

# FFLUX v3.0 Manual

Benjamin C. B. Symons

March 28, 2022

# 1 Introduction

FFLUX is written as an add on to DL\_POLY 4.08 (Note two files have been taken from version 4.09 for bug fixes). As such the DL\_POLY manual is a good starting place for understanding FFLUX. The purpose of this manual is to outline what FFLUX adds above and beyond standard DL\_POLY. This manual is a work in progress.

FFLUX is written in FORTRAN90 and is integrated into DL\_POLY's domain decomposition (DD) MPI implementation. FFLUX also contains some OpenMP (OMP) around key loops.

This manual is delivered alongside the developer manual. The latter is an in depth look at the FFLUX source code and is intended for developers (or those that are interested). Note that a set of example inputs can be found on the GitHub page in the unit tests folder. Before running anything read the "Important Notes" section at the end of this manual.

## 2 How to run

In order to compile and then DL\_FFLUX you must load some Fortran compilers. We typically use Intel compilers, others should work in principle but in practice your mileage may vary. On csf load compilers with *module load mpi/intel-18.0/openmpi/3.1.4* or on ffluxlab load the compilers with *module load compilers/intel/18.0.3* (these are the 2018 compilers, feel free to use others). In order to compile the code on CSF/ffluxlab, simply go to the source directory and then type *make hpc*. There is also an archer target that works to compile on ARCHER2 at the time of writing.

Figures 1 and 2 show some example jobscripts. Figure 1 is a pure MPI run with 8 MPI processes (1 process = 1 core) and Figure 2 is a hybrid OpenMP + MPI run which is using a total of 16 cores: 4 MPI which each spawn 4 OpenMP threads. To run in serial then simply set both OMP\_NUM\_THREADS and -np to 1.

```
#!/bin/bash
#$ -S /bin/bash
#$ -V
#$ -cwd
#$ -pe smp 8
#$ -N TEST
#$ -j y

export OMP_NUM_THREADS=1
mpirun -np 8 ~/development/dl_fflux_3.0/Code/execute/DLPOLY.Z &> Energies
```

Figure 1: Example jobscript (ffluxlab) to run DL\_FFLUX with 8 MPI processes.

```

#!/bin/bash
#$ -S /bin/bash
#$ -v
#$ -cwd
#$ -pe smp 16
#$ -N TEST
#$ -j y
export OMP_NUM_THREADS=4
mpirun -np 4 ~/development/dl_fflux_3.0/Code/execute/DLPOLY.Z &> Energies

```

Figure 2: Example jobscript (ffluxlab) to run DL\_FFLUX with 4 MPI processes and 4 OpenMP threads per process (16 cores total).

## 3 Files

### 3.1 CONFIG

The CONFIG file is largely the same as in vanilla DL\_POLY. However, on each line with an atom name, the name of the model for that atom must be included as well e.g. If the model file for atom C95 is METHANOL\_IQA\_C1.model then in the CONFIG you need the line:

C 95 METHANOL\_C1.

Note: For now, the model names in CONFIG must appear in numerical order.

### 3.2 CONTROL

The following is a summary of the extra options that have been added to the CONTROL file in FFLUX. New commands can be added using the DL\_POLY routine *read\_control.f90*.

Keyword	Description
<b>fflux ewald L1</b>	Sets the electrostatics method as Ewald. L1 refers to point charge only. Multipoles up to hexadecapole can be enabled (L5). Needs to be accompanied by standard Ewald sum settings used in vanilla DL_POLY e.g. <b>spme precision 1d-6</b>
<b>fflux cluster L1</b>	Sets the electrostatics method as cluster. This should not be run with PBC's or in parallel (at the moment). All pairs of atoms not in the same model will interact electrostatically.
<b>fflux print i j</b>	Print energies (IQA, electrostatic, vdW) to FFLUX file every j steps starting at step i.
<b>print dipole i j</b>	Prints intraatomic dipole moments as well as the information need to calculate charge transfer dipole moments to the DIPOLES file. Starting at step i, every j steps. (Note this printing is not MPI safe at the moment).
<b>fflux conv</b>	Switches on the electrostatic convergence mode in DL_FFLUX.
<b>fflux energy</b>	Prints out the IQA energies for each atom.
<b>fflux force</b>	Prints out the IQA forces for each atom.

### 3.3 FIELD

The FIELD file is largely unmodified from its DL\_POLY form. However, there are some key points to note. FFLUX is written such that it repurposes a large amount of the existing DL\_POLY code, as such some values in FIELD need to be 'spoofed'.

- If multipole electrostatics is to be used then the **Multipolar x** keyword needs to be present in the FIELD file per the DL\_POLY manual. (x is that rank of the highest enabled multipole).
- Molecules in FFLUX are flexible. However, DL\_POLY does not know this and as such, spoof harmonic bonds and angles/dihedrals should be set in FIELD between all atoms in a given model. Bonds should be given parameters of 0. There are multiple ways of spoofing but, the simplest is to set a 'bond' between every atom pair in the molecule.
- Note that FFLUX follows the full rank convention for multipolar interactions in Ewald (as does DL\_POLY). i.e. If multipoles up to rank x are enabled then all interactions up to and including x-x will be computed. However, in the cluster electrostatics code the rank of interaction is given by  $L = l_A + l_B + 1$  i.e. not full rank.

### 3.4 MPOLES

Similar to the FIELD file, the MPOLES file needs to be spoofed. Multipole moments in FFLUX are predicted every time step so there is no need to specify them at the start.

However, including an MPOLES file is necessary if multipoles above point charge are enabled (as in vanilla DL\_POLY). This ensures creation of the necessary multipolar arrays. The MPOLES file can be filled with any spoof values. I think zeros are fine but to be on the safe side fill with non-zero values (it may impact something in *set\_bounds.f90*).

### 3.5 model\_krig directory

This directory is unique to FFLUX and contains all of the GPR model files. Note DL\_FFLUX will automatically determine how many models files there.

## 4 Convergence mode

The convergence mode in DL\_FFLUX allows you to run electrostatic convergence tests on dimers (currently they must be dimers of two of the same molecule). You can either use predicted moments or supply AIMAll moments in an additional input file called *cMoments.txt*. To switch between modes, you must currently go into the file *fflux\_electro\_converge.f90* and switch the boolean **Pred** between true and false for predicted and AIMAll moments respectively. If AIMAll moments are provided then they must be in global Cartesian form and in DL\_POLY units i.e. the Cartesian moments can be taken directly from AIMAll outputs with just a unit conversion from Bohr to Å. For a given  $L'$ , only the relevant moments should be provided e.g. for  $L'=1$ , the file *cMoments.txt* should contain just the charges and dipole moments for every atom in the dimer. The format is one multipole moment component per line and the order of the atoms should match CONFIG and FIELD e.g. for  $L'=0$  and a dimer of 6 atoms, there will be a total of  $4 \times 6 = 24$  lines. The output will be of the format  $i\ j\ E_{ij}$  i.e. a pair of atoms and the energy of their interaction (in kJ/mol).

## 5 Important Notes

- Currently the order of the model names in the CONFIG file must be in increasing numerical order.
- Cluster electrostatics should **NOT** be used with PBC's or in parallel. It is only for small clusters run in serial.
- The number of OpenMP threads must always be set in the job script, even if it is 1 (OpenMP is bad and if the number is not specified it will just use the max available).
- MPI is only enabled and functional for  $L'=0,1,2$ . There is no MPI support for  $L'=3,4$ .
- When the dipole printing is done, the writes are not currently MPI safe so the code must be run in serial.
- If, when running in parallel you get the error "link cell algorithm in contention with SPME sum precision" then increase the precision of the SPME sum (e.g. from  $1d-6$  to  $1d-7$ ) and it should work.
- If using OpenMP (OMP), never set OMP\_PROC\_BIND to true.

- There is a DL\_POLY bug in code that is called for DL\_FFLUX but not actually used (i.e. it is called but doesn't do anything useful). The bug means DL\_POLY will get stuck in an infinite while loop if you have a linear molecule. The while loop is in the file mpoles\_container.f90 in the rotate\_mpoles and rotate\_mpoles\_d routines. If needed, it can be commented out without causing any problems for DL\_FFLUX.
- Currently longest allowed model name is 64 characters. This should be way more than enough but if not then it can be changed in fflux\_module.f90.