Ludwig User Tutorial

Fraser Mackay and Oliver Henrich

July 19, 2022

Contents

1 Introduction

This tutorial document takes new users of the `Ludwig` code through the process of how to obtain and build the code before outlining some short tutorial exercises where simulations of simple example cases will be run and visualised. It is assumed that the reader is familar with UNIX-based operating systems and has a basic knowledge of hydrodynamics, complex fluids, and to some extent statistical physics. This knowledge will be required to make sense of the input and output involved in using the code.

The system requirements are an up-to-date version of a C compiler as well as working installations of a text editor like `vi` or `emacs`, the visualisation tool ParaView (https://www.paraview.org), the MPI-library for compiling the parallel version of the code and `gnuplot` for simple visualisations. The instructions for compilation of this tutorial and the full documentation assume a LaTeX distribution and `pdflatex` are installed.

2 Access and Compilation

2.1 Obtaining the Code

The `Ludwig` project is currently hosted at our central repository at GitHub: (https://github.com/ludwig-cf/ludwig).

You can either download or clone the repository. The following command clones the repository and requires a working git installation on your local system:

```
$ git clone https://github.com/ludwig-cf/ludwig
```

A copy of the `Ludwig` repository will now be available in your current directory. You should see:

```
$ ls ludwig
CHANGES.md     Makefile      config        src           util
CONTRIBUTING.md Makefile.mk  docs          target        version.h
LICENSE        README.md     mpi_s         tests
```

A full documentation of `Ludwig` can be obtained by running the command `make pdf` twice (for references) in the `/docs` directory:

```
$ cd ./ludwig/docs/
$ make pdf
```

This tutorial and directories with input files and sample output is contained in a subdirectory of `/docs`. It can be compiled in the following way (issue twice for TOC):

```
$ cd ./ludwig/docs/tutorial
$ make
```

2.2 Configuration and Build

The following section outlines the configuration and compilation procedure. The configuration step is based on the GNU Compiler Collection gcc.

1. Go to the top level directory `/ludwig`:

   ```
   $ cd ludwig
   ```

2. Copy the (parallel) configuration file `lunix-mpicc-default.mk` to top level directory and rename to `config.mk` (both steps exectued below at once):

   ```
   $ cp config/unix-mpicc-default.mk ./config.mk
   ```

Note that you may need to specify the correct path to the MPI wrapper compiler in this file (assumed to be `mpicc`).

For production runs you may want to use thie compiler flag below instead of the default one:

```
CFLAGS = -O3 -g -DNDEBUG -DNSIMDVL=4
```

3. Issue `make` in the top level directory:

   ```
   $ make
   ```

   This creates the executable file `./Ludwig.exe` in `/src`. Note that more than one thread can be used in the compilation process, which can lead to a considerable speedup

   ```
   $ make -j 4
   ```

   If you want a serial build, clean your directory tree with

   ```
   $ make clean
   ```

   use the conifguation file `config/unix-gcc-default.mk` and include the extra keyword `serial`:

   ```
   $ make serial
   ```

3 Tutorial Test Exercises

3.1 Test 1: Poiseuille Flow of a Simple Fluid in a Cavity

In this section the Poiseuille flow of a simple fluid in a cavity, is outlined. The example is a step-by-step guide for performing a short run ($10,000$ time-steps) with `Ludwig` code, followed by a short introduction to the post-processing and data analysis procedures. Before starting, the code should be compiled in parallel and the executable `Ludwig.exe` should be copied into your working directory.

1. Copy the input file `input` located in the directory `/docs/tutorial/test1` to your working directory. Note that the input files in this tutorial are modified versions of the reference input file `input.ref` in `/src` with a limited set of parameters for each run.

2. Run the code in your working directory by issuing the command:

   ```
   $ mpirun -np 4 ./Ludwig.exe input
   ```

3. You should now see statistics being reported on your standard output and output files being created with names of the form `vel-XXXXXXXX.001-001`. These files contain the components of the velocity vectors in Cartesian coordinates at each lattice point in the simulation.

4. The first step to the post-processing is to get hold of the `extract` executable in `/util`, which is created during the main compilation of `Ludwig`, and copy it into your tutorial working directory.

5. Post-process the raw data by issuing

   ```
   $ ./extract -a -i meta_file_name data_file_name_stub
   ```

   replacing the filenames with those of the present case (i.e. `vel.001-001.meta` and e.g. `vel-00010000`). The command line options '`-a`' and '`-i`' give output in ASCII format and a lattice index for easy visualisation in `gnuplot`.

6. You should now see files named `vel-XXXXXXXX`. The first three columns of these ASCII files are the spatial lattice indices followed by the Cartesian components of the velocity vectors.

7. To plot the velocity profile of the system, you can plot the y-component of the velocity against the spatial x-coordinate on the lattice with the command:

   ```
   gnuplot> p 'vel-XXXXXXXX' u 1:5
   ```

   By plotting the data in each file, you should be able to see the velocity profile converging to the final profile as shown in Fig. 1.

   In the directory `/docs/tutorial/test1` a reference file of the final output at time step $10,000$ can be found, in addition to a file containing the standard output of the code.

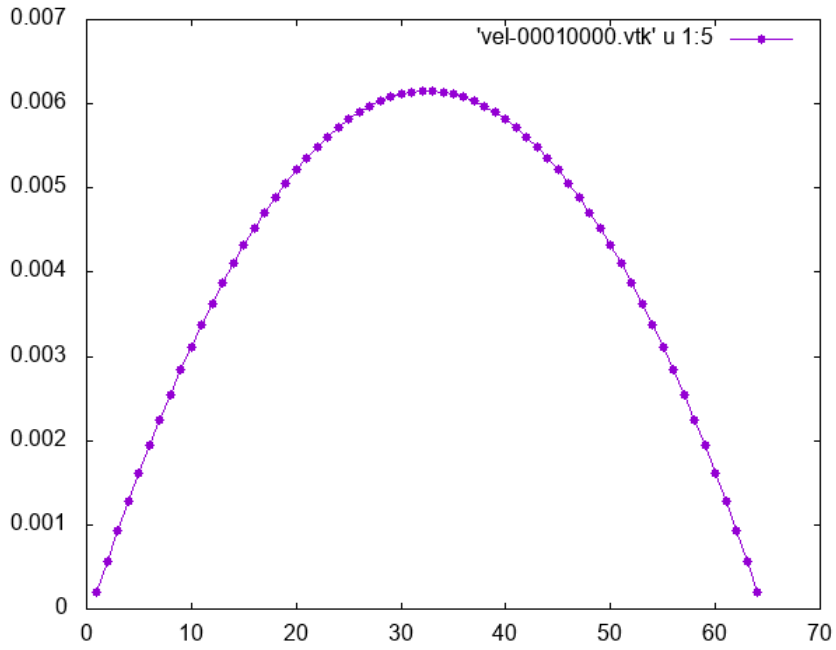

Figure 1: The parabolic velocity profile of a simple fluid in a cavity after $10,000$ time-steps.

3.2 Test 2: Spinodal Decomposition of a Binary Fluid

This section outlines the second example, the spinodal decomposition of a binary fluid in three-dimensions. The example describes the post-processing procedure for this system and a brief guide to the visualisation of a 3D system using ParaView. As in the previous example, the code should be compiled in parallel and the executable `Ludwig.exe` should be copied into your working directory.

1. Copy the input file `input` located in the directory `/docs/tutorial/test2` to your working directory. This input file contains the parameters required for this run.

2. Run the code in your working directory by issuing the command:

   ```
   $ mpirun -np 8 ./Ludwig.exe input
   ```

3. As before, you should now see statistics being reported on your standard output. But in this case two sets of output files are being created, one containing the velocity and the other compositional order parameter (ϕ) data at each lattice point in the simulation. In this example, these data are in binary format.

4. You are able to use the same exectuable `extract` from the previous example for the post-processing with different arguments and the `-k` flag to produce files in VTK-format:

```
$ ./extract -k vel.001-001.meta vel-00010000.001-001
$ ./extract -k phi.001-001.meta phi-00010000.001-001
```

5. A bash script like this one below can be used for convenient post-processing of multiple raw data files and time steps (with appropriate start, increment and end in the `for` loop):

```
#!/bin/bash
for i in `seq 1000 1000 10000`;
do
  tstep=$(printf "%08d" $i)
  ./extract -k vel.001-001.meta vel-${tstep}.001-001
  ./extract -k phi.001-001.meta phi-${tstep}.001-001
done
```

6. You should now see files with the extension `.vtk` corresponding to the velocity and ϕ data ready to be visualised using ParaView.

   (a) In Paraview open the files containing the ϕ data.

   (b) Click on the group of files in the pipeline browser and apply a contour map using Filters/Common/Contour or in the tool bar:

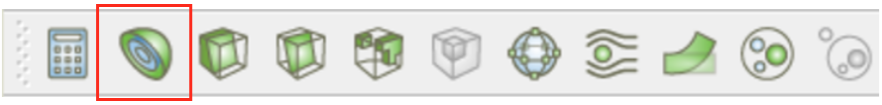

Figure 2: The contour button in the tool bar.

   You should now see a three-dimensional rendering of the binary fluid.

   (c) Open the files containing the velocity data.

   (d) Click on the group of files in the pipeline browser and apply the glyph filter using Filters/Common/Glyph or in the tool bar:

Figure 3: The glyph button in the tool bar

   (e) Change the colour code to GlyphVector Magnitude.

   (f) In the pipeline browser, you should see:

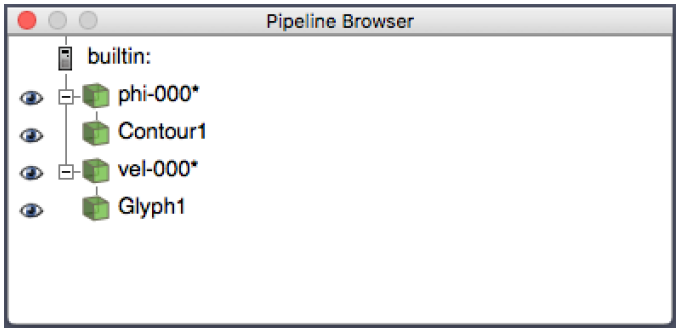

Figure 4: The complete pipline browser for the visualisation of spinodal decomposition.

A 3D visualisation of the system should also now be shown:

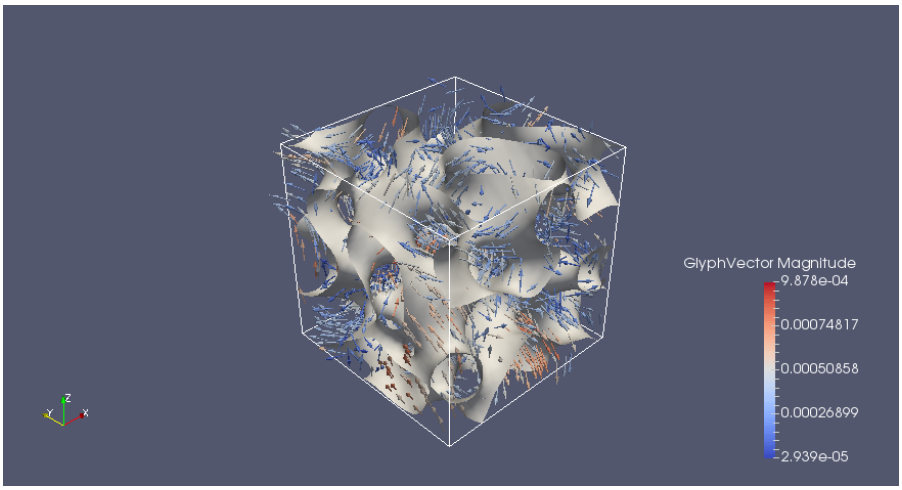

Figure 5: A visualisation of a separating binary fluid using ParaView.

(g) You can move through different frames of the simulation with the arrow buttons. As you do this, you should see the coarsening of the binary fluid.

Figure 6: Control buttons for the visualisation in ParaView.

(h) In the File menu you can save a screen shot of the visualisation or save the state of the system in order to reload the entire visualisation.

3.3 Test 3: Colloids Dispersed in a Liquid Crystalline Fluid

For this third example, the system being used is a that of colloidal particles suspended in a liquid crystalline fluid. The example illustrates the post-processing procedure for a colloidal system in addition to demonstrating the ways to visualise liquid crystals in ParaView. As before, the code should be compiled in parallel and the executable `Ludwig.exe` should be copied into your working directory.

1. Copy the input file `input` located in the directory `/docs/tutorial/test3` to your working directory. We see in this input file that there are many more parameters

than in the previous examples. Specifically, we note that `colloid_cell_min` must be set if colloids are present and must be at least $2r + \delta$ where r is colloid radius and δ - here set to 0.5 - catches any colloid-colloid interactions. This is shown for our example - where this value is set to 8.0 - in Fig. 7.

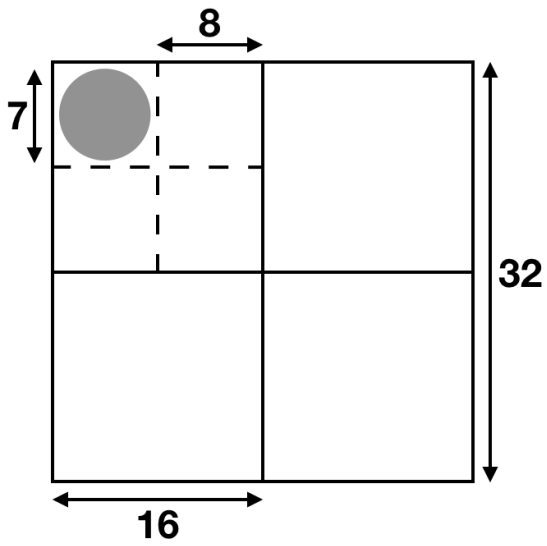

Figure 7: A diagrammatic representation of a colloid in a 32^3 system showing that the value, `colloid_cell_min` must be set such that it can contain the entire colloid plus colloid-colloid interactions.

2. The following steps outline the preliminary stages required in order to initialise the colloidal system and for the visualisation of colloids. Firstly, and as before, compile the `Ludwig` code in serial (see section 2.2) and go to the directory `ludwig/util`. There you should see the files `colloid_init.c` and `extract_colloid.c`. These are the files required for the initialisation and post-processing of colloids, respectively.

3. In the file, `colloid_init.c` set the flags for:

   (a) `NMC = 0`,

   (b) `periodic[3] = 1,1,1`
   for our fully periodic system,

   (c) `file_format=ASCII`
   resulting in ASCII output,

   (d) `a0 = 3.5`
   which sets the radius of the colloids,

   (e) `ah = 3.5`
   which sets the hydrodynamic radius of the colloid,

   (f) `vf = 0.02`
   this sets the volume fraction of colloids to 2%, for our system of size 32^2 with colloids of radius 3.5, this will result in 3 colloids,

   (g) `dh = 0.5`
   to set the grace distance for interactions (note that this means that one colloid plus this distance is less than `colloid_cell_min` which was set in the input file.

4. Now issue

```
$ make colloid_init
$ ./colloid_init
```

this produces a file `config.cds.init.001-001`, which you have to copy into your working directory.

5. Next, run the code in your working directory by issuing the command:

```
$ mpirun -np 8 ./Ludwig.exe input
```

6. An `extract_colloids` executable is produced in `/util` during the main compilation of `Ludwig`. For our problem we need to set potentially different parameters. In the file, `extract_colloids.c` set:

   (a) The system size, `NX`, `NY`, `NZ` all = 32,

   (b) `iread_ascii = 1`
   since the output is in ASCII format,

   (c) `cds_with_v = 1`
   which will output the positions and velocity data of the colloids,

7. Now issue

```
$ make extract_colloids
```

and copy the executable `extract_colloids` into your working directory.

8. For the post-processing of other data either copy the existing executable `extract` from previous examples or compile in `/util` in the usual manner.

9. Run the post-processing routines as before using different arguments:

```
$ ./extract -k vel.001-001.meta vel-00010000.001-001
$ ./extract -k -s -d q.001-001.meta q-00010000.001-001
$ ./extract_colloids config.cds00010000 1 col-cdsvel-00010000.csv
```

The `-s` and `-d` flags produce scalar order parameter and director field output. `.csv` files with the colloid data should be seen in addition to the `.vtk` files for the velocity, order parameter and director field.

10. Again, a bash script like this one below can be used for convenient post-processing of multiple raw data files and time steps (with appropriate start, increment and end in the `for` loop):

```
#!/bin/bash
for i in ‘seq 1000 1000 10000‘;
do
  tstep=$(printf "%08d" $i)
  ./extract -k vel.001-001.meta vel-${tstep}.001-001
  ./extract -k -s -d q.001-001.meta q-${tstep}.001-001
  ./extract_colloids config.cds${tstep} 1 col-cdsvel-${tstep}.csv
done
```

11. The system can now be visualised, again using ParaView:

   (a) The liquid crystal, scalar order parameter can be visualised by opening the files: `lcs-XXXXXXXX.vtk` and applying the ‘contour’ filter. By setting a threshold minimum 1×10^{-4} the order parameter in the colloids can be ignored, this can be done using the tool bar:

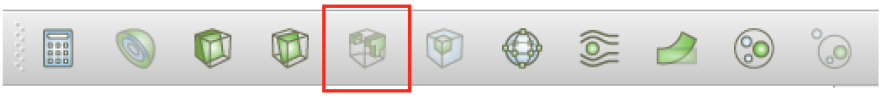

Figure 8: The threshold button in the tool bar.

(b) For the director field open the `lcd-XXXXXXXX.vtk` and apply the 'glyphvector' filter. In this case, change the glyph type from 'arrow' to 'line' and scale the colour of the lines by the x-component of the vector.

(c) The velocity field can be visualised using arrows as before with the 'glyphvector' filter.

(d) To visualise the colloids, open the files `col-cdsXXXXXXXX.csv`. This will open a table, which can be closed. Next apply the filter 'Table to Points' found in Filters/Alphabetical/Table to Points and set the X, Y and Z Columns in the 'Properties' tab below the Pipeline Browser to be the x, y and z-coordinates in the table via the drop down menus:

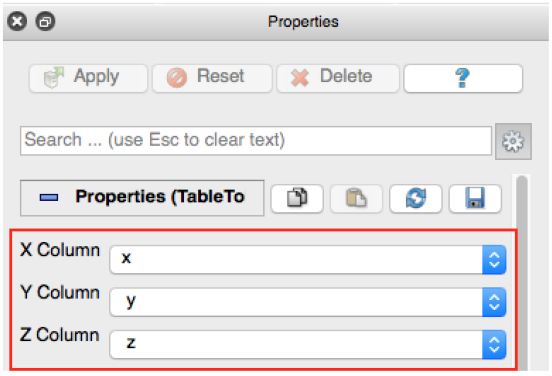

Figure 9: Setting the coordinates of the colloid locations in the properties tab.

You should now see points in the render window at the location of the centres of each of the colloids.

(e) To fully visualise the colloids, apply the 'glyphvector' filter and set glyph type to 'sphere.' In the properties tab, set the radius to 3.5, scale mode 'off' and scale factor to 1.0 in the properties. In order to access these settings, you will need to click the 'cog' icon shown here:

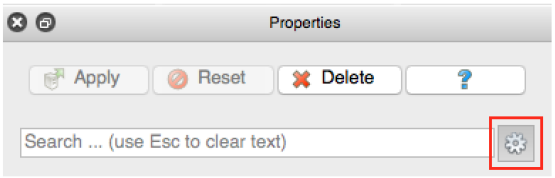

Figure 10: Location of the 'cog' icon in the properties tab.

(f) Next, glyph mode must be set to 'all points' and you should set the colour of teh spheres to 'Solid Color'. Finally, you can increase the resolution of the spheres by increasing the 'Theta Resolution' and 'Phi Resolution' quantities.

(g) In the pipeline browser, should now see:

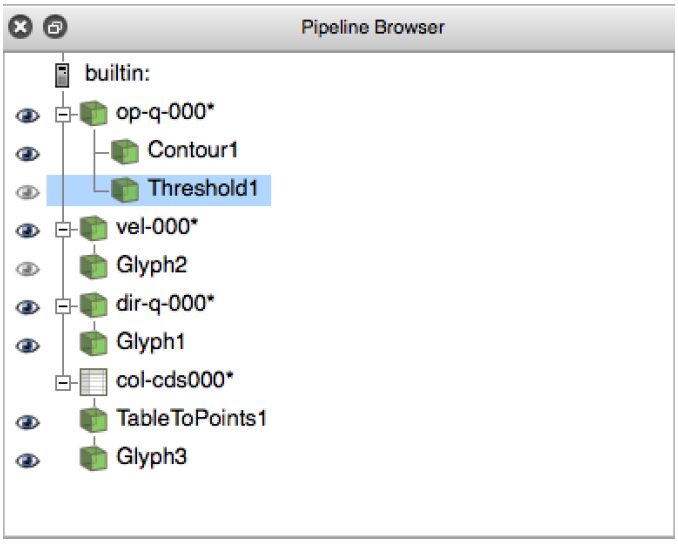

Figure 11: The pipeline browser for the colloidal suspension in liquid crystals.

The 3D visualisation of this system should appear as:

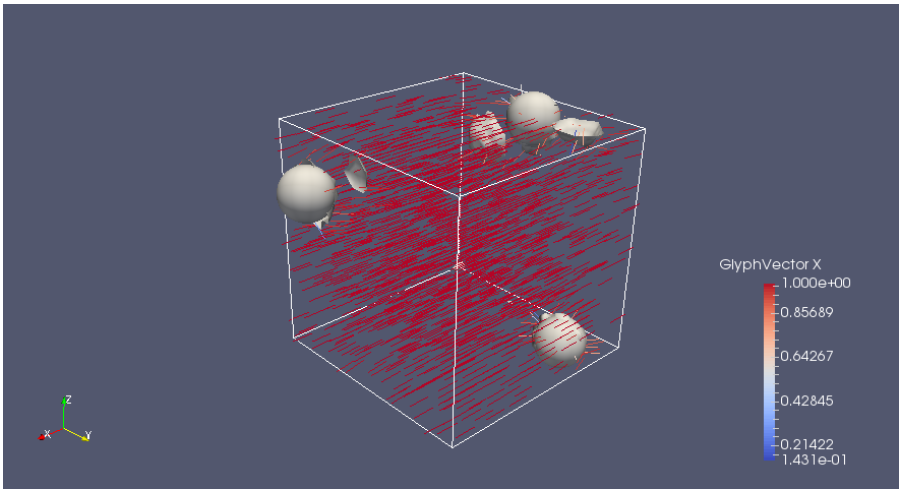

Figure 12: The visualisation of the colloidal suspension in liquid crystals.