LatticeQM.jl

Release alpha

Tobias M. R. Wolf

Jan 03, 2020

CONTENTS:

1 Getting started **1**
- 1.1 Installation . . . 1
- 1.2 Usage . . . 1
- 1.3 Example code . . . 1

2 Tutorial **3**
- 2.1 Creating a lattice . . . 3
- 2.2 Getting a tight-binding Hamiltonian . . . 3
- 2.3 Bandstructure calculations . . . 3

3 Examples **5**

4 Packages **7**

5 Indices and tables **9**

CHAPTER

ONE

GETTING STARTED

1.1 Installation

- Make sure Julia is installed and is at least version 1.2.
- Use Julia's builtin package manager *Pkg* to add the git repository LatticeQM.jl. Note that at the time of writing, the repository is private and requires login credentials. Just contact me if you want access.

  Option 1: Start an interactive julia session (*REPL*), hit the `]` key and run:

  ```
  add https://gitlab.ethz.ch/wolft/LatticeQM.jl
  ```

  Once the installation is done, leave *Pkg mode* with the backspace key.

  Option 2: Execute the julia code

  ```
  using Pkg
  Pkg.add("https://gitlab.ethz.ch/wolft/LatticeQM.jl.git")
  ```

1.2 Usage

```
using LatticeQM
```

Note that the first import may take quite long (due to compilation). If you get error messages about third-party packages try to install them manually through the Pkg package manager. If it ran successfully, then you are good to go.

First-time users may want to check out the tutorials and examples.

1.3 Example code

```
# Imports
using LinearAlgebra, Plots
pyplot() # use the matplotlib backend for plots
using LatticeQM

# Set up lattice
lat = Geometries2D.honeycomb()
```

(continues on next page)

(continued from previous page)

```
# Get nearest-neighbor hops in the honeycomb lattice
hops = Materials.graphene(lat; mode=:nospin) # or mode=:spinhalf for spin-1/2

# Compile Bloch Hamiltonian
h = get_bloch(hops)

# Path in k-space
ks = Structure.get_kpath(lat; num_points=200)

# Get bandstructure
bands = get_bands(h, ks)

save(bands, "bands_example.h5") # save raw data
p = plot(bands, size=(330,240)) # plot data
# display(p) # might need to be uncommented in non-interactive mode

savefig("bands_example.pdf") # save the figure
```

CHAPTER

TWO

TUTORIAL

The typical workflow is as follows:

1. Import the package via `using LatticeQM`
2. Create or load a predefined lattice object
3. Obtain a tight-binding Hamiltonian from the lattice object
4. Use exact diagonalization to calculate bandstructure, linear response coefficients and topological invariants

In what follows, we shall elaborate each point. Afterwards, you will have a good idea of how to use this package.

2.1 Creating a lattice

Text here.

2.2 Getting a tight-binding Hamiltonian

Text here.

2.3 Bandstructure calculations

Text here.

CHAPTER

THREE

EXAMPLES

This where examples will go.

CHAPTER

FOUR

PACKAGES

`LatticeQM` provides several subpackages:

- `Structure` contains several types and methods to define and manipulate lattice vectors, unit cells and paths in reciprocal space. It is an abstract class for many problems.
- `Geometries2D` contains predefined two-dimensional lattice geometries (so far mostly honeycomb multilayers). It is a collection of tested examples.
- `TightBinding` contains types and methods to define real-space hopping Hamiltonians (non-interacting), in particular via given `Lattice` objects. A tight-binding Hamiltonian can then be converted in to a Bloch Hamitlonian matrix. It is an abstract class intended to fit many problems.
- `Materials` is a collection of physical examples and general modifiers to get tight-binding operators, e.g. for graphene, or zeeman fields, spin-orbit coupling etc.
- `BlochTools` contains methods to perform, save and display results from exact diagonalization, such as bandstructure, optical conductivity, topological invariants, etc.

There is another subpackage `KPM.jl` implementing the Kernel Polynomial Method. It is tested and working, but for now neither integrated nor properly documented.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search