

Langevin FTS

Langevin Field-Theoretic Simulation (L-FTS) for Python

This branch includes FFTW library, which is distributed under GPL license.

Features

- SCFT and L-FTS
- AB Diblock Copolymer Melt
- Conformational Asymmetry
- Stress Calculation
- Chain Model: Continuous, Discrete
- Periodic Boundaries
- 3D, 2D and 1D
- Pseudospectral Method, Anderson Mixing
- Platforms: FFTW, MKL (CPU) and CUDA (GPU)

Dependencies

- **Linux System**
- **C++ Compiler**
Any C++ compiler that supports C++14 standard or higher. To use MKL, install Intel oneAPI toolkit.
- **CUDA Toolkit**
<https://developer.nvidia.com/cuda-toolkit>
Required for the GPU computation. If it is not installed, ask admin for its installation.
- **Anaconda**
<https://www.anaconda.com/>

Environment variables must be set so that **nvcc** and **conda** can be executed in the command line (Type **which nvcc** and **which conda** to check the installation).

Compiling

```
conda create -n lfts python=3.9 cmake=3.19 conda git pybind11 scipy
fftw openmpi
conda activate lfts
mkdir build && cd build
cmake ../
make
make test
make install
```

To use MKL, choose Intel C++ compiler with the following command.

```
cmake ../ -DCMAKE_CXX_COMPILER=icpc
```

If you encounter **segmentation fault**, type following commands.

```
ulimit -s unlimited  
export OMP_STACKSIZE=1G
```

If you want to remove all installations 🙄, type following commands.

```
conda deactivate  
conda env remove -n lfts
```

User Guide

- This is not an application but a library for SCFT and L-FTS, and you need to write your own program using Python language. It requires a little programming, but this approach provides flexibility and you can easily customize your applications.
- **A few basic SCFT and L-FTS implementations are provided in **examples** folder. Please start from those scripts: **scft/Gyroid.py**, **fts/DiscreteGyroid.py**, **fts/ContinuousLamellar.py****
- To use this library, first activate virtual environment by typing **conda activate lfts** in command line. In Python script, import the package by adding **from langevinfts import ***.
- Be aware that unit of length in this library is end-to-end chain length $aN^{(1/2)}$, not gyration of radius $a(N/6)^{(1/2)}$, where a is statistical segment length and N is polymerization index.
- The fields acting on chain are described using **per chain** language instead of **per segment** language for both SCFT and L-FTS. The same notation is used in [*Macromolecules* **2013**, 46, 8037]. If you want to obtain the same fields used in [*Polymers* **2021**, 13, 2437], multiply $1/N$ to each field.
- Use FTS in 1D and 2D only for the test. It does not have a physical meaning.
- Open-source has no warranty. Make sure that this program reproduces the results of previous FTS studies, and also produces reasonable results.
- Matlab and Python tools for visualization and renormalization are included in **tools** folder.

Developer Guide

- **Platforms**

This program is designed to run on different platforms such as FFTW, MKL and CUDA, and there is a family of classes for each platform. To produce instances of these classes for given platform **abstract factory pattern** is adopted.

- **Anderson Mixing**

It is necessary to store recent history of fields during iteration. For this purpose, it is natural to use `circular buffer` to reduce the number of array copies. If you do not want to use such data structure, please follow the code in [*Polymers* **2021**, 13, 2437]. There will be a performance loss of 5~10%.

- **Parameter Parser**

A parser is implemented using `regular expression` and `deterministic finite automaton` to read input parameters from a file. If you want to modify or improve syntax for parameter file, reimplement the parser using standard tools such as `bison` and `flex`. In Python scripts, it is no longer necessary. You can use a `yaml` or `json` file as an input parameter file instead. Using `argparse` is also a good option.

- **Python Binding**

`pybind11` is utilized to generate Python interfaces for the C++ classes.

<https://pybind11.readthedocs.io/en/stable/index.html>

References

- **CUDA Implementation**

G.K. Cheong, A. Chawla, D.C. Morse and K.D. Dorfman, Open-source code for self-consistent field theory calculations of block polymer phase behavior on graphics processing units. *Eur. Phys. J. E* **2020**, 43, 15

- **Langevin FTS**

M.W. Matsen, and T.M. Beardsley, Field-Theoretic Simulations for Block Copolymer Melts Using the Partial Saddle-Point Approximation, *Polymers* **2021**, 13, 2437

Citation

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