Παράλληλη επίλυση με MPI

Συμπληρώστε και τρέξτε με το MPI των κώδικα: **fd_mpi.f90**

Οι τιμές των diag και beta υπολογίζονται στην υπορουτίνα **genSPD** του κώδικα που σας δίνετε.

Ο υπολογισμός του εσωτερικού γινομένου γίνεται στη συνάρτηση **dotproduct**

Ο υπολογισμός του γινομένου πίνακα με διάνυσμα γίνεται στην υπορουτίνα **matvec**

Η σχέση που δίνει το στοιχείο i (με i=2, 3, ..., n-1) του γινομένου Ax δίνεται από:

```
do i=2,n-1

  a=x(i-1)
  b=x(i)
  c=x(i+1)

  y(i)=(-1.)*a+(diag)*b+(-1.)*c

enddo
```

Ο πίνακας A κατανέμεται στους επεξεργαστές κατά γραμμές. Η κατανομή είναι η ίδια με αυτή που ακολουθήθηκε στην άσκηση του MPI για το εσωτερικό γινόμενο.

Στον κώδικα που σας δίνεται πρέπει να συμπληρώσετε τα κενά στα εξής σημεία:

1) function dotproduct:
Κλήση της κατάλληλης υπορουτίνας του MPI για την άθροιση της μεταβλητής local σε όλους τους επεξεργαστές.

2) subroutine matvec, subroutine communication:
Συμπληρώστε τις τιμές των μεταβλητών a, c (subroutine matvec) και τα κενά στην υπορουτίνα communication, έτσι ώστε ο πολλαπλασιασμός πίνακα με διάνυσμα να υλοποιείται σύμφωνα με το παρακάτω παράδειγμα:

Έστω n=6 και numprocs=2 (ids: 0 και 1)

myid=0 (κόκκινη διεργασία)
myid=1 (μπλε διεργασία)

$$\begin{bmatrix} 1 & & & & & \\ -1 & d & -1 & & & \\ & -1 & d & -1 & & \\ & & -1 & d & -1 & \\ & & & -1 & d & -1 \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1x_1 \\ -1x_1 + dx_2 - 1x_3 \\ -1x_2 + dx_3 - 1x_1 \\ -1x_3 + dx_1 - 1x_2 \\ -1x_1 + dx_2 - 1x_3 \\ 1x_3 \end{bmatrix} \qquad \begin{aligned} x_1 &\neq x_1 \\ x_2 &\neq x_2 \\ x_3 &\neq x_3 \end{aligned}$$

η κόκκινη διεργασία πρέπει να στείλει στην μπλε διεργασία την μεταβλητή x_3 και να λάβει από την μπλε διεργασία την μεταβλητή x_1

η μπλε διεργασία πρέπει να στείλει στην κόκκινη διεργασία την μεταβλητή x_1 και να λάβει από την κόκκινη διεργασία την μεταβλητή x_3

Για την υπορουτίνα communication συμβουλευτείτε το Παράδειγμα 6: non - Blocking message passing των παρουσιάσεων του MPI.

3) Επιλέξτε n=100, stopping_criterion=1.d-20 και iter_max=n. Τρέξτε τον κώδικα για numprocs από 1 έως 10 (mpirun -np numprocs ./a.out) και ελέγξτε τα αποτελέσματα ως προς τη μεταβλητή norm2_residual και το πλήθος των επαναλήψεων της CG καθώς και με τη λύση (αρχείο res.dat)

4) Χρησιμοποιήστε την υπορουτίνα MPI_WTIME για να χρονομετρήσετε την υπορουτίνα cg. Παράδειγμα χρήσης της MPI_WTIME:

```
real(8) stime,etime


stime=MPI_WTIME()
call cg(diag,.....)
etime=MPI_WTIME()

print *, myid,etime-stime
```

5) Φτιάξτε το διάγραμμα της παράλληλης επιτάχυνσης για n=1E4 και n=1E5.

6) Ελέγξτε την υπορουτίνα **cg.** Εντοπίστε τις λειτουργίες που αυξάνουν το υπολογιστικό κόστος και ταυτόχρονα μπορούν να αντικατασταθούν με τη χρήση προσωρινών μεταβλητών. Μετατρέψτε κατάλληλα την υπορουτίνα αυτή.