FFEA Workshop part I: Running and Visualising FFEA

Rob Welch

July 10, 2017

1 Introduction

This tutorial will show you how to run your first FFEA simulation, using input files that we made earlier (later tutorials will delve more deeply into creating and parameterising FFEA models). We will conclude this tutorial by using the FFEA PyMOL plugin to visualize the results.

This tutorial is running in a customized Xubuntu live USB environment. For those not familiar with Xubuntu,

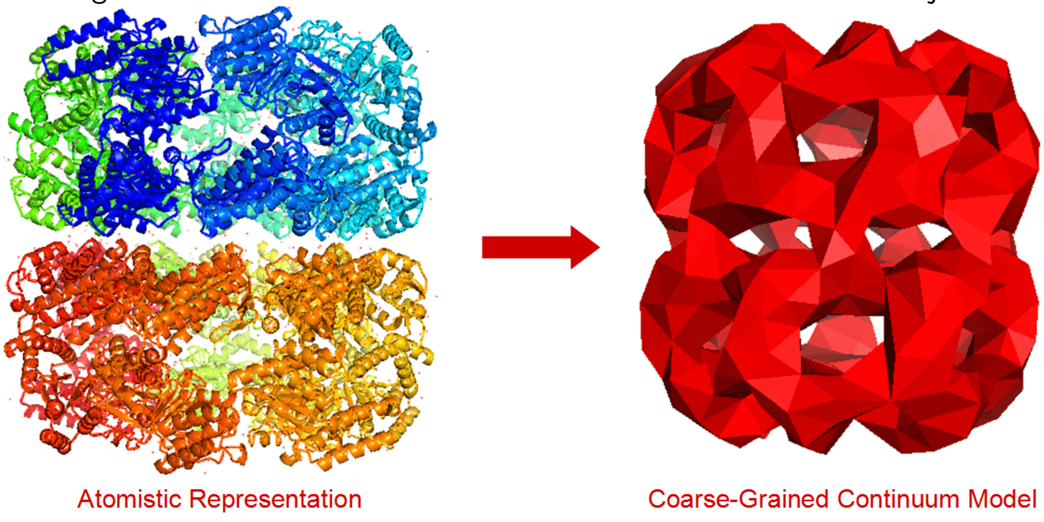
- The application launcher is the mouse-shaped icon in the top-left of the screen.
- The default text editor is called Mousepad (although you are welcome to install your own). You can find it in the application launcher, under Accessories › Mousepad)
- The default file manager is Thunar, although it is identified simply as 'File Manager' in the launcher. If you right-click somewhere in the current folder inside Thunar, you can open a terminal in the current folder.

Note: This workshop is by no means an exhaustive look at FFEA's features. If you want to know more, make sure to visit our official documentation: `http://ffea.readthedocs.io`.

2 The FFEA input file

FFEA objects are made from volumetric data. The volumetric data can be direct from Cryo-EM, or created from atomistic data. In the FFEA model creation process, the volumetric data is cleaned up, and then it is coarse-grained - the level of detail in the 3D mesh is reduced. Then, we fill the inside of the mesh with tetrahedrons.

Figure 1: Conversion of atomistic structure for GroEL to FFEA object



Finally, we create the FFEA input files needed to run our simulation (you will learn more about this in the next tutorial). The input files are:

- The .ffea script file - this is the top level file which contains global simulation parameters, and contains an overview of all of the objects in the system, with links to the other files that describe them.
- The .node file - contains the position of each node in the mesh.
- The .top file - describes how the nodes are connected together to form a tetrahedral mesh.
- The .mat file - contains the material parameters (for example, density and elastic moduli) of each tetrahedron.
- The .surf file - the same as the .top file, but it describes only the surface nodes.
- The .stokes file - contains the effective hydrodynamic radius of each vertex in the mesh.

- The .vdw file - contains the Van der Waals interaction 'type' for each exterior face. The interaction between every vdw type and every other vdw type can be set in the .lj file.
- The .pin file - contains a list of all restrained nodes, which do not move during the simulation.
- The .lj file - describes the interaction between every vdw type and every other vdw type.

Step 1 You should be able to find these input files by opening the file manager and navigating to home » ffea » Workshop » 1-Intro-to-FFEA » Simulation_files. Verify that you have them, and open them up in a text editor, if you want. These input files are created mostly as 'blank slates' with a set of sensible defaults. Once they've been created, they can be modified, either using our python package (FFEA tools), graphically in the FFEA viewer, or by editing the text files manually.

> **Note:** You can find a more detailed description of each if these files in our documentation: http://ffea.readthedocs.io/en/latest/ioFiles.html.

3 Running your first simulation

For your first simulation, we have provided you with pre-made ffea input files for Cytoplasmic Dynein. Dynein is a motor protein that moves processively along a microtubule. For this initial simulation, we will simulate a single dynein molecule floating in solvent.

Step 2 Take a look inside your .ffea file. It should look like this:

```
<param>
        <restart = 0>

        <dt = 1e-13>
        <check = 100>
        <num_steps = 10000>

        <trajectory_out_fname = dynein_out.ftj>
        <measurement_out_fname = dynein_out.fm>
        <checkpoint_in = dynein_in.fcp>
        <checkpoint_out = dynein_out.fcp>
        <vdw_forcefield_params = dynein.lj>

        <calc_stokes = 1>


        <calc_vdw = 1>
        <inc_self_vdw = 0>
        <vdw_type = ljsteric>

        <calc_noise = 1>

        <num_blobs = 1>
</param>

<system>
        <blob>
                <motion_state = DYNAMIC>
                <topology = dynein.top>
                <material = dynein.mat>
                <stokes = dynein.stokes>
                <nodes = dynein.node>
                <surface = dynein.surf>
                <vdw = dynein.vdw>
                <scale = 1.00e-10>
        </blob>
</system>
```

Some important features of the .ffea file:

- The global simulation parameters block:
  - dt - the length of a timestep.
  - kT - a way of defining the temperature $(k_B \cdot T)$.
  - check - how often to save a frame. For example, check $=$ 100 will save a frame after every 100 steps.
  - num_steps - the number of steps to run before the simulation terminates. For example, num_frames will run a 10,000 step simulation, saving a frame every 100 steps, for a total of $1ns$ spread across 100 frames.
  - output filenames - the files that contain the trajectory and analysis data.
  - calc_stokes, calc_vdw, calc_noise - turn on and off the stokes viscosity, Van der Waals interactions and thermal noise.

- vdw_type - how FFEA handles short range forces (more info here: http://ffea.readthedocs.io/en/latest/shortRange.html)

- The structure of the system
  - A list of all the blobs in the system.
  - A blob is an FFEA object. Systems can be comprised of many objects, but the simplest just have one.
  - Each blob has a set of files, described in the section above, which are linked together in this file.
  - Each blob has a motion state, which determines whether it moves or is fixed, a scale, and (optionally) a translation and rotation for its initial state.

Step 3 In order to run your simulation, open up a terminal (the easiest way is to right-click somewhere in the folder containing your .ffea file and click 'Open Terminal Here'), and enter the command

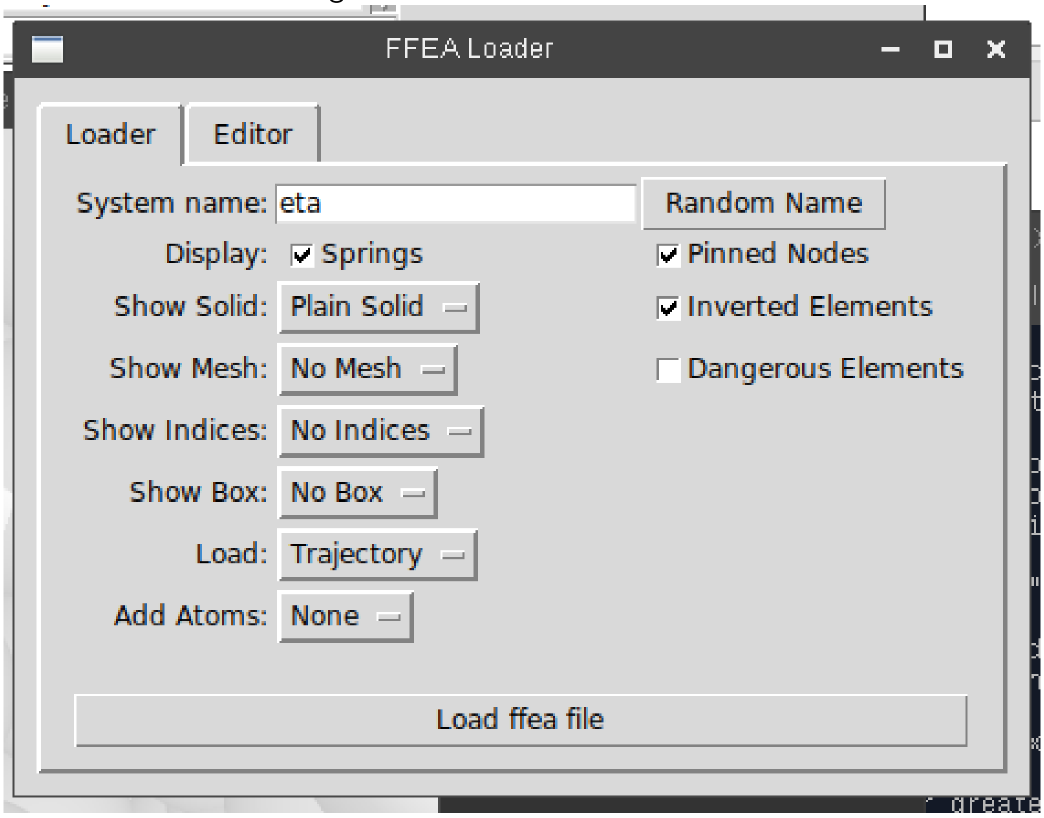
```
ffea dynein.ffea
```

into the terminal. FFEA will initialise and start printing information to the screen. When it's finished, you should notice that there is a dynein_out.ftj file in the folder. This is the FFEA trajectory file, and it contains the positions of the nodes at each frame of your simulation.

4 The FFEA Viewer

FFEA comes with a viewer plugin for PyMOL. In order to load an FFEA trajectory, open up PyMOL, and select Plugin › FFEA Loader from the menu.

Figure 2: FFEA viewer loader tab



System name: eta

Display: ✓ Springs ✓ Pinned Nodes

Show Solid: Plain Solid ✓ Inverted Elements

Show Mesh: No Mesh ☐ Dangerous Elements

Show Indices: No Indices

Show Box: No Box

Load: Trajectory

Add Atoms: None

4.1 Load the Trajectory

The FFEA viewer contains a number of options which can be selected before the trajectory is loaded:

- Show solid: colours the faces of the mesh depending on how it has been parameterised. For example, setting it to 'vdw' will cause the different types of vdw interactions to show up in different colours.
- Show mesh: shows the edges of the mesh as a wireframe. This can be useful for seeing the inside of the structure.
- Show indices: display the number of each node in the viewer as a 3D model (note: this is a legacy option - there are other ways of displaying node indices that you will probably find easier to read).
  - 'Node indices - linear' shows only linear nodes, which are the kind of nodes we're interested in.

  - 'Show indices - second-order' also shows second-order nodes.
  - 'Element indices' will display the index of each element (the tetrahedra).
  - 'Faces' will display the index of each face (no tetrahedra - only the triangles on the surface).

- Show box: whether or not to show the edges of the simulation box.
- Load:
  - Trajectory - loads the entire trajectory, if one exists.
  - System - loads only the initial state of the system. If you have a trajectory but only want to edit the parameters, this will load much faster.
  - CGO - loads the system as one large CGO (compiled graphics object). This can be faster in some cases, particularly if you have a very large number of blobs, but is not useful here.
- Add atoms:
  - PyMOL cannot perform analysis on the 3D objects drawn by FFEA into the viewer. In order to get around this, we create a set of fake 'atom' objects at the positions of then nodes, numbered according to their index. As with 'show indices' (see above), we can place these atoms onto nodes, linear nodes, faces and elements.
  - Make sure to try this feature, because it will become important later! Adding atoms will enable another set of options in the viewer (under the 'Editor' tab) that you can use to parameterise a model - for example, setting vdw interactions and pinning nodes.

Step 4 Try loading a few trajectories with different combinations of these settings.

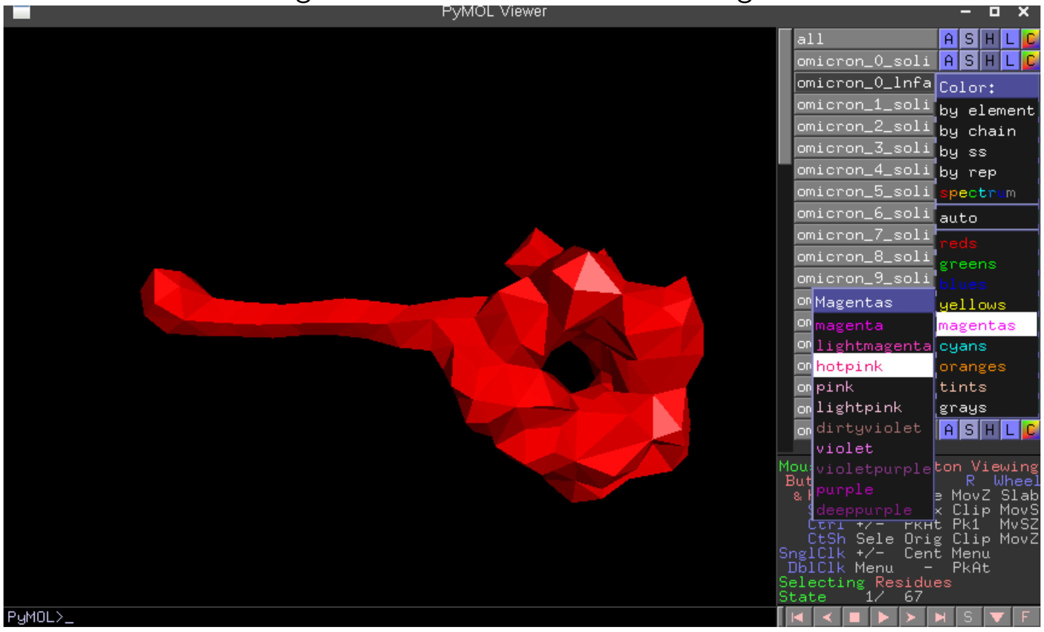
> **Note:** When a new trajectory is loaded. the viewer will not remove the old one by default. In order to remove the current trajectory, go to File > Reinitialize > Everything in the PyMOL main menu.

4.2 Atom Measurements

If you have loaded a trajectory with 'add atoms' turned off, clear it now, and load it with 'Add Atoms' set to 'onto linear nodes'. The right-hand bar on PyMOL's

main interface will display a list of all of the objects that are currently loaded in.
Step 5 To toggle the visibility of one of these objects, click on its name. To change the colour, click the 'c' icon on the right hand side.

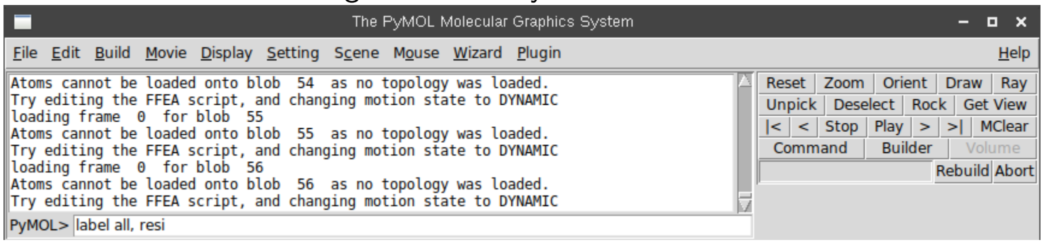
Figure 3: FFEA viewer colour settings.



The solid, wireframe and atoms will all be listed as different objects by the viewer. For example, the atoms object is named <systemname>_lnfa. To display
Step 6 the indices of the nodes that you enabled via the 'add atoms' setting, go to the PyMOL console and enter the line

```
label all, resi
```

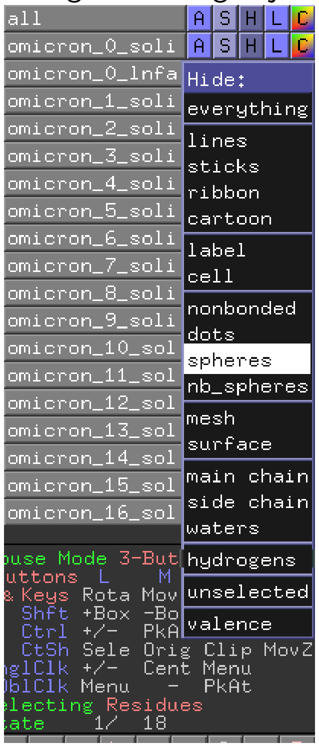
Figure 4: The PyMOL console.


The PyMOL Molecular Graphics System
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Reset Zoom Orient Draw Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Command Builder Volume
Rebuild Abort

```
Atoms cannot be loaded onto blob  54  as no topology was loaded.
Try editing the FFEA script, and changing motion state to DYNAMIC
loading frame  0  for blob  55
Atoms cannot be loaded onto blob  55  as no topology was loaded.
Try editing the FFEA script, and changing motion state to DYNAMIC
loading frame  0  for blob  56
Atoms cannot be loaded onto blob  56  as no topology was loaded.
Try editing the FFEA script, and changing motion state to DYNAMIC
PyMOL> label all, resi
```

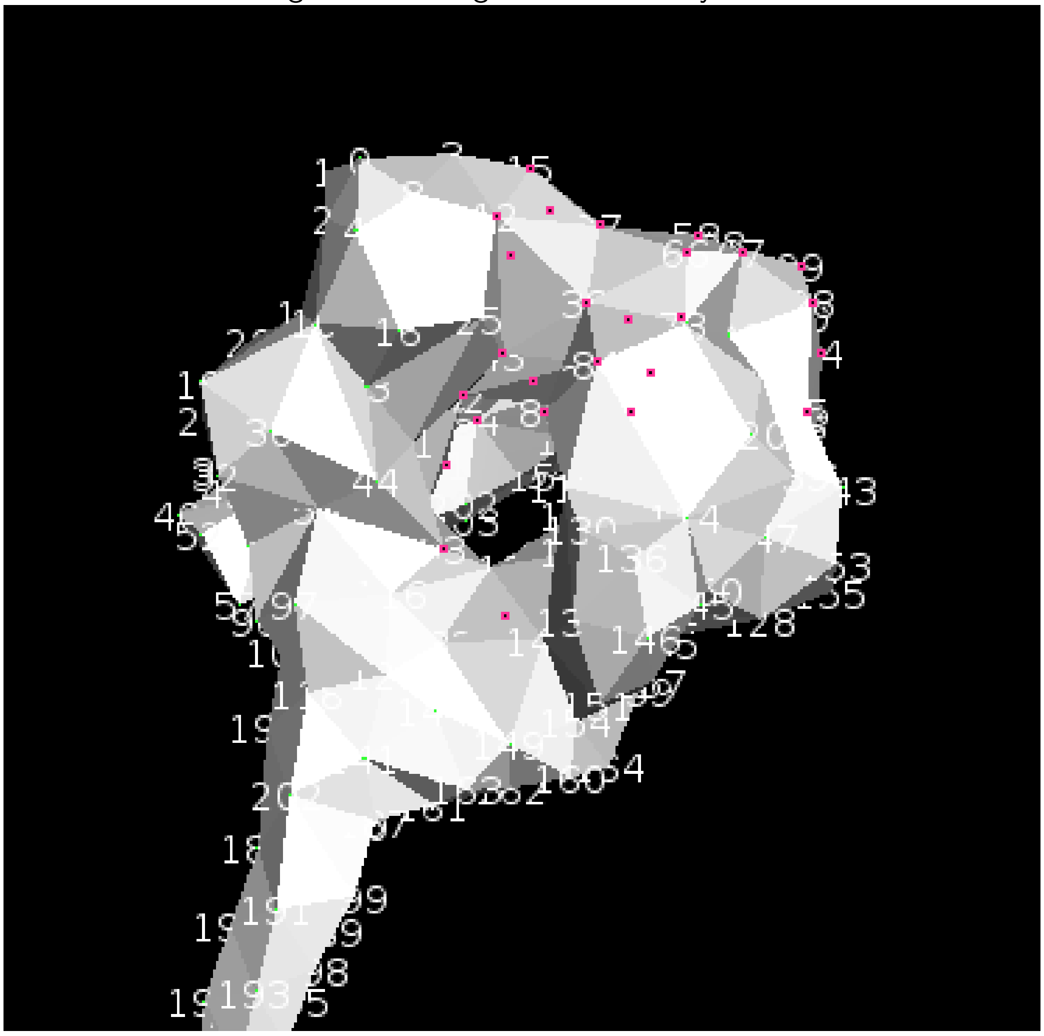
Step 7 If you can't read the labels, click the 'h' in menu for your _lnfa object, and select 'spheres'. This will hide the spheres from view, but keep the labels. You can re-enable them in the corresponding 's' menu, by clicking the 's' to the left of the 'h', and hitting 'spheres'.

Figure 5: Showing and hiding objects in PyMOL



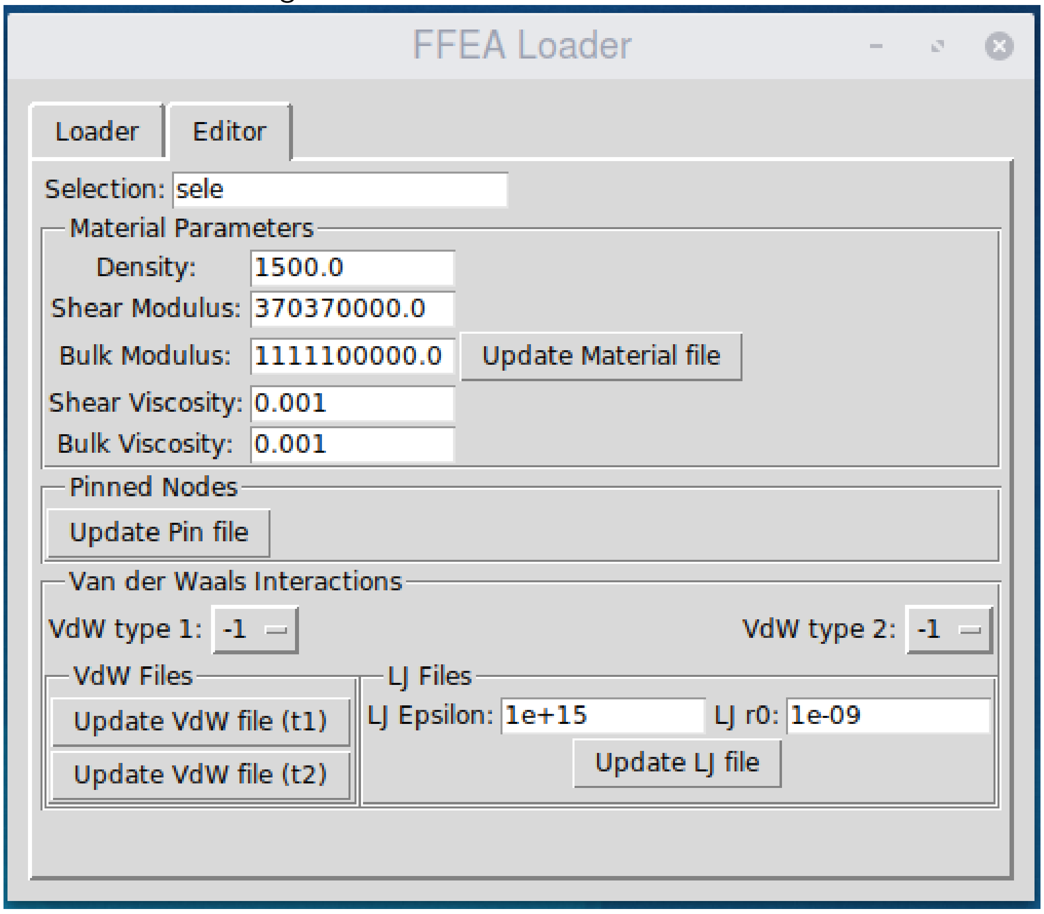
Step 8 You can also select a group of 'atoms' by holding shift and dragging the mouse across an area of the PyMOL window. To delete your selection, click the 'A' button next to the object named 'sele' on the right-hand menu, and click 'delete selection'. Try making a new selection by holding shift and clicking on individual 'atoms'.

Figure 6: Making a selection in PyMOL



Note: Later in the workshop, we will use these selections in order to fine-tune our model parameters. The 'editor' tab in the FFEA loader contains a set of utilities for modifying the parameters from selections.

Figure 7: FFEA Loader Parameter Editor



Selection: sele

Material Parameters

Density: 1500.0

Shear Modulus: 370370000.0

Bulk Modulus: 1111100000.0

Shear Viscosity: 0.001

Bulk Viscosity: 0.001

Pinned Nodes

Van der Waals Interactions

VdW type 1: -1

VdW type 2: -1

VdW Files

LJ Files

LJ Epsilon: 1e+15

LJ r0: 1e-09

- Selection: by default, this is 'sele', which is the default name of the user's current selection.
- Material parameters - the density, shear modulus, bulk modulus, shear viscosity and bulk viscosity (all given in SI units) determine the mechanical properties of the object.
- Pinned nodes - pinning nodes will fix them to their initial position in the simulation.
- Van der Waals interactions - each face can be set to one of 8 different vdw types (where -1 has vdw interactions turned off). Editing LJ epsilon and LJ r0 will set the parameters of the Lennard-Jones interactions between the two types set above.

Finally, creating these atoms allows you to use the native PyMOL tools to analyse your trajectories, found in the 'Wizard' setting of the PyMOL menu. For example, to measure end-to-end distance, select Wizard 〉 Measurement. The wizard will then prompt you to select two atoms and create an end-to-end distance measurement. When finished, hit the 'done' button that appears on the right-hand sidebar.

This is the end of the first part of the workshop! If you have finished early, why not try exporting your trajectory to an atomistic format?

5 An Aside: Exporting to PDB

If PyMOL is not your analysis tool of choice, you can convert your FFEA trajectory into a 'pseudo-atomistic' one, in which there is an atom at the position of each FFEA node. This allows you to perform the same measurements on FFEA files as you can on atomistic ones - RMSD, for example.

Note: FFEA has two built in 'FFEA to atoms' features. The one described here maps FFEA nodes directly onto 'dummy' atoms, which is good enough for most analysis. FFEA can also fit atomistic structures back inside FFEA trajectories. This can be handy for situations in which dynamics at multiple scales are required - the user can run an FFEA simulation, stop it, convert to atomistic, minimize it, and continue running it. To learn more, visit http://ffea.readthedocs.io/en/stable/FFEAPDBMapper.html.

In order to use this feature, close pymol and return to your terminal. Then, enter the command

```
ffeatools
```

FFEAtools is a python library and API designed to support the FFEA simulation package. You can use FFEA tools to create new FFEA models from atomistic and volumetric structures, and also to analyze FFEA trajectories. To get help on an FFEA tool, use the –help parameter. For example, we are interested in the

Step 9 'makepseudopdb' tool, so we type

```
ffeatools makepseudopdb --help
```

Step 10 The makepseudopdb tool tells us that the syntax is as follows

```
ffeatools makepseudopdb [FFEA trajectory (.ftj)] -o
    [Output fname]
```

We supply our trajectory file, create an output file, and specify that we want our

Step 11 output file in pdb

```
ffeatools makepseudopdb <dynein_name>.ftj -o <
   dynein_name_psuedo>.pdb
```

The resulting pdb (or mdcrd) can be used like any other atomistic trajectory. FFEA's built-in PCA tools use this to interface with PyPCAZip (bitbucket.org/ramonbsc/pypcazip).