MS01 Calcul scientifique parallèle
Master 2 AMS

TP 3 : Construction du préconditionneur ASM

Ces TPs sur la décomposition de domaine s'appuient sur un code éléments finis écrit en C++ sous la forme d'une archive `femtool.zip` disponible sur la page web du cours. Vous êtes invités à disposer de ce code à votre convenance en le modifiant si vous en ressentez le besoin pour répondre aux questions.

Exercice 1

Le but de cet exercice est de compléter les routines de paritionnement que vous avez écrites au cours du TP précédent afin que ces routines agissent non plus au niveau des maillages, mais directement au niveau des espaces éléments finis. Dans la suite on considèrera les alias de type suivant

```
using FeSpace2D = FeSpace<2>;
using FeSpace2DxCoo = std::pair<FeSpace2D,CooMatrix<double>>;
```

Question 1 Dans le fichier `partition.hpp`, ajoutez une une fonction `Restrict` qui assemble l'espace élément fini obtenu en restreignant un espace élément fini de départ à une partie de son maillage

```
FeSpace2DxCoo
Restrict(const FeSpace2D& Vh,
         const Mesh2D& Gamma,
         const std::vector<std::size_t>& tbl)
```

où les données d'entrée sont décrites ainsi :

- `Vh` est un espace élément fini construit sur un maillage `Mesh2D` noté `Omega`,
- `Gamma` est un `Mesh2D` dont les éléments forment une partie de `Omega`
- `tbl` est un vecteur de taille `Gamma.size()`, et tel que `tbl[j]` est le numéro dans `Omega` du `j`-ième élément de `Gamma`.

Les données en sortie de la fonction `Restrict` consistent en un couple `(Uh,P)` tel que :

- `Uh` est un espace élément fini défini sur `Gamma`
- `P` est un objet `CooMatrix<double>` représentant une matrice creuse $(P_{j,k})$ de taille `dim(Uh)`$\times$`dim(Vh)`. Cette matrice est booléenne, c'est-à-dire que $P_{j,k} = 0$ ou bien

$P_{j,k} = 1$. Et plus précisément, si $\varphi_i, i = 0, \ldots,$ `dim(Uh)` $- 1$ sont les fonctions de forme de `Uh`, et $\psi_i, i = 0, \ldots,$ `dim(Vh)` $- 1$ sont les fonctions de forme de `Vh`, et Γ est le domaine recouvert par le maillage `Gamma`, alors on doit avoir

$$P_{j,k} = 1 \quad \text{si} \quad \psi_k|_\Gamma = \varphi_j$$
$$P_{j,k} = 0 \quad \text{si} \quad \psi_k|_\Gamma \neq \varphi_j.$$

Question 2 On reprend la Question 4, Exercice 2 du TP 2, où on a obtenu une décomposition du maillage `Omega` du domaine $\Omega =]0,1[\times]0,1[$ en sous-domaines avec recouvrement $\Omega = \Sigma_0 \cup \cdots \cup \Sigma_3$, où chaque Σ_j est modélisé par un maillage `Sigma[j],j=0,...,3`. On suppose à nouveau ici que `Omega` a été obtenu à l'aide de `Gmsh` avec un pas de maillage $h = 0.01$.

On note `Vh` l'espace élément fini construit à partir de `Omega`, et `Uh_0` l'espace élément fini construit à partir de `Sigma[0]`. On note `fh` le vecteur des valeurs nodales obtenu en évaluant la fonction $f(x_1, x_2) = \cos(7\pi(x_1+x_2))$ au sommets du maillage `Omega`. En partant du vecteur `fh` et en utilisant la fonction `Restrict`, représentez la fonction f restreinte au sous-domaine Σ_0 avec `vizir4`.

Question 3 Dans le fichier `partition.hpp`, ajoutez une surcharge de la fonction `Partition4` prenant en argument un espace éléments finis au lieu d'un maillage

```
std::vector<FeSpace2DxCoo>
Partition4(const FeSpace2D& Vh, const std::size_t& nl)
```

Cette fonction doit :

- récupérer le maillage `Omega` de `Vh`
- construire une décomposition avec recouvrement de ce maillage par l'appel de fonction `Partition4(Omega,nl)`
- pour chaque sous-domaine `Sigma[j]` ainsi obtenu, assembler un tableau `tbl[j]` de type `std::vector<std::size_t>` établissant la correspondance entre la numérotation des éléments dans `Sigma[j]` et la numérotation de ces même éléments dans `Omega`
- Renvoyer un tableau de paire `(Sigma[0],P[0]),...,(Sigma[3],P[3])` où chaque `(Sigma[j],P[j])` a été obtenu par l'appel de la fonction `Restrict`.

Question 4 Dans le fichier `partition.hpp`, ajoutez une fonction

```
std::vector<FeSpace2DxCoo>
Partition16(const FeSpace2D& Vh, const std::size_t& nl)
```

Cette fonction réalise la même tâche que la fonction `Partition4(const FeSpace2D&, const std::size_t&)` de la question précédente, mais avec un partitionnement en 16 sous-domaines au lieu d'un partitionnement en 4 sous-domaines.

Exercice 2

Implémentez une classe `AdditiveSchwarz` modélisant un préconditionneur de Schwarz additif défini par l'équation (10) du polycopié de cours. Cette classe répondra au cahier des charges suivant.

Fonctions membres

- un constructeur prenant en argument d'entrée :
  — une matrice creuse `CooMatrix<double> A` modélisant la matrice éléments finis du problème à résoudre
  — un vecteur de matrices `std::vector<CooMatrix<double>> R` modélisant les opérateurs $\mathrm{R}_j, j = 1, \ldots, N$ introduits en page 4 du polycopié de cours.
- une surcharge d'opérateur `auto operator*(const std::vector<double>& u) const` permettant d'appliquer le préconditionneur ASM au vecteur `u`. Cette surcharge doit permettre de considérer les objets de type `AdditiveSchwarz` comme préconditionneurs pour l'algorithme `PCG` que vous avez implémenté au TP2.

Alias de type
Pour alléger les notations, on introduira l'alias de type local à la classe suivant :
`using ItemType = std::pair<InvCooMatrix<double>,CooMatrix<double>>;`

Données membres
La seule donnée membre stockée sera un tableau `std::vector<ItemType> InvAxR` de taille le nombre de sous-domaines. Dans ce tableau chaque élément `InvAxR[j]` est une paire `InvAj,Rj` définie de la manière suivante

- `InvAj` est un objet de type `InvCooMatrix<double>` qui modélise l'opérateur inverse $(\mathrm{R}_j^T \mathrm{A} \mathrm{R}_j)^{-1}$ où A est la matrice éléménts finis du problème à résoudre.
- `Rj` est un objet de type `CooMatrix<double>` qui modélise les opérateurs de plongement introduits en page 4 du polycopié de cours.

Exercice 3

On se place dans les mêmes conditions qu'à l'Exercice 2 du TP 1. On considère $\Omega =]0,1[\times]0,1[\subset \mathbb{R}^2$, $u_{ex}(\boldsymbol{x}) := \cos(10\pi x_1)$ et $f := -\Delta u_{ex} + u_{ex}$ et on s'intéresse au problème aux limites suivant

$$-\Delta u + u = f \quad \text{dans } \Omega \quad \text{et} \quad \partial_{\boldsymbol{n}} u = 0 \quad \text{sur } \partial\Omega. \tag{1}$$

Question 1 A nouveau on considère 4 maillages du domaine $\Omega =]0,1[^2$ associés aux finesses $h = 0.025$, $h = 0.01$, $h = 0.005$ et $h = 0.0025$ [1], on considère les données A et $\boldsymbol{b}$ correspondant au problème (1) après discrétisation par éléments finis $\mathbb{P}_1$. On choisit le préconditionneur de Schwarz additif implémenté à l'Exercice 2 ci-dessus et basé sur une décomposition en 4 sous-domaines avec `nl = 2` couches d'éléments supplémentaires.

On note $\boldsymbol{x}_k \in \mathbb{R}^N$ le k-ième itéré dans l'algorithme PCG. Tracez, sur la même figure, les 4 courbes représentant le résidu relatif $\|\boldsymbol{b} - \mathrm{A}\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2$ en fonction de k, où $\|\cdot\|_2$ désigne la norme euclidienne canonique. L'algorithme PCG sera stoppé lorsqu'on aura atteint un résidu relatif de $\|\boldsymbol{b} - \mathrm{A}\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2 < 10^{-10}$.

Question 2 Tracez la courbe représentant en abscisse la finesse du maillage variant entre $h = 0.05$ et $h = 0.001$, et en ordonnée le nombre d'itérations de PCG (avec préconditionneur de Schwarz additif) nécessaires pour atteindre le résidu relatif de $\|\boldsymbol{b} - \mathrm{A}\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2 < 10^{-10}$.

1. Idéalement on reprendra les mêmes maillages que ceux utilisés à la Question 5 Exercice 2 du TP 1.