2022 2023 Home exam A

Gradding system of the exam:

The exam consists of two types of tasks.

The mandatory tasks contribute to 100% of the grade according to the following distribution:

- 40% for discussion, equally weighted among all tasks;
- 40% for coding, equally weighted among all developed codes; and
- 20% for the overall quality of the exam solution.

The optional extra credit tasks can be used to supplement the grade obtained from the mandatory questions and tasks, up to a maximum of 100% of the total grade.

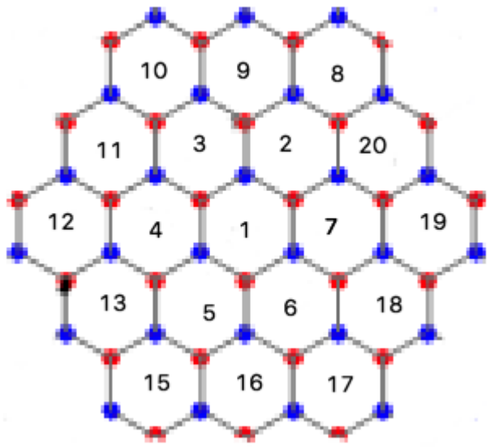
If the combined grade obtained from the mandatory tasks, along with the extra credit, exceeds 100%, the final grade of the exam will be capped at 100%.

Instructions:

1. The home exam consists of several tasks and is designed to be completed within 24 hours (plus 24 hours of tolerance). Read the instructions and questions carefully before proceeding.
2. You are allowed to use any reference materials, textbooks, notes, and online resources during the exam. However, collaboration with others or seeking direct assistance from individuals is strictly prohibited. Include the references of the material consulted in each answer.
3. The exam is open-book, but it is expected that you answer the questions independently, using your own understanding and knowledge. In other words, while you may use external resources to support your answers and provide accurate information, you should use your own words and present your understanding of the concepts and ideas being assessed in the exam. Your responses should reflect your comprehension, critical thinking, and ability to apply the knowledge you have acquired throughout the course.
4. The exam must be completed individually. Do not discuss the questions or share your answers with classmates or anyone else.
5. The exam must be submitted by the specified deadline indicated in the moodle webpage under exam submission. Late submissions will not be accepted.
6. Type your answers directly on the jupyter notebook. Ensure that your responses are clear, legible, and organized.
7. Clearly label each question and provide the corresponding answer. You may use bullet points, numbered lists, or paragraphs as appropriate.
8. If a question requires calculations or mathematical expressions, you may use software or tools like calculators, spreadsheets, or programming languages. However, show your work and provide explanations for each step to demonstrate your understanding.
9. Plagiarism and academic dishonesty will not be tolerated. Any evidence of copying or unauthorized collaboration will result in severe penalties, including possible academic misconduct charges.
10. Save your completed exam as a single PDF file indicating your name in the file name. Also submit the original jupyter notebook with all the outputs of all the code cells. Submit your exam electronically through the designated submission method found in the moodle webpage.
11. Double-check your submission to ensure that the file is attached correctly and that all pages are included.
12. Keep a backup copy of your completed exam until the grading process is finalized.
13. If you have any questions or concerns regarding the exam, please contact the instructor or designated course staff for clarification. A forum will be made available during the exam to contact the instructor, all questions and answers will be visible to the remaining students.
14. Remember to manage your time effectively to allocate sufficient time for each question and to review your answers before submitting.
15. Violation of any of these instructions or any other form of academic dishonesty will have serious consequences, as per the institution's academic integrity policy.

Good luck with your exam!

Problem Statement:



Consider a physical quantum system comprising a single particle trapped in several coupled two-dimensional potential wells arranged in a hexagonal lattice structure, as depicted in the figure above.

Your objective is to create a quantum simulation experiment of this system using Qiskit.

IMPORTANT NOTE:

Unless explicitly specified, the term "quantum simulation" refers to a numerical experiment conducted using Qiskit to emulate a quantum computer. If there is a specific requirement to use an actual IBM quantum computer, it will be clearly mentioned.

Physical model

To provide a more comprehensive description of the physical model of the particle, we will first discuss the states of the particle in a single well. Subsequently, we will explain the process of generating the lattice.

Single Finite Circular Potential in Polar Coordinates

The classical state of the particle is determined by its position $\mathbf{r} = (x, y)$ and linear momentum $\mathbf{p} = (p_x, p_y)$ in two dimensions.

The potential describing a single finite circular well positioned at $\mathbf{R} = (X, Y)$ is represented by the function:

$$V_{\mathbf{R}}(\mathbf{r}) = \begin{cases} 0, & \text{if } (x-X)^2 + (y-Y)^2 > a^2 \text{ (the region I outside the well)} \\ -V_0, & \text{if otherwise (the region II inside the well)} \end{cases}$$

To solve for the quantum states of the particle, we can use polar coordinates centered at $\mathbf{R} = (X, Y)$. In these coordinates, the potential simplifies to:

$$V(r) = \begin{cases} 0, & \text{if } r > a, \\ -V_0, & \text{if } r \leq a, \end{cases}$$

Our goal is to find the bounded eigenstates for this potential. We can separate the variables in the Schrödinger equation and solve for the radial and angular parts independently.

The radial part of the wavefunction can be obtained by solving the radial equation:

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{dR}{dr}\right) + \frac{2\mu}{\hbar^2}(E - V(r))R(r) = 0,$$

where μ is the reduced mass of the system, E is the energy, and $V(r)$ is the potential function.

The angular part of the wavefunction corresponds to the eigenstates of the angular momentum operator. In polar coordinates, the angular equation can be written as:

$$\frac{d^2\Theta}{d\theta^2} + 2i\ell\frac{d\Theta}{d\theta} = -m^2\Theta(\theta),$$

where ℓ is the eigenvalue of the angular momentum operator and m is the eigenvalue of the angular momentum projection operator.

From this point on, we consider $\ell = m = 0$.

To obtain the solution for the bounded eigenstates, we express the wavefunction as a product of the radial and angular parts:

$$\psi(r, \theta) = R(r)\Theta(\theta).$$

The solutions for the radial part can be expressed in terms of Bessel functions.

For the angular part, the eigenfunctions are given by:

$$\Theta(\theta) = e^{im\theta},$$

where m is an integer representing the eigenvalue of the angular momentum projection operator.

By combining the radial and angular parts, the approximate bounded eigenstates can be written as:

$\psi(r, \theta) \approx R(r)e^{im\theta}$.

The energy eigenvalues corresponding to these eigenstates can be obtained by solving the radial equation and applying the appropriate boundary conditions.

For a finite circular potential well with radius a, the radial boundary conditions are typically specified as follows:

1. At the origin ($r = 0$): We require that the radial wavefunction, $R(r)$, remains finite. This condition ensures that there is no singularity at the origin.

2. At the boundary of the potential well ($r = a$): We impose either the condition $R(a) = 0$ or the condition $\frac{dR(a)}{dr} = 0$, depending on the specific form of the potential.

The choice of boundary condition at the potential well boundary depends on whether we are interested in finding the bound state energy eigenvalues or the scattering state solutions.

For bound state solutions, we seek the energies at which the radial wavefunction vanishes at the well boundary, i.e., $R(a) = 0$. This condition corresponds to the confinement of the particle within the potential well.

For scattering state solutions, we are interested in energies at which the radial wavefunction and its derivative are continuous at the well boundary, i.e., $R(a)$ is finite and $\frac{dR(a)}{dr}$ is finite. This condition ensures that the wavefunction remains well-behaved outside the potential well.

In this problem, our goal is to find the bounded solutions.

The solution to the radial equation within the potential well ($r \leq a$) is typically expressed in terms of Bessel functions. The transcendental equation arises from the boundary condition $R(a) = 0$:

$$J_0(kr)\Big|_{r=a} = 0,$$

where $J_0(kr)$ is the Bessel function of the first kind of order zero, and $k = \sqrt{\frac{2\mu}{\hbar^2}(E + V_0)}$.

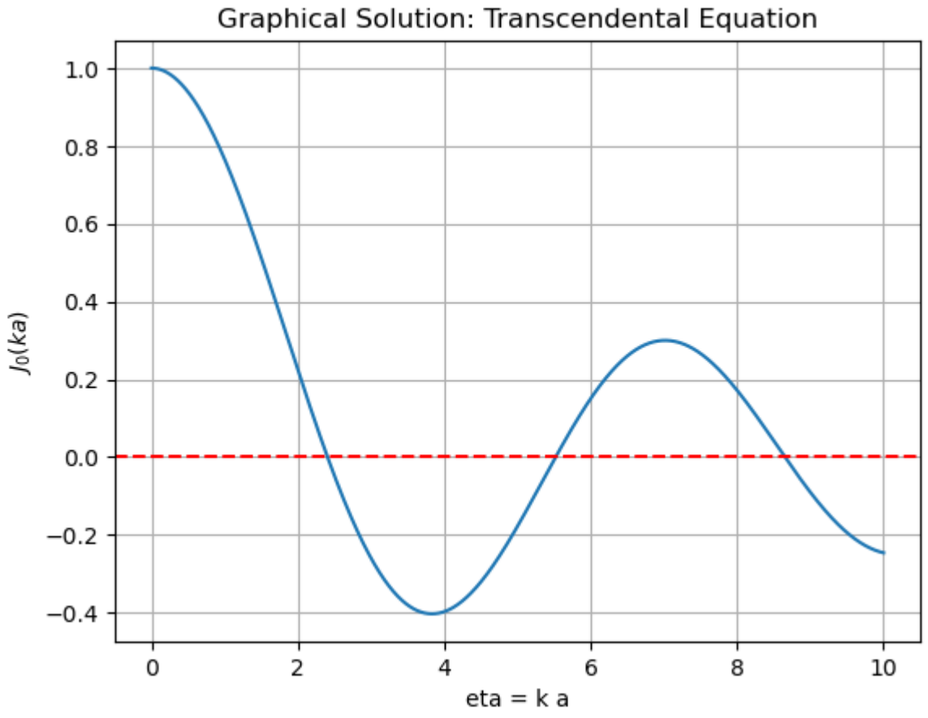
The solution to this transcendental equation provides the energy eigenvalues for the bound states of the given potential.

Please note that the solutions to this transcendental equation can be obtained numerically or using approximation methods such as graphical methods or root-finding algorithms.

For simplicity, we choose $\frac{2\mu}{\hbar^2} = 1$ as a system of units.

Below is an example code provided in the following cell to find the solutions to the transcendental equation:

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import brentq
from scipy.special import jv

# Function to define the transcendental equation for R(a)=0
def transcendental_eq(eta):
    return jv(0, eta)

# Function to define the transcendental equation for R'(a)=0
#def transcendental_eq(k, a):
#    return jv(1, k*a)

#physical parameters
a = 1.0
V0 = 20.0
# Graphical method
eta_vals = np.linspace(0, 10, 1000)
trans_eq_vals = transcendental_eq(eta_vals)

plt.figure()
plt.plot(eta_vals, trans_eq_vals)
plt.xlabel('eta = k a')
plt.ylabel('$J_0(ka)$')
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Graphical Solution: Transcendental Equation')
plt.grid(True)
plt.show()

# Root-finding algorithm (Brent's method)
solution1 = brentq(transcendental_eq, 2, 4)
print(f"Solution using root-finding algorithm: eta1 = {solution1}")
solution2 = brentq(transcendental_eq, 4, 6)
print(f"Solution using root-finding algorithm: eta2 = {solution2}")
solution3 = brentq(transcendental_eq, 8, 10)
print(f"Solution using root-finding algorithm: eta3 = {solution3}")
```




```
Solution using root-finding algorithm: eta1 = 2.404825557695773
Solution using root-finding algorithm: eta2 = 5.520078110286316
Solution using root-finding algorithm: eta3 = 8.653727912911013
```

Task for extra points 1: Please choose a value of V_0 such that the circular potential well has exactly two bounded eigenstates. This requirement pertains to the principles of quantum mechanics. Please provide a justification for your choice.
If you were unable to determine the appropriate value, you can proceed with $V_0 = 100.0$. In this case, consider only the two eigenstates with higher energies for the subsequent questions.

Reply:

Lattice of Finite Circular Potentials

The current particle-in-a-well model may not be particularly interesting as it closely resembles the single-well model discussed in class. To introduce more complexity, we will now consider a hexagonal lattice. The confinement potential of the lattice can be expressed as a sum of individual well potentials:

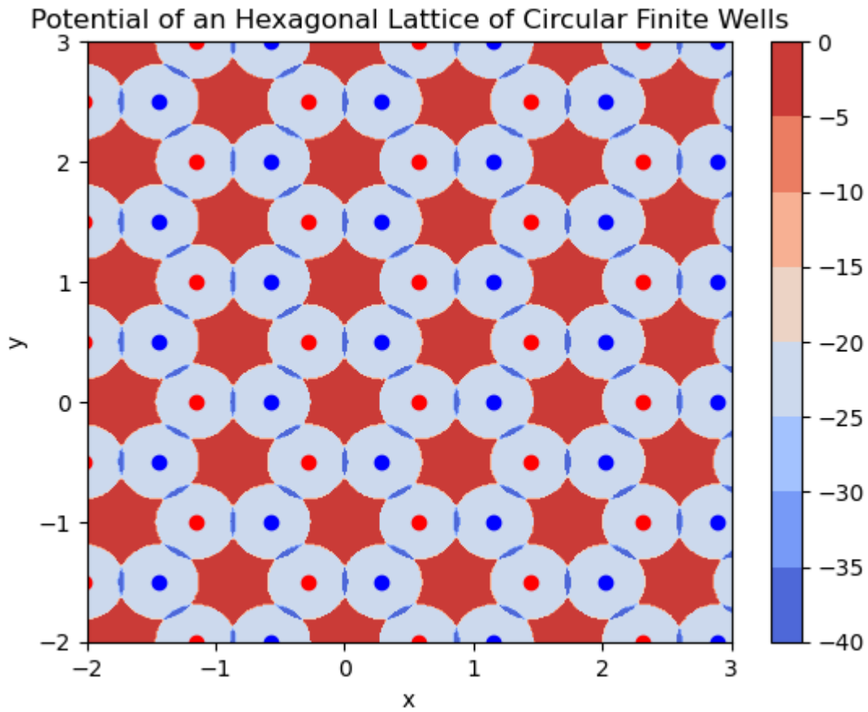
$$V(\mathbf{r}) = \sum_{\mathbf{R}} V_{\mathbf{R}}(\mathbf{r})$$

where $\mathbf{R}$ is the position of each well.

The hexagonal lattice consists of two wells in each cell, designated as A and B, as shown in the visualization:

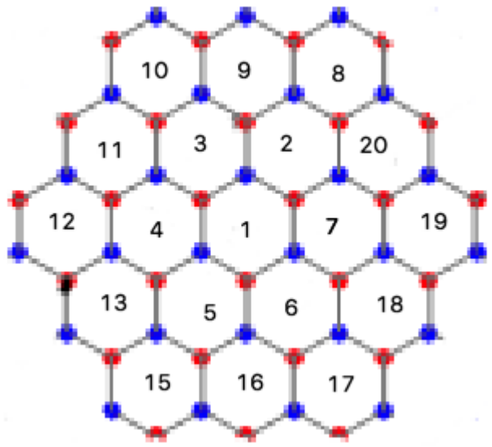
In [69]:
```python
import numpy as np
import matplotlib.pyplot as plt

a = 0.32  # radius of the wells
b = 1.0  # lattice constant
V0 = 20.0

# Define the potential function for a single well
def V(x, y, X, Y, a, V0):
    r = np.sqrt((x - X)**2 + (y - Y)**2)
    return np.where(r <= a, -V0, 0.0)

# Define the potential function for the hexagonal lattice of wells
def V_hex(x, y, b, a, V0):
    # Generate lattice points
    a1 = np.array([b * np.cos(np.pi/6.0), b * np.sin(np.pi/6.0)])
    a2 = np.array([-b * np.cos(np.pi/6.0), b * np.sin(np.pi/6.0)])
    x_range = np.arange(-5, 6)
    y_range = np.arange(-5, 6)
    x_coords = []
    y_coords = []
    for i in range(len(x_range)):
        for j in range(len(y_range)):
            lattice_point = x_range[i] * a1 + y_range[j] * a2
            x_coords.append(lattice_point[0])
            y_coords.append(lattice_point[1])
    xA = np.array(x_coords) + b / np.sqrt(3)
    yA = np.array(y_coords)
    xB = np.array(x_coords) - b / np.sqrt(3)
    yB = np.array(y_coords)

    potential_sum = 0.0
    for X, Y in zip(xA, yA):
        potential_sum += V(x, y, X, Y, a, V0)

    for X, Y in zip(xB, yB):
        potential_sum += V(x, y, X, Y, a, V0)

    return potential_sum, xA, yA, xB, yB

# Define the range of x and y values to plot
x_min, x_max = -2.0, 3.0
y_min, y_max = -2.0, 3.0

# Define the number of points to use in the x and y directions
n_points = 400

# Create a meshgrid of x and y values
x_vals = np.linspace(x_min, x_max, n_points)
y_vals = np.linspace(y_min, y_max, n_points)
x_mesh, y_mesh = np.meshgrid(x_vals, y_vals)

# Calculate the potential at each point in the meshgrid
V_mesh, xA, yA, xB, yB = V_hex(x_mesh, y_mesh, b, a, V0)

# Create a contour plot of the potential
plt.contourf(x_mesh, y_mesh, V_mesh, cmap='coolwarm')
plt.colorbar()

plt.scatter(xA, yA, c='red', marker='o', label='Well A')
```

```
plt.scatter(xB, yB, c='blue', marker='o', label='Well B')



# Set the range for the x and y axes
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

# Add labels and a title to the plot
plt.xlabel('x')
plt.ylabel('y')
plt.title('Potential of an Hexagonal Lattice of Circular Finite Wells')

# Show the plot
#plt.legend()
plt.show()
```



It is important to note that the radius a of each well should be slightly larger than the distance between the centers of neighboring wells. This small overlap between the wavefunctions of the individual wells enables the coupling of states between different wells, allowing the particle to transition or "hop" between them. However, if the radius a becomes much larger than the distance between neighboring well centers, the validity of the physical model presented here starts to diminish.

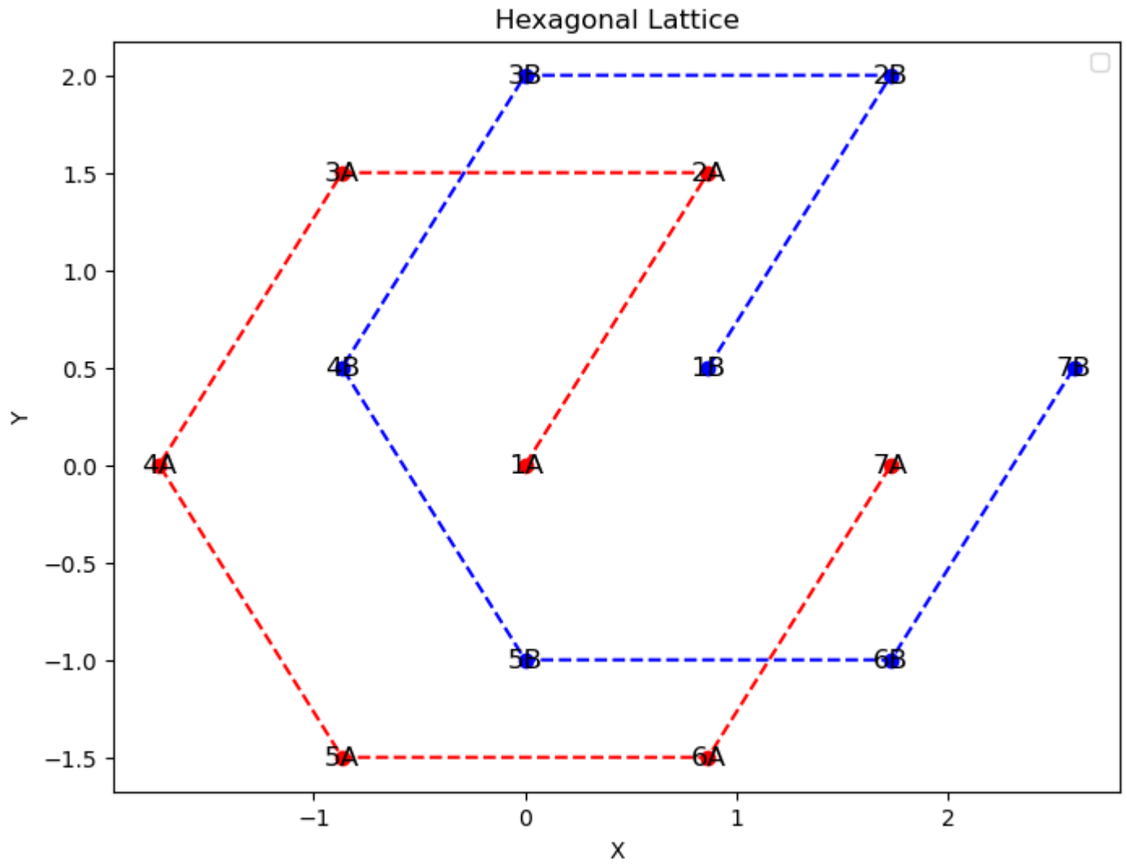
Simulating an infinite hexagonal lattice with a limited number of qubits in the quantum simulator is not feasible without making additional approximations. Therefore, we will focus on simulating only a subset of cells, specifically cells 1 to 7, as shown in the following picture:



Below is the code to visualize the schematic of the simulated portion of the lattice:

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

def hexagonal_lattice(n):
    x = np.zeros(3*n*(n-1) + 1)
    y = np.zeros(3*n*(n-1) + 1)
    for i in range(1, n):
        r = i * np.sqrt(3)
        for j in range(6):
            x[6 * (i - 1) + j] = r * np.cos(-j * np.pi / 3)
            y[6 * (i - 1) + j] = r * np.sin(-j * np.pi / 3)
    return x, y

X, Y = hexagonal_lattice(2)
X = np.flip(X)
Y = np.flip(Y)

plt.figure(figsize=(8, 6))
plt.plot(X, Y, 'ro--')
plt.plot(X + np.cos(np.pi/6.0), Y + np.sin(np.pi/6.0), 'bo--')

# Annotate each point with its number
for i, (xi, yi) in enumerate(zip(X, Y)):
    plt.text(xi, yi, str(i+1)+'A', color='black', fontsize=12, ha='center', va='center')
    plt.text(xi + np.cos(np.pi/6.0), yi + np.sin(np.pi/6.0), str(i+1)+'B', color='black', fontsize=12, ha='center', va='center')

plt.xlabel('X')
plt.ylabel('Y')
plt.title('Hexagonal Lattice')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



The provided code generates the coordinates of the lattice sites in a hexagonal lattice. The lattice consists of a central cell surrounded by consecutive layers of cells, following the ordering scheme depicted in the figure.

The function hexagonal_lattice accepts an integer n as input and returns two arrays, x and y, containing the x and y coordinates of the lattice sites. The parameter n determines the number of layers of cells generated around the central cell.

To identify each potential well within the lattice, the following quantities are used:

- q ranges from 1 to 7 and represents the number of the cell containing the well.
- x can take the values "A" or "B" to differentiate between the two wells present in each cell.

For instance, the notation (q = 3, x = A) refers to the leftmost well in the provided image.

The states of the particle in the lattice

The next step involves employing a Sommerfeld-type solution, which assumes that the solutions of the individual wells serve as reasonable approximations for the stationary solutions of the entire system. In other words, the influence of the other wells on the initial solutions is minimal. If each well possesses two bound solutions, then a system consisting of N wells will have $2N$ solutions.

Let us consider that the Hilbert space describing the particle's state has a quasi-orthogonal basis, where each element corresponds to one of the two states of each potential well. We denote this basis as

$$\{|q, x, s\rangle\},$$

where q denotes the cell, s takes the values of 1 or 0 depending on which of the two bound states of the potential well is being described, and x takes the values of A or B to indicate whether the potential well corresponds to element site A or B of the unit cell in the lattice. For example, the state

$$|3, A, 1\rangle,$$

represents the quantum state where the particle is located in unit cell 3, specifically at element site A, and in the state 1. Here, 0 represents the lower energy bound state, or ground state, while 1 represents the higher energy bound state, or excited state.

By utilizing this approach, we can simplify the analysis and calculations involved in studying the entire lattice system. Instead of solving the complex problem of an infinite hexagonal lattice, we can focus on the individual wells and use their solutions as a basis to construct the approximate solutions for the entire system. This approximation is valid as long as the corrections introduced by neighboring wells are negligible compared to the individual well solutions.

The Hamiltonian for the particle in the lattice can be expressed as:

$$H = \frac{p^2}{2\mu} + \sum_{\mathbf{R}} V_{\mathbf{R}}(\mathbf{r}).$$

The decomposition of the Hamiltonian in the quasi-orthogonal basis is given by:

$$\begin{aligned}
&\langle q, x, s|H|q', x', s'\rangle = && (1)\\
&= \langle q, x, s|\frac{p^2}{2\mu} + V_{\mathbf{R}=(q,x)}(\mathbf{r}) + \sum_{\mathbf{R}\neq(q',x')} V_{\mathbf{R}}(\mathbf{r})|q', x', s'\rangle \approx && (2)\\
&\approx E_s\delta_{q,q'}\delta_{x,x'}\delta_{s,s'} + \langle q, x, s| \sum_{\mathbf{R}\neq(q',x')} V_{\mathbf{R}}(\mathbf{r})|q', x', s'\rangle && (3)
\end{aligned}$$

Here, we make use of the first neighbor approximation:

$$\langle q, x, s| \sum_{\mathbf{R}\neq(q',x')} V_{\mathbf{R}}(\mathbf{r})|q', x', s'\rangle \approx \begin{cases} t(s, s'), & \text{if the lattice sites } (q, x) \text{ and } (q', x') \text{ are first neigbours,} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The terms $t(s, s')$ represent the hopping parameter between nearest-neighbor wells, obtained from the overlap of their states. It describes the energy of the particle as it moves between neighboring lattice sites.

Therefore, a good approximation of the Hamiltonian in the quasi-orthogonal basis can be written as:

$$H = \sum_{(q,x,s)} E_s|q, x, s\rangle\langle q, x, s| + \sum_{(q,x,s),(q',x',s') \text{ are first neighbors}} t(s, s')|q, x, s\rangle\langle q', x', s'|$$

Task for extra points 2:
Provide estimates of t by computing the hopping overlap integral (or some approximation of it) for different values of a. Select a specific combination of parameters a and b and use these parameters for the rest of the exam.

If you were not able to compute these integrals use $t(s,s) \approx E_s/10$ and $t(0,1) = t(1,0) \approx |E_0 - E_1|/100$.

Reply:

You now have all the necessary information about the physical system to generate a quantum simulation using Qiskit. In this exam, we will begin with a simple model and gradually introduce improvements as we progress.

Exercise 1

Mandatory task 1: Establish a correspondence map between the elements of the semi-orthogonal basis in the original physical system and the elements of a basis in the quantum simulator (i.e. your Qiskit circuit).

Reply:

I'm going to structure the semi-orthogonal basis as: $[q, A, 0], [q, A, 1], [q, B, 0], [q, B, 1] = |q_0\rangle \otimes |q_1\rangle \otimes |q_2\rangle$

$$|1, A, 0\rangle = \begin{pmatrix} 1 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |000\rangle \otimes |0\rangle \otimes |0\rangle \qquad |1, A, 1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |000\rangle \otimes |0\rangle \otimes |1\rangle \ldots\ldots\ldots |7, B, 0\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix} = |110\rangle \otimes |1\rangle \otimes |0\rangle \qquad |7, B, 1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} = |110\rangle \otimes |1\rangle \otimes |1\rangle$$

Mandatory task 2: Let's consider the initial state of the physical system where the particle is trapped in cell site A of the cell with $q = 0$, and it is in the excited state $s = 1$. Using the mapping we discussed earlier, we can determine the corresponding state in the quantum simulator.

Reply:

$$|1, A, 1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = |000\rangle \otimes |0\rangle \otimes |1\rangle$$

Mandatory task 3: Identify the first neighbors in the lattice of wells.

Reply:

The cell 1 has neighbors {2,3,4,5,6,7}

The cell 2 has neighbors {1,3,7}

The cell 3 has neighbors {1,2,4}

The cell 4 has neighbors {1,3,5}

The cell 5 has neighbors {1,4,6}

The cell 6 has neighbors {1,5,7}

The cell 7 has neighbors {1,6,2}

But not every well {A,B} has the same neighbors as the cells, i will follow the mapping presented:

$$\begin{aligned} 1A \Rightarrow (1,5,6)B \text{ and } 1B \Rightarrow (1,2,3)A \\ 2A \Rightarrow (1,2,7)B \text{ and } 2B \Rightarrow (2)A \\ 3A \Rightarrow (1,3,4)B \text{ and } 3B \Rightarrow (3)A \\ 4A \Rightarrow (4,5)B \text{ and } 4B \Rightarrow (3,4)A \\ 5A \Rightarrow (5)B \text{ and } 5B \Rightarrow (1,4,5)A \\ 6A \Rightarrow (6)B \text{ and } 6B \Rightarrow (1,6,7)A \\ 7A \Rightarrow (6,7)B \text{ and } 7B \Rightarrow (2,7)A \end{aligned}$$

Mandatory task 4: Using the previous map, write the Hamiltonian for the quantum simulator that emulates the original physical system.

**Reply

It is relevant to state that i'm only going to assume hopping between different wells.

$$H = \sum_{(q,x,s)} E_s|q,x,s\rangle\langle q,x,s| + \sum_{(q,x,s),(q',x',s') \text{ are first neighbors}} t(s,s')|q,x,s\rangle\langle q',x',s'| =$$

$$= \begin{vmatrix}
E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} \\
0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} \\
\frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & E_s & \frac{|E_1-E_0|}{100} & E_s & 0 & E_s & \frac{|E_1-E_0|}{100} & 0 & 0 & E_s & \frac{|E_1-E_0|}{100} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 \\
\frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & E_s & 0 \\
\frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & E_s \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} \\
\frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{E_s}{10} & \frac{|E_1-E_0|}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{vmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & \frac{|E_1-E_0|}{100} & \frac{E_s}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Mandatory task 5: The measurement in the quantum simulator must provide you information to determine the probability of finding the particle in each well with a specific state, namely $s = 0$ or $s = 1$. Now discuss the algorithm or strategy to implement these measurements in the quantum simulator and how they are related with potential measurements on the original physical system.

Reply:

To implement im going to use 5 qubits, where the last qubit is going to keep track of the state (s), the one before, the ato at which it is (x), and the first three qubits are created to keep track of what cell the particle is on, with the idea that 3 qubits generate a basis of 8 elements, and we only need 7, and our basis in the quantum computer is going to have the following assingment to the physical cells:

$$q_0:$$
$$|000\rangle \equiv 1$$
$$|001\rangle \equiv 2$$
$$|010\rangle \equiv 3$$
$$|011\rangle \equiv 4$$
$$|100\rangle \equiv 5$$
$$|101\rangle \equiv 6$$
$$|110\rangle \equiv 7$$
$$|111\rangle \equiv \text{will not be relevant}$$

$$q_1:$$
$$|0\rangle \equiv A$$
$$|1\rangle \equiv B$$

$$q_2:$$
$$|0\rangle \equiv \text{Ground state}$$
$$|1\rangle \equiv \text{Excited state}$$

Mandatory task 6: To emulate the original physical system using a quantum circuit, implement a circuit with a finite number of qubits. Distribute the lattice sites being simulated as evenly as possible around the cell site A of the cell with $n = 0$ and $m = 0$, considering the computational capabilities of your computer.
Once the quantum circuit is constructed based on the previous tasks, execute it starting from the initial state mentioned in Task 2.

Reply:

In [34]:
```
import numpy as np
import math as mt
import scipy as sp
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import random
import re             # regular expressions module
import sys



from tqdm.notebook import tqdm
from scipy.linalg import block_diag


from pylab import plot
from qiskit import *
from qiskit.visualization import *

# importing specific packages of Qiskit
from qiskit import Aer
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, execute



from IPython.display import display, Image, SVG, Math, YouTubeVideo

%matplotlib inline
```

In [35]:
```
n_qubits=5

initial_state = [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]


#Constructing the Hamiltonian

#This part of code was adapted from Notebook 9


E1= -20
E0= -70

t0=-E0/10
t1=-E1/10
u=-abs(E1-E0)/100

H = np.array([
    [E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,0,0,t0,u,0,0,0,0,0,0,0,0],
    [0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,0,u,t1,0,0,0,0,0,0,0,0],
    [t0,u,E0,0,t0,u,0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [u,t1,0,E1,u,t1,0,0,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,0,0,0,0],
    [0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,0,0,0],
    [0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,t0,u,0,0,t0,u,E0,0,t0,u,0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,u,t1,0,0,u,t1,0,E1,u,t1,0,0,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0],
    [t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0],
    [u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0],
    [0,0,0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,0,0],
    [0,0,0,0,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]])

t_max = 10
dt = 0.1

time = np.arange(dt,t_max,dt)

shots = 2048
# Create a Quantum Circuit acting on a quantum register
data = [[0,{"00001":shots}]]
for t in time:
```

```
    #print(t)
    q = QuantumRegister(n_qubits, name = 'q')
    c = ClassicalRegister(n_qubits, name = 'c')
    circuit1 = QuantumCircuit(q,c,name='qc')
    # Create the initial state
    circuit1.initialize(initial_state, [q[0],q[1],q[2],q[3],q[4]])

    circuit1.hamiltonian(H,t,[q[0],q[1],q[2],q[3],q[4]])

    circuit1.measure(0,0)
    circuit1.measure(1,1)
    circuit1.measure(2,2)
    circuit1.measure(3,3)
    circuit1.measure(4,4)
    simulator = Aer.get_backend('qasm_simulator')
    job = execute(circuit1, simulator,shots=shots)
    results = job.result()
    counts = results.get_counts()

    data.append([t,counts])
```

Mandatory task 7: Provide the graphics to illustrate the evolution of the measurement statistics over time as obtained from the quantum simulator. Use the type of graphics you consider adequate.
`Suggestion: You can solve tasks 6 and 7 together.`

Reply:

First to edit the data set, i'm going to use Structuring data functions made by the professor, present in the 9th Notebook.

In [36]:
```
def binary_list(n):
    return ['{:0{}b}'.format(i, n) for i in range(2**n)]
```

In [37]:
```
# Python program to get dictionary keys as list
def getList(dict):
    return [*dict]
```

In [38]:
```python
def getStruturedData_grid(data, n_qubits):
    X = []
    Y = []
    Z = []

    states_list = binary_list(n_qubits)
    for a in data:
        time, dictOfElements = a
        listOfItems = dictOfElements.items()
        rowX = []
        rowY = []
        rowZ = []
        for state in states_list:
            if state in dictOfElements.keys():
                rowX.append(time)
                rowY.append(int(state,2))
                rowZ.append(dictOfElements[state])
            else:
                rowX.append(time)
                rowY.append(int(state,2))
                rowZ.append(0)
        X.append(rowX)
        Y.append(rowY)
        Z.append(rowZ)

    return np.array(X,dtype=float), np.array(Y,dtype=float), np.array(Z,dtype=float)
```

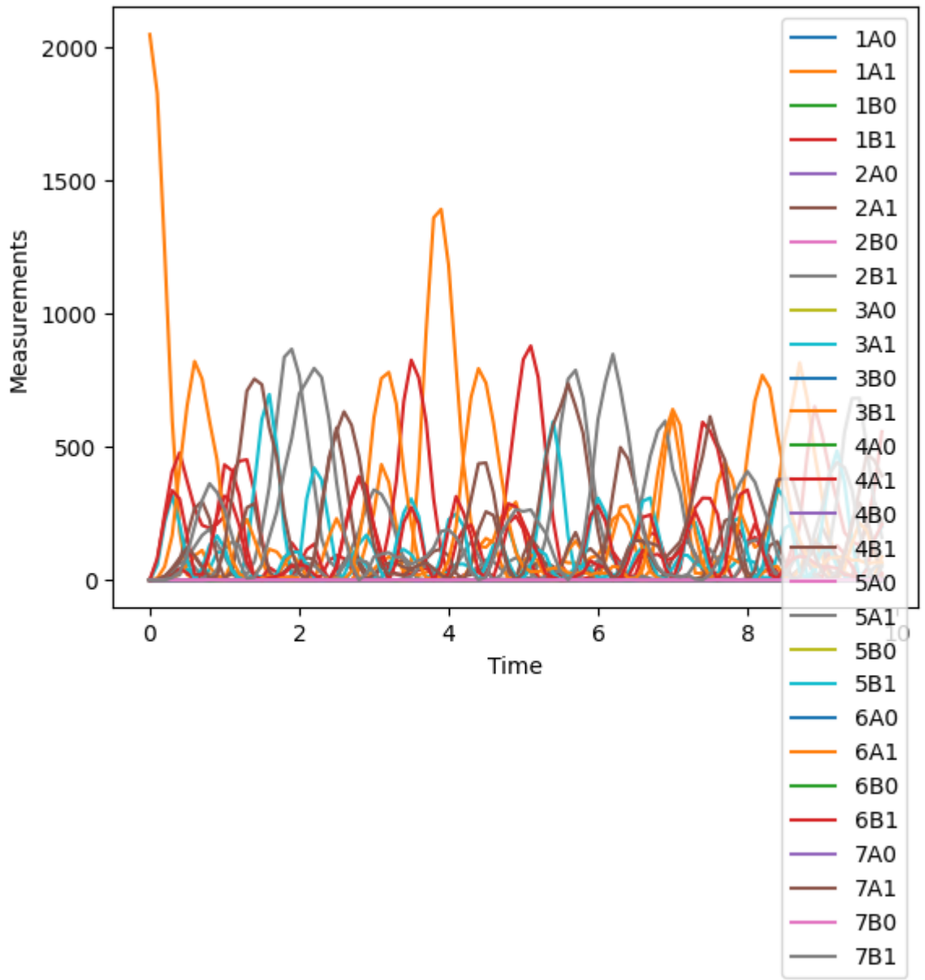
In [39]:
```python
def getStruturedData_seq(data, n_qubits):
    X = []
    Y = []
    Z = []

    if n_qubits == 0:
        for a in data:
            time, dictOfElements = a
            listOfItems = dictOfElements.items()
            for item in listOfItems:
                X.append(time)
                Y.append(int(item[0],2))
                Z.append(item[1])
        return np.array(X,dtype=float), np.array(Y,dtype=float), np.array(Z,dtype=float)
    else:
        states_list = binary_list(n_qubits)
        for a in data:
            time, dictOfElements = a
            listOfItems = dictOfElements.items()
            for state in states_list:
                if state in dictOfElements.keys():
                    X.append(time)
                    Y.append(int(state,2))
                    Z.append(dictOfElements[state])
                else:
                    X.append(time)
                    Y.append(int(state,2))
                    Z.append(0)
        return np.array(X,dtype=float), np.array(Y,dtype=float), np.array(Z,dtype=float)
```

And now we need to remove the last 4 quantum states as they refer to the 8th lattice, that doesn't exist:

In [40]:
```
def remove_111_seq(seq):
    sequence=[]
    n=0
    for t in time:
        sequence.append(list(seq[n:n+28]))
        n+=32
    return np.array(sum(sequence, []),dtype=float)
```

In [41]:
```
def remove_111_grid(grid):
    grid_new=[]
    for n in range(0,len(time)):
        grid_new.append(grid[n][0:28])
    return np.array(grid_new,dtype=float)
```

In [42]:
```
times,states,counts=getStruturedData_seq(data, n_qubits)

times=remove_111_seq(times)
states=remove_111_seq(states)
counts=remove_111_seq(counts)

basis=["1A0","1A1","1B0","1B1","2A0","2A1","2B0","2B1","3A0","3A1","3B0","3B1","4A0","4A1","4B0","4B1","5A0","5A1","5B0","5B1","6A0","6A1","6B0","6B1","7A0","7A1","7B0"

for i in range(len(basis)):
    index = np.where(states==i)
    plt.plot(times[index],counts[index],label=basis[i])
plt.xlabel("Time")
plt.ylabel("Measurements")
plt.legend()
plt.show()
```



Mandatory task 8: This task is an extension of Task 7.
Please provide images that illustrate the evolution of the probability of finding the particle in each lattice well or at each point in space (either representation will suffice). The purpose of these images is to observe how the probability of the particle diffuses throughout the lattice. Choose a suitable graphical representation that effectively visualizes this diffusion.

These images should be obtained from the data generated by the quantum simulator.

Reply:

If we want to represent the distribution of probability by the lattice we will need:

In [43]:
```
a=1
V0=100
```

Wave function

In [44]:
```
from scipy.optimize import brentq
from scipy.special import jv

# Function to define the transcendental equation for R(a)=0
def transcendental_eq(k,r):
    jv_r=jv(0, k*r)
    for i in range(len(r)):
        for j in range(len(r[0])):
            if r[i,j]>a:
                jv_r[i,j]=0
    return jv_r
```

The lattice

In [45]:
```
def hexagonal_lattice(n):
    x = np.zeros(3*n*(n-1) + 1)
    y = np.zeros(3*n*(n-1) + 1)
    for i in range(1, n):
        r = i * np.sqrt(3)
        for j in range(6):
            x[6 * (i - 1) + j] = r * np.cos((j-1) * np.pi / 3)
            y[6 * (i - 1) + j] = r * np.sin(-(j-1) * np.pi / 3)
    return x, y
```

A function to know the position of each well

In [46]:
```
def text_positions(X,Y):
    positions={}
    # Annotate each point with its number
    j=0
    for i, (xi, yi) in enumerate(zip(X, Y)):
        plt.text(xi, yi, str(i+1)+'B', color='black', fontsize=12, ha='center', va='center')
        plt.text(xi + np.cos(np.pi/6.0), yi - np.sin(np.pi/6.0), str(i+1)+'A', color='black', fontsize=12, ha='center', va='center')
        positions[j]=[xi + np.cos(np.pi/6.0), yi - np.sin(np.pi/6.0)]
        positions[j+1]=[xi + np.cos(np.pi/6.0), yi - np.sin(np.pi/6.0)]
        positions[j+2]=[xi, yi]
        positions[j+3]=[xi, yi]
        j+=4
    return positions
```

And a function to plot the probability distribution in the lattice at each time.

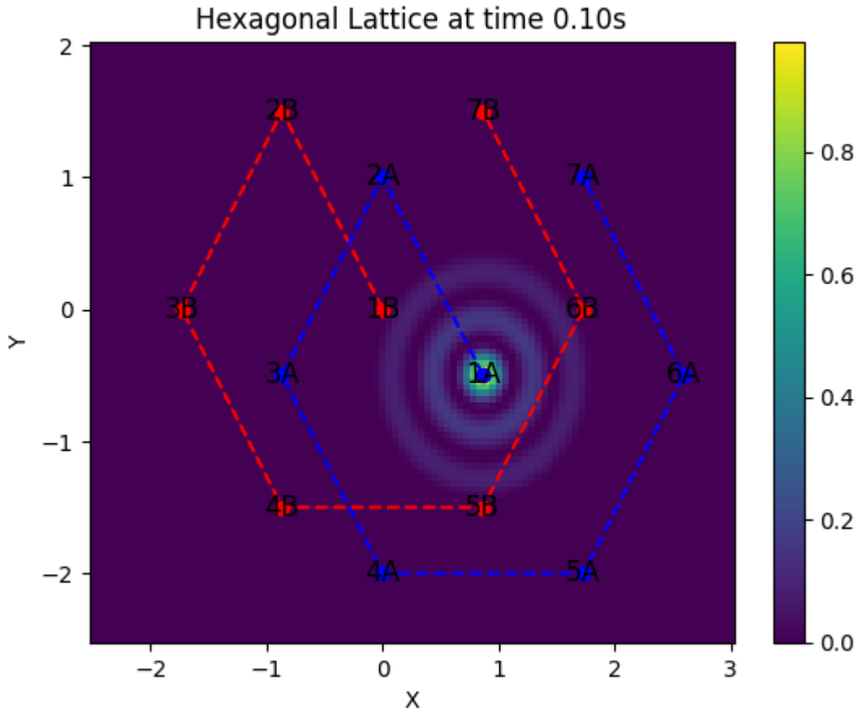
I took the probability distribution as:

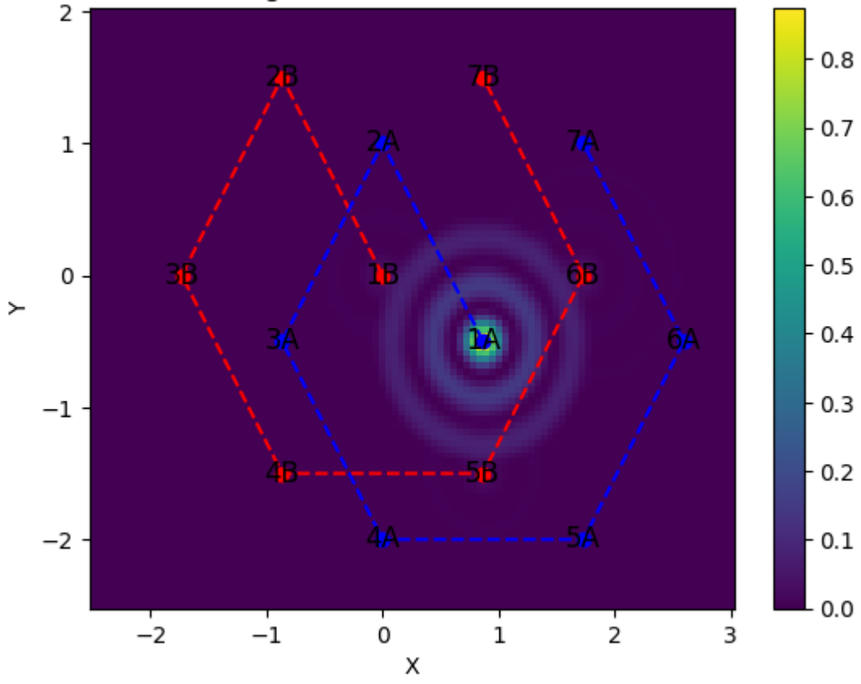
$$P(r) = |\psi_s|^2 P_{measure}$$

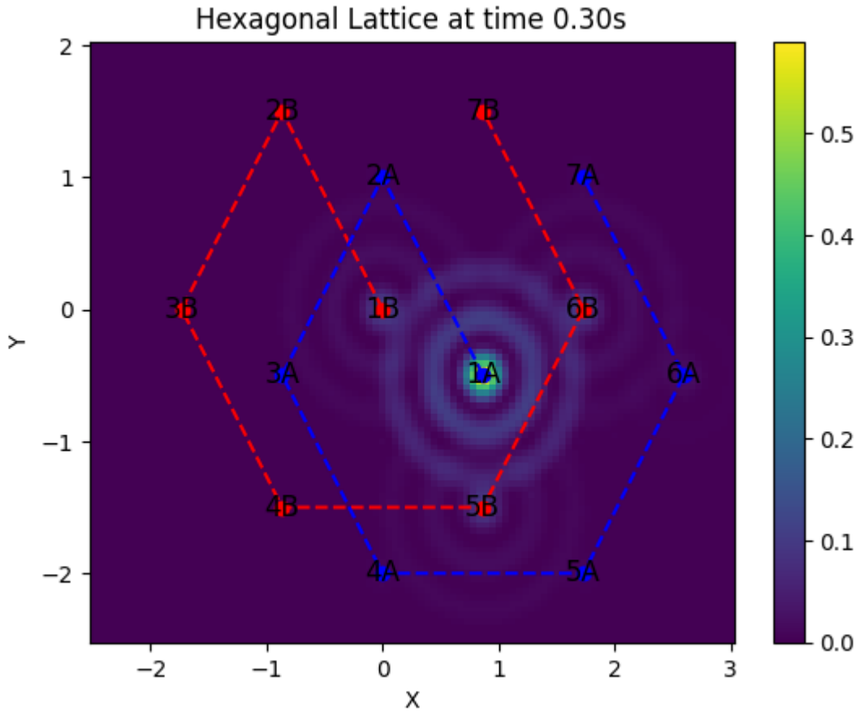
where ψ_s is the wave function at the s state (ground/excited), and $P_{measure}$ the probability of the particle being in the state measured.

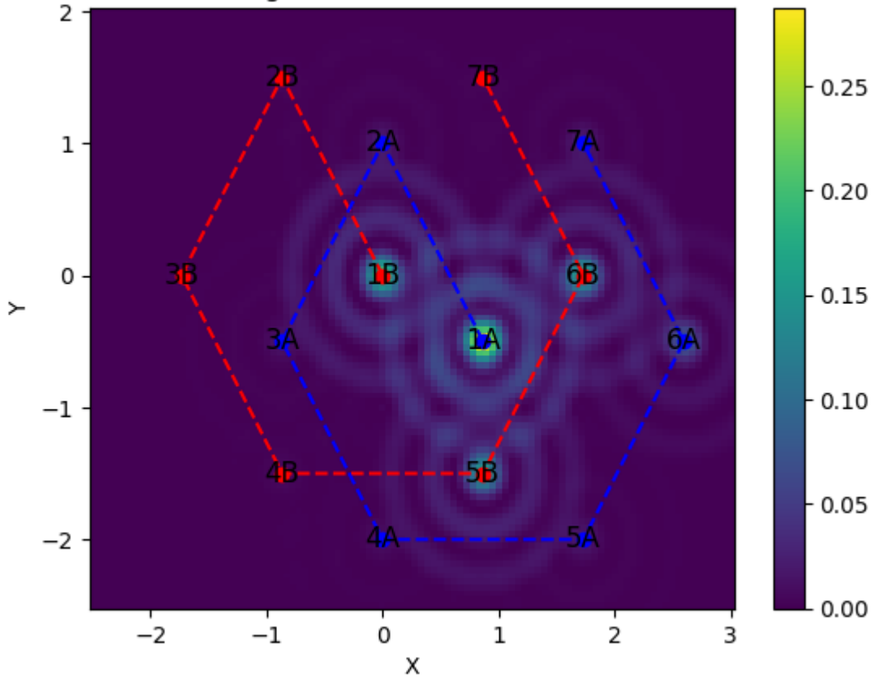
In [47]:
```
def plot_time(count,X_matrix,Y_matrix,time):
    X, Y = hexagonal_lattice(2)
    X = np.flip(X)
    Y = np.flip(Y)

    positions=text_positions(X,Y)

    plt.figsize=(8, 6)

    plt.plot(X, Y, 'ro--')
    plt.plot(X + np.cos(np.pi/6.0), Y - np.sin(np.pi/6.0), 'bo--')
    n=0
    for i in range(len(count)):
        loc=positions[i]
        if (i % 2) == 0:
            E=E0
        else:
            E=E1
        k=np.sqrt(E+V0)
        if n==0:
            Prob=transcendental_eq(k,np.sqrt((X_matrix-loc[0])**2+(Y_matrix-loc[1])**2))**2*(count[i]/shots)
            n=1
        else:
            Prob+=transcendental_eq(k,np.sqrt((X_matrix-loc[0])**2+(Y_matrix-loc[1])**2))**2*(count[i]/shots)
        plt.pcolormesh(X_matrix,Y_matrix,Prob)

    plt.colorbar()
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title('Hexagonal Lattice at time {:.2f}s'.format(time))
    plt.show()
```

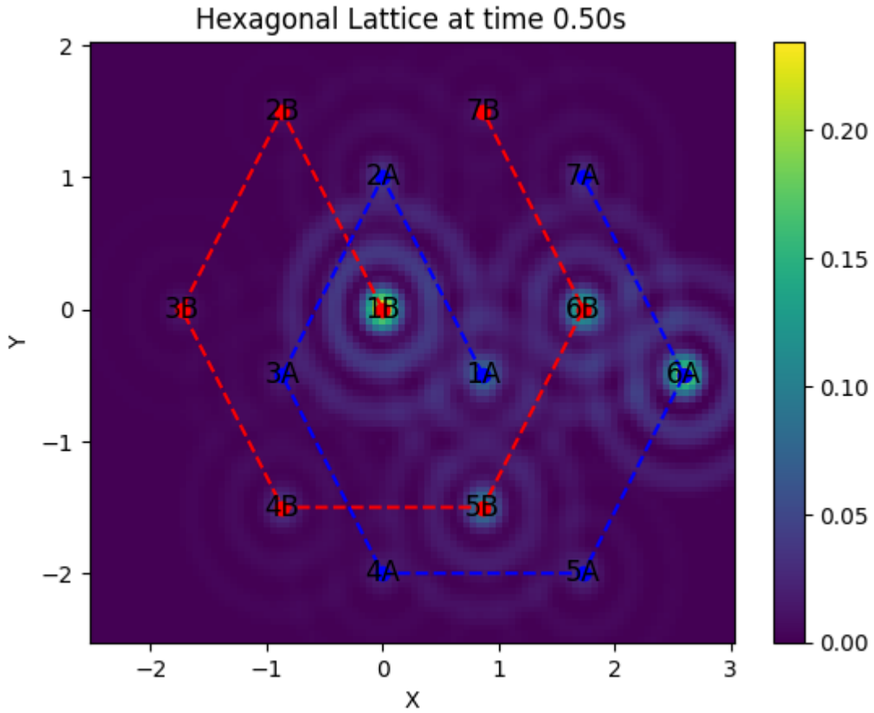
In [48]:
```
times,states,counts=getStruturedData_grid(data, n_qubits)

times=remove_111_grid(times)
states=remove_111_grid(states)
counts=remove_111_grid(counts)

x=np.linspace(-2.5,3,100)
y=np.linspace(-2.5,2,100)

X, Y = np.meshgrid(x,y)
i=0
for count in counts:
    plot_time(count,X,Y,time[i])
    i+=1
```

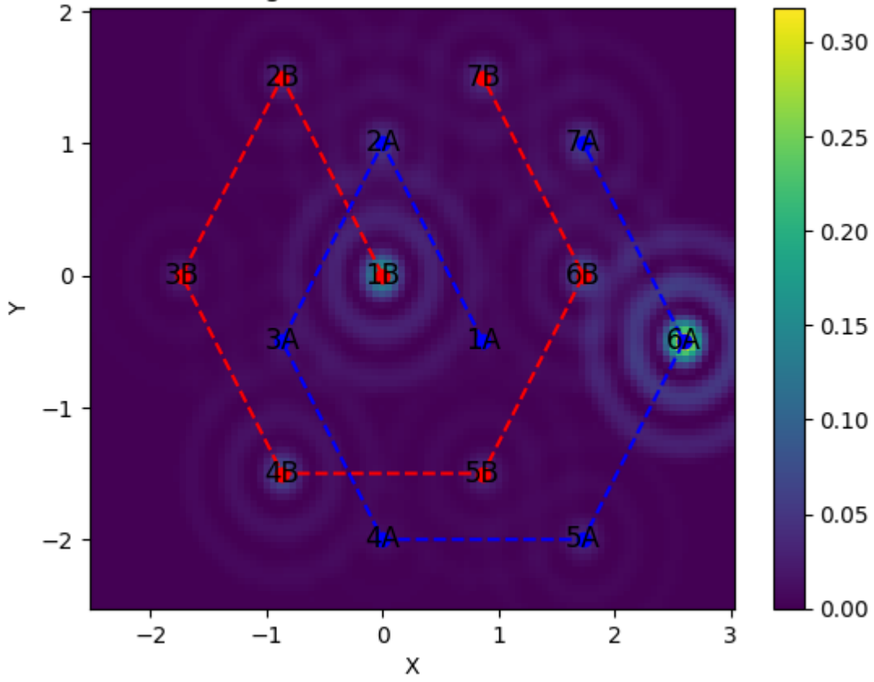

Hexagonal Lattice at time 0.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

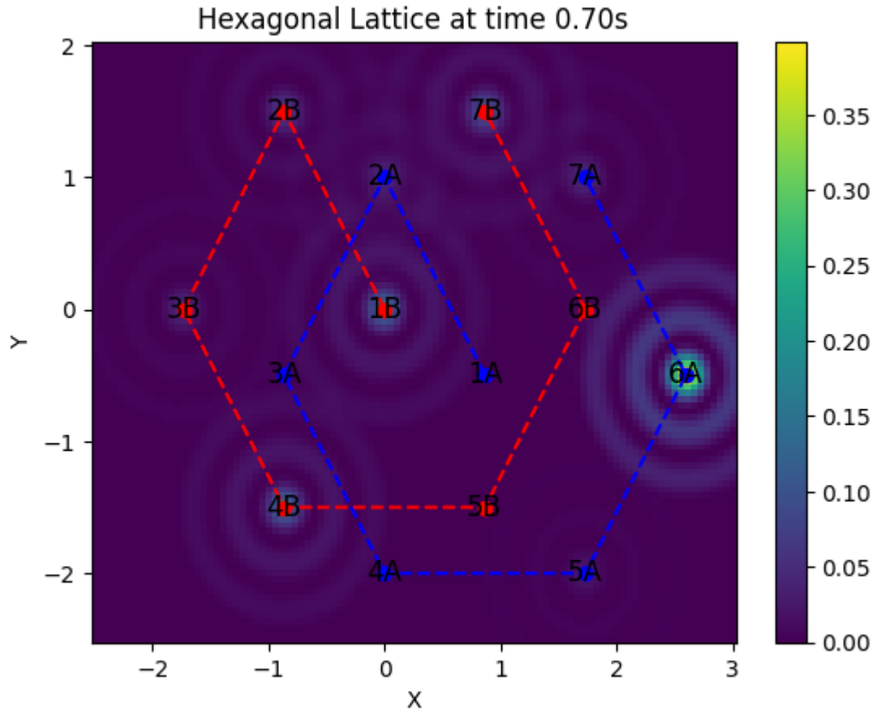

Hexagonal Lattice at time 0.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

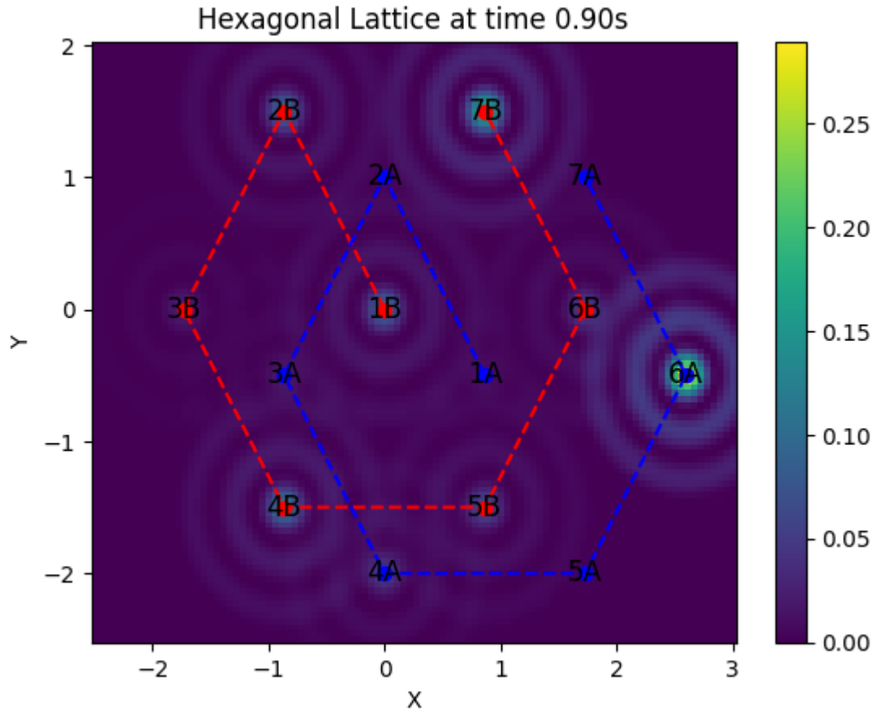

Hexagonal Lattice at time 0.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

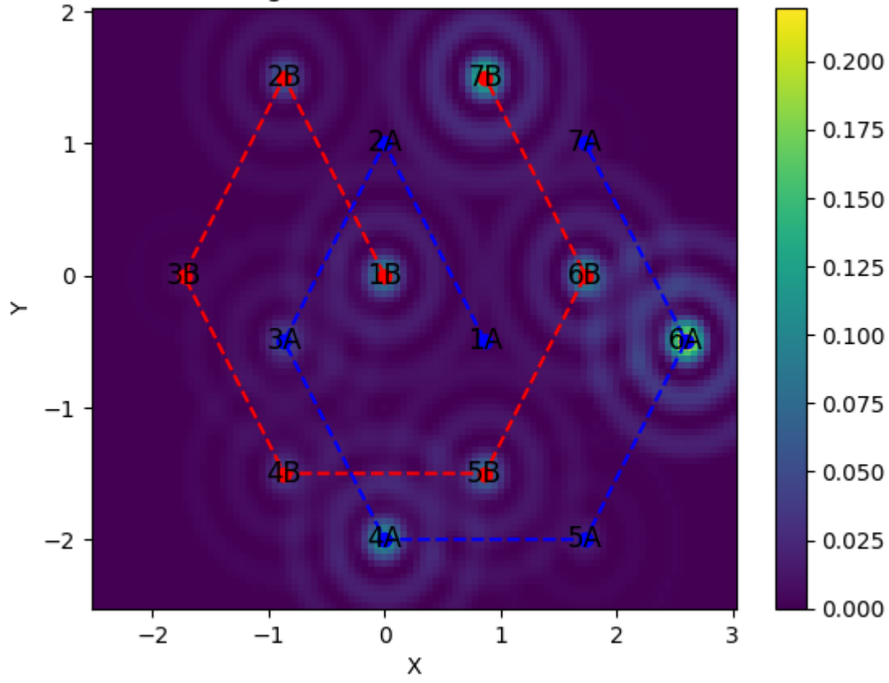

Hexagonal Lattice at time 0.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

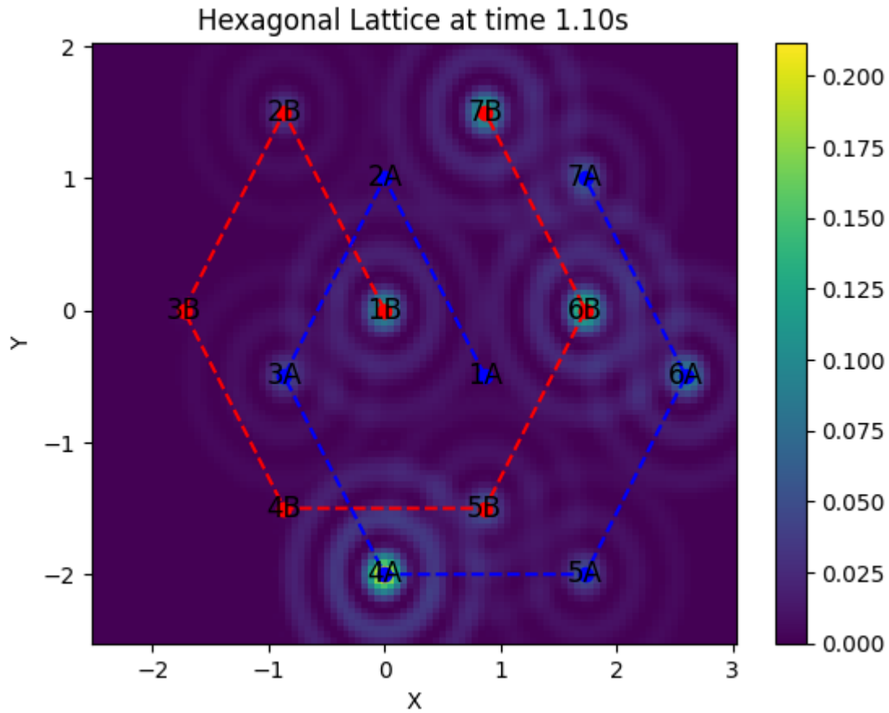

Hexagonal Lattice at time 0.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

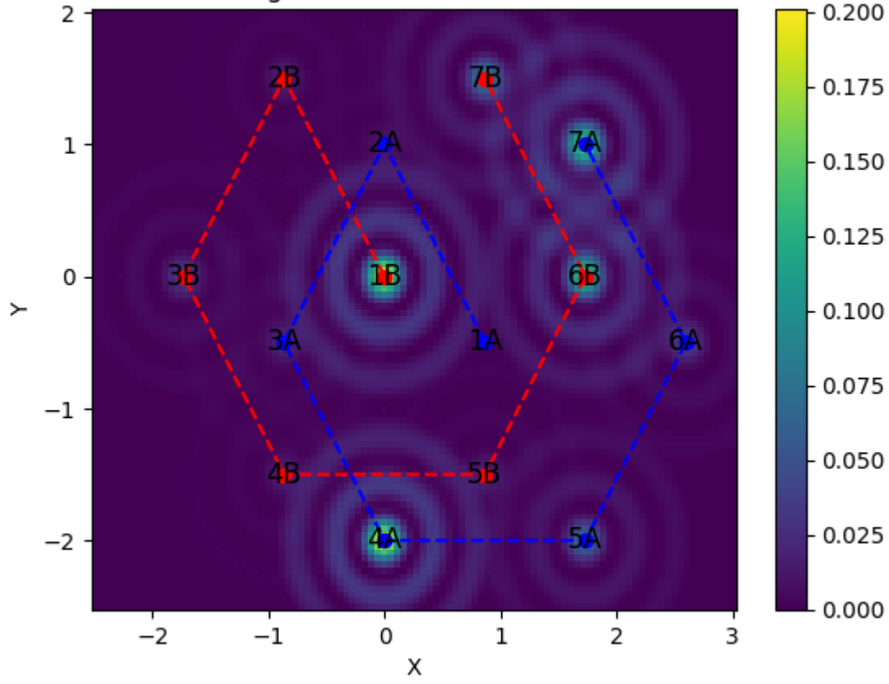

Hexagonal Lattice at time 0.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

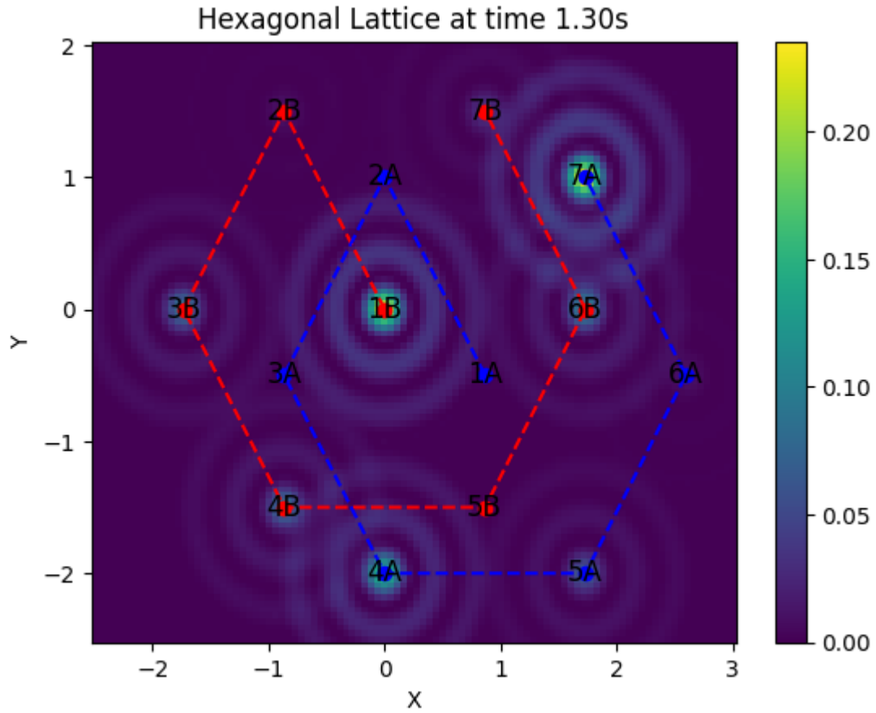

Hexagonal Lattice at time 0.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 0.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

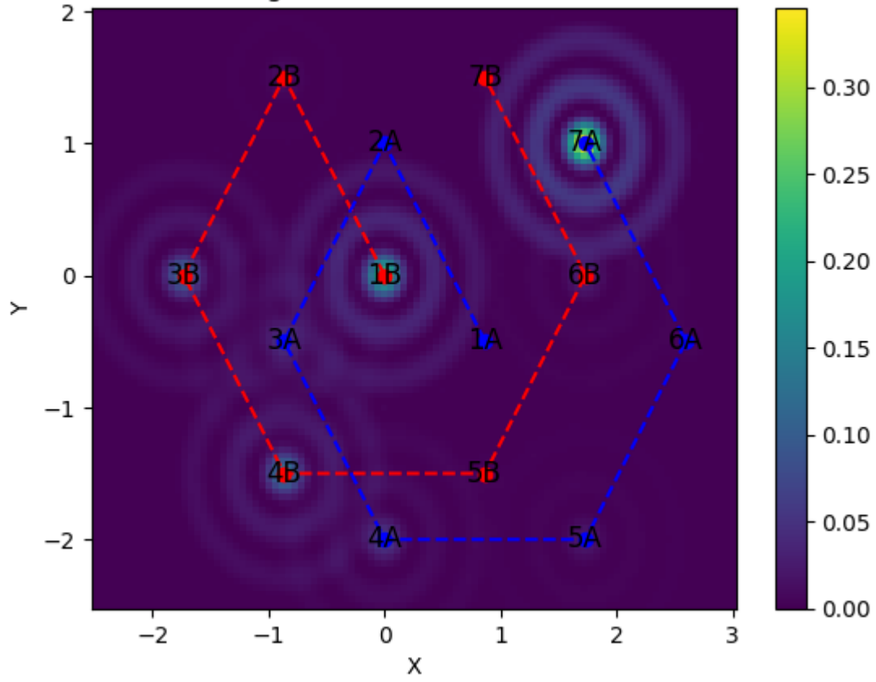

Hexagonal Lattice at time 0.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

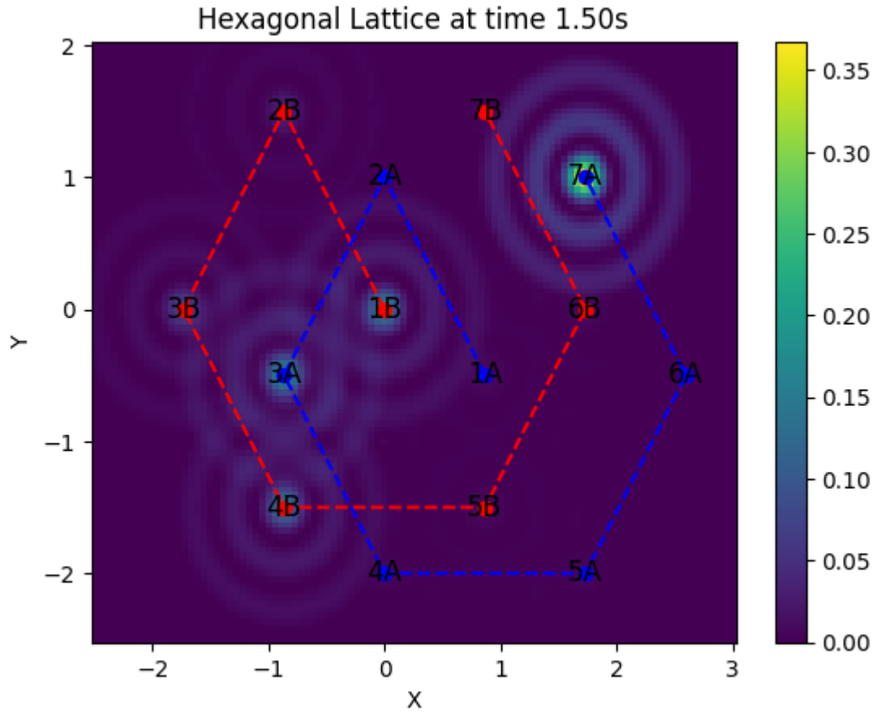

Hexagonal Lattice at time 1.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

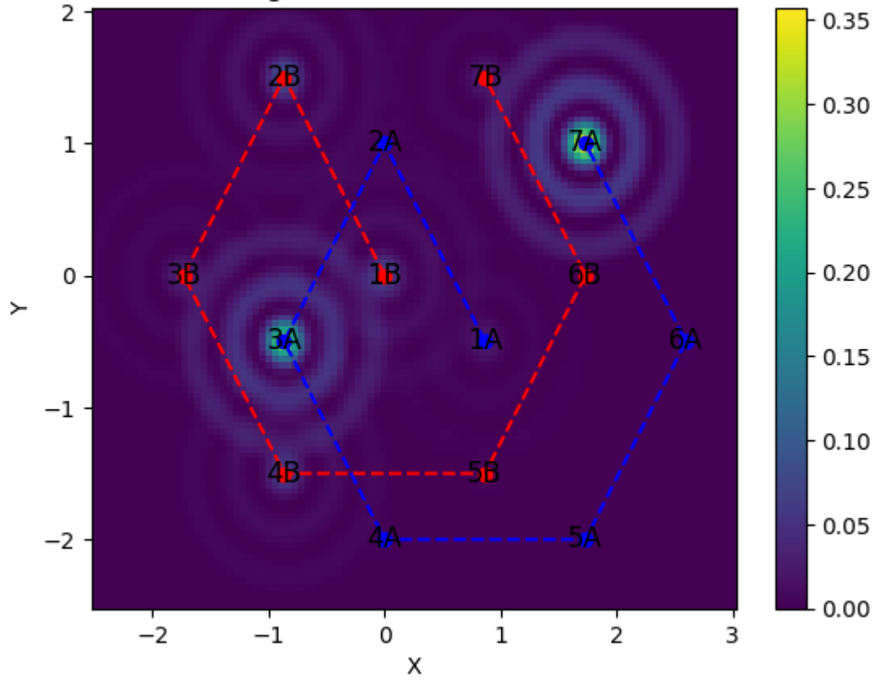

Hexagonal Lattice at time 1.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

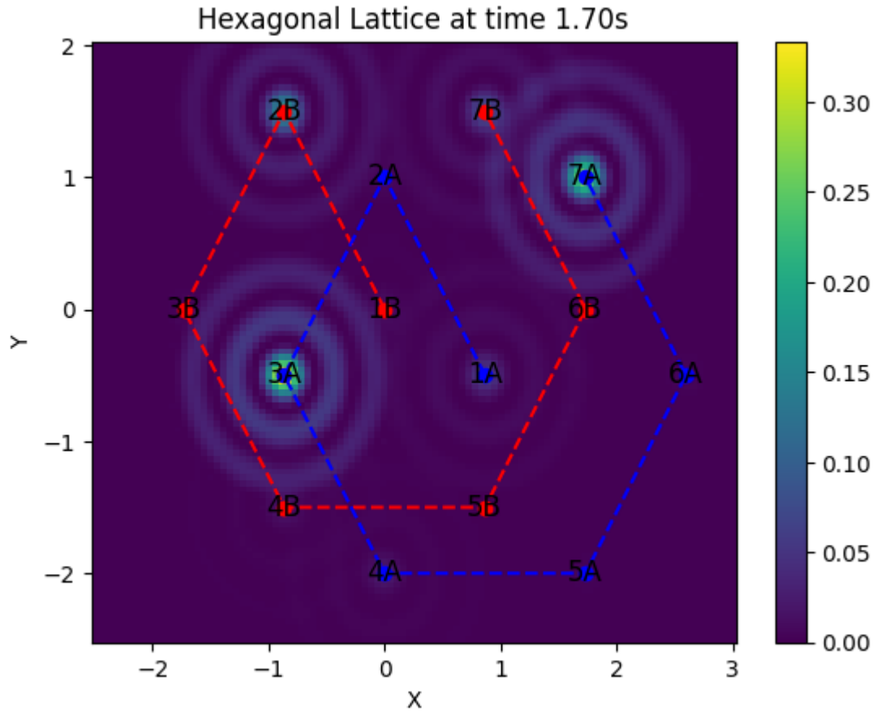

Hexagonal Lattice at time 1.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

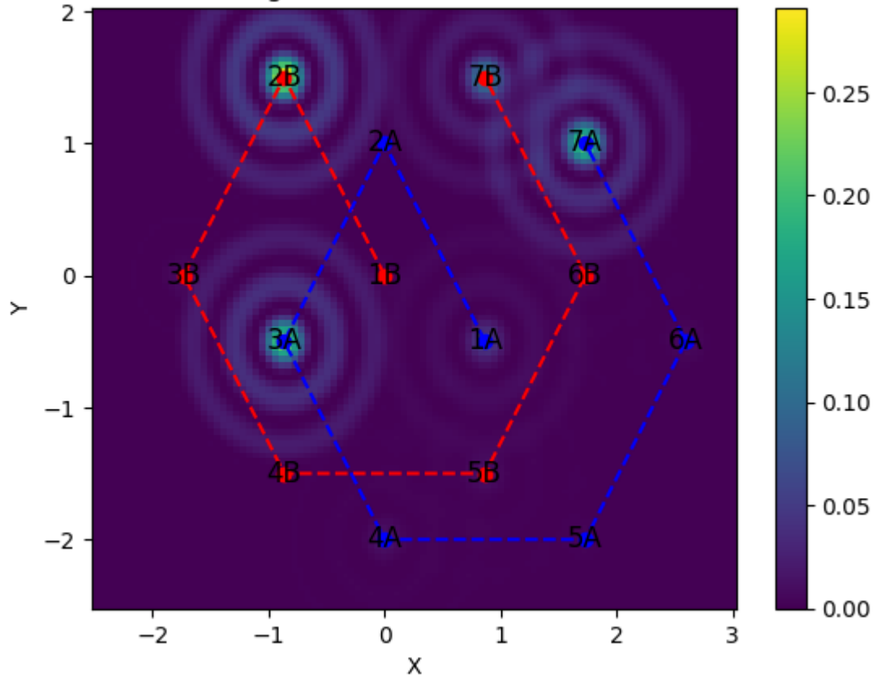

Hexagonal Lattice at time 1.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

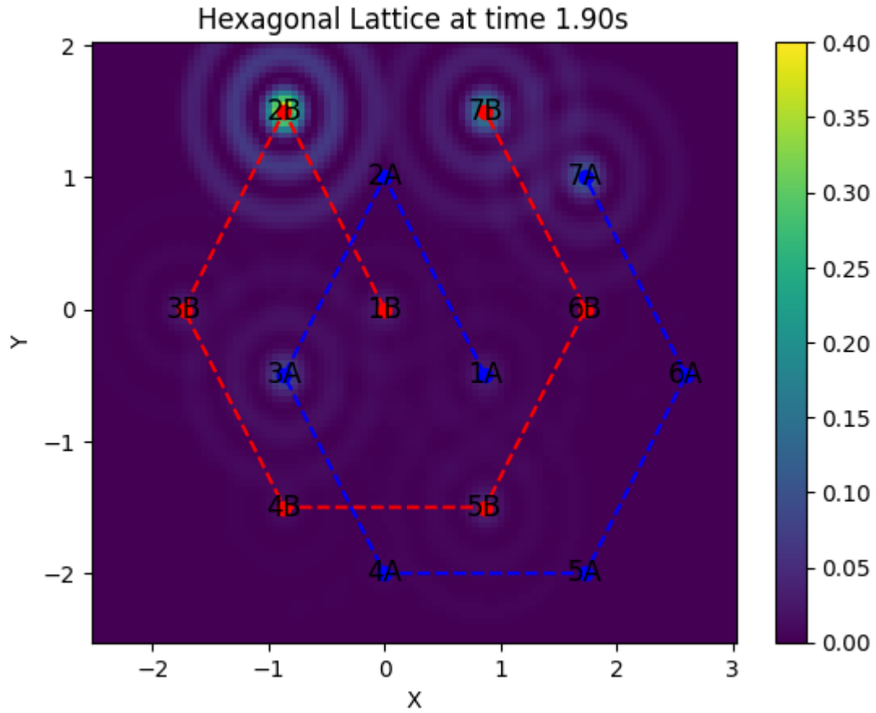

Hexagonal Lattice at time 1.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

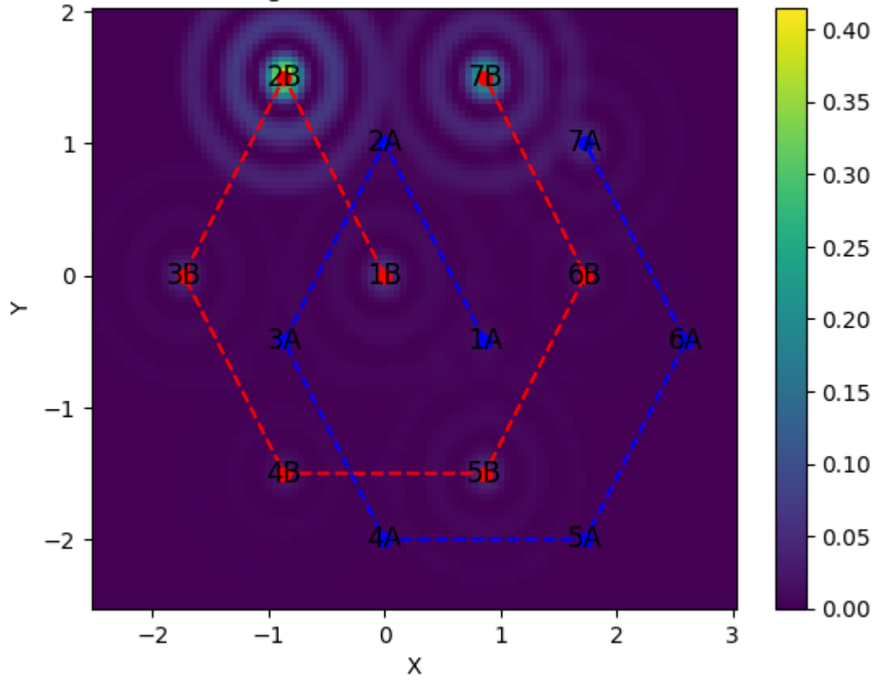

Hexagonal Lattice at time 1.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

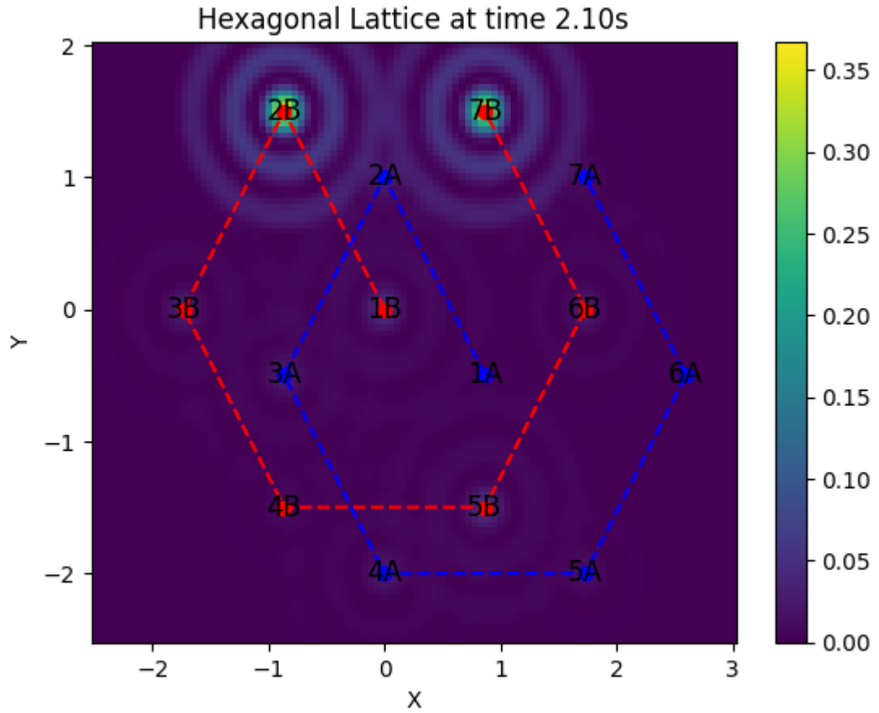

Hexagonal Lattice at time 1.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

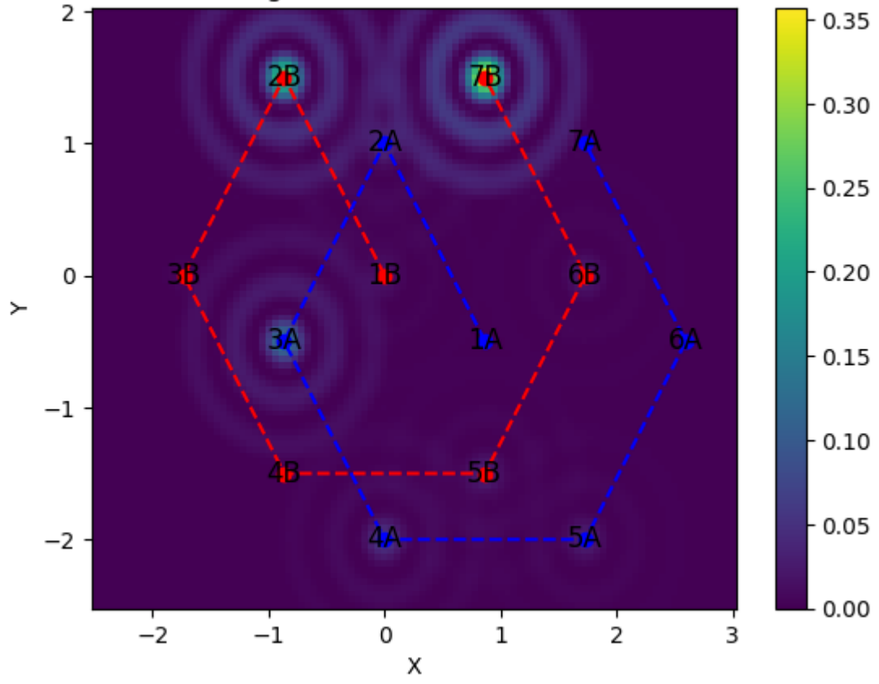

Hexagonal Lattice at time 1.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

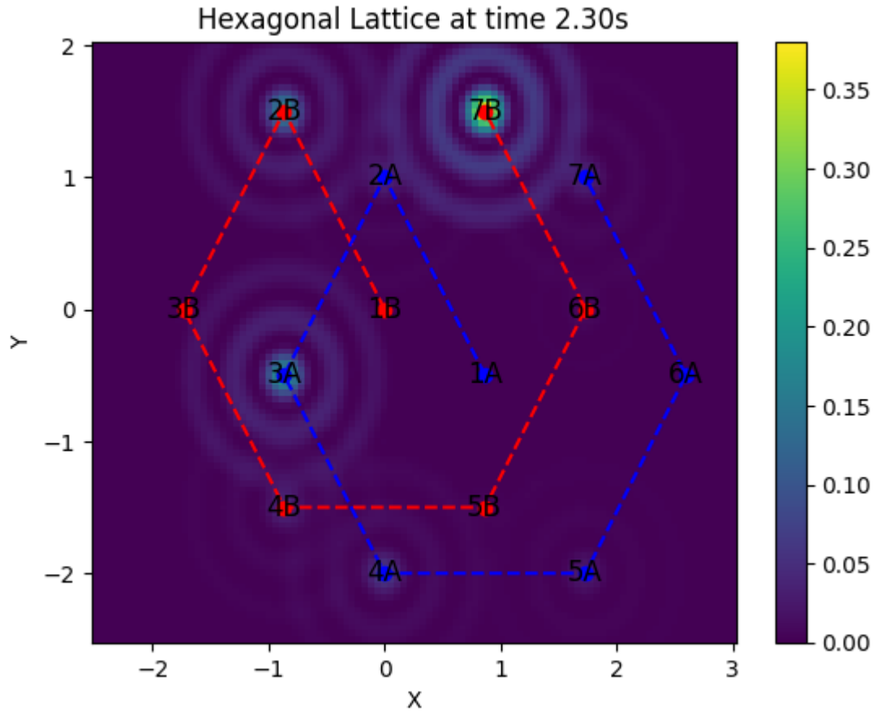

Hexagonal Lattice at time 1.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

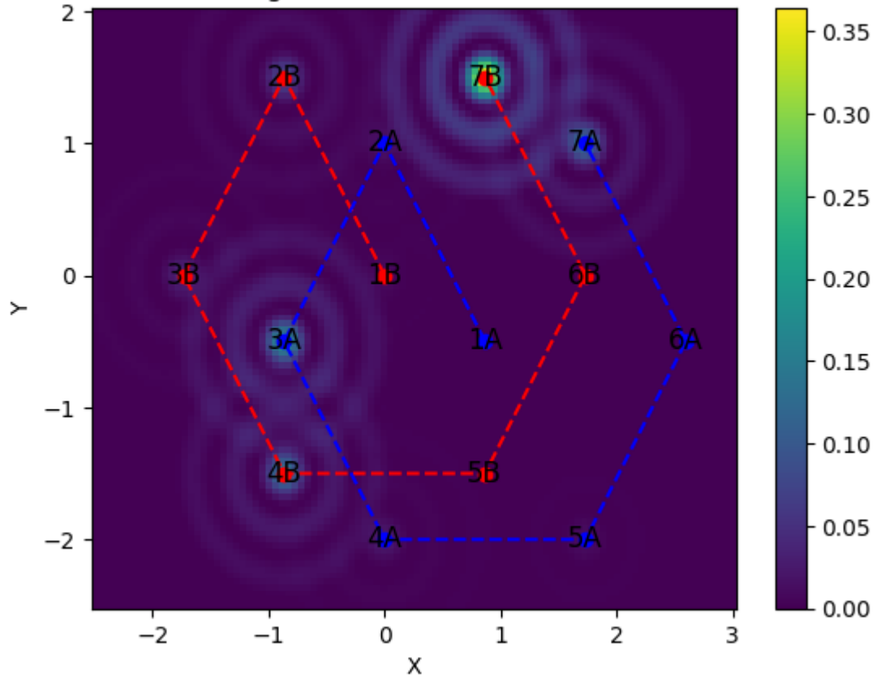

Hexagonal Lattice at time 1.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

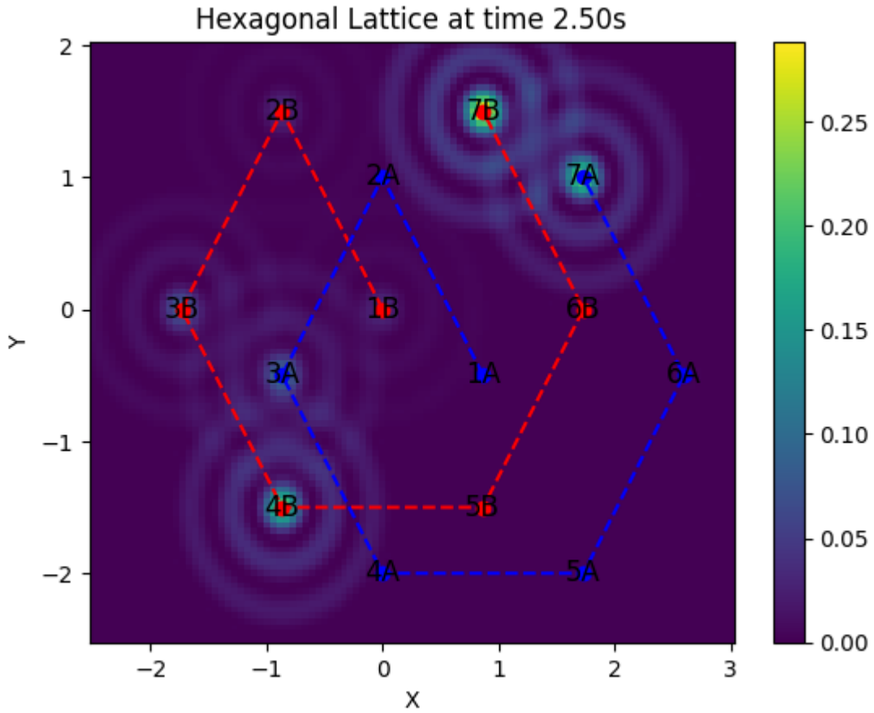

Hexagonal Lattice at time 2.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

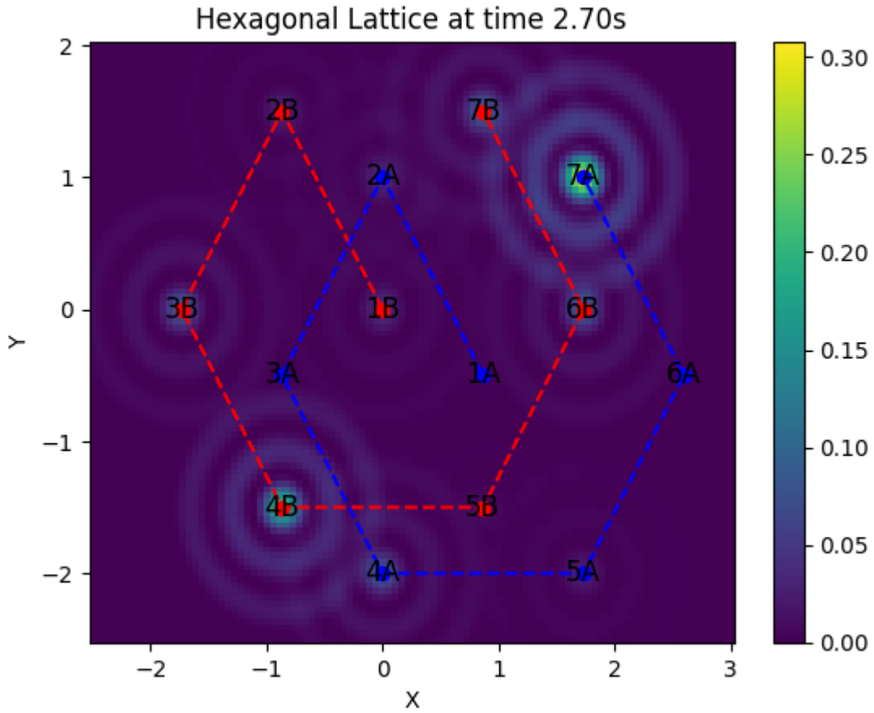

Hexagonal Lattice at time 2.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

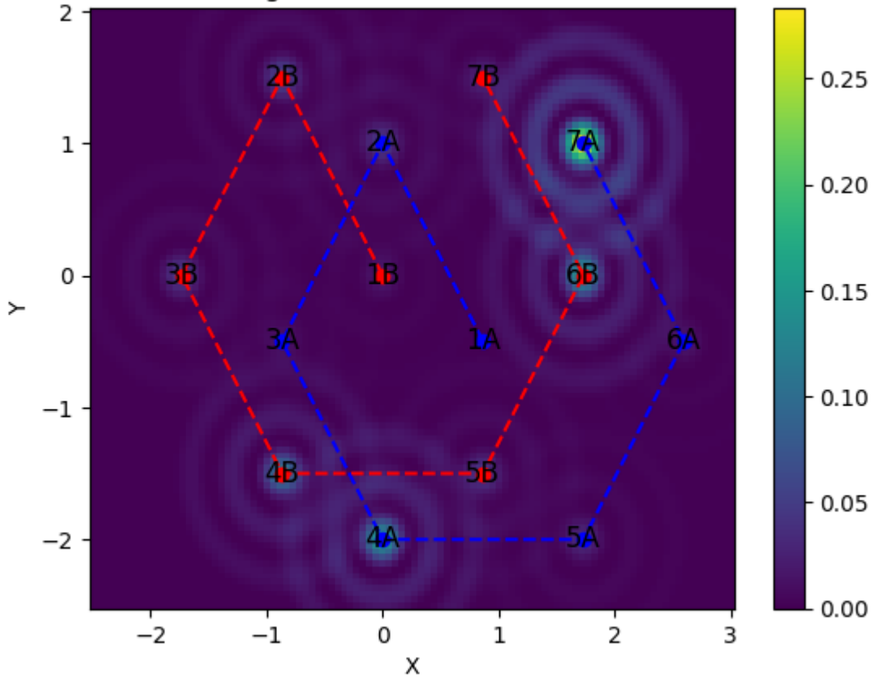

Hexagonal Lattice at time 2.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

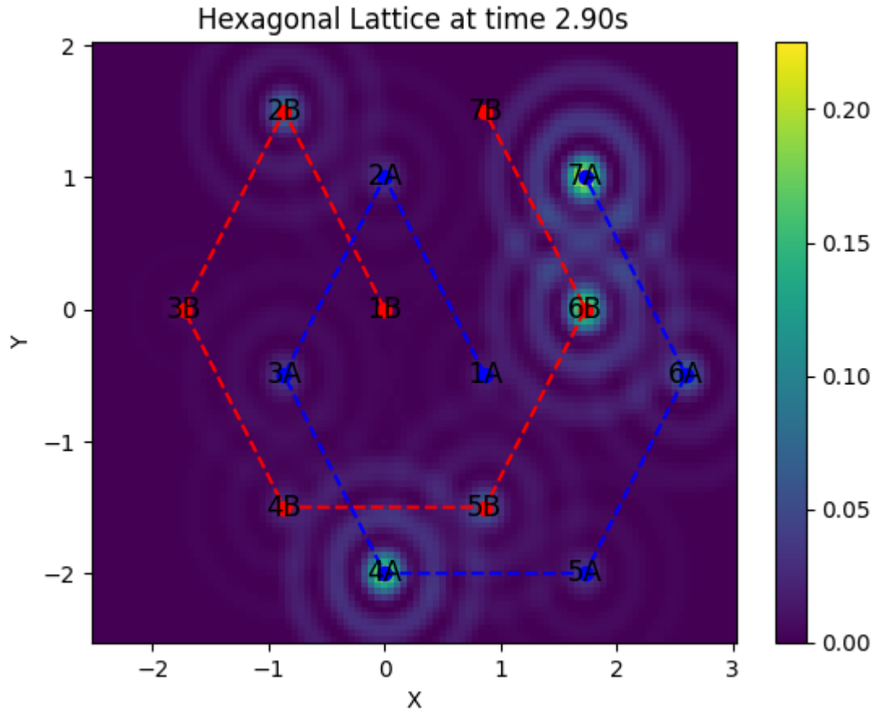

Hexagonal Lattice at time 2.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

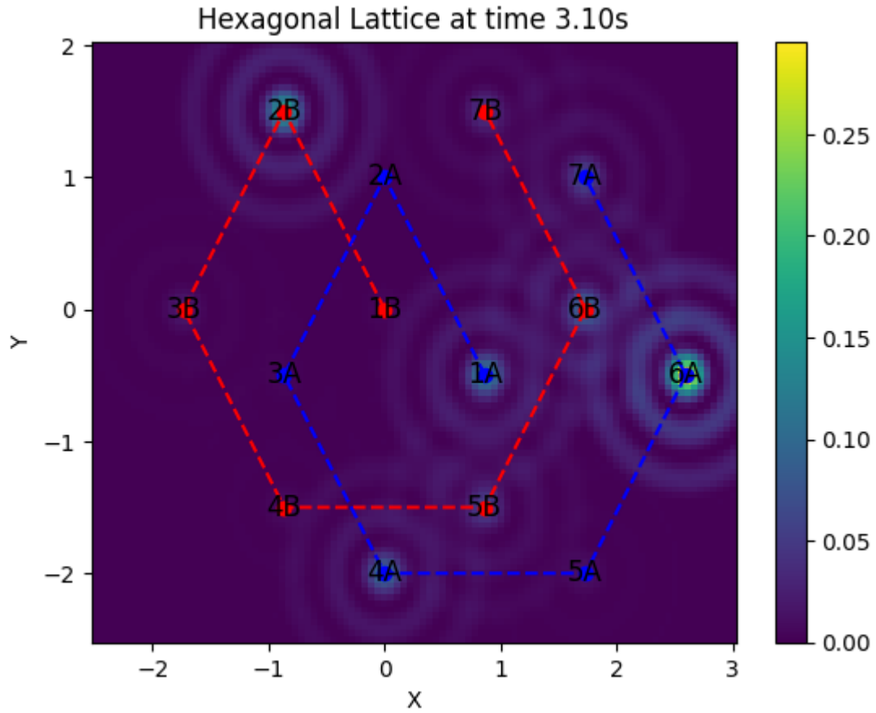

Hexagonal Lattice at time 2.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

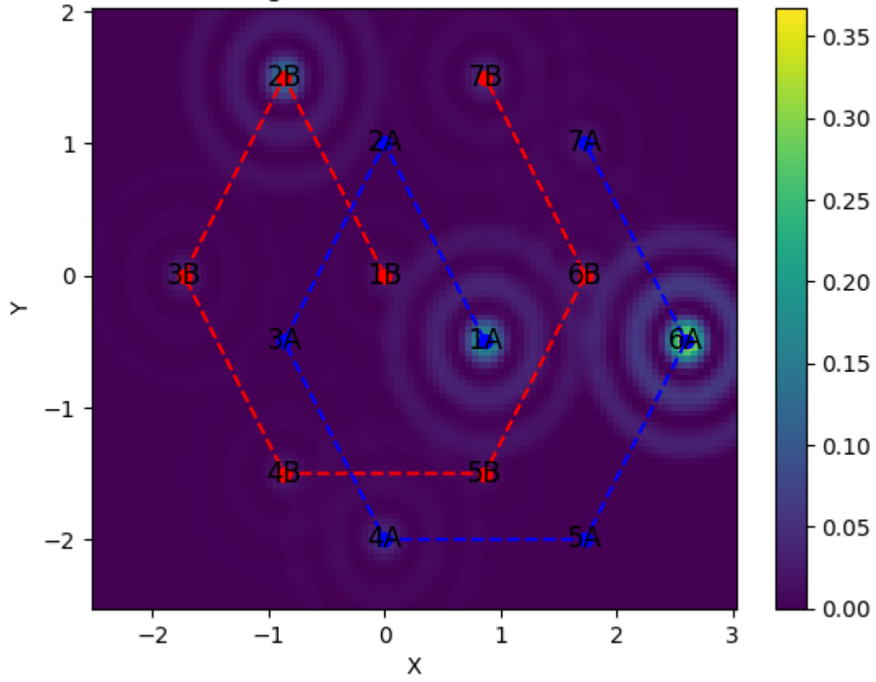

Hexagonal Lattice at time 2.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 2.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

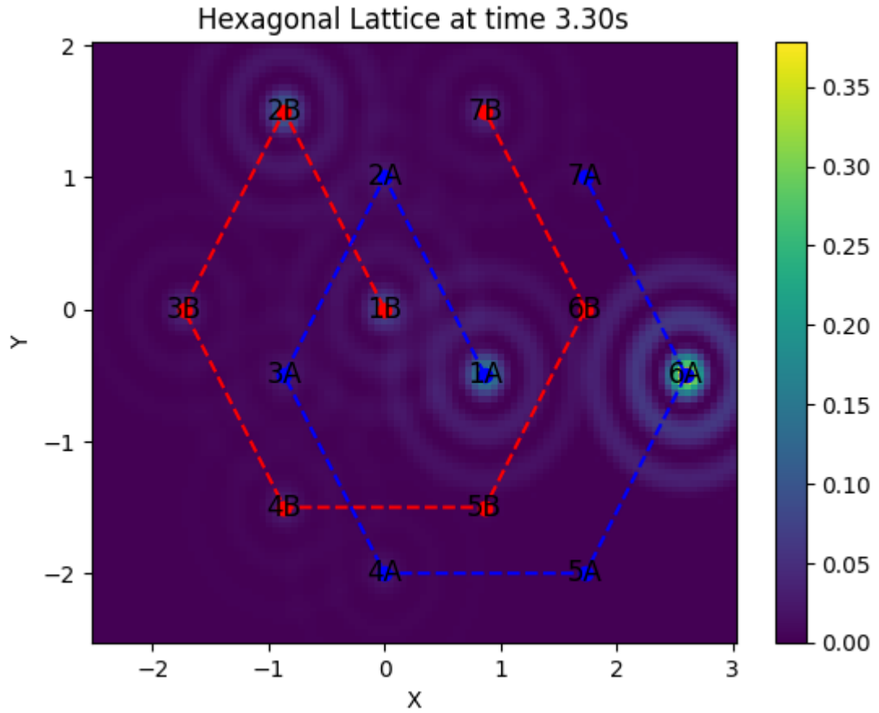

Hexagonal Lattice at time 2.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

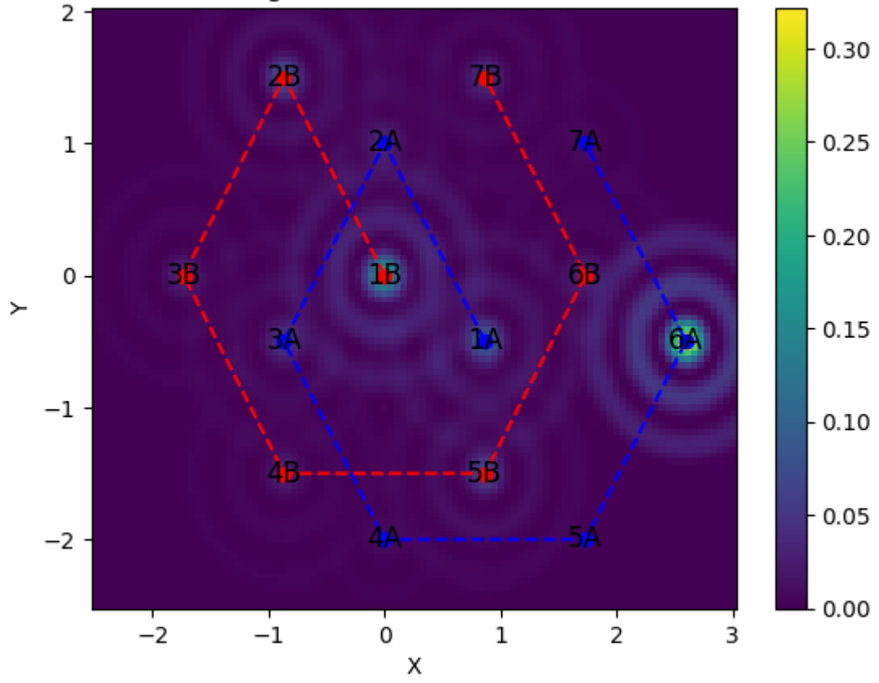

Hexagonal Lattice at time 2.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

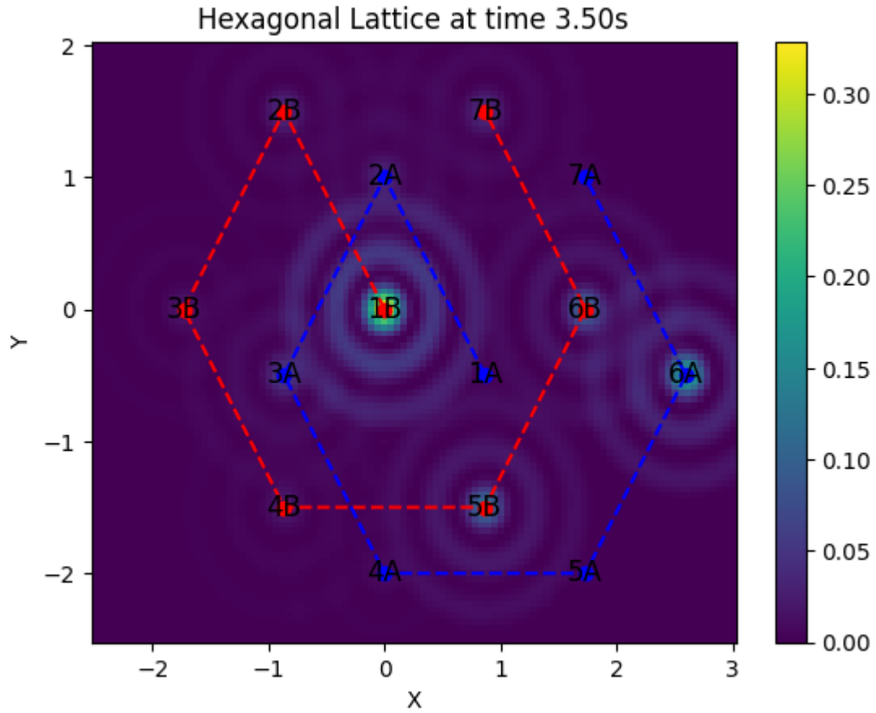

Hexagonal Lattice at time 2.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 3.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

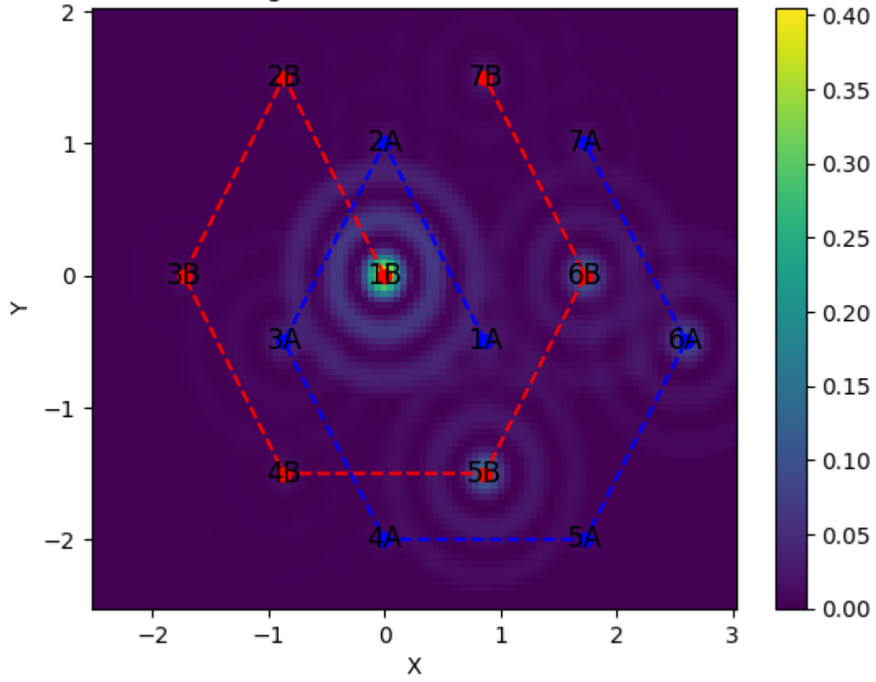

Hexagonal Lattice at time 3.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

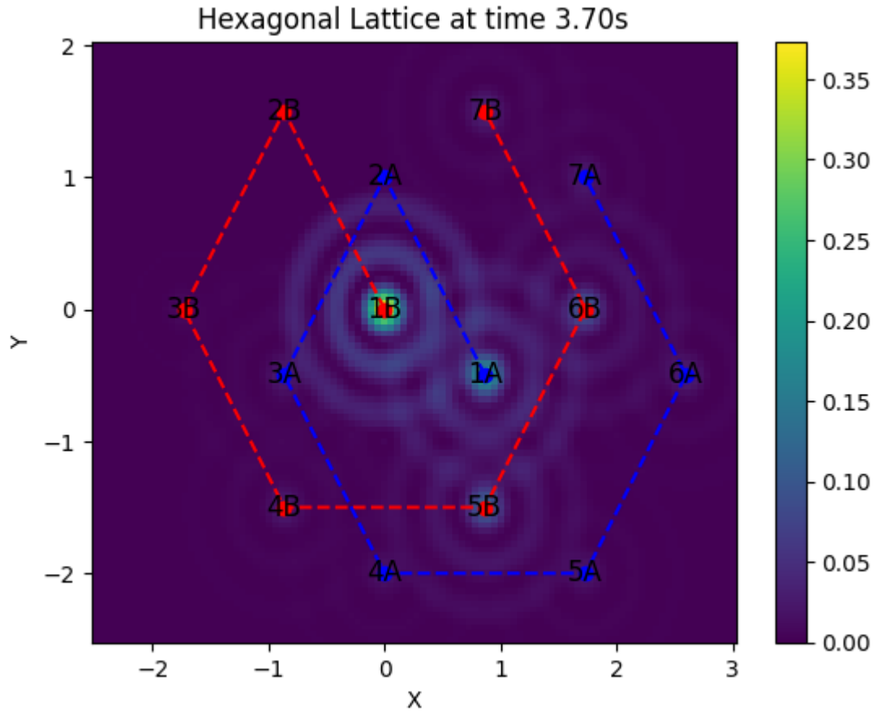

Hexagonal Lattice at time 3.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

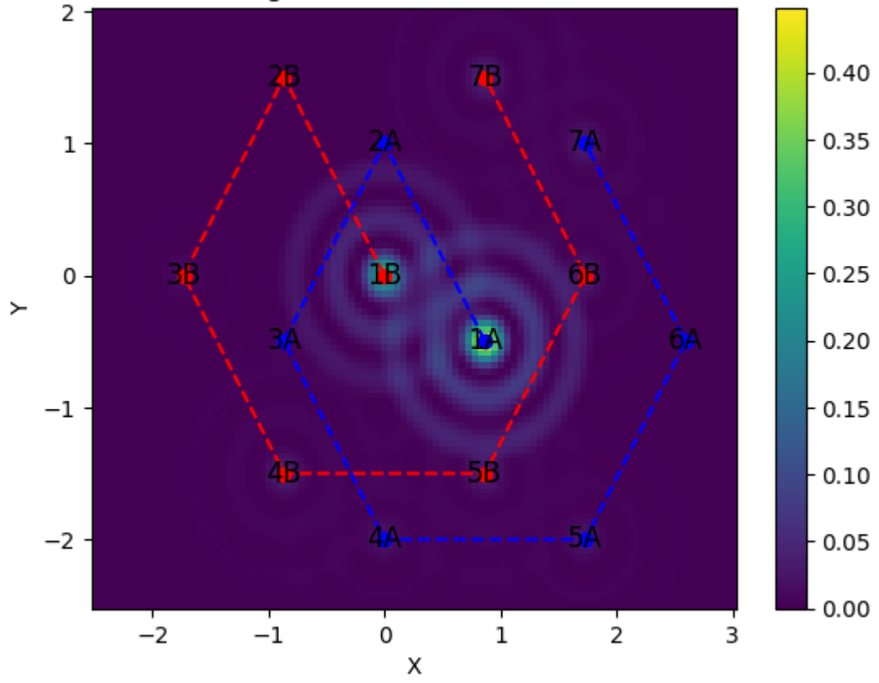

Hexagonal Lattice at time 3.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

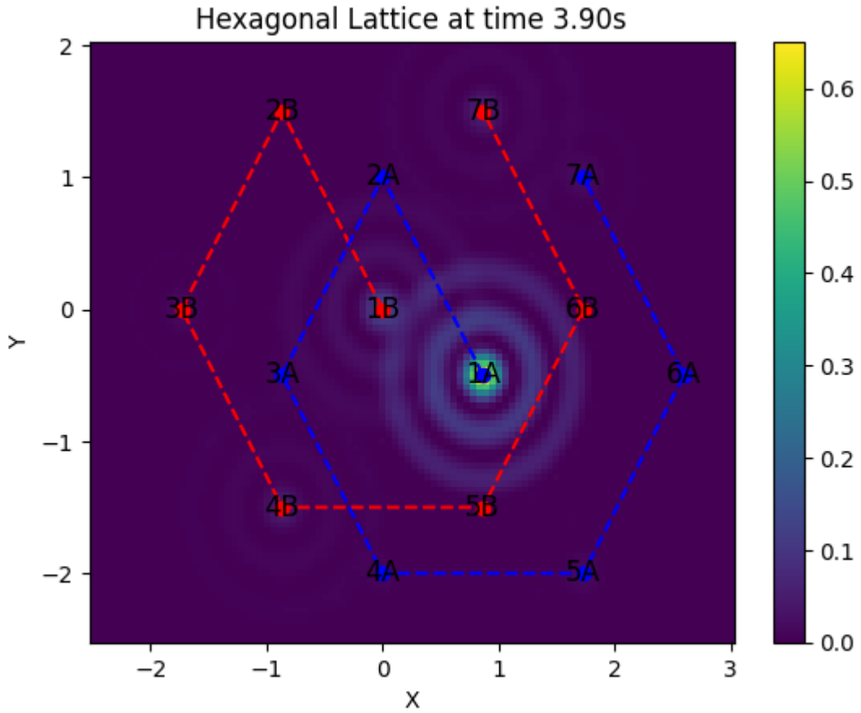

Hexagonal Lattice at time 3.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

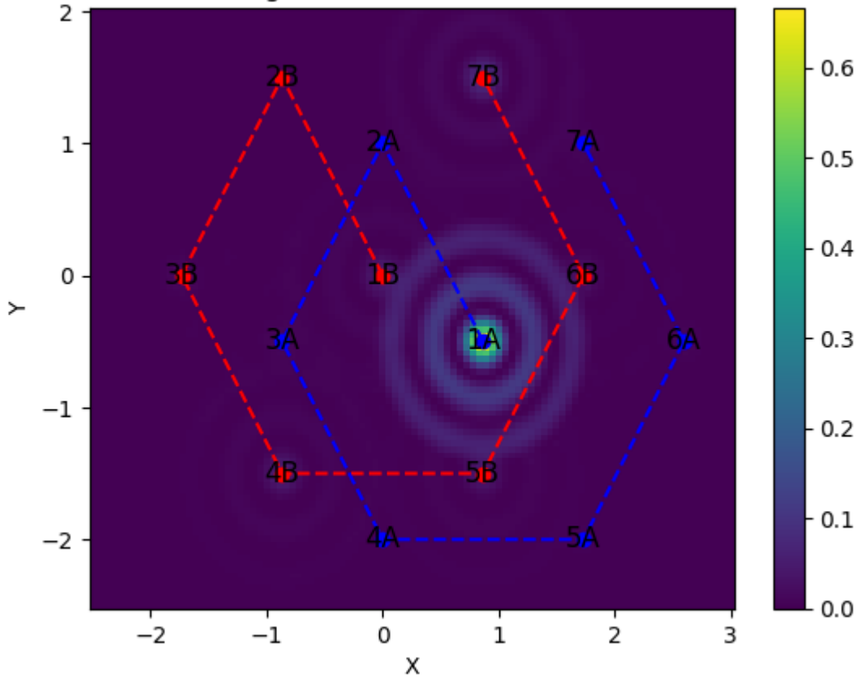

Hexagonal Lattice at time 3.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

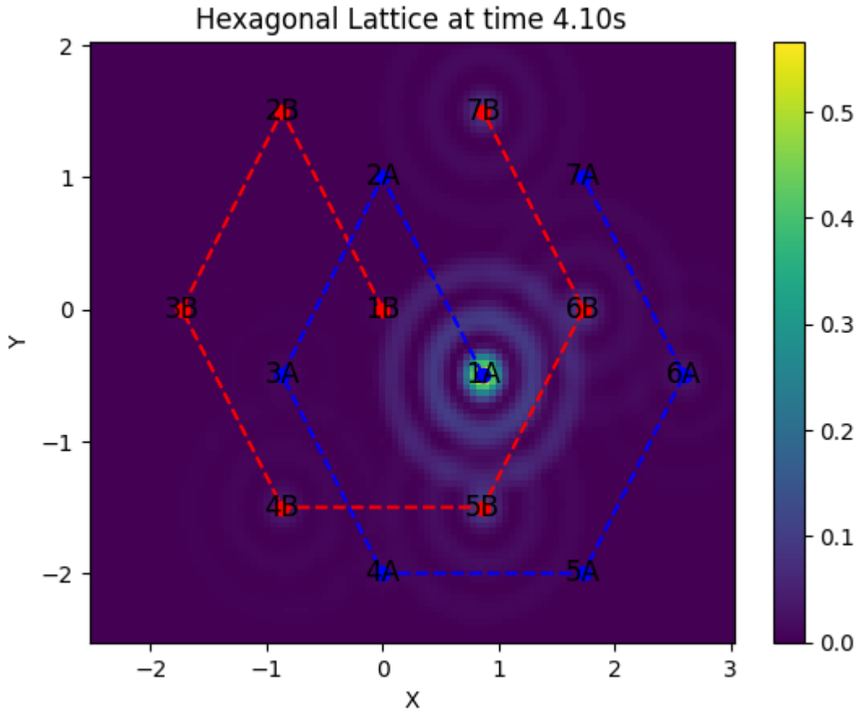

Hexagonal Lattice at time 3.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

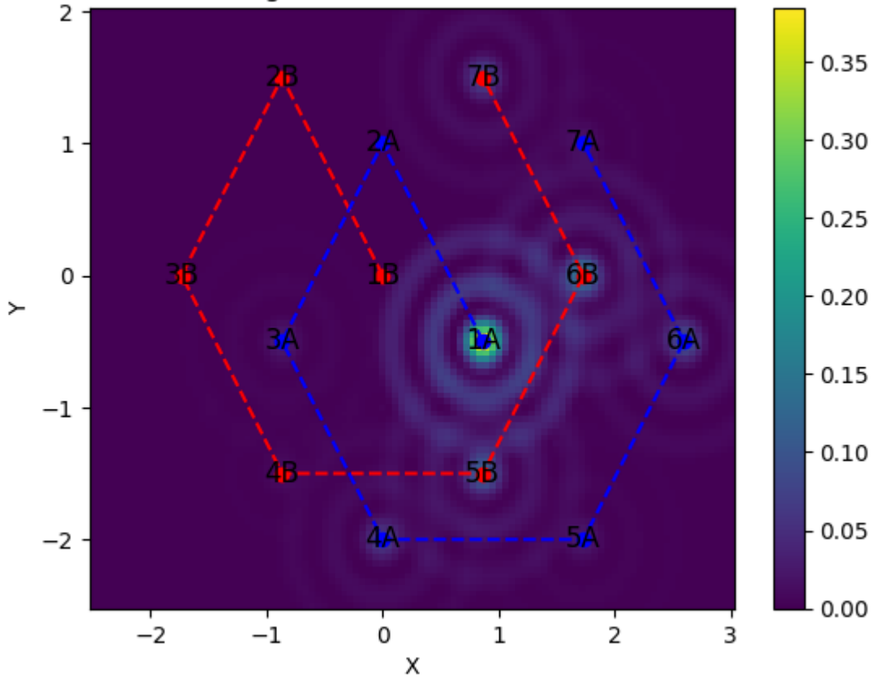

Hexagonal Lattice at time 3.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

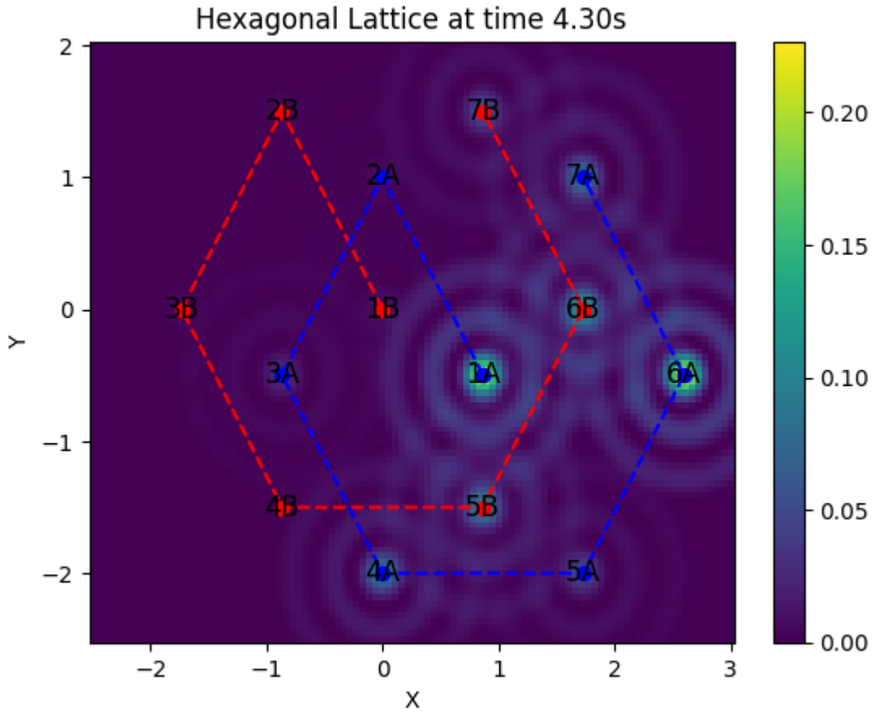

Hexagonal Lattice at time 3.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

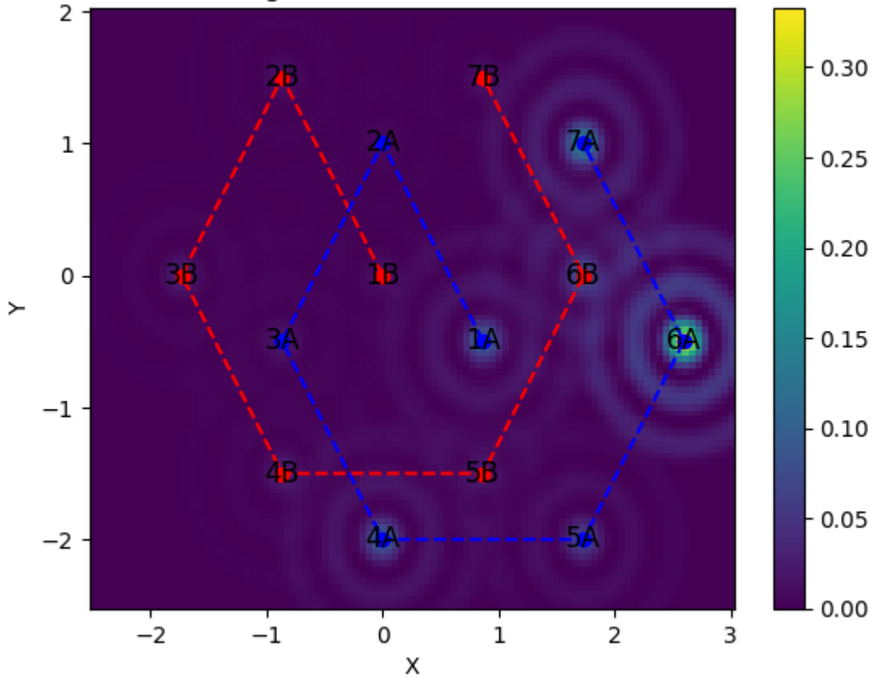

Hexagonal Lattice at time 3.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

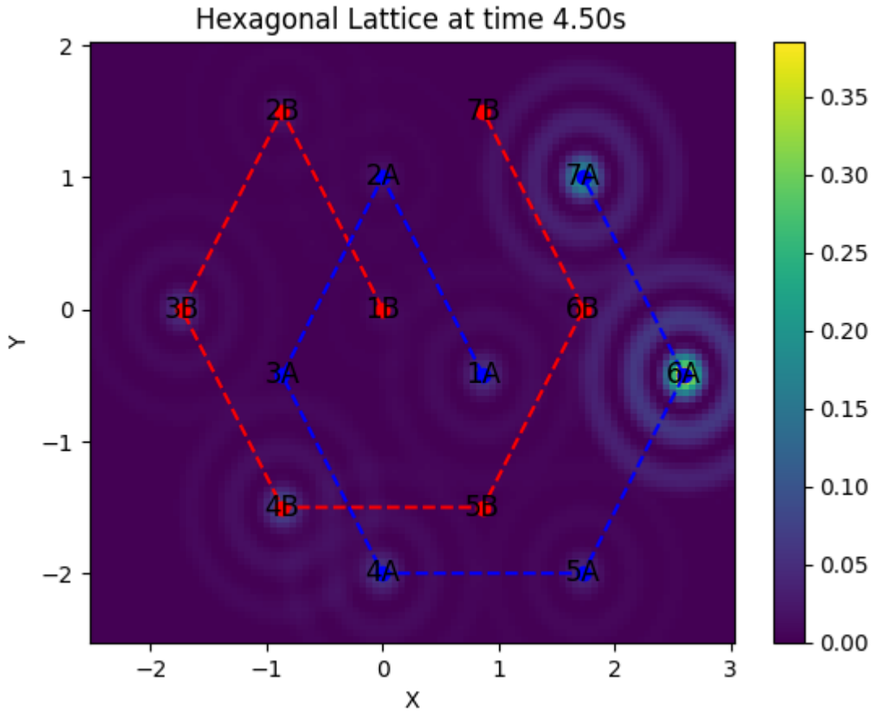

Hexagonal Lattice at time 4.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

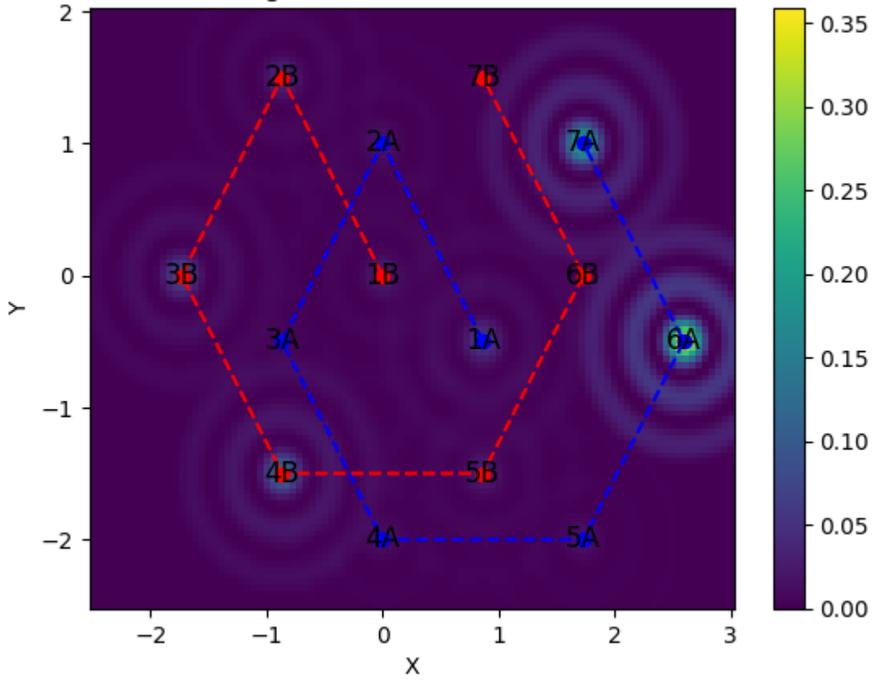

Hexagonal Lattice at time 4.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

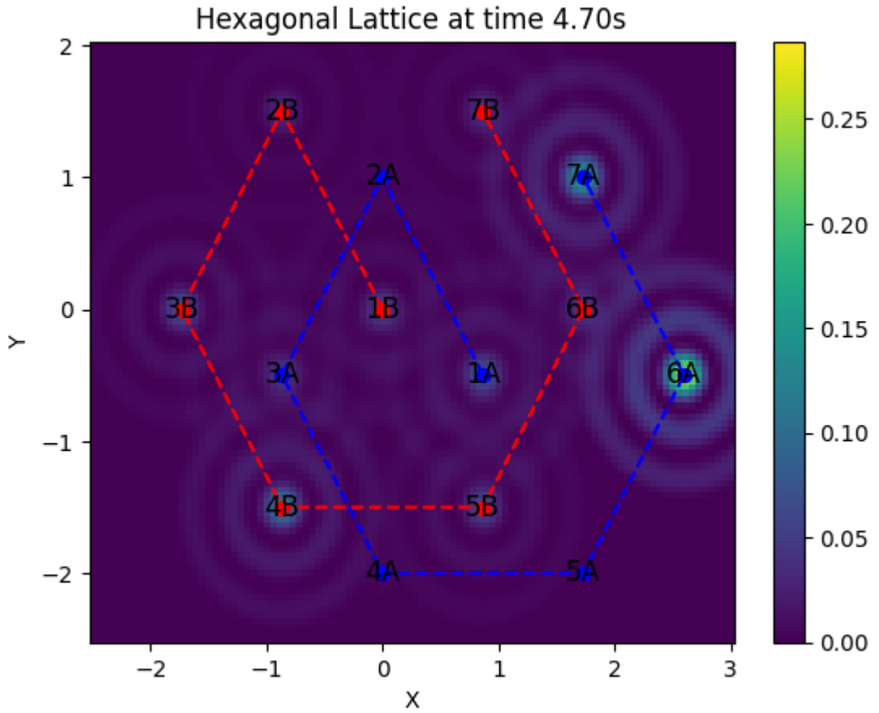

Hexagonal Lattice at time 4.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

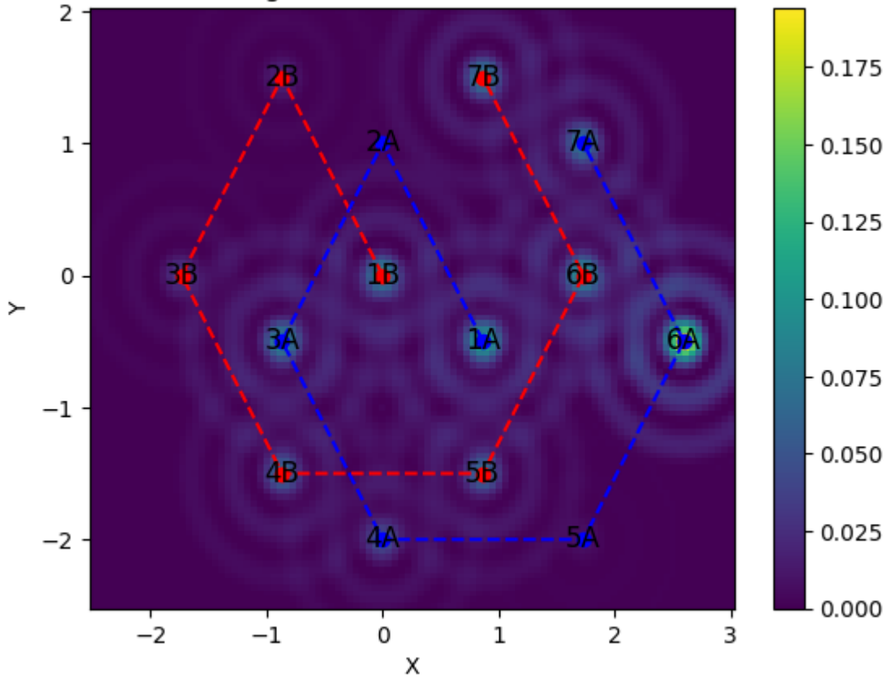

Hexagonal Lattice at time 4.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

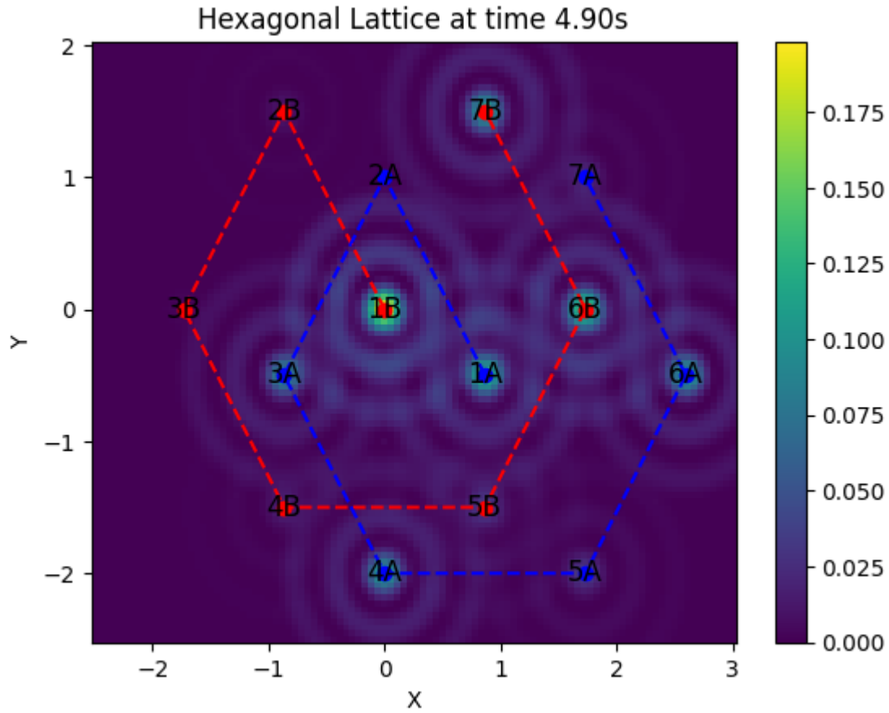

Hexagonal Lattice at time 4.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

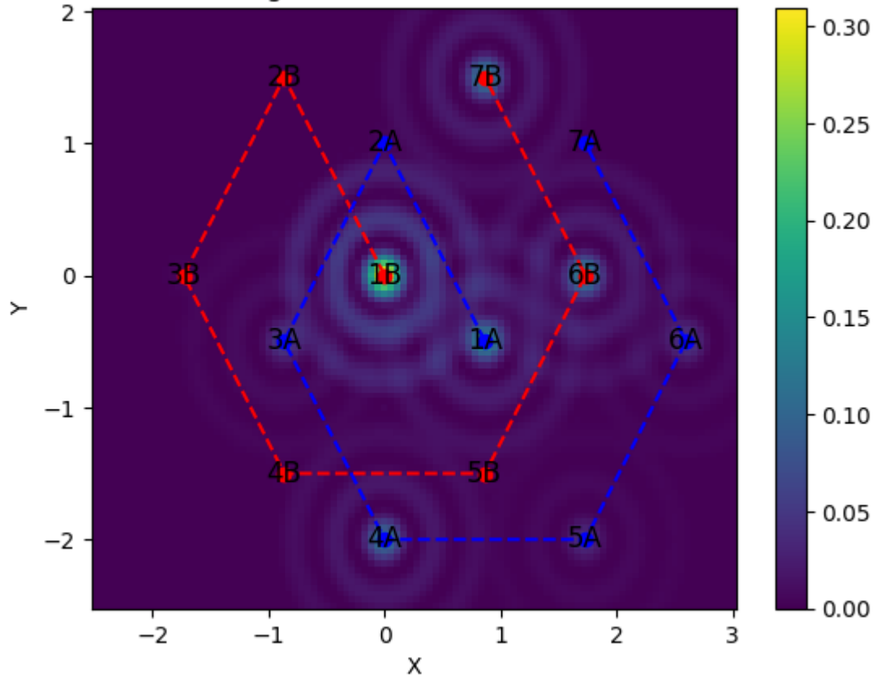

Hexagonal Lattice at time 4.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

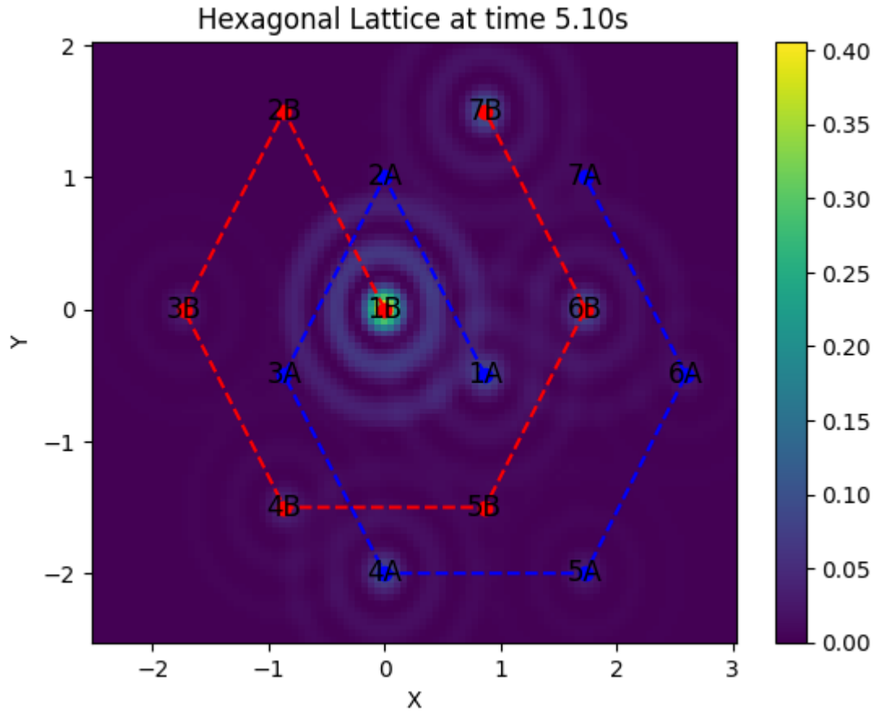

Hexagonal Lattice at time 4.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

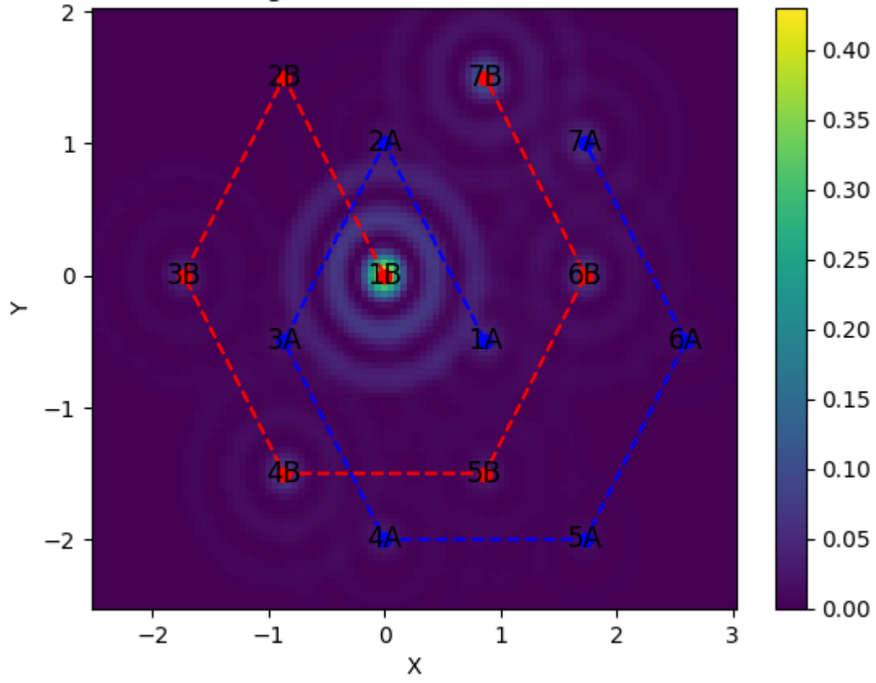

Hexagonal Lattice at time 4.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

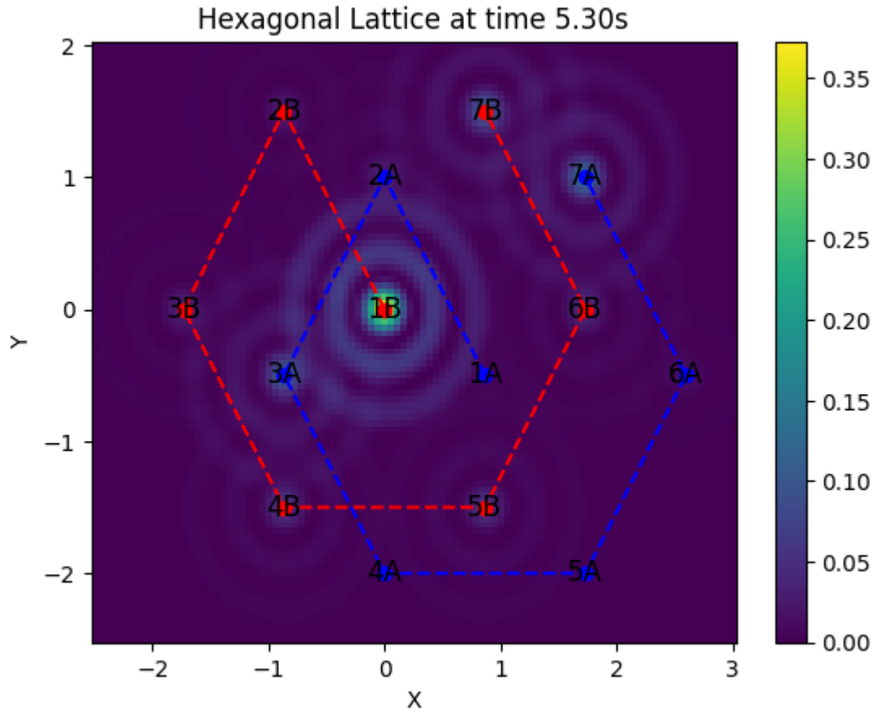

Hexagonal Lattice at time 4.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

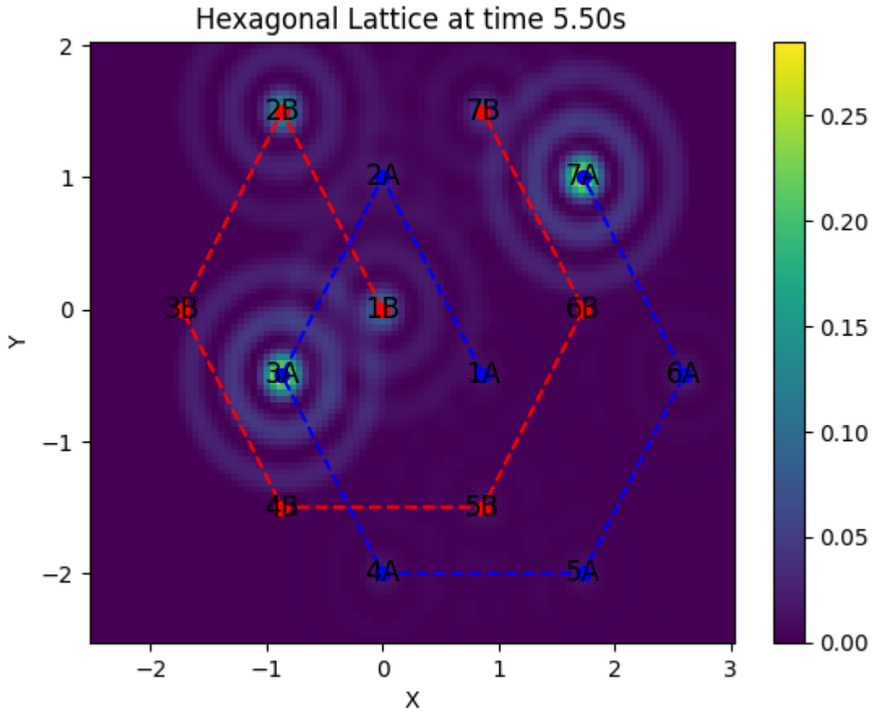

Hexagonal Lattice at time 4.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

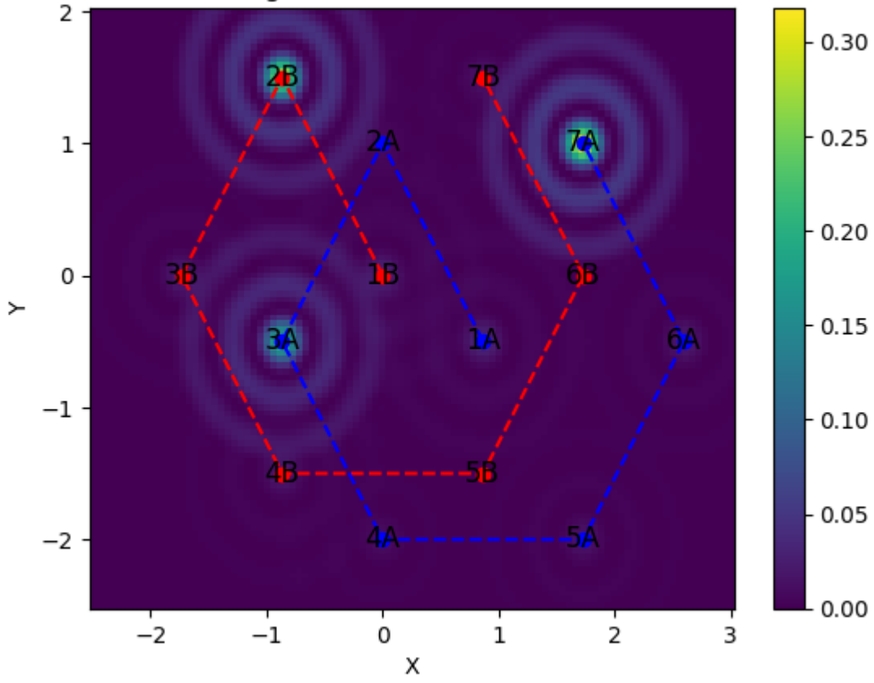

Hexagonal Lattice at time 5.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

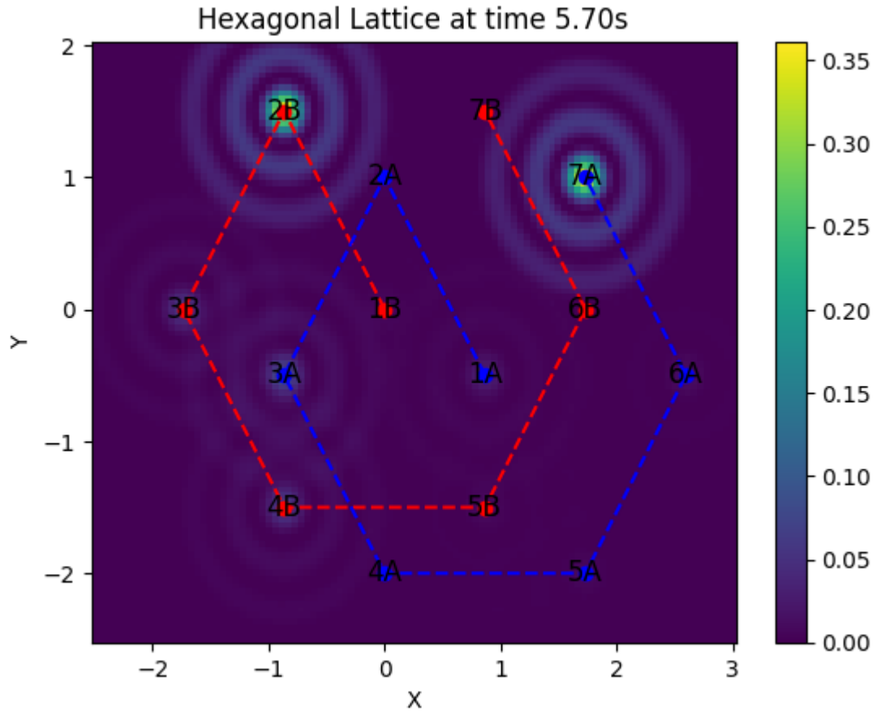

Hexagonal Lattice at time 5.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

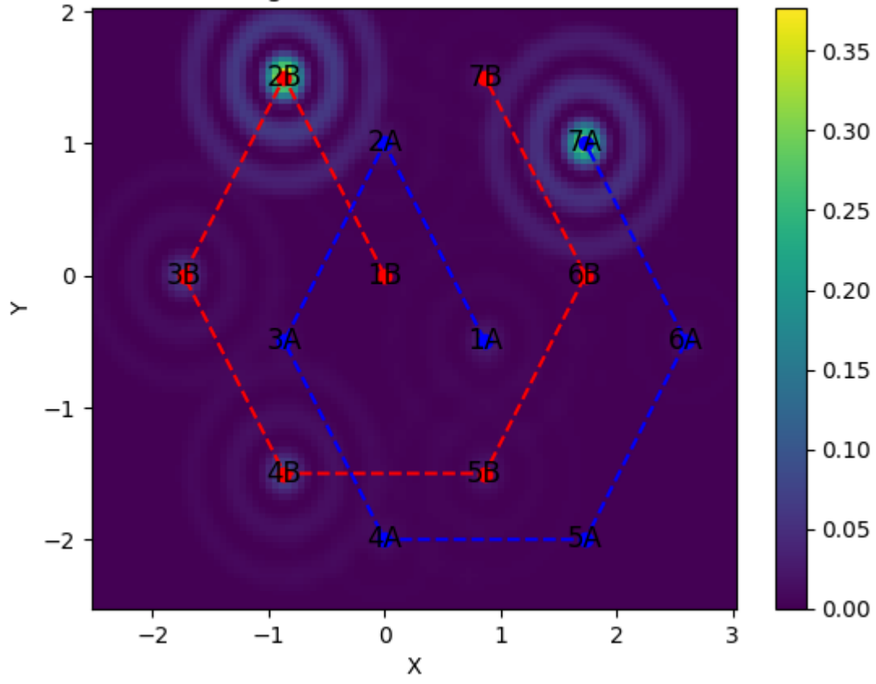

Hexagonal Lattice at time 5.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

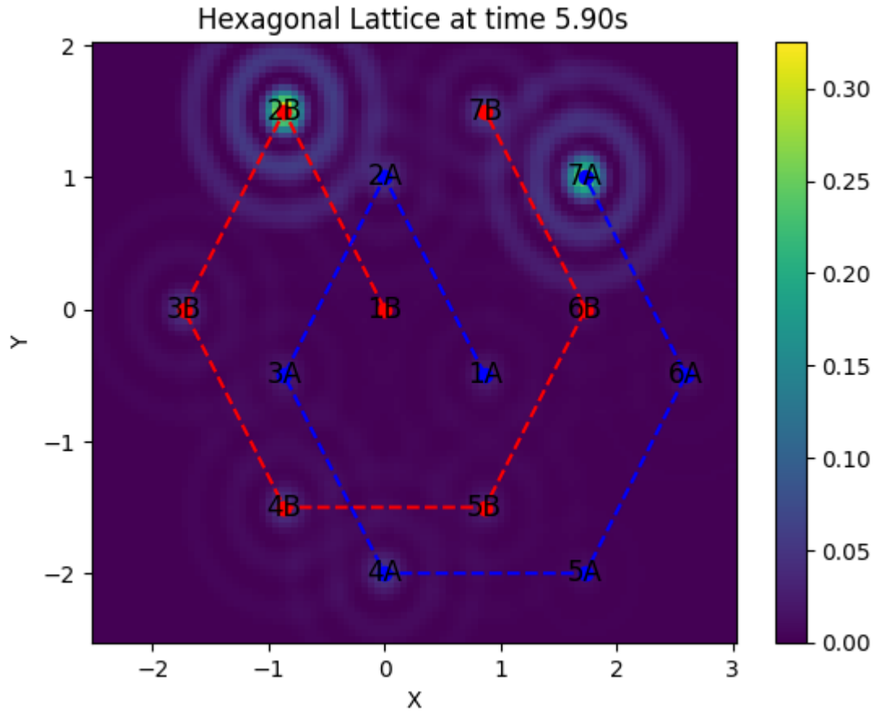

Hexagonal Lattice at time 5.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 5.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

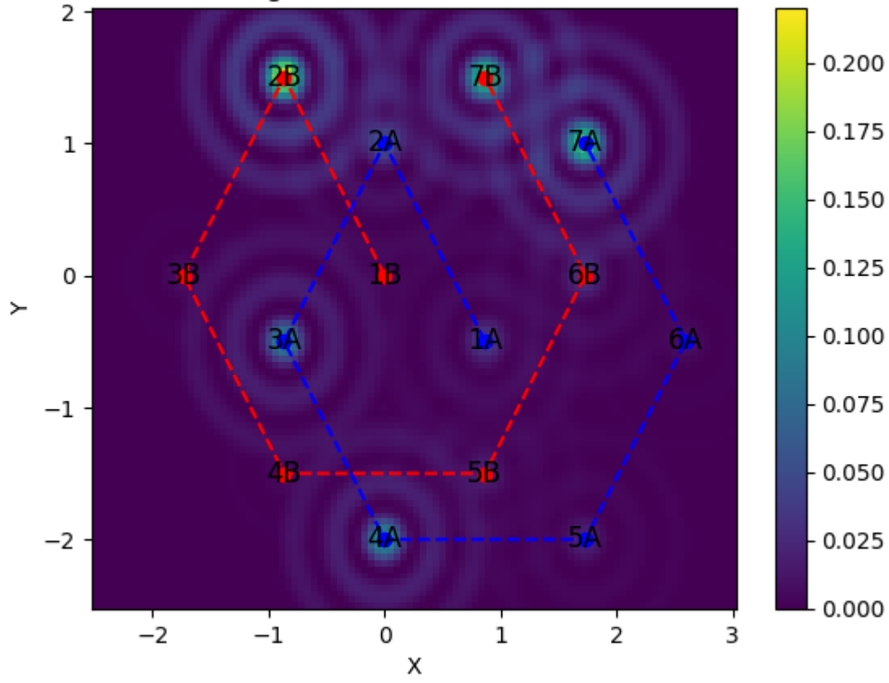

Hexagonal Lattice at time 5.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

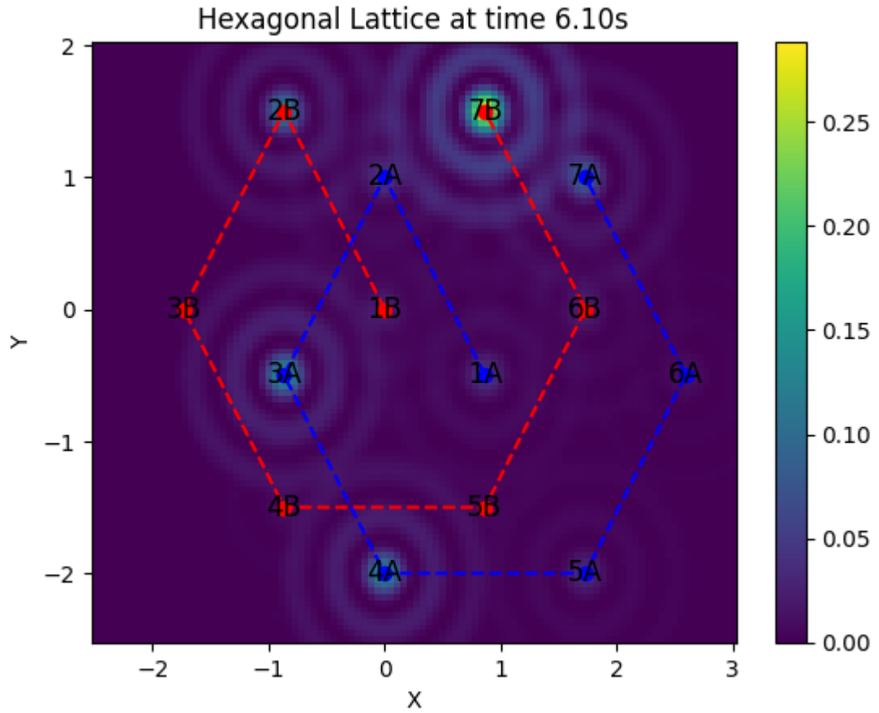

Hexagonal Lattice at time 5.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

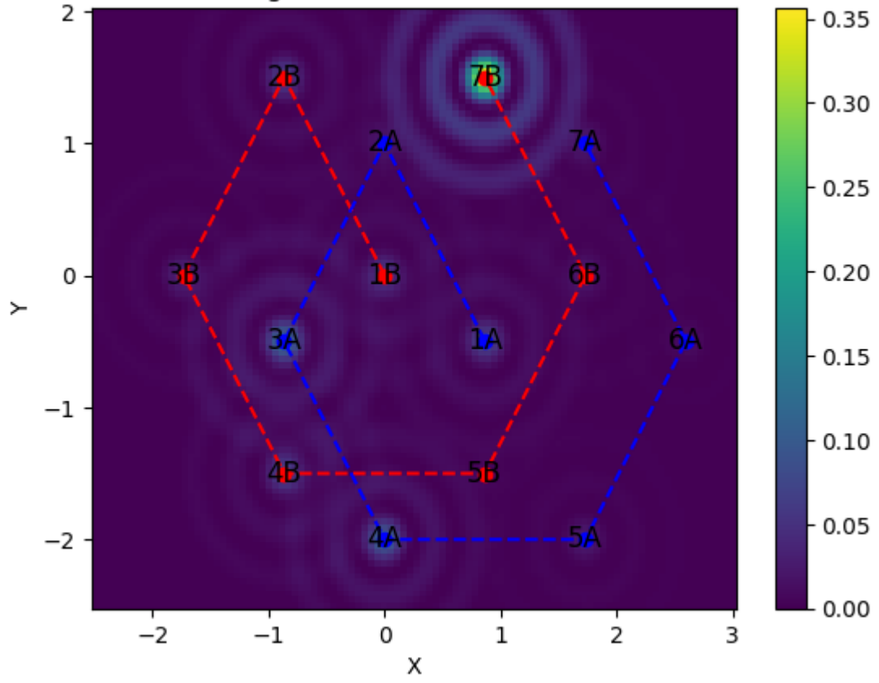

Hexagonal Lattice at time 5.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

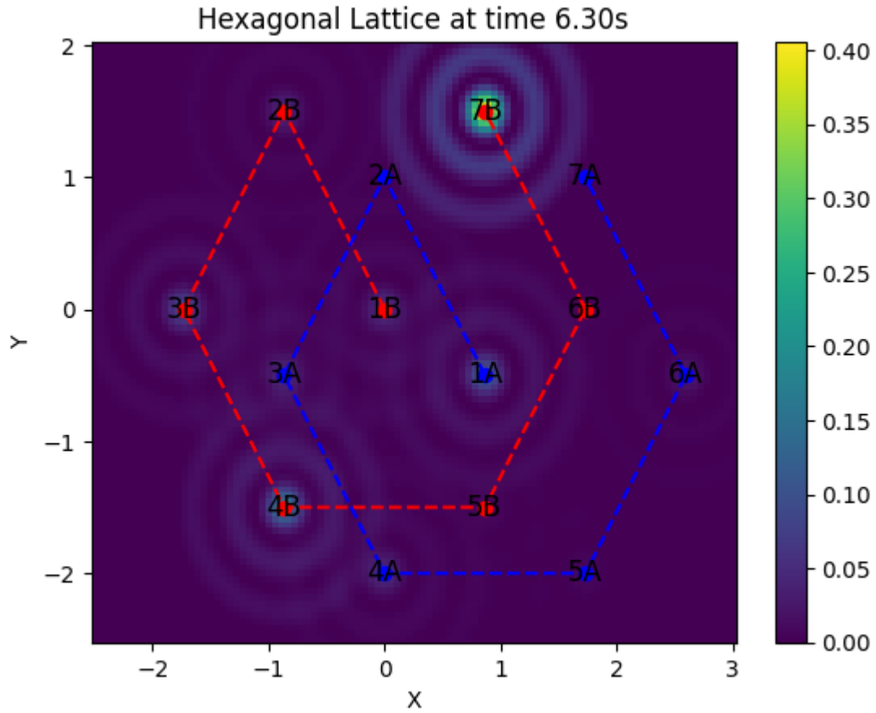

Hexagonal Lattice at time 5.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

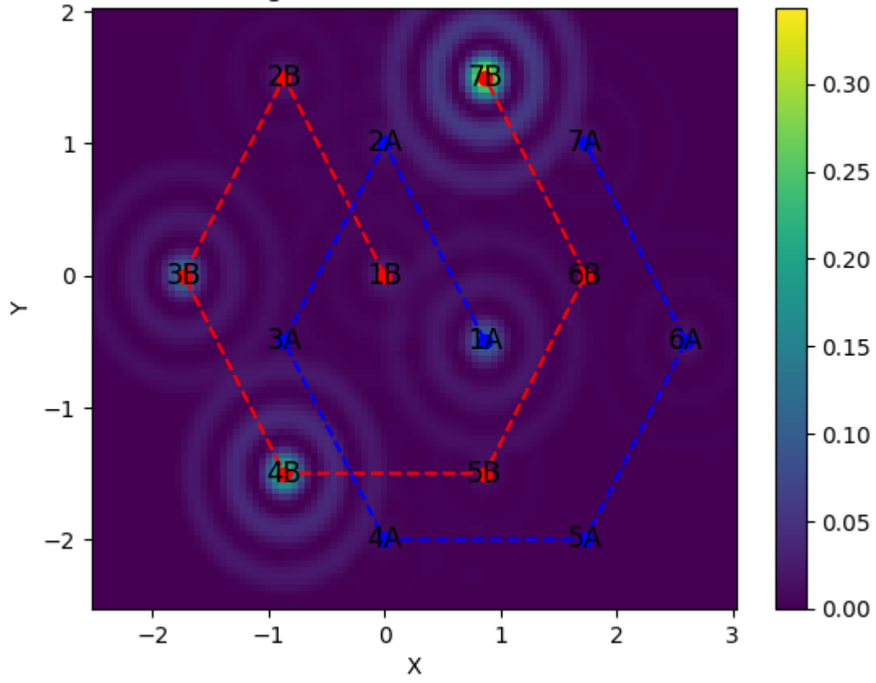

Hexagonal Lattice at time 5.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

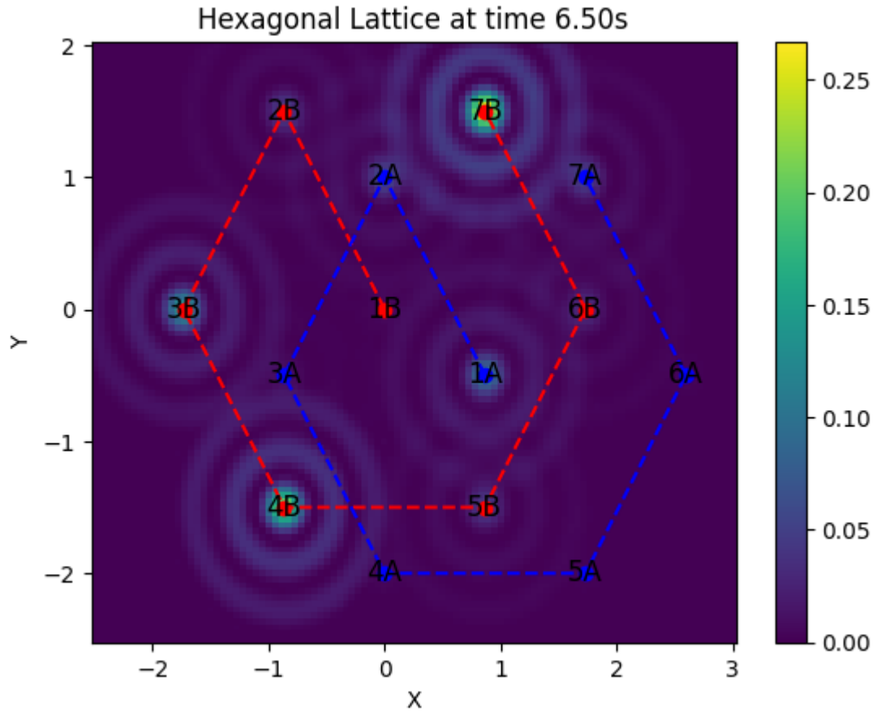

Hexagonal Lattice at time 6.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

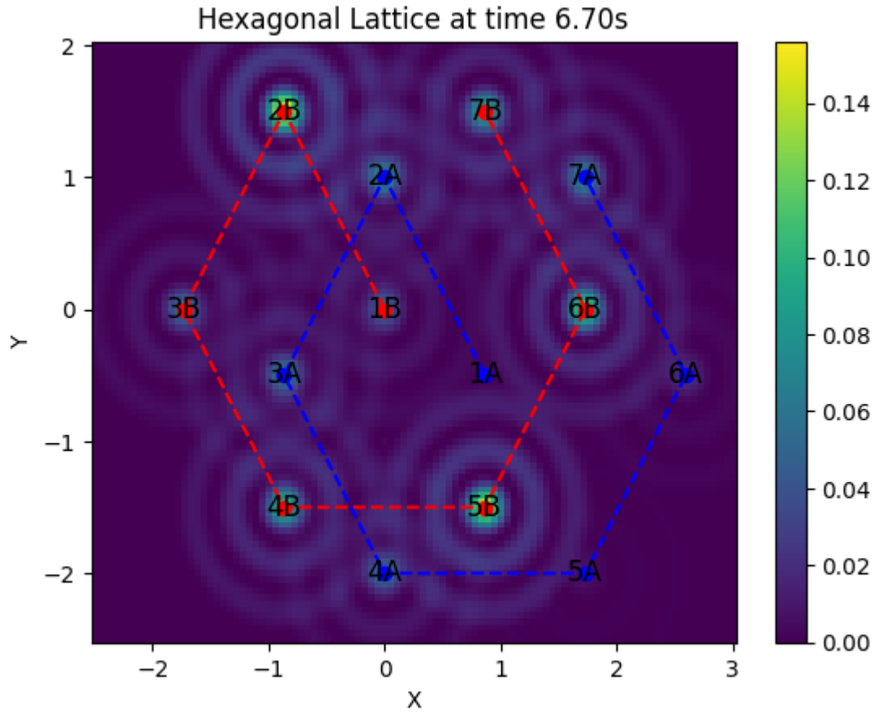

Hexagonal Lattice at time 6.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

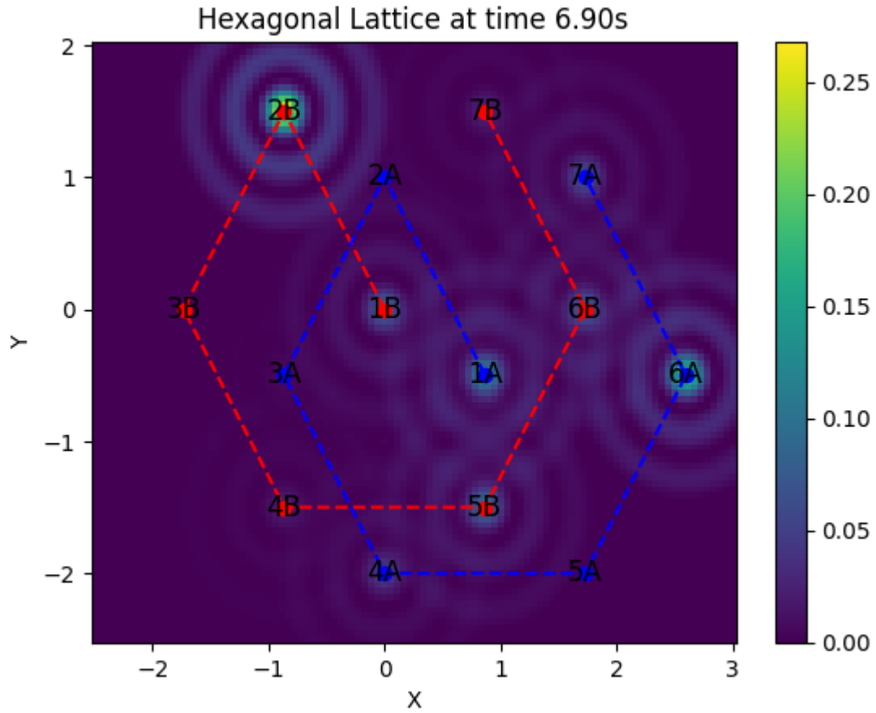

Hexagonal Lattice at time 6.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

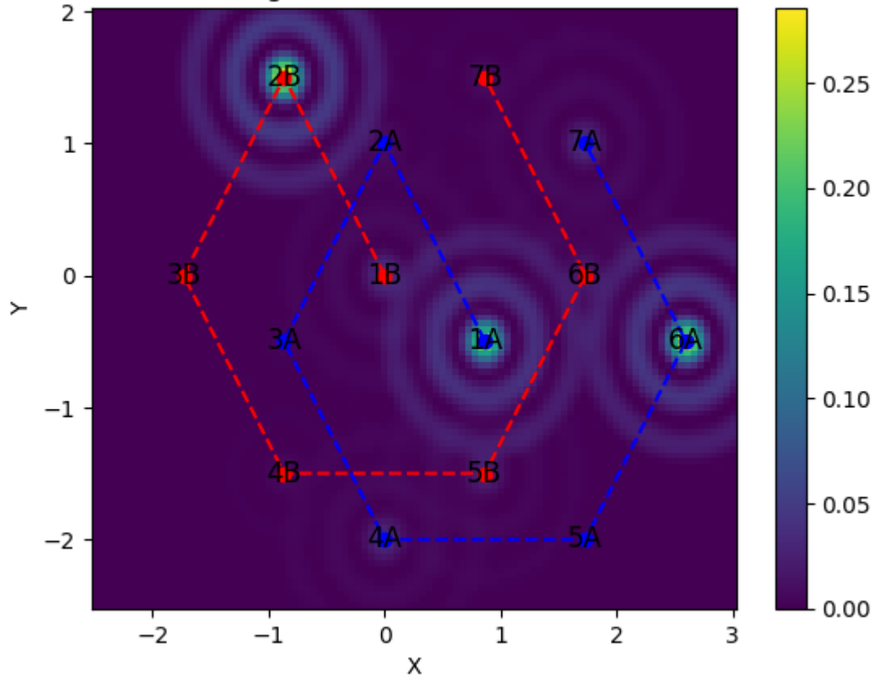

Hexagonal Lattice at time 6.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

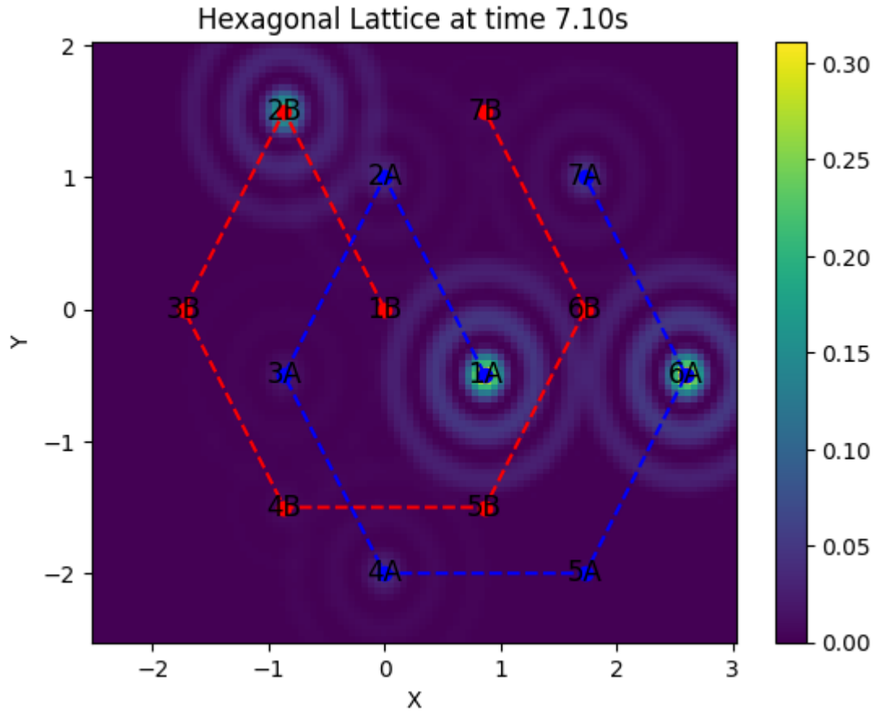

Hexagonal Lattice at time 6.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

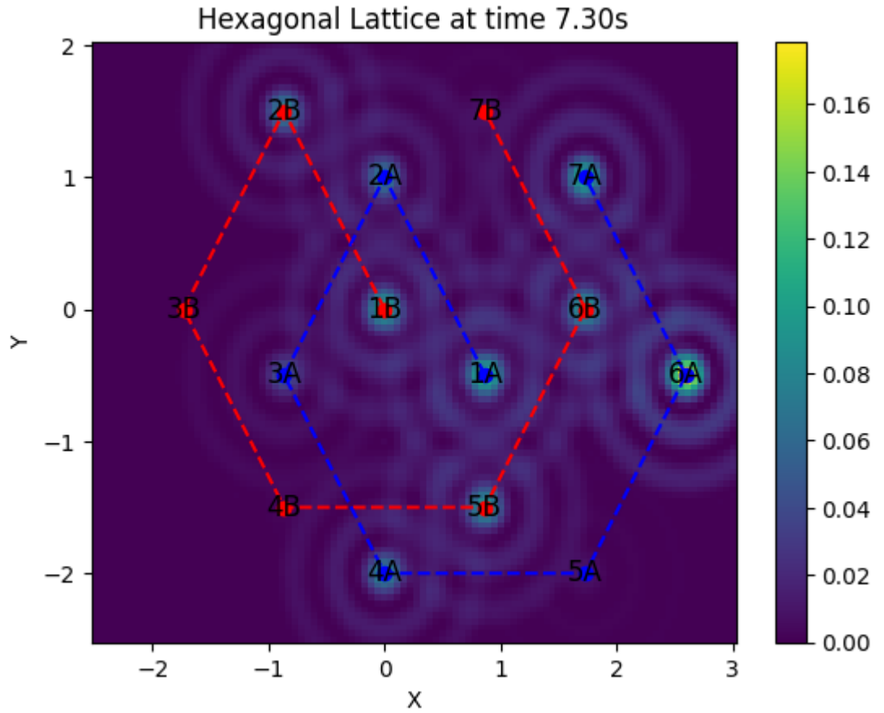

Hexagonal Lattice at time 6.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 6.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

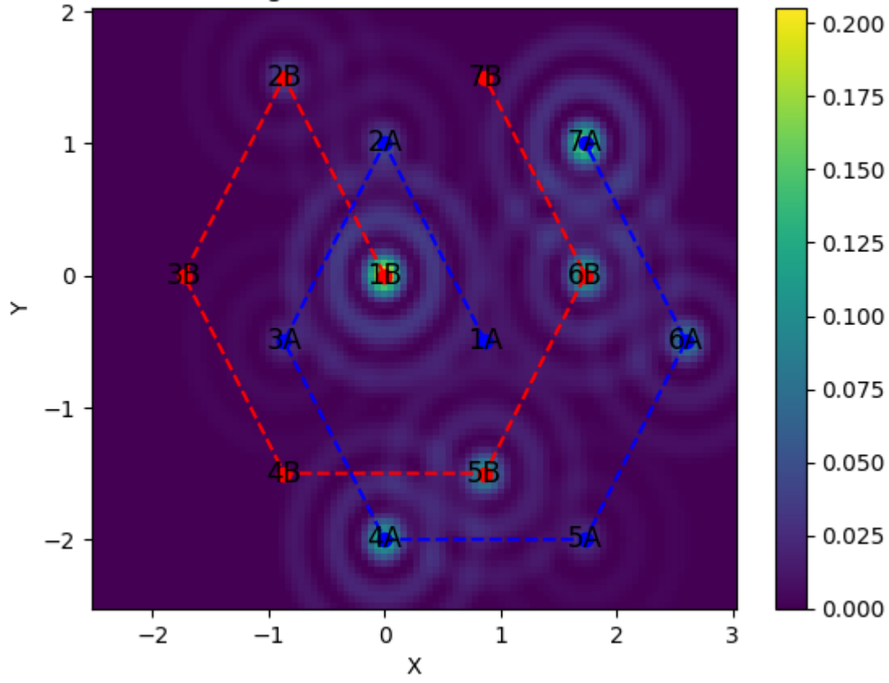

Hexagonal Lattice at time 6.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 6.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

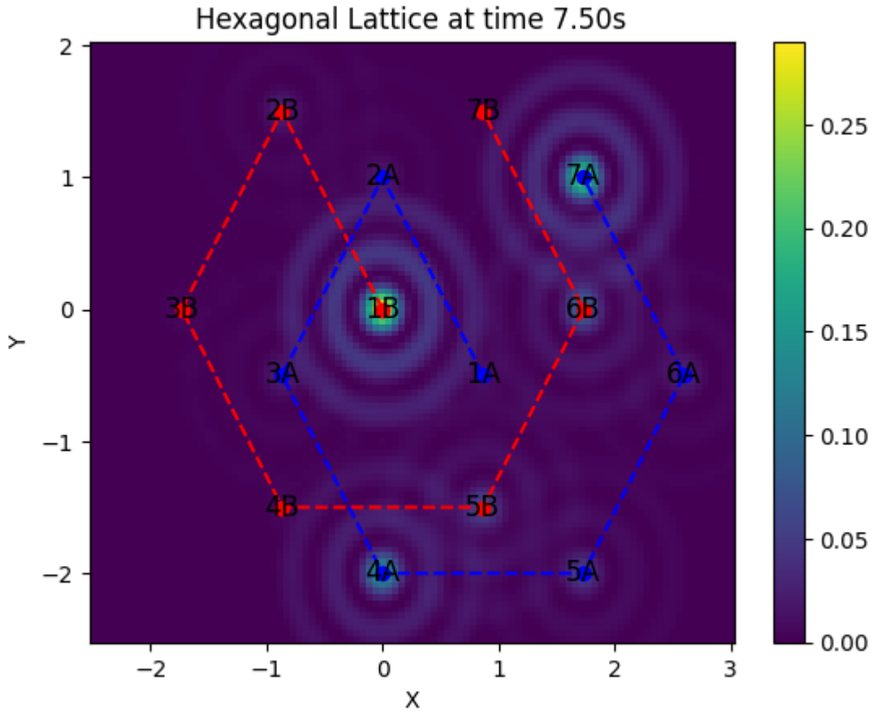

Hexagonal Lattice at time 6.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

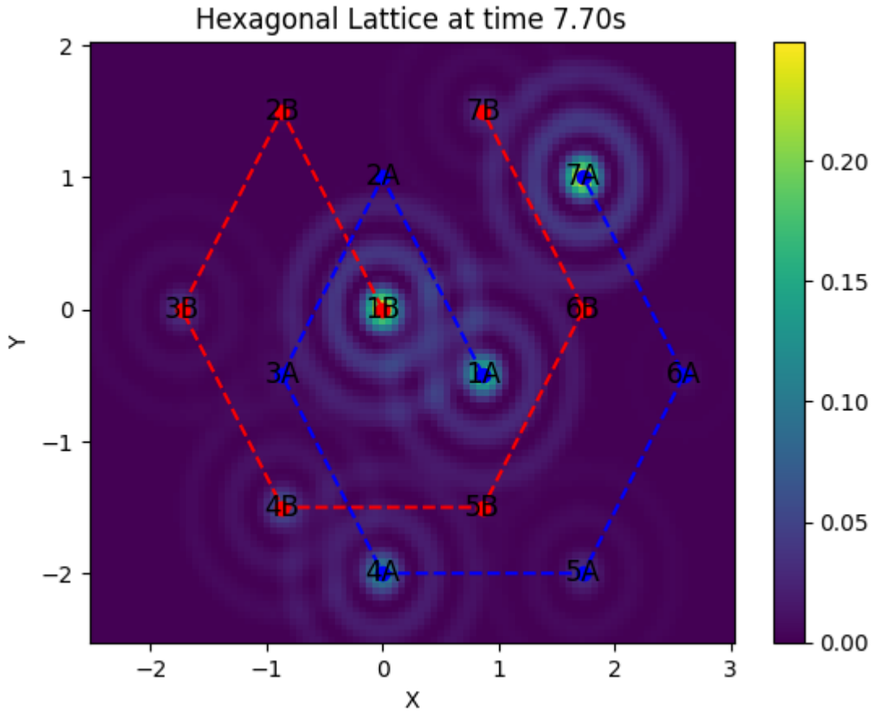

Hexagonal Lattice at time 7.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

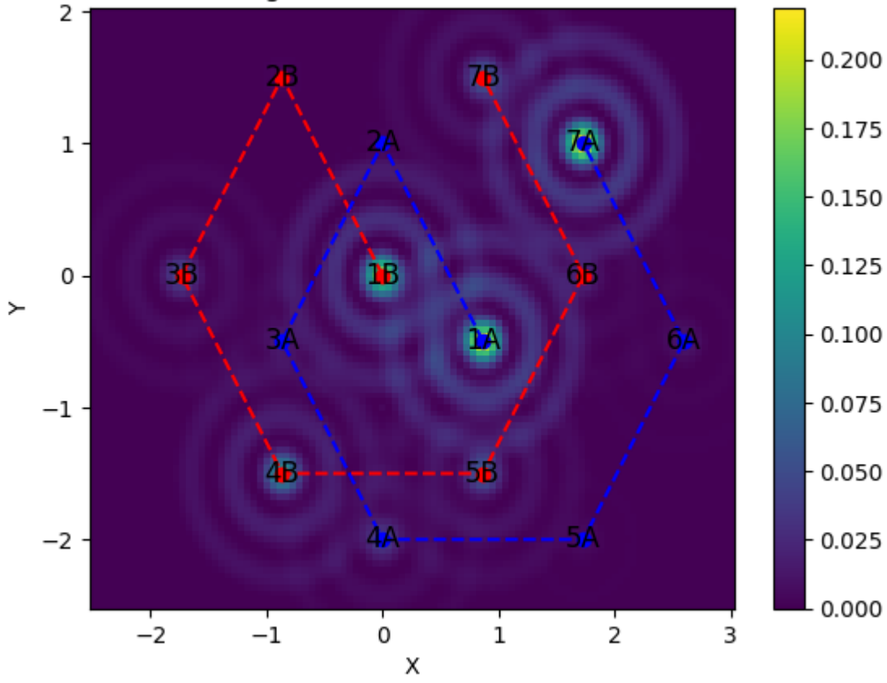

Hexagonal Lattice at time 7.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 7.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

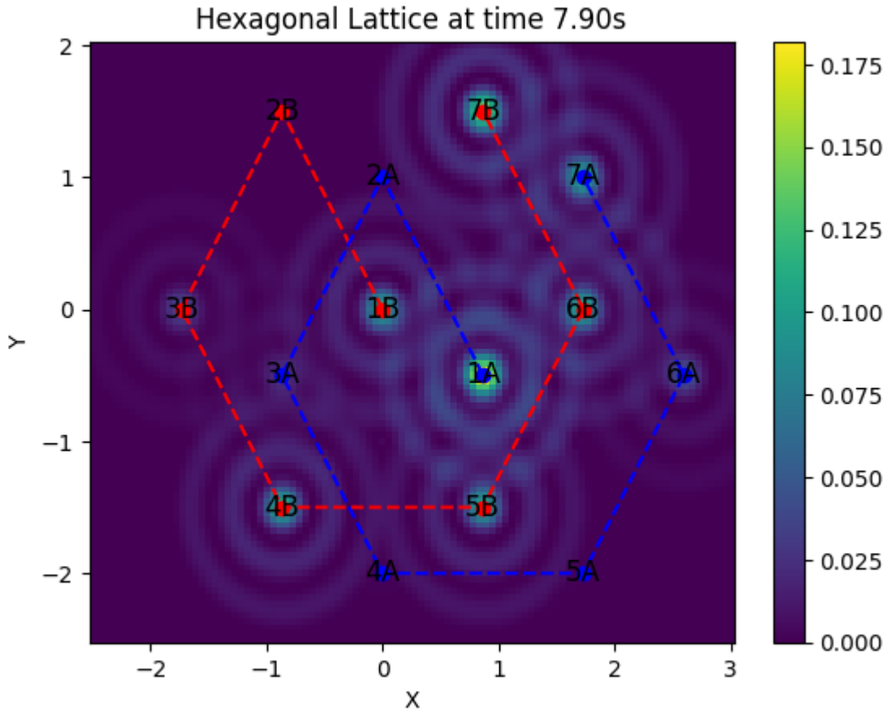

Hexagonal Lattice at time 7.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00

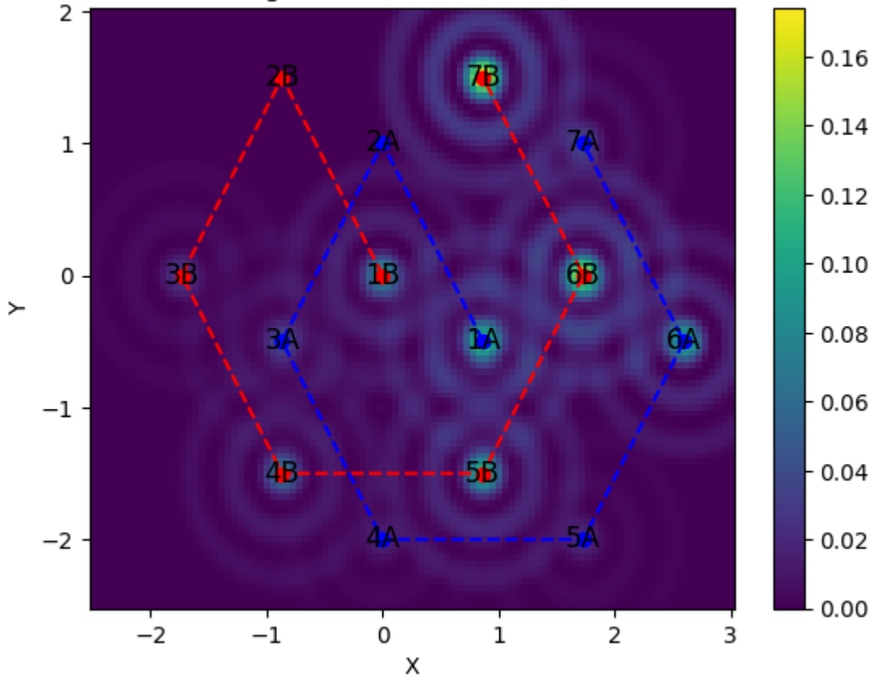

Hexagonal Lattice at time 7.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

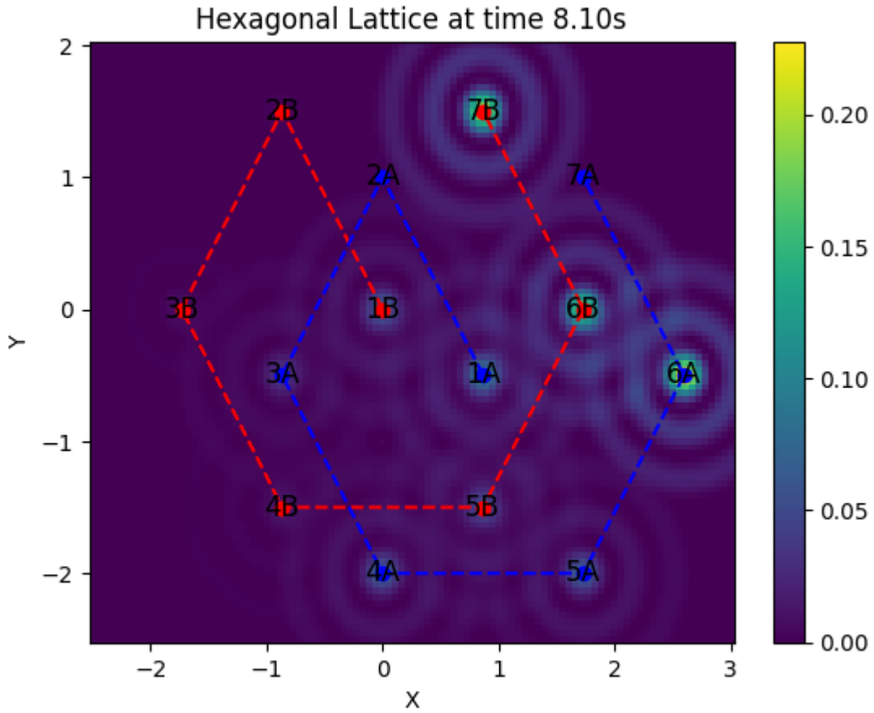

Hexagonal Lattice at time 7.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 7.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

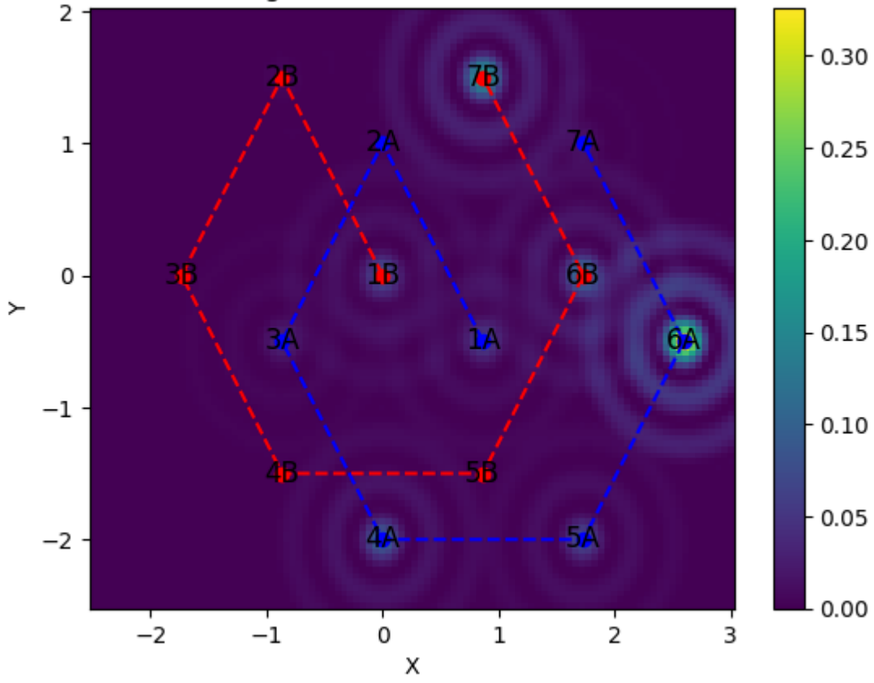

Hexagonal Lattice at time 7.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

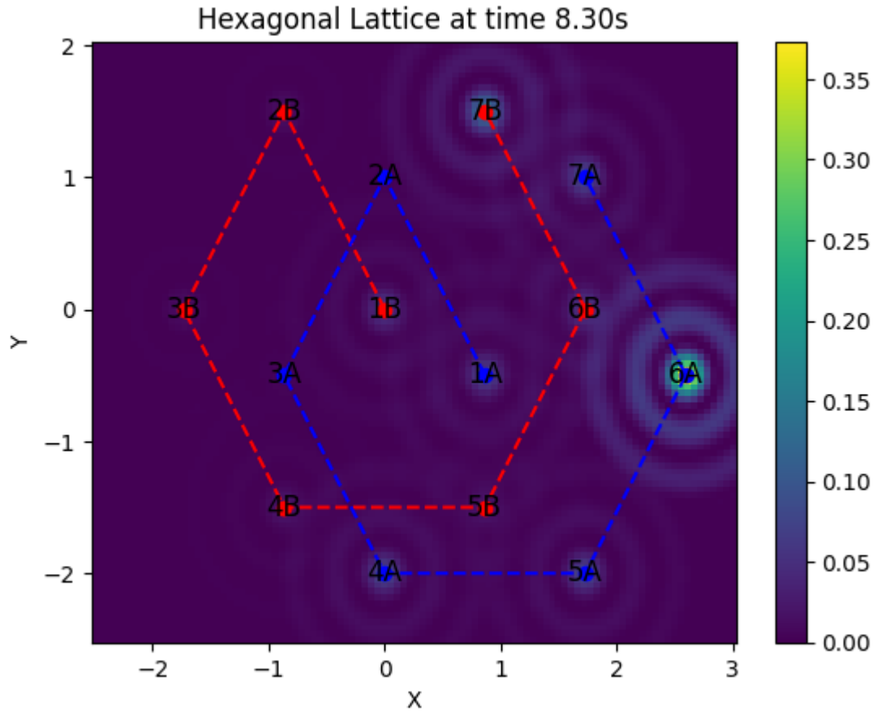

Hexagonal Lattice at time 7.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

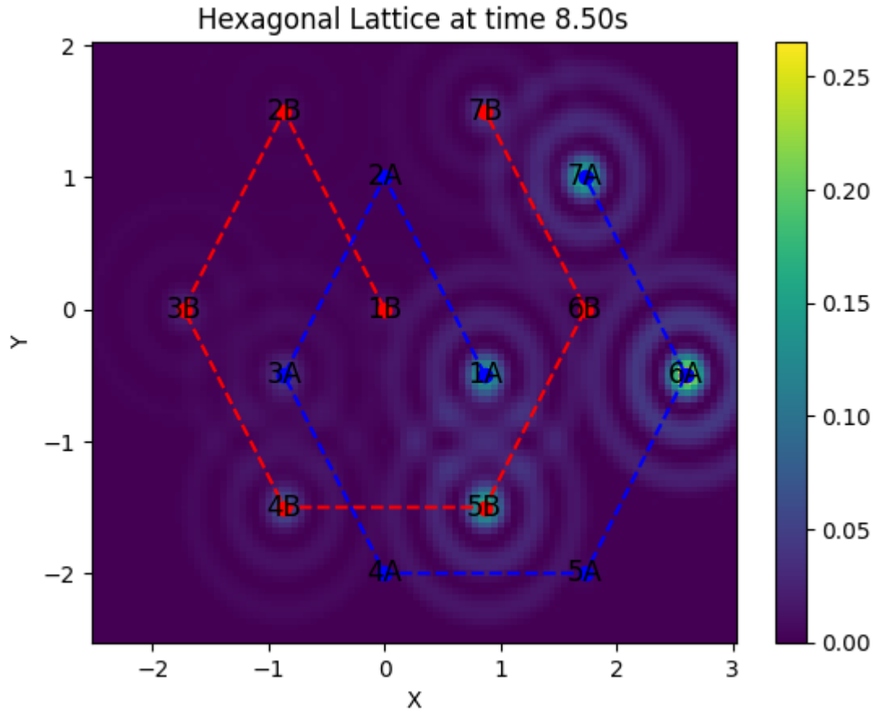

Hexagonal Lattice at time 7.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

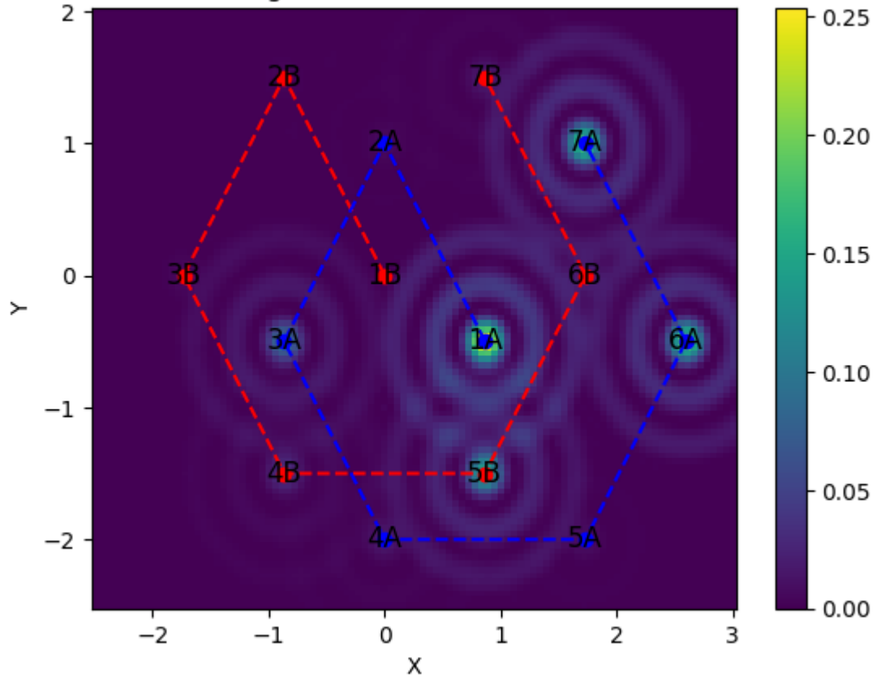

Hexagonal Lattice at time 8.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00

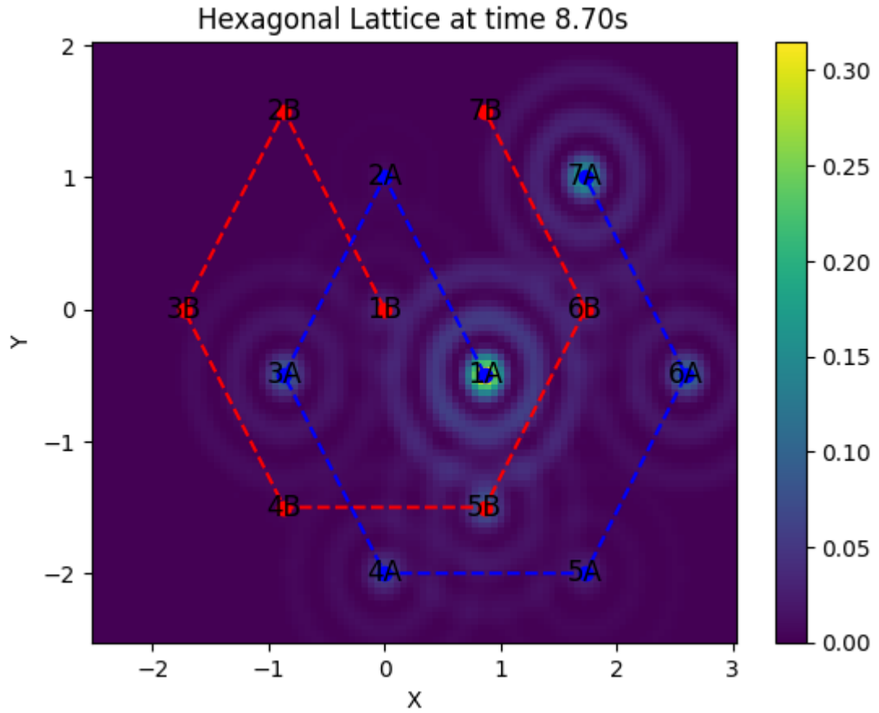

Hexagonal Lattice at time 8.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

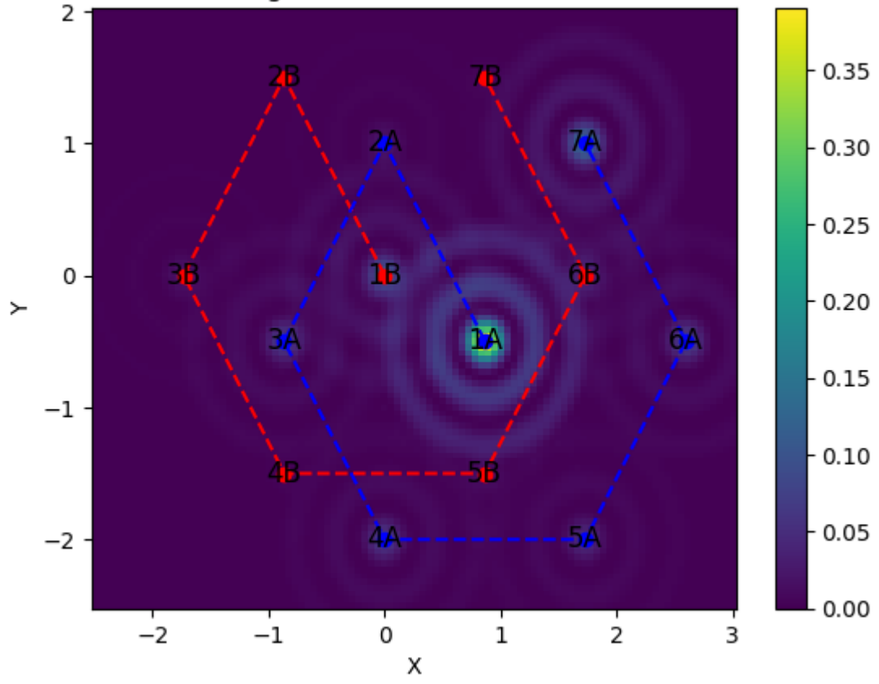

Hexagonal Lattice at time 8.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

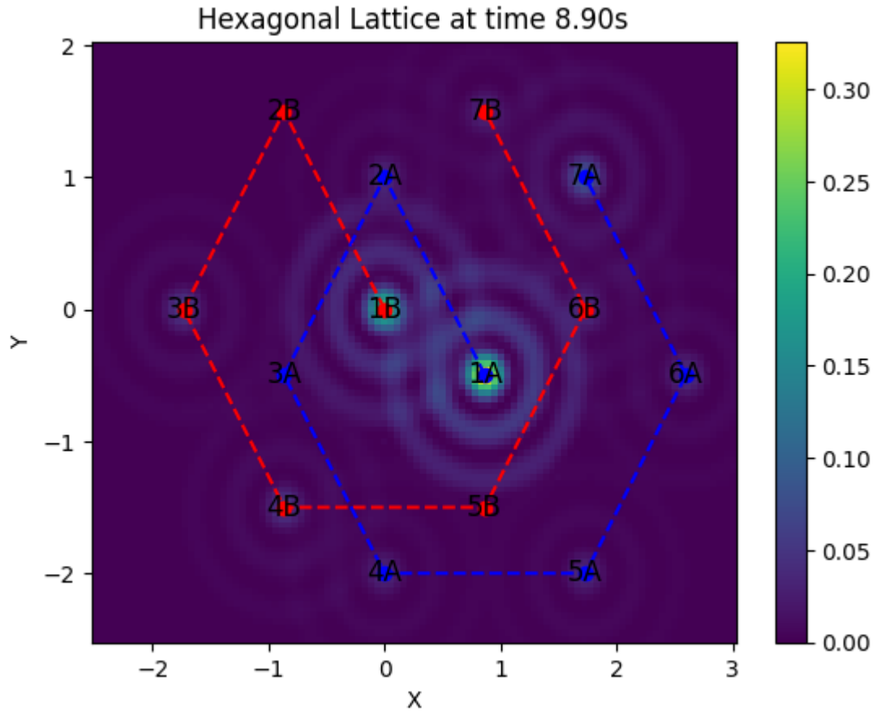

Hexagonal Lattice at time 8.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 8.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

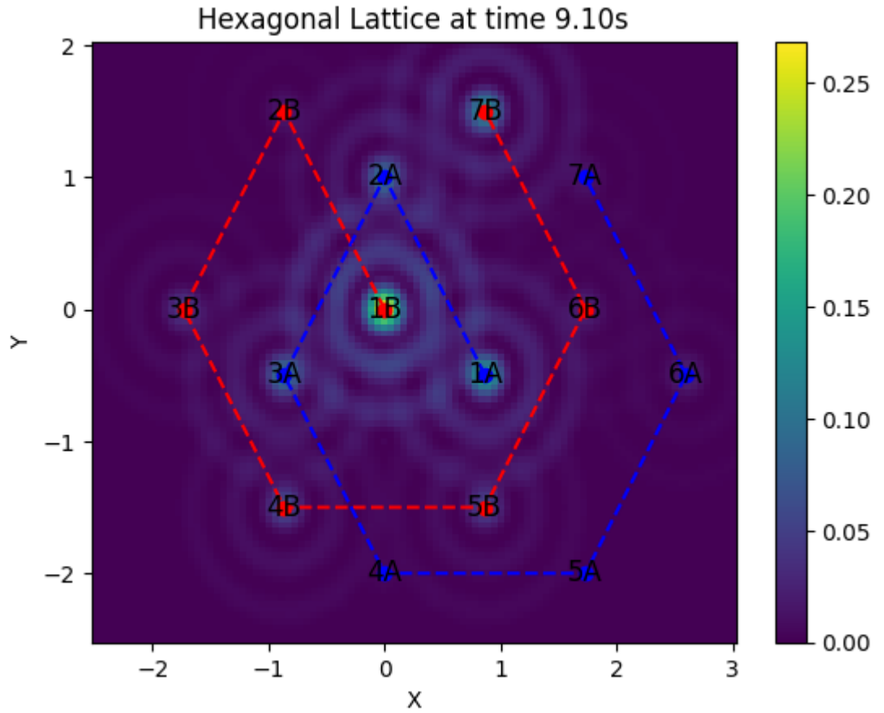

Hexagonal Lattice at time 8.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

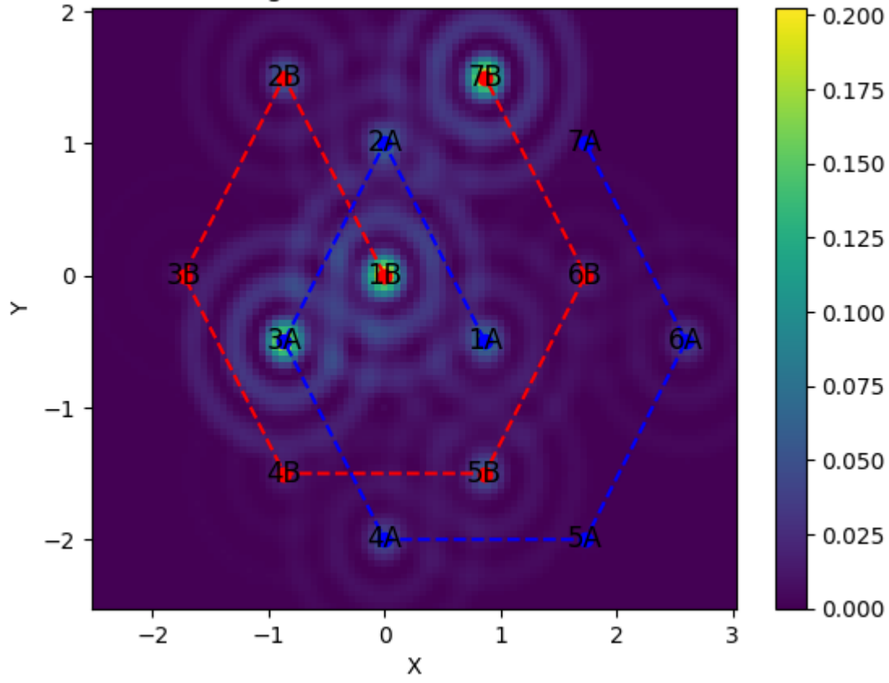

Hexagonal Lattice at time 8.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

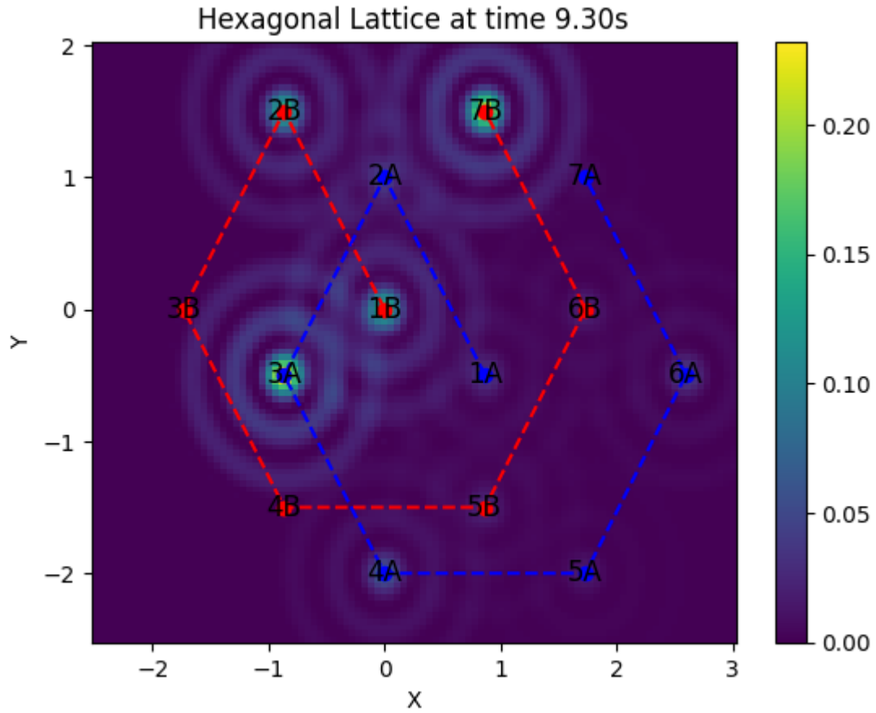

Hexagonal Lattice at time 8.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

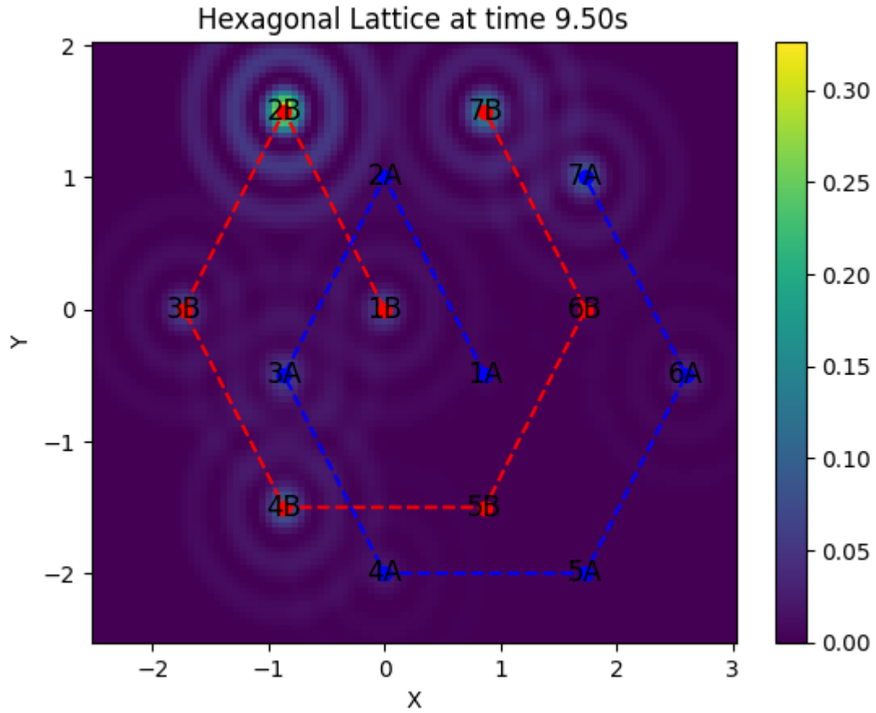

Hexagonal Lattice at time 8.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

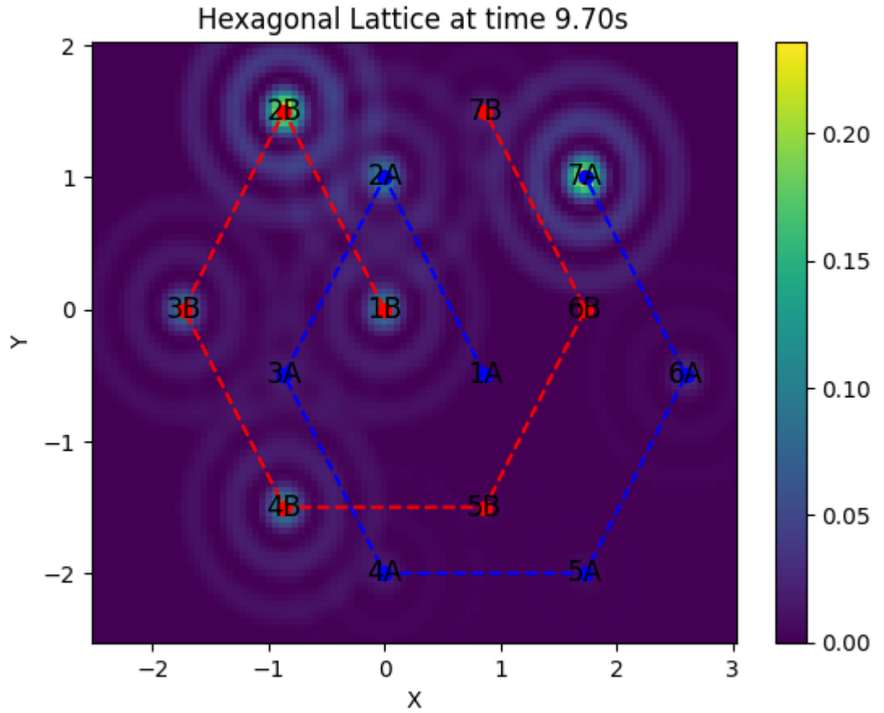

Hexagonal Lattice at time 8.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 9.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

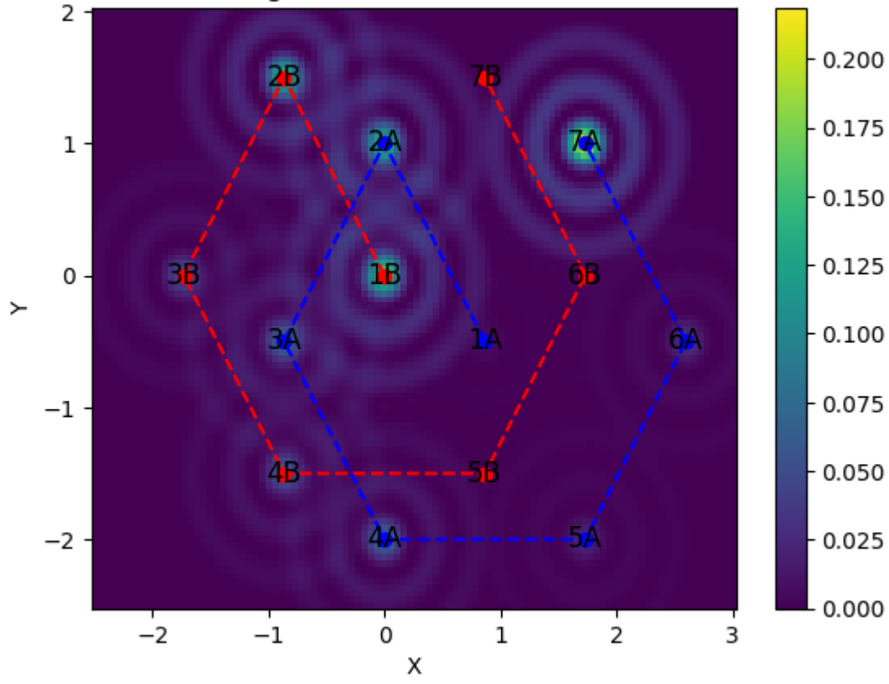

Hexagonal Lattice at time 9.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

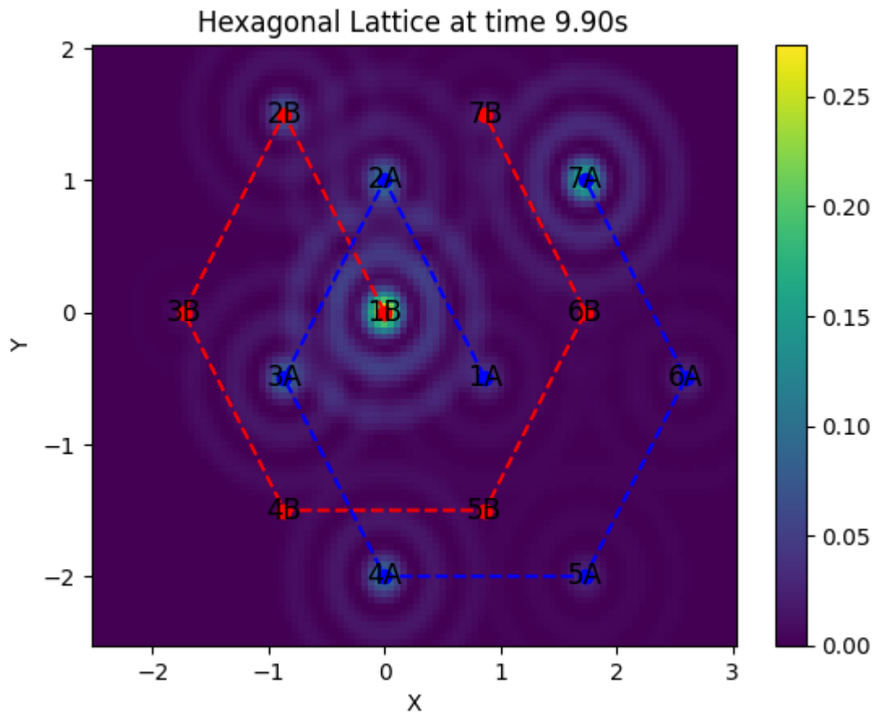

Hexagonal Lattice at time 9.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y


Hexagonal Lattice at time 9.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 9.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y


Hexagonal Lattice at time 9.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 9.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y


Hexagonal Lattice at time 9.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

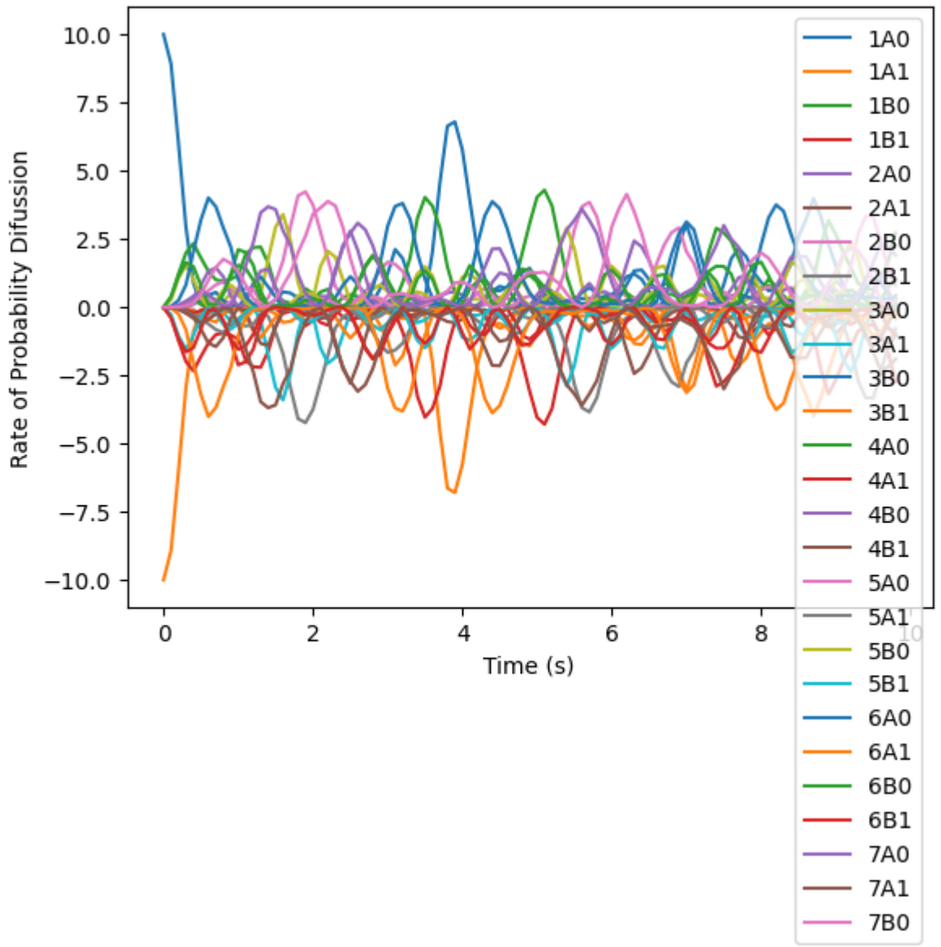

Hexagonal Lattice at time 9.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000



Mandatory task 9: From the data obtained in the quantum simulation, extract the rate of probability diffusion, i.e. how much probability flows between first neighbors.

Reply:

In [49]:
```
times,states,counts=getStruturedData_seq(data, n_qubits)

times=remove_111_seq(times)
states=remove_111_seq(states)
probs=remove_111_seq(counts/shots)

for i in range(len(basis)-1):
    index_old = np.where(states==i)
    index_new = np.where(states==i+1)
    plt.plot(times[index_old],(probs[index_new]-probs[index_old])/dt,label=basis[i])

plt.xlabel("Time (s)")
plt.ylabel("Rate of Probability Difussion")
plt.legend()
plt.show()
```



Mandatory task 10: Throughout this exercise, several approximations have been introduced, which may impose limitations on the validity of the results obtained from the quantum simulation. These approximations arise from both the description of the physical model and the conversion process into the model running on the quantum simulator. Please, list and discuss these approximations and their impact on the quality of the simulation results, as well as potential improvements.
Do not forget that, despited having used the numerical emulation provided by qiskit, you could in principle run it in a real quantum computer. As mentioned in during the classes, this introduce limitations to the quality of the results. List and discuss the limitations of running this quantum simulation in a real quantum computer, such as IBM's quantum computer, and their impact on the quality of the simulation results, as well as potential improvements.

Reply:

The approximations done were:

Modelling only 7 cells, because in doing so we are limiting the hopping possibilities of the outer cells;

Tight binding with first nearest neighboring atoms, here this is a good approximation, as the first nearest neighbors are going to have a far greater contributions than the second nearest neighbors;

Potential well, this is an approximation to simplify the actual interatomic potential of realistic wells (atoms);

Boundary conditions of an infinite potential well, we are following an approximation where we assume the boundary conditions of the finite potential well;

Task for extra points 3:
The purpose of this task is to assess your ability to interpret the outcomes of the simulated experiment and extract meaningful physical insights about the original physical system. There is no predefined answer for this task, as it encourages individual interpretation.

Considering this, what physical insights can you glean from the simulation results? What phenomena can you observe or interpret based on the obtained outcomes? Feel free to modify the simulation parameters to enrich your response.

Reply:

Task for extra points 4:
Create an animated GIF to illustrate the temporal evolution of the spatial distribution of particle localization based on the results obtained from your simulation.

Reply:

In [50]:
```
from IPython.display import HTML

# Path to the animated GIF file
gif_path = 'animation.gif'

# Embed the animated GIF in the notebook
html_code = f'<img src="{gif_path}">'
HTML(html_code)
```

Out[50]:

Exercise 2

In the previous model, we added a touch of spice, but now we're aiming to make the physical simulation truly hot! Get ready to introduce noise and bring out the Kraus operators!

Let's revisit the model of exercise 1, but this time, we'll assume that as the system evolves, there is a type of bit flip noise where the particle can randomly transition between the states $s = 0$ and $s = 1$ at each potential well. For simplicity, we will only consider transitions within the same well and exclude jumps between states in different wells.

Mandatory task 11: Develop the quantum circuit capable of simulate the original physical system initialized in the pure state used in exercise 1 but now with a noise model following the given guidelines.

Reply:

In [51]:
```
from qiskit.providers.aer.noise import NoiseModel
from qiskit.providers.aer.noise.errors import pauli_error

# Example error probabilities
p_gate1 = 0.03

# QuantumError objects
error_gate1 = pauli_error([('X',p_gate1), ('I', 1 - p_gate1)])
error_gate2 = error_gate1.tensor(error_gate1)

# Add errors to noise model
noise_bit_flip = NoiseModel()
noise_bit_flip.add_quantum_error(error_gate1,["x"],[4])
```

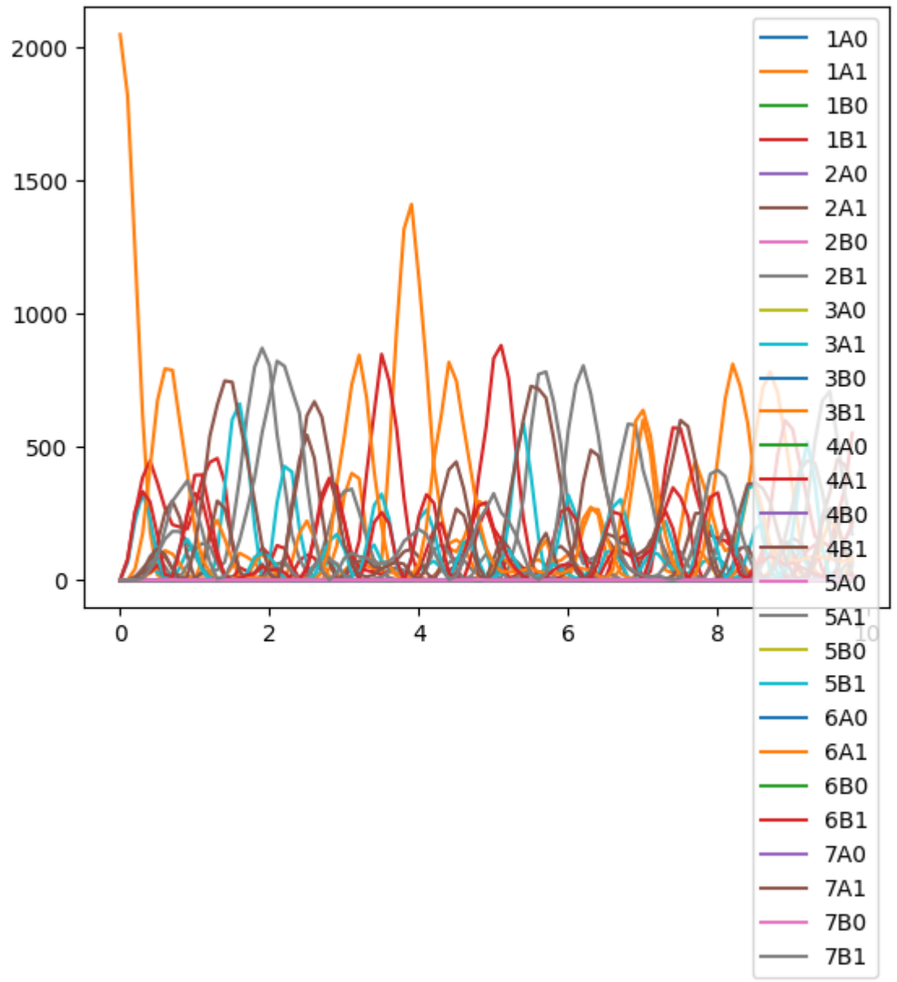
In [52]:
```python
n_qubits=5

initial_state = [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]


#Constructing the Hamiltonian


E1= -20
E0= -70

t0=-E0/10
t1=-E1/10
u=-abs(E1-E0)/100

H = np.array([
    [E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,0,0,t0,u,0,0,0,0,0,0,0,0],
    [0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,0,u,t1,0,0,0,0,0,0,0,0],
    [t0,u,E0,0,t0,u,0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [u,t1,0,E1,u,t1,0,0,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,0,0,0,0],
    [0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,0,0,0],
    [0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,t0,u,0,0,t0,u,E0,0,t0,u,0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,u,t1,0,0,u,t1,0,E1,u,t1,0,0,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,u,t1,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,t0,u,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,u,t1,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0,0,0],
    [t0,u,0,0,0,0,0,0,0,0,0,0,t0,u,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0,0,0],
    [u,t1,0,0,0,0,0,0,0,0,0,0,u,t1,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0,0,0],
    [t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0,0,0],
    [u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,t0,u,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,u,t1,0,0,0,0],
    [0,0,0,0,t0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,t0,u,E0,0,0,0,0,0],
    [0,0,0,0,u,t1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,t1,0,E1,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]])


# Time
t_max = 10
dt = 0.1

time = np.arange(dt,t_max,dt)

shots =2048
# Create a Quantum Circuit acting on a quantum register
data_noise = [[0,{"00001":shots}]]
for t in time:
```

```
#print(t)
q = QuantumRegister(n_qubits, name = 'q')
c = ClassicalRegister(n_qubits, name = 'c')
circuit1 = QuantumCircuit(q,c,name='qc')
# Create the superposition state to show that things are working
circuit1.initialize(initial_state, [q[0],q[1],q[2],q[3],q[4]])

circuit1.hamiltonian(H,t,[q[0],q[1],q[2],q[3],q[4]])

circuit1.measure(0,0)
circuit1.measure(1,1)
circuit1.measure(2,2)
circuit1.measure(3,3)
circuit1.measure(4,4)
simulator = Aer.get_backend('qasm_simulator')
job = execute(circuit1, simulator,
          basis_gates=noise_bit_flip.basis_gates,
          noise_model=noise_bit_flip,shots=shots)
results = job.result()
counts = results.get_counts()

data_noise.append([t,counts])
```

Mandatory task 12: Run the quantum circuit and obtain the evolution of statistics of the measurements over time.

Reply:

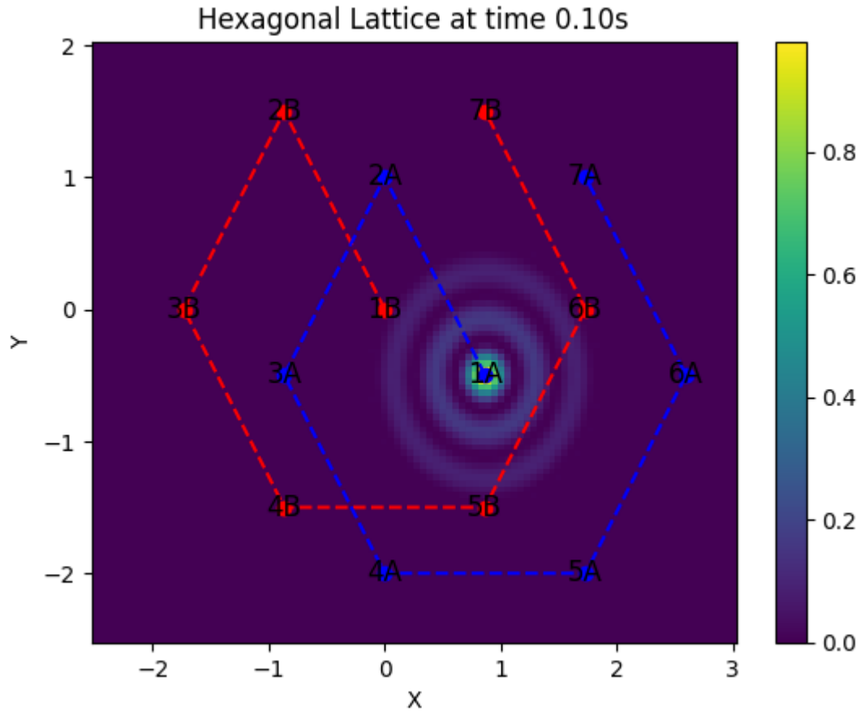
In [53]:
```
times_noise,states_noise,counts_noise=getStruturedData_seq(data_noise, n_qubits)


times_noise=remove_111_seq(times_noise)
states_noise=remove_111_seq(states_noise)
counts_noise=remove_111_seq(counts_noise)

basis=["1A0","1A1","1B0","1B1","2A0","2A1","2B0","2B1","3A0","3A1","3B0","3B1","4A0","4A1","4B0","4B1","5A0","5A1","5B0","5B1","6A0","6A1","6B0","6B1","7A0","7A1","7B0

for i in range(len(basis)):
    index = np.where(states_noise==i)
    plt.plot(times_noise[index],counts_noise[index],label=basis[i])

plt.legend()
plt.show()
```

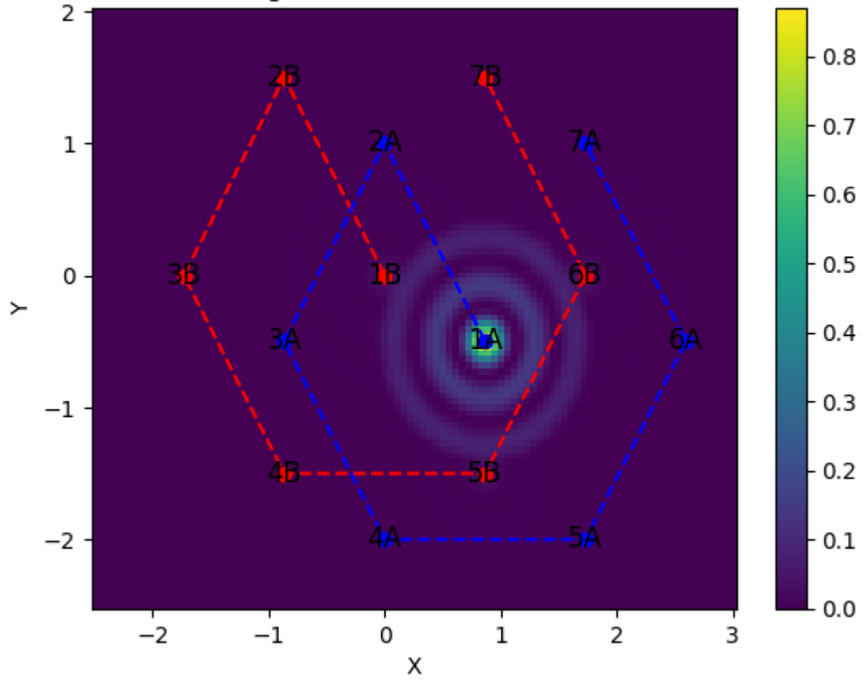


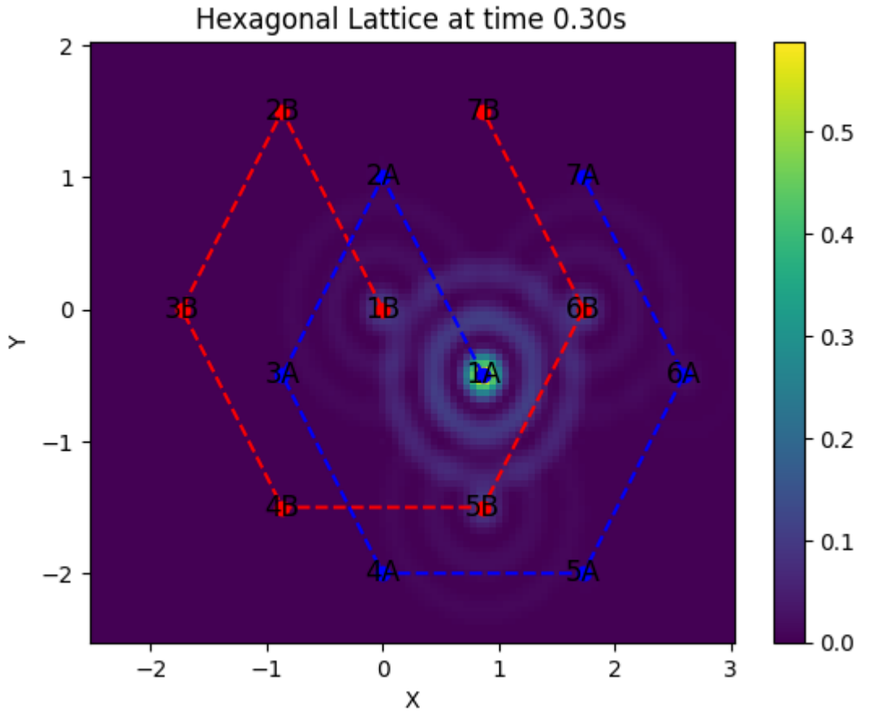
Mandatory task 13: Provide images the illustrate the evolution of the probability of finding the particle in each lattice well or on each point of space (either one will do). The goal of this image is to identify differences relative to the results of exercise 1.

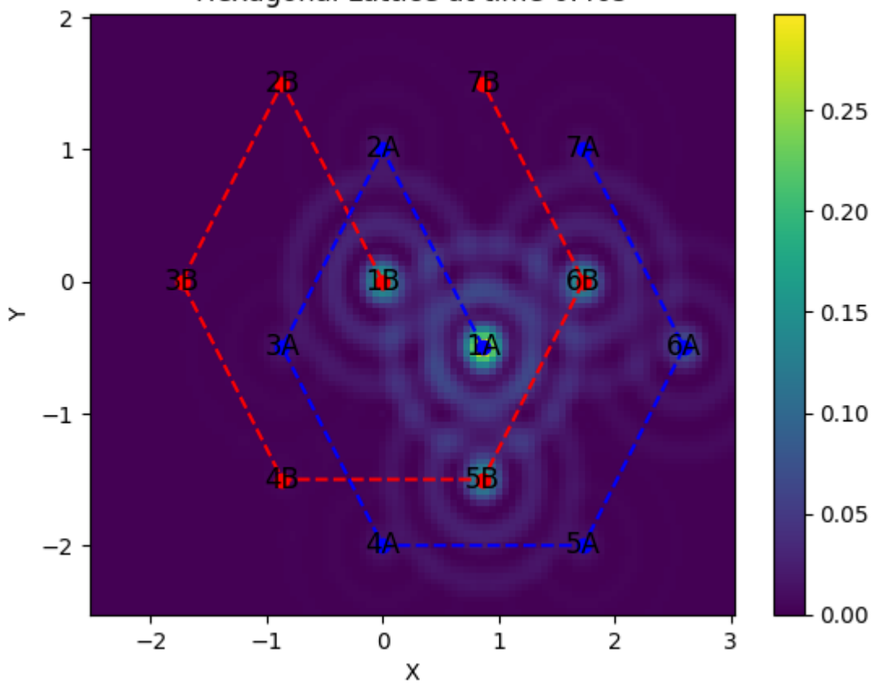
Reply:

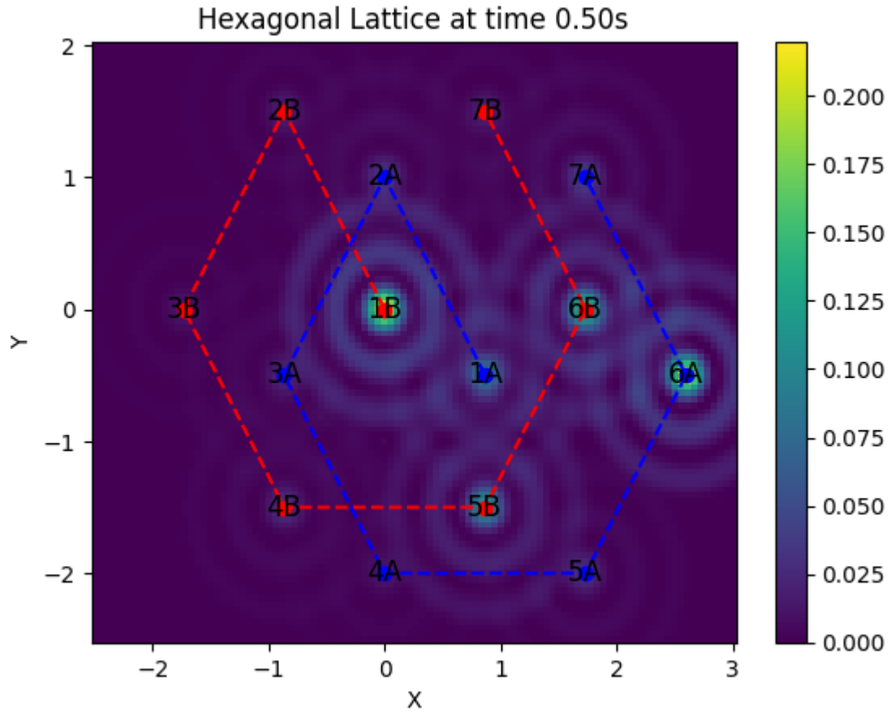
In [54]:
```
times_noise,states_noise,counts_noise=getStruturedData_grid(data_noise, n_qubits)

times_noise=remove_111_grid(times_noise)
states_noise=remove_111_grid(states_noise)
counts_noise=remove_111_grid(counts_noise)

x=np.linspace(-2.5,3,100)
y=np.linspace(-2.5,2,100)

X, Y = np.meshgrid(x,y)
i=0
for count in counts_noise:
    plot_time(count,X,Y,time[i])
    i+=1
```

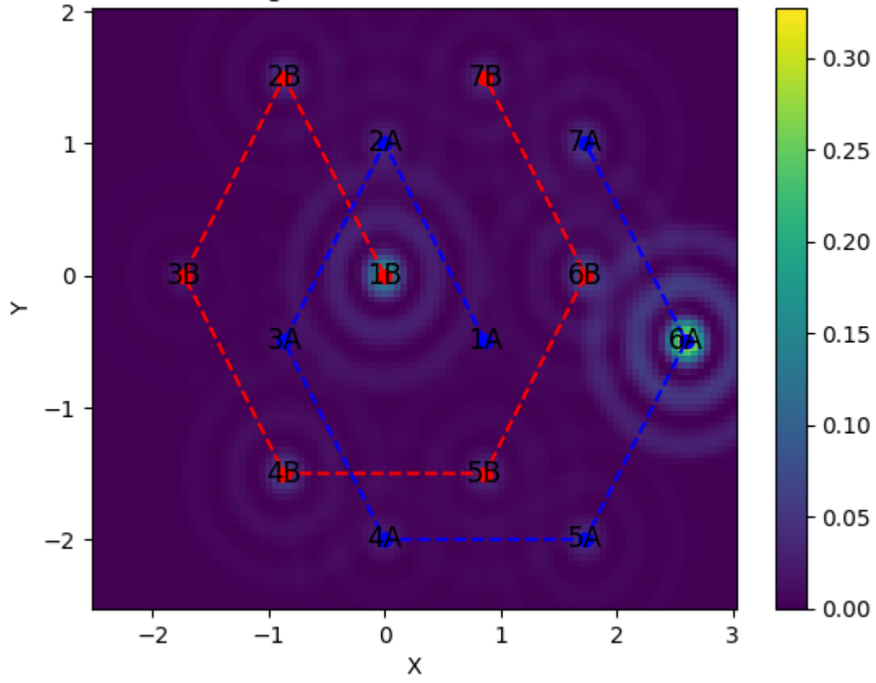

Hexagonal Lattice at time 0.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

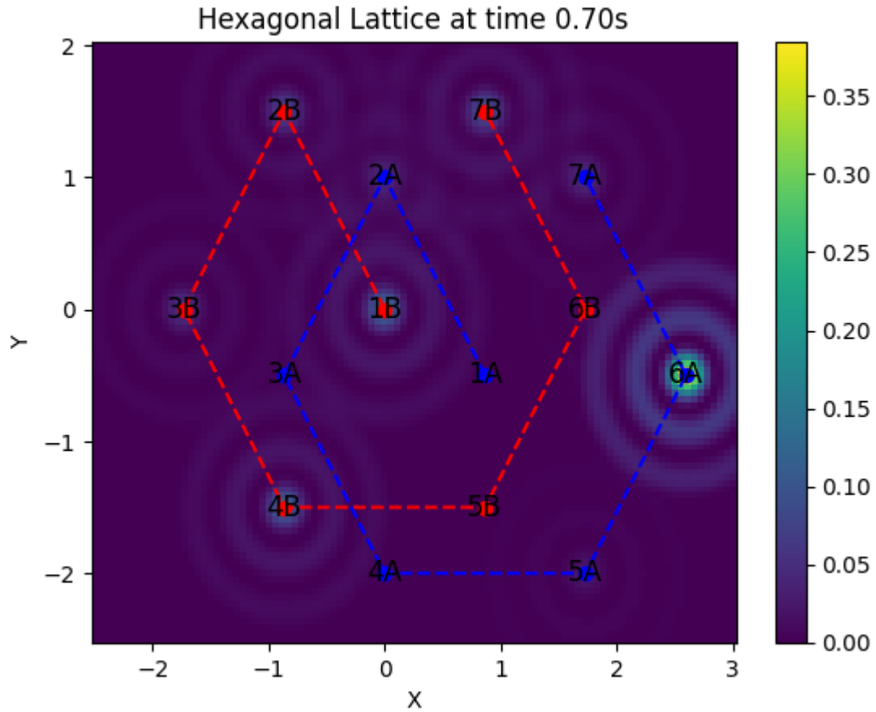

Hexagonal Lattice at time 0.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

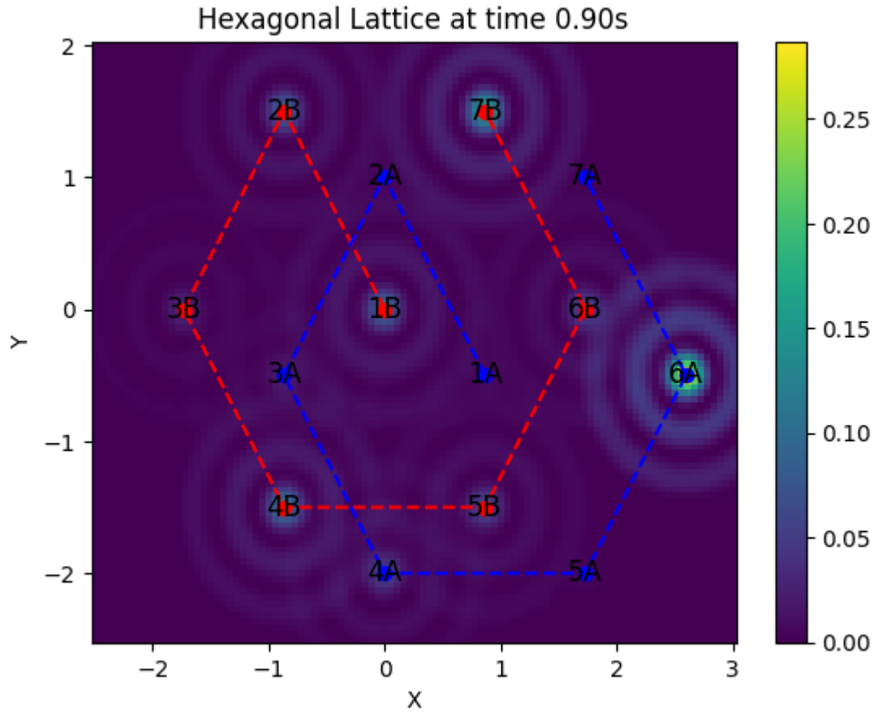

Hexagonal Lattice at time 0.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

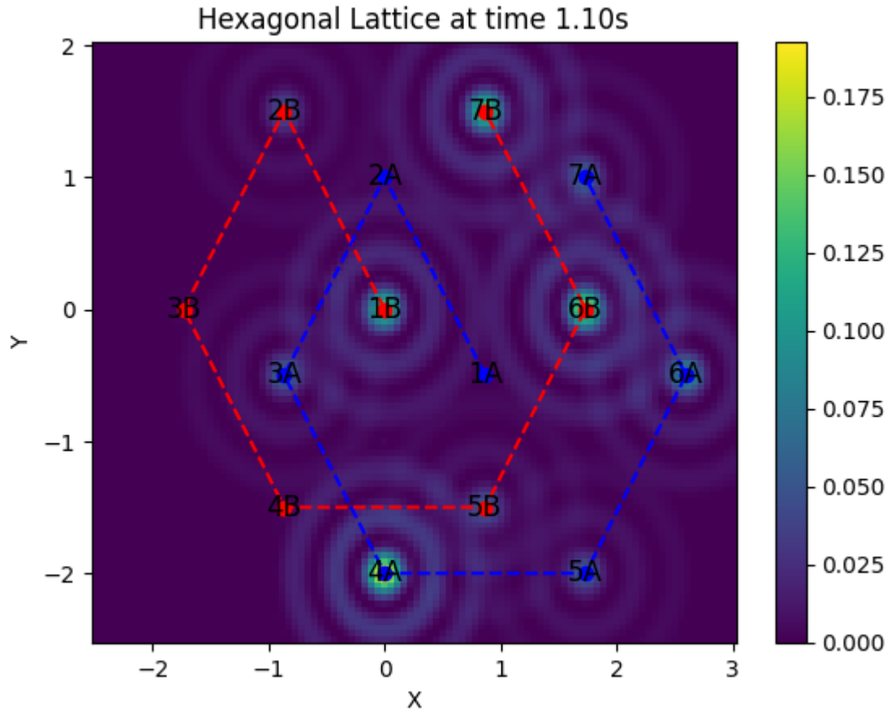

Hexagonal Lattice at time 0.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

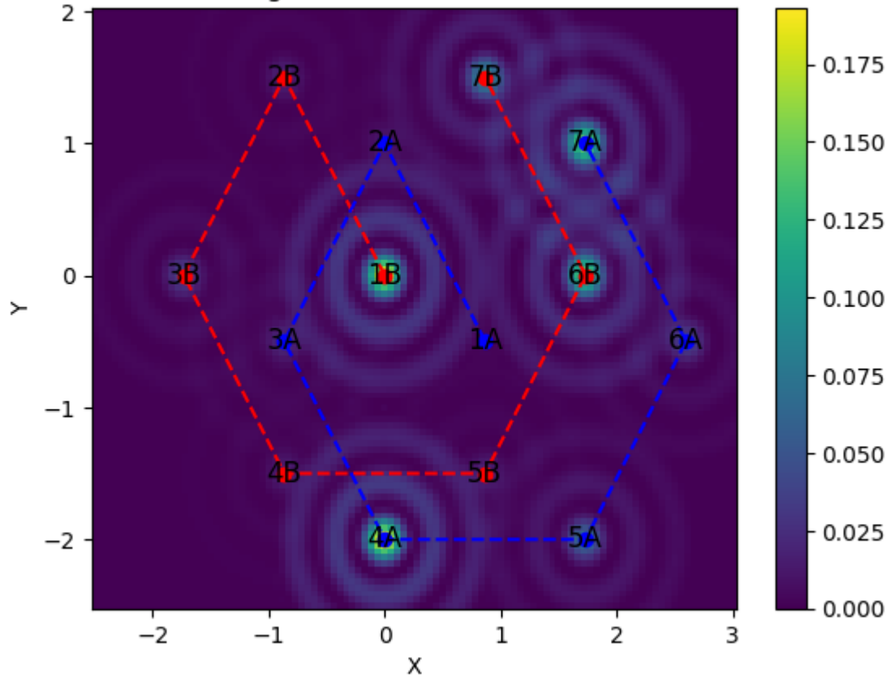

Hexagonal Lattice at time 0.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

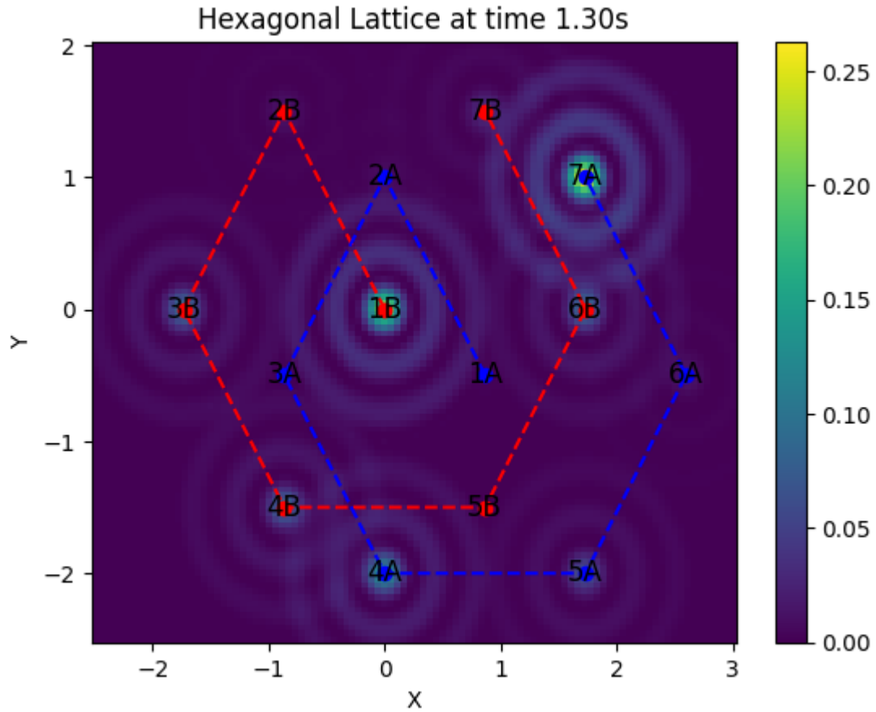

Hexagonal Lattice at time 0.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

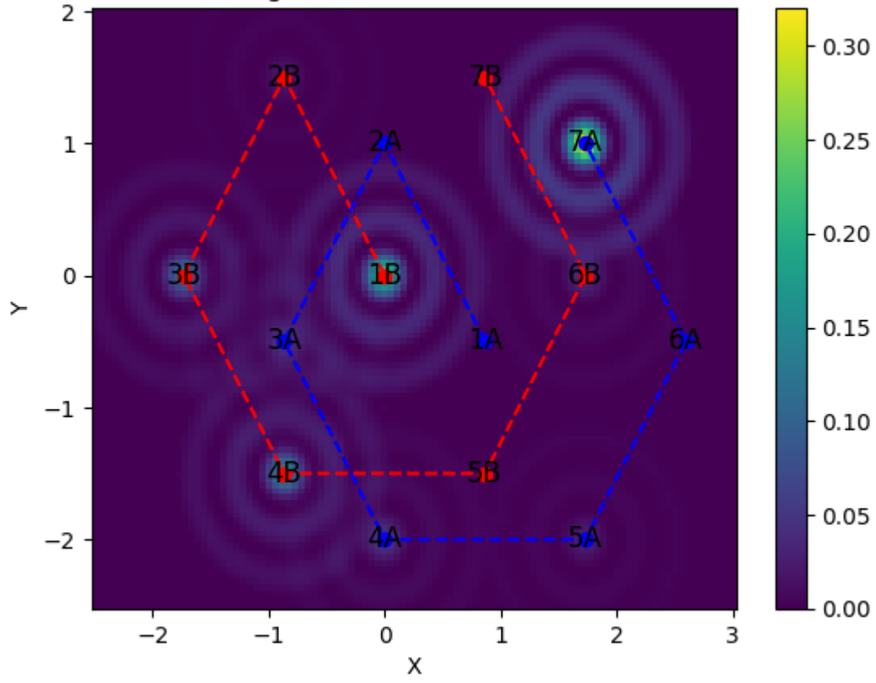

Hexagonal Lattice at time 0.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 0.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

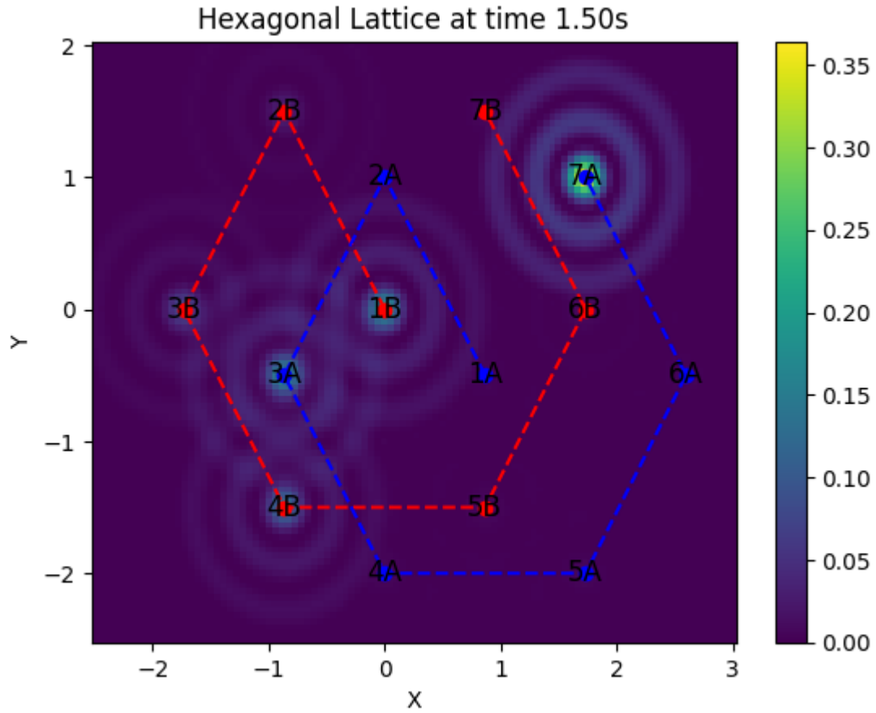

Hexagonal Lattice at time 0.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 1.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

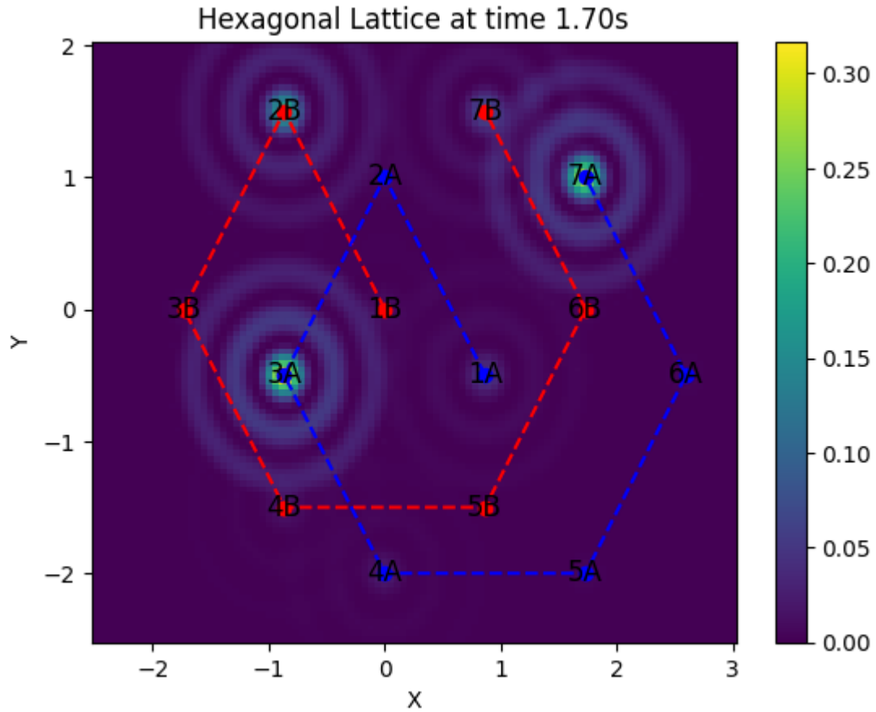

Hexagonal Lattice at time 1.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

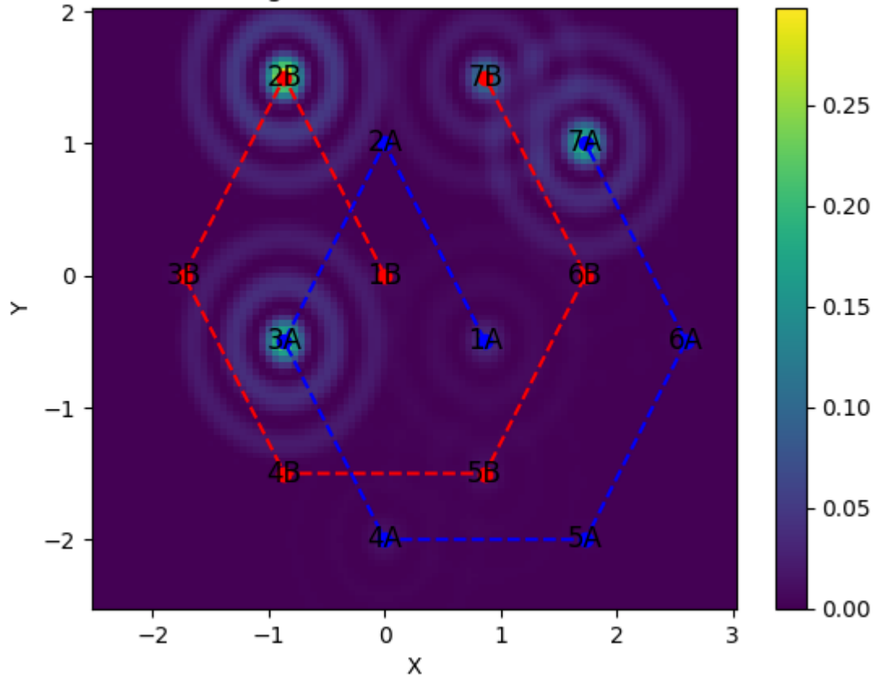

Hexagonal Lattice at time 1.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

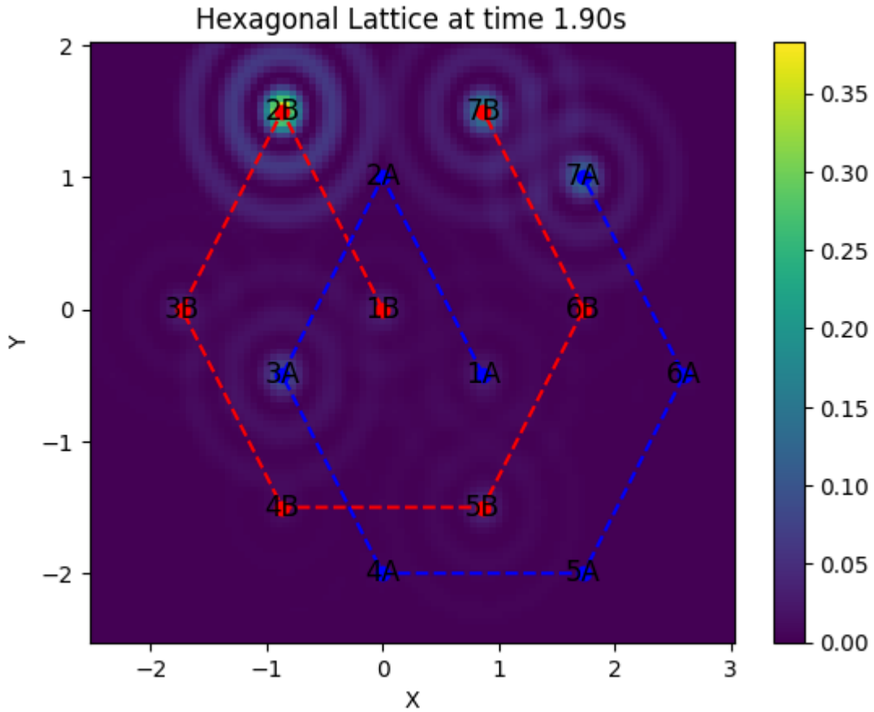

Hexagonal Lattice at time 1.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

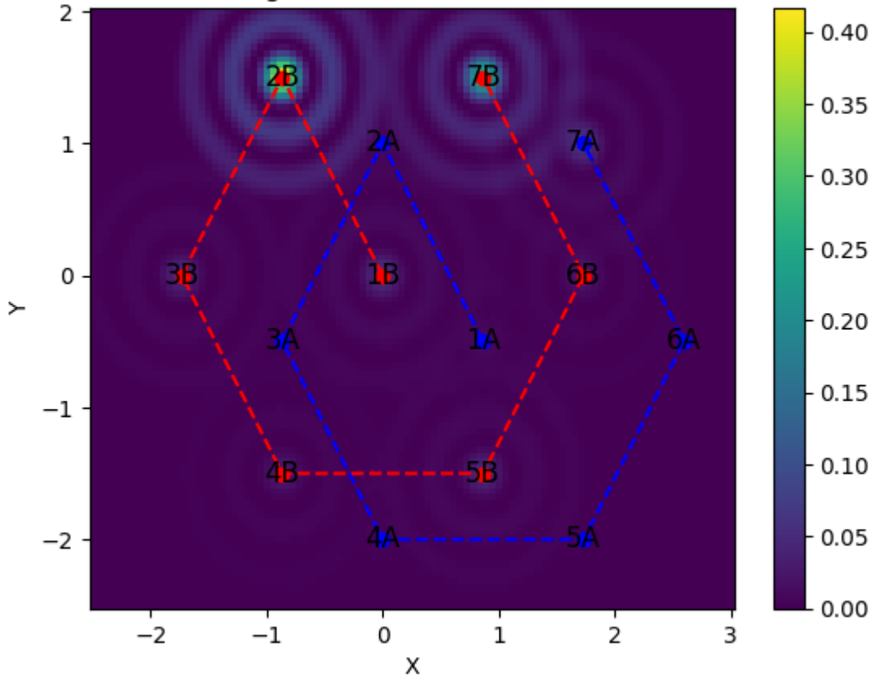

Hexagonal Lattice at time 1.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

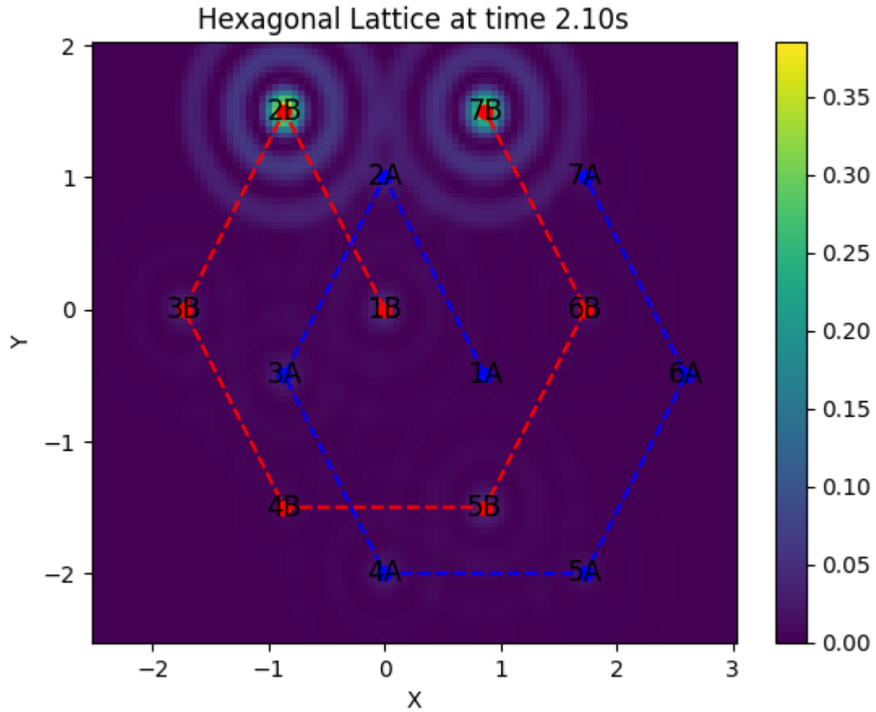

Hexagonal Lattice at time 1.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 1.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

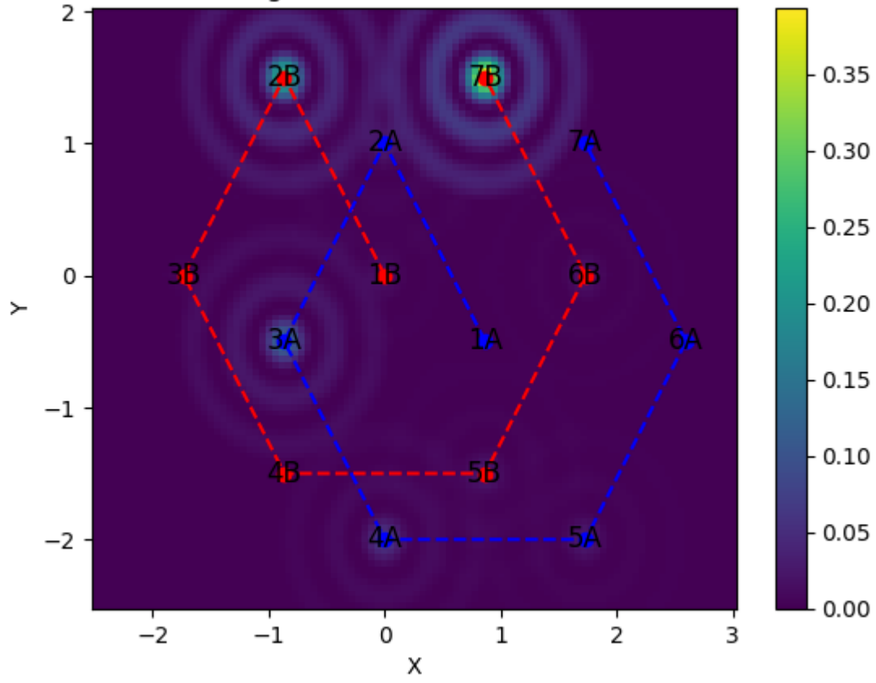

Hexagonal Lattice at time 1.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

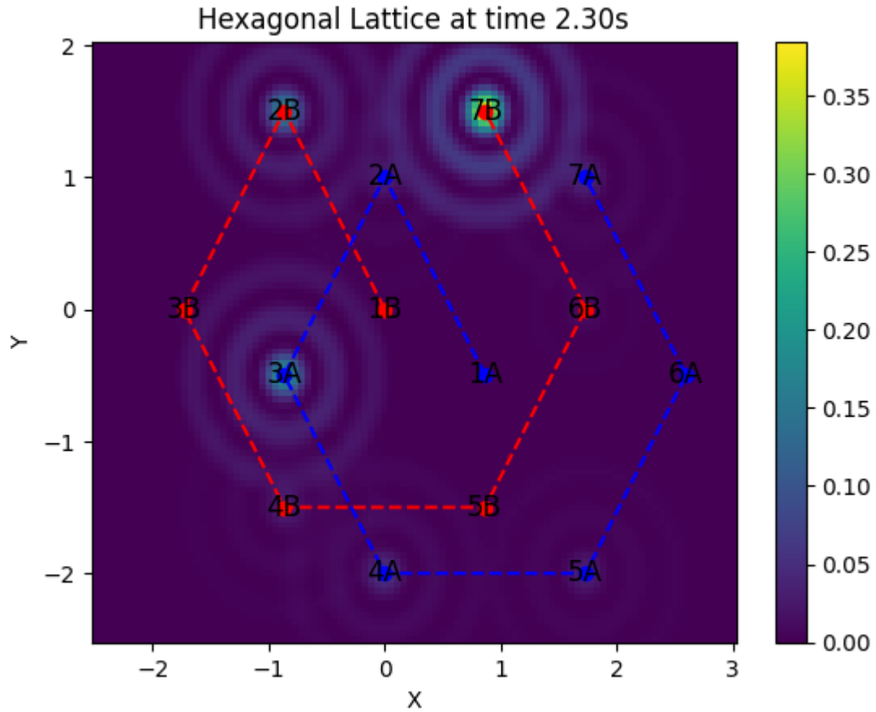

Hexagonal Lattice at time 1.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

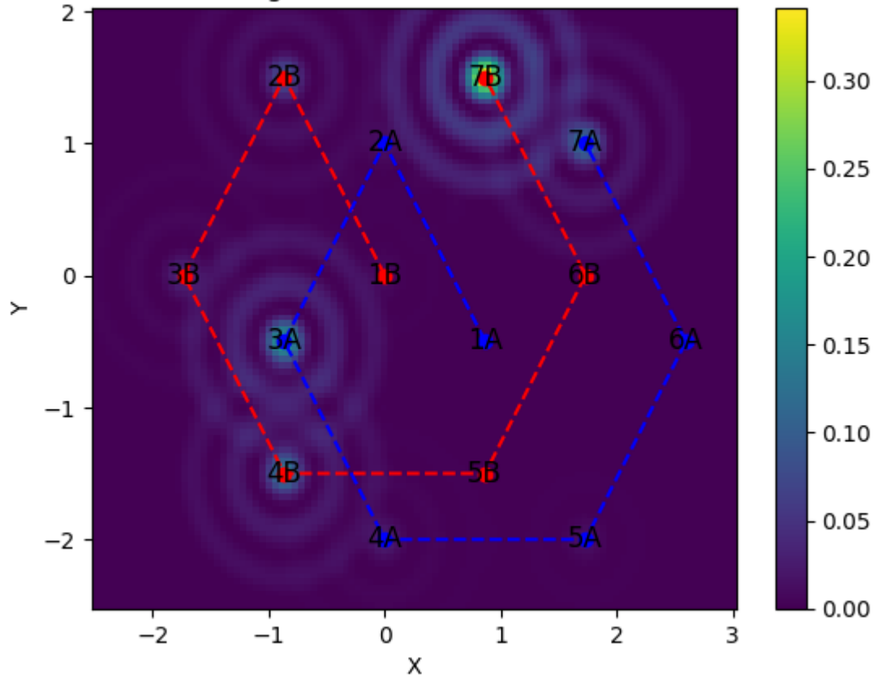

Hexagonal Lattice at time 1.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

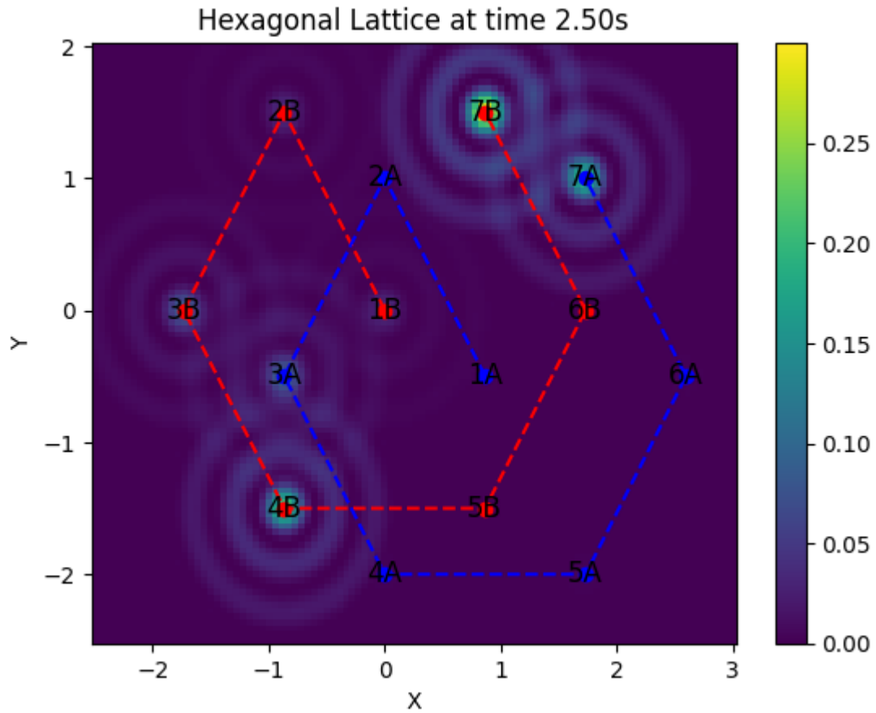

Hexagonal Lattice at time 2.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

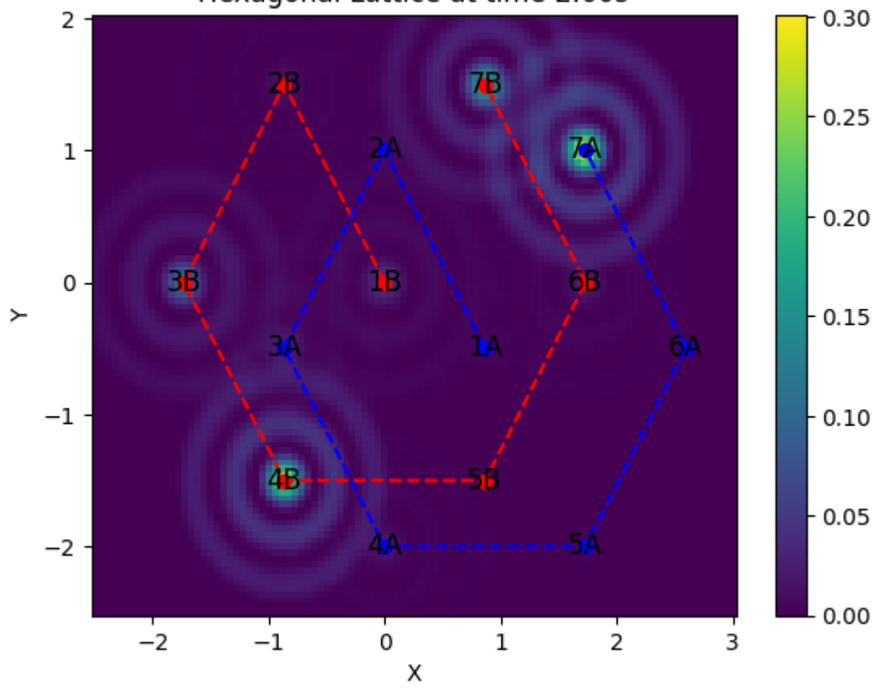

Hexagonal Lattice at time 2.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

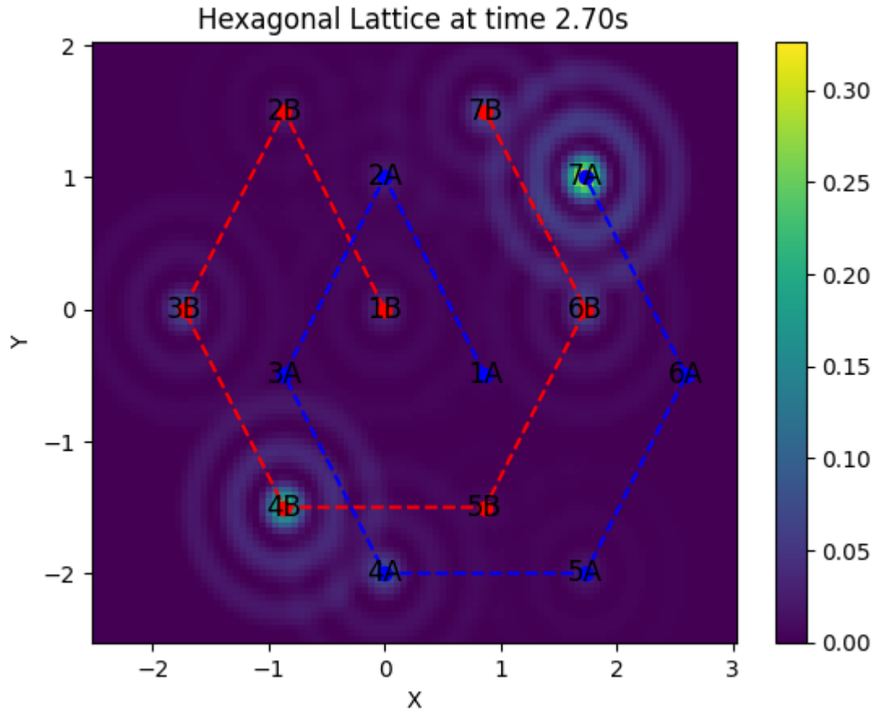

Hexagonal Lattice at time 2.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

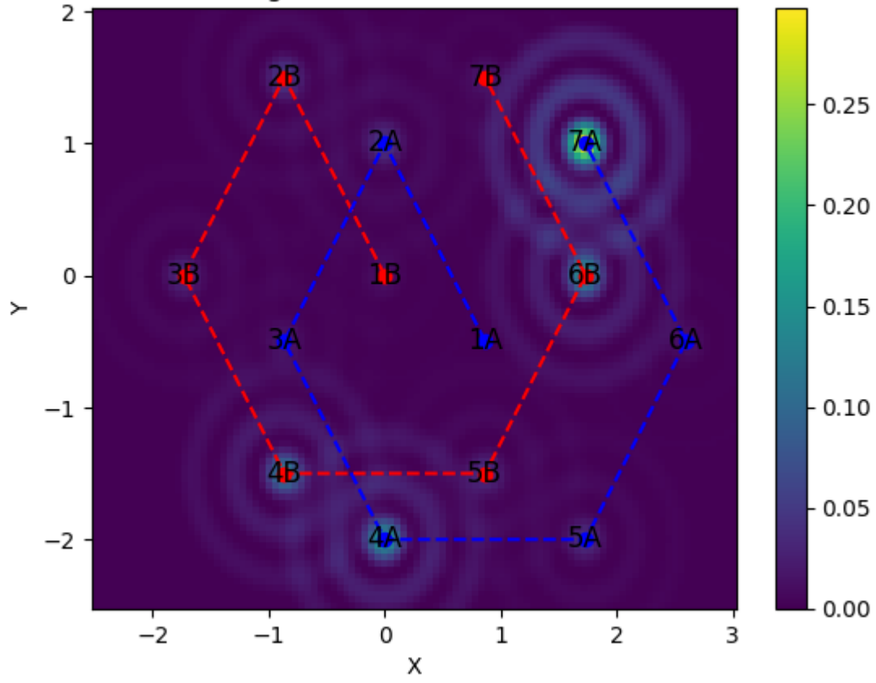

Hexagonal Lattice at time 2.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

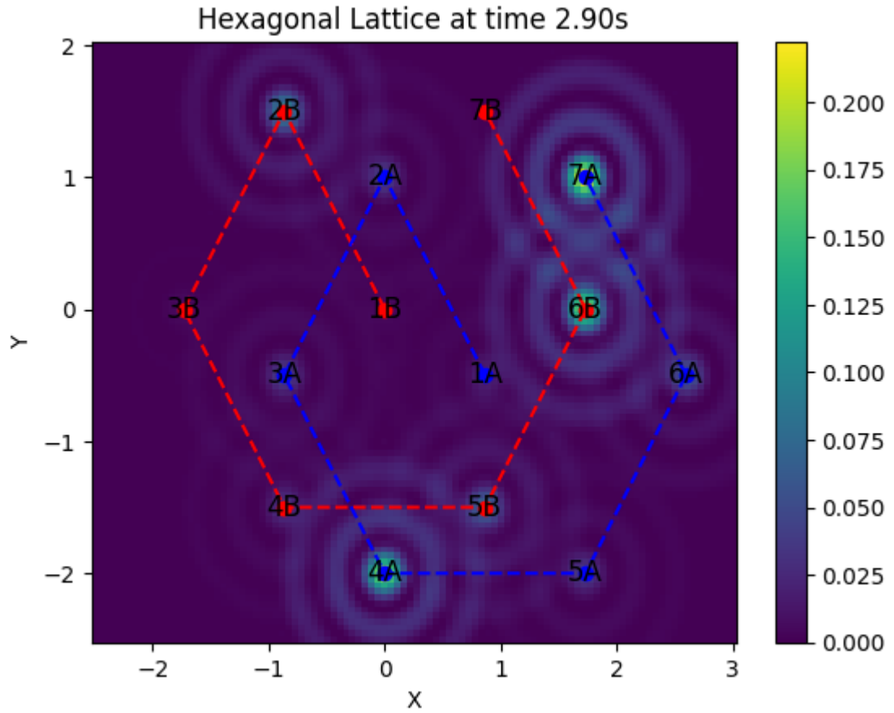

Hexagonal Lattice at time 2.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

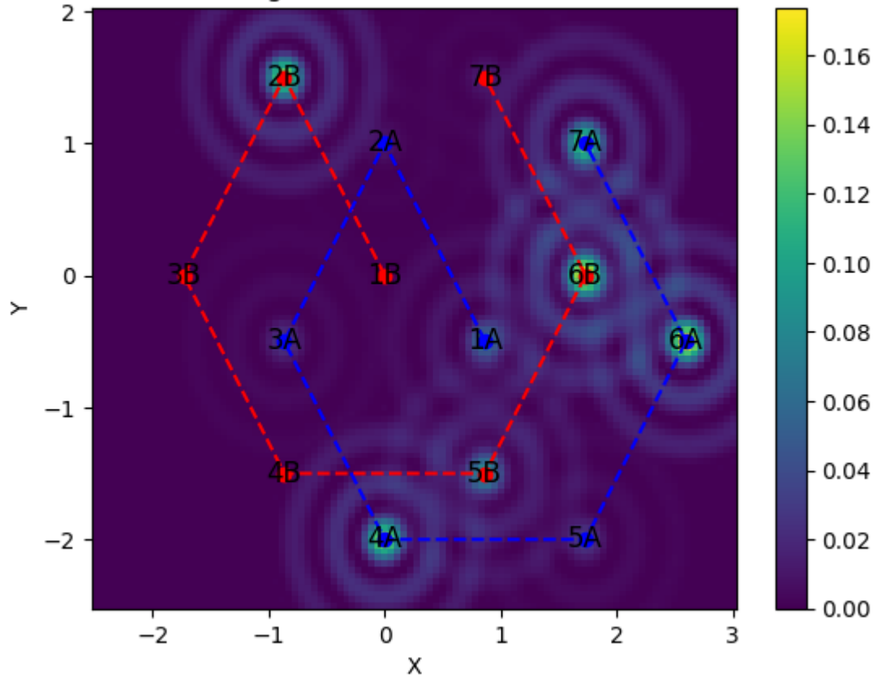

Hexagonal Lattice at time 2.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

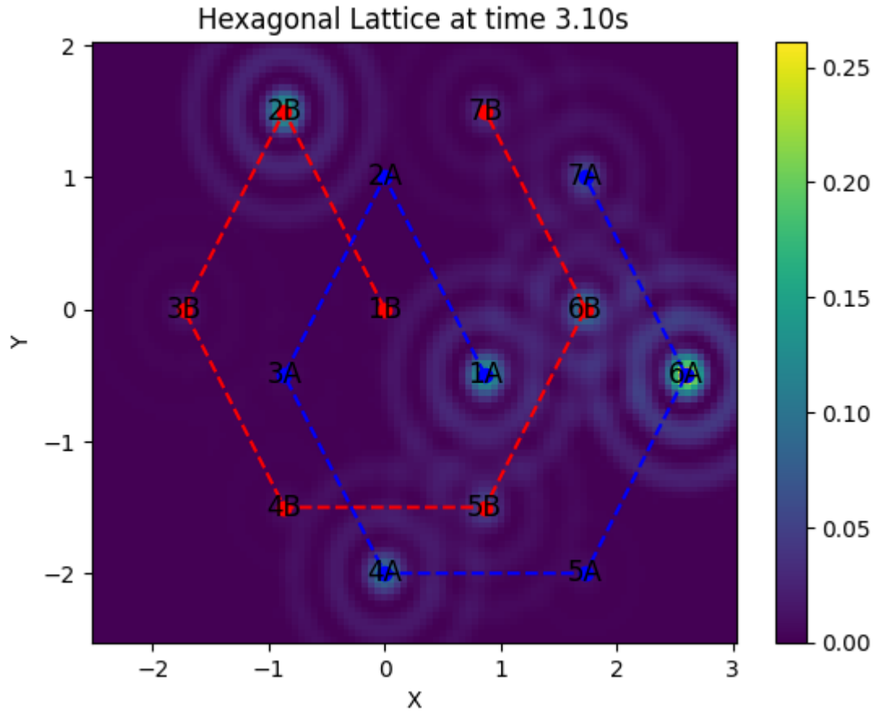

Hexagonal Lattice at time 2.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

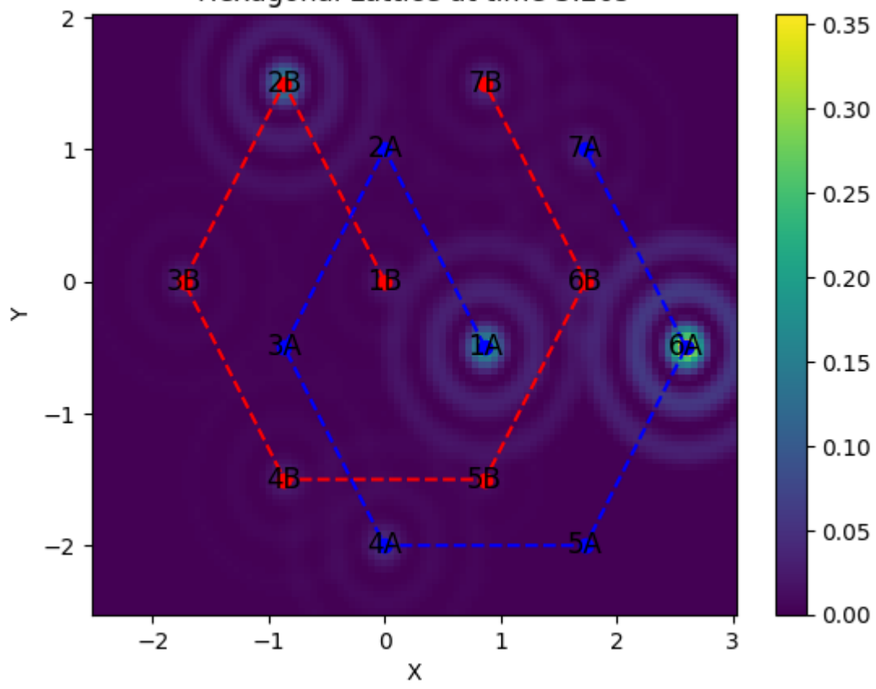

Hexagonal Lattice at time 2.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

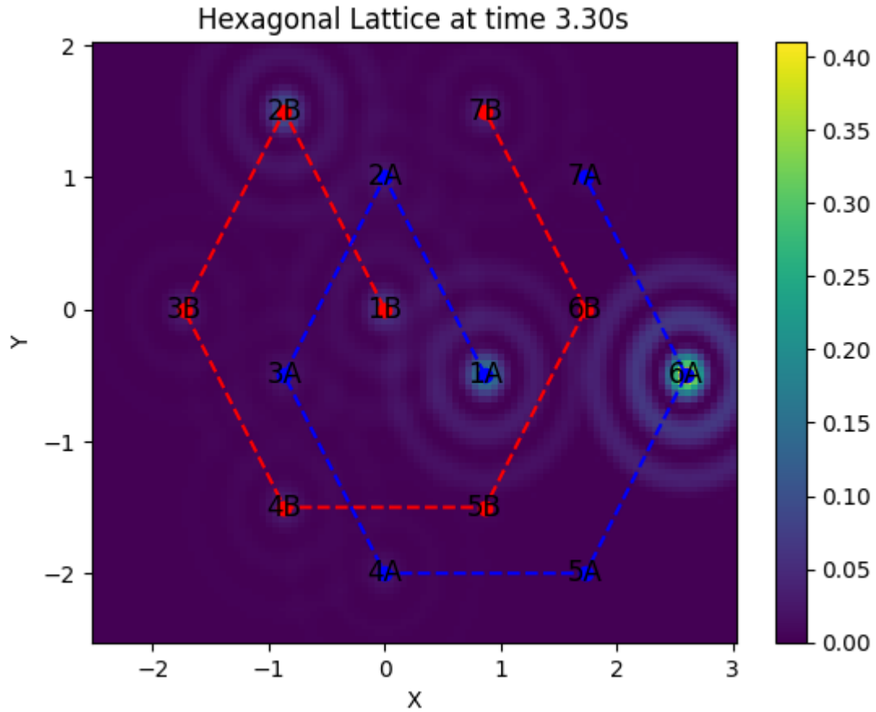

Hexagonal Lattice at time 2.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

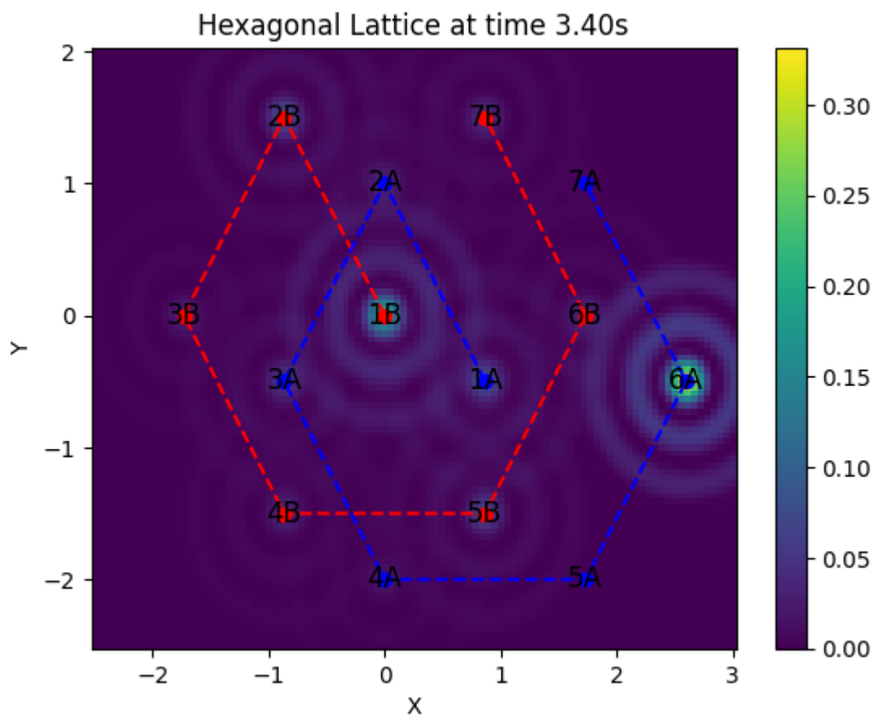

Hexagonal Lattice at time 2.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

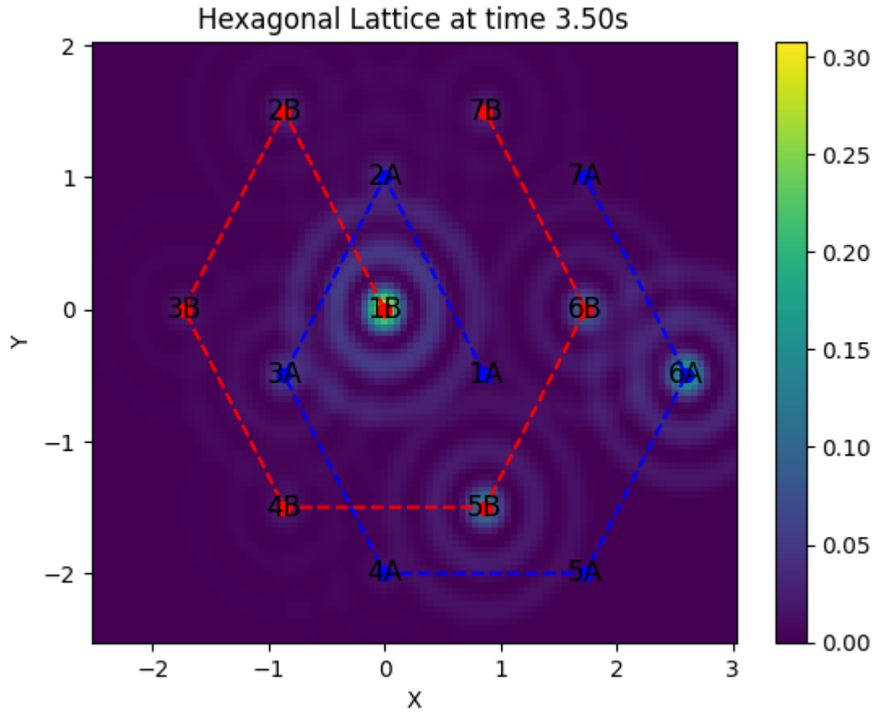

Hexagonal Lattice at time 3.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00

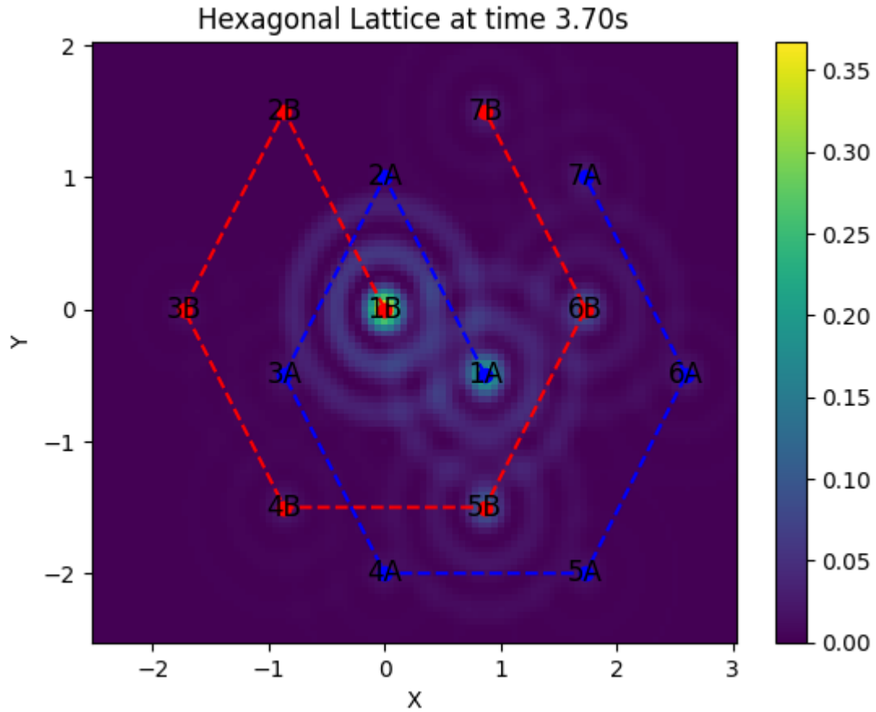

Hexagonal Lattice at time 3.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

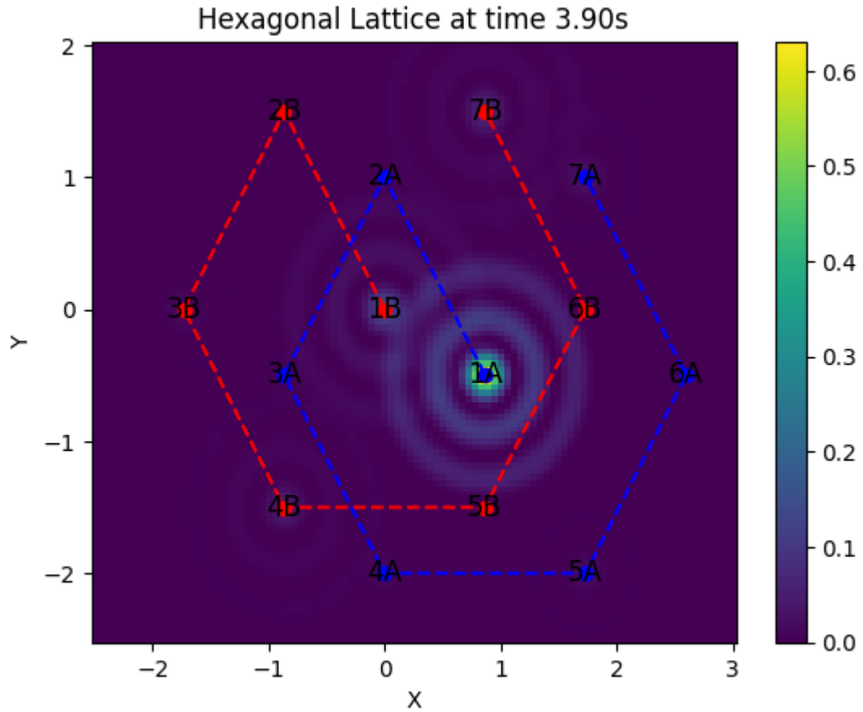

Hexagonal Lattice at time 3.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

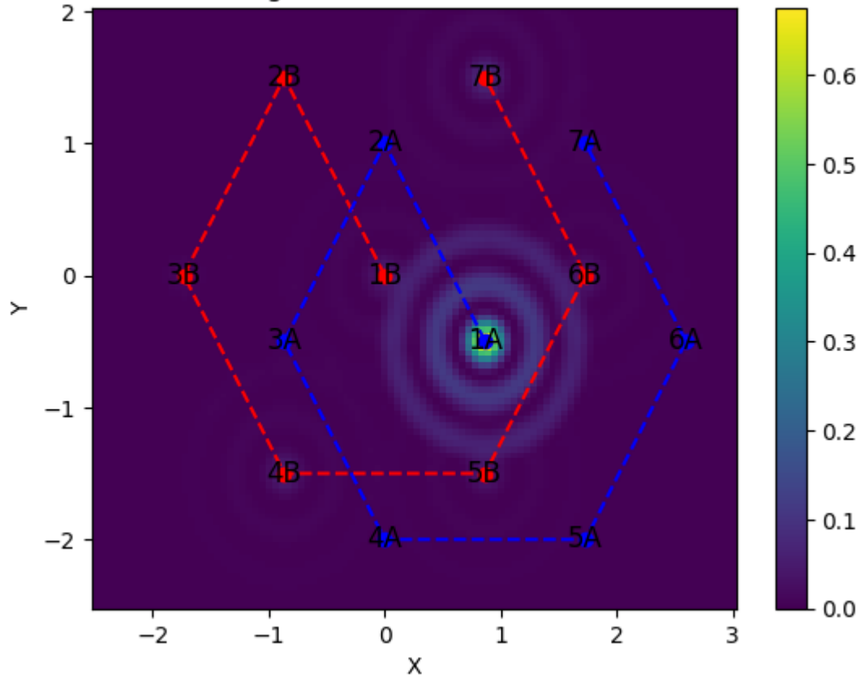

Hexagonal Lattice at time 3.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

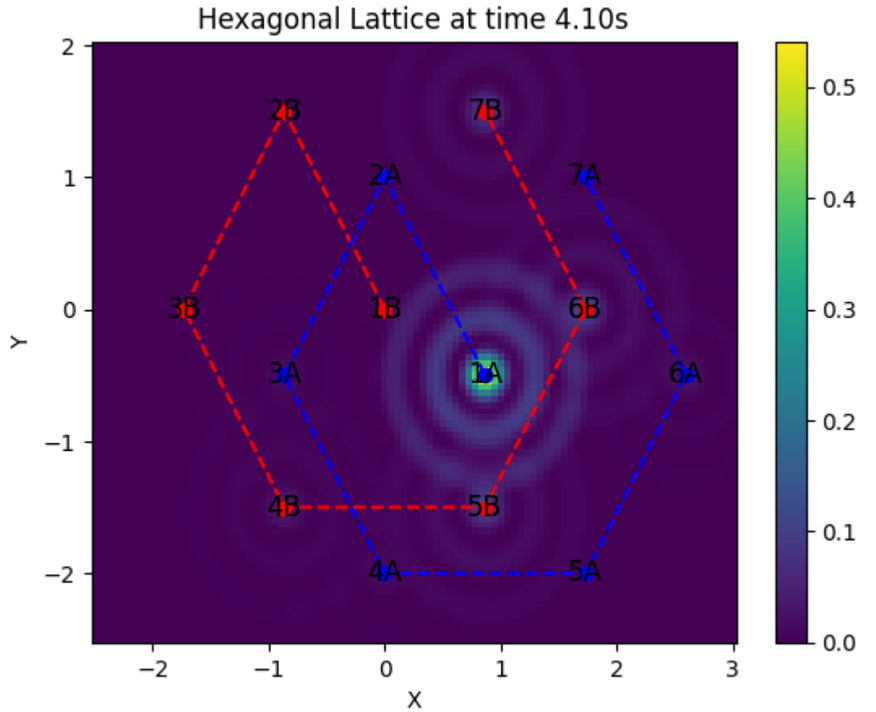

Hexagonal Lattice at time 3.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

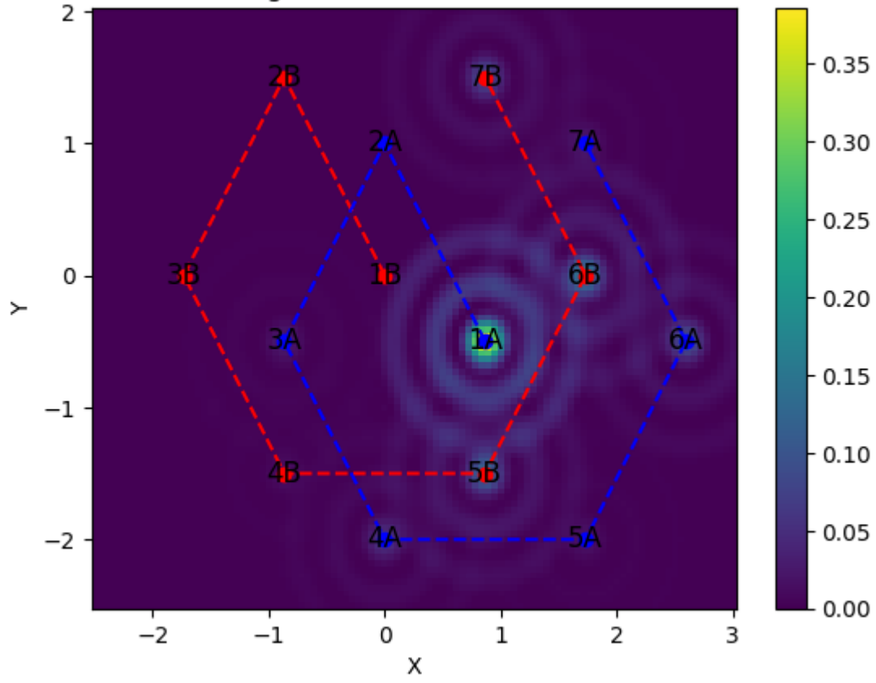

Hexagonal Lattice at time 3.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 3.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

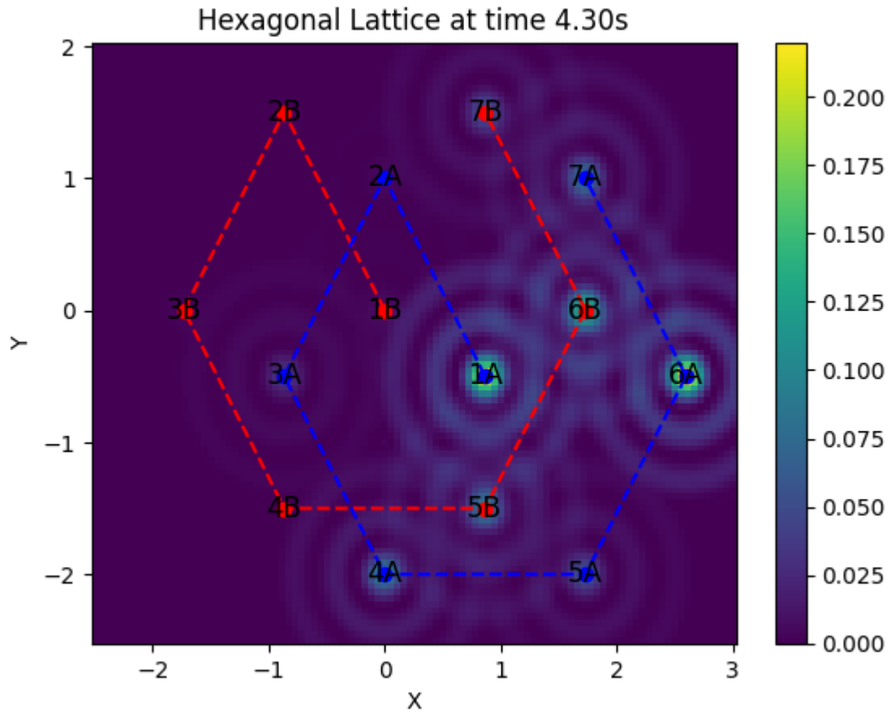

Hexagonal Lattice at time 3.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 3.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

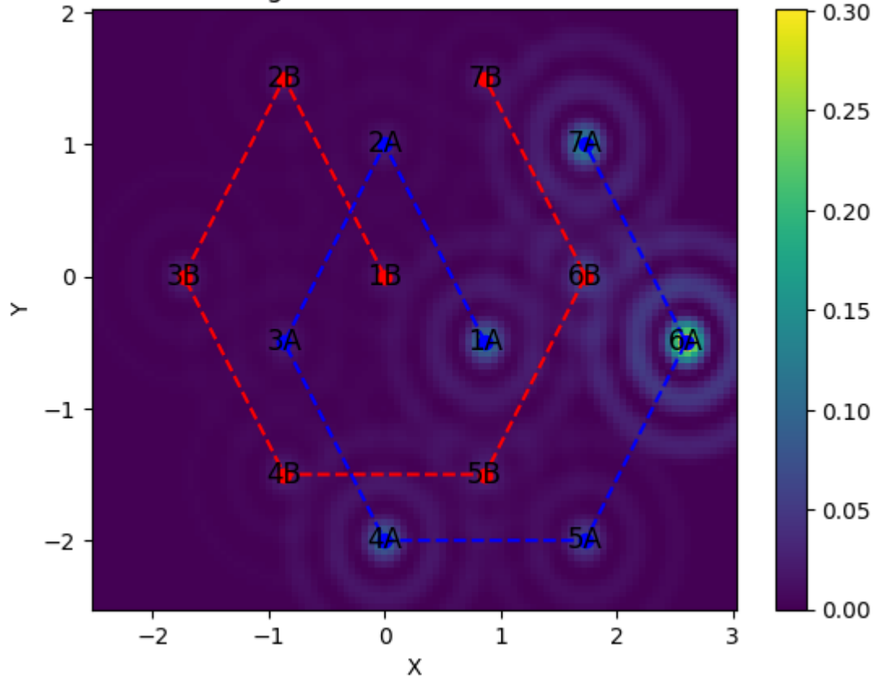

Hexagonal Lattice at time 3.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

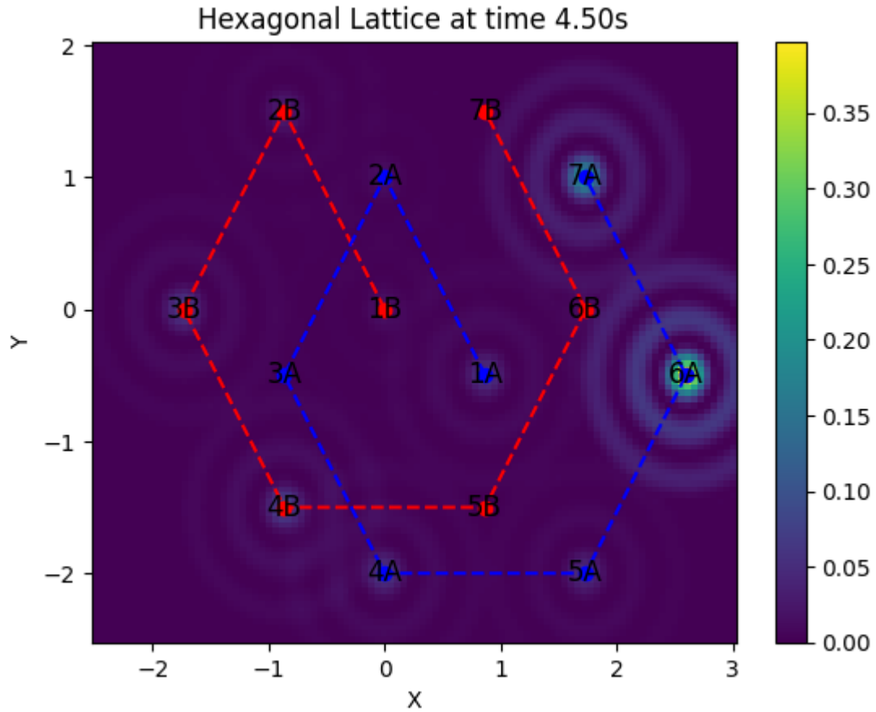

Hexagonal Lattice at time 4.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

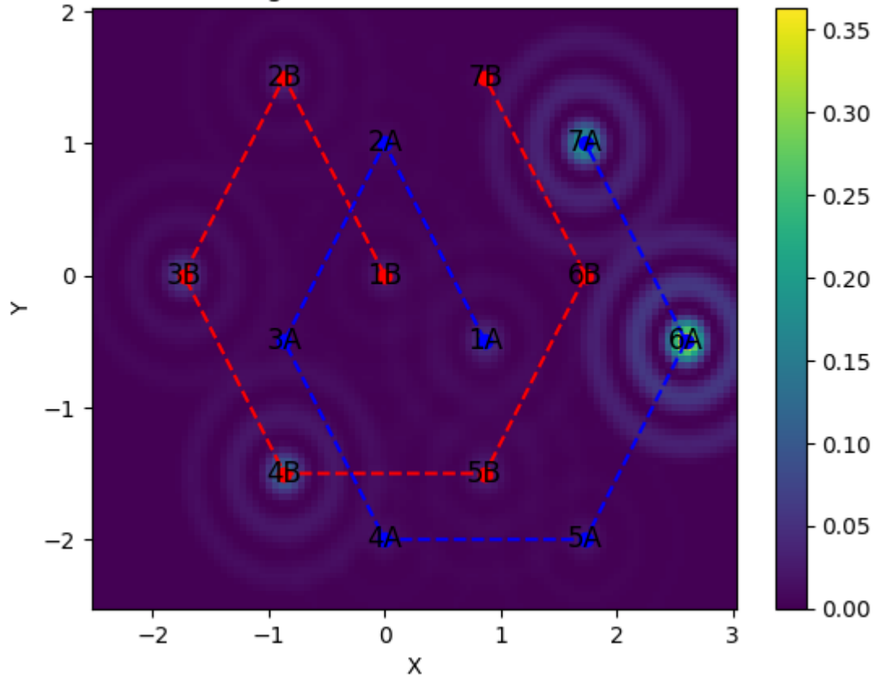

Hexagonal Lattice at time 4.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

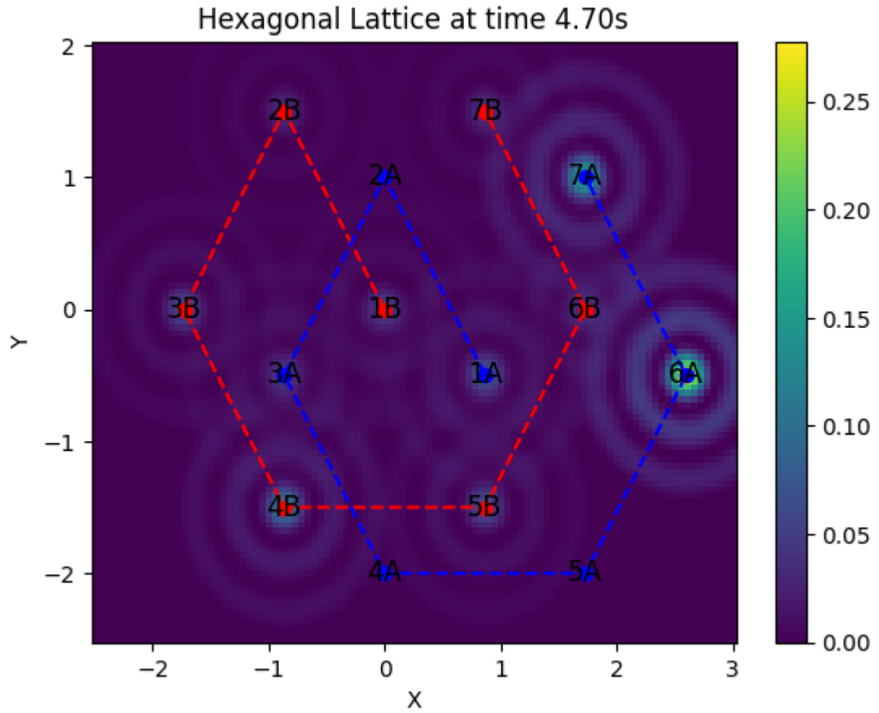

Hexagonal Lattice at time 4.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

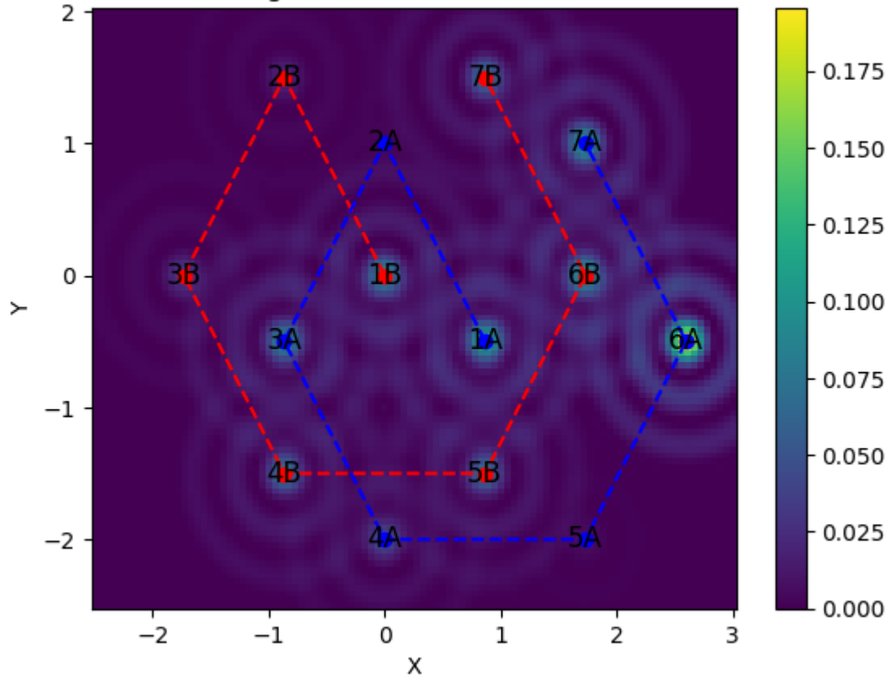

Hexagonal Lattice at time 4.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

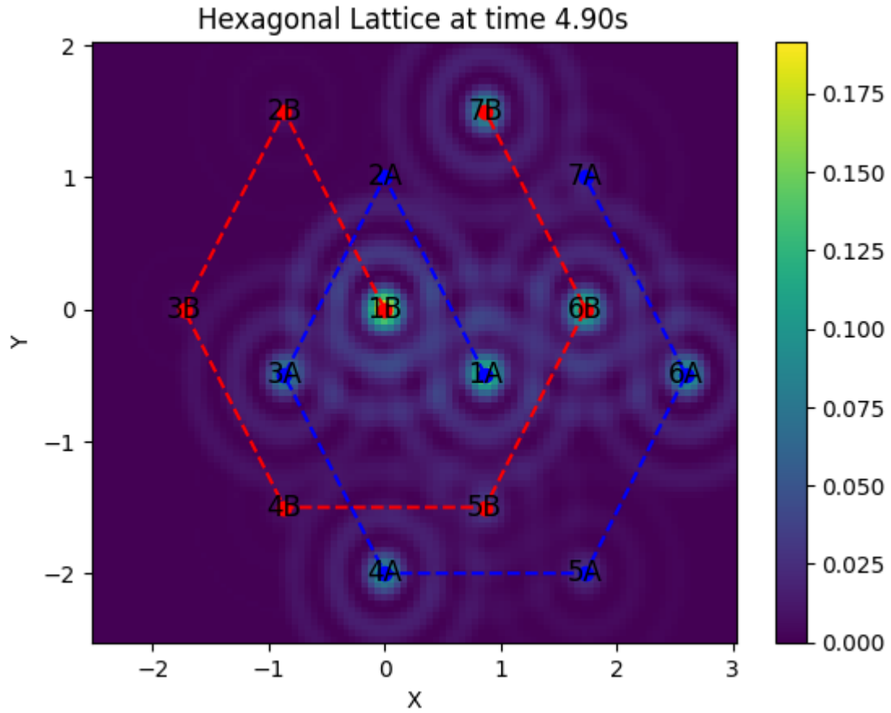

Hexagonal Lattice at time 4.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

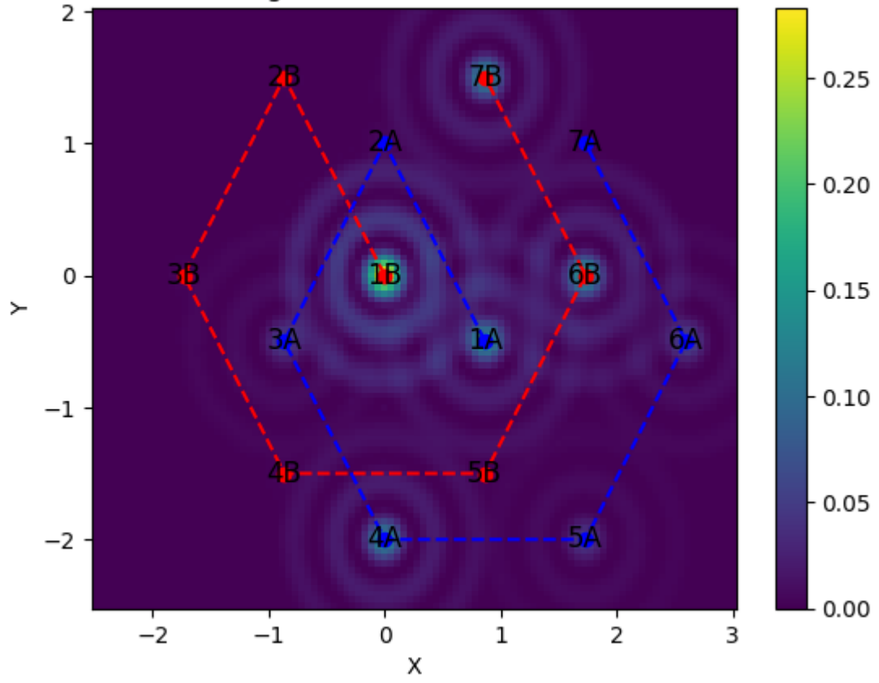

Hexagonal Lattice at time 4.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

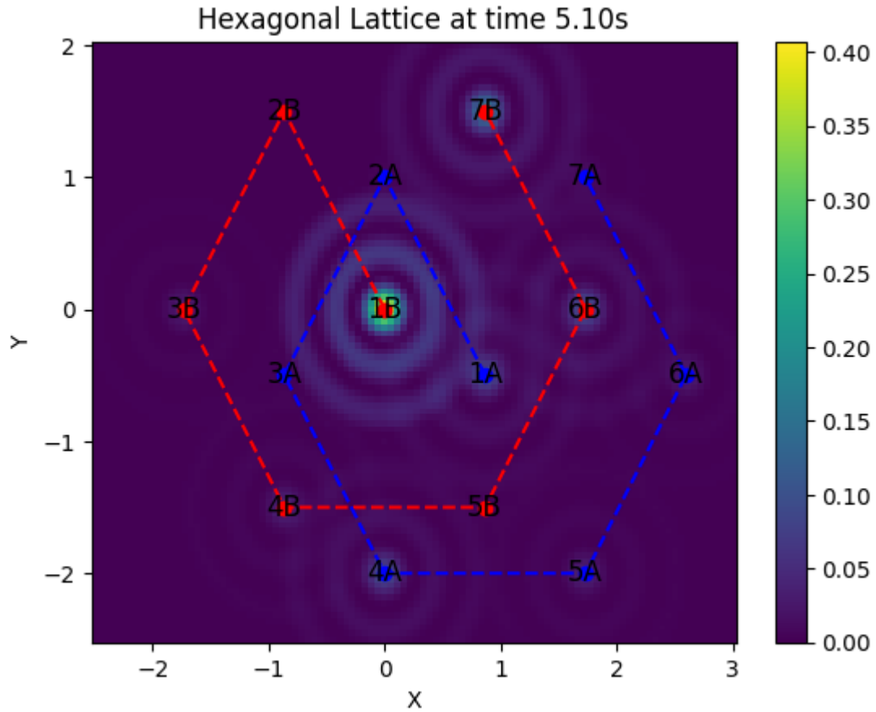

Hexagonal Lattice at time 4.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

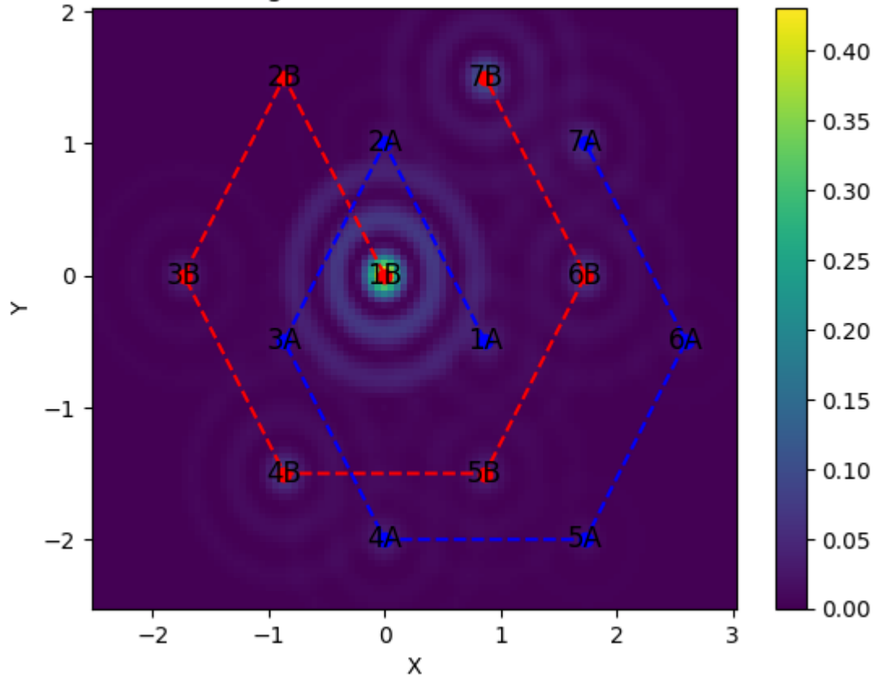

Hexagonal Lattice at time 4.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

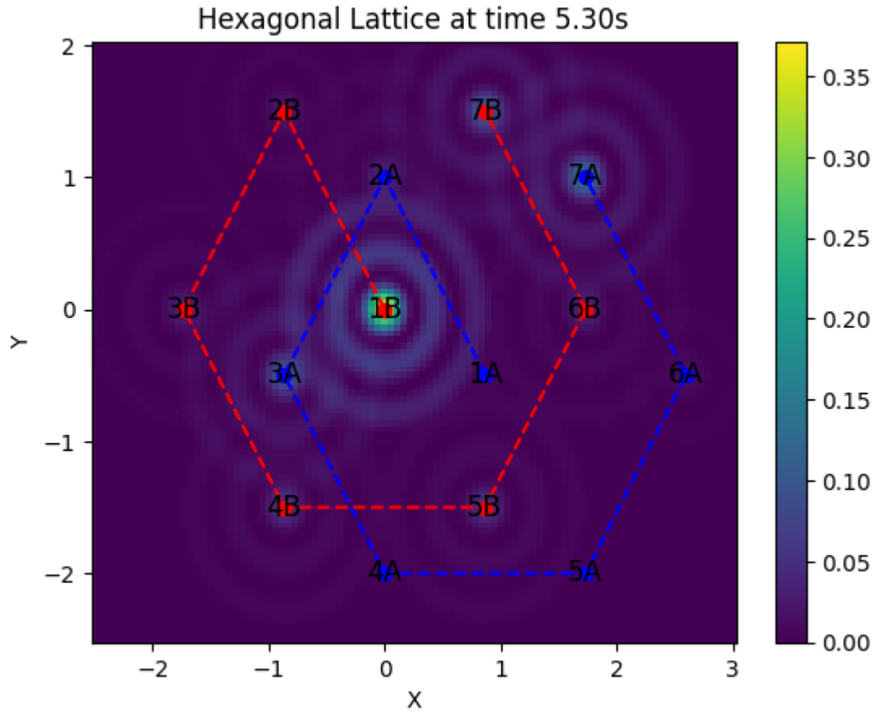

Hexagonal Lattice at time 4.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

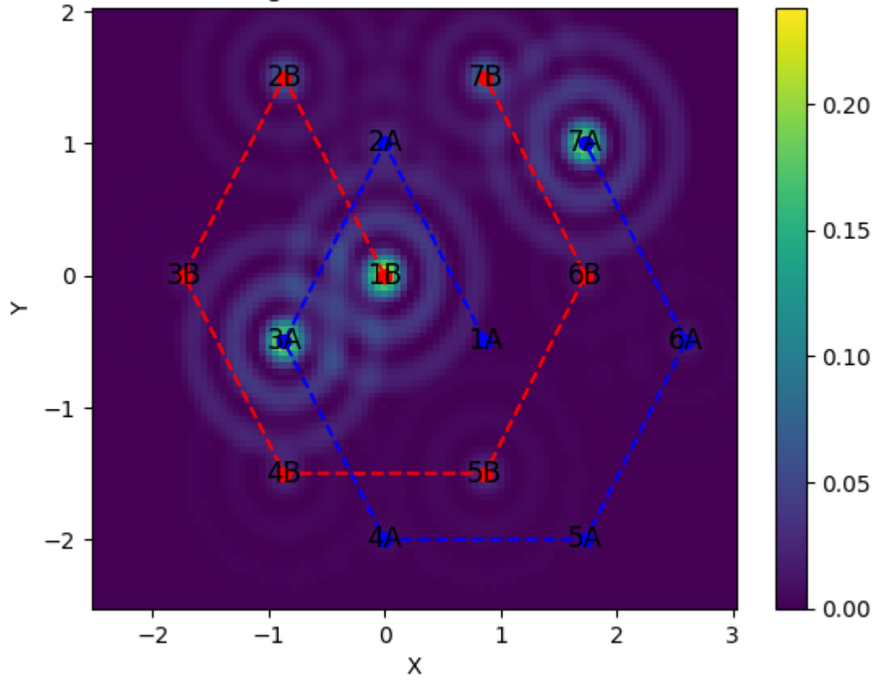

Hexagonal Lattice at time 4.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

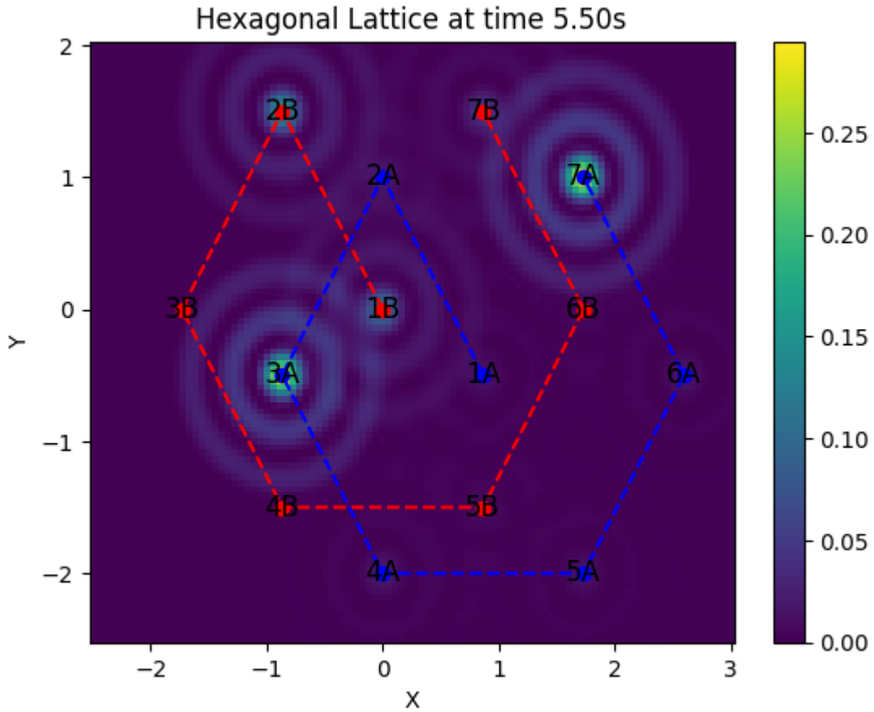

Hexagonal Lattice at time 5.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

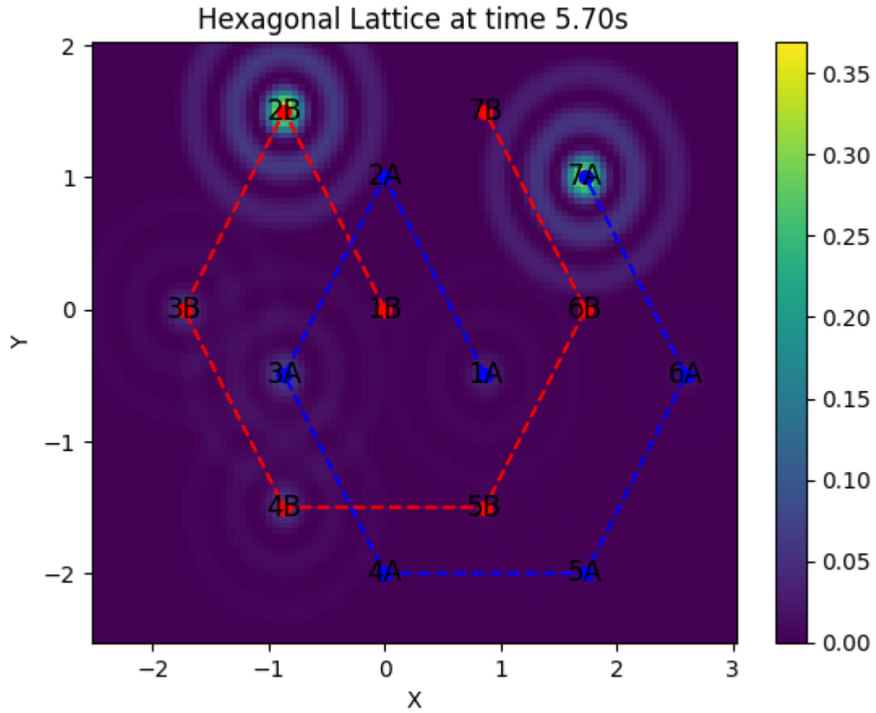

Hexagonal Lattice at time 5.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

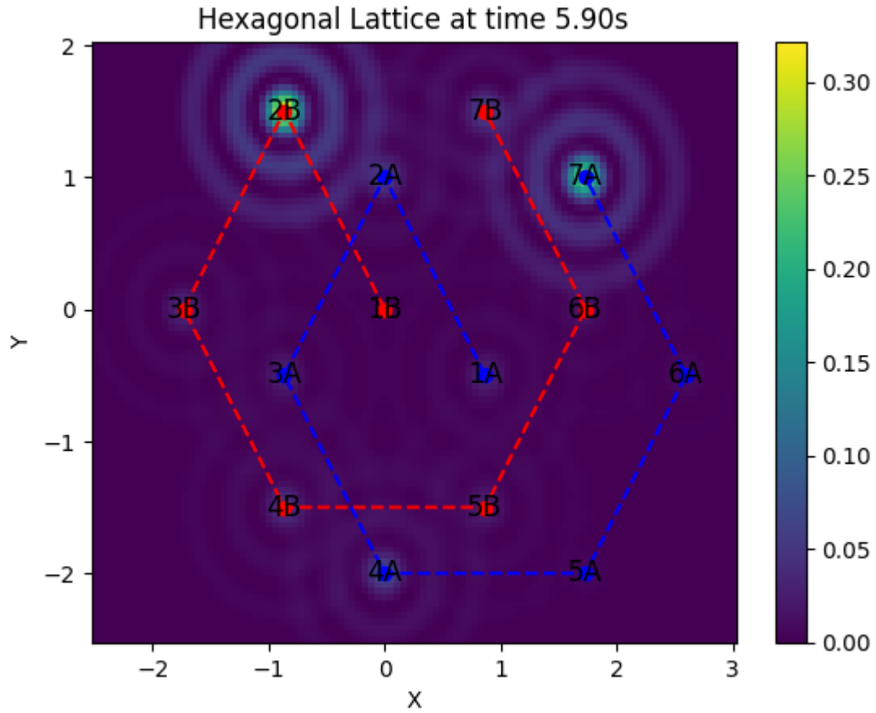

Hexagonal Lattice at time 5.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

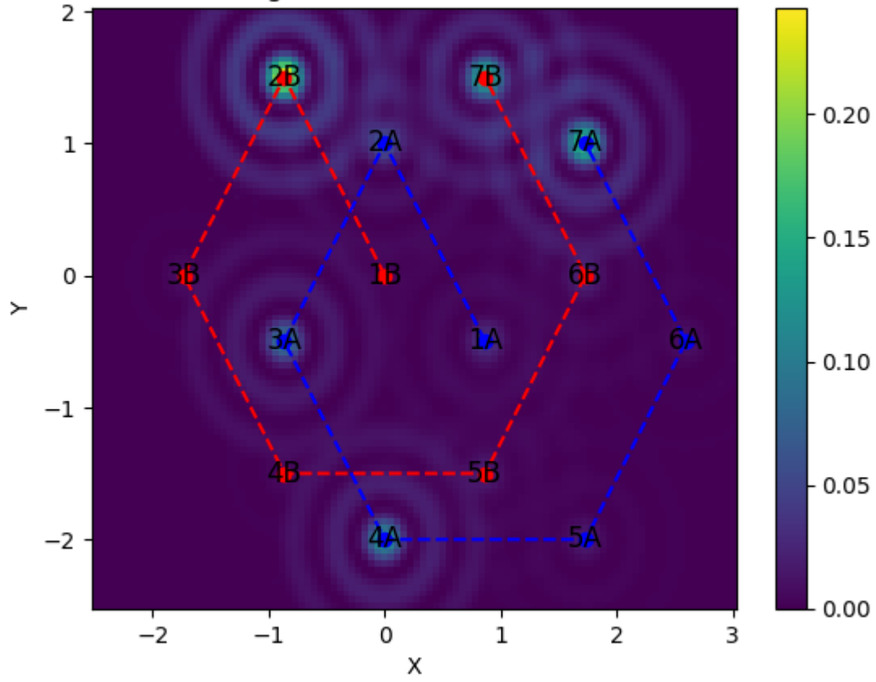

Hexagonal Lattice at time 5.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

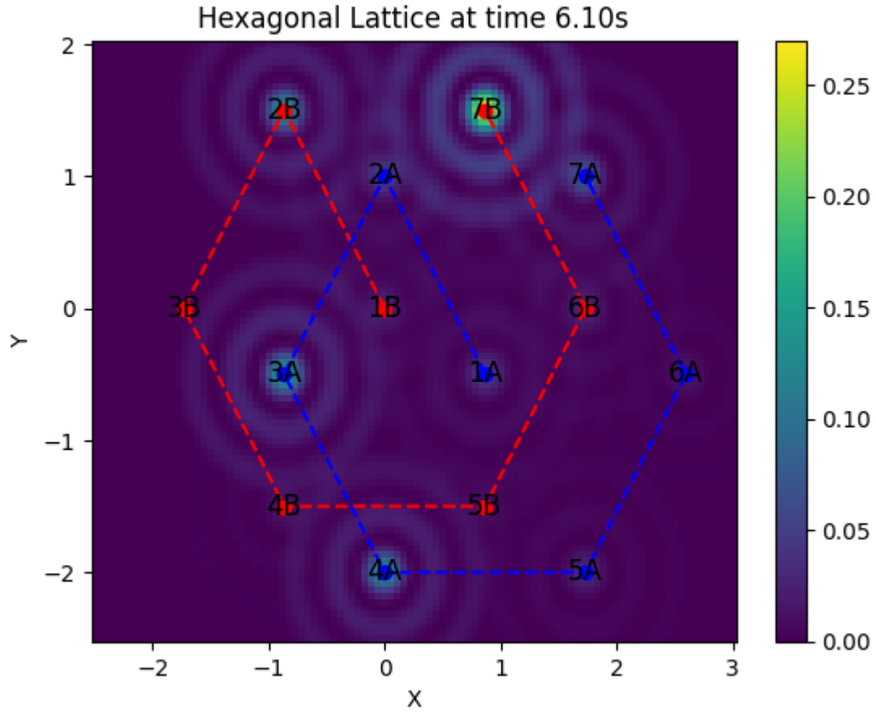

Hexagonal Lattice at time 5.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

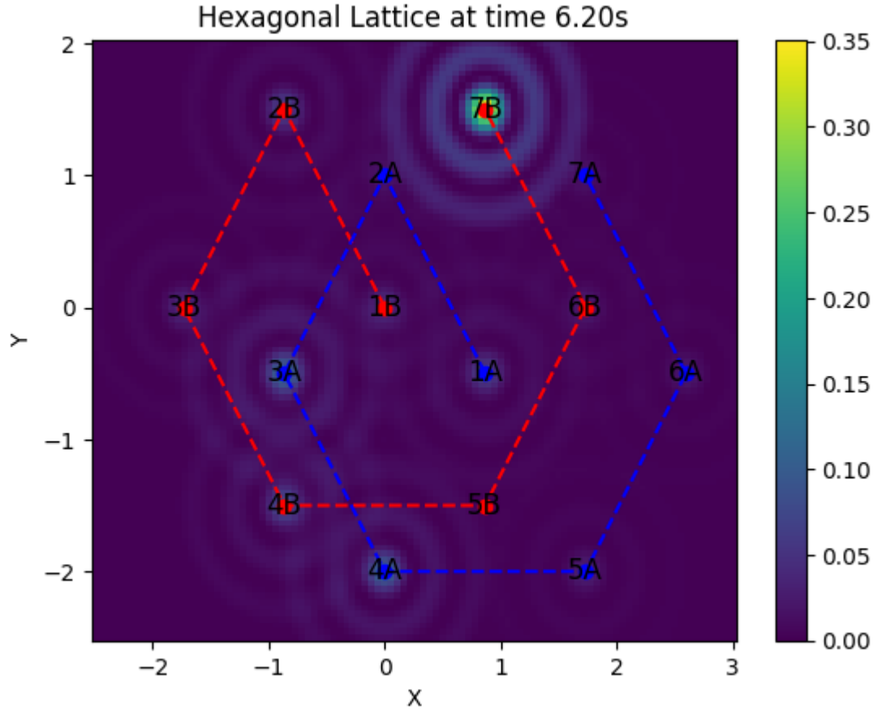

Hexagonal Lattice at time 5.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 5.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

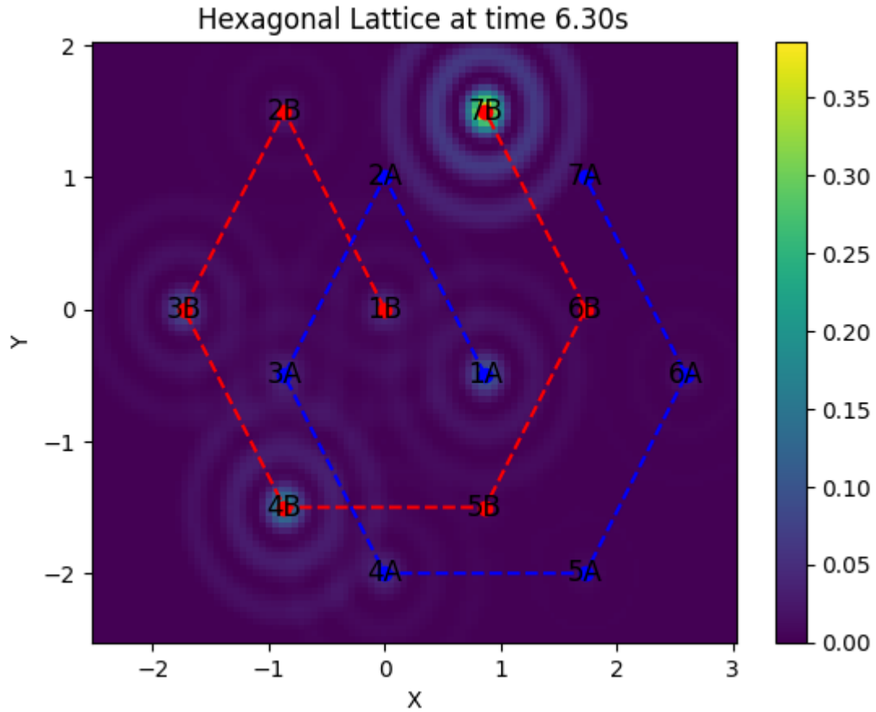

Hexagonal Lattice at time 5.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 5.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

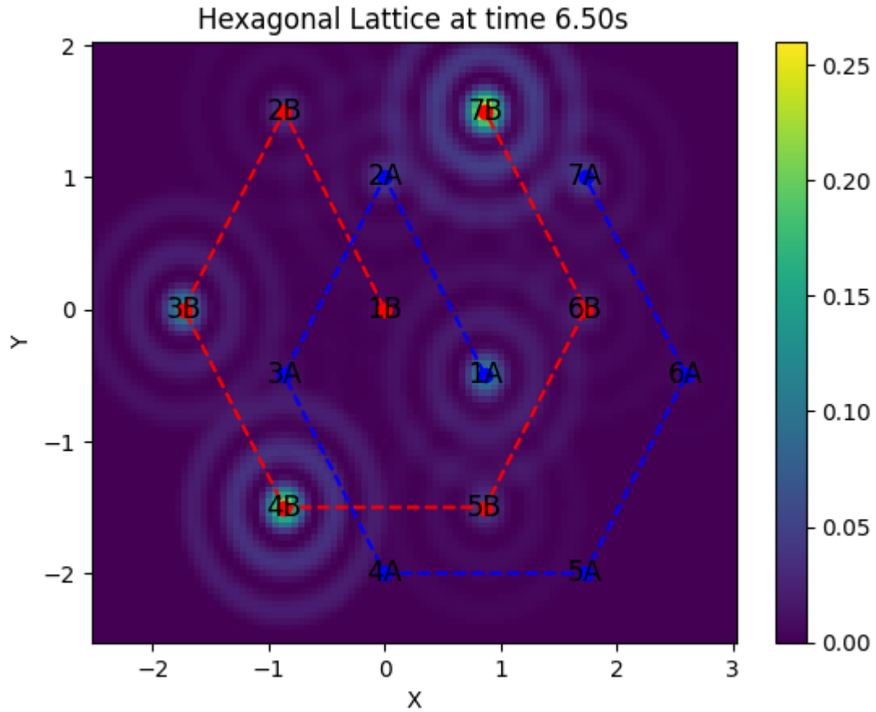

Hexagonal Lattice at time 5.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

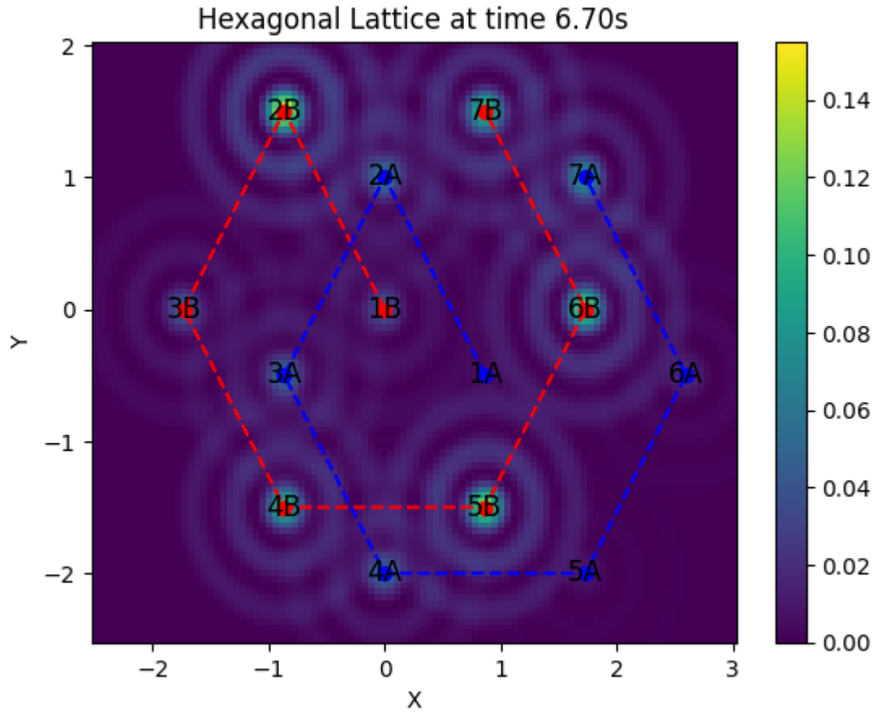

Hexagonal Lattice at time 6.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

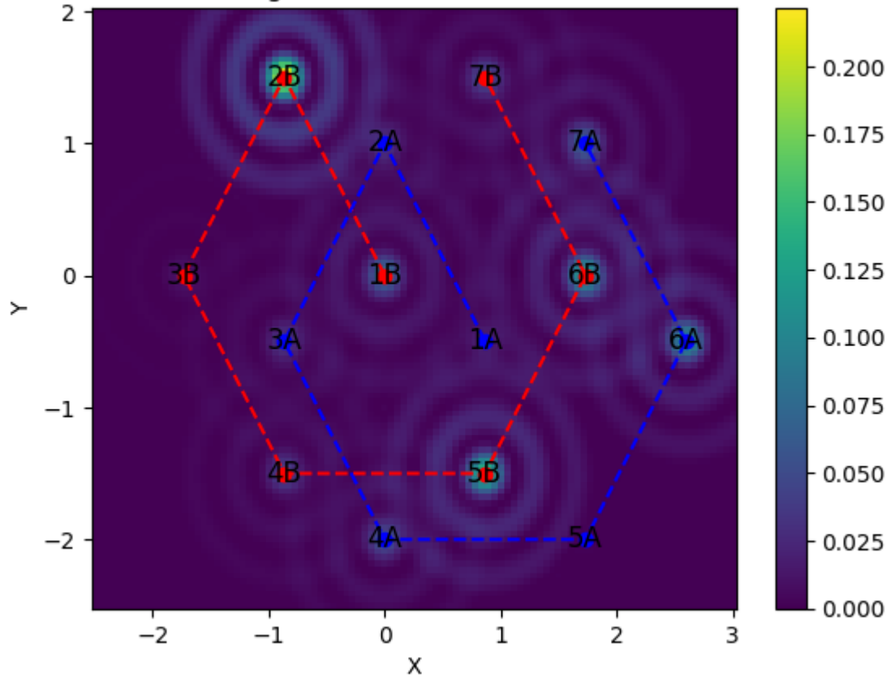

Hexagonal Lattice at time 6.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

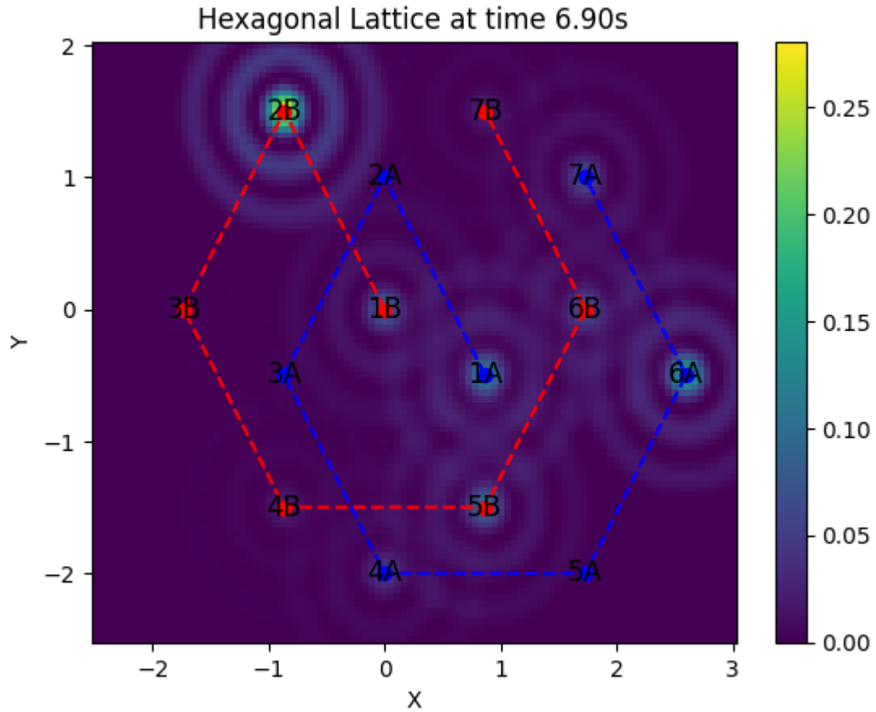

Hexagonal Lattice at time 6.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

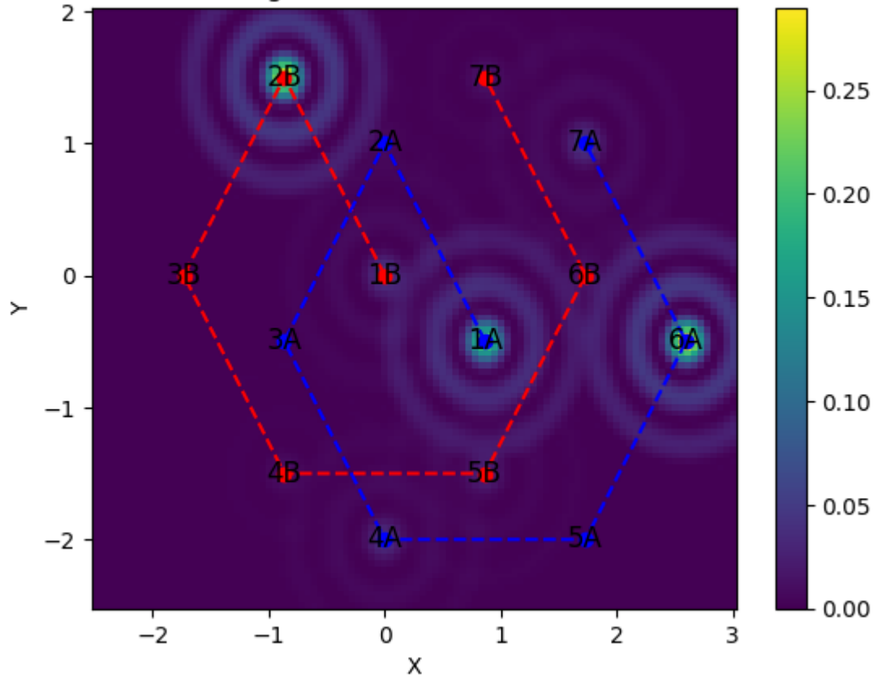

Hexagonal Lattice at time 6.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 6.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

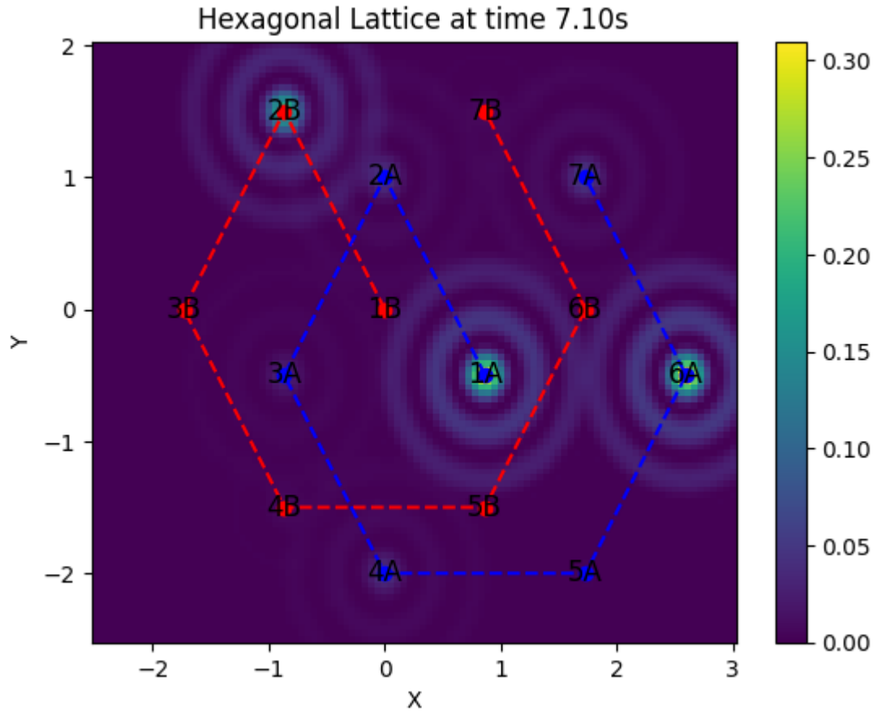

Hexagonal Lattice at time 6.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 6.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

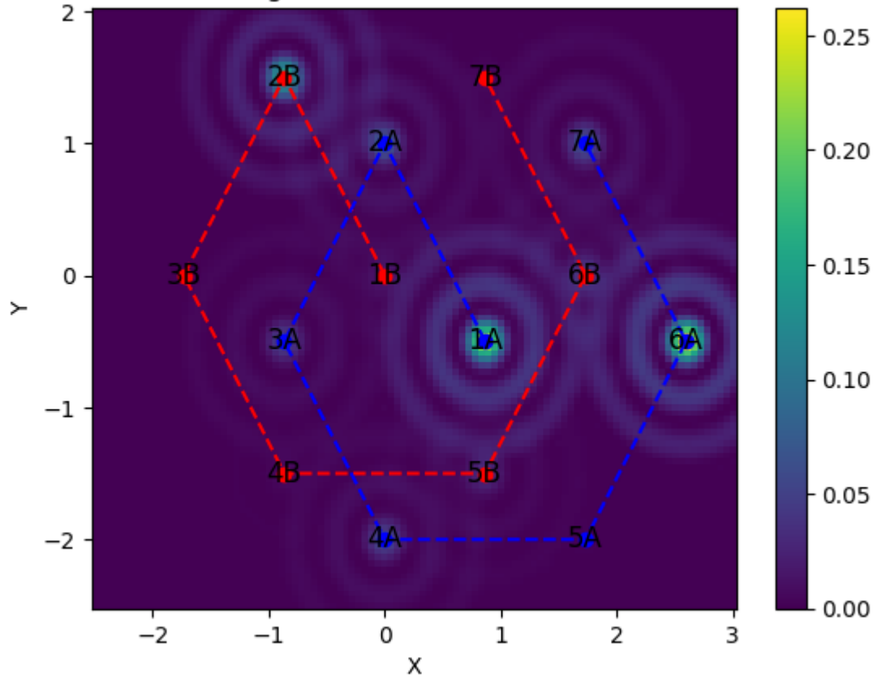

Hexagonal Lattice at time 6.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00

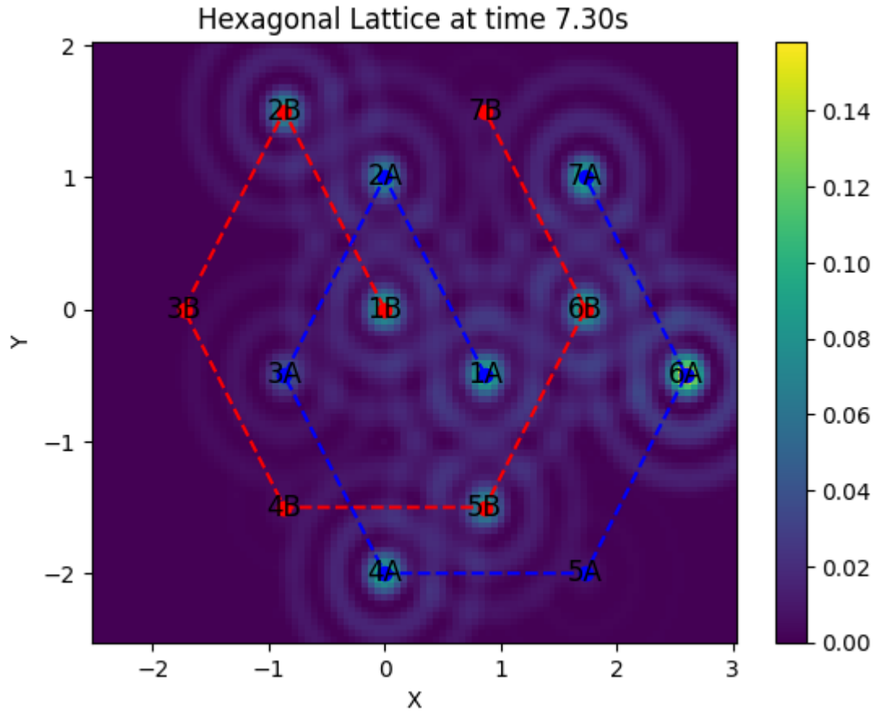

Hexagonal Lattice at time 6.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

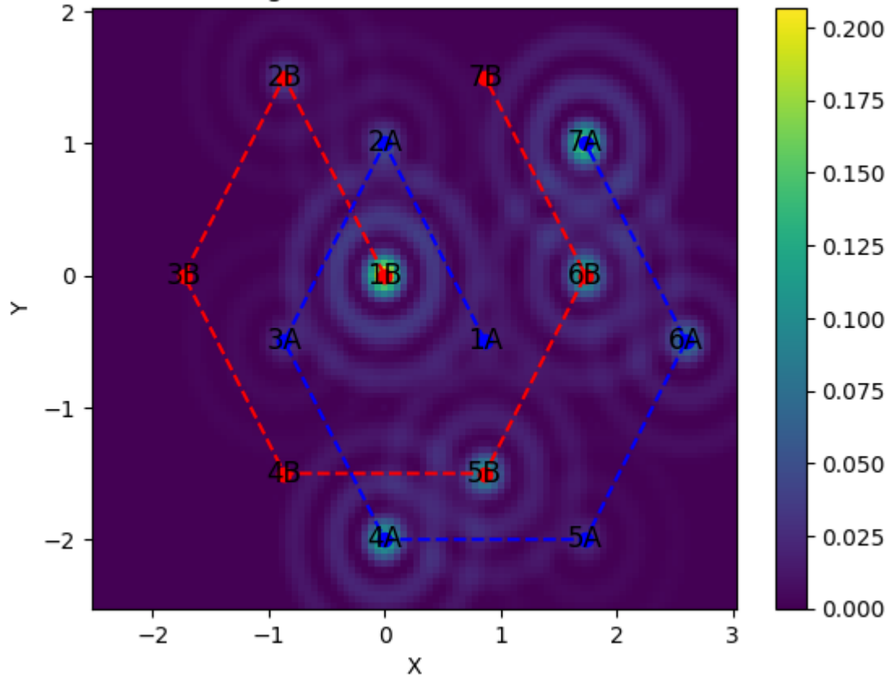

Hexagonal Lattice at time 6.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

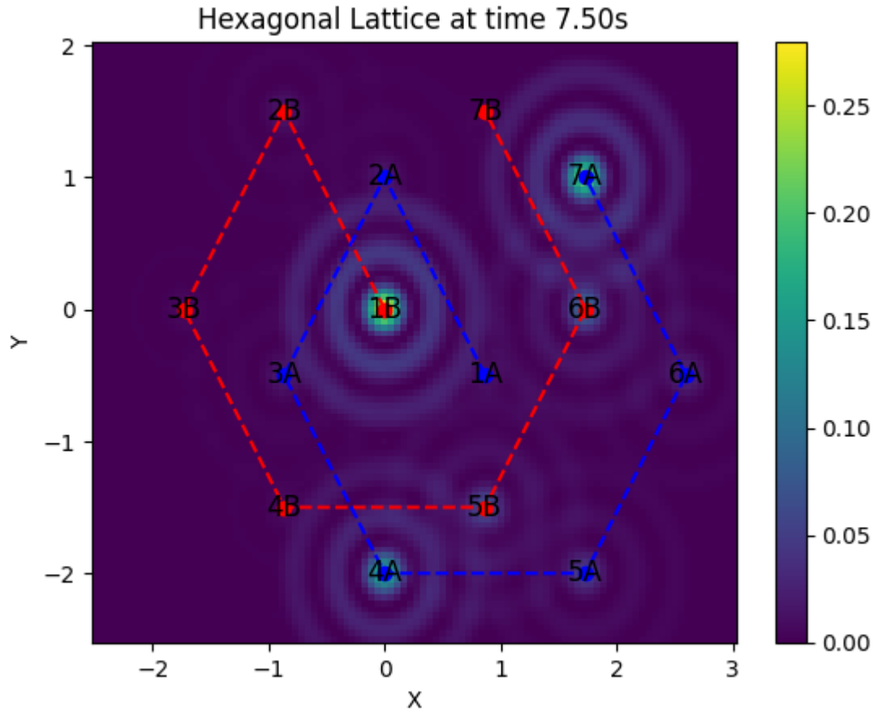

Hexagonal Lattice at time 7.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

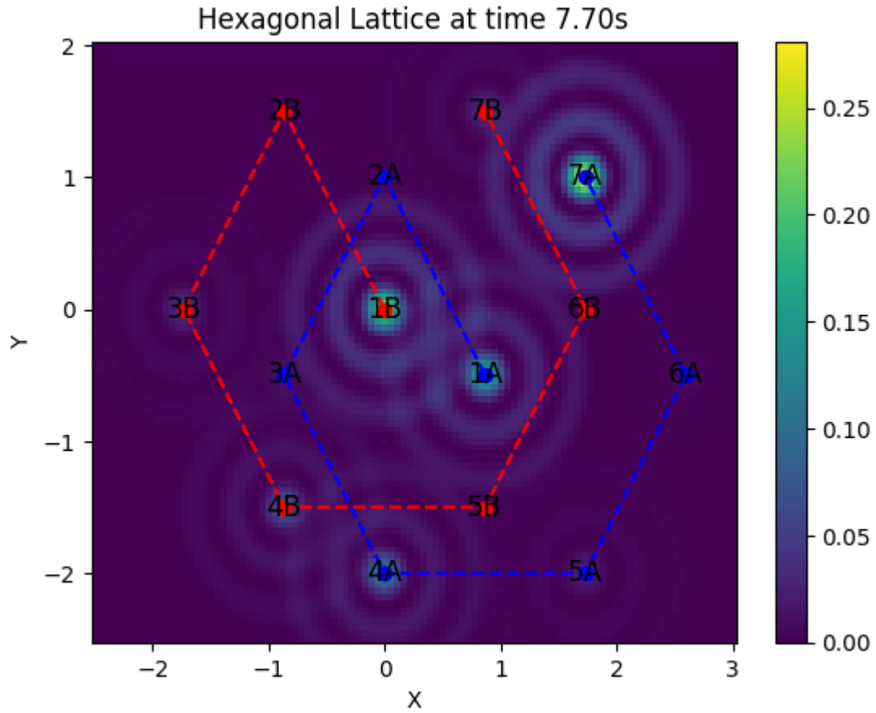

Hexagonal Lattice at time 7.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

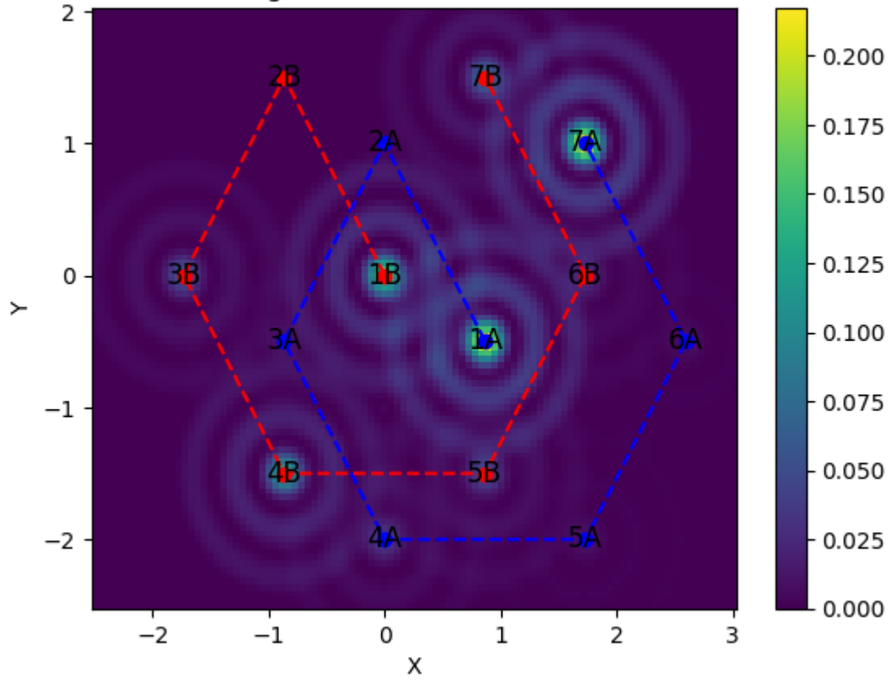

Hexagonal Lattice at time 7.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

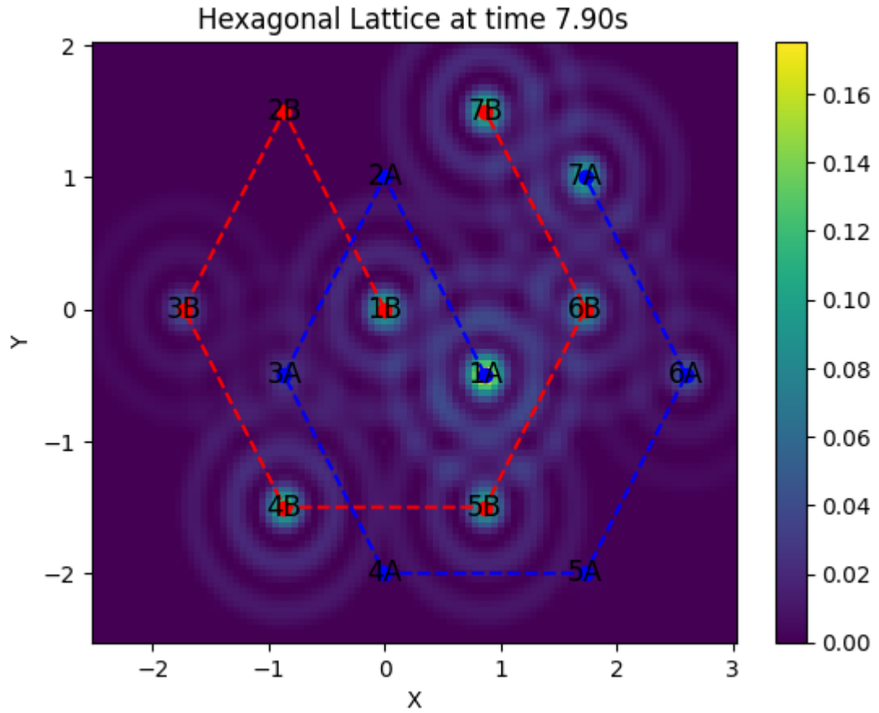

Hexagonal Lattice at time 7.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00

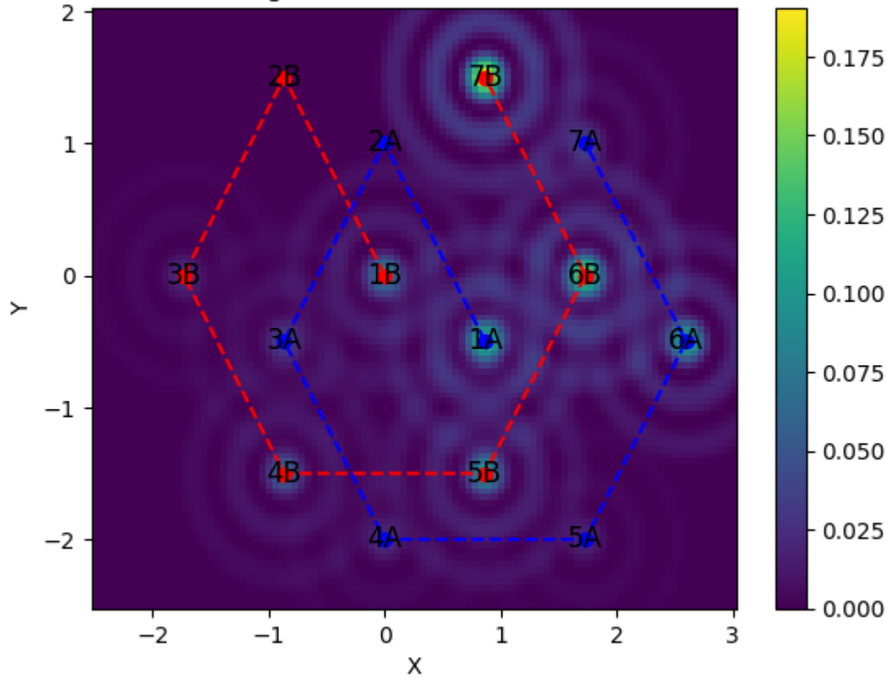

Hexagonal Lattice at time 7.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

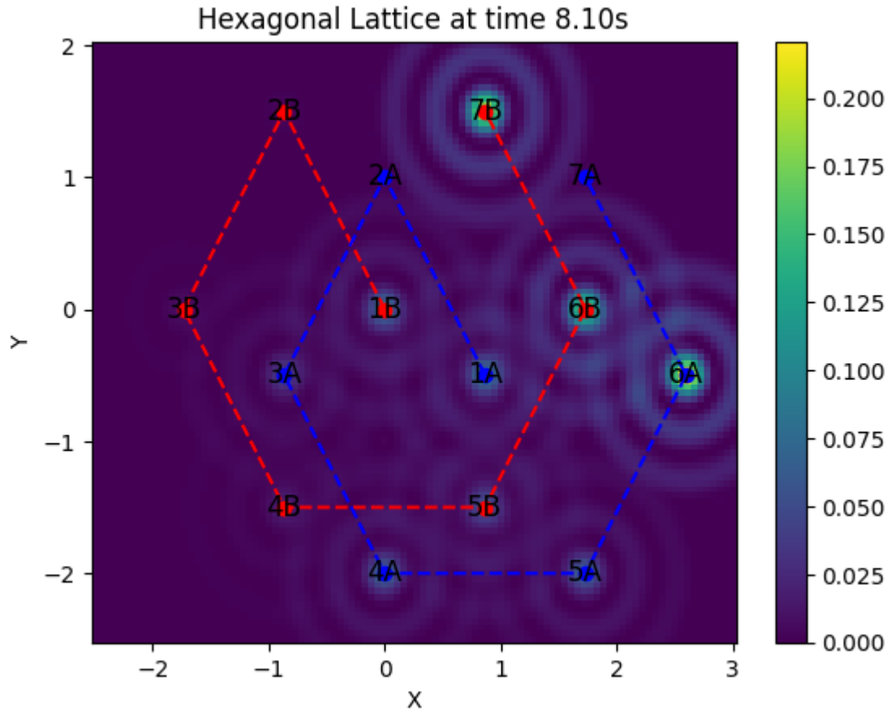

Hexagonal Lattice at time 7.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
Hexagonal Lattice at time 7.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y

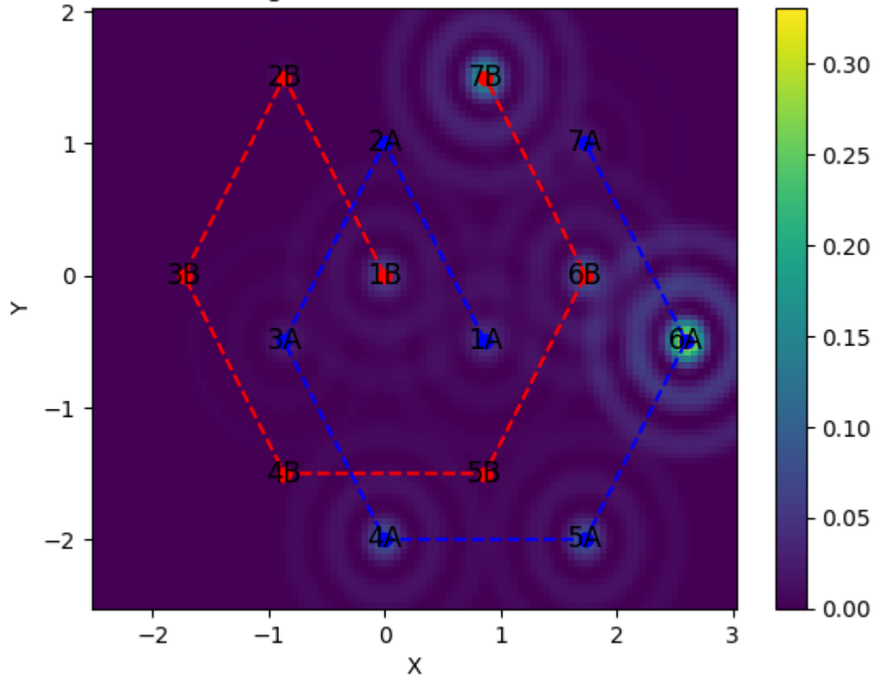

Hexagonal Lattice at time 7.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

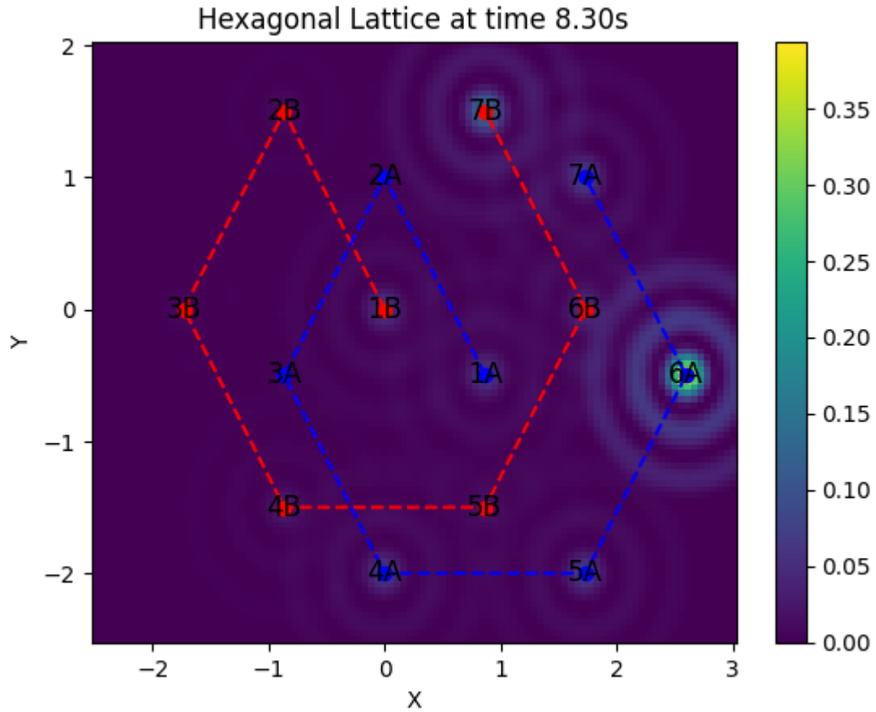

Hexagonal Lattice at time 7.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

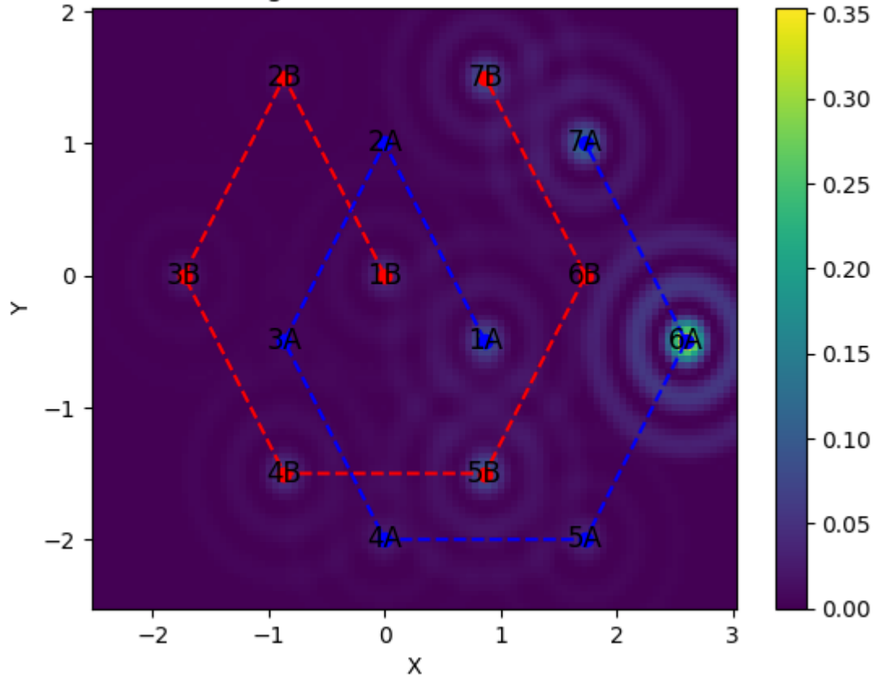

Hexagonal Lattice at time 7.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00

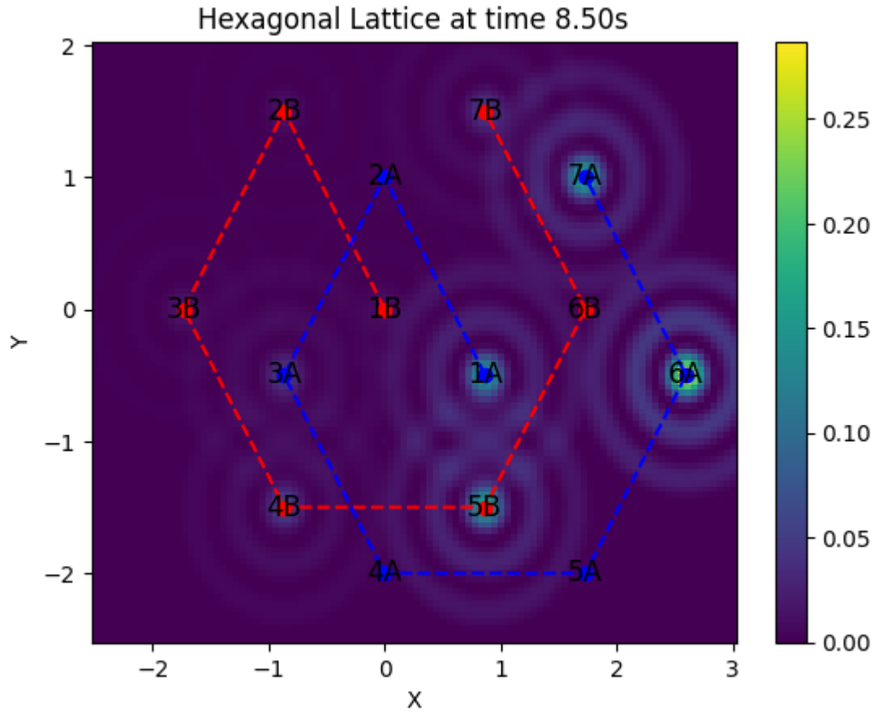

Hexagonal Lattice at time 8.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

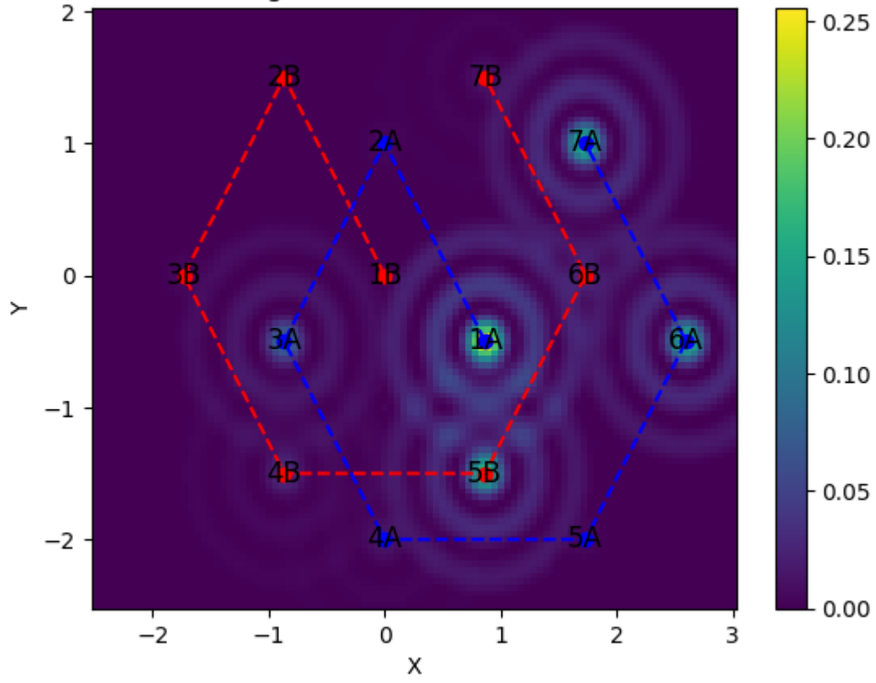

Hexagonal Lattice at time 8.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

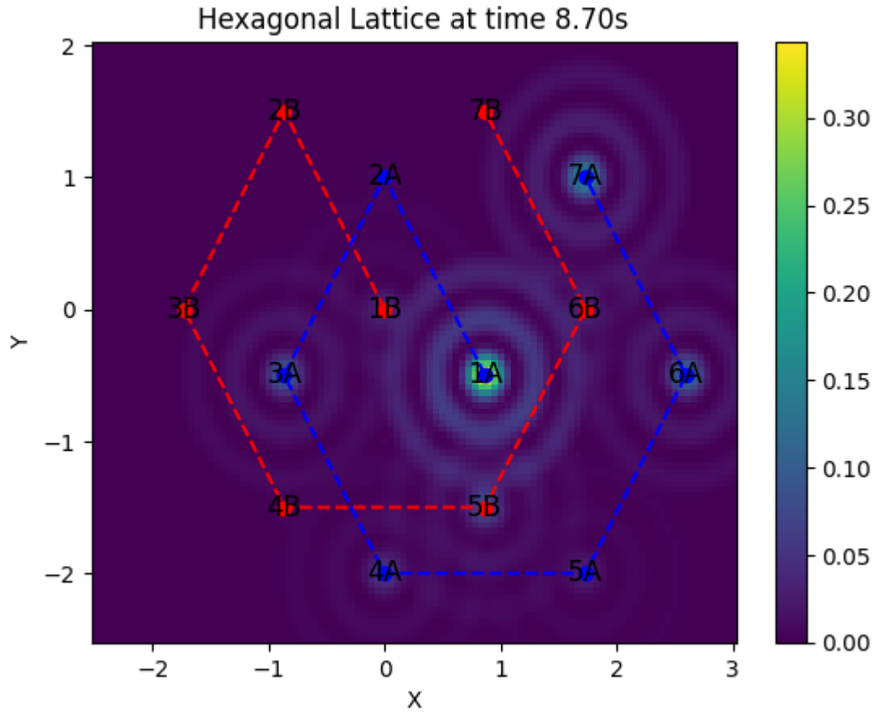

Hexagonal Lattice at time 8.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

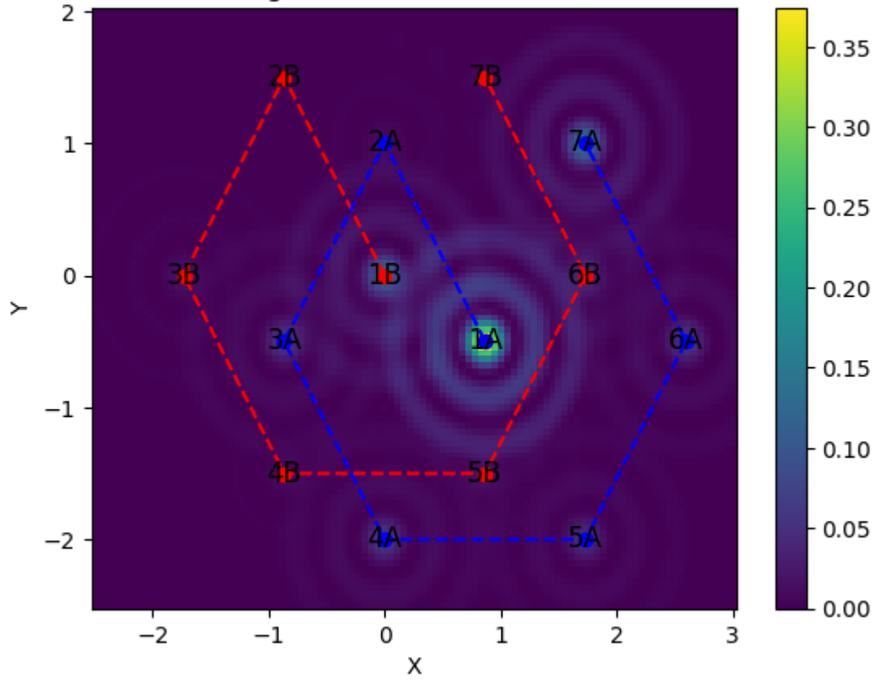

Hexagonal Lattice at time 8.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

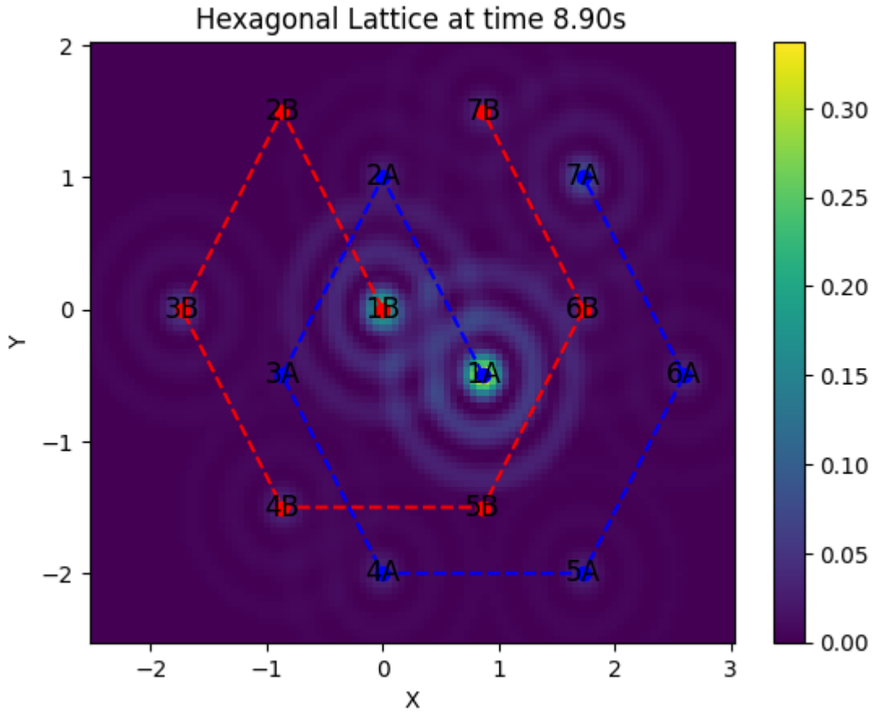

Hexagonal Lattice at time 8.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

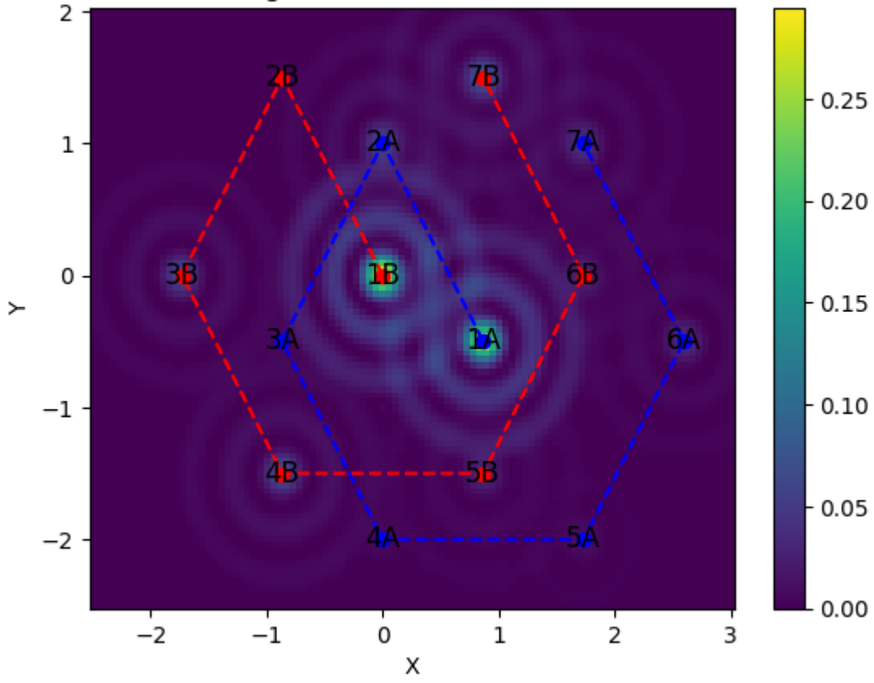

Hexagonal Lattice at time 8.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

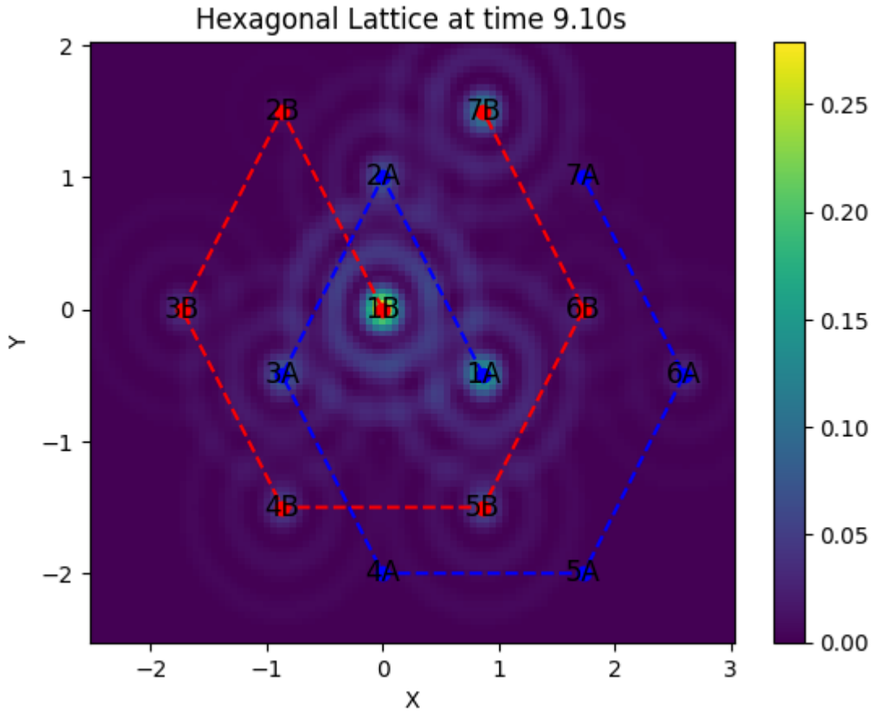

Hexagonal Lattice at time 8.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

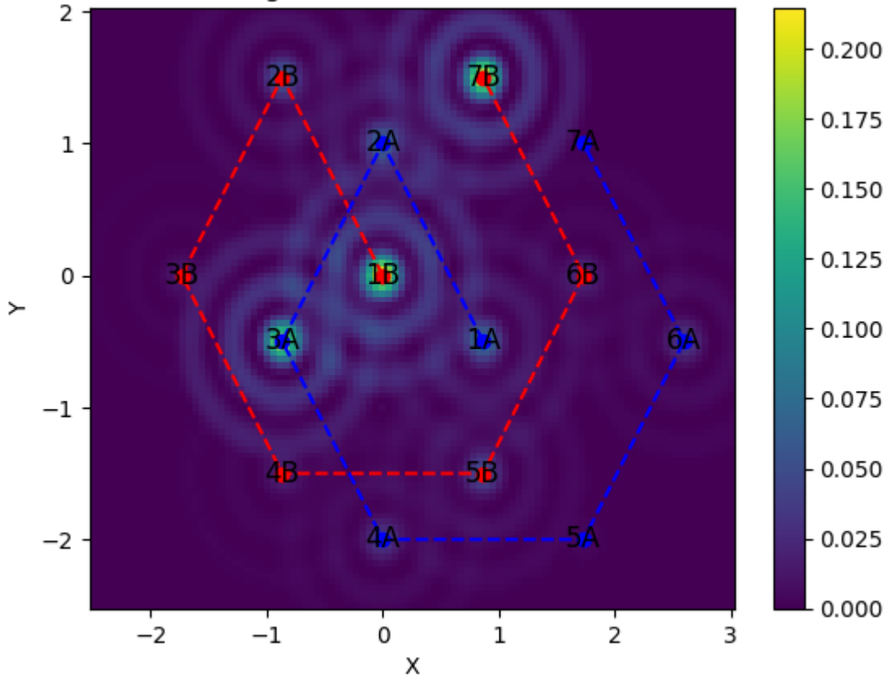

Hexagonal Lattice at time 8.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

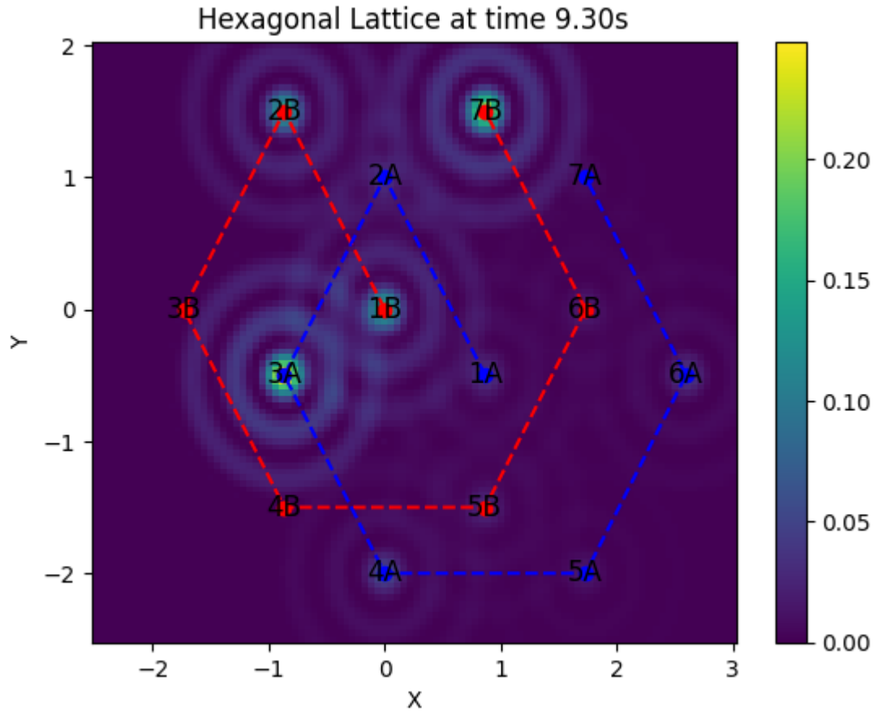

Hexagonal Lattice at time 8.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

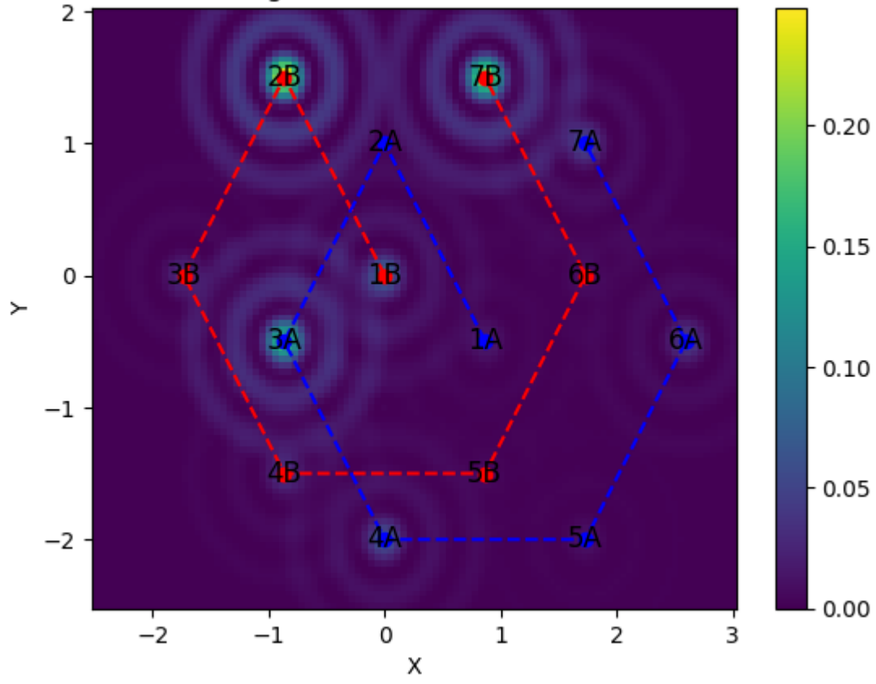

Hexagonal Lattice at time 8.90s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00

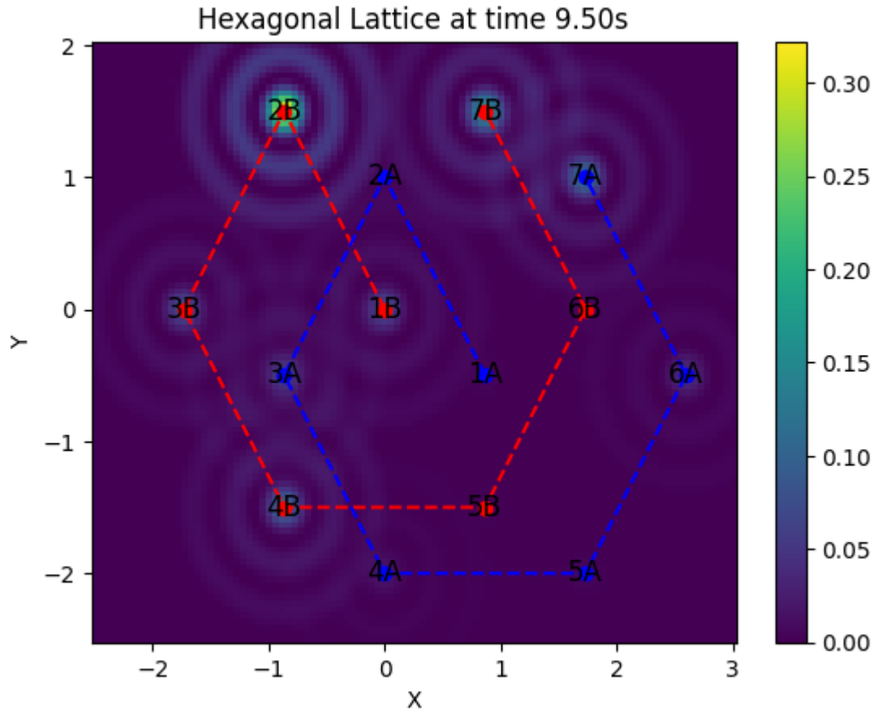

Hexagonal Lattice at time 9.00s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

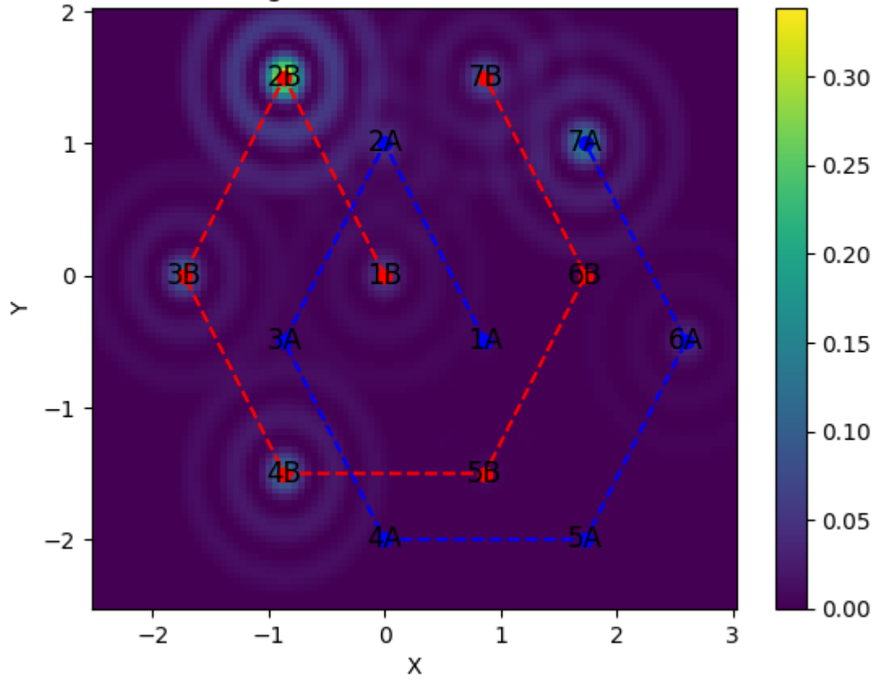

Hexagonal Lattice at time 9.10s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

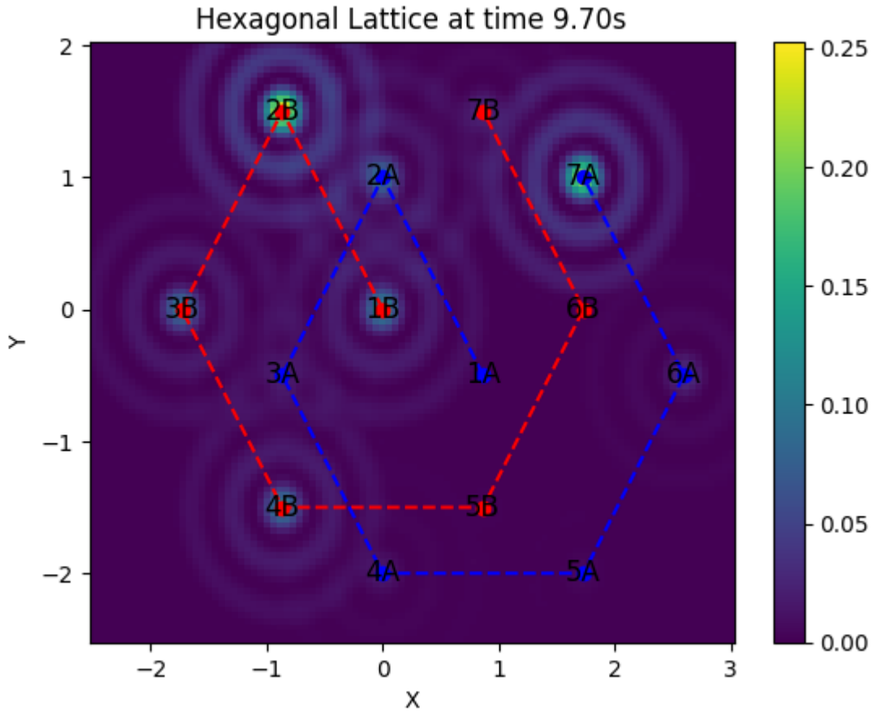

Hexagonal Lattice at time 9.20s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

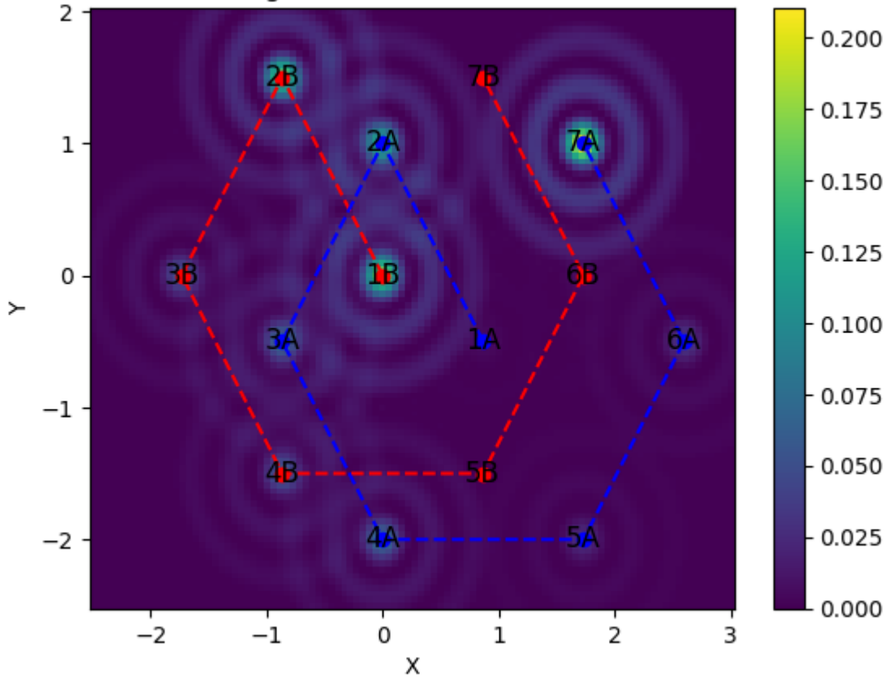

Hexagonal Lattice at time 9.30s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00

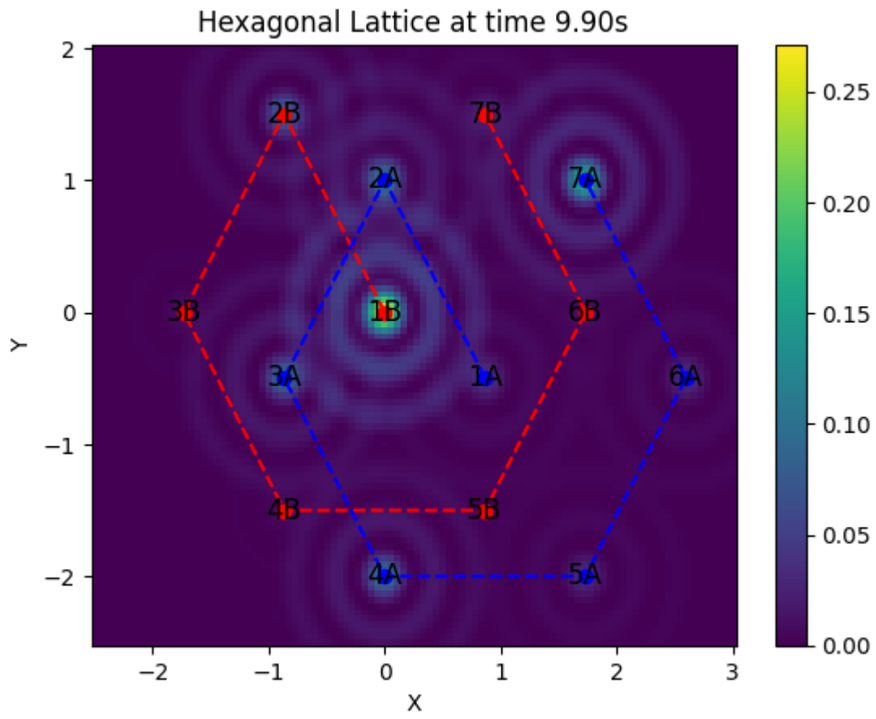

Hexagonal Lattice at time 9.40s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.20
0.15
0.10
0.05
0.00


Hexagonal Lattice at time 9.50s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00


Hexagonal Lattice at time 9.60s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.30
0.25
0.20
0.15
0.10
0.05
0.00


Hexagonal Lattice at time 9.70s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.25
0.20
0.15
0.10
0.05
0.00

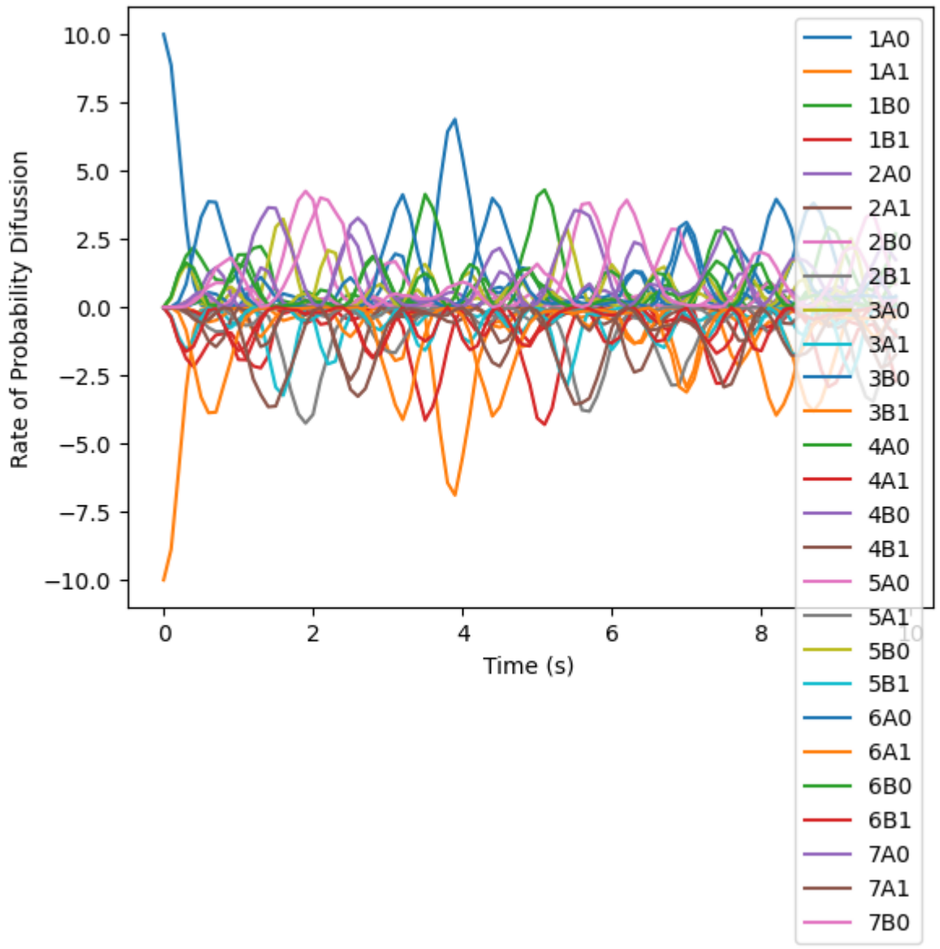

Hexagonal Lattice at time 9.80s
2B
7B
2A
7A
3B
1B
6B
3A
1A
6A
4B
5B
4A
5A
X
Y
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000



Mandatory task 14: Estimate the rate of transition between first neighbors from the results obtained from the quantum simulator. Can you identify differences relative to the results of exercise 1.

Reply:

In [55]:
```
times_noise,states_noise,counts_noise=getStruturedData_seq(data_noise, n_qubits)

times_noise=remove_111_seq(times_noise)
states_noise=remove_111_seq(states_noise)
probs_noise=remove_111_seq(counts_noise/shots)

for i in range(len(basis)-1):
    index_old = np.where(states_noise==i)
    index_new = np.where(states_noise==i+1)
    plt.plot(times_noise[index_old],(probs_noise[index_new]-probs_noise[index_old])/dt,label=basis[i])

plt.xlabel("Time (s)")
plt.ylabel("Rate of Probability Difussion")
plt.legend()
plt.show()
```

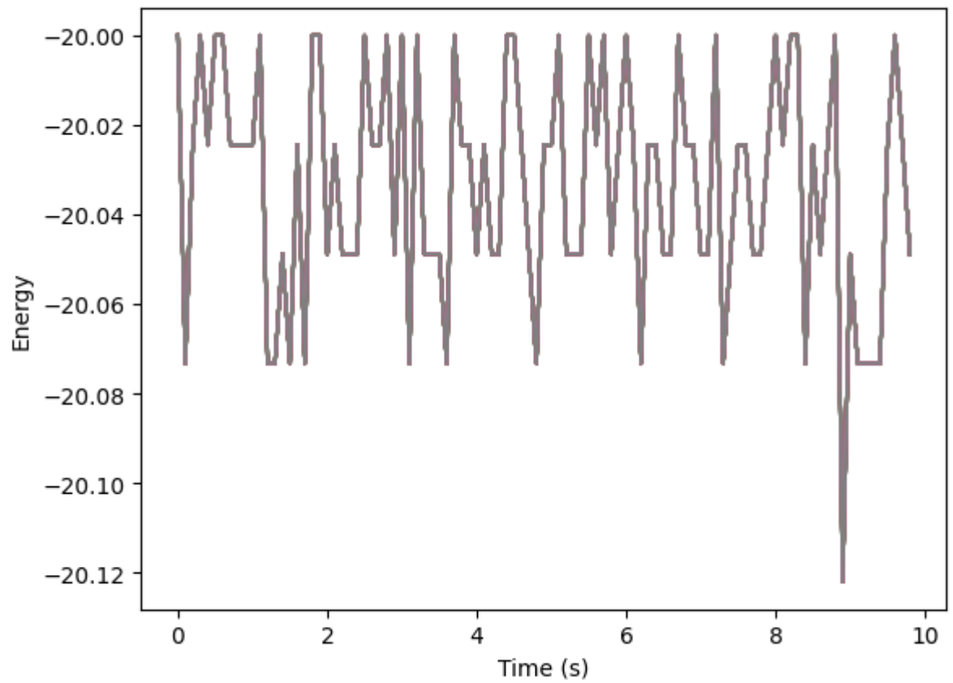


Mandatory task 15 : Provide a graph estimating the time evolution the energy of the state of the particle. Comment your result.

Reply:

In [56]:
```
times_noise,states_noise,counts_noise=getStruturedData_grid(data_noise, n_qubits)

times_noise=remove_111_grid(times_noise)
probs_noise=remove_111_grid(counts_noise/shots)

Energy=[]
for prob in probs_noise:
    E=0
    for i in range(len(prob)):
        if (i % 2) == 0:
            E+=E0*prob[i]
        else:
            E+=E1*prob[i]
    Energy.append(E)
```

In [57]:
```
plt.plot(times_noise,Energy)
plt.xlabel("Time (s)")
plt.ylabel("Energy")
plt.show()
```



Mandatory task 16 :
Are the results obtained from measurements conducted in the simulation sufficient to estimate the time evolution the entropy of the state of the particle. Justify.

Reply:

Yes, we only need to know how the probability changes over time, and apply Shannon's entropy.

Mandatory task 17 :
Estimate the magnitude of the fluctuations in particle position observed in the results of the experiment as a variance in position.

Reply:

Mandatory task 19 :
Discuss the physical effects which determine the variance referenced in the previous task.

Reply:

Mandatory task 20:
Throughout this exercise, several approximations have been introduced, which may impose limitations on the validity of the results obtained from the quantum simulation. These approximations arise from both the description of the physical model and the conversion process into the model running on the quantum simulator. Please, list and discuss these approximations and their impact on the quality of the simulation results, as well as potential improvements.

Reply:

As we introduce the noise model, we are making out physical system open to energy exchanges with the exterior. Even though we get similar results, the particle behaviour in the figure in Exercise 13, shows an erratic behaviour not observed without the implementation of noise.

Task for extra points 5:
Once again, the purpose of this task is to assess your ability to interpret the results of your simulated experiment and extract physical insights regarding the original physical system. As such, there is no predefined answer for this task.

With that in mind, please compare the results obtained with those of the previous exercise. Feel free to adjust the simulation parameters as needed to enhance your analysis.

Reply:

Task for extra points 6:
This is a special task. Run exercise 1 in IBM's quantum computer. Yes, this is the time to actually use a real quantum computer.

Reply:

Task for extra points 7:
Once again, the purpose of this task is to assess your ability to interpret the results of your simulated experiment and extract physical insights regarding the original physical system. As such, there is no predefined answer for this task.

With that in mind, please compare the results obtained with those of the previous exercises and the results obtained in the quantum computer. Feel free to adjust the simulation parameters as needed to enhance your analysis.

Reply:

Task for extra points 8:
Discuss how to improve the realism of the simulations implemented in this exam.

Reply:

We could have assumed more cells, it would be difficult for the computer, and assuming a interatomic potential.

The end!

In [ ]: