Dessiner avec Python à partir d'un script C++

Le C++ n'est que peu adapté à l'affichage de sorties graphiques. Il est souvent plus simple de faire appel à une librairie ou un logiciel externe (Python, Paraview, Gnuplot...). Le mode opératoire est toujours le même: les données à afficher sont écrites dans un fichier, puis le programme tiers est exécuté: il lit le fichier et génère l'affichage désiré.

Pour ce projet, nous vous proposons d'utiliser un script Python pour afficher des nuages de points 2D et colorier les clusters que votre programme va définir.

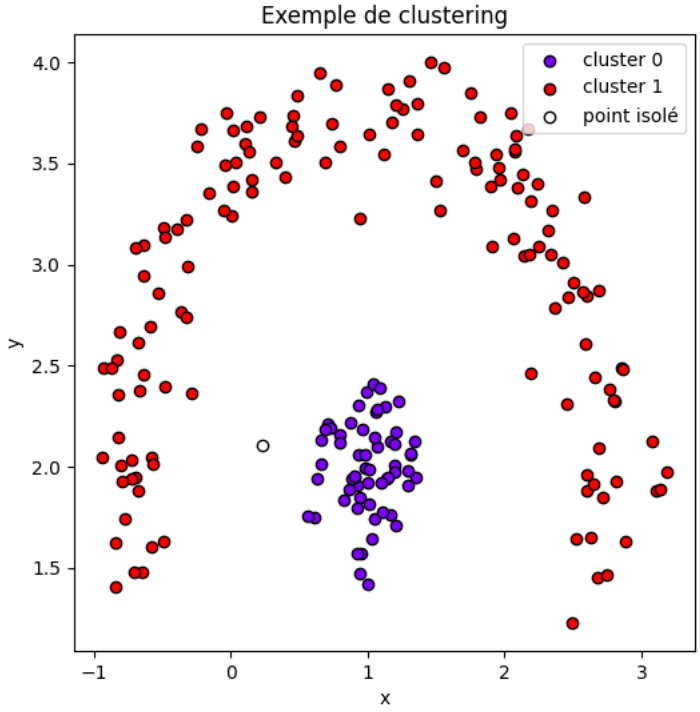
Dans le script fourni, on suppose que le projet est compilé et exécuté dans un dossier *./build* (comme en TP avec CMake). Le script Python sera dans un dossier *./plot* voisin du précédent. Les données y seront également stockées.

L'aide fournie fonctionnera lorsque le type du template i.e. le type des points sera itérable, par exemple avec *std::valarray* et comme donné dans le sujet. Sinon, le code devra être adapté. Il faut inclure *fstream*. Les librairies Python Numpy et Matplotlib seront nécessaires.

Voici la méthode venant de la classe *Clustering* qui est proposée:

```
void Clustering < T >::plot ()
{
        // test que T est du type std::valarray, peut être changé
        if (! std::is_same < T, std::valarray < double > >::value)
        {
                std::cerr << "type inconnu pour plot ()" << std::endl;
                exit (1);
        }

        // création / ouverture du fichier en mode écriture
        // l'option trunc dit qu'on efface le fichier avant d'écrire
        std::ofstream ofile("../plot/data.txt", std::ios::out |
std::ios::trunc);

        // si le fichier est ouvert
        if (ofile)
        {
                // on écrit les coordonnées de chaque point ligne par ligne
                // avec l'identifiant de cluster
                int size = M_nuage->get_nb_points ();
                for (int i = 0; i < size; ++i)
                {
                        for (auto elem : M_nuage->get_points (i))
                                ofile <<  elem << " ";
                        ofile << this->get_cluster_ids (i) << "\n";
                }

                ofile << std::flush;
        }

        // si ofile n'a pas marché, on arrête et retourne l'erreur
        else
        {
                std::cerr << "échec de l'ouverture de data.txt" <<
std::endl;
                exit (1);
        }

        // exécution du script python, on vérifie la bonne exécution
        int status = system("python3 ../plot/plot.py");
        assert(status == EXIT_SUCCESS);
}
```

Enfin, le script Python *plot.py* est fourni sur Moodle. Voici un exemple:



Bon courage.