Deep Langevin FTS

Langevin Field-Theoretic Simulation (L-FTS) Accelerated by Deep Learning (DL)

Features

- L-FTS incorporated with DL
- AB Diblock Copolymer Melt
- Chain Model: Continuous, Discrete
- Periodic Boundaries
- Pseudospectral Method
- Platform: CUDA

Dependencies

Linux System

Anaconda

Langevin FTS

Install `v1.0-paper` release.

```
git clone -b v1.0-paper https://github.com/yongdd/langevin-fts.git
```

Installation

`Langevin FTS`, `PyTorch` and `PyTorch-lightning` should be installed in the same virtual environment. For instance, if you have installed `Langevin FTS` in virtual environment `lfts`, install `PyTorch` and `PyTorch-lightning` after activating `lfts` using the following commands. (Assuming the name of your virtual environment is `lfts`)

```
conda activate lfts
git clone -b v1.0-paper \
    https://github.com/yongdd/deep-langevin-fts.git
conda install pip protobuf=3.19 matplotlib pytorch \
    torchvision torchaudio cudatoolkit=11.3 -c pytorch
pip install pytorch-lightning
```

The above commands will install the following libraries.

- **PyTorch**
  An open source machine learning framwork
  https://pytorch.org/get-started/locally/
- **PyTorch-lightning**
  High-level interface for PyTorch

https://www.pytorchlightning.ai/

Usage

1. Set Simulation Parameters

```
vi input_parameters.yaml
```

Edit `input_parameters.yaml`. All the system parameters are stored in this file. You may proceed to step 4 to run L-FTS without DL. If you plan to use DL but you do not want to touch the details, only edit the upper part of this file.

2. Generate Training Data

```
python make_training_data.py
```

You may need to change the initial fields by modifying `w_plus` and `w_minus` in `make_training_data.py`. Training data will be stored in `data_training` folder, and it will generate `LastTrainingStep.mat` file. This generated file will be used as inital field for `find_best_epoch.py` and `run_simulation.py`.

3. Train a Neural Network

```
python train.py
python find_best_epoch.py
```

If you plan to use multiple GPUs for training, edit `gpus` in `train.py`. To obtain the same training results using multiple GPUs, you need to change `batch_size` so that `gpus` * `batch_size` does not change. For example, if you use 4 GPUs, set `gpus=4` and `batch_size=8`, which is effectively the same as setting `gpus=1` and `batch_size=32`. For each epoch, the weight of model will be stored in `saved_model_weights` folder.

Lastly, `find_best_epoch.py` will tell you which training result is the best. The training result is not always the same. If you are not satified with the result, run `train.py` once again.

4. Run Simulation

For those who do not want to use DL, set `use_deep_learning = False`. When using DL, edit `run_simulation.py` to use the best epoch. For example, set `model_file = "saved_model_weights/epoch_92.pth"` if the 92nd epoch was the best one. Then, run the simulation.

```
python run_simulation.py
```

Polymer density, fields and structure function will be stored in `data_simulation` folder.

5. Data Visualization

Matlab and Python scripts for visualization and renormalization are provided in `tools` folder of `yongdd/langevin-fts` repository.

Citation

Daeseong Yong, and Jaeup U. Kim, Accelerating Langevin Field-theoretic Simulation of Polymers with Deep Learning, *Macromolecules* **2022**, accepted