ETO-HEOM Tutorial

April 15, 2025

Quick Start: Workflow Summary

This section provides a summary of the full workflow. Follow these steps to run a full 2DES simulation using ETO-HEOM:

1. **Build the Solver** Modify the compiler and linking path to fit your sustem and compile the CPU and/or GPU solvers:

   ```
   cd /path/to/ETO-HEOM/
   make
   ```

2. **Initialize Working Directory** Use the setup script to create a simulation workspace:

   ```
   ./setup_cpu_job.sh JOBNAME BATHTYPE
   # or
   ./setup_gpu_job.sh JOBNAME BATHTYPE
   ```

3. **Edit Template Input File** Modify `key.key-tmpl` to set up:

   - System size, HEOM level, Hamiltonian, Disorder
   - Dipole directions, pulse settings, and time domain

4. **Generate Bath Parameters** Automatically fill in the bath section by executing:

   ```
   python3 {BATHTYPE}_ETOM.py
   # or run {BATHTYPE}_ETOM_example.ipynb
   ```

5. **Generate Input Files** Configure and run the input generation script:

   ```
   ./gen_2d_data.sh
   ```

6. **Submit PBS Jobs** Submit the simulation jobs to the cluster:

   ```
   ./submit_jobs.sh
   ```

7. **Wait Until Completion** Track your jobs with `qstat` or log files. Ensure all output files are generated in `./2d-output`.

8. **Analyze Results and Generate 2D Spectrum** After all jobs finish, analyze the results:

   ```
   python3 gen_2d_spectrum.py
   # or run gen_2d_spectrum_example.ipynb
   ```

For detailed instructions, continue reading the full tutorial below.

1 Build Instructions

Before setting up any jobs, first check the compiler configuration to fit your system. You can modify the following path in the Makefile if needed:

```
# C++ compiler and flags
CXX := g++

# CUDA compiler and flags
NVCC := /path/to/cuda-XX.X/bin/nvcc

# Include and library paths
INCLUDES := -I/usr/include
CUDA_INC := -I/path/to/cuda-XX.X/include
CUDA_LIB := -L/path/to/cuda-XX.X/lib64
```

Now you can compile the ETO-HEOM using the unified Makefile in the project root:

```
cd /path/to/ETO-HEOM/
make
```

If you only want to build one version:

```
make cpu        # Build only CPU version
make gpu        # Build only GPU version
```

To clean all build artifacts:

```
make clean      # Remove all object files and binaries
```

2 Job Setup Scripts

Before using any setup scripts, ensure that the `HOME_PATH` variable in the script files is set to the current path of your `ETO-HEOM` project.

```
HOME_PATH = /path/to/ETO-HEOM/
```

2.1 CPU Job Setup

Initializes a simulation folder for CPU-based 2DES simulations.

Usage

```
./setup_cpu_job.sh JOBNAME BATHTYPE
```

Arguments

- `JOBNAME`: Custom identifier for the simulation (e.g., `dimer`)
- `BATHTYPE`: Type of spectral density — must be one of:
  - `debye_lorentz`
  - `ohmic`
  - `superohmic`

Resulting Structure

```
CPU_JOBNAME_BATHTYPE/
|-- 2d-input/                    # Input .key files
|-- 2d-output/                   # Output .out files
|-- pbs-script/                  # PBS scripts
|   |-- pbserr/                  # PBS error logs
|   |-- pbslog/                  # PBS output logs
|-- key.key-tmpl                 # Input template file
|-- gen_2d_data.sh               # Script to generate .key files
|-- clean_2d_data.sh             # Script to clean .key and .out files
|-- gen_2d_spectrum.py           # Script to gen 2d spectrum
|-- gen_2d_spectrum_example.py   # gen 2d spectrum example
|-- BATHTYPE_ETOM.py             # ETO model for the specified bath type
|-- BATHTYPE_ETOM_example.ipynb  # ETO model example
|-- submit_jobs.sh               # Script to generate & submit PBS jobs (CPU)
|-- README.md                    # Usage of working file
```

2.2 GPU Job Setup

Initializes a simulation folder for GPU-based 2DES simulations.

Usage

./setup_gpu_job.sh JOBNAME BATHTYPE

Arguments

- `JOBNAME`: Custom identifier for the simulation (e.g., `dimer`)
- `BATHTYPE`: Type of spectral density — must be one of:
  - `debye_lorentz`
  - `ohmic`
  - `superohmic`

Resulting Structure

```
GPU_JOBNAME_BATHTYPE/
|-- 2d-input/                    # Input .key files
|-- 2d-output/                   # Output .out files
|-- pbs-script/                  # PBS scripts
|   |-- pbserr/                  # PBS error logs
|   |-- pbslog/                  # PBS output logs
|-- key.key-tmpl                 # Input template file
|-- gen_2d_data.sh               # Script to generate .key files
|-- clean_2d_data.sh             # Script to clean .key and .out files
|-- gen_2d_spectrum.py           # Script to gen 2d spectrum
|-- gen_2d_spectrum_example.py   # gen 2d spectrum example
|-- BATHTYPE_ETOM.py             # ETO model for the specified bath type
|-- BATHTYPE_ETOM_example.ipynb  # ETO model example
|-- submit_jobs.sh               # Script to generate & submit PBS jobs (GPU)
|-- README.md                    # Usage of working file
```

3 Template Input Setup

After setting up your job folder, you must modify the `key.key-tmpl` file located in the working directory. This is the template input file that defines all system-specific parameters for the simulation. Update the following sections to reflect your system configuration:

SIZE

Define the system size including the ground and first excited states.

HEOM

Define the HEOM configuration:

- Format: `(SITE_NUMBER) (TRUNCATION_LEVEL)`

HAMILTONIAN

Define the system Hamiltonian, including the ground and excited states. All values are in cm^{-1}.

DISORDER

Static disorder matrix of the Hamiltonian, also in cm^{-1}. **First row:** number of samples and random seed.

BATH

Bath information, automatically generated by `{BATHTYPE}_ETOM.py`. *You do not need to modify this section.*

DIPOLE

- First row: Number of transition dipole vectors, usually same as `(SITE_NUMBER)`
- Following rows: Dipole direction and amplitude in XYZ components

POLARIZATION

Define the polarization angles for the four pulses. *You typically do not need to modify this section.*

PULSE

- First row: Number of pulses (currently only supports 3)
- Next 3 rows: Amplitude (< 10 cm^{-1}), central time (fs), width (fs), and frequency (cm^{-1})
- Use placeholders `TAU1`, `TAU2`, `TAU3` that will be replaced automatically by `gen_2d_data.sh`

TIME

Simulation time settings:

- Format: `start time`, `end time (T_END)`, `time step`, and `sampling interval`
- `T_END` will be replaced dynamically during `gen_2d_data.sh` execution

4 Input File Generation Script

Modify `gen_2d_data.sh`

This shell script controls how the 2DES input files are generated. You may adjust the following key parameters at the top of the script:

- `t0` — Initial central time for the first pulse (in fs). It is recommanded to set `t0` to $2 * pulse\ width + 10$,ensuring it cover the pulse envelop.
- `propagate_time` — Duration to propagate after the last pulse (in fs). 600 fs is recommanded.
- `tau_step` — Step size for coherent time between different files. (in fs) 10 fs is recommanded.
- `tau_bound` — Upper and lower bounds for coherent time (in fs, `-tau_bound` to `tau_bound`)
- `T=""` — list of population times T (in fs). You can add as more values as needed.
- `input_file="key.key-tmpl"` — Path to the input template file

Script Functionality

The script will:

- Loop over values of coherent time from $-$`tau_bound` to $+$`tau_bound`
- Automatically compute `TAU1`, `TAU2`, `TAU3`, and `T_END` based on `t0`, `tau_step`, and each value in `T`
- Replace placeholders in the template file with computed values
- Output input files as: `./2d-input/key_{tau2}_{T}.key`

You can modify this script to include more population times or to adjust the pulse timing logic as needed.

Run `gen_2d_data.sh`

Once the script is configured, execute it to generate input files:

```
./gen_2d_data.sh
```

The generated input files will be saved in the `./2d-input` directory.

Clean Input Files: `clean_2d_data.sh`

To remove all generated input files and reset the input directory, use:

```
./clean_2d_data.sh
```

This will delete all files inside the `./2d-input/` folder.

5 Generate Bath Parameters

After the working directory is created using the setup script, the bath type (`BATHTYPE`) is already specified and embedded into the directory structure. You do not need to manually modify or set the bath type again.

Execute `BATHTYPE_ETOM.py`

To generate bath parameters for your simulation, run the corresponding Python script inside your working directory. This script will automatically insert bath information into `key.key-tmpl`:

```
python3 debye_lorentz_ETOM.py
```

This command calculates the ETO model parameters based on the specified spectral density type and updates the `BATH` section in the template file accordingly.

Use Jupyter Notebook: `BATHTYPE_ETOM_example.ipynb`

You may also use the accompanying Jupyter notebook to interactively generate bath parameters:

- Launch the notebook with:

  ```
  jupyter notebook debye_lorentz_ETOM_example.ipynb
  ```

- Execute the notebook cells step-by-step to calculate and visualize the bath parameters.

This method is recommended if you wish to explore or fine-tune the bath model before applying it.

6 Submit PBS Jobs

After generating all input files, execute the submission script to generate and submit PBS job scripts.

Run `submit_jobs.sh`

From your working directory, run:

```
./submit_jobs.sh
```

This script will automatically:

- Create PBS job scripts for the CPU or GPU version based on your setup
- Divide the τ_2 range into manageable chunks (CPU only)
- Submit the generated PBS scripts using `qsub`

Important Parameters to Check

Before execution, ensure the following variables in `submit_jobs.sh` are properly set:

- `HOME_PATH` — Automatically set by the setup script
- `JOBNAME` — Automatically set by the setup script
- `CPU_2DES` or `GPU_2DES` — Path to your binary executable
- `INPUT_DIR` — Should be `./2d-input`
- `OUTPUT_DIR` — Should be `./2d-output`
- `ERR_DIR`, `LOG_DIR` — Should point to subfolders inside `./pbs-script/`
- `TLIST` — Array of population times T (e.g., `0` or `(0 100 200)`)
- `START_TAU`, `END_TAU`, `STEP_TAU` — Define the τ_2 scan range

PBS Script Generation: CPU Version

For CPU-based simulations, the script:

- Splits the τ_2 scan range into equal parts (`NUM_SCRIPTS`)
- Generates one PBS script per chunk under `pbs-script/`
- Submits them using `qsub`

Each PBS job:

- Executes up to 11 parallel CPU processes (`ppn=11`)
- Launches `CPU_2DES` for each `key_{TAU}_{T}.key` input
- Outputs result to `2d-output/out_{TAU}_{T}.out`

PBS Script Generation: GPU Version

For GPU-based simulations, the script:

- Generates a single PBS job script under `pbs-script/`
- Sets GPU-specific resources (e.g., `nodes=gpu02`, CUDA environment variables)

The job:

- Executes `GPU_2DES` for each input file
- Logs results and errors in `pbs-script/pbslog` and `pbs-script/pbserr`

Output

Submitted PBS jobs will automatically:

- Read input files from `./2d-input`
- Write output files to `./2d-output`
- Write logs to `./pbs-script/pbslog/`
- Write errors to `./pbs-script/pbserr/`

Use `qstat`, `tail -f`, or PBS web portal to monitor job progress.

7 Post-Processing: Generate 2D Spectrum

After all PBS jobs have completed and the output files are available in the `./2d-output` directory, you can generate the 2D spectrum by running the analysis script.

Execute `gen_2d_spectrum.py`

This script will parse the output files in `2d-output/`, perform 2D Fourier transforms over coherent times, and assemble the final 2D spectrum data.

To run the script:

```
python3 gen_2d_spectrum.py
```

The resulting 2D spectrum will be saved in a format specified within the script. (e.g., png, svg, eps, etc.)

Use Jupyter Notebook: `gen_2d_spectrum_example.ipynb`

Alternatively, if you prefer interactive analysis or visualization, you can use the accompanying Jupyter notebook:

- Launch the notebook with:

  ```
  jupyter notebook gen_2d_spectrum_example.ipynb
  ```

- Follow the instructions and code cells to:
  - Load output data
  - Apply any window or filtering
  - Compute the 2D Fourier transform
  - Plot the 2D spectrum using Matplotlib

This notebook is useful for adjusting analysis parameters and visually verifying results.