BIN/IGM-RUN.SH

INPUT FILES
- Configuration file (**.json**)
- Data inputs (.hcs, .txt or h5f)

PREPROCESS

`cfg = igm.Config`
(**core/config.py**)

`igm.Preprocess`
(**./_preprocess.py**)

`igm.SetupLogging`

`igm.RandomInit`
(**steps/RandomInit.py**)

`igm.RelaxInit`
(**steps/RelaxInit.py**)

Introduce igm/model/ model.py object,

add restraints/ restraint.py like envelope.py, steric.py and polymer.py

$\mathbf{X}^{(0)}$

ASSIGNMENT (A)

`igm.ActivationDistanceStep.py`
(./hcs. >> **steps/ActivationDistanceStep.py** >> tmp/.h5)

`igm.DamidActivationDistanceStep.py`
(./txt. >> **steps/DamidActivationDistanceStep.py** >> tmp/.txt)

`igm.FishAssignment.py`
(./h5. >> **steps/FishAssignment.py** >> tmp/.h5)

`igm.SpriteAssignment.py`
(./h5. >> **steps/SpriteAssignment.py** >> tmp/.h5)

MODELING (M)

`igm.ModelingStep.py`
(**.steps/ModelingStep.py**

Define Python model object
(**igm/model, forces, particles, addRestraint**)

Add particles and restraints to model

(**.restraints/hic.py**	addForce
restraints/damid.py	addForce
restraints/fish.py	addForce
restraints/sprite.py)	addForce

The python model is used to create the input/script files to run Simulated Annealing/Conjugate Gradient

model.optimize >> model/kernel/Lammps.py
.lam, .input files >> lamps executable >> .log, .lamptrj, .hms
>> optimized population .hss $\mathbf{X}^{(t+1)}$

Use **X** and `igm.model` to compute violations

OUTPUT FILES
- Population stats file (**.hss**)
- Logger (**.log**)
- (Temp files in tmp folder

(all steps are called as function of the configuration object cfg)

(temporary files are not always listed in the scheme)

Computing Environment

`Controller(cfg)`
(**parallel/__init__.py**)

`async_file_operations`
(**parallel/_async_file_operations.py**)

`Class Step()`
(**core/step.py**)

Core ingredients

`Configuration`
(**core/config.py**)

`DB database`
(**core/job_tracking.py**)

Optimization

`A/M Steps`
(**model/steps/restraints**)

`Postprocess`
(**report**)