# DATASET FOR THE $LJ_{6-14}$ AGGREGATION/DEFORMATION NETWORK AND MATLAB CODES FOR ITS CREATION

MARIA CAMERON, YAKIR FORMAN, SEBASTIAN SOUSA CASTELLANOS

Contents



The package `LJaggregation.zip` contains the folders named `LJNdata`, $N = 5, \ldots, 14$, `Connect_data` and `MatlabCodes`.

## 1. The dataset for the $LJ_{6-14}$ aggregation/deformation network

1.1. **Minima and saddles.** The folders LJNdata ($N = 6, \ldots, 14$) contain the data used for building the aggregation/deformation $LJ_{6-14}$ network described and analyzed in [1]. These folders contain the following files: `LJNmin_data.txt`, `LJNmin_xyz.txt`, `LJNts_data.txt`, `LJNts_xyz.txt`.

Format of `LJNmin_data.txt`:

$$V \quad \log(\text{abs}(\det(Hess))) \quad PGO$$

where $V$ is the potential energy of the minimum, $Hess$ is the Hessian matrix at the minimum, $PGO$ is the point group order.

Format of `LJNts_data.txt`:

$$V \quad \log(\text{abs}(\det(Hess))) \quad \omega \quad PGO \quad \min 1 \quad \min 2$$

where $V$ is the potential energy of the Morse index one saddle, $Hess$ is the Hessian matrix at the saddle, $\omega$ is the absolute value of the only negative eigenvalue of the Hessian at the saddle, $PGO$ is the point group order, min1 and min2 are the indices of the minima separated by the saddle.

Format of `LJNmin_xyz.txt` and `LJNts_xyz.txt`:

$$\begin{array}{cccc} x_{11} & x_{12} & \dots & x_{1N} \\ y_{11} & y_{12} & \dots & y_{1N} \\ z_{11} & z_{12} & \dots & z_{1N} \\ \vdots & \vdots & & \vdots \\ x_{M1} & x_{M2} & \dots & x_{MN} \\ y_{M1} & y_{M2} & \dots & y_{MN} \\ z_{M1} & z_{M2} & \dots & z_{MN} \end{array}$$

where $x_{ij}$, $y_{ij}$, $z_{ij}$ are are the $xyz$ coordinates of atom $j$ of minimum or saddle $i$.

1.2. **Connection of $\mathbf{LJ}_N$ to $\mathbf{LJ}_{N-1}$ and $\mathbf{LJ}_{N+1}$.** The folder `Connect_data` contains files `AtomRemovalNtoN-1.txt` and `ConnectNandN+1.txt`

Format of `ConnectNandN+1.txt`

$$\text{min of LJ}_N \quad \text{min of LJ}_{N+1} \quad \text{transition probability}$$

Only nonzero transition probabilities are displayed.

Format of `AtomRemovalNtoN-1.txt`

$$\text{min of LJ}_N \quad \text{atom of LJ}_N \quad \text{min of LJ}_{N-1} \quad E$$

where $E$ is the energy of interaction of atom $i$ with the rest of the atoms in the cluster.

## 2. The List of Matlab codes

The programs in the folder `MatlabCodes` were used to create the data in `LJNdata` and `Connect_data`.

(1) `find_minima.m` is the first program to run. It offers two options for finding local energy minima: from random configurations and using basin hopping.
(2) `find_saddles.m` calling `find_saddle_minmode_dimer.m` finds Morse index one saddles and finds new local minima. Some collection of local minima must be precomputed, e.g., by `find_minima.m`. The implemented technique for finding saddles proposed by S. Sousa Castellanos combines the gentlest ascend method and the dimer method.
(3) `point_group_order.m` computes the point group order by the technique based on the orbit-stabilizer theorem proposed by Y. Forman. `pgo = point_group_order(xyz)` is called by any program in the package that has the capacity to find a new minimum or a new saddle. Input: `xyz` , the 3-by-$N$ matrix of coordinates of the configuration of atoms. Output: `pgo`, the point group order.
(4) `glue_networks.m` computes the transition probabilities from minima of $\text{LJ}_N$ to minima of $\text{LJ}_{N+1}$ as a result of a new atom arrival. The implemented technique is developed by Y. Forman and M. Cameron. Its details are described in [1].
(5) `remove_one_atom.m` finds the minima of $\text{LJ}_{N-1}$ resulting from a one atom removal from minima of $\text{LJ}_N$.

(6) `sort_data.m` sorts the energy minima in ascending order and modifies the other data files referring to minima by their indices accordingly.
(7) `visualconf.m` draws energy minima as polyhedra. Indicate $N =$ `Na`, the number of atoms, and `iconf`, the index of minimum to be visualized, inside the program.
(8) `drawLJconf.m` draws atom configurations using balls. Input: `xyz`, the 3-by-$N$ matrix of coordinates of the configuration of atoms, and `fig`, the figure number in which you want the configuration to be displayed.
(9) `BFGS_trust_region.m` finds local minima using Trust Region BFGS method with maximal trust region radius `Rmax = 0.1`. (`Rmax` can be changed inside the program). Input: `xyz`, the 3-by-$N$ matrix of coordinates of the configuration of atoms. `tol` (the tolerance), stop if the norm of the gradient is less than `tol`. We used `tol = 1e-9`. `iter_max`, the maximal number of iterations allowed. We used `iter_max = 500`. Output: `xyz`, the 3-by-$N$ matrix of coordinates of the configuration of atoms. `f`, the value of energy. `freq` = log(abs(det($Hess$))). `flag`, a logic variable. `flag = 1` if the method found an isolated local minimum, `flag = 0` otherwise.
(10) `Newton_trust_region.m` finds local minima using Trust Region Newton method with maximal trust region radius `Rmax = 0.1`. (`Rmax` can be changed inside the program). Input: `xyz`, the 3-by-$N$ matrix of coordinates of the configuration of atoms. `tol` (the tolerance), stop if the norm of the gradient is less than `tol`. We used `tol = 1e-9`. `iter_max`, the maximal number of iterations allowed. We used `iter_max = 500`. Output: `xyz`, the 3-by-$N$ matrix of coordinates of the configuration of atoms. `f`, the value of energy. `freq` = log(abs(det($Hess$))). `flag`, a logic variable. `flag = 1` if the method found an isolated local minimum, `flag = 0` otherwise.
(11) `make_random_configuration.m` creates a random atomic configuration such that any atom touches at least one other atom, and given any two atoms, there is a sequence of touching atoms (i.e., each consecutive pair of atoms in the sequence touches) beginning at one of the given atoms and ending at the other. Input: `Na` = the number of atoms.
(12) `remove_rotations_translations.m` applies a translation and an orthogonal transformation to the input configuration `xyz` so that atom 1 is moved to the origin, atom 2 is placed on the $x$-axis, and atom 3 is put on the $xy$-plane. Input: `xyz`, the 3-by-$N$ matrix of coordinates of the configuration of atoms. Output: `x`, the $(3N-6)$-by-1 column vector of the new coordinates where the zeros corresponding to $x_1$, $y_1$, $z_1$, $y_2$, $z_2$, and $z_3$ are removed.
(13) `transition_rate.m` computes the transition rate via the saddle with index `tsindex` of $LJ_N$ at temperature `tem`. $N =$ `Na`. The saddle `tsindex` separates minima `mi1 = ts(tsindex,5)` and `mi2 = ts(tsindex,6)`. This code is provided to demonstrate how to use the data in `LJNdata` to calculate the transition rates via each saddle in $LJ_N$.

## 3. Getting started

If you want to generate a dataset using our codes, pick the number of atoms $N =$ `Na` and start with creating a collection of local minima for $\mathrm{LJ}_N$ using `find_minima.m`. Rename the files to `LJNmin_data.txt` and `LJNmin_xyz.txt`, create a folder `LJNdata` and place these files there. Then run `find_saddles.m` to find saddles. You probably will find some more local minima as well, and they will be added to `LJNmin_data.txt` and `LJNmin_xyz.txt`. If new minima are found, make sure to run `find_saddles.m` for all newly found minima. Do this for a collection of consecutive values of the number of atoms $N =$ `Na`. Then connect each $\mathrm{LJ}_N$ to $\mathrm{LJ}_{N+1}$ by running `glue_networks.m`. If new minima are found, they will be added to `LJNmin_data.txt` and `LJNmin_xyz.txt`. If you want to connect $\mathrm{LJ}_N$ to $\mathrm{LJ}_{N-1}$, run `remove_one_atom.m`. (This is not done in the network studied in [1].) You can sort all local minima for each $N$ in increasing order of their energies using `sort_data.m` afterwards.

## References

[1] Y. Forman and M. Cameron, Modeling aggregation processes of Lennard-Jones particles via stochastic networks, `arXiv1612.09599`