1CPN-model Documentation

Release 1.0

Joshua Lequieu

Mar 31, 2021

Contents

1 Quick Start **3**
- 1.1 Getting 1CPN and compiling with LAMMPS . . . 3
- 1.2 Running your first simulation . . . 4
- 1.3 Analyzing the Simulation . . . 5

2 1CPN Implementation in Lammps **7**
- 2.1 Linking 1CPN with LAMMPS . . . 7
- 2.2 1CPN Performance across many processors . . . 7
- 2.3 Custom 1CPN Potentials . . . 7
- 2.4 Maintaing Compatability with LAMMPS . . . 10

3 Initialization **13**

4 Vizualization of 1CPN **15**
- 4.1 Option 1: Basic VMD . . . 15
- 4.2 Option 2. Enhanced VMD (recommended) . . . 16
- 4.3 Option 3. Blender . . . 17

5 Analysis of 1CPN Simulations **21**

6 Corrections of the 1CPN Paper **23**
- 6.1 Updates on Fig. 6 of 1CPN Paper . . . 23
- 6.2 $\varepsilon_{0,H4cut}$ in TABLE. S3 of 1CPN paper . . . 24

1CPN Website: https://lequieu.github.io/1cpn-model/

1CPN GitHub: https://github.com/lequieu/1cpn-model

This is the documentation for the 1CPN model implementation in LAMMPS. If you haven't read the 1CPN paper yet, please read it before reading this documentation:

Lequieu, Cordoba, Moller, de Pablo "1CPN: A coarse-grained multi-scale model of chromatin" (2019) J. Chem. Phys. 150, 215102

If you're just looking to install and compile 1CPN with LAMMPS, begin with the "quickstart" guide.

If you have any issues with 1CPN, please let us know on the our GitHub issues page page. Please don't hesitate to reach out, we're grateful for your interest in 1CPN and would love to help you get up and running.

Also, we welcome pull requests and so if you'd like to contribute to any part of the 1CPN project please let us know.

CHAPTER 1

Quick Start

This a quick start guide to getting going quickly with the 1CPN model.

1.1 Getting 1CPN and compiling with LAMMPS

1. Install git

```
sudo apt-get install git
```

2. Navigate to a directory where you would like to install the 1CPN Model.

```
mkdir 1cpn
cd 1cpn
echo "export D_1CPN=`pwd`" >> ~/.bashrc
source ~/.bashrc
```

3. Download a fresh copy of LAMMPS and checkout the *stable_16Mar2018* tag. Even if you already have a copy of LAMMPS, its recommended that you download a new copy specifically for 1CPN.

```
git clone -b stable https://github.com/lammps/lammps.git lammps-1cpn
git checkout stable_16Mar2018
```

4. Clone the 1CPN model

```
git clone https://github.com/lequieu/1cpn-model.git
```

5. Link 1CPN with LAMMPS

```
cd $D_1CPN/1cpn-model/src/lammps
make link
```

The Makefile assumes that the LAMMPS src code is located at ${D_1CPN}/lammps-1cpn. If you have LAMMPS located at another location, you can specify by redefining the `LAMMPS_SRC` variable.

```
make link LAMMPS_SRC=<path to LAMMPS src>
```

6. Build LAMMPS with the *ASPHERE* and *MOLECULE* package.

```
cd $D_1CPN/lammps-1cpn/src
make yes-ASPHERE
make yes-MOLECULE
make serial
```

1.2 Running your first simulation

Now that LAMMPS has been linked and compuled with the 1CPN-model, we're ready to setup and analyze our first 1CPN simulation. First lets make a new directory called *example* where the input and output files from this simulation will be stored. After the directory is made, navigate into it.

```
cd ${D_1CPN}/example
mkdir -p ${D_1CPN}/example
```

1.2.1 Generating 1CPN Input Files

1. The next step is to generate the initial configuration of the 1CPN-model. This is achieved using the *${D_1CPN}/init/init_1cpn.py* script. To start out, we'll generate a section of chromatin consisiting of 20 nucleosomes, each with a nucleosome repeat length (NRL) of 187 base pairs. From the *${D_1CPN}/example* directory, issue the following command:

```
${D_1CPN}/1cpn-model/init/init_1cpn.py -n 20 -nrl 187
```

The full list of arguements that can be passed to `init_1cpn.py` see *Initialization* or issue `init_1cpn.py` with the `-h` flag:

```
${D_1CPN}/1cpn-model/init/init_1cpn.py -h
```

1.2.2 Running a Simulation

2. Next we need to copy the LAMMPS input files necessary for setting up and running a simulation. These files are located at *${D_1CPN}/inputs*. Copy them to our *example* directory.

```
cp ${D_1CPN}/1cpn-model/inputs/in.* .
```

3. Now we're ready to run a simple simulation. This is as simple as executing

```
${D_1CPN}/lammps-1cpn/src/lmp_serial -i in.1cpn
```

Users unfamilair with LAMMPS are referred to the LAMMPS docmentation for descriptions on LAMMPS output, and how to run different simulations.

Congratulations! You've just performed (hopefully) your first simulation with 1CPN!

1.2.3 Vizualizing the Simulation

Now that you've run your simulation. You'll probably want to visualize it. 1CPN comes packaged with several different vizualization options. To learn more, check out *Vizualization of 1CPN*.

1.3 Analyzing the Simulation

Visualizing a simulation is fun, but you'll probably want to perform some sort of analysis on it. For example, you might want to compute the the end-to-end distance if a single chromatin fiber, or the distance (or angle) between two nucleosomes.

To perform these sorts of analysis, 1CPN comes packaged with a variety of analysis scripts. See more at *Analysis of 1CPN Simulations*.

CHAPTER 2

1CPN Implementation in Lammps

Unfortunately, this page is still a bit rough. Its goal is to give an idea of the custom potentials specific to 1CPN that we implemented into LAMMPS and what the different parameters mean.

In this regard, this page is currently okay, but I'll conceed that it could use a bit of work.

2.1 Linking 1CPN with LAMMPS

To link 1CPN to LAMMPS please check out *Quick Start*.

2.2 1CPN Performance across many processors

The key to good 1CPN performance across multiple processors is the LAMMPS *fix balance* command.

2.3 Custom 1CPN Potentials

While developing the 1CPN model, many custom potentials were necessary

2.3.1 Pair Zewdie

Give functional form of potential? And an explanation of the pair_coeffs arguments

Usage:

```
pair_style zewdie ${pe000} ${pecc2} ${pe220} ${pe222} ${pe224} ${ps000} ${pscc2} $
↪{ps220} ${ps222} ${ps224}

pair_coeff   1 1 zewdie ${pe0} ${ps0} ${cutoff}
```

- *pe000* - See ϵ_{000} in (2.3)
- *pecc2* - See ϵ_{cc2} in (2.3)
- *pe220* - See ϵ_{220} in (2.3)
- *pe222* - See ϵ_{222} in (2.3)
- *pe224* - See ϵ_{224} in (2.3)
- *ps000* - See σ_{000} in (2.2)
- *pscc2* - See σ_{cc2} in (2.2)
- *ps220* - See σ_{220} in (2.2)
- *ps222* - See σ_{222} in (2.2)
- *ps224* - See σ_{224} in (2.2)
- *pe0* - See ϵ_0 in (2.3)
- *ps0* - See σ_0 in (2.2)

$$U_{Zewdie}(r_{ij}, \hat{f}_i, \hat{f}_j\,;\,\sigma_0, \epsilon_0) = 4\epsilon\left[\left(\frac{\sigma_0}{r_{ij}-\sigma+\sigma_0}\right)^{12} - \left(\frac{\sigma_0}{r_{ij}-\sigma+\sigma_0)}\right)^{6}\right] \tag{2.1}$$

where

$$\sigma = \sigma_0[\sigma_{000}S_{000} + \sigma_{cc2}(S_{022}+S_{202}) + \sigma_{220}S_{220} + \sigma_{222}S_{222} + \sigma_{224}S_{224}] \tag{2.2}$$

$$\epsilon = \epsilon_0[\epsilon_{000}S_{000} + \epsilon_{cc2}(S_{022}+S_{202}) + \epsilon_{220}S_{220} + \epsilon_{222}S_{222} + \epsilon_{224}S_{224}]. \tag{2.3}$$

and the S-functions are defined as

$$\begin{aligned}
S_{000} &= 1,\\
S_{202} &= (3a_1^2-1)/2\sqrt{5},\\
S_{022} &= (3a_2^2-1)/2\sqrt{5},\\
S_{220} &= (3a_0^2-1)/2\sqrt{5},\\
S_{222} &= \frac{1}{\sqrt{70}}(2-3a_1^2-3a_2^2-3a_0^2+9a_0a_1a_2),\\
S_{224} &= \frac{1}{4\sqrt{70}}(1+2a_0^2-5a_1^2-5a_2^2-20a_0a_1a_2+35a_1^2a_2^2)
\end{aligned}$$

with

$$\begin{aligned}
a_0 &= \hat{f}_i \cdot \hat{f}_j,\\
a_1 &= \hat{f}_i \cdot \hat{r}_{ij},\\
a_2 &= \hat{f}_j \cdot \hat{r}_{ij}
\end{aligned}$$

Discuss how sphere-ellipse interactions are handled

2.3.2 Pair Gauss Aniso

Usage:

```
pair_style gauss/aniso ${gauss_rcut}

pair_coeff 2 3  sigma d0 r0 theta0 phi0 Ktheta Kphi
```

- *sigma* - Gaussian width. Given by σ in (2.4)
- *d0* - Gaussian depth. Given by d_0 in (2.4)
- *r0* - Gaussian Center Position. Given by r_0 in (2.4)
- *theta0* - Equilibrium position with with respect to θ, where $\theta = \arccos(\hat{r}_{ij} \cdot \hat{u}_i)$. See θ_0 in (2.4)
- *phi0* - Equilibrium position with with respect to ϕ, where $\phi = \arccos(\hat{r}_{ij} \cdot \hat{f}_i)$. See ϕ_0 in (2.4)
- *Ktheta* - Width of modulating function with respect to θ. See K_θ in (2.4)
- *Kphi* - Width of modulating function with respect to ϕ. See K_ϕ in (2.4)

Warning: In the implementation, of this potential, the atom_types of sites i and j cannot be the same.

This is because the ith particle is always chosen to be the atom with the lower type index.

Not symmetric.

For example

$$
\begin{aligned}
U_{gauss,aniso} &= f(K_\theta, \Delta\theta) f(K_\phi, \Delta\phi)\, U_{gauss} \\
&= f(K_\theta, \Delta\theta) f(K_\phi, \Delta\phi) \left(-d_0 e^{-(r-r_0)^2/2\sigma^2}\right)
\end{aligned}
$$

$$
f(K_\theta, \Delta\theta) = \begin{cases} 1 & -\frac{\pi}{2K_\theta} < \Delta\theta < \frac{\pi}{2K_\theta} \\ 1 - \cos^2(K_\theta \Delta\theta) & \frac{-\pi}{K_\theta} < \Delta\theta < \frac{-\pi}{2K_\theta} \text{ or } \frac{\pi}{2K_\theta} < \Delta\theta < \frac{\pi}{K_\theta} \\ 0 & \Delta\theta < -\frac{\pi}{K_\theta} \text{ or } \Delta\theta > \frac{\pi}{K_\theta} \end{cases} \tag{2.4}
$$

2.3.3 Angle Orient

Usage::

```
angle_style orient
angle_coeff 1 angle_{f,v,u} ktheta1 ktheta2 kphi theta1 theta2 phi
```

- *angle_vector* - Possible values *angle_f*, *angle_v*, or *angle_u*. Defines whether $\hat{w} = \{\hat{f}, \hat{v}, \hat{u}\}$.
- *ktheta1* - Spring constant for deformations in θ_1. See (2.5).
- *ktheta2* - Spring constant for deformations in θ_2. See (2.5).
- *kphi* - Spring constant for deformations in ϕ. See (2.5).
- *theta1* - Equilibrium value of θ_1. See $\theta_{1,0}$ in (2.5).
- *theta2* - Equilibrium value of θ_2. See $\theta_{2,0}$ in (2.5).
- *phi* - Equilibrium value of ϕ. See ϕ_0 in (2.5).

$$
U = \frac{1}{2}\left(k_{\theta_1}(\theta_1 - \theta_{1,0})^2 + k_{\theta_2}(\theta_2 - \theta_{2,0})^2 + k_\phi(\phi - \phi_0)^2\right) \tag{2.5}
$$

where θ_1, θ_2, ϕ are given by

$$
\begin{aligned}
\theta_1 &= \arccos(\hat{w}_i \cdot \hat{r}_{ij}) \\
\theta_2 &= \arccos(\hat{w}_j \cdot \hat{r}_{ij}) \\
\phi &= \arccos(\hat{w}_i \cdot \hat{w}_j)
\end{aligned}
$$

2.3.4 Angle Orient Cosine

Usage::

```
angle_style orient/cosine
angle_coeff 1 angle_{f,v,u} ktheta1 ktheta2 kphi theta1 theta2 phi
```

- *angle_vector* - Possible values *angle_f*, *angle_v*, or *angle_u*. Defines whether $\hat{w} = \{\hat{f}, \hat{v}, \hat{u}\}$.
- *ktheta1* - Spring constant for deformations in θ_1. See (2.6).
- *ktheta2* - Spring constant for deformations in θ_2. See (2.6).
- *kphi* - Spring constant for deformations in ϕ. See (2.6).
- *theta1* - Equilibrium value of θ_1. See $\theta_{1,0}$ in (2.6).
- *theta2* - Equilibrium value of θ_2. See $\theta_{2,0}$ in (2.6).
- *phi* - Equilibrium value of ϕ. See ϕ_0 in (2.6).

$$U = \frac{1}{2}\left[k_{\theta_1}\left(1 - \cos(\theta_1 - \theta_{1,0})\right) + k_{\theta_2}\left(1 - \cos(\theta_2 - \theta_{2,0})\right) + k_{\phi}\left(1 - \cos(\phi - \phi_0)\right)\right] \tag{2.6}$$

where θ_1, θ_2, ϕ are given by

$$\begin{aligned}\theta_1 &= \arccos(\hat{w}_i \cdot \hat{r}_{ij}) \\ \theta_2 &= \arccos(\hat{w}_j \cdot \hat{r}_{ij}) \\ \phi &= \arccos(\hat{w}_i \cdot \hat{w}_j)\end{aligned}$$

2.3.5 Angle WLC Twist

Usage::

```
angle_style wlctwist
angle_coeff 2 wlctwist ${kalign} ${ktwist} ${omega0}
```

- *kalign* - Alignment spring constant
- *ktwist* - Twist spring constant
- *omega0* - Equilibrium Twist

This potential is based off of the implementation of Brackley et al.

$$U = k_{\omega}\left(1 - \cos(\omega_i - \omega_0)\right) + k_{\psi}\left(1 - \cos(\psi_i)\right) \tag{2.7}$$

2.4 Maintaing Compatability with LAMMPS

In order to maintain compatability of 1CPN with the most recent version of LAMMPS it is helpful to know which core LAMMPS potential the 1CPN potentials were derived from. By seeing what changed between the core LAMMPS potentials (i.e. diff old and new version), it is typically straightforward to make the necessary minor changes to the 1CPN potential to allow 1CPN-LAMMPS to compile.

When trying a new version of LAMMPS, be sure to run the integration tests in *${D_1CPN}/test/integ_tests*, to make sure the model is behaving correctly

1CPN Potential	Original Lammps Potential
pair_style zewdie	pair_style gayberne
pair_style gauss/aniso	pair_style gayberne, pair_style gauss
angle_style wlctwist	angle_style wlctwist (Brackley2014)
angle_style orient	angle_style wlctwist
angle_style orient/cosine	angle_style wlctwist

CHAPTER 3

Initialization

Initial configurations for 1CPN can be generated using *input/init_1cpn.py*

init_1cpn.py can take various command line arguements:

- `--stemangle` (`-a`) : This is the angle alpha
- `--nrl` (`-nrl`) : Nucleosome Repeat Length
- `--nrlends` (`-nrlends`) : Nucleosome Repeat Length of DNA at beginning and end of fiber. This arguement defaults to set `nrlends=nrl` if not specified
- `--nnucl` (`-n`): number of nucleosomes
- `--linkerhistone` (`-lh`): turn on linker histones

Note: If you only want to generate a long strand of DNA (without any nucleosomes), you can do this by adding `--nnucl 0` to the command line arguements. Then specify the length of DNA (in base pairs) with the `--nrl` entry.

Note: If you use the `-lh` flag to turn generate the linker histone, you'll also need to change a single line in `in.1cpn`. You'll want to change this line to

```
variable lh equal 1
```

CHAPTER 4

Vizualization of 1CPN

Vizualizing the 1CPN model is more chalenging than many other molecular models because each coarse-grained sit in 1CPN in represented by an anisotropic potential. Most vizualization tools (like VMD) are designed for isotropic sites, and generally represent each site by a sphere. Consequently, due to the anisotropic potentials in 1CPN, vizualizing the model requires some additional steps.

The 1CPN model includes offers several workflows for vizualizing 1CPN trajectories. The ideal workflow depends on the type of simulation you have run, and the phenomena of interest that you want the simulation to highlight. Below we describe two approaches for visualizing 1CPN with VMD (Visual Molecular Dynamics) and one approach for high quality vizualization using Blender.

Users who do not have VMD should install it following the instructions at the VMD website (note that VMD 1.9.2 was used in this tutorial).

Throughout this tutorial it is assumed that VMD is accessed though *vmd* at the command line.

4.1 Option 1: Basic VMD

Speed = Fast, Vizualization Quality=Poor

The 1CPN can be quickly vizualized by loading *traj.dump* directly into LAMMPS.

```
vmd in.psf -lammpstrj traj.dump
```

Now that the trajectory is loaded into LAMMPS, a suitable visual representation can be loaded by Opening the *Tk console* by clicking on Extensions->Tk Console, in the *VMD Main* window, and executing the following command.

```
source <path-to-D_1CPN>/1cpn-model//utils/vmd/1cpn.vmd
```

Note that in this case the bash variable *${D_1CPN}* cannot be used and *<path-to-D_1CPN>* must be hardcoded manually. If you will be vizualizing the 1CPN Model frequently, it is recommended that you add this *source* command a file located at *~/.vmdrc* so that it will be executed automatically when VMD opens.

Though this visualization can be useful for some purposes (like examining bond lengths and angles), it is generally unsuitable for visualizing large simulations.

FIXME make 1cpn.vmd look acceptable for vizualizing the raw traj.dump FIXME SAMPLE VIZUALIZATION

4.2 Option 2. Enhanced VMD (recommended)

Speed = Medium-Fast, Vizualization Quality=High

Another approach for vizualizing the 1CPN model is to convert *traj.dump* into a new file where anisotropic sites are more easily vizualized, and then visualizing this new trajectory file direcly in VMD.

In 1CPN, this conversion tool is an executable *dump-to-xyz* that is located in *${D_1CPN}/bin/* and is compiled by

```
cd ${D_1CPN}/1cpn-model/src/viz
make
```

FIXME, this breaks if LAMMPS_SRC isn't canonical. FIXME remove quat_vec_rot from trajectory_iterator.h FIXME have a single Makefile in ${D_1CPN}

The executable *dump-to-xyz* takes two arguements:

1. The filename of an existing trajectory (*traj.dump* in this example), and
2. The filename prefix of the new trajectory files to be generated (*traj_1cpn* here).

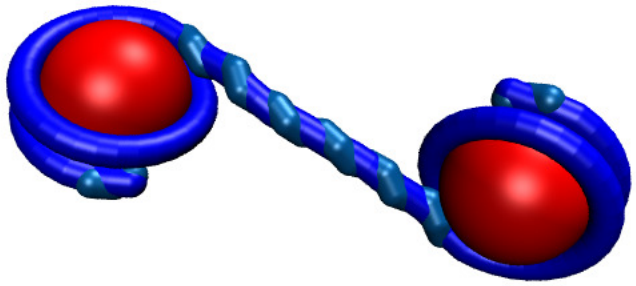
```
${D_1CPN}/1cpn-model/bin/dump-to-xyz traj.dump traj_1cpn
```

After executing this command, two new files (*traj_1cpn.psf* and *traj_1cpn.xyz*) are generated. These files contain a modified topology file (*.psf*) and a modified trajectory file (*.xyz*). Specifically, these new files contain create additioal (fictitious) sites that show the orientation of each 1CPN site. For Nucleosome sites, these fictitious sites are created to highlight their anisotropic shape (from the Zewdie potential LINKME). For DNA sites, fictitious sites are created to highlight the twist of the DNA molecule.

These new files can be vizualized with VMD

```
vmd traj_1cpn.psf traj_1cpn.xyz
# or
vmd traj_1cpn.*
```

An example of a nucleosome rendered using Option 2 is shown here:



4.3 Option 3. Blender

Speed=Low, Vizualization Quality=Very High

The final option for vizualizing the 1CPN model is to use the open source 3D conputer graphics toolset Blender. Blender is a very powerful flexible tool for vizualization and premits the customization of virtually any feature of a digital vizualization. Blender is not strictly designed for molecular vizualizaiton, and therefore

This tutorial assumes a basic familiarity with Blender

FIXME Is blender okay for many frames

FIXME Clean up front of Blender script so that its easier for a new person

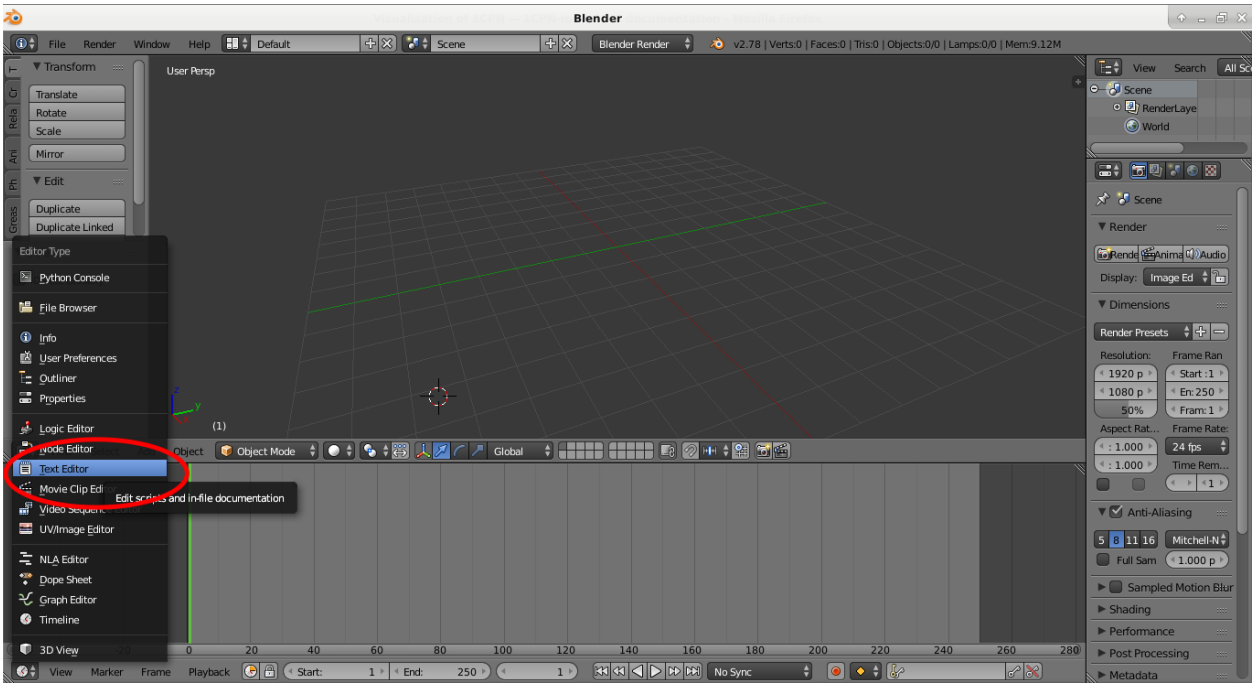
1. Setting up file paths and rendering options.

The main script of interest is *${D_1CPN}/1cpn-model/utils/blender/import_dump.py* and cha

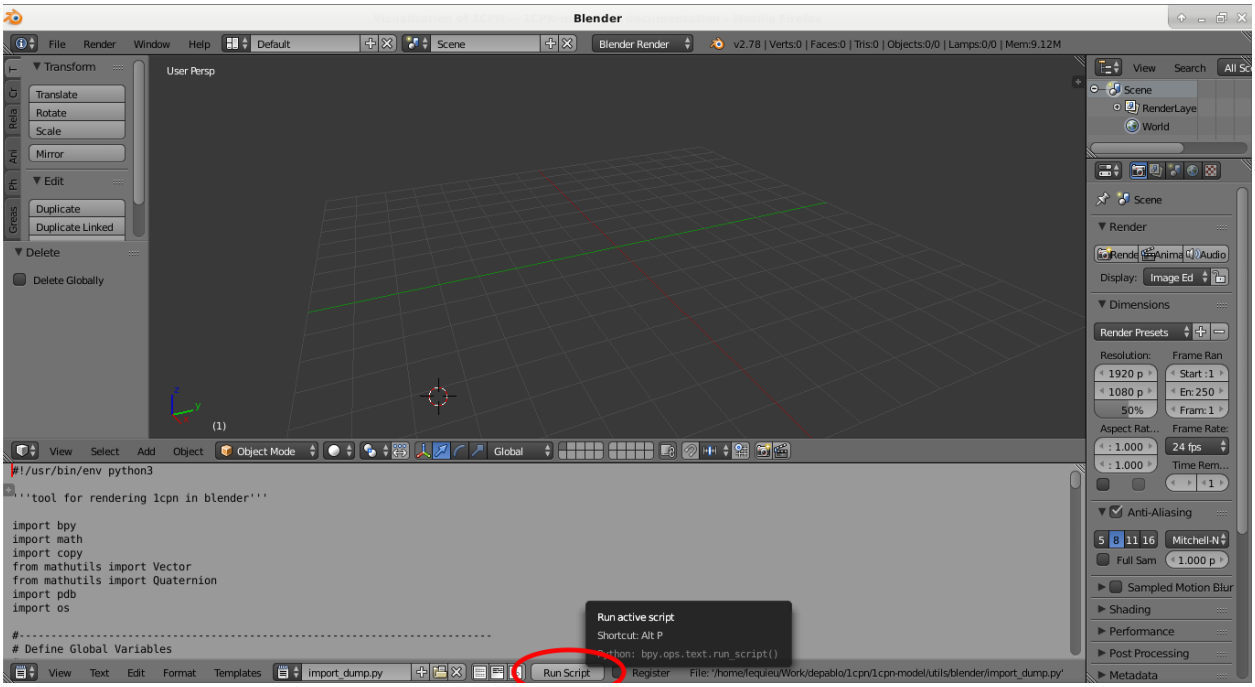
PATH_ZEWDIE = PATH_TRAJ =

:: cd /tmp ln -s ${D_1CPN}/example/in.dump frame.dump

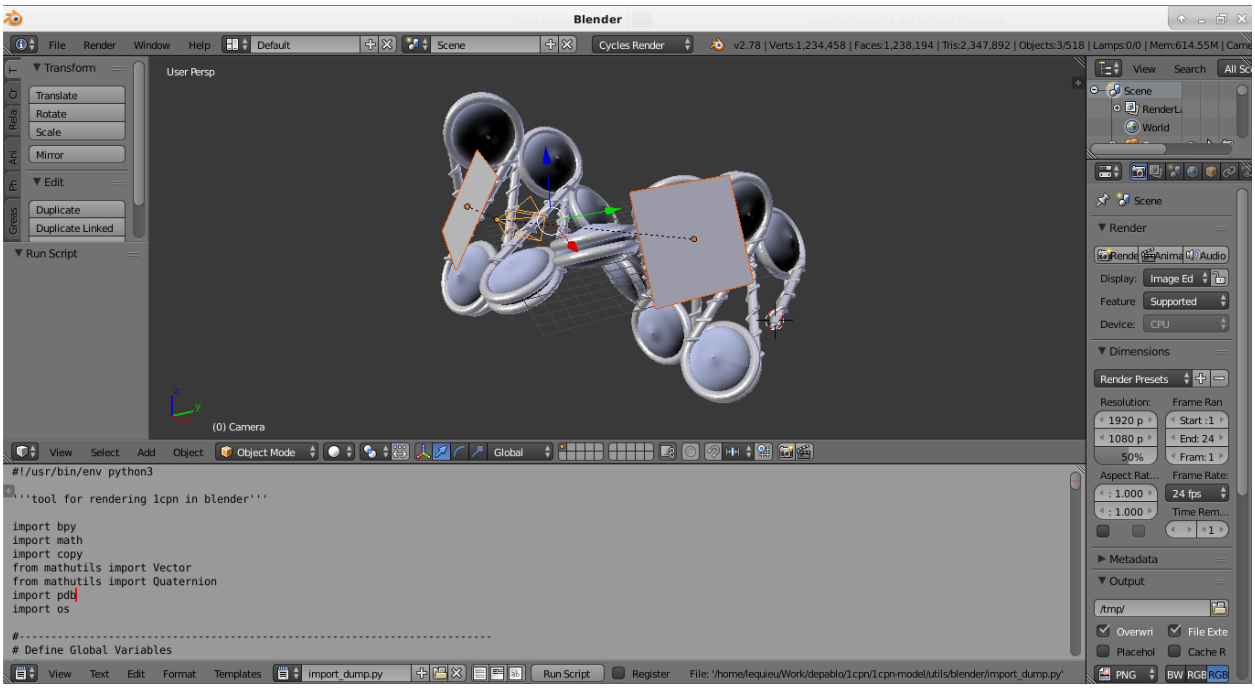
2. Open blender, and change the bottom pannel to text editor



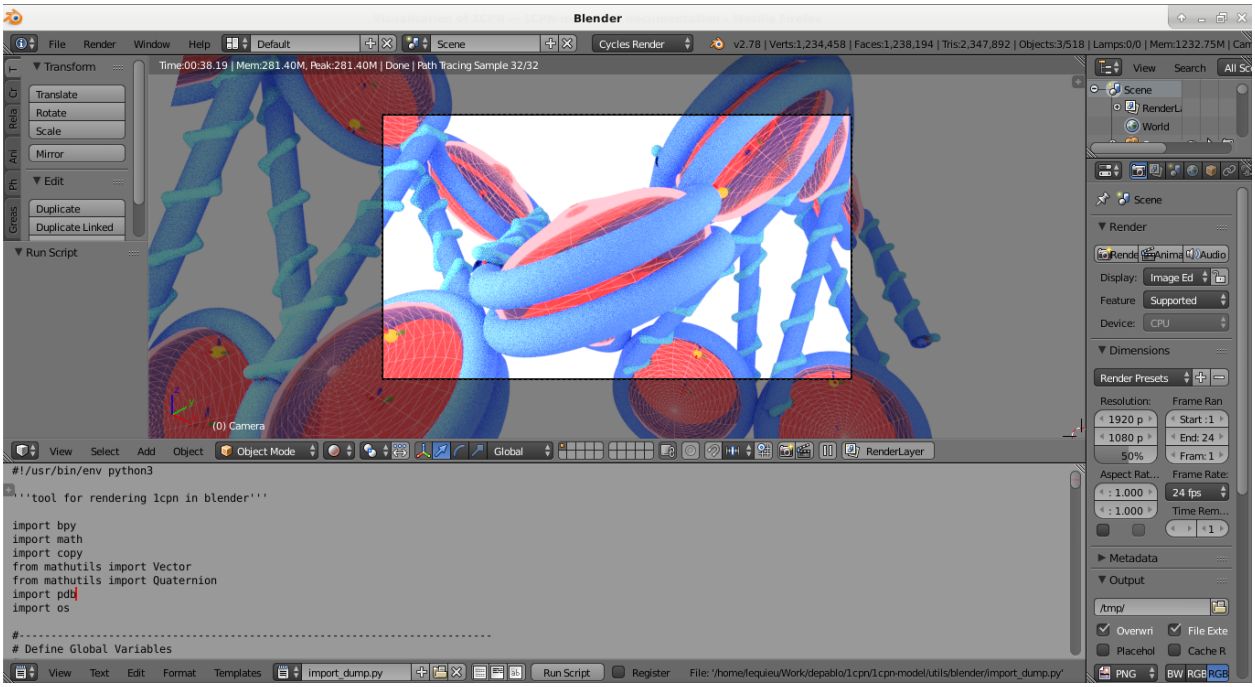
3. In the Text Editor window, click the *Open* button, and navigate to *${D_1CPN}/1cpn-model/utils/blender/import_dump.py*. Once the file is loaded, click the *Run Script* button. This could take a minute or two to run.



4. Now the 1CPN Model should be loaded into Blender.



5. To see the artistic rendering, move your mouse into the *3D View* panel and press *Shift-Z* to turn on Materials Rendering (or select *Viewport Shading -> Rendered* using your mouse.) Switching to camera view results in the following image.



Now that 1CPN is loaded into Blender, anything is possible. Change try new materials, add camera animations, go nuts!

Note: If you're going to be adventerous and start making significant modifications to *${D_1CPN}/1cpn-model/utils/blender/import_dump.py*, I would suggest making file edits in your own text editor and not Blender's build-in one. If your choose to use your own text edior, just make sure to reload the updated *import_dump.py* in Blender's text editor before clicking the *Run Script* button. Blender should notice when *import_dump.py* is changed and notify you, but its important to remember that Blender doesn't refresh its internal text editor by default.

CHAPTER 5

Analysis of 1CPN Simulations

Sorry! This part of the documentation is still under construction.

Take a look in the the *src/analysis* directory to see some sample analysis scripts. If you know a bit of C++ you should be able to copy one of these *.cpp* files and modify it to perform your analysis of interest.

You'll also want to checkout *src/include/trajectory_iterator.h*. This is a library we've put together that makes parsing of LAMMPS files a bit easier. Almost all of the analysis files in *src/analysis/* include *trajectory_iterator.h* and show the basic commands on how to use it.

CHAPTER 6

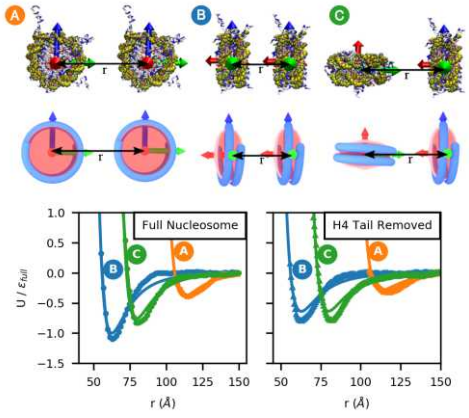
Corrections of the 1CPN Paper

This page is about the corrections made to the 1CPN paper:

Lequieu, Cordoba, Moller, de Pablo "1CPN: A coarse-grained multi-scale model of chromatin" (2019) J. Chem. Phys. 150, 215102

6.1 Updates on Fig. 6 of 1CPN Paper

The original version of Fig. 6 included in the 1CPN paper was inconsistent with the parameters of the Zewdie potential provided in Table S3. The corrected figure is shown here:



(the points are the pair-potential calculated using the 3SPN-AICG model, and the lines are the Zewdie potential fit. Notice that the y label should be ε_{full} rather than $\varepsilon_{0,full}$)

6.2 $\varepsilon_{0,H4cut}$ in TABLE. S3 of 1CPN paper

The value of the $\varepsilon_{0,H4cut}$ included in the TABLE. S3 was 1.303 kcal/mol, and should be corrected as 1.0081 kcal/mol as in the github repository *${D_1CPN}/inputs/in.var-zewdie-H4*.

Note that none of these corrections actually affect the results produced by the 1cpn model.