schrodinger Documentation

Release 1.0

Tammo van der Heide and Alexander Jochim

Sep 13, 2018

CONTENTS

1 Introduction **1**

2 Input and Output **3**

3 Starting Options **5**
- 3.1 Optional Parameters . . . 5

4 Modules **7**
- 4.1 schrodinger.py . . . 7
- 4.2 schrodinger_io.py . . . 7
- 4.3 schrodinger_solver.py . . . 7
- 4.4 schrodinger_visualize.py . . . 8
- 4.5 test_schrodinger_solver.py . . . 8

5 Scientific Notes **11**

Python Module Index **13**

CHAPTER

ONE

INTRODUCTION

A program that solves the 1d schrodinger equation. It reads a file *schrodinger.inp* that specifies a given potential and calculates energies, wavefunctions, expectation values and standard deviation. All data is plotted afterwards and saved to *schrodinger.pdf*.

CHAPTER

TWO

INPUT AND OUTPUT

The name of the input file must be called **schrodinger.inp**. By default *schrodinger* assumes it is located in the same directory as itself. Other directory can be accessed via the *-d* starting option. The input file must have the following structure:

```
4.0            # Mass
-5.0 5.0 1999  # xMin xMax nPoint
1 5            # first and last eigenvalue to include in the output
polynomial     # interpolation type
3              # # nr. of interpolation points and xy declarations
-1.0 0.5
0.0 0.0
1.0 0.5
```

You can use as many interpolation points as you want. The number has to fit the number of xy declarations afterwards. Possible interpolation types are *linear*, *csplines* and *polynomial*. *nPoints* is the number of discrete points that are used for calculating.

All calculations are saved in the following files. They are used to plot the data and can be used for further work:

- *potential.dat*

  interpolated potential in XY-Format:

  ```
  x1 V(x1)
  x2 V(x2)
  :    :
  ```

- *energies.dat*

  calculated eigenvalues

  ```
  E1
  E2
  E3
  :
  ```

- *wavefuncs.dat*

  calculated wavefunctions in NXY-Format

  ```
  x1 wf1(x1) wf2(x1) wf3(x1) ...
  x2 wf1(x2) wf2(x2) wf3(x2) ...
  :
  ```

- *expvalues.dat*

expectation values and standard deviation

```
exp_val1 st_dev1
exp_val2 st_dev2
:
```

CHAPTER

THREE

STARTING OPTIONS

Starting options (optional parameters) for the main module *schrodinger.py*. Use the long form as *--name* or use the short version showed below.

3.1 Optional Parameters

- directory: -d [path]

  Used to specify the path of the input file *schrodinger.inp*

- split: -s

  Splitting the wavefunctions, expectation values and standard deviations in the plot for a better view.

- stretch: -st [float]

  Multiplies the wavefunctions with a factor for a better view.

- markersize: -m [float]

  Changes the markersize of the expectation values and standard deviation.

CHAPTER

FOUR

MODULES

4.1 schrodinger.py

Main module that can be used with the starting options (optional parameters) above.

4.2 schrodinger_io.py

Module that reads the input data, processed by the schrodinger solver

schrodinger_io.**output**(*data*)
Write calculated data to files

Parameters **data** (*dict*) – dictionary with calculated data from solve1d

schrodinger_io.**read_input**(*file*)
Read given parameters and potential data of schrodinger.inp

Parameters **file** (*str*) – path to input file schrodinger.inp

Returns obtained input from file

Return type obtained_input (dict)

Raises OSError – if input file is not present or has wrong permissions/name

4.3 schrodinger_solver.py

Module that solves the onedimensional Schrodinger equation for arbitrary potentials

schrodinger_solver.**interpolate**(*obtained_input*)
Interpolation routine, that interpolates the given potential data with a specified interpolation method

Parameters **obtained_input** (*dict*) – obtained input of schrodinger_io.read_input()

Returns contains interpolated potential

Return type interppot (numpy array)

schrodinger_solver.**solve1d**(*obtained_input*, *pot*)
Solves the discretized 1D schrodinger equation

Parameters

- **obtained_input** (*dict*) – obtained input of schrodinger_io.read_input()

- **pot** (*numpy array*) – contains potential

Returns calculated data; potential, energies, wavefuncs, expvalues

Return type data (dict)

4.4 schrodinger_visualize.py

Visualization module for the schrodinger_solver

schrodinger_visualize.**show**(*stretchfactor=1*, *split=False*, *markersize=10*)

Visualizes the output of the solver function and saves to schrodinger.pdf

Parameters

- **stretchfactor** (*float*) – stretches the wavefunctions in y-direction
- **split** (*bool*) – creates offset to wavefunction, expectationsvalues
- **standard deviation if set to 'True'** (*and*) –
- **markersize** (*float*) – changes markersize of expectations values and standard deviation

Raises OSError – if input file is not present or has wrong permissions/name

4.5 test_schrodinger_solver.py

Contains routines to test the solvers modules

test_schrodinger_solver.**test_compare**(*testname_compare*)

Tests the constant functioning of the code by comparing previously calculated energy-levels and potential data with the current output of the solver

Parameters **testname_compare** (*str*) – to create path of reference and input files

Asserting: test_compare_assert (bool): if True, test passes

test_schrodinger_solver.**test_energies**(*testname_energie*)

Tests the energy-levels of schrodinger_solver by comparing thoose with exact results or groundstates calculated on paper

Parameters

- **testname_energie** (*str*) – to create path of reference and input files
- **energie data/parameters** (*containing*) –

Asserting: test_energies_assert (bool): if True, test passes

test_schrodinger_solver.**test_interpolation**(*testname_interp*)

Tests the interpolation of schrodinger_solver by comparing the interpolated potential with the given XY data

Parameters

- **testname_interp** (*str*) – to create path of reference and input files
- **potential data/parameters** (*containing*) –

Asserting: test_interp (bool): if True, test passes

CHAPTER

FIVE

SCIENTIFIC NOTES

All calculations are numeric! The potential defined by discrete reference points inside the input file is interpolated using numerical algorithms (e.g., finite differences and integrals as Riemann sums). The constructions of a tridiagonal matrix allows solving the time independent schrodinger equation at discrete points as an eigenvalue problem. This results in inaccuracies both due to discretization errors and due to rounding errors in the floating-point number calculation.

The probability of the particle to be inside the given x-boundaries is treated as 100%, so the probablity outside has to be 0. Therefore the problem has aquivalent additonal infinite high potential walls at xmin and xmax. In case of non-decreasing wavefunction amplitudes outside the potential boundaries, reconsideration of the calculated solutions is recommended.

PYTHON MODULE INDEX

s

schrodinger_io, 7
schrodinger_solver, 7
schrodinger_visualize, 8

t

test_schrodinger_solver, 8