# Plan du code CHP

18 mars 2020

## Table des matières

## 1 Code séquentiel

### 1.1 Première résolution

Il faut construire une seule fonction qui crée une liste des points définissant un maillage, il faut donc qu'il y ait les variables :

— $N_x$ et $N_y$
— $L_x$ et $L_y$
— $h_x$ et $h_y$
— un vecteur qui représente les points (coordonnées $x$ et $y$)
— un truc (vecteur de vecteurs ?) pour représenter la matrice du problème

Il faut ensuite créer un code séquentiel en une seule fonction, qui créer une matrice, qui impose les conditions de Dirichlet sur les bords. Concernant la résolution grâce au gradient conjugué, tu peut la laisser de côté ou tu peux essayer de la coder (genre un while). Il faudrait faire des produits scalaires et des produits matriciels.

Il faut ensuite créer au moins un fichier *.cpp* regroupant les différentes fonction de test.

### 1.2 Découpage en fonctions

Il faudra regarder quelles sont les actions principales et transformer ensuite ton code séquentiel en une suite de fonctions indépendantes : comme par exemple

— Une fonction qui crée automatiquement la liste des points.
— Une fonction qui retourne la matrice du problème (on pourra la construire directement en imposant un condition de type Dirichlet sur les bords [Je ne sais pas si c'est bien utile de passer les fonctions f et g en paramètre par exemple, il faudra en discuter]).
— Une fonction qui fait le gradient conjugué.

— Une fonction qui sauvegarde les fichiers de valeurs (au format .dat).

| $\#x$ | $y$ | $u$ |
|---|---|---|
| 0.0 | 0.0 | 3.9 |
| 0.3 | 0.0 | 1.3 |
| 0.7 | 0.0 | 0.5 |
| 1.0 | 0.0 | 4.6 |

etc...

### 1.3 Optimisation du stockage

Par exemple voir pour stocker que qqls coefficients de la matrice et trouver une règle pour tous les obtenir. On en a déjà un peu discuter normalement, mais il faudra voir pourquoi c'est bien et à quel point c'est utile ou non.
Ensuite le stockage de la totalité de points est certainement superflu, on pourrait créer un espèce d'objet où on peut directement récupérer les coordonnées du point $(i, j)$ À voir au fur et à mesure je pense.

## 2 Code parallèle

On verra après une fois que ce sera bien découpé, mais ça va demander beaucoup de dessins de communications je pense étant donné que dans l'algo du gradient conjugué il y a beaucoup de produit matriciel et de produits scalaires... va falloir dessiner !

## 3 Courbes de speedup et tout leur trucs

Il va falloir revoir comment on calcule tous ces trucs là, et surtout comparer nos différentes mises en place (stockage creux obligatoire selon le sujet mais on peut essayer de comparer les deux types de stockage ?) Après il faudra comparer ce qu'il se passe lorsqu'on augmente le nombre de points sur le maillage et si on augmente le nombre de processus qui passent sur le programme (il faudrait savoir avec exactitude s'il y en a qui sont ralentis par d'autres ou s'ils travaillent tous au même rythme et tout, ça va pas être du gateau je sens. Mais après cela fait partit intégrante de l'amélioration du code... donc un peu le final de ce "gros" TP mdr)