DL_FIELD Training Workshop

Update version: April 2019

Tutorial

A thousand-mile journey begins with a single step - Lao Tzu (604-531 BC)

Remember to read the Manual.

It helps in the long run.

Getting started

Unpack your *dl_field_XX.tar.gz* by issuing the following commands (where XX is the version number):

gunzip dl_field_XX.tar.gz

tar -xvf dl_field_XX.tar

Now examine the DL_FIELD XX file structures. Amongst other files, you will notice the directory */tutorial*, which is only available to the Workshop attendees.

To compile the program type:

make

And DL_FIELD will be compiled automatically according to the instructions in the *Makefile*.

If you have previously compiled DL_FIELD XX in other machines, type
make clean before you type *make*.

How to run example files in tutorial

The *tutorial/* directory contains a collection of DL_FIELD's *control* files named according to the tutorial numbers.
Before running DL_FIELD for each tutorial, you must edit the *file dl_f_path* and insert the correct *control* filename. For instance, for Tutorial X, insert or change the following statement in the *dl_f_path*.

control = tutorial/tutorial_X.control

Where *X* is the tutorial number (1 to 9).

Alternatively, if you are confident, you can use the default *control* file (the *dl_field.control*, located in the DL_FIELD's home directory) for all tutorials by changing the appropriate options in the *control* file.

To run DL_FIELD, type:

./dl_field

If you compiled your own program. Otherwise, if you are provided with a Window version executable file, double click the application icon labelled
DL_F_win

An application console window will open while DL_FIELD is running and close itself once completed. Note: it may appear and quickly disappear, especially for small system samples.

Tutorial (1) - *benzene.pdb*

This tutorial will convert a simple benzene molecule using different potential schemes.

Use an editor and open *the dl_f_path* file and you will notice it contains the following statement:

```
control = tutorial/tutorial_1.control
```

Which indicates the location of the *control* file, along with all other file components.

If you inspect the *tutorial_1.control* file, you will notice the 'charmm22_prot' potential scheme is selected and the user's configuration is *benzene.pdb* located in the */tutorial* directory.

Now run the conversion by typing *./dl_field* (or clicking the DL_F_win icon for Window version executable file).

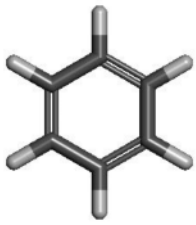
If the program is run successfully, you should see the following text:

```
Structures have been converted successfully.
Program executed successfully. Thank you for using DL_FIELD.
Time taken for conversion: some_value s
```

An output file *dl_field.output* is produced, along with corresponding DL_POLY files located in the */output* directory (*dl_poly.FIELD*, *dl_poly.CONFIG*).

Look at some of these files to get familiarise how DL_FIELD converts the structure. You may also wish to see the contents of *benzene.pdb* if you are not familiar with the PDB format.

Now, try other potential schemes (opls2005, pcff, dreiding, etc) and see what happens. If you see an error, it most probably means no benzene structure is defined for that particular potential scheme.



Benzene molecule

Tutorial (2)

This tutorial demonstrates the conversion of a fully solvated biological structure. It is a protein molecule called dihydrofolate reductase. The example structure was obtained from the AMBER website.

Go to */tutorial* directory and inspect *amber_test.pdb* file. It is a standard protein model in the PDB format. The model consists of the protein molecule fully solvated in explicit water molecules. The model consists of 23558 atoms in a cubic periodic box of size 62.23 Å.

You can use any molecular visual software (such as VMD) to look at the structure.

Edit *dl_f_path* and change the *control* file to *tutorial_2.control* and run DL_FIELD. Alternatively, use the default control file and do the following changes:

(1) Open *dl_field.control* file and select AMBER potential scheme.

(2) Configuration file:
tutorial/amber_test.pdb

(3) Apply the periodic boundary conditions as follows:

```
…
…
1        * Periodic condition ? 0=no, other number = type of box.
62.23  0.00  0.00     * Cell vector a (x,y,z)
0.00   62.23  0.00    * Cell vector b (x,y,z)
0.00   0.000  62.23   * Cell vector c (x,y,z)
…
```

(4) Save *dl_field.control* and run DL_FIELD to convert the structure. Remember to check *dl_f_path* if the correct *control* file is used.

(5) You can switch on the MM energy calculation option (line 38 of the *tutorial_2.control* file) and rerun DL_FIELD. Depending on your machine, you will notice it takes a while to do the energy calculations. Switch off (0) the MM energy option if this takes too long.

Now, repeat the conversion by using different potential schemes (CHARMM22_prot or CHARMM36_prot). This tutorial demonstrate the ease of using different potential schemes.

You can actually run this structure directly using DL_POLY_4. To do that you need to rename the DL_FIELD output files as follows:

dl_poly.CONFIG to *CONFIG*
dl_poly.FIELD to *FIELD*
dl_poly.CONTROL to *CONTROL* (or use your own CONTROL file)

Note: The system is charged due to the absence of counter ions and so the simulations are just for demonstration purposes.

dihydrofolate reductase in ribbon notation.

Try to run a short simulation to produce a HISTORY file. You can use DL_ANALYSER to analyse the result. Remember to insert the trajectory filename (such as the HISTORY file) into *dl_analyser.input* file.

Tutorial (2b)

This tutorial demonstrates the use of DL_FIELD to launch DL_POLY.

Before you proceed, please note that you can only do this tutorial if you compiled and can run DL_FIELD and DL_POLY in Linux systems, or cygwin (slow!) environment within the Window systems.

(1) Single-point calculation

By referring to the same protein system from **Tutorial 2**, this tutorial demonstrates the setting up of the force field model, follows by a single point calculation using DL_POLY.

Open and inspect the relevant control file called *tutorial_2b.control*. Locate the DL_POLY control section:

```
************* DL_POLY control ******************
1           * Run DL_POLY program
DLPOLY.Z    * DL_POLY executable filename
/home/user/dl_field_4.5/output * absolute path to DL_POLY program
1           * MM calculation 1=on  0=off
0   3       * Equilibration on(1)/off(0)  level (1,2 or 3)
1000        * Timestep for DL_POLY runs.
9.0         * cutoff (vdw and electrostatic)
1000        * Time limit for DL_POLY run (in seconds)
```

This is the section that instructs DL_FIELD how to run DL_POLY once the force field model conversion has been completed. In this case, a single point calculation (zero MD step) will be carried out.

Note that correct DL_POLY path must be given (highlighted in bold), which depends on your machine. By default, the filename for DL_POLY is DLPOLY.Z.

When DL_FIELD is run, a force field model will be setup as mentioned in **Tutorial (2)**. Then, DL_FIELD will automatically create a CONTROL file, the CONFIG (copied from *dl_poly.CONFIG*) and FIELD (copied from *dl_poly.FIELD*) files **in the directory where the DL_POLY program** is located. After that, DL_POLY program will run the calculation.

(2) Equilibration

If your initial system model is in an unstable high energy conformation, you can carry out a series of equilibration processes. DL_FIELD allows level 1, 2 or 3 equilibration processes. The number refer to the tiered DL_POLY runs each with an independent CONTROL file that contains the directives to eventually relax the system model. Level 1 is the least restricted process and run DL_POLY only once, Level 2 will run DL_POLY twice with two different CONTROL files, and so on.

Now edit the t*utorial_2b.control* file as follows (in bold):

```
************* DL_POLY control ******************
1           * Run DL_POLY program
DLPOLY.Z    * DL_POLY executable filename
/home/example/dl_field_4.5/output * absolute path to DL_POLY program
0           * MM calculation 1=on  0=off
1   2       * Equilibration on(1)/off(0)  level (1,2 or 3)
500        * Timestep for DL_POLY runs.
9.0         * cutoff (vdw and electrostatic)
1000        * Time limit for DL_POLY run (in seconds)
```

This instructs DL_FIELD to run equilibration twice, each with a different CONTROL file and start the simulation afresh from the previously equilibrated CONFIG file.

Now run DL_FIELD.

(Continue next page)

Tutorial (2b) continue…

Once the force field model is setup , DL_FIELD will then call DL_POLY. Depends on the computational power of your machine, the following message will eventually appear in sequence:

```
Setting up DL_POLY files for equilibration runs...
Run 2
Program is still running...please be patient.
…
…
Finish running.
A copy of OUTPUT file has been created: OUTPUT2
A copy of STATIS file has been created: STATIS2
A copy of CONTROL file has been created: CONTROL2
Run 1
…
…
Finish running.
A copy of OUTPUT file has been created: OUTPUT1
A copy of STATIS file has been created: STATIS1
A copy of CONTROL file has been created: CONTROL1
```

If DL_POLY runs are too slow, try to reduce the number of MD steps, say, from 500 to 100.

Compare CONTROL2 and CONTROL1 files. You will notice Run 2 (first DL_POLY run) contained DL_POLY directives such as *force capping* and *zero* to 'tame' and restrict the system movement, whereas, these directives were removed in Run 1 (second DL_POLY run), to allow more dynamic motions.

Note that the equilibration feature as mentioned here is by no means a substitution to the proper equilibration processes. This feature is included to enable user to carry out small MD runs in order to relax initial structures that maybe in high-energy conformation states. The final CONFIG file, which contained the relaxed structure, can then be transferred to a HPC system to carry out further equilibration processes.

Tutorial (3)

Sometimes it is essential to impose certain restriction on molecular models as part of the equilibration processes. This is frequently applied to bio-molecular systems such as proteins. For instance, tether restrains on C-alpha atoms.

By using the same biological structure from **Tutorial (2)**, this tutorial demonstrates the ease of setting the states of atoms in the model. This is done by changing the various options in the *control* file in lines 24-26 (freeze, tether and constrains). Use *tutorial_3.control* for this exercise.

(1) The model uses the TIP3P model (TP3O). You can change this to other water models such as SPC/E. Edit *amber_test.pdb* and substitute the residue name TP3O to SPCE.

Run the conversion using charmm22_prot potential scheme. You can try other water models as shown in *DLPOLY_CHARMM22_prot.sf* file (somewhere at line 243), by substituting the appropriate MOLECULE_KEY as shown above..

(2) Some water models such as SPCE do not assign the van-der-Waals parameter on the hydrogen atom. The water molecule structures must be preserved using some constrain methods. This advice is given in the ***MOLECULE*** water templates in the *.sf* file. Otherwise, the water structures will collapse, leading to MD run error in DL_POLY.

A special constrain command can be given to fix the structure of the water molecule. Select the option in *tutorial_3.control* that instructs DL_FIELD to read the constrain commands.

Issue the following directive command:

CONSTRAIN WAT ***rigid_water***

Remember to remove the remark '#' at the first column. Remember also to switch on the option to activate constrain bonds at line 24.

The 'WAT' is the Molecular Group name for water molecules contained in this model and ***rigid_water*** is the command to determine how to constrain the molecule in the Molecular Group. You can locate the Molecular Group WAT in the file *amber_test.pdb*.

Inspect the *dl_poly.FIELD* file somewhere at line 25675 onwards. You will notice water molecules are now constrained and, during simulation, the SHAKE algorithm will be used to preserve the water geometry.

(3) However, it is usually computationally more efficient if a whole water molecule is completely rigidified as a single entity. To do this, do the following changes:

(I) Include a remark to switch off the ***CONSTRAIN*** command
    # ***CONSTRAIN*** WAT ***rigid_water***
(II) Switch off (0) the constrain bond option at line 26.
(III) Switch on (1) the rigid body option at line 27.
(IV) Issue the directive command
    RIGID WAT
    Remember to remove the remark '#'.
(V) Run DL_FIELD.
(VI) Inspect the *dl_poly.FIELD* file at line 25675 and compare with **(2)**.

Note that the command ***h_bond*** should not be used for water models, which will only constrain the bonds containing hydrogen atoms. It is insufficient to hold the structure of the water molecules as the hydrogen atoms can still collapse onto each other (the angle H-O-H is not constrained).

Alternatively, instead of issuing the explicit the ***RIGID*** body command, you can solvate the model using the solvating feature at line 34 of the *control* file, which will automatically solvate the protein and implement the ***RIGID*** body feature on water molecules. To do this, you would need to remove all the water molecules in the PDB file.

To use DL_FIELD solvating feature, see **Tutorial (8)**.

Tutorial (3) continue…

(4) You can impose H bond constrains on the protein, and at the same time the constrains on the water model as follows

CONSTRAIN WAT ***rigid_water***
CONSTRAIN 5DFR ***h-bond***

Remember to switch off rigid body option at line 27 and switch on constrain bond option on line 26. Otherwise, you have bond constrains and at the same time assign rigid body on water molecules!

Run the DL_FIELD program and inspect the *dl_poly.FIELD* file somewhere at lines 12472 and 25675.

(5) You can also assign freeze or tether directives on the protein backbone molecule itself. To do that, just remove the '#' remark on the directive commands, remember to switch on/off the appropriate options on line 24 (freeze) or line 25 (tether).

You can leave the constrain commands intact.

The ***FREEZE*** and ***TETHER*** commands in the example control file instruct DL_FIELD to freeze or tether the C-alpha atoms of the protein molecule.

Try to issue various combinations of ***CONSTRAIN***/***FREEZE***/***TETHER***/***RIGID*** directives on the protein molecule itself. For more details on these directives, please refer to Chapter 4 of the User Manual.

Note:

(a) FREEZE and TETHER commands are commonly used for equilibration of protein molecules. Usually, this is to minimise the 'unnatural' movement of protein backbones while equilibrating the side chains and solvents. Eventually, the FREEZE and/or TETHER are removed to carry out a full equilibration before sampling.

(b) The H-bond constrain is usually applied in order to fixed the length of the bonds containing the hydrogen atoms. By freezing out the fastest vibrational motions, you can use a larger timestep, from say, 0.5 fs to 2 fs. However, this is at the expense of larger computational cost of carrying out SHAKE algorithm in DL_POLY.

Tutorial (4)

In this tutorial we will look at the *udff* features (Chapter 5 of the User Manual), using the benzene molecule from the previous tutorial (1) as an example. The *udff* file for this example is located in *tutorial/benzene.udff*. You need to instruct DL_FIELD to look for this file, at line 9 of the *tutorial_4.control* file. To disable the *udff* features, replace the filename with the word 'none'.

(1) By looking at the contents of the *udff* file, you will notice it contains the same command structures from the **.par* and **.sf* files. In other words, the *udff* file is the only file where DL_FIELD will be able to recognise the ***DIRECTIVES*** from both types of library files (*.par* and *.sf*).

The *udff* example given only works for CHARMM22_prot force field.

(2) To switch off the *udff* file, put *none* in line 9. Now switch on (by inserting the *udff* file location: *tutorial/benzene.udff*) the option and repeat the conversion. Inspect *dl_field.output* file in each case and note the difference.

You will notice DL_FIELD will produce a *dl_poly.FIELD* file according to the definition of the ***MOLECULE*** benzene template contained in the *udff* file. In this case, it is the addition of some improper interactions. The *dl_poly.FIELD* file has two additional entry to reflect these additions (lines 115-116 in the *dl_poly.FIELD* file). The *dl_field.output* file also reports the overriding of the ***MOLECULE*** benzene (line 105 in *dl_field.output* file), where the ***MOLECULE*** benzene template in the library file will not be considered.

(3) Now edit the *benzene.pdb* file and change the residue name (MOLECULE_KEY) from BENZ to BEND. Rerun the conversion. DL_FIELD now produces a deuterated molecule according to the molecular definition in the *udff* file.

In this case, DL_FIELD looks for the ***MOLECULE*** deuterated_benzene in the *udff* file, using the new ATOM_TYPE D_aromatic for the hydrogen atoms. The D_aromatic uses the same ATOM_KEY (HP) for HC_aromatic but with a different mass.

(4) The new ATOM_TYPE (new_H_aromatic) introduced in the *udff* file for the benzene ***MOLECULE*** seems redundant. In fact this is a powerful way to redefine an interaction while keeping other types of interaction the same.

Suppose the van-der-Waal (vdw) parameters for the hydrogen atom (ATOM_KEY HP) in the benzene molecule need some adjustment in order to fit your model requirements, **but** keeping all other types of interactions remain the same. One way to achieve this is to directly override the vdw parameters for HP by redefining the new parameters in the *udff* file. However, in doing so, you essentially apply the new vdw parameters to **all** other molecules that are using the same ATOM_KEY (HP) for the hydrogen atoms.

In order to apply the new parameters for the benzene MOLECULE **only**, you can define a completely new ATOM_TYPE with a new ATOM_KEY just for this purpose. This is done as follows:

Change the following command in the ***ATOM_TYPE*** directive in the *benzene.udff* file:

new_H_aromatic HP H 1.00797 New atom type, same as aromatic H (HC_benzene)

To

new_H_aromatic H_ben H 1.00797 New atom type, same as aromatic H (HC_benzene)

You have now created a completely new ATOM_KEY called *H_ben*. Insert the following new commands towards the end of the *udff* file (before the ***END POTENTIAL***).

EQUIVALENCE
H_ben > bond_HP angle_HP dihedral_HP imp_HP inv_HP
END EQUIVALENCE

Tutorial (4) continue…

And in the ***VDW*** directive, insert the following parameters for the new hydrogen atom.

H_ben -0.04 1.588 -0.01 1.22 My new vdw for new hydrogen

Now, run DL_FIELD to reconvert the benzene structure. Remember to change the residue name back to BENZ.

The ***EQUIVALENCE*** directive instructs DL_FIELD to treat H_ben the same as HP for all interactions. The only exception is the vdw type, which will take the vdw as defined in the ***VDW*** directive.

By also adding *vdw_HP* to the ***EQUIVALENCE*** statement it means H_ben is completely equivalent to HP. In other words, without the ***EQUIVALENCE*** statement, this will then be the same as

new_H_aromatic HP H 1.00797 New atom type, same as aromatic H (key HP)

Tutorial (5)

This exercise is to show how to construct a new MOLECULE, octanol. The structure is given in the *tutorial/* directory as *octanol.pdb*. However, the residue key of the molecule is missing.

(1) In order to convert this structure using CHARMM force field, the residue key OCOH must be inserted at column 18-21 of the file. This structure is already defined in *DLPOLY_CHARMM22_prot.sf*. Try to convert this structure using the charmm22_prot force field. For more information about the PDB format, please consult the User Manual (Chapter 6).

If you do not insert the residue key, you will get the following error:

```
Error: Can't find residue in Molecular Group m in config file
```

(2) Octanol has not been defined in some other potential schemes. However, you can define this structure using the general force field such as PCFF, OPLS2005 or CVFF in the *udff* file.

Now create and name a new *udff* file. The content of the *udff* file is shown roughly below:

```
POTENTIAL potential_scheme

UNIT kcal/mol

MOLECULE_TYPE
octanol    key    mw.       Remark
END MOLECULE_TYPE


MOLECULE octanol     num_of_atom  99.0
...
...
END MOLECULE

END POTENTIAL
```

Notice the charge value is defined with some arbitrary value of 99.0. When DL_FIELD encounter such 'ridiculous' value, it means DL_FIELD will determine the charge automatically based on the bond charge increment (BCI) method that applies to certain force fields like PCFF, CVFF and OPLS2005. This basically involves reading some pre-defined charge values from a library list for all neighbouring atoms that are connected to the atom of interest. The charge for each atom are then determined by DL_FIELD by summing up all the charge values.

To determine what ATOM_TYPEs to use, consult the appropriate *.sf* file. For instance, in the *DLPOLY_PCFF.sf* file, look for MOLECULE ethanol. This will give you some clues how the MOLECULE octanol can be constructed.

In fact, force fields such as CVFF, PCFF and OPLS2005 use the DL_F Notation and you will find it is not too hard to construct the molecule, if you are familiar with the chemistry of the molecule!

(3) Once you have created the *udff* file, run DL_FIELD to convert the octanol structure. Remember to change the force field scheme at line 2 and insert your new udff file location at line 9 of the *control* file.

Tutorial (6)

In this tutorial, you are going to assign rigid atoms to your molecule.

Look at *DLPOLY_CHARMM22_prot.sf* file and locate the ***MOLECULE*** benzene. Notice the [***RIGID***] statement towards the end of the ***MOLECULE*** definition.

Consult Chapter 2, part (5) *Optional directives* in the user manual to see how the [***RIGID***] directive is used in a ***MOLECULE*** definition. Consult Chapter 4, Option 24 to show how ***RIGID*** must also be applied in the *control* file as well.

(1) Switch on (1) the rigid body application on line 27 of *dl_field.control* or *tutorial_6.control*.

(2) Insert the following statement between the two hash lines (##########…) in the *dl_field.control* file.

RIGID not_define

Note that because the benzene structure in *benzene.pdb* does not define a Molecular Group, DL_FIELD will automatically assign the molecular system belongs to the Molecular Group 'not_define'. The above statement therefore means 'apply rigid body to the Molecular Group called 'not_define'. **If there are other Molecular Groups, no rigid body will be applied** unless they are also explicitly specified.

In addition, the way how the rigid body is applied to a molecule is shown in the ***MOLECULE*** template.

(3) Convert the *benzene.pdb* structure and see what happens. Make sure the MOLECULE_KEY is BENZ. Notice that the number of bonds and other number of bonded interactions were reduced when the rigid body feature is applied.

Tutorial (7)

In this tutorial, you are going to see the use of the auto-CONNECT feature. Please read Section 2.3.1 of the DL_FIELD manual before getting on this tutorial.

This tutorial takes the files from the *Examples/* directory and use the charmm22_prot scheme.

(1) Take a look at the *alcohol.udff* file located in the */Examples* directory. In this *udff* file, you can see the MOLECULE ethanol contains the auto-CONNECT features. The MOLECULE ethanol is in fact the identical MOLECULE defined in the standard library file (*DLPOLY_CHARMM22_prot.sf*), but is now overridden by that of in the *udff* file.

With the auto-CONNECT feature, DL_FIELD does not do the usual template matching, as would be the case for the MOLECULEs with the normal CONNECT feature. With the auto-CONNECT feature, DL_FIELD will identify all bonds automatically and assign all force field parameters accordingly.

(2) Examine *ethanol_auto1.pdb* and *ethanol_auto2.pdb*. Both files essentially contains two identical structures. The only difference is the way the ethanol molecules are defined. In the former structure, the two ethanol molecules are clearly defined with two separate residue id, ETOH 1 and ETOH 2. In the latter case, the molecules are lumped up as a single entity and deliberately inter-mixed the atomic constituents from each other molecules.

(3) Examine also the file *ethanol.pdb,* which is used for the original MOLECULE ethanol with the normal CONNECT feature. Note that the atom labels in the PDB files as mentioned in **(2)** have to be the force field specific ATOM_KEYs for MOLECULEs defined with the auto-CONNECT feature. Whereas, the atom labels can be generic for the *ethanol.pdb* structure.

(4) Now convert the structures mentioned at **(2)**, with the *udff* option switch ON and compare with the *ethanol.pdb*, with the *udff* option switch to OFF. To switch the *udff* option to ON, insert the filename *Examples/alcohol.udff*. To switch it OFF, insert *none*.

You will notice that *ethanol_auto1.pdb* can be successfully converted with or without switching on the *udff* file. The only difference is that *dl_field.output* will report whether the template is of auto-CONNECT type. However, the *ethanol_auto2.pdb* would need the *udff* file.

(5) The alcohol.*udff* file also contains the MOLECULE aliphatic_alcohol. As the name implies, this MOLECULE can be used to convert ***any*** aliphatic alcohols. Try to convert the *alcohols.pdb* located in the *Examples/* directory and see how it works. The PDB file contains a mixture of both ethanol and 2-propanol molecules. Remember to use the *alcohol.udff* file.

(6) The MOLECULE aliphatic_alcohol can also be used to convert the *octanol.pdb* obtained from the previous **Tutorial (5)**. Edit the PDB file and change the residue name to ROH (which is the MOLECULE_KEY for aliphatic_alcohol. See *alcohol.udff*) and change all the atom labels to the correct ATOM_KEYs in the *octanol.pdb* file.

(7) Convert the resulting structure and see what happens. Remember to use the *alcohol.udff* file.

Tutorial (8)

In this tutorial, you are going to see how solvents are added. We are going to solvate a protein molecule. This structure is located in *6pti.pdb* in the *Examples/* directory. It is a bovine pancreatic trypsin inhibitor.

Now, edit the *dl_field.control* (or *tutorial_8.control*) file:

(1) Use CHARMM22_prot as the force field scheme.

(2) Specify periodic boundary condition. We are going to define a cubic box of size 40 Å. At line 28, select the value 1 to indicate a cubic shape periodic box. Then, specify the cell vectors as follows:

```
40.0   0.0   0.0
0.0    40.0  0.0
0.0    0.0   40.0
```

(3) At line 35, we want to position the solute (which is the 6pti protein) at the center of the box before it is solvated. Make sure this value is 1 (ON).

(4) Line 36 specifies the type of solvent use. It takes two values: the solvent type and minimum distance of the solvent molecules from solute. Choose the following values:

```
TIP3P_C    1.9
```

This instructs DL_FIELD to solvate the model with a TIP3P water model. The value 1.9 is the minimum distance, in Angstrom, that the solvent can locate from the solute. The default 'none' means the protein will not be solvated.

(5) Run DL_FIELD. Examine *dl_field.output* file. Towards the end of the output file DL_FIELD reports the solvating status. You will notice that DL_FIELD gives a warning due to a net charge of +6.0 in the structure.

(6) A common way to neutralise the system is to insert counter ions. To do this, edit *control* file and look for line 37. Select this option to 1 to instruct DL_FIELD to add counter ions. Note that DL_FIELD will automatically detect if this is needed, the type and number of counter ions depending on the net charge of the system. Set the second parameter to 10.0. This is the minimum distance, in Angstrom, the counter ions should be located from the solute.

(7) Run DL_FIELD. Now examine the *dl_field.output* file and the *dl_poly.FIELD* file. You will notice six chloride ions have been introduced to neutralise the system.

(8) Try to use other solvents. You can see the complete list of solvents available in the file *solvent_list* located in the *solvent/* directory.

Suggested solvents: TIP4P, EtOH, IPA, etc.

Note: If you get an error, it either means you mistype the solvent labels or the solvent template is not available for the particular force field scheme chosen.

Note also that the solvated system must be properly equilibrated before collecting the sample runs in DL_POLY. For protein systems, this means constrains (like tether) must be applied, say on the C-alpha, to make sure the structure is not perturbed while equilibrating the solvent in the system (see **Tutorial (3)**).

Tutorial (9)

In this tutorial, you will learn how to set up force field models using *xyz* as the input format. By using this format, DL_FIELD will carry out full automatic assignment of atom types, bypassing the need to construct MOLECULE templates. This tutorial also demonstrates the universality of atom-typing using the DL_F Notation, independent of the force field schemes used.

Before proceeds, please refer to Section 6.2 of the Manual.

(1) Go to *utility/* directory and compile the program *pdb_to_xyz.c* (read *readme.txt* file if you are stuck).

(2) Run the *pdb_to_xyz* program to convert *octanol.pdb* to *octanol.xyz* file.

For input file, type this: `../tutorial/octanol.pdb`

For output file, type this: `../tutorial/octanol.xyz`

This creates the xyz file in the tutorial directory.

(3) Go to the *tutorial/* directory and edit *octanol.xyz*. Remove all the numeric labels. For instance, C12 becomes C, H42 becomes H, etc. Essentially, the labels now show the actual element symbols of the atoms, which is the requirement for DL_FIELD to convert the *xyz* structures.

(4) Edit *dl_field.control* (or use *tutorial_9.control*) and change the force field scheme to OPLS2005 and use *octanol.xyz* as the input file.

(5) Run DL_FIELD program.

Upon successful run, open *dlf_notation.output* file in the *output/* directory. The file contains atomic structure information, showing the actual chemical nature of every atom using the DL_F_ Notation.

```
# List of ATOM_TYPEs in DL_F Notation
Atom_index   ATOM_TYPE   neighbour_number   neighbour1  neighbour2 ...
MOLECULAR_GROUP XYZ  POTENTIAL opls2005
1     Cs_alkane                      4     2     3     4     6
2     HC_alkane                      1     1
3     HC_alkane                      1     1
4     O_alcohol                      2     1     5
5     HO_alcohol                     1     4
6     Cs_alkane                      4     1     7     8     9
7     HC_alkane                      1     6
8     HC_alkane                      1     6
9     Cs_alkane                      4     6    10    11    12
10    HC_alkane                      1     9
11    HC_alkane                      1     9
12    Cs_alkane                      4     9    13    14    15
13    HC_alkane                      1    12
14    HC_alkane                      1    12
15    Cs_alkane                      4    12    16    17    18
16    HC_alkane                      1    15
17    HC_alkane                      1    15
18    Cs_alkane                      4    15    19    20    21
19    HC_alkane                      1    18
20    HC_alkane                      1    18
21    Cs_alkane                      4    18    22    23    24
22    HC_alkane                      1    21
23    HC_alkane                      1    21
24    Cp_alkane                      4    21    25    26    27
25    HC_alkane                      1    24
26    HC_alkane                      1    24
27    HC_alkane                      1    24
```

(6) Go back to step **(3)** and use PCFF or CVFF as the force field scheme.

You will notice basically PCFF and CVFF display the same ATOM_TYPEs as those of OPLS2005. However, the *dl_poly.FIELD* will show different force field parameters.

Tutorial (9) continue…

(7) Now go back to *control* file at line 15 and change the atom format display from DL_FIELD format (1) to the standard force field format (2) and *vice versa*. Rerun DL_FIELD and examine the converted files (such as *dl_poly.FIELD*) in the *output/* directory.

You will notice when using the DL_FIELD format (DL_F Notation), the ATOM_KEYs are identical for different force fields but with different parameters according to the chosen force field schemes. This demonstrates the universal format of the DL_F Notation.

Note:

In some instance, it would be useful to use DL_F Notation as atom labels for DL_POLY runs. This is because certain features in DL_ANALYSER require the use of the Notation. For instance, in detecting and quantifying various modes of atomic interactions.

Tutorial (10)

There are some other example structures contain in the *tutorial/* directory. Convert these structures using a potential scheme of your choice and see what happens. You may also want to convert some of these PDB files into the *xyz* files.

More example structures are given in the */Examples* directory, which are given out to all registered users. Consult Chapter 14 of the user manual for a brief description for each structure. The *control* files for these example structures can be found in the *control_files/* directory.

—-- End of Tutorial —-
C W Yong, April 2019