Programming Skills

Meshes, Graphs, Data structures

We would like to test your programming skills with two tasks representative for some daily-life projects. Although the real tasks are already solved (and in 3D) you are going to work with similar data, data structures when doing scientific computing.

- We ask you to solve the tasks in Python and/or C++ (please avoid using pure C).
- Write your own code. We would like to see your skills and your programming style. Take into account that you will work in a team on larger projects. Other people should have a chance to understand and to use your code.
- Your code has to be portable, we will compile and run it on Ubuntu Linux.
  - in case of Python: Use version 3.x.
  - in case of C++: Use c++17 (c++20 is not fully implemented on all compilers). We will check your code with different compilers:
    * `g++ -O3 -std=c++17 -Wall -pedantic -Wextra -Weffc++ -Woverloaded-virtual -Wfloat-equal -Wshadow -Wredundant-decls -fmax-errors=1`
    * `clang++ -O3 -std=c++17 -Weverything -Wno-sign-conversion -Wno-c++98-compat -Wno-c++98-compat-pedantic -Wno-shorten-64-to-32 -Wno-padded -ferror-limit=1`
- Document and publish your code for further use.

Don't hesitate to contact Gundolf Haase (*gundolf.haase@uni-graz.at*) regarding further questions on details of the task.

1. A bounded domain in 2D is represented by a discretization consisting of triangular **elements**. Assume a general polygonal domain in 2D although we use a square[1] for simplicity. Each element is uniquely defined via the global index of its vertices. Vertices as well as elements are consecutively numbered starting with 0.
   The following data files are provided with a tiny and a larger example:

   - Element file *3x3_elements_2d.txt*[2] resp. *117x34_elements_2d.txt*[3] includes:

     ```
     #number of Elements
     vertex_1    vertex_2    vertex_3     // element 0
     vertex_1    vertex_2    vertex_3     // element 1
                    ...
     ```

   - Coordinate file *3x3_coords_2d.txt*[4] resp. *117x34_coords_2d.txt*[5] includes:

     ```
     #number of Coordinates
     x    y     // node 0
     x    y     // node 1
           ...
     ```

   (a) Read the elements from file and generate the following mappings:
   Nodes $\longrightarrow$ Elements containing this node as vertex;
   Elements $\longrightarrow$ (neighboring) Elements;
   Nodes $\longrightarrow$ Nodes (node-to-node relation), i.e. find the next neighbors of a node including itself.

   (b) Find, with the nodal coordinates, the element with the shortest distance to a given coordinate in 2D. Test for coordinates inside/ouside the given domain.

   (c) How do you have to change your data structures for storing a mixed discretization consisting of triangles and quadriliterals?

2. Write a data structure/class `Graph` for a directed graph that will be initialized by the node-to-node relation from task 1.

   (a) Derive from `Graph` a weighted graph data structure/class, i.e. by taking into account the inverse of the squared node distance from the given discretization.
   The weight of reflexive edges (v, v) should be the negative sum of the weights from all non-reflexive edges (v, w) of node v.

   (b) Derive further a data structure/class for sparse matrices stored in CRS format (compressed row storage).

   (c) Realize the following methods/functions:

   - Matrix-Vector product and provide an example for correctness of the result. You don't have to implement a vector class, use standard vectors.
   - Read/write matrix to an ASCII-file / binary file.

   (d) Visualize the graph/matrix pattern.

[1] `http://imsc.uni-graz.at/haasegu/Download/square_3x3.pdf`
[2] `http://imsc.uni-graz.at/haasegu/Download/3x3_elements_2d.txt`
[3] `http://imsc.uni-graz.at/haasegu/Download/117x34_elements_2d.txt`
[4] `http://imsc.uni-graz.at/haasegu/Download/3x3_coords_2d.txt`
[5] `http://imsc.uni-graz.at/haasegu/Download/117x34_coords_2d.txt`