

1.

a) O processo 0 irá enviar de forma síncrona para o processo 3, o vector,  $n$  vezes. Este vector contém números do tipo `MPI_REAL`. Em cada ciclo a Tag associada à comunicação será incrementada.

b) `MPI_Recv(b, 2000000, MPI_REAL, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);`

c) `MPI_Isend(b, 2000000, MPI_REAL, i, MPI_COMM_WORLD, &request);`  
Este parâmetro extra `request`, serve para mais tarde se verificar o status da comunicação ou para esperar que a comunicação conclua.

d) `MPI_Status status;`  
`MPI_Request request;`

if ...

for ...

`MPI_Isend(b, 2000000, MPI_REAL, i, MPI_COMM_WORLD, &request);`

(...)

`MPI_Wait(&request, &status);`

for ...

e) A vantagem de se usar `MPI_Isend` é que permite avançar no programa e executar instruções enquanto a informação é enviada, no entanto é necessário garantir que a informação que está a ser enviada só pode ser modificada após a comunicação terminar.

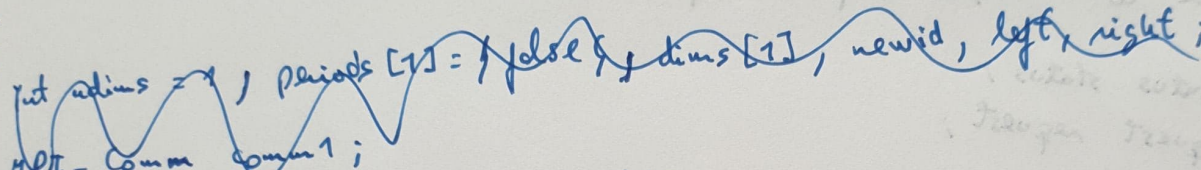


2.

a)  $MPI\_Type\_vector$  (<sup>conto-inf-direito</sup> ~~cont~~, 1,  $\frac{m}{2} \times \frac{m}{2}$ ,  $MPI\_DOUBLE$ , &stype);  
 $MPI\_Type\_commit$ (&stype);  
 $MPI\_send$ (&gr[3], 1, stype, 0, 0, comm);

b)  $MPI\_Type\_create\_subarray$ (2,  $\frac{m}{2}$ ,  $\frac{m}{2}$ , <sup>conto-inf-direito</sup> ~~start indices~~,  $MPI\_ORDER\_C$ ,  
<sup>stype</sup>  $MPI\_DOUBLE$ , &stype);  
 $MPI\_Type\_commit$ (&stype);  
 $MPI\_send$ (&gr[3], 1, stype, 0, 0, comm);

3.

a)   
 $MPI\_Comm\_comm1$ ;  
 $MPI\_Comm\_rank$ (comm1, &newid);  
 $MPI\_Cart\_shift$ (comm1, 1, 1, &left, &right);

b) No caso dos processos 0 e 3, têm apenas 10001 colunas pois o processo 0 ~~está na última~~ não pode modificar a primeira coluna e o processo 3 não pode modificar a última coluna, pois são valores fixos. No entanto têm uma coluna a mais para além dos 10000 pois o processo 0 precisa dos ghost points vindos do processo à sua direita e o processo 3 dos ghost points do processo à sua esquerda.

No caso dos processos 1 e 2, ambos têm 10002 colunas em vez dos 10000 ~~esperáveis~~, pois têm ambos uma coluna a mais à esquerda e à direita para guardarem os ghost points vindos dos seus vizinhos.

c) Inicialmente o processo 0 ~~envia~~ envia para os restantes processos, localizados à distância inicial com tamanho  $\frac{M_x}{4}$ .  
 Depois envia os valores de Tlocal para o vizinho da direita e recebe os valores do vizinho da esquerda.

Por fim envia os valores de Tlocal para o vizinho da esquerda e recebe os do vizinho da direita.

O objectivo deste programa é enviar para os processos ~~os valores que~~ <sup>os valores que</sup> ~~os valores que~~ <sup>os valores que</sup> são feitos nos seus cálculos e depois receber e actualizar os valores de Tlocal.