Curso HPC en MC. OpenMP [goo.gl/m6qbhV]
TPL4. Trabajo Práctico de Laboratorio. [2018-08-07]

PASSWD PARA EL ZIP: JPF8 QQVW XMJ6

Ejercicios

Dado 4 puntos **xj** (**j=1-4**) en 3D, definen un tetrahedro. Dados dos tetrahedros **Tj** y **Tk** podemos determinar si **Tj** está incluido **Tk** si todos los vertices de **Tj** están incluidos en **Tk**. Ahora bien, dado una lista de tetrahedros **(T0,T1,...)** queremos separarlos en dos conjuntos de tetrahedros que llamaremos **cover** y **notcover**, donde **notcover** contiene aquellos tetrahedros **Tj** que están incluidos en al menos uno de los otros tetrahedros **Tk** (con **k** distinto de **j**).
Los tetrahedros los guardamos en una matriz de Eigen
MatrixXd(ndim,ndim1)+ donde la columna **j** de la matriz son las coordenadas del vértice **j**.
Los tetrahedros están almacenados en un **vector<MatrixXd>**
Consigna: Escribir funciones
void getcover(vector<MatrixXd> &tetras, vector<MatrixXd> &cover, vector<MatrixXd> &notcover); que separa los tetrahedros **tetras** en los conjuntos disjuntos **cover** y **notcover** de acuerdo a lo explicado previamente utilizando OpenMP.
Se provee la función **bool inside(MatrixXd &T, VectorXd &xprobe);**
que determina si el punto **xprobe** está dentro del **símplice** definido por sus vértices **T**. El algoritmo está explicado más abajo (aunque no es necesario comprenderlo para el desarrollo del ejercicio).
Ayuda:

- Escribir la función **bool simplexinc(MatrixXd &T1, MatrixXd &T2);**
  que determina si el tetrahedro **T2** está incluido en el tetrahedro **T1**. Simplemente se van recorriendo los vértices de **T2** y se utiliza **inside()** para verificar si están todos adentro de **T1** o no.
- Para cada **j** vericar si el tetrahedro **j** está incluido en alguno de los tetrahedros **k** y en caso afirmativo incluirlo en **notcover**, caso contrario en **cover**

[Ej. 1] **[getcover-critical]** Escribir una versión paralela **getcover_critical()** usando OpenMP, para evitar race conditions sobre los contenedores **cover** y **notcover** utilizar una región crítica.

[Ej. 2] **[getcover-tidstor]** Escribir una versión paralela usando OpenMP, para evitar race conditions sobre los contenedores **cover** y **notcover** utilizar un almacenamiento para cada thread (thread-id storage) **vector< vector<MatrixXd> > cover_tid(nthreads), notcover_tid(nthreads);** De esta forma cada thread va almacenando sus tetrahedros en su área de almacenamiento independiente **cover_tid[tid]** y **notcover_tid[tid]**. Finalmente se debe conactenar todos las áreas de cad thread en el global.

[Ej. 3] **[getcover-lock]** Escribir una versión paralela usando OpenMP, para evitar race conditions usar dos locks independientes **cover_lock** y **notcover_lock** para controlar cada vez que se quiere acceder a las estructuras **cover** y **notcover**.

Verificación:
Para diferentes semillas (**seed**) las funciones deben arrojar lo siguiente:

```
seed 123456, tetras 100, cover/notcover 32/68
seed 123457, tetras 100, cover/notcover 28/72
seed 123458, tetras 100, cover/notcover 33/67
seed 123459, tetras 100, cover/notcover 41/59
seed 123460, tetras 100, cover/notcover 34/66
seed 123461, tetras 100, cover/notcover 41/59
seed 123462, tetras 100, cover/notcover 32/68
seed 123463, tetras 100, cover/notcover 31/69
seed 123464, tetras 100, cover/notcover 26/74
seed 123465, tetras 100, cover/notcover 42/58
```

NOTA: Algoritmo para determinar si un punto está incluido en un tetrahedro: basta con determinar si existen **ndim+1** escalares $0 \leq \alpha_j \leq 1$ tales que

$$\mathbf{x}_p = \sum_j \alpha_j \mathbf{x}_j \tag{1}$$

Con la restricción que $\sum_{j=1}^{n_d+1} \alpha_j = 1$. O sea que debemos resolver un sistema de $(n_d+1) \times (n_d+1)$ de la siguiente manera:

$$\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_p \\ 1 \end{bmatrix} \tag{2}$$

$$\mathbf{C}\boldsymbol{\alpha} = \mathbf{b}$$

Una vez resuelto el sistema, debe verificarse que todos los α_j estén en el rango $[0, 1]$.