < x_{n_h-1} < x_h = 1.$$

19 Write a function

```
void ElementStiffnessMatrix (double xa, double xb, Mat22& elMat);
```

which for given nodes xa=x_{k-1} and xb=x_k returns the element stiffness matrix elMat=$K_h^{(k)}$ of the element T_k.

20 Write a function

```
void ElementLoadVector (RealFunction f, double xa, double xb,
                        Vec2& elVec);
```

which for a given function f=$f \in C[0, 1]$ and the nodes xa=x_{k-1} and xb=x_k returns the approximated 2-dimensional element load vector elVec$\approx f_h^{(k)}$ on the element T_k. Use the trapezoidal rule to approximate the involved integrals.

21 Design a data type `Mesh` to store the mesh information that you need later on to assemble of the stiffness matrix. Make sure that your data type allows

- initializing (e.g. with an equidistant mesh with a certain number of nodes)
- asking for the number of nodes
- asking for the "coordinate" of an arbitrary node

22 Design an *efficient* data type `SMatrix` to store the stiffness matrix later on. Make sure that your data type allows

- initializing (with a certain number of rows=columns and zero entries)
- asking for any entry in the diagonal and the two off-diagonals
- adding a value to a certain entry

23 Write a function

```
void AssembleStiffnessMatrix (const Mesh& mesh, SMatrix& mat);
```

that assembles the $(n_h + 1) \times (n_h + 1)$ stiffness matrix mat=K_h (see Exercise 16) for a given mesh mesh=$\mathcal{T}_h$ of (0, 1).

Hint: Set $K_h = 0$, then loop over all elements. For each element, call `ElementStiffnessMatrix` and add the entries of $K_h^{(k)}$ at the correct positions of K_h.

24 Write a function

```
void AssembleLoadVector (RealFunction f, const Mesh& mesh,
                         Vector& vec);
```

that assembles the load vector `vec`=$\underline{f}_h$ for the given function `f`=$f \in C[0,\, 1]$ and the given mesh.

Here `Vector` is your favourite vector type (you can for instance use that from `vectors.cc` on the tutorial website).

Hint: Set $\underline{f}_h = 0$, then loop over all elements. For each element, call `ElementLoadVector` and add the entries of the element load vector to the right places.

Test all your functions for

$$f(x) = 2\,x + 1$$

and at least an equidistant mesh with 20 elements. To see if your functions are correct it might be good to add a *print* function for `SMatrix` that prints the entries to the screen.