TP sur les versions MPI et MPI/OpenMP du code

- On fournit une version du code, dans laquelle le découpage en sous-domaines et la mise en place des relations de voisinage sont déjà codées. Voir la fonction `init_mpi` dans le fichier `mpi_funcs.c`.

  Cette fonction initialise MPI puis remplit une structure C `mpi_vars` qui contient:

  - le communicateur MPI (`mpi_vars.comm2d`) où les processus sont organisés suivant une grille 2D
  - un tableau de 4 entiers (`mpi_vars.neighb`) qui contient le rang des processus voisins (suivant x et y) du processus courant (rangés dans l'ordre : ouest (i-1, j), est (i+1,j), sud (i, j-1) et nord (i, j+1))
  - des types utilisateurs MPI (`mpi_vars.block_dim1` et `mpi_vars.block_dim2`) qui décrivent l'emplacement des données à échanger suivant x et y

  Examiner la fonction `init_mpi`.

- Les échanges MPI consistent au début de chaque itération à envoyer aux processus voisins les valeurs des cellules de bord et à ranger les valeurs reçues de ces voisins dans les cellules miroir.

  Les échanges seront réalisés dans la fonction `make_boundary` (fichier `make_boundary.c`).

  Les variables sendToEAST, sendToWest, sendToSouth, sendToNorth (resp. recieveFromEast, recieveFromWest, recieveFromSouth, recieveFromNorth) désignent le début des zones dans le tableau des inconnues à envoyer aux (resp. à recevoir des) processus voisins.

  Ajouter les appels MPI dans la fonction `make_boundary`*.*

- La version MPI calcule la solution dans chaque sous-domaine comme un problème indépendant, en insérant au début de chaque itération, les échanges avec les problèmes sur les sous-domaines voisins.

  Comparer les versions OpenMP et MPI, remarquer que les appels MPI se font dans les parties séquentielles (pour OpenMP) du code.

  "Fusionner" les versions MPI et OpenMP, pour produire une version hybride.
  Exécuter les différentes versions et comparer les temps de calcul.