A Python class for B-spline finite elements

Python classes provide all the standard features of Object Oriented Programming. Objects can contain arbitrary amounts and kinds of data ("attributes") as well as procedures ("methods"). The methods of an object can access and modify the data associated to it via the "self"-identifier.

Python objects are instances of classes:

```python
obj = MyClass()
```

generates the object `obj` from the class `MyClass`, whose defintion starts with:

```python
class MyClass:
    """A simple example class""
    i = 12345
    def f(self):
        return 'hello world'
```

An object’s attributes and methods are accessed with the dot; here, `a = obj.i` is an attribute and `m = obj.f` is a method, which can later be called via `m()`. See [https://docs.python.org/3.4/tutorial/classes.html](https://docs.python.org/3.4/tutorial/classes.html) for details on Python classes.

Our aim is to write a class `BsplFEM` that can generate the tools for a finite element method (FEM) based on B-splines. B-splines (basis splines) are piece-wise polynomials of degree $k$ that can serve as basis functions with local support in FEM (see the lecture notes Computational Plasma Physics from SS2019). The family of $n$ B-splines of degree $k$ is defined by a non-decreasing sequence of knots $X = \{x_i\}_{0 \leq i \leq n+k}$ on the real line. There can be several knots at the same position. In the case when there are $m$ knots at the same point, we say that the knot has multiplicity $m$. The $j$-th B-Spline ($0 \leq j \leq n-1$) denoted by $N_j^k$ of degree $k$ is defined by the recurrence relation:

$$
N_0^k(x) = \chi_{[x_j,x_{j+1})}(x), \quad N_j^k(x) = w_j^k(x)N_j^{k-1}(x) + (1 - w_{j+1}^k(x))N_{j+1}^{k-1}(x),
$$

$$
w_j^k(x) = \frac{x-x_j}{x_{j+k} - x_j},
$$

where $\chi$ denotes the indicator function. B-splines are non negative; their support is $[x_j,\ldots,x_{j+k+1}]$ and they are a partition of unity: $\sum_{i=0}^{n-1} N_i^k(x) = 1, \forall x \in \mathbb{R}$. Moreover, if a knot $x_i$ has a multiplicity $m$ then the B-spline is $C^{k-m}$ at $x_i$. In the case where all knots, except the boundary knots are of multiplicity 1, the set $(N_i^k)_{0 \leq i \leq n-1}$ of B-splines of degree $k$ forms a basis of the spline space

$$
S^k = \{ v \in C^{k-1}([x_0,x_n]) \mid v|_{[x_i,x_{i+1}]} \in P_k([x_i,x_{i+1}]) \},
$$

where $P_k$ denotes the space of polynomials of degree $k$. 1
We want to accomplish the following tasks:

1. Use `scipy.interpolate.BSpline` to create and manipulate B-splines in Python.

2. Plot single B-splines of various degrees and print the coefficients of the polynomials in each interval.

3. Take a look at the notebook `Basics-of-Lagrange-FEM.ipynb` from the git repository to see how the class `LagrangeShape` is used there.

4. Write a Python class `BsplFEM` that takes the degree $k \in \mathbb{N}$ and an arbitrary knot sequence $X = \{(x_i)_{0 \leq i \leq n+k}\}$ as input and returns:
   
   (a) B-spline basis functions of degree $k$ for periodic boundary conditions,
   
   (b) the mass matrix $M$, transport matrix $T$ and stiffness matrices $S$ defined by

   \[
   M_{ij} = \int_{x_0}^{x_n} N_i^k N_j^k \, dx, \quad T_{ij} = \int_{x_0}^{x_n} N_i^k (N_j^k)' \, dx, \quad S_{ij} = \int_{x_0}^{x_n} (N_i^k)'(N_j^k)' \, dx, \]

   (c) the collocation matrix $C$ and the histopolation matrix $H$ defined by

   \[
   C_{ij} = N_i^k(x_j), \quad H_{ij} = \int_{x_j}^{x_{j+1}} N_i^k \, dx. \]