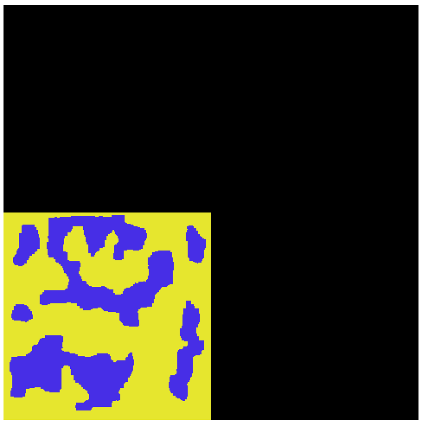
Assignment 4 Report

GETTING MPI UP AND RUNNING

First added to comm.c MPI calls to initialise, finalise, store rank and size, and print the run time



Initial snapshot of the global grid after 1000 MC cycles on 4 processors. Each processor repeats the same work on only the bottom left sub-grid because no cartesian topology is set up and no grid_spin communications are in place.

SETTING UP CARTESIAN TOPOLOGY

Added following MPI calls to comms_processor_map:

```
MPI_Cart_create(MPI_COMM_WORLD,2,dims,pbc,reorder,&cart_comm);
MPI_Comm_rank(cart_comm,&my_rank_in_cart);
MPI_Cart_coords(cart_comm,my_rank_in_cart,2,my_rank_coords);
MPI_Cart_shift(cart_comm,y,+1,&my_rank_neighbours[down],&my_rank_neighbours[up]);
MPI_Cart_shift(cart_comm,x,+1,&my_rank_neighbours[left],&my_rank_neighbours[right]);
```

The calls format the pre-defined cart_comm as a cartesian communicator, assign the current rank in cart_comm to my_rank_in_cart (same as my_rank as reorder is 0), assign the coordinates of the current rank to my_rank_coords, and assign the neighbouring ranks of the current rank to the elements of the array my_rank_neighbours.

Lines from print statement:

```
1 processor:  I am rank 0 with coords 0 0 and my neighbours are ranks (left, right, down, up): 0 0 0 0

4 processors: I am rank 0 with coords 0 0 and my neighbours are ranks (left, right, down, up): 2 2 1 1
              I am rank 1 with coords 0 1 and my neighbours are ranks (left, right, down, up): 3 3 0 0
              I am rank 2 with coords 1 0 and my neighbours are ranks (left, right, down, up): 0 0 3 3
              I am rank 3 with coords 1 1 and my neighbours are ranks (left, right, down, up): 1 1 2 2

9 processors: I am rank 0 with coords 0 0 and my neighbours are ranks (left, right, down, up): 3 6 2 1
              I am rank 1 with coords 0 1 and my neighbours are ranks (left, right, down, up): 4 7 0 2
              I am rank 2 with coords 0 2 and my neighbours are ranks (left, right, down, up): 5 8 1 0
              I am rank 3 with coords 1 0 and my neighbours are ranks (left, right, down, up): 6 0 5 4
              I am rank 4 with coords 1 1 and my neighbours are ranks (left, right, down, up): 7 1 3 5
              I am rank 5 with coords 1 2 and my neighbours are ranks (left, right, down, up): 8 2 4 3
              I am rank 6 with coords 2 0 and my neighbours are ranks (left, right, down, up): 0 3 8 7
              I am rank 7 with coords 2 1 and my neighbours are ranks (left, right, down, up): 1 4 6 8
              I am rank 8 with coords 2 2 and my neighbours are ranks (left, right, down, up): 2 5 7 6
```

Each rank in cartesian grid has correct neighbours, working on different parts of the global grid now.

Cameron Fantham

COLLECTING DATA FROM ALL PROCESSORS

Added MPI reduce call to comms_get_global_mag:

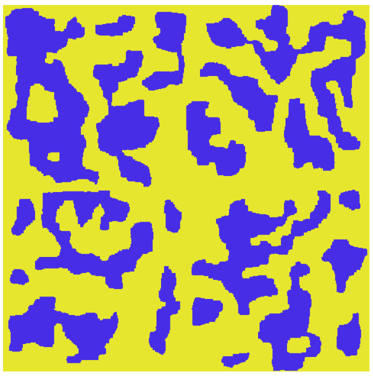
```
MPI_Reduce(&local_mag,global_mag,1,MPI_DOUBLE,MPI_SUM,0,MPI_COMM_WORLD);
```

Sum the values of local_mag across all processors into one value global_mag, on rank 0. This is then divided by the number of processors/ranks to give the average global magnetisation.

Added following MPI calls to comms_get_global_grid:

```
MPI_Recv(remote_domain_start,2,MPI_INT,ip,0,MPI_COMM_WORLD,&status);
MPI_Recv(combuff,grid_domain_size,MPI_INT,ip,1+iy,MPI_COMM_WORLD,&status);
MPI_Send(grid_domain_start,2,MPI_INT,0,0,MPI_COMM_WORLD);
MPI_Send(grid_spin[iy],grid_domain_size,MPI_INT,0,1+iy,MPI_COMM_WORLD);
```

On rank 0 receive into grid_domain_start into remote_domain_start from all other ranks ip with tag 0; other ranks send grid_domain_start to rank 0 with tag 0. On rank 0 receive current row of grid_spin (denoted by iy) into combuff from all other ranks ip with tag 1 + iy; other ranks send grid_spin[iy] to rank 0 with tag 1 + iy.



Completed grid on 4 processors with appropriate MPI sends, receives and reduce in place, but with halo array set to spin 1 for all sub-grids (no halo swaps comms in place), hence colour is yellow along each sub-grid edge.

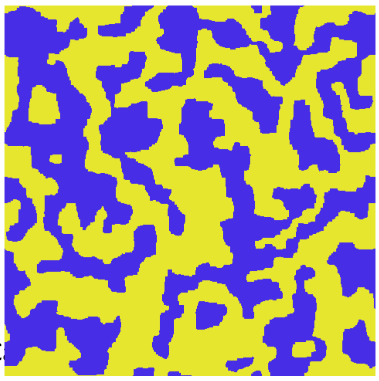
HALO SWAPPING

Added lines to complete comms_halo_swaps routine:

```
for (iy = 0 ; iy < grid_domain_size ; iy++) { sendbuf[iy] = grid_spin[iy][0]; }
MPI_Sendrecv(sendbuf,grid_domain_size,MPI_INT,my_rank_neighbours[left],my_rank_neighbours[left],recvbuf,grid_d
omain_size,MPI_INT,my_rank_neighbours[right],my_rank,cart_comm,&status);
for (iy = 0 ; iy < grid_domain_size ; iy++) { grid_halo[right][iy] = recvbuf[iy];
```

(plus 3 similar blocks for sending and receiving right, down, up into halo arrays)

Copy left edge elements of grid_spin into sendbuf then send sendbuf to left neighbour. At the same time, receive left edge elements from right neighbour (in the form of recvbuf) and copy them into grid_halo[right]. The send and receive tags ensure tags for sending and receiving always match, avoiding deadlocks.



Completed grid on 4 processors with appropriate MPI sends, receives and reduce in place, with communication between processors to correctly fill out halo arrays of each sub-grid, allowing blue colour to span between sub-grids. Global grid is periodic.

C

Steps to achieve this:

1) Complete Initialise and finalise routines
2) Complete comms_processor_map routine with MPI calls listed above
3) Complete comms_get_global_grid routine with MPI calls listed above
4) Complete comms_get_global_mag routine with MPI calls listed above
5) Complete comms_halo_swaps routine with MPI calls listed above

TIMINGS

Processors Grid Size	P = 1	P = 4	P = 9	P = 16	P = 25
480	Time: 16.599171 s Speed-up: 0.0	Time: 3.978406 s Speed-up: 4.1723	Time: 2.377951 s Speed-up: 6.9804	Time: 1.136461 s Speed-up: 14.6060	Time: 0.844639 s Speed-up: 19.6524
600	Time: 29.736194 s Speed-up: 0.0	Time: 6.766447 s Speed-up: 4.3946	Time: 3.096900 s Speed-up: 9.6019	Time: 1.800676 s Speed-up: 16.5139	Time: 1.237043 s Speed-up: 24.0381
720	Time: 48.989375 s Speed-up: 0.0	Time: 11.62862 s Speed-up: 4.2128	Time: 5.018595 s Speed-up: 9.7616	Time: 2.619646 s Speed-up: 18.7008	Time: 1.789396 Speed-up: 27.3776
840	Time: 75.052110 s Speed-up: 0.0	Time: 17.875471 s Speed-up:4.1986	Time: 7.675335 s Speed-up: 9.7783	Time: 3.767711 s Speed-up: 19.9198	Time: 2.448146 s Speed-up: 30.6567

Speedup generally increases with increasing P, and seems to increase with Ngrid as well, which is more noticeable for greater values of P. The increase in speedup with Ngrid has diminishing returns as Ngrid keeps increasing; for example, the speed-up from 480 to 600 is greater than that of 720 to 840, due to a non-linear dependence, possibly polynomial. Whereas the speedup increase from increasing P seems to be increasing somewhat linearly.

Cameron Fantham