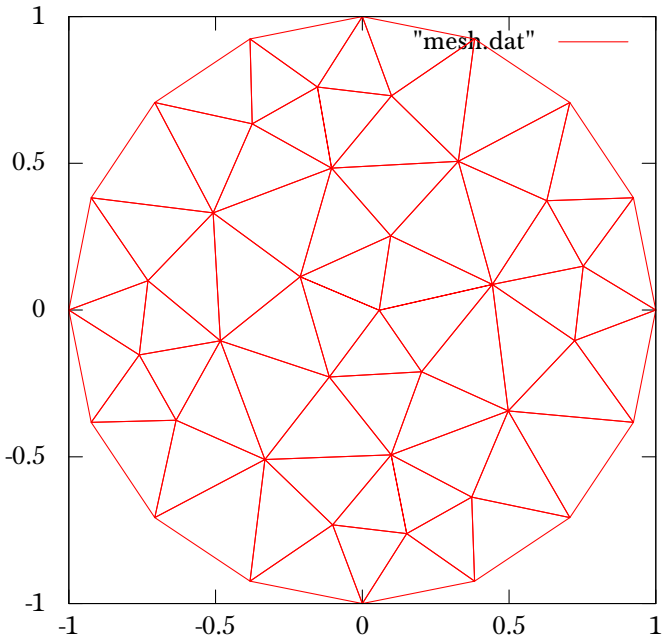
MM031 - TP7
Maillages

Maxime Chupin : chupin@ann.jussieu.fr

10 février 2016



Le but de ce TP est de créer avec soin une classe maillage que vous continuerez d'utiliser plus tard et qui pourra évoluer selon les besoins.

Les données de la classe seront composées d'une dimension `D`, d'un nombre de point `Np`, d'un nombre de triangle `Nt`, d'un vecteur `P` contenant les points (2D) du maillage, d'un vecteur `T` contenant les triangles du maillage et d'un vecteur `X` contenant dans la cas 1D la discrétisation de l'espace (on utilisera le conteneur `vector` de la librairie standard obtenue par le fichier d'entête `vector`) :

```
// fichier mesh.hpp

class mesh{
    int D;
    int Np;
    int Nt;
    vector<point> P;
    vector<triangle> T;
    vector<double> X;
};
```

Exercice 1 : Classe point

1. Définir une classe `point` constitué d'un tableau permettant de stocker les coordonnées réelles (x, y, z) d'un point géométrique.
2. Définir un constructeur qui prend comme arguments trois réels et qui construit le point associé (par défault on construira le point $(0, 0, 0)$).
3. Définir un constructeur par recopie.
4. Surdéfinir l'opérateur d'égalité.
5. Définir une fonction d'affichage.
6. Définir l'opérateur d'accès `[]` permettant d'accéder aux champs privées de la classe.
7. Tester toutes ces fonctions dans un fichier `main.cpp`.

Exercice 2 : Classe triangle

1. Définir une classe `triangle` constitué d'un tableau permettant de stocker les indices entiers `(p1,p2,p3)` de trois points.
2. Ajouter un constructeur qui prend comme arguments trois entiers et qui par défaut initialise les indices à 0.
3. Définir un constructeur par recopie.
4. Sur-définir l'opérateur d'égalité.
5. Définir une méthode d'affichage.
6. Ajouter un opérateur d'accès `[]`.
7. Tester toutes ces fonctions dans un fichier `main.cpp`.

Exercice 3 : Classe mesh

Nous utilisons donc dans cette classe `mesh`, le *conteneur* `vector`. Nous n'aurons besoin que des fonctionnalité très basiques de cette classe.

Attention, cette classe n'est pas une classe de vecteurs au sens mathématique. En particulier, elle n'implémente pas les opérations algébriques comme on pourrait l'attendre d'une classe de vecteurs mathématiques.

L'avantage d'une telle classe est la gestion de mémoire automatique. L'utilisateur peut donc aisément ajouter ou rétrécir son vecteur.

Allez voir la section 2.6 du document de Thomas Haberkorn sur la page : `https://ljll.math.upmc.fr/~chupin/&page=3` pour une brève mais suffisante introduction à cette classe.

1. Définir la classe `mesh` présentée ci-dessus.
2. Ajouter un constructeur qui prend comme arguments un entier d, un entier n, et quatre réels a, b, c et d.
   Ce constructeur attribuera d à la dimension `D`, et construira :
   — dans le cas 1D une discrétisation du segment $[a, b]$ avec n points.
   — dans le cas 2D une discrétisation du rectangle $[a, b] \times [c, d]$ avec n^2 points et $2(n-1)^2$ triangles.
   On pourra définir un comportement par défaut.
3. Définir un constructeur qui prend en argument une chaine de type `string` (qui contiendra un nom de fichier), lit le fichier et construit le maillage correspondant. Sachant que le fichier d'entrée aura le format du fichier `mesh3.mymesh` vu en cours.
4. Définir une fonction d'affichage qui trace le maillage avec `gnuplot`.
5. Tester dans le main (on pourra prendre les fichiers d'extension `.mymesh` du cours pour tester le dernier constructeur).