# Serie 6

**Exercise 1** (Parallel file writing with MPI-IO)**.** In this exercise, we'll tackle the main bottleneck of the Poisson 2D problem in parallel : the Input/Output using MPI-IO (MPI-2). You should have a working MPI version from serie5. If not, one is available on the git repository.

This MPI version divided the problem into horizontal 2D subdomains declared as :

```
float **u_local;
float **uo_local;
float **f_local;
```

Remember : your data should be contiguous in memory in order to be written on disk with a decent performance. In the current state, you could call `N/p` times `MPI_File_write()` which is not very efficient. Transform the 2D arrays in your code into 1D arrays that represent a 2D arrays. This can be done efficiently using a macro :

```
#define U (r,c) (u_local  [(r)*(N + 1) + (c)])
#define Uo(r,c) (uo_local[(r)*(N + 1) + (c)])
#define F (r,c) (f_local  [(r)*(N + 1) + (c)])
```

Then your local array is declared and accessed in that way :

```
float  *u_local;
float  *uo_local;
float  *f_local;

...

u_local  = (float*) malloc(N_local*N*sizeof(float));
uo_local = (float*) malloc(N_local*N*sizeof(float));
f_local  = (float*) malloc(N_local*N*sizeof(float));

...

for(i=1;i<N_local-1;i++){
   for(j=1;j<N-1;j++){
      U(i,j) = 0.25*(Uo(i-1,j) + Uo(i+1,j) + Uo(i,j-1) + Uo(i,j+1)
                     - F(i,j)*h*h);
      l2 += (Uo(i,j) - U(i,j))*(Uo(i,j) - U(i,j));
   }
}
```

The second problem with the existing MPI version is that it writes the checkpoints as *.pgm* files. These files are ASCII files and thus, unusable with MPI-IO. Use a modified version of `int write_to_file` that write the files in BMP format with a 1D array (an example is on the git repository):

You are now ready to write the checkpoints (files) in parallel. You are then asked to :

- Make your arrays 1D (contiguous memory)

- Parallelize the IO using a parallel version of `int write_to_file_binary`
- check the consistency of the results with the reference MPI version