PyThermal Documentation

Release v2.2.0

Dinesh Pinto, St. Stephen's College

Jul 09, 2016

CONTENTS

CHAPTER

ONE

MAIN

class main.**System**(*initial_values*, *lattice_a*, *lattice_b*)

__init__(*initial_values*, *lattice_a*, *lattice_b*)
Generates metadata about the system.

Parameters

- **initial_values** – List of initial state values of the system
- **lattice_a** – Lattice sites in A
- **lattice_b** – Lattice sites in B

__weakref__
list of weak references to the object (if defined)

check_existence(*names*)
Checks whether variables exists on hard disk. Generally deprecated for try-except statements.

Parameters **names** – list of names of variables

Returns Boolean list whether files exists on hard disk

check_system()
Runs checks to make sure all inputs are valid.

:raises Value errors for invalid inputs

folder_path
The naming convection is as follows: P[Total no. of particles] D[Dimensionality of lattice] A[No. of sites in A] B[No. of sites in B]

Returns Path for storing program output

static plotting_metadata()
Stores metadata for MatPlotLib plots.

Returns Filename of images

Returns Image titles

Returns y axis labels

Returns x axis labels

Returns y axis limits

main.**main_states**(*initial_values*, *chosen_eigenstates*, *lattice_a*, *lattice_b*)
Contains function calls to determine the density matrix of a subsystem in its energy basis and compare the

diagonal/off-diagonal elements. Uses time() module to time execution of various functions and output to standard output.

Parameters

- **chosen_eigenstates** – Eigenstates for which to compute DM's
- **initial_values** – List of initial values for system initial_values = [

  Total no. of particles(nop), Dimension of lattice(ndims)]
- **lattice_a** – List of sites in A
- **lattice_b** – List of sites in B

Returns True if execution successful

main.**main_time**(*initial_values*, *chosen_eigenstate*, *t_initial*, *t_final*, *t_steps*, *lattice_a*, *lattice_b*)

Contains functions to time evolve the entire system (both sub-lattices) starting in an initial state where all particles are in one sub-lattice. Possible issues.

Parameters

- **initial_values** – List of initial values for system initial_values = [

  Total no. of particles(nop), Dimension of lattice(ndims)]
- **chosen_eigenstate** – Initial state for time evolution
- **t_initial** – Starting time
- **t_final** – Ending time
- **t_steps** – No. of steps
- **lattice_a** – List of sites in A
- **lattice_b** – List of sites in B

Returns True if execution successful

CHAPTER

TWO

ROUTINES

routines.**position_states**(*lat*, *nop*, *del_pos=None*)

Returns position states for a given lattice and number of particles. Parameter del_pos can be used to delete lattice sites.

Parameters

- **lat** – Array of lattice sites
- **nop** – Nop of particles in lattice
- **del_pos** – Lattice sites to delete

Returns Positions states

Returns Total no. of states

routines.**distribute**(*n_items*, *n_processes*, *i*)

Defines a starting and stopping point for a particular task to be allocated to a process. Returns a (start, stop) tuple.

Parameters

- **n_items** – Total no. of items
- **n_processes** – Total no. of processes
- **i** – Process no. (not same as PID)

Returns Start & Stop point index in no. of items

routines.**hamiltonian_parallel**(*lattice*, *ndims*, *nop*)

Wrapper for _hamiltonian. Creates multiple processes, each of which calls _hamiltonian simultaneously. Defines a global 'ham', which is a multiprocessing array interfaced with ctypes and reshaped to generate an empty (filled with zeros) hamiltonian array.

Parameters

- **lattice** – Lattice used
- **ndims** – Dimensionality of lattice
- **nop** – No. of particles

Returns Hamiltonian matrix

routines.**diagonalize**(*h*)

Calculates eigenvectors and eigenvalues used Pade algorithm (see SciPy documentation).

Control threads using OpenMP. Link to lower level OpenBLAS (Basic Linear Algebra Subroutines) written in Fortran for parallel processing.

Parameters **h** – Matrix

Returns Real array of eigenvalues

Returns Complex array of eigenvectors

routines.**ncr**(*n*, *r*)
No. of combinations of k items taken from n items.

Parameters

- **n** – Total no. of items
- **r** – No. of items chosen

Returns Total no. of combinations

routines.**sum_ncr**(*n*, *k*)
Calculates nC0 + nC1 + ... + nCr.

Parameters

- **n** – Total no. of items
- **k** – No. of items chosen

Returns Sum of combinations

routines.**relabel**(*e_states*, *nop*, *nol_b*, *lat_a*)
The state is |psi>AB = Sigma_{i, n, mu} a_i_n mu |i_nmu> where the product state is written as |i_nmu>. Refer to "The Density Matrix using Relabelled States.pdf"

Parameters

- **lat_a** – Sub-lattice B
- **e_states** – Eigenstates
- **nop** – No. of particles
- **nol_b** – No. of lattice sites in B

Returns Array of relabelled states

routines.**rho_a_pbasis**(*label*, *e_vec*, *nos*, *nol_a*, *nop*)
Calculates density matrix for sub-lattice A. Error checks density matrix (trace should be 1.0). Warning raised if trace differs by 1(+-)0.1.

Parameters

- **label** – Relabelled states
- **e_vec** – Eigenvectors
- **nos** – No. of states
- **nol_a** – No. of lattice sites in A
- **nop** – No. of particles

Returns Density matrix of sub-lattice A

routines.**rho_b_pbasis**(*label*, *e_vec*, *nos*, *nol_b*, *nop*)
Calculates density matrix for sub-lattice B. Error checks density matrix (trace should be 1.0). Warning raised if trace differs by 1(+-)0.1.

Parameters

- **label** – Relabelled states

- **e_vec** – Eigenvectors
- **nos** – No. of states
- **nol_b** – No. of lattice sites in B
- **nop** – No. of particles

Returns Density matrix of sub-lattice B

routines.**h_block_diagonal**(*lat_b*, *n_dim*, *nop*)
Creates a block diagonal matrix containing the hamiltonian (for various no. of particles) of B placed in blocks along the diagonal.

Parameters

- **lat_b** – Lattice sites in B
- **n_dim** – No. of dimensions
- **nop** – No. of particles

Returns Block diagonal matrix

routines.**transform_basis**(*rho_pbasis*, *e_vecs_bd*)
Transforms DM from position basis to DM in energy basis.

Parameters

- **rho_pbasis** – Rho in position basis
- **e_vecs_bd** – Eigenvectors of block diagonal hamiltonian

Returns Rho in energy basis

routines.**naive_thermal**(*rho*)
Compare maximum diagonal and off diagonal terms in density matrix. Note: Makes copy of DM (bypass will destroy original DM)

Parameters **rho** – Density matrix in energy basis

Returns Max diagonal element

Returns Max off-diagonal element

routines.**initial_sublattice_state**(*e_vec*, *label*, *nos*, *nop*, *e_vec_num*)
Returns a normalized initial state with all particles in one sub-lattice. Control which sub-lattice is used by passing appropriate eigenvectors (i.e. eigenvectors of the chosen sub-lattice) as an argument.

Parameters

- **e_vec** – Eigenvectors of either sub-lattice
- **label** – Array of relabelled states
- **nos** – No. of states
- **nop** – No. of particles
- **e_vec_num** – Initial eigenvector chosen

Returns Normalized initial state

routines.**trace_squared**(*rho*)
Calculate the trace of the square of the density matrix.

Parameters **rho** – Density matrix

Returns Trace of the square of the density matrix

routines.**vn_entropy_b**(*psi_t*, *label*, *nos*, *nol_b*, *nop*)
Calculates Von-Neumann entropy as S = - tr(rho * ln(rho)). Also calculates trace of square of density matrix (measure of entanglement). Uses a filter to suppress 'WARNING: The logm input matrix may be nearly singular'. Wraps loop in tqdm for progress bar.

Parameters

- **psi_t** – Psi(t)
- **label** – Relabelled states
- **nos** – No. of states
- **nol_b** – No. of lattice sites in B
- **nop** – No. of particles

Returns Real Von-Neumann entropy

Returns Trace of density matrix of B

routines.**time_evolution**(*psi_0*, *h*, *nos*, *timesteps*)
Psi evolved as |Psi(t)> = exp(-i * h * t)|Psi(0)>. Uses matrix exponential for computation.

Parameters

- **psi_0** – Initial state
- **h** – Hamiltonian matrix
- **nos** – No. of states
- **timesteps** – Array of times

Returns Array of Psi(t)

routines.**avg_particles**(*psi_t*, *timesteps*, *labels*, *nop*)
Calculates the average number of particles in sub-lattices A and B.

Parameters

- **psi_t** – Array of psi at various times
- **timesteps** – Array of times
- **labels** – Relabelled states
- **nop** – Total no. of particles

Returns Avg. particles in sub-lattice A

Returns Avg. particles in sub-lattice B

CHAPTER

THREE

OUTPUT

output.**status**(*time_taken=0.0*)

Prints current status (execution time) of program execution. Note: Times returned from here are not true measures of algorithm speed. For rigorous function time testing use the timeit module.

Parameters **time_taken** – Block execution time

output.**warning**(**objects*)

Handles non-fatal warnings. Output to stderr.

Parameters **objects** – Objects

output.**write_file**(*path*, *filename*, *data=None*, *fmt='%.18e'*)

Checks if output directory exists, if not, creates it. Writes arrays/lists to the disk. Performs IO using NumPy. Possible switch in future to the more robust Pandas (dependant code must be unaffected).

Parameters

- **path** – Folder path to write to
- **filename** – Name of file (include extension)
- **data** – Data to be written
- **fmt** – Format specifier

output.**write_image**(*path*, *filename*)

Checks if output directory exists, if not, creates it. Then writes to disk. Writes images to the disk. Performs IO using MatPlotLib.

Parameters

- **path** – Folder path to write to
- **filename** – Name of file (including extension)

output.**read_file**(*path*, *filename*, *dtype=<class 'numpy.float64'>*)

Reads data from files stored locally.Performs IO using NumPy. Possible switch in future to the more robust Pandas (dependant code must be unaffected).

Parameters

- **path** – Path to folder
- **filename** – Name of file
- **dtype** – Data type of file

Returns Array of read data

Raise IOError if file not found

output.**plot_write**(*x*, *y*, *title=None*, *y_label=None*, *x_label=None*, *y_limit=None*, *path=None*, *filename=None*, *checkbox=None*)

Generate graphs using MatPlotLib. Uses write_image() to save to hard disk. Checkbox can be used to control plot display during execution.

Parameters

- **x** – x axis
- **y** – y axis
- **title** – Graph title
- **y_label** – y axis label
- **x_label** – x axis label
- **y_limit** – y axis limits
- **path** – Path for saving file
- **filename** – Name of file
- **checkbox** – Show images during execution(1) or not(0)

CHAPTER

FOUR

ABOUT

about.**about**(*test=True*)

Checks dependencies and return version numbers for PyThermal. Tests NumPy and SciPy. Checks OpenBLAS for Numpy.

Parameters **test** – Run tests for NumPy and SciPy

CHAPTER

FIVE

GUI

gui.**fetch**(*values*, *values2*)
Prints data stored in entries for debugging purposes.

Parameters

- **values** – List storing initial values
- **values2** – List storing sub-lattices A and B

gui.**graphical_interface**(*base*)
Base layout for text fields and their labels. The order in which they are placed matters. This may be addressed in a future update.

Parameters **base** – Root

Returns List of entries

gui.**execute**(*initial_values*, *optional_values*)
Calls main() from main.py.

Parameters

- **optional_values** – List of optional values
- **initial_values** – List of initial values

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

Symbols

A

C

D

E

F

G

H

I

M

N

O

P

R

S

T

V

W