Homework 5

Finite-volume method for unstructured grids

due February 22, 23:59

Guidelines:

- Use only a **single pdf**. Preferably scan your homework, if you take a photo, make sure it is sharp and bright;
- **Include all plots in the pdf** in the right place;
- Do not present plots without commenting them: always write a (short) description of what the plot tells you and what you can conclude from it;
- You can work in groups up to 3;
- If you work in groups:
  - write the names of all group members at the beginning of the report;
  - you can work and discuss together but each group member is required to submit an individual report written with his/her own words; **copy-paste reports will not be accepted**
- Please use the following naming convention: "**surname_hwX.pdf**" and include all Matlab or Python files as a single **separate** archive: "**surname_hwX.zip**". X is the number of the homework.

In order to solve the flow equations in complex geometries, most of the time unstructured grids are unavoidable. However, the accuracy of the finite-volume method is strongly dependent on the quality of the grid. The aim of this assignment is to provide an understanding of the finite-volume discretisation and its accuracy on unstructured grids. Using the Green's theorem, we can write

$$(\nabla f)_c \approx \frac{1}{V_c}\int_V \nabla f \,\mathrm{d}V = \frac{1}{V_c}\oint_S f\bar{n}\,\mathrm{d}S \ , \tag{1}$$

and

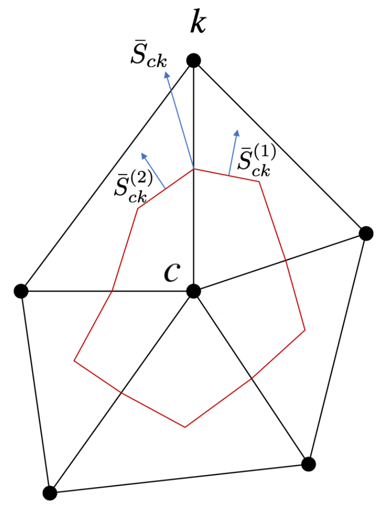
$$(\nabla^2 f)_c \approx \frac{1}{V_c}\int_V \nabla^2 f \,\mathrm{d}V = \frac{1}{V_c}\oint_S \nabla f \cdot \bar{n}\,\mathrm{d}S \ . \tag{2}$$

As we mentioned during the lectures, the straight-forward approximations of equations (1) and (2) are, respectively,

$$(\nabla f)_c \approx \frac{1}{V_c}\sum_{k=1}^{m_c} \frac{f_k + f_c}{2}\bar{S}_{ck} \tag{3}$$

and

$$(\nabla^2 f)_c \approx \frac{1}{V_c}\sum_{k=1}^{m_c} \frac{f_k - f_c}{r_{ck}} S_{ck} \ , \tag{4}$$



where m_c is the number of neighbouring nodes, $\bar{S}_{ck}$ is the sum of vectors $\bar{S}_{ck}^{(1)}$ and $\bar{S}_{ck}^{(2)}$ (vectors normal to the edges of the dual grid with a length corresponding to the length of the edges) and r_{ck} the distance between nodes c and k.

Your task is to run a Matlab (Python) code for different shapes of the domain with different types of elements, and compute the first and second-order derivatives of a given function in order to investigate the order of accuracy of the discretisation scheme.

Four different choices for geometry and mesh are implemented in the code:

- Square geometry with quadrilateral elements: geocase='squre_quad';
- Square geometry with uniform triangular elements: geocase='squre_tri';
- Triangular geometry with uniform triangular elements: geocase='triangle';
- Triangular geometry with non-uniform triangular elements: geocase='triangle_random'.

Task 1:

Analyze the structure of the code (unstructured_FV.m) to familiarise yourself with the different steps of the implemented algorithm. Set $N = 20$: the total number of points NP is proportional to N^2. For **each choice of geometry** perform the following analysis:

a) First we need to define the function f whose derivatives will be approximated. This is done in the function setf.m, the variables dfxa, dfya, ddfa represent the analytical derivatives in x, y and the Laplacian, respectively. **Note that the function, as well as the derivatives, are defined in every point of the domain, hence they must be arrays of length NP.** Choose the function f to be a constant and run the code. Observe the computed values of $\partial f/\partial x$ and $\partial f/\partial y$. How do they compare with the analytical values? Report your choice of function and plot both the computed values and the analytical solutions, using trisurf function.

b) Repeat the task for a linear function, *i.e.* $f = ax + by$. How do the computed first derivatives compare with the analytical values in this case? Report your choice of function and the computed values.

What can be concluded regarding the order of accuracy of the discretisation scheme?

Note: The code computes derivatives only for the inner nodes. The boundary nodes require a special treatment which is not implemented in this version. When comparing results and computing the errors use only inner nodes. This is easily done as the indices of the inner nodes are stored in the array nodes_in. For example f(nodes_in) will extract the function value at all inner nodes.

Task 2

In this task we want to analyse the effect of the discretisation on the accuracy of the computed derivatives. This is done by changing the number of points NP (*i.e.* the parameter value N, using several values between 10 and 100). The error ϵ is computed using the L_2 norm

$$\epsilon = \frac{\sqrt{\sum_{i=1}^{NP} |f_i - \hat{f}_i|^2}}{\sqrt{\sum_{i=1}^{NP} |f_i|^2}} \,, \tag{5}$$

where NP is the total number of points, f the exact and $\hat{f}$ the computed value of either first or second derivatives (Laplace equation). **Hint: The L_2 norm can easily be computed using the Matlab function norm**, or the numpy linalg.norm function in Python.

a) For this task we will consider a slightly more complicated function

$$f(x, y) = \sin \pi x \cos \pi y. \tag{6}$$

Compute the derivatives and the Laplacian, then implement them in using the setf.m function.

b) An alternative way to compute the second derivative of f is to apply the first-derivative operator two times consecutively. Implement this approximation using the function grad.m. Using equation 6, compute the errors:

- errx, on the first derivative in x
- erry, on the first derivative in y
- errl, on the Laplacian with the function laplace
- errln, on the Laplacian with the alternative method

Plot the errors as a function of N (or $\sqrt{NP}$) in a $\log - \log$ plot. Determine the accuracy of the scheme. Note that the grid size is inversely proportional to N ($h \sim N^{-1}$). For reference plot also the lines corresponding to theoretical first- and second-order accuracy in the same figure.

Perform this computation and compare the results for different meshes. What is the accuracy of the Laplace operator computed in these two ways? In case of quadrilateral element, which one has lower error?

c) What can be concluded from your observations? How do these observations confirm what was discussed in the lecture?