Exercices de TD Parallélisme - MPI

Exercice 1

On suppose que le processus 0 possède deux vecteurs **X** et **Y** de même longueur **N**, dont il veut faire le produit scalaire **X.Y**

Rappel: [1,2,3] . [10, 20, 30] = 1*10 + 2*20 + 3*30

X, **Y** et **N** ne sont connus que par le processus 0

On suppose que l'on a la fonction de base **produitScalaire(X,Y)** qui renvoie le résultat du produit scalaire de **X** par **Y**.

Question 1: Donner l'algorithme parallèle MPI pour paralléliser le produit scalaire sur **P** processeurs.

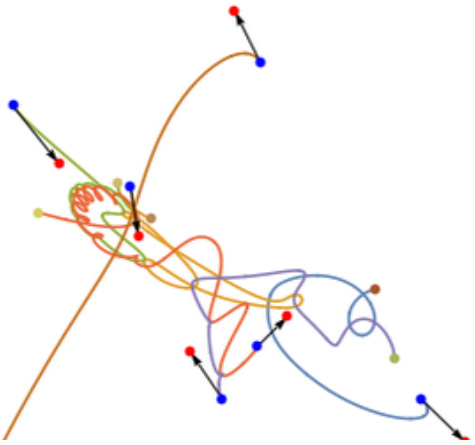
Exercice 2

On suppose avoir un vecteur de **N** réels sur P0, **P** processeurs.

Question 1: Calculer de manière parallèle combien d'éléments du vecteur sont strictement supérieurs à la moyenne. Le calcul de la moyenne doit aussi se faire en parallèle. Le résultat final est à afficher par P0.

Exercice 3 : Problème des N-Bodies

Problème des N-corps: Evolution d'un ensemble de corps qui interagissent les uns avec les autres.



Exemple en astrophysique : Le déplacement des étoiles en tenant compte des forces de gravitation.

Pour chaque paire d'étoiles, chacune applique à l'autre une force f où G est une constante, m1, m2 sont les masses respectives des deux étoiles, et x1, x2 leurs vecteurs position.

$$\vec{f} = \frac{G.m_1.m_2.(\overrightarrow{x_1}-\overrightarrow{x_2})}{|\overrightarrow{x_1}-\overrightarrow{x_2}|^3}$$

A chaque pas de temps, on calcule les forces qui s'appliquent à chacune des étoiles, puis on met à jour les données qui la concernent (position, vitesse et accélération).

La solution de base consiste à chaque itération à :

1. calculer, à chaque étape, les forces subies par chaque étoile de la part de chacune des autres étoiles
2. faire la somme de toutes les forces subies par chaque étoile
3. calculer, à partir des forces subies, les nouvelles positions, vitesse et accélération de chaque étoile et ce sur un ensemble de pas de temps (itérations) donné.

Le programme séquentiel est le suivant :

```
for t in range(0, NB_STEPS)
	force = []
	for i in range(0,N)
		force.append((0,0)) # force[i]
		for j in range(0, N)
			force[i] = force[i] !+! interaction(data[i], data[j])

	for i in range(0, N)
		data[i] = update(data[i], force[i])
```

où **data[i]** est une structure de données qui comprend la position, la vitesse et l'accélération de l'étoile **i**, **N** est le nombre d'étoiles, **interaction(data[i],[j])** est une fonction qui renvoie la force appliquée par l'étoile **j** à l'étoile **i** (cette fonction lit **data[i]** et **data[j]** mais ne les modifie pas) et **update(data[i],force[i])** est une fonction qui calcule les nouvelles position, vitesse et accélération de l'étoile **i** en fonction de la force qu'elle subit. On suppose de plus que seul le processus de rang **0** connait **data** et que le nombre de processus divise le nombre d'étoiles. Attention, pour réellement implémenter cet algorithme il faut plus de lignes de code pour additionner les couples de force lors de la mise à jours de **force[i]**

Question 1: Analyser les dépendances entre calculs et écrire une version parallèle MPI de ce code.

L'algorithme séquentiel ne tient pas compte du fait que les forces sont symétriques : si **f(i,j)** représente la force appliquée par l'étoile **i** à l'étoile **j**, on a **f(i,j) = −f(j,i)**. Il n'est donc pas nécessaire de calculer séparément l'intensité de ces deux forces.

Pour prendre en compte cette observation, on pourrait modifier l'algorithme séquentiel de calcul des forces :

```
for t in 0 , NB_STEPS
	force = [[0,0] for _ in range(N)]
	for i in 0 , N
		for j in 0 , i
			force_j_sur_i = interaction(data[i], data[j])
			force[i] = force[i] + force_j_sur_i
			force[j] = force[j] - force_j_sur_i
	for i in 0 , N
		data[i] = update(data[i], force[i])
```

Question 2: Écrire une version parallèle MPI de ce code.
Question 3: Quels sont les inconvénients de cette version ? Proposer une solution pour les atténuer.