To familarize ourselves with the Discontinuous Galerkin Method, we consider the prototype model for hyperbolic PDEs: the linear advection equation. We want to find $u(x,t)$, $x \in [0,L]$, $t \geq 0$, solution of

$$\partial_t u + c\,\partial_x u = 0 \tag{1}$$

with initial conditions

$$u(x,0) = f(x) \tag{2}$$

and periodic boundary conditions

$$u(0,t) = u(L,t). \tag{3}$$

1. Find the analytical solution of equation (1) with conditions (2) and (3) using, e.g., the method of characteristics.

2. Write a `python` code that solves the problem using the Discontinuous Galerkin Method. At the end, you must end up with a function

```
def advection1d(L, n, dt, m, p, c, f, a, rktype, anim):
```

   Input parameters of `advection1d` are described below:

   - We assume that the mesh is uniform both in space and time. Thus, the 1D domain $[0,L]$ is divided into `n` elements of size L/n (we thus have $n+1$ nodes) and we assume a constant timestep `dt`. There are m time steps in the simulation so the final time is $m \times dt$.
   - On each element $D^k := \left[\frac{k-1}{L}, \frac{k}{L}\right]$, we choose to represent the solution locally as a polynomial of arbitrary order p as

   $$u_h^k(x^k(r),t) = \sum_{i=0}^{p} \hat{u}_i^k(t)\mathcal{P}_i(r)$$

     where $\hat{u}_i^k$ are the unknown coefficients, ψ_i are the normalized Legendre polynomials of order i and $x^k(r)$ is an affine mapping from the standard element $r \in [-1,1]$ to element D^k.
   - Parameter c is the advection speed in equation (1)
   - Parameter f is a python function that describes initial conditions. It must be of the form:

```
def f(x, L):
```

   - Parameter a controls the upwindness of the scheme. Choosing $a = 1$ leads to a full upwind scheme, $a = 0$ leads to a centered scheme and $a = -1$ leads to a downwind (unstable) scheme.
   - Parameter `rktype` is a string that defines the time stepping scheme that is used during the simulation. You must implement the following schemes: `rktype = ‘‘ForwardEuler’’`, `rktype = ‘‘RK22’’` and `rktype = ‘‘RK44’’`.

- When parameter `anim` is set to `True`, the solution should be plotted as an animation.

The output parameter of `sol` is a `numpy` array of size $(p+1) \times n \times m$:

$$\mathrm{sol}(i, j, k) = \hat{u}_i^j(k \times dt).$$

3. With `rktype = ''RK44''`, change the time step manually and determine how the stable time step size scale with the element size $h = L/n$ and the polynomial order p.
4. Using a smooth initial function f, e.g., $f(x) = \sin(2\pi x/L)$, describe how the numerical solution behaves compared to the exact solution in time? For example, do you observe losses in amplitude? Changes of initial solution profile? What is the numerical advection speed? Change the numerical flux type by changing a.
5. Do the same with a non-smooth initial function, e.g., the heaviside step function.
6. Solutions to the linear advection equation conserves energy if u is assumed periodic:

$$\frac{d}{dt}\int_0^L u^2(x,t)dx = 0.$$

Is the energy numerically conserved? The answer to this question essentially depends on parameter a. It is possible to answer this question analytically but here, we ask you to do numerical observations only!

Hints:

- Legendre polynomials are provided by the SciPy library.
- The mass and stiffness matrices can be found analytically; your code should not compute them, no need to use numerical integration for these terms.
- The coefficients $\hat{u}_i^k(0)$ of the initial solution f can be found by performing an L^2 projection of f onto the basis polynomials ψ_i, i.e., solving

$$\sum_j M_{ij}\hat{u}_j = (f, \psi_i)_{D^k},$$

where M_{ij} is the element mass matrix. Use a 5-point Gaussian quadrature rule to compute the integral (quadrature points and weights can be hard-coded).

- Note that we are using a *modal* expansion to approximate the solution (i.e., using the Legendre polynomials). In particular, think about how the numerical fluxes need to be computed.

- If you are stuck, the following reference might help:
  Emilie Marchandise, Nicolas Chevaugeon, and Jean-François Remacle. *Spatial and Spectral Superconvergence of Discontinuous Galerkin Method for Hyperbolic Problems.* Journal of Computational and Applied Mathematics 215, no. 2 (June 2008): 484–94.

Practical information:

Groups: *You are asked to form groups of two or three under the constraint that each group should contain at least one TFMASA student and one EPL student (the list of TFMASA students is published on Moodle). The groups should be formed on Moodle before Monday 7/2 at 18:15.*

Collaboration: *You are allowed, and even encouraged, to exchange ideas on how to address this assignment with students from other groups. However, you must do all the writing (report and codes) only with your own group; it is strictly forbidden to share the production of your group. Plagiarism will be checked.*

Writing: *The report should contain your answers to the questions as well as your figures. The report should have maximum 4 pages (including figures). Please, do not include your code in the report. Take care of the writing of your report and the structure of your codes. Concision and quality of the writing, clarity of explanations and rigor will be taken into account.*

Language: *All reports and communications are equally accepted in French and English.*

Deliverables: *Each group is asked to submit on Moodle*

- *A report named* `report.pdf`*,*
- *A Python module named* `advection1d.py` *containing the function* `advection1d` *as specified above; this module should not execute anything so that it can be tested.*
- *A Python script named* `run.py` *that reproduces all the figures in your report.*

Deadline: *The homework is due Friday 18/2 at 12:45.*

Questions: *You can address questions either during the weekly office hours (see time and location on Moodle), or by sending an email to the teaching assistants. Direct messages on Teams will not be considered.*