Reference Manual: Simulation of Highly Multidimensional Chemical Dynamics with Low-Rank Tensor-Train Chebyshev Quantum Dynamics

Micheline B. Soley,[†,‡] Paul Bergold,[¶] Alex Gorodetsky,[§] and Victor S. Batista[*,†,‡,‖]

†*Yale Quantum Institute, Yale University, P.O. Box 208334, New Haven, CT, 06520-8263, USA*

‡*Department of Chemistry, Yale University, P.O. Box 208107, New Haven, CT, 06520, USA*

¶*Zentrum Mathematik, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany*

§*Department of Aerospace Engineering, University of Michigan, 1320 Beal Avenue Ann Arbor, MI 48109-2140, USA*

‖*Energy Sciences Institute, Yale University, P.O. Box 27394, West Haven, CT, 06516-7394, USA*

E-mail: victor.batista@yale.edu

Introduction

We present tensor-train (TT)-Chebyshev and function-train (FT)-Chebyshev codes for the simulation of exact molecular dynamics in high dimensionality. These codes employ the low-rank tensor train decomposition (a relative of matrix product states) and their continuous analogue. An example is given for simulation of a fifty-dimensional model of hydrogen bonding in adenine-thymine DNA base pairs.

Dependencies

The tensor-train code requires the following:

- MATLAB (tested for R2018a and R2021a)
- H-Tucker Tensor Toolbox (path must be included in MATLAB code at runtime)
- TT-Toolbox (path must be included in MATLAB code at runtime)

The function-train code requires the following:

- C compiler
- CMake
- Compressed Continuous Computation (C3) library (which requires BLAS, LAPACK, and SWIG-python [optional])

Comparison of the tensor-train and function-train codes is facilitated with the following:

- Gnuplot

Tensor Train Code Instructions

Code for simulation of TT-Chebyshev dynamics is contained in the folder `TensorTrain`. Tensor-train split operator Fourier transform (TT-SOFT) dynamics are also performed for comparison.

Dynamics

Tensor train Chebyshev propagation is performed by running `tt_CHEBSOFT.m` in MATLAB.

Propagation parameters are defined as follows:

- `d` – Dimensionality
- `ns` – Number of time steps
- `dump` – Number of time steps after which the wavefunction is saved to file
- `eps` – Precision of tensor train rounding
- `ceps` – Precision of tensor train rounding for exponential
- `rmax` – Maximum tensor train rank
- `alpha` – Harmonic oscillator eigenstate parameter $\alpha = m\omega/\hbar$, which determines the width of the initial wavefunction
- `x0` and `x0array` – Position parameters for the initial wavefunction
- `p0` – Initial momentum
- $\mathtt{nx} = 2^{\mathtt{q}}$ – Number of grid divisions `nx` in position space as specified by the integer number of quantics `q`
- `np` – Number of grid divisions in momentum space
- `L` – Length of the simulation grid in each physical dimension
- `dx` – Position grid spacing
- `dp` – Momentum grid spacing
- `tau` – Time step
- `Npoly` – Number of Chebyshev polynomials

- `m` – Mass in each physical dimension (currently only implemented for unit mass)
- `x` – Position grid in each direction
- `p` – Momentum grid in each direction
- `verticalscale` – Vertical stretch parameter for DNA potential
- `gamma` – Coupling parameter for DNA potential

The code visualizes the $x_1 - x_2$ slice of the potential energy surface for two-, three-, and four-dimensional potential energy surfaces. Figures comparing TT-Chebyshev and TT-SOFT are printed to screen: the autocorrelation function (real and imaginary parts) at each time step; the probability density every `dump` time steps; and, after the run is complete, the norm at each time step. In addition, the following output files are created:

- `resultofstep` $**$`.mat` – Saved MATLAB variables at time step $**+$`1`
- `resultofstepfinal.mat` – Saved MATLAB variables after propagation

Example output images and files are given in the folder entitled `ExampleOutput`.

The code requires H-Tucker Tensor Toolbox and TT-Toolbox to be included in the path specified at the beginning of the `tt_CHEBSOFT.m` file. The potential energy surface can be changed by revising the definition of the tensor train `PE_tt` and the minimal and maximal energies `Emin` and `Emax`. An alternative initial wavefunction can be defined in the function in file `psi0.m`.

Conversion of Output Data Files

To facilitate the comparison of tensor train and function train data, after propagation is complete, the `GnuplotConversion.m` code is run in MATLAB given the following input data files held in the same folder:

- `resultofstepfinal.mat` – Saved variables at final time

- `resultofstep` $**$`.mat` – Saved variables at time step $**$

Example output files are provided in folder `ExampleOutput`.

The code produces the following data files in Gnuplot format:

- `norm.dat` – Norm at each time (column 1) for TT-SOFT (column 2) and TT-Chebyshev (column 3)
- `autocorrelation.dat` – Autocorrelation at each time (column 1) for TT-SOFT (real part in column 2 and imaginary part in column 3) and TT-Chebyshev (real part in column 4 and imaginary part in column 5)
- `wave` $**$`.dat` – Absolute value squared of the density in directions x_0 (column 1) and x_1 (column 2) at chosen time step `ii`$=**+1$ for TT-SOFT (column 3) and TT-Chebyshev (column 4), where `ii`$=**+1$ corresponds to the `ii`$^{\text{th}}$ element in array `numlist`

The path at the beginning of the file must correspond to the location of the TT-Tensor Toolbox library. The conversion code can be implemented for visualization of other time steps through modification of the array `numlist`, which contains the number of each time step under consideration. The conversion code is currently implemented in fifty dimensions, as slices of the density are computed as follows:

```
v1=reshape(abs(full(G_tt(:,:,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2))).^2,[nx nx]);
v3=reshape(abs(full(Gc_tt(:,:,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
```

```
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,...
nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2,nx/2))).^2,[nx nx]);
```

where the 48 `nx/2` entries refer to the index of the fixed position of the bath modes. The fixed position of the bath modes can be changed by replacing the `nx/2` with the index of choice, and the calculation can be updated for B bath modes by replacing the 48 `nx/2` entries with B `nx/2` entries.

Visualization of these data files with Gnuplot is discussed in "Tensor Train/Function Train Comparison Instructions" at the end of this reference manual.

Function Train Code Instructions

Code for function-train propagation is included in the folder `FunctionTrain`. Details on installation of dependences and troubleshooting are included in the `readme.txt` file contained in the folder.

The program is compiled with the commands:

```
cmake .
make
```

The program is then run as follows:

```
./ft_cheb
```

The program has been tested for macOS Mojave and Big Sur. The following steps can be employed for troubleshooting if compilation or runtime errors occur.

- If errors concerning missing `c3axpy` arguments in `2dpro.c` appear when `cmake` is run and if `epsilon` is zero, add `NULL` as the missing argument in all instances.

- If the file `Accelerate` is not found by the compiler, but `Accelerate.tbd` exists in any folder, copy the `Accelerate.tbd` file into the same folder and rename it `Accelerate`.

- When updating the `cmake` file, remove outdated cmake output as follows:
  `rm` $-$ `r./CMakeFilesCMakeCache.txt`.

- If make yields an error that `c3/array.h` is missing, $-$`lc3` cannot be found, etc., try the following:

  - Ensure that `c3` is downloaded. Then update the version of `cmake` as follows:
    `cmake_minimum_required(VERSION 3.0)`
  - Change the final argument of `target_link_libraries` to the following: `${c3lib}`.
  - Update the location of the C3 library `lib` files in `CMakeLists.txt` inside the `if APPLE loop` as follows:

    ```
    set(CMAKE_LIBRARY_PATH ${CMAKE_LIBRARY_PATH} /usr/local/lib)
    ```

  - Update the location of the C3 library `lib` files in `CMakeLists.txt` inside the `if APPLE loop` as follows:

    ```
    find_library(c3lib c3 PATHS /usr/local/lib)
    include_directories(/usr/local/include)
    ```

Propagation is controlled according to the following parameters:

- `NCHEB` – Number of Chebyshev polynomials
- `NE` – Number of grid points in each dimension
- `NX` – Total number of grid points in the $x_1 - x_2$ grid (*i.e.*, $\mathtt{NE}^2$)
- `DIM` – Number of dimensions
- `NSTEPS` – Number of propagation steps
- `NDUMP` – Number of time steps after which the wavefunction is saved to file
- `dis` – Initial displacement parameter

- `dt` – Time step
- `m` – Mass
- `lb` – Minimum value of the position-space grid
- `ub` – Maximum value of the position-space grid

The tensor train approximation can be modulated by changing:

- `init_rank` – Initial rank
- `round_eps` – Rounding accuracy parameter

and the final arguments of the functions:

- `function_train_round_maxrank_all` – Rounding maximum rank
- `c3approx_set_adapt_kickrank` – Cross approximation kick rank
- `c3approx_set_cross_maxiter` – Cross approximation maximum iterations
- `c3approx_set_cross_tol` – Cross approximation tolerance parameter
- `c3approx_set_round_tol` – Cross approximation rounding tolerance parameter
- `c3approx_set_adapt_maxrank_all` – Cross approximation maximum rank

The program outputs the files:

- `timings` – Time required for each propagation step
- `xi` – Autocorrelation function at each time step (real and imaginary parts)
- `norm` – Norm at each time step
- `wave.*` – Probability density at the $*^{\text{th}}$ multiple of printing frequency `NDUMP`

Example output is included in folder `ExampleOutput`. The output can be visualized with the scripts described in the final section of this reference manual.

The initial wavefunction can be modified in function `GS` and `GSfix`, and the potential can be modified in function `Vpot` (which requires an update of the maximum and minimum potential energy values `Vmax` and `Vmin`, respectively).

Tensor Train/Function Train Comparison Instructions

The results of the TT- and FT-Chebyshev simulations can be visualized with the Gnuplot scripts in folder `Comparisons`. For visualization, the TT-Chebyshev program data files `wave.*.dat` and `autocorrelation.dat` must be included in the folder `TensorTrainData`, and the FT-Chebyshev program data files `wave.*` and `xi` must be included in the folder `FunctionTrainData`.

The probability density comparison is created with the following command:

```
gnuplot  wavefunctionsnapshots.gpt
```

which produces the files `wavefunctionsnapshots.eps` and `wavefunctionsnapshots.pdf`.

The autocorrelation function comparison is created with the following command:

```
gnuplot  autocorrelationcompare.gpt
```

which produces the files `autocorrelationcompare.eps` and `autocorrelationcompare.pdf`.

Example output is given in folder `ExampleOutput` for verification.