

Solving the Two-dimensional Schrödinger Equation Numerically

Lars Bogner

<https://github.uio.no/larsbog/FYS4150>

(Dated: December 9, 2025)

We present a numerical framework for solving the two-dimensional, time-dependent Schrödinger equation. Our method uses the unconditionally stable, second-order accurate in both space and time Crank-Nicolson finite difference scheme. We implement the discretized system efficiently in C++ using the Armadillo linear algebra library to handle sparse matrix operations. Our framework is highly configurable and allows us to simulate various potentials, with a specific focus on modeling single- and multiple-slit experiments. Our results demonstrate excellent numerical accuracy, maintaining probability conservation to within an order of 10^{-15} . The simulations successfully reproduce key quantum mechanical phenomena, including wave packet diffraction and the characteristic interference patterns from double and triple slits. Additionally, we demonstrate the approach's flexibility by computing the probability distribution of particle detection on a virtual screen. This work establishes a robust, extensible foundation for computational exploration of two-dimensional time-dependent quantum dynamics.

CONTENTS

I. Introduction	1
II. Methods	1
A. Discretization of the Schrödinger equation	1
B. Initial conditions and potential	2
C. Implementation details	2
Use of Artificial Intelligence	3
III. Results and discussion	3
IV. Conclusion	4
A. Evolution of wave function in presence of a double slit	5
References	5

I. INTRODUCTION

The Schrödinger equation, introduced in 1925, is a fundamental equation in quantum mechanics that describes how the quantum state of a physical system changes over time. It is a key result of wave mechanics, one of the two main formulations of quantum mechanics (the other being matrix mechanics). The equation is named after Erwin Schrödinger, who developed it and won the Nobel Prize in Physics in 1933 for his work.

The time-dependent Schrödinger equation is given by:

$$i\hbar \frac{\partial}{\partial t} \psi = \hat{H} \psi, \quad (1)$$

where i is the imaginary unit, $\hbar$ is the reduced Planck constant, and ψ is the wave function of the quantum system. The Hamiltonian operator, $\hat{H}$, represents the total energy of the system, i.e., the sum of kinetic and potential energies.

In [section II](#), we describe the numerical methods used to solve the two-dimensional Schrödinger equation. In [section III](#), we present and discuss the results obtained from our numerical simulations. Finally, in [section IV](#), we summarize our findings and discuss potential future work.

II. METHODS

A. Discretization of the Schrödinger equation

All numerical methods for solving differential equations rely on the discretization of continuous variables. In this project, we will discretize both space and time. We will begin with the two-dimensional time-dependent Schrödinger equation:

$$i\hbar \frac{\partial}{\partial t} \psi(x, y, t) = -\frac{\hbar^2}{2m} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \psi(x, y, t) + V(x, y) \psi(x, y, t). \quad (2)$$

We define a grid with spacing h in both the x and y directions to discretize space. The grid points are given by $x_i = ih$ and $y_j = jh$, where i and j are integers. The wave function at these grid points is denoted as $\psi_{i,j}(t) = \psi(x_i, y_j, t)$. We also discretize time with a time step of size Δt , such that $t_n = n\Delta t$, where n is an integer. The wave function at time t_n is denoted as $\psi_{i,j}^n = \psi_{i,j}(t_n)$. For simplicity, we set both $\hbar = 1 \text{ s}^{-1}$ and $m = 1$. This yields the dimensionless form of the Schrödinger equation:

$$i \frac{\partial}{\partial t} u(x, y, t) = -\frac{\partial^2}{\partial x^2} u(x, y, t) - \frac{\partial^2}{\partial y^2} u(x, y, t) + v(x, y) u(x, y, t). \quad (3)$$

In this form, we substitute u for ψ and v for V to eliminate constants. We can then reobtain the physical probability density, $P = \psi^* \psi$, by appropriately normalizing

the wave function:

$$P(x, y, t) = \frac{u^*(x, y, t)u(x, y, t)}{\int \int u^*(x, y, t)u(x, y, t) dx dy}. \quad (4)$$

In practice, the integral in the denominator is equal to the sum of all grid points multiplied by the area element h^2 .

Using the Crank-Nicolson scheme, we can approximate the time evolution of the wave function. The Crank-Nicolson method is a second-order accurate, unconditionally stable implicit finite difference method in both space and time. The update rule for the wave function using the Crank-Nicolson scheme is given by the following formula [1]:

$$\left(I + \frac{i\Delta t}{2}H\right)u^{n+1} = \left(I - \frac{i\Delta t}{2}H\right)u^n, \quad (5)$$

where I is the identity matrix, H is the discretized Hamiltonian operator, and u^n and u^{n+1} are the wave functions at time steps n and $n+1$, respectively. The Hamiltonian operator H can be discretized using finite difference approximations for the second derivatives:

$$H_{i,j} = -\frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) + v_{i,j}u_{i,j}. \quad (6)$$

By defining the constant as $r \equiv \frac{i\Delta t}{2h^2}$, we can rewrite the update rule as follows:

$$\begin{aligned} u_{i,j}^{n+1} + r(u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1} - 4u_{i,j}^{n+1}) \\ = u_{i,j}^n - r(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n) \\ - i\Delta t v_{i,j} \frac{u_{i,j}^{n+1} + u_{i,j}^n}{2}. \end{aligned} \quad (7)$$

We can rearrange this equation to form a linear system that can be solved for u^{n+1} at each time step. We define the index mapping $k = i + jM$, where M is the number of grid points in one dimension and $M = \frac{1}{h} + 1$. This allows us to rewrite the two-dimensional grid as a one-dimensional array [2]. The resulting linear system can be expressed in matrix form as follows:

$$Au^{n+1} = Bu^n, \quad (8)$$

where A and B are matrices that depend on the discretization parameters and the potential v . Only the internal points of the problem are contained in A , B , u^n . Boundary conditions are applied separately. For this project, we will use Dirichlet boundary conditions and set the wave function to zero at the grid boundaries. Matrix A is tridiagonal with additional diagonals due to the two-dimensional nature of the problem. The matrix B has a similar structure. To solve for u^{n+1} , first calculate $b^n = Bu^n$, then solve the linear system $Au^{n+1} = b^n$ using an appropriate numerical method, such as LU decomposition or iterative solvers.

B. Initial conditions and potential

To simulate the time evolution of the wave function, we must specify the initial conditions and the potential, $v(x, y)$. A common choice for the initial wave function is a Gaussian wave packet, defined as follows [2]:

$$u(x, y, 0) = \exp\left(-\frac{(x - x_C)^2}{2\sigma_x^2} - \frac{(y - y_C)^2}{2\sigma_y^2}\right) \cdot \exp(i(p_x x + p_y y)), \quad (9)$$

where (x_C, y_C) is the center of the wave packet and where σ_x, σ_y are the widths of the packet in the x and y directions, respectively, and where the initial momenta in the respective directions are represented by (p_x, p_y) . The potential, $v(x, y)$, can be chosen based on the physical system being modeled. For example, a double-slit-like potential can be used. To implement slits in the potential grid, we use the algorithm outlined in [algorithm 1](#). This algorithm modifies the potential grid, V , by adding walls of a specified thickness and position while leaving apertures for the slits. An initialization function finalizes the potential setup by calculating the matrices A and B based on the modified potential.

Algorithm 1 Add Slits to Potential Grid

Require: $h, M, V, V_{\text{wall}}, \text{wall_thickness}, \text{wall_position}, \text{slit_aperture}, \text{slit_separation}, \text{num_slits}$

- 1: $x_{\text{start}} \leftarrow \left\lfloor \frac{\text{wall_position} - 0.5 \text{ wall_thickness}}{h} \right\rfloor$
- 2: $x_{\text{end}} \leftarrow \left\lfloor \frac{\text{wall_position} + 0.5 \text{ wall_thickness}}{h} \right\rfloor$
- 3: $y_c \leftarrow M/2$
- 4: $\text{slit_half} \leftarrow \left\lfloor \frac{0.5 \text{ slit_aperture}}{h} \right\rfloor$
- 5: $\text{pitch} \leftarrow \text{slit_aperture} + \text{slit_separation}$
- 6: $\text{pitch}_{\text{idx}} \leftarrow \text{pitch}/h$
- 7: **for** $i = x_{\text{start}}$ **to** x_{end} **do**
- 8: **for** $j = 0$ **to** $M - 1$ **do**
- 9: $\text{in_slit} \leftarrow \text{false}$
- 10: **for** $s = 0$ **to** $\text{num_slits} - 1$ **do**
- 11: $\Delta \leftarrow \left(s - \frac{\text{num_slits} - 1}{2}\right) \text{pitch}_{\text{idx}}$
- 12: $c_s \leftarrow \lfloor y_c + \Delta + 0.5 \rfloor$ ▷ rounded center index
- 13: **if** $j \in [c_s - \text{slit_half}, c_s + \text{slit_half}]$ **then**
- 14: $\text{in_slit} \leftarrow \text{true}$
- 15: **break**
- 16: **if not** in_slit **then**
- 17: $V[i, j] \leftarrow V_{\text{wall}}$
- 18: **INITIALIZE_POTENTIAL**

C. Implementation details

To optimize performance, the numerical methods were implemented in C++, compiled with optimization flags, and compiled using the g++ compiler. Version g++ (GCC) 15.2.1 20251111 (Red Hat 15.2.1-4) was used for compilation. The Armadillo library [3, 4] was used for

all linear algebra operations because it provides efficient implementations of matrix and vector operations. Matrices A and B are constructed as sparse [4], complex matrices to save memory and improve computational efficiency. We solved the linear system $Au^{n+1} = b^n$ using the `spsolve` function from the Armadillo library. This function is optimized for sparse matrices and uses an implementation provided by SuperLU [5].

To allow for easy experimentation with different parameters, the program reads a configuration file in TOML format during runtime. The configuration file specifies parameters in four main categories: Discretization parameters: h , Δt , and total simulation time T ; Initial wave packet parameters (center position, width, and momentum); Potential parameters (wall height, wall thickness, slit aperture, slit separation, and number of slits); Output parameters (output file name). The program uses Armadillo’s built-in functions to parse the wave function results at each time step and write them to a file in `arma_ascii` format.

All analyses and visualizations of the results are performed in Python using the NumPy [6] and Matplotlib [7] libraries. The probability density is computed from the wave function data, and various plots are generated to illustrate the time evolution of the wave packet and the effects of the potential. A self-developed parser for the Armadillo ASCII format reads the output files generated by the C++ program, as described in [algorithm 2](#).

Algorithm 2 Load Armadillo ASCII Complex Cube

```

1: procedure LOADCUBE(path, slices_first)
2:   txt  $\leftarrow$  read file at path
3:   lines  $\leftarrow$  split(txt)
4:   header  $\leftarrow$  first two lines
5:   ( $n_{\text{rows}}, n_{\text{cols}}, n_{\text{slices}}$ )  $\leftarrow$  parse(header[1])
6:   rest  $\leftarrow$  remaining lines
7:   for each line in rest do
8:     remove all “(” and “)”
9:     apply regex

       ([+-]?[.:eE+-]*) , ([+-]?[.:eE+-]*)

       which captures two floating-point values  $r$  and  $i$ , possibly
       with signs or scientific notation
10:    replace with

           $1+$2j

       yielding Python-style complex numbers  $r + ij$ 
11:    append transformed line to fixed
12:    arr2D  $\leftarrow$  NUMPY.LOADTXT(fixed)
13:    arr3D  $\leftarrow$  reshape(arr2D, ( $n_{\text{slices}}, n_{\text{cols}}, n_{\text{rows}}$ ))
14:    if slices_first then
15:      return arr3D
16:    else
17:      return transpose(arr3D)

```

Use of Artificial Intelligence

In this project, artificial intelligence in the form of large language models (LLMs) was employed to assist with various stages of the research and development process. The LLMs generated initial drafts of code snippets that the researcher reviewed and refined for correctness and efficiency. GitHub Copilot, an LLM, was used for this task. LLMs were also used to edit the report’s language by providing grammar and style suggestions. For this task, the DeepL Write service by DeepL GmbH was used.

III. RESULTS AND DISCUSSION

We evaluate the accuracy of the resulting solutions by assessing the conservation of probability in the system with and without a double slit, i.e., with $V_{\text{wall}} = 0$ and in the presence of such a potential. We simulate the systems for a total time of 0.008 s in steps of $\Delta t = 2.5 \times 10^{-5}$ s. Spatial discretization is carried out using a step size of $h = 0.005$. In the setup with a non-zero potential wall, the initial wave packet is centered at $x = 0.25$ and $y = 0.5$, with widths of $\sigma_x = 0.05$ and $\sigma_y = 0.10$, respectively. Its momenta are $p_x = 200.0$ and $p_y = 0.0$, resulting in strongly directed motion along x . A double-slit barrier with a thickness of 0.02 is placed at $x = 0.5$. Each slit has an aperture of 0.05 and is separated by 0.05. The wall potential is effectively infinite at $V_{\text{wall}} = 1.0 \times 10^{10}$. In the case of a constant potential, the wave packet has widths of $\sigma = 0.05$ in both x and y . Otherwise, the setup is identical. The resulting wave functions are evaluated for loss of probability.

$$\delta = 1 - \iint u^*(x, y, t)u(x, y, t)dx dy, \quad (10)$$

where $\iint u^*(x, y, 0)u(x, y, 0)dx dy = 1$ was fixed using normalization. In a perfect physical system, this quantity is conserved; however, due to limited numerical accuracy, especially in the derivatives and solving of the linear system, it is not conserved. Therefore, it is a good indication of the algorithm’s potential accuracy. [Figure 1](#) shows the resulting analysis. The maximum deviation is about 2×10^{-15} , demonstrating performance that is only one order of magnitude worse than floating-point precision. Notably, the accuracy of the system without a double slit is lower than the corresponding values with the perturbation. Additionally, the simulation’s dependence on potential is evident, precluding any general attestation of performance for arbitrary potentials. Nevertheless, it can be assumed that the simulation will produce very accurate results for many potentials, attesting to the numerical accuracy of the results.

In addition to the quantitative evaluation of performance, we conduct a qualitative analysis of the system’s evolution. To correctly assess the evolution of the wave packet, we perform a double slit experiment on a system for a total duration of 0.002 s. Three time points are

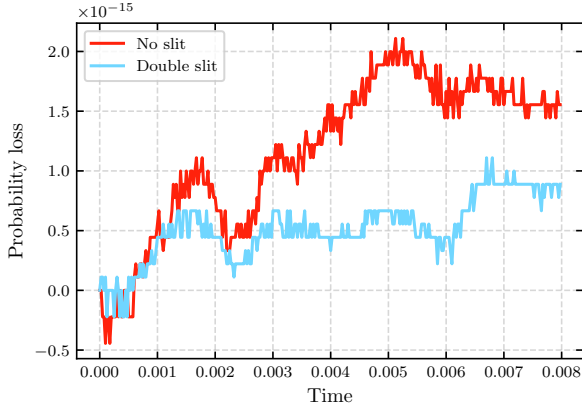


Figure 1. Loss of total probability δ in dependence on the simulation duration for two systems of different potential.

sampled to demonstrate the wave packet’s interference with itself behind the double slit. The results of this analysis are presented in [figures 3 to 5](#) in [Appendix appendix A](#). We observe the expected qualitative behavior with respect to interference and the correct diffusion of the wave function. Therefore, we can conclude that the simulation framework behaves correctly. To evaluate the evolution of the probability distribution, we produced animations of the entire runtime for each simulation. These animations can be accessed via the repository of this paper, though they are not essential to understanding the paper.

Lastly, we demonstrate the flexibility of our approach to simulating the numerical evolution of such a system by simulating the probability of detecting a particle on a screen inserted into the system at a specific time. We computed the probability density function (PDF) for a screen position of $y = 0.8$ at $T = 0.002$ s for three systems: a single slit, a double slit, and a triple slit. The detection PDF is given by the slice

$$P(x) = \frac{u^*(x, 0.8, 0.002 \text{ s})u(x, 0.8, 0.002 \text{ s})}{\int dx u^*(x, 0.8, 0.002 \text{ s})u(x, 0.8, 0.002 \text{ s})}, \quad (11)$$

where the integral is evaluated as a sum over the total time slice. The resulting PDF is displayed in [figure 2](#). Once again, the interference of the wave packet with itself is clearly visible, leading to the expected result. More importantly, this shows how such a numerical simulation can be used to compute arbitrary metrics because the entire numerical solution of the wave function is saved to a file that can be repurposed in multiple ways.

IV. CONCLUSION

This project successfully developed and implemented a numerical framework for solving the two-dimensional, time-dependent Schrödinger equation. Using the Crank–Nicolson finite difference scheme, we were able

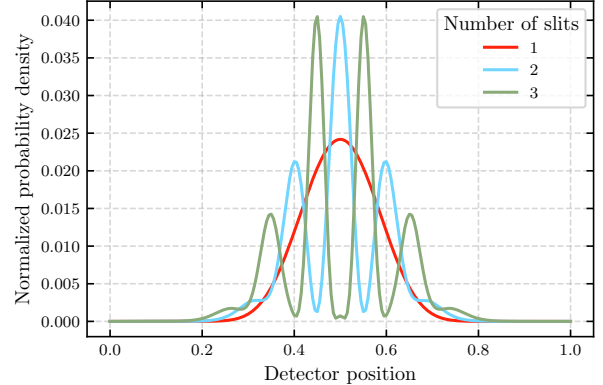


Figure 2. PDF of detecting the wave packet at a certain position on a screen inserted at $y = 0.8$ and $T = 0.002$ s.

to simulate quantum wave packet dynamics stably and accurately. The implicit method enabled comparatively large time steps to be taken while maintaining second-order accuracy in both space and time.

The C++ implementation leveraged the Armadillo library for efficient sparse linear algebra operations, enabling the simulation of systems with high spatial resolution. A key strength of the framework is its configurability via TOML input files, which allows for straightforward experimentation with various initial conditions, potential landscapes and discretisation parameters, eliminating the need for recompilation.

Our results demonstrate the numerical robustness of the approach. The total probability was conserved to within an order of 10^{-15} over the simulation duration, indicating excellent numerical stability. The simulations correctly reproduced fundamental quantum mechanical phenomena, including the diffraction and interference of a wave packet passing through single, double, and triple slit potentials. The characteristic interference patterns observed on a virtual detector screen qualitatively match theoretical expectations, validating the physical correctness of the implementation.

The framework’s output of the complete wave function at each time step allows for maximum flexibility in post-processing. As demonstrated, this enables the computation of arbitrary observables, such as probability density slices at any spatial location, extending beyond the simple screen detection example.

Several extensions are promising for future work. For example, the potential function could be generalised to model more complex systems, such as quantum dots, wells or tunnelling barriers. Implementing absorbing boundary conditions, such as complex absorbing potentials (CAP) or perfectly matched layers (PML), would enable the simulation of scattering problems and open systems without reflections. Furthermore, adapting the code to leverage GPU acceleration or parallel computing techniques would enable the simulation of larger domains

or three-dimensional systems, significantly expanding the range of physically interesting problems that can be explored.

In summary, this project provides a reliable and extensible foundation for the numerical investigation of time-dependent quantum mechanical phenomena in two dimensions, bridging the gap between theoretical concepts and computational experimentation.

Appendix A: Evolution of wave function in presence of a double slit

The simulation uses a grid spacing $h = 0.005$ and a time step of $\Delta t = 2.5 \times 10^{-5}$ s. The double slit has a thickness of 0.02, an aperture of 0.05, a separation of 0.05, and a barrier potential $V_{\text{wall}} = 1.0 \times 10^{10}$. The wave packet is initially broader in the y -direction to enhance

the visibility of interference effects. Each figure displays three subplots: the probability density in 2D, and the real and imaginary parts of the wave function u , illustrating the full complex evolution of the quantum state. The time evolution of the wave function in the double-slit setup is illustrated in figures 3 to 5. Initially, as shown in figure 3, the wave packet is localized before the barrier, with its probability distribution concentrated around the initial position and negligible interference structure. By the intermediate time in figure 4, the wave packet has reached the slits, and partial transmission has produced oscillations in the real and imaginary components, while faint interference fringes start forming in the probability distribution. Finally, after the wave has passed the slits, figure 5 shows well-developed interference fringes in $|u|^2$, and alternating positive and negative regions in the real and imaginary parts, reflecting the coherent superposition of the two transmitted components.

-
- [1] Jin Liu and Yifei Hao. Crank–Nicolson method for solving uncertain heat equation. *Soft Computing*, 26(3):937–945, February 2022.
 - [2] Anders Kvellestad. Project 5 — FYS3150/FYS4150 course material. <https://anderkve.github.io/FYS3150/book/projects/project5.html>.
 - [3] Conrad Sanderson and Ryan Curtin. Armadillo: An Efficient Framework for Numerical Linear Algebra. In *2025 17th International Conference on Computer and Automation Engineering (ICCAE)*, pages 303–307, March 2025.
 - [4] Conrad Sanderson and Ryan Curtin. Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation. *Mathematical and Computational Applications*, 24(3):70, July 2019.
 - [5] James W. Demmel, John R. Gilbert, and Xiaoye S. Li. SuperLU users’ guide. Technical Report LBNL–44289, 751785, November 1999.
 - [6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
 - [7] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

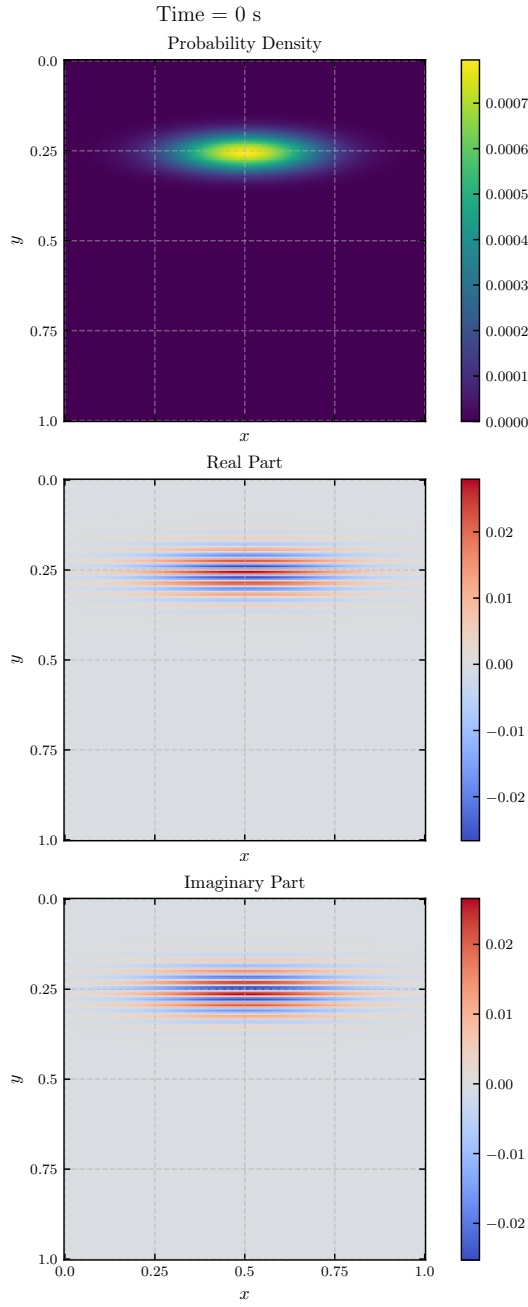


Figure 3. Initial state of the double-slit simulation at $t = 0.000$ s. The three panels show (from top to bottom) the probability distribution $|u|^2$, the real part of the wave function $\text{Re}(u)$, and the imaginary part $\text{Im}(u)$. The wave packet is initially centered at $(x_c, y_c) = (0.25, 0.5)$ with widths $\sigma_x = 0.05$ and $\sigma_y = 0.20$, moving primarily in the x -direction with momentum $p_x = 200.0$.

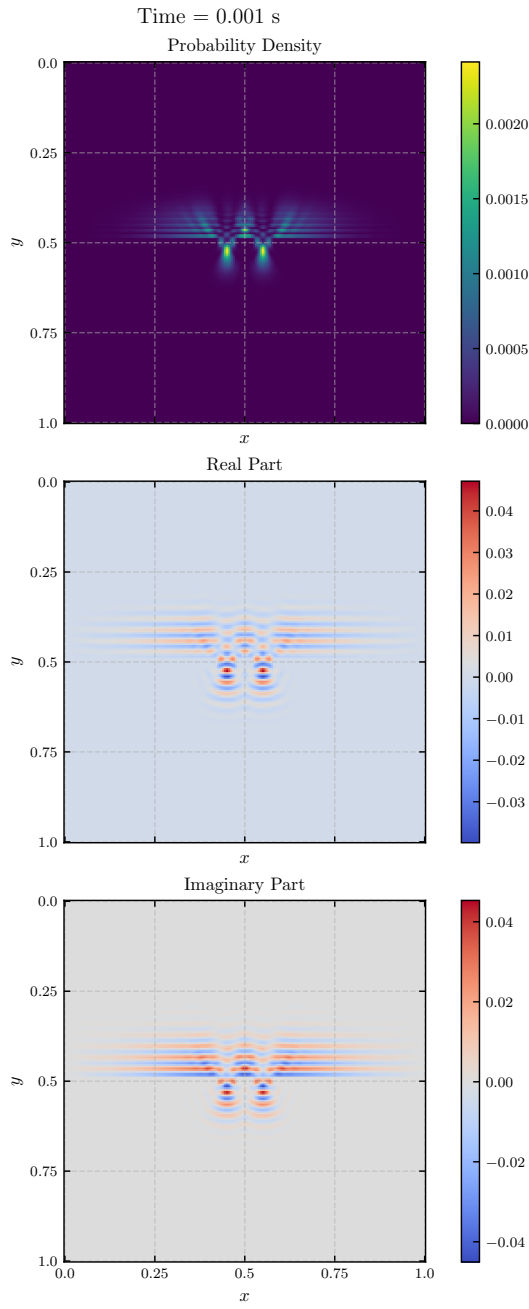


Figure 4. Evolution of the wave function at $t = 0.001$ s. The wave packet approaches the double slit located at $x = 0.5$. The interference pattern begins to emerge in the probability distribution $|u|^2$, while the real and imaginary parts of the wave function start to show oscillatory structure due to partial transmission through the slits.

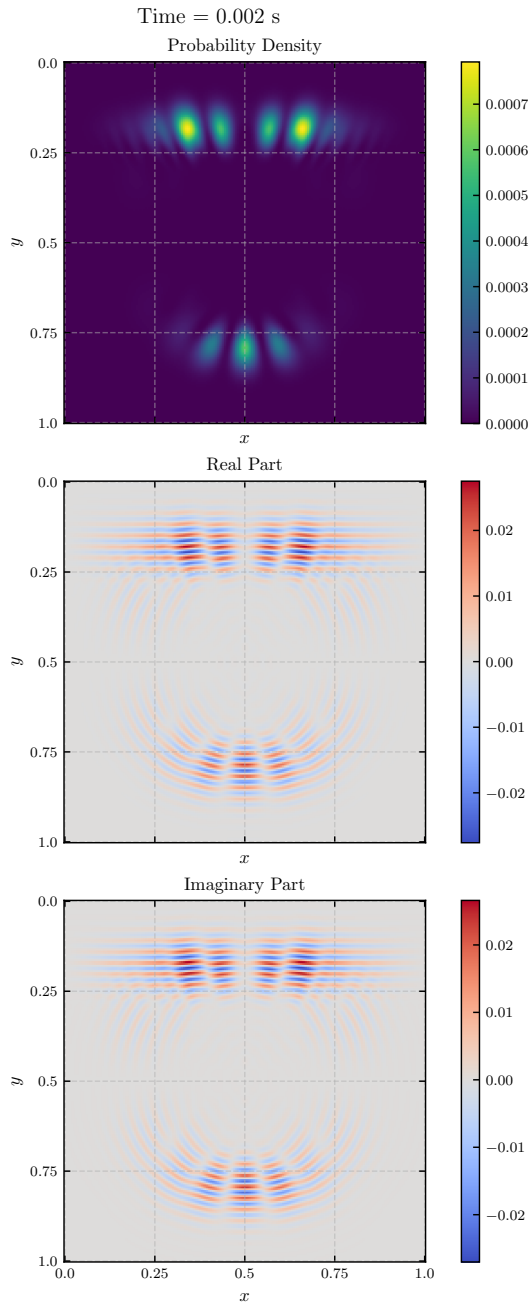


Figure 5. Wave function after passing the slits at $t = 0.002$ s. Clear interference fringes are visible in the probability distribution $|u|^2$, indicating coherent superposition of the transmitted wave packets. The real and imaginary components exhibit alternating regions of positive and negative amplitude, reflecting the phase differences between the paths through the two slits.