# Read me when in doubt

Nathan Ngo

August 2023

## [1]

### 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 tunnelling (illustrated in figure 1.1 and figure 1.2), 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.

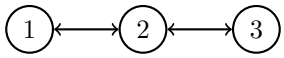


Figure 1.1 - A 1D path of 3 nodes

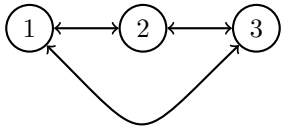


Figure 1.2 - A 1D cycle of 3 nodes

### 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.

## [2]

### What does this code do?

This code generates the 2D cycle or path adjacency matrix without any spin. In figure 2.1, the configuration of a $3 \times 3$ lattice (path) is shown. The size of the

path adjacency matrix is $9 \times 9$ because there are 9 nodes. For every possible tunnelling, 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.

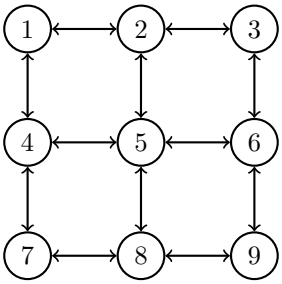


Figure 2.1 - A 2D path of 9 nodes

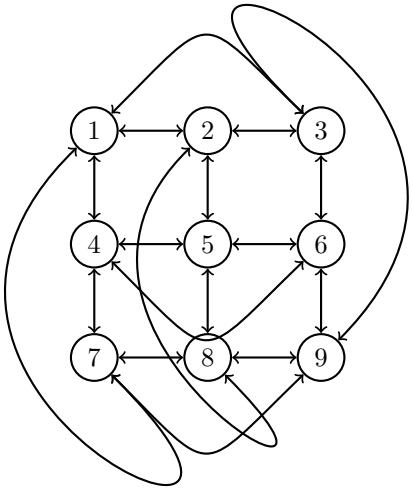


Figure 2.2 - A 2D cycle of 9 nodes

## 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.

# [3]

## What does this code do?

This code generates the 1D cycle or path adjacency matrix. Unlike [1], spinning is considered this time. However, the change of spin direction is not allowed while tunnelling. For every possible tunnelling (illustrated in figure 3.1 and figure 3.2), 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.

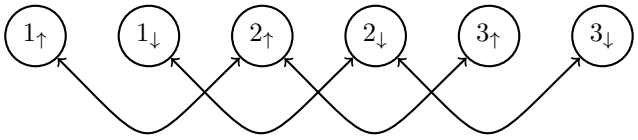


Figure 3.1 - A 1D path of 3 nodes

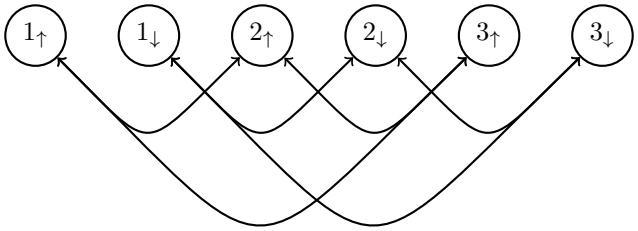


Figure 3.2 - A 1D cycle of 3 nodes

## 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.

# [4]

## What does this code do?

This code generates the 2D cycle or path adjacency matrix. Unlike [2], spinning is considered this time. However, the change of spin direction is not allowed while tunnelling. For every possible tunnelling (illustrated in figure 4.1), 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.

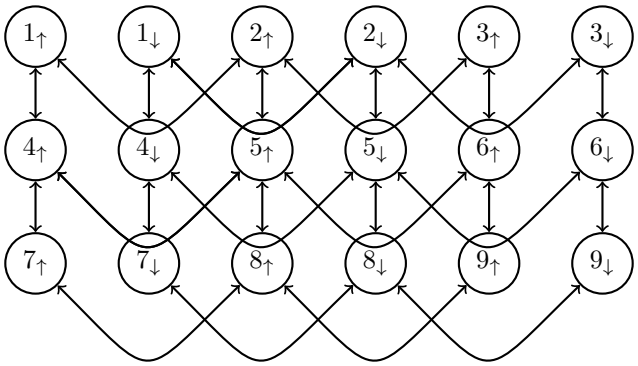


Figure 4.1 - A 2D path of 9 nodes

## 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.

# [5]

## What does this code do?

This code generates the 1D cycle or path adjacency matrix, but the change of spin direction is allowed while tunnelling. For every possible tunnelling (illustrated in figure 5.1), 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.

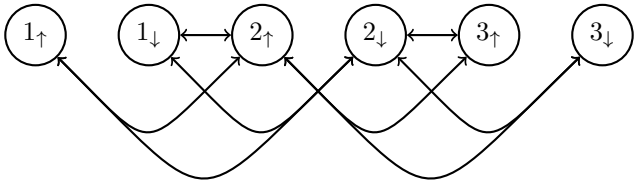


Figure 5.1 - A 1D path of 3 nodes

### 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.

## [6]

### 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 tunnelling (illustrated in figure 6.1), 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.

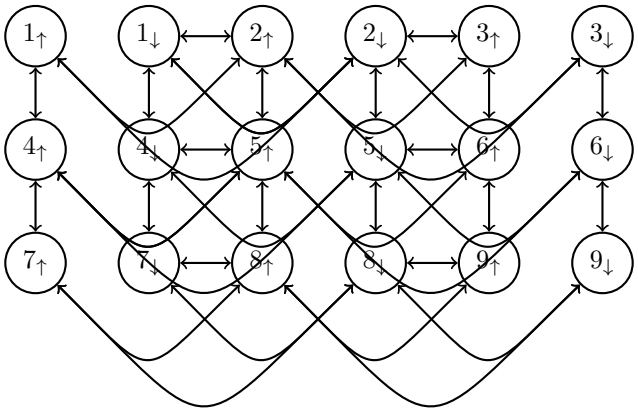


Figure 6.1 - A 2D path of 9 nodes

## 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.

# [7]

## What does this code do?

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

$$H_{hop} = -t \sum_{j_x,j_y,s} (b^\dagger_{j_x+1,j_y,s} b_{j_x,j_y,s} + b^\dagger_{j_x,j_y+1,s} b_{j_x,j_y,s} + H.c.)$$

$$H_{zhop} = t_z \sum_{j_x,j_y} (b^\dagger_{j_x+1,j_y,\uparrow} b_{j_x,j_y,\uparrow} + b^\dagger_{j_x,j_y+1,\uparrow} b_{j_x,j_y,\uparrow} - b^\dagger_{j_x+1,j_y,\downarrow} b_{j_x,j_y,\downarrow} - b^\dagger_{j_x,j_y+1,\downarrow} b_{j_x,j_y,\downarrow} + H.c.)$$

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

$$H_{det} = m_z \sum_{j_x,j_y} (b^\dagger_{j_x,j_y,\uparrow} b_{j_x,j_y,\uparrow} - b^\dagger_{j_x,j_y,\downarrow} b_{j_x,j_y,\downarrow})$$

## 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 tunnelling, 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.

# [8]

## What does this code do?

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

$$B = Q^T A Q$$

in which $B$ is the 2-particle Hamiltonian and $A$ is the 1-particle Hamiltonian generated in [7].

## How to use this code?

Nothing needs to be entered. The generator generates the 2-particle Hamiltonian, its eigenvalues and eigenvectors automatically.
Note that code [7] MUST be run before running [8].