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
- Platforms: CUDA

Dependencies

Linux System

Anaconda

Langevin FTS

L-FTS for Python
https://github.com/yongdd/langevin-fts

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 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/

After the installation, you can run `python test_performance.py` in `examples/Gyroid` folder, which performs a L-FTS with a pretrained model to test your installation. You can compare its performance with Anderson mixing by repeating simulation after setting `use_deep_learning=False` in `test_performance.py`.

Usage

1. Set Simulation Parameters

```
vi input_parameters.yaml
```

Edit `input_parameters.yaml`. All the system parameters are stored in this file. If you do not want to touch the DL part, only edit this file and proceed. If you want to use a pre-trained model stored in `examples` folder, go to step 4.

2. Generate Training Data

```
python make_training_data.py
```

Initial fields are currently for gyroid phase. 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. A sample `LastTrainingStep.mat` file already exists, and this file or the 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
```

If you are 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 weights of model will be stored in `saved_model_weights` folder.

```
python find_best_epoch.py
```

Lastly, `find_best_epoch.py` will tell you which training result is the best, and it will copy the weights of best epoch as `best_epoch.pth`. A sample file already exists. The training result is not always the same. If you are not satified with the result, run `train.py` once again.

4. Run Simulation

```
python run_simulation.py
```

This will use `best_epoch.pth` obtained at the previous step. You can use a pre-trained model in `examples` folder instead. For example, set `model_file = "example/Gyroid/gyroid_atr_cas_mish_32.pth"` if you want to run simulation for gyroid phase. Polymer density, fields and structure function will be recored in `data_simulation` folder.

Notes

- Matlab and Python scripts for visualization and renormalization are provided in `tools` folder of `yongdd/langevin-fts` repository.
- In `examples` folder, input fields obatained using SCFT, yaml files for input parameters, pre-trained model weights, and field configurations at equilibrium states for several known BCP morphologies are provided.
- Currently, the best neural network model is `LitAtrousCascadeMish` in `model/model/atr_cas_mish.py`, and it is set as default model in `train.py` and `inference_net.py`.

Citation

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