Homework 5 review

Alltoall barrier

```
integer :: p, rank, comm, ierror, sendbuff(0:p-1)
integer :: recvbuff(0:p-1)
sendbuff = 0
recvbuff = 0
call mpi_alltoall(sendbuff, 1, mpi_integer, recvbuf, mpi_integer, comm, ierror)
return
```

Non-blocking

```
integer :: req_array(1:2+(p-1)), count, flag
flag = 0
do i = 1, p-1
  call mpi_irecv(flag, 1, mpi_integer, mpi_any_source, 0, comm, req_array(i), ierror)
enddo
count = p
do i = 0,p-1
  if(i /= rank) then
    call mpi_isend(flag, mpi_integer, i, 0, comm, req_array(count), ierror)
    count = count+1
  endif
enddo
```

Central manager

```
if(rank == 0) then
  do i = 1, p-1
    call mpi_recv(flag, 1, mpi_integer, mpi_any_source, mpi_status_ignore, ierror)
  enddo
else
  call mpi_send(flag, 1, mpi_integer, i, 0, comm, ierror)
endif

call mpi_bcast(flag, ...)
```

Homework 6 Review

```
t1 = mpi_wtime()
call mpi_isend(A(1,1), n+n, dp, 0, request(1), ierror)
call mpi_irecv(....., request(2), ...)
call mpi_waitall(2, request, mpi_statuses_ignore, ierror)
t2 = wtime()
time = t2-t1
```

Domain decomposition

Standard procedure for solving PDEs in parallel