# Numerical Project

Cyril Tsilefski

October 29, 2020

## Contents



# 1 Functional requirement of the program

## 1.1 The project

The goal of this project is to simulate the movement of a fluid through different geometries. The program creates a box of a choosen size, builds a geometry inside it and simulates the movement of a given fluid.

## 1.2 Files

In order to increase readability, the project is made of several files. I made the choice to work with Object Oriented Programming.

- main.py: This file calls for the needed functions/class
- matrices.py: This file contains the class "Matrices", it builds the geometry, the different matrices to plot and stores them
- plot.py: This file plots the matrices built in "matrices.py"
- parameters.py: This file contains all the variables that can be changed by the user
- data_check.py: This file checks the variables and makes sure that the program will run

## 1.3 Data

This project uses several piece of data set by the user to work.

- $N_x$ and $N_y$ are the size of the domain
- $h$ represents the size of a cell
- $geometry$ corresponds to the choosen geometry
- $angle$ corresponds to the angle of the widening/shrinkage geometry
- $v_x$ is the Neuman condition
- $\phi_{ref}$ is the Dirichlet condition

Be careful in the case of a widening/shrinkage geometry ! In order for the program to generate a domain from one end to another, there is a restriction on the angle, if the restriction is not met, the program will output a ValueError. The restriction is as follows:

$$|angle| < \arctan\left(\frac{0.5 \times N_y - 1}{N_x}\right)$$

The angle parameter should be set in degree, the program will convert it to radians for the computation.

### 1.4 Outputs

As of the alpha version, the program outputs 4 pdf files, one for each plot. The files are saved in a subfolder named "figures" and the filenames are set with the following rule:

`<data>_<geometry>_Nx=<Nx>_Ny=<Ny>.pdf`

`data` stands for the plotted data (potential, velocity, streamlines, pressure).

### 1.5 Concerning the running time

Due to the function `numpy.linalg.solve()` being slow for big matrices, the bigger the size of the domain, the higher the running time.
For a domain size of 3600 cells ($60 \times 60$), it takes around 20 seconds to run, for a domain size of 14 400 cells ($120 \times 120$), it increases to 23 minuts.

I searched for a faster method to solve the linear system in vain, thus I recommend to stay on relatively low values for $N_x$ and $N_y$, the graphs are easily readable for a value of 60 each.

## 2 Internal structure of the program

### 2.1 Description of the physical model

In order to build the model, the program uses a squared structured lattice model (matrix). The values are computed at each point of the matrix.