Ground State of Spin-Orbit Coupled Atoms in a Two-Dimensional Lattice

Nathan Ngo and Supervisor: David Feder
University of Calgary
(Dated: August 2023)

I. INTRODUCTION

The study of the ground state properties of spin-orbit coupled atoms in two-dimensional lattices is important for understanding quantum systems with both spin and spatial effects. These systems are useful for applications in quantum information and materials with unique quantum properties.

This work uses numerical simulations to model spin-orbit coupled atoms in lattices. Adjacency matrices are used to represent tunneling under different conditions, including one- and two-dimensional configurations, boundary conditions, and spin effects. These models calculate eigenvalues and eigenvectors to provide insights into the system's physics.

The following sections describe the methods used, including how adjacency matrices are generated for various tunneling conditions and how Hamiltonian terms are added to model spin-orbit coupling. This approach provides a framework for studying the ground state behavior in these systems.

II. CODE [1]

A. What does this code do?

This code generates the 1D cycle or path adjacency matrix without any spinning. For a path, it means that a hard-wall boundary condition is used. For a cycle, it means that a periodic boundary condition is used. For every possible (illustrated in Fig. 1 and Fig. 2), the corresponding matrix element equals to 1.

For instance, since the particle can tunnel from site 1 to site 2 and vice versa, the corresponding matrix elements [1][2] and [2][1] equals to 1.

B. How to use this code?

Firstly, enter the number of nodes, then enter 0 if you want to generate a cycle adjacency matrix, enter 1 if you want to generate a path adjacency matrix. The generator then generates the corresponding matrix, its eigenvalues and eigenvectors. Note that the eigenvectors are read vertically.

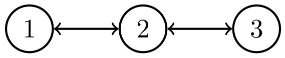

FIG. 1. A 1D path of 3 nodes

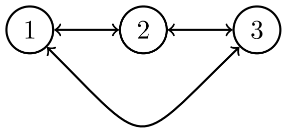

FIG. 2. A 1D cycle of 3 nodes

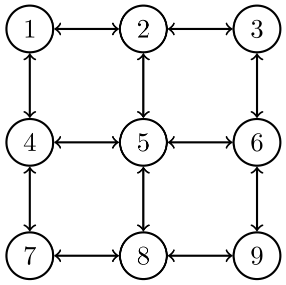

FIG. 3. A 2D path of 9 nodes

III. CODE [2]

A. What does this code do?

This code generates the 2D cycle or path adjacency matrix without any spin. In Fig. 3, the configuration of a 3×3 lattice (path) is shown. The size of the path adjacency matrix is 9×9 because there are 9 nodes. For every possible tunneling, the corresponding matrix element equals to 1. For instance, since the particle can tunnel from site 1 to site 2 and vice versa, the corresponding matrix elements [1][2] and [2][1] equals to 1.

B. How to use this code?

Firstly, enter the number of nodes, note that this number must be the square of an integer, such that it is a square lattice. Enter 0 if you want to generate a cycle adjacency matrix, enter 1 if you want to generate a path adjacency matrix. The generator then generates the corresponding matrix, its eigenvalues and eigenvectors.
Note that the eigenvectors are read vertically.

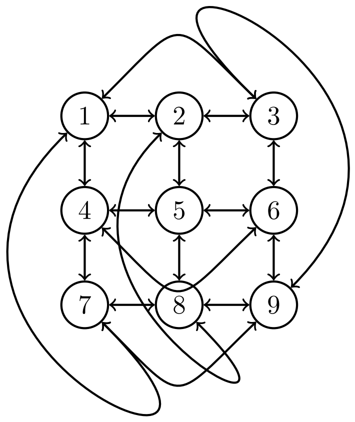

FIG. 4. A 2D cycle of 9 nodes

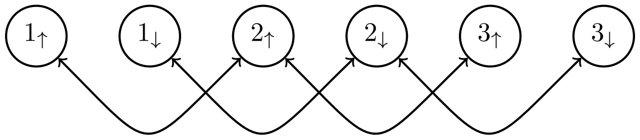

FIG. 5. A 1D path of 3 nodes when tunneling between neighboring sites with the same spin direction is allowed

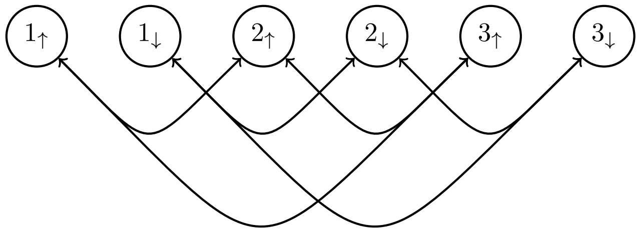

FIG. 6. A 1D cycle of 3 nodes when tunneling between neighboring sites with the same spin direction is allowed

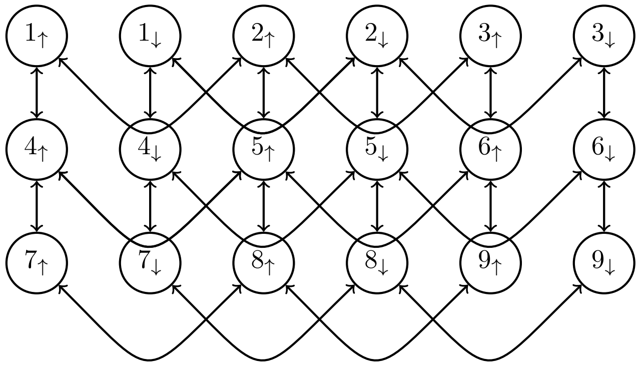

FIG. 7. A 2D path of 9 nodes when tunneling between neighboring sites with the same spin direction is allowed

IV. CODE [3]

A. What does this code do?

This code generates the 1D cycle or path adjacency matrix. Unlike code [1], spinning is considered this time. However, the change of spin direction is not allowed while tunneling. For every possible tunneling (illustrated in Fig. 5 and Fig. 6), the corresponding matrix element equals to 1. For the configuration of the matrix, the rows and columns go like $1_\uparrow, 1_\downarrow, 2_\uparrow, 2_\downarrow, 3_\uparrow, 3_\downarrow$, etc.

B. How to use this code?

Firstly, enter the number of nodes, then enter 0 if you want to generate a cycle adjacency matrix, enter 1 if you want to generate a path adjacency matrix. The generator then generates the corresponding matrix, its eigenvalues and eigenvectors. Note that the eigenvectors are read vertically.

V. CODE [4]

A. What does this code do?

This code generates the 2D cycle or path adjacency matrix. Unlike code [2], spinning is considered this time. However, the change of spin direction is not allowed while tunneling. For every possible tunneling (illustrated in Fig. 7), the corresponding matrix element equals to 1. For the configuration of the matrix, the rows and columns go like $1_\uparrow, 1_\downarrow, 2_\uparrow, 2_\downarrow, 3_\uparrow, 3_\downarrow$, etc.

B. How to use this code?

Firstly, enter the number of nodes, note that this number must be the square of an integer, such that it is a square lattice. Enter 0 if you want to generate a cycle adjacency matrix, enter 1 if you want to generate a path adjacency matrix. The generator then generates the corresponding matrix, its eigenvalues and eigenvectors.
Note that the eigenvectors are read vertically.

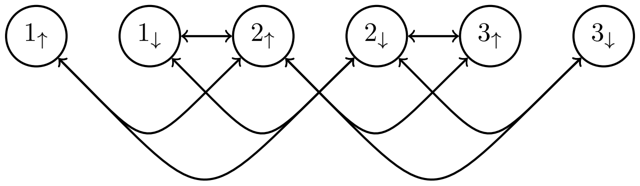

FIG. 8. A 1D path of 3 nodes when tunneling between neighboring sites with both the same and different spin directions is allowed

VI. CODE [5]

A. What does this code do?

This code generates the 1D cycle or path adjacency matrix, but the change of spin direction is allowed while tunneling. For every possible tunneling (illustrated in Fig. 8), the corresponding matrix element equals to 1. For the configuration of the matrix, the rows and columns go like $1_\uparrow, 1_\downarrow, 2_\uparrow, 2_\downarrow, 3_\uparrow, 3_\downarrow$, etc.

B. How to use this code?

Firstly, enter the number of nodes, then enter 0 if you want to generate a cycle adjacency matrix, enter 1 if you want to generate a path adjacency matrix. The generator then generates the corresponding matrix, its eigenvalues and eigenvectors. Note that the eigenvectors are read vertically.

VII. CODE [6]

A. What does this code do?

This code generates the 2D cycle or path adjacency matrix, but the change of spin direction is allowed while tunneling. For every possible tunneling (illustrated in Fig. 9), the corresponding matrix element equals to 1. For the configuration of the matrix, the rows and columns go like $1_\uparrow, 1_\downarrow, 2_\uparrow, 2_\downarrow, 3_\uparrow, 3_\downarrow$, etc.

B. How to use this code?

Firstly, enter the number of nodes, then enter 0 if you want to generate a cycle adjacency matrix, enter 1 if you want to generate a path adjacency matrix. The generator then generates the corresponding matrix, its eigenvalues and eigenvectors. Note that the eigenvectors are read vertically.

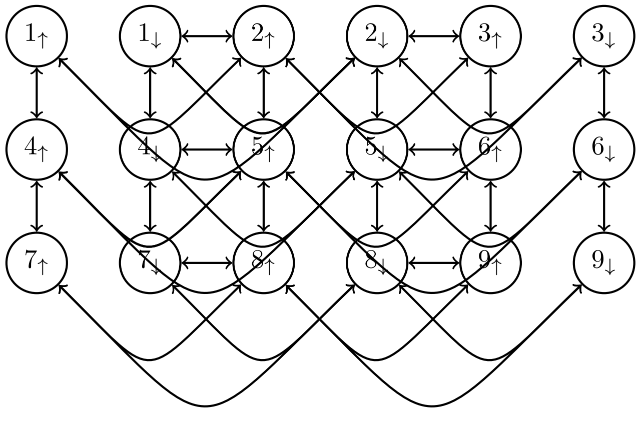

FIG. 9. A 2D path of 9 nodes when tunneling between neighboring sites with both the same and different spin directions is allowed

VIII. CODE [7]

A. What does this code do?

This code generates the 2D cycle or path adjacency matrix with the tunneling terms t, t_z, m_z, t_{so} added. The equations of them are

$$\begin{aligned}
H_{hop} =& -t \sum_{j_x,j_y,s} (b^\dagger_{j_x+1,j_y,s} b_{j_x,j_y,s} + b^\dagger_{j_x,j_y+1,s} b_{j_x,j_y,s} + H.c.) \\
H_{zhop} =& t_z \sum_{j_x,j_y} (b^\dagger_{j_x+1,j_y,\uparrow} b_{j_x,j_y,\uparrow} + b^\dagger_{j_x,j_y+1,\uparrow} b_{j_x,j_y,\uparrow} \\
& - b^\dagger_{j_x+1,j_y,\downarrow} b_{j_x,j_y,\downarrow} - b^\dagger_{j_x,j_y+1,\downarrow} b_{j_x,j_y,\downarrow} + H.c.) \\
H_{so} =& t_{so} \sum_{j_x,j_y} (b^\dagger_{j_x+1,j_y,\uparrow} b_{j_x,j_y,\downarrow} + b^\dagger_{j_x+1,j_y,\downarrow} b_{j_x,j_y,\uparrow} \\
& - b^\dagger_{j_x,j_y+1,\uparrow} b_{j_x,j_y,\downarrow} - b^\dagger_{j_x,j_y+1,\downarrow} b_{j_x,j_y,\uparrow} \\
& - i b^\dagger_{j_x+1,j_y+1,\uparrow} b_{j_x,j_y,\downarrow} + i b^\dagger_{j_x+1,j_y+1,\downarrow} b_{j_x,j_y,\uparrow} \\
& + i b^\dagger_{j_x+1,j_y-1,\uparrow} b_{j_x,j_y,\downarrow} - i b^\dagger_{j_x+1,j_y-1,\downarrow} b_{j_x,j_y,\uparrow} + H.c.) \\
H_{det} =& m_z \sum_{j_x,j_y} (b^\dagger_{j_x,j_y,\uparrow} b_{j_x,j_y,\uparrow} - b^\dagger_{j_x,j_y,\downarrow} b_{j_x,j_y,\downarrow})
\end{aligned}$$

B. How to use this code?

Firstly, enter Lx, Ly which are the number of nodes along the x-axis and y-axis, then enter 0 if you want to generate a cycle adjacency matrix, enter 1 if you want to generate a path adjacency matrix. After that, enter the value of t, t_z, t_{so}, m_z.
The generator then generates *matrix_up_up*, *matrix_down_down*, *matrix_up_down*, *matrix_down_up*. They are matrices related to the spinning direction. For instance, if a particle which is spinning upward is still

spinning upward after tunneling, its information goes into *matrix_up_up*. The generator also generates the Hamiltonian, its configuration is

$$H = \begin{pmatrix} H_{\uparrow\uparrow} & H_{\uparrow\downarrow} \\ H_{\downarrow\uparrow} & H_{\downarrow\downarrow} \end{pmatrix}$$

Finally, the eigenvalues and eigenvectors of the Hamiltonian are generated. Note that the eigenvectors are read vertically.

IX. CODE [8]

A. What does this code do?

This code generates the 2D cycle or path adjacency matrix for two-particle tunneling, with the tunneling terms t, t_z, t_{so}, m_z added. In the code, I found Q first, then use it to find the two-particle Hamiltonian such that

$$B = Q^T A Q$$

in which B is the two-particle Hamiltonian and A is the 1-particle Hamiltonian generated in code [7].

B. How to use this code?

Nothing needs to be entered. The generator generates the two-particle Hamiltonian, its eigenvalues and eigenvectors automatically. Note that code [7] MUST be run before running code [8].