Simulating the double-slit experiment by evolving the two-dimensional time-dependent Schrödinger equation using the Crank-Nicolson scheme.

Einar Skoglund, Emil J. Kvernevik
(Dated: December 12, 2023)

In this study, we employ the two-dimensional time-dependent Schrödinger equation and a Gaussian wave packet to model a particle in a box, while a high valued constant potential is used to simulate slit barriers. The Schrödinger equation is evolved over time by using the Crank-Nicolson scheme for partial differential equations (PDE). We run simulations for single, double and triple slit barriers, and we observe the resulting diffraction pattern. We find that the Crank-Nicolson scheme, along with the `arma::spsolve` matrix solver[1], conserves the probability of a wave function going through a double slit with an accuracy that is within machine precision. A good visual representation of the wave characteristics of quantum particles, can be found in the two dimensional animations given in the GitHub repository.

The GitHub repository for this article can be found under the "Project 5" folder at https://github.uio.no/emiljk/FYS3150.git.

I. INTRODUCTION

The double-slit experiment serves as a pivotal study in quantum mechanics. In addition to playing a fundamental role in the development of quantum mechanics, it demonstrated that matter and energy satisfy the seemingly incompatible classical definitions for both waves *and* particles.

The experiment was first performed in 1801 by British polymath, Thomas Young, as a means to demonstrate the wave behavior of visible light [2]. The experiment consisted of directing a beam of particles towards a barrier with two slits, and then measuring their position with a detector screen placed behind the barrier. At the time of the first experiment, it was believed that light could only be characterized as *either* waves *or* particles. It would not be until approximately a hundred years later, along with the dawn of modern physics, that it would be revealed that light could in fact display both wave *and* particle characteristics - this is known as the **wave-particle duality**.

In short, the wave-particle duality suggests that when not observed, the particles seem to behave as waves, creating these interference patterns. Yet when observations or measurements are made, they seem to behave as discrete particles, instead appearing at specific points on the detector screen. This enforces the inherently probabilistic nature of the particles, which can be described by the Schrödinger equation.

The Schrödinger equation is a linear partial differential equation (PDE) which describes the time evolution of a particle wave function. It was postulated by Erwin Schrödinger in 1925, published in 1926 [3], and would go on to win him the nobel prize in 1933.

This brings brings us to the focus of this study. The aim is to simulate variations of the double-slit experiment by evolving the time-dependent Schrödinger equation. As the Schrödinger equation is a linear PDE, a numerical method must be applied to solve it. In order to achieve this, a variety of numerical methods can be utilized. Among them, the Crank-Nicolson scheme stands out as a particularly well-suited approach, as it possesses a small magnitude error and stability for all choices of time and position step sizes.

The structure of this study is as follows. In section II we provide the necessary theoretical framework to understand the model, along with a thourough explanation of the numerical method used, namely the Crank-Nicolson scheme. An extensive description of the approach used in the simulation procedure is provided, along with any quantities and parameters used in the process of evolving the Schrödinger equation. In section III, the results of the simulations are presented, and comparisons are made for different variations of the double-slit experiment. The findings from the simulations are discussed, and different approaches and future work to improve the accuracy of the model are suggested. Lastly, in section IV we present the findings from the simulation and summarize all of the key conclusions, and conclude the study.

II. METHODS

This section introduces the formalism and quantities we employ for simulating variations of the double-slit experiment. It provides an explanation of the implementation of the Crank-Nicolson scheme in our simulation, as well as a detailed description of the methods and quantities utilized to derive our results.

A. The Schrödinger equation

To model the double-slit experiment we make use of the time-dependent Schrödinger equation. This describes the time evolution of a quantum state, and is in general represented as

$$i\hbar\frac{d}{dt}|\Psi\rangle = \hat{H}|\Psi\rangle \tag{1}$$

where the $|\Psi\rangle$ denotes the quantum state, $\hat{H}$ is some Hamiltonian operator, $\hbar$ is the reduced Planck constant and i the imaginary unit. Though it is not clear *what* exactly the nature of a quantum state is, it's correlation with probability has been proven through the Born rule[4].

For the scope of this project, we consider the case of a single, non-relativistic particle in two dimensions. Given that we work in "position space", this allows us to then express the quantum state $|\Psi\rangle$ through the wave function $\Psi(x, y, t)$. The Schrödinger equation is then expressed as a linear **partial differential equation** (PDE):

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

where $-\frac{\hbar^2}{2m}\frac{\partial^2\Psi}{\partial x^2}$ and $-\frac{\hbar^2}{2m}\frac{\partial^2\Psi}{\partial y^2}$ represent the kinetic energy, with m as the particle mass, and V as the potential. Additionally, only the case of a *time-independent* potential, i.e. $V = V(x, y)$, will be explored. With the assumptions above in mind, the Born rule can then be written as

$$p(x, y\,; t) = |\Psi(x, y, t)|^2 = \Psi^*(x, y, t)\,\Psi(x, y, t), \tag{3}$$

where p is the probability density of detecting the particle at a position (x, y) at a time t. The star in Ψ^* stands for the complex conjugate of Ψ. Further, the expression for the probability density p must equal 1, as the particle must exist somewhere in the space where the measurements are made - in other words, the wave function must be normalized in order to apply the Born rule.

By scaling away dimensional variables, and allowing $x, y \in [0, 1]$ and $t \in [0, T]$, we can then reduce equation (2) to a simplified PDE on the form

$$i\frac{\partial u}{\partial t} = -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + v(x, y)u, \tag{4}$$

where $v(x, y)$ is some potential.

The Born rule then takes the form

$$p(x, y; t) = |u(x, y, t)|^2 = u^*(x, y, t)\,u(x, y, t). \tag{5}$$

For the initial wave function, i.e. $u(x, y, t = 0)$, we use an unnormalised Gaussian wave packet

$$u(x, y, t = 0) = e^{-\frac{(x-x_c)^2}{2\sigma_x^2} - \frac{(y-y_c)^2}{2\sigma_y^2} + ip_x x + ip_y y}. \tag{6}$$

Here, x_c and y_c denote the coordinates of the centre of the initial wave packet, σ_x and σ_y its initial widths in the x and y directions, and p_x and p_y its wave packet momenta.

In order to simulate our case of the Schrödinger equation, we need to apply a numerical method, such that the continuous x, y-plane is divided into a two dimensional grid of discrete points. In this grid, both the x- and y-directions evolve with the same step size h, such that

$$x \to x_i = ih,$$
$$y \to y_j = jh,$$

where $i, j = 0, 1, ..., M-1$. For the x- and y-direction, M corresponds to the number of points along the axes, including their boundary points; hence being discretized using $M-1$ steps. Further, the time t evolves with a time step Δt, such that

$$t \to t_n = n\Delta t$$

where $n = 0, 1, ..., N_t$, where N_t corresponds to the number of points in time.

This modification causes our expression for $u(x, y, t)$ to take the form $u(ih, jh, n\Delta t)$, which can then be written as u^n_{ij}. In this modification, the i, j-indices denote the two spatial coordinates, while n denotes the time index. Similar to our expression for $u(x, y, t)$, the same procedure happens to our equations for the potential $v(x, y)$ and the probability $p(x, y\,; t)$, where they take the form

$$v(x, y) \to v(ih, jh) \equiv v_{ij}$$
$$p(x, y\,; t) \to p(ih, jh\,; n\Delta t) \equiv p^n_{ij}$$

where we now have that

$$p^n_{ij} = u^{n*}_{ij}\; u^n_{ij},$$

instead of the former expression given in equation (5).

With the discretization in order, we then look toward applying the numerical method to solve our PDE.

B. The Crank Nicolson scheme

In this section we discuss and compare the explicit scheme, the implicit scheme and the Crank-Nicolson scheme. By using the latter we then derive an algorithm for solving the Schrödinger equation. In order to solve the PDE we make use of **finite difference methods**. These are methods that approximate a derivative in a point using finite (discretized) differences from nearby points.

To simplify the explanation of the different schemes, we employ the one dimensional PDE

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}. \tag{7}$$

We start by showing the **explicit scheme**, which uses a forward difference method in order to evaluate the first derivative. That is, we are using the difference between the point u^n_{ij} at our current time step and the point u^{n+1}_{ij} at the next time step.

The time derivative is then approximated as

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}, \tag{8}$$

with a truncation error of $\mathcal{O}(\Delta t)$ [5].

To approximate the second derivative for the spatial dimension, we employ the midpoint formula [5]

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}, \tag{9}$$

with a truncation error of $\mathcal{O}(\Delta x^2)$ [5].

By setting the right hand side of this equation equal to F_i^n, we can then approximate (7) with the explicit scheme as

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = F_i^n. \tag{10}$$

However, the downside of the explicit scheme is that it has a stability condition of $\Delta t/\Delta x^2 \leq 1/2$ [5]. A stability condition ensures that a numerical algorithm continues to produce accurate results over a large number of calculations, without increasing the magnitude of the error from the input data.

The **implicit scheme** has the same truncation error as the explicit scheme, but is stable for all choices of Δt and Δx [5]. Moreover, in this method we use the point at the *previous* time step u_i^{n-1} to approximate the time derivative as

$$\frac{\partial u}{\partial t} \approx \frac{u_i^n - u_i^{n-1}}{\Delta t}. \tag{11}$$

When using the same approximation for the second spatial derivative (9) we can now approximate (7) with the implicit scheme as

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} = F_i^n. \tag{12}$$

To more easily relate this to the explicit scheme (10), we can rewrite (12) as

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = F_i^{n+1}. \tag{13}$$

By using a linear combination of the explicit and the implicit scheme, we can derive what is known as the **Crank-Nicolson scheme** [6]:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2}(F_i^{n+1} + F_i^n). \tag{14}$$

Similar to the explicit scheme, the Crank-Nicolson scheme is also stable for all choices of Δt and Δx. Important to note however, the Crank-Nicolson scheme possesses a truncation error in time of $\mathcal{O}(\Delta t^2)$ [5], not $\mathcal{O}(\Delta t)$ as for the implicit and explicit scheme. As such, using the Crank-Nicolson approach should therefore result in a more accurate approximation of the time evolution of the Schrödinger equation.

By discretizing the Schrödinger equation (4) according to the Crank-Nicolson approach (14) and sorting the terms, we show in appendix A that we can modify equation (4) to become

$$\begin{aligned} u_{ij}^{n+1} &- r\left[u_{i+1,j}^{n+1} - 2u_{ij}^{n+1} + u_{i-1,j}^{n+1}\right] \\ &- r\left[u_{i,j+1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j-1}^{n+1}\right] + \frac{i\Delta t}{2} v_{ij} u_{ij}^{n+1} \\ = u_{ij}^n &+ r\left[u_{i+1,j}^n - 2u_{ij}^n + u_{i-1,j}^n\right] \\ &+ r\left[u_{i,j+1}^n - 2u_{ij}^n + u_{i,j-1}^n\right] - \frac{i\Delta t}{2} v_{ij} u_{ij}^n, \end{aligned} \tag{15}$$

where $r \equiv \frac{i\Delta t}{2h^2}$.

In appendix B we show that this equation can be written as the matrix equation

$$\mathbf{A}\vec{u}^{n+1} = \mathbf{B}\vec{u}^n, \tag{16}$$

where $\vec{u}$ contains all the internal points (i.e. not including the boundary points) of the xy-grid, and $\mathbf{A}$ and $\mathbf{B}$ are sparse matrices (see appendix B).

To find the next $\vec{u}^{n+1}$ from the current $\vec{u}^n$ we can first perform the matrix multiplication $\mathbf{B}\vec{u}^n = \vec{b}$, before solving the matrix equation $\mathbf{A}\vec{u}^{n+1} = \vec{b}$.

As the $\mathbf{A}$-matrix is quite sparse in nature, it is preferable to use iterative methods such as Jacobi's method or the Gauss-Seidel method, as these only requires that we are storing the non-zero elements of $\mathbf{A}$. This is in contrast to direct methods, like LU-decomposition, which usually require storing the matrix in its entirety.

For the purposes of this study, we will employ the `arma::spsolve` method - see the Tools subsection. This solver uses the SuperLU library [1] which performs a LU-decomposition, and not the iterative methods as suggested. However, since the SuperLU library is specialized for handling sparse matrices, it should be particularly well suited for solving our matrix equation.

C. Simulation

As mentioned in section II A, the total probability in the probability function p_{ij}^n should be conserved over time - in simpler terms, the total probability should be equal to 1. To test how consistent our model is with the theory, we run two simulations with different slit configurations and observe how the total probability deviates from 1 over time.

Since most of the parameters that govern the simulations, e.g. the step size h and the time step Δt, are held fixed across all simulations, they will not be explicitly reiterated. Their values are instead presented in TABLE I.

For the initial simulation, the model is ran over a time of $T = 0.008$. In addition to the fixed parameters in TABLE I, the wave packet width is set to $\sigma_y = \sigma_x = 0.05$.

TABLE I. An overview of the fixed parameters for the initlal Gaussian wave packet (6) and the simulation. Here h denotes the position step size, and Δt the time step size. The parameters x_c and y_c denote the center of the wave packet, and σ_x and p_x its width and momenta, respectively, in the x-direction. The momenta in the y-direction is denoted by p_y. All values are represented as dimensionless quantities.

Parameter	Value
h	0.005
Δt	2.5×10^{-5}
x_c	0.25
σ_x	0.05
p_x	200
y_c	0.5
p_y	0

Further, the initial potential is set to $v_0 = 0$, meaning that the simulation is ran without any potential slit barrier.

A second simulation is then performed with a wider wave packet width in the y-direction, with $\sigma_y = 0.10$. A double slit barrier is also added with $v_0 = 1 \times 10^{10}$, effectively reflecting the part of the wave that does not escape through the slits. We do not include quantum tunneling in the model of the potential slit barriers.

For all simulations, we maintain roughly the same setup of the slit barrier, with the only thing changing being the number of slits. For the double slit barrier, the wall is positioned in the middle of the x-axis, at $x = 0.5$ with a thickness of $\Delta x = 0.02$. The slit setup is symmetric around $y = 0.5$, where the both the aperture of the slits, and the length of the wall piece separating the two slits, is $\Delta y = 0.05$. Subsequently for the double slit, the wall piece separating the two slits is centered around $y = 0.5$. With the double slit barrier in order, the deviations of the total probability for both simulations is then observed to determine how well the model concurs with the theoretical expectations.

Upon determining how well the total probability p_{ij}^n is conserved, we then shift our focus towards modeling the time evolution of p_{ij}^n along the two dimensional grid by including a double slit barrier. This is done by performing the same simulation for $T = 0.002$, with an even wider wave packet width in the y-direction, with $\sigma_y = 0.20$. In addition to look at the probability, we also observe and compare the real and imaginary parts of the wave function.

Further, by making the assumption that a detector screen measures the particle at $x = 0.8$ at time $t = 0.002$ and using our results from the previous simulation, we determine the detection probability along the screen at this time. Since we make the assumption that the particle is detected *somewhere* along this screen, the one-dimensional probability function is normalized to sum to 1.0. In simpler terms, what is determined is $p(y|x = 0.8\,; t = 0.002)$.

The model is then adjusted to be able to simulate the cases of single-slit and triple-slit experiments, where we employ same slit aperture and slit separation as for the double slit case. We then repeat the procedure, and determine the one-dimensional probability function given above.

To get a better understanding of the time evolution of the particle wave, we also make 2D colormap animations of the probability function in the xy-plane for the three slit configurations.

D. Tools

In this section we present the tools and libraries employed to perform the simulations and to visualize our results.

The Crank-Nicolson scheme is implemented as code in C++. We use the `armadillo`-library [7][8] to create and perform calculations on complex vectors and matrices, as well as to store our results. The `arma::spsolve`-solver from the SuperLU library [1] is used to solve the matrix equation (16).

The results from the simulations are processed and visualized through Python. The results made from our simulation in C++ are import into Python through the `pyarma`-library, after which the `numpy`-library is employed to ease the process of organizing the results. Lastly, to visually represent our findings and the simulations, we employ the `matplotlib`-library.

III. RESULTS AND DISCUSSION

In this section we present and discuss the results of the simulations, in addition to providing animations that show the time evolution of the probability function in a more intuitive manner.

We start by seeing how well our numerical algorithm preserves the probability over time. In FIG 1 we see the deviation of the total probability from 1.0 for the two simulations: the one with a double slit barrier and the one without any barrier. We see that for both slit configurations the probability seems to drift away from the conserved quantity as the time increases. The deviation is somewhat greater for the simulation without any barrier compared to the one with a double slit barrier. However, as both deviations have a magnitude of order 10^{-14} this difference seems negligible. As the computer can represent doubles with the precision of about 15 decimal digits, the accuracy of the simulation seems to be mostly limited by computer accuracy. As we use the well documented `arma::spsolve` method,

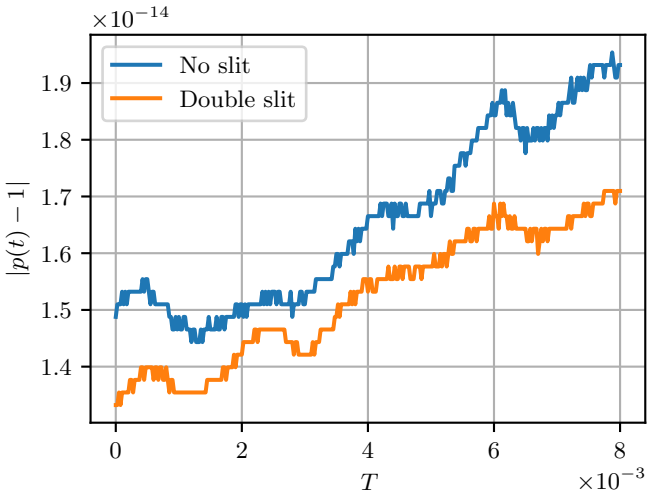

FIG. 1. The absolute value of deviation of the total probability (summed over all points in the xy-grid) from 1.0 as a function of time for two different slit barrier configurations. With the double slit barrier we use an initial wave packet width in the y-direction of $\sigma_y = 0.10$, while without the barrier we use $\sigma_y = 0.05$.

and not some matrix solver we have made by ourselves, this small deviation from the conserved probability is as expected.

Even if the two simulations are initialized with different widths, with $\sigma_y = 0.10$ for the double slit and $\sigma_y = 0.05$ for the one without any barrier, it is not very clear form FIG 1 if this is affecting the results. As the less wider wave (the one without any barrier) is the least accurate, one possible hypothesis could be that it has a greater margin of error since it is confined within a smaller number of points compared to the wave with the double slit barrier. However, we can not conclude as we here only have a single example to make any assumptions from. Later studies could investigate if there is a correlation between initial wave width and accurate conservation of probability.

We have also only investigated the total probability at each time step, and not the specific wave function at each position at each time. Although the conservation of probability is a good indicator that the simulation yields accurate result, further studies may investigate whether the specific wave function is accurately calculated at each iteration in time.

We have made visual representations of the 2D probability function in the xy-plane. For the double slit configuration with an initial width in the y-direction of $\sigma_y = 0.20$ the probability function is visualized in FIG 2a at three different times. At $t = 0$ we see that the wave packet is centered around the initial configurations as planned. At $t = 0.001$ the wave has reached the double slit. Here we can see that the wave is split in two parts at the other side of the slit, while the remaining part of the wave is reflected away from the barrier. The potential barrier therefore seems to be working as expected. At $t = 0.002$ we see that the wave is clearly split over multiple positions, showing the wave characteristic of the particle.

In FIG 2b and 2c we have illustrated the real and imaginary part of the wave function for the same time frames as for the probability. The real and imaginary colormaps looks more discretely distributed compared to the more confined probability colormap. The reason is found from the fact that the real and imaginary part of a complex number is represented as cosine and sine functions, respectively. Therefore, the real and imaginary part will oscillate causing a non-continuous pattern when illustrated in the xy-plane.

With the born rule (5) we get the more confined probability illustration in FIG 2a. Here, the phase difference between the real and imaginary part of the wave function causes constructively interference which at $t = 0.002$ could be read of a screen.

In FIG 3 we have illustrated the interference pattern of the probability for the simulation above at $t = 0.002$ along $x = 0.8$. We have also included the results from two similar simulations, but now with single and triple slit configurations.

For all slit configurations we see a symmetric pattern. This indicates that the symmetric potential slit barriers have been implemented correctly.

For the double slit we recognize the three peaks from FIG 2a where the peak at the middle is the most intense. For the single slit we see only one peak as one would easily predict, while the triple slit causes a more complex pattern with two big peaks at the middle and two smaller peaks at the sides. This is caused by the interference of the wave by itself, as a consequence of being "split" by the slits.

To more easily get an understanding of how the wave interfere with itself after hitting the potential barrier, we have created 2D animations (in the style of FIG 2) of the simulations for the three different slits.[1] These can be found at https://github.uio.no/emiljk/FYS3150/tree/main/Project_5/animations. From these we see

[1] For the double slit barrier we have also made a 3D animation. Though it might not be making the wave function any easier to interpret, it could be quite fun to look at.

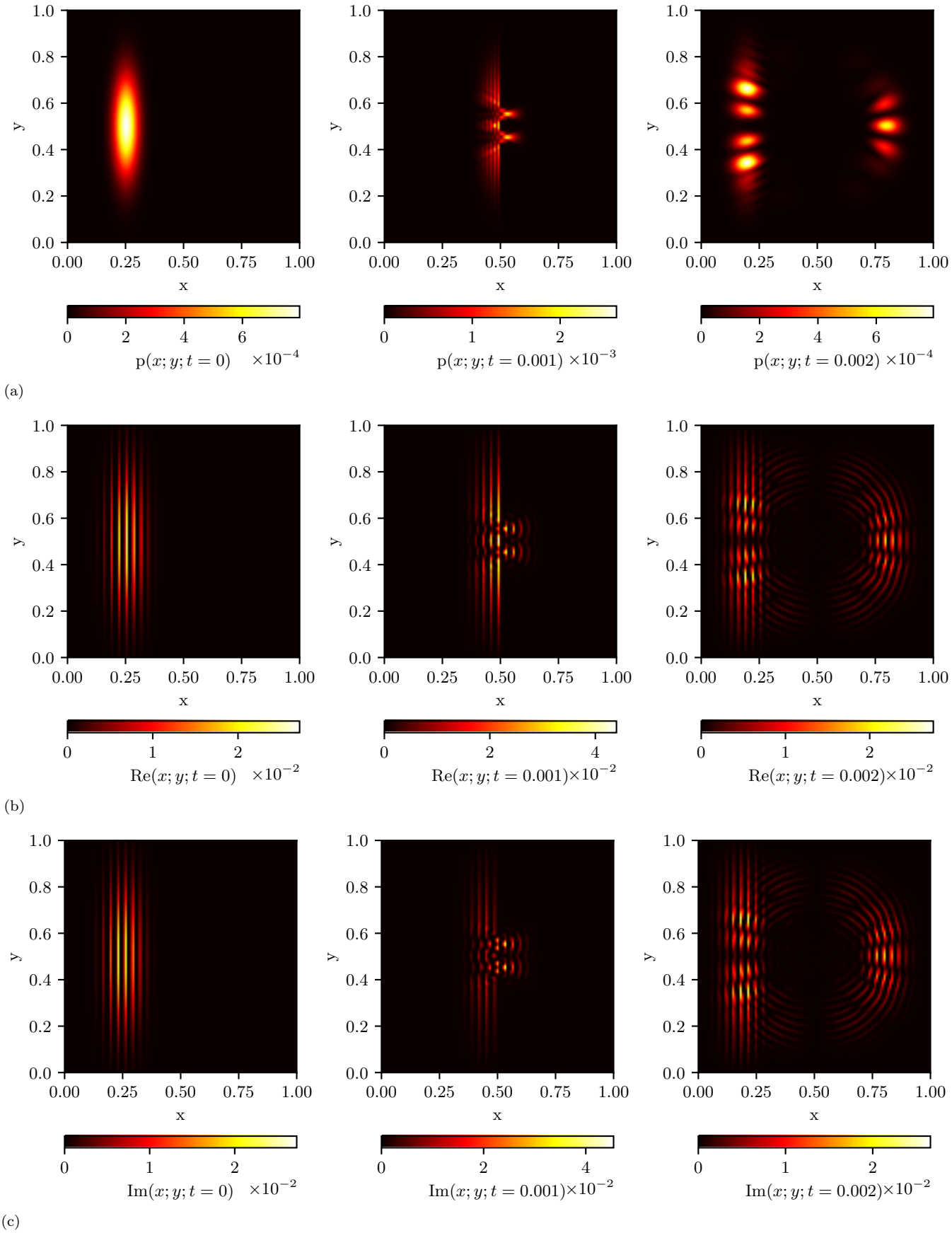

FIG. 2. Colormaps at three different times generated by simulating a wave with initial width in the y-direction $\sigma_y = 0.20$. In (a) we show the probability calculated by the Born rule (3), in (b) we show the real part of the wave function and in (c) we show the imaginary part of the wave function. The colormaps are adjusted according to the maximum value at each frame.

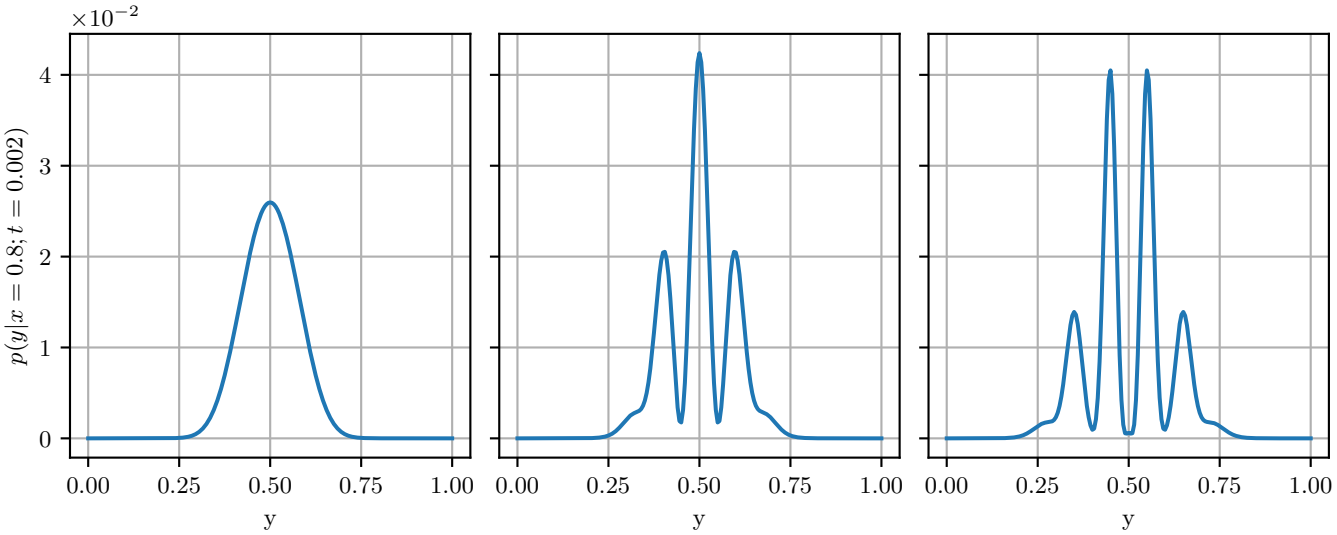

FIG. 3. The normalized detection probability at $T = 0.002$ along $x = 0.8$ for three different slit configurations: single (left), double (middle) and triple (right). All other simulation parameters are the same as used to generate FIG 2.

how the particle very much resembles a wave rather than what we would think of as a "particle ball", again illustrating the wave characteristics of the particles. The numerical algorithm therefore seems suitable for illustrating the double slit experiment.

IV. CONCLUSION

Our application of the Crank-Nicolson scheme to solve the time-dependent Schrödinger equation has provided valuable insights into the intricacies of the double-slit experiment, and the interesting interplay between wave and particle characteristics.

The fact that our utilization of the `arma::spsolve` matrix solver[1][7][8] has shown that the probability of the wave function is conserved, as proven by its adherence to the Born rule, attests the proficiency of our numerical algorithm in generating accurate results. However, there are still several avenues for future work in this study and into the specific evolution of the wave function; e.g. the effects of quantum tunneling or nonlinear quantum mechanics, in addition to exploring how well other numerical methods fare in the same scenario.

Furthermore, our exploration of the detection probability of the particle across three slit configurations, in addition to visually representing the time evolution, establishes that our model of the potential slit barriers has been adequately implemented.

In addition to our study yielding a visual representation that corresponds nicely with quantum mechanics theory, it has also enhanced our intuitive understanding of the wave characteristics exhibited by particles. Our findings provide an extensive insight into the simulation and comprehension of the double-slit experiment; one of the most important and fundamental studies within the field of quantum mechanics.

[1] Xiaoye S. Li. An overview of SuperLU: Algorithms, implementation, and user interface. *ACM Transactions on Mathematical Software*, 31(3):302–325, September 2005.

[2] Young Thomas. Experiments and calculations relative to physical optics. *Phil. Trans. R. Soc.*, 94:1–16, 1804.

[3] E. Schrödinger. An undulatory theory of the mechanics of atoms and molecules. *Physical Review*, 28:1049–70, 1926.

[4] Born M. Zur quantenmechanik der stoßvorgänge. *Zeitschrift für Physik*, 37:863–867, 1926.

[5] M.H Jensen. *Computational Physics*, chapter 10, pages 304–312. Department of Physics, University of Oslo, 2015.

[6] Nicolson P. Crank J. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 1:50–67, 1947.

[7] Curtin R Sanderson C. Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 1:26, 2016.

[8] Curtin R Sanderson C. A user-friendly hybrid sparse matrix class in c++. *Lecture Notes in Computer Science (LNCS)*, 10931:422–430, 2018.

APPENDIX A: DISCRETIZING THE SCHRÖDINGER EQUATION ACCORDING TO THE CRANK-NICOLSON APPROACH

In this section we show how to represent the Schrödinger equation (4) using the Crank-Nicolson scheme (14).

By discretizing the components of the Schrödinger equation (4) using (8) for the time derivative and (9) for the spatial derivatives we get

$$i\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\Delta t}=F_{ij}^{n}, \tag{17}$$

where

$$\begin{aligned}F_{ij}^{n}=&-\frac{u_{i+1,j}^{n}-2u_{ij}^{n}+u_{i-1,j}^{n}}{\Delta x^{2}}\\&-\frac{u_{i,j+1}^{n}-2u_{ij}^{n}+u_{i,j-1}^{n}}{\Delta y^{2}}+v_{ij}u_{ij}^{n}.\end{aligned} \tag{18}$$

Now we insert this in the Crank-Nicolson scheme (14) which gives

$$\begin{aligned}i\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\Delta t}=&\frac{1}{2}\Bigg[-\frac{u_{i+1,j}^{n+1}-2u_{ij}^{n+1}+u_{i-1,j}^{n+1}}{\Delta x^{2}}\\&-\frac{u_{i,j+1}^{n+1}-2u_{ij}^{n+1}+u_{i,j-1}^{n+1}}{\Delta y^{2}}+v_{ij}u_{ij}^{n+1}\\&+\frac{u_{i+1,j}^{n}-2u_{ij}^{n}+u_{i-1,j}^{n}}{\Delta x^{2}}\\&+\frac{u_{i,j+1}^{n}-2u_{ij}^{n}+u_{i,j-1}^{n}}{\Delta y^{2}}-v_{ij}u_{ij}^{n}\Bigg].\end{aligned} \tag{19}$$

By now using that $\Delta x=\Delta y=h$, multiplying both sides by $i\Delta t$ and introducing $r\equiv\frac{i\Delta t}{2h^{2}}$ we get

$$\begin{aligned}u_{ij}^{n}-u_{ij}^{n+1}=&-r(u_{i+1,j}^{n+1}-2u_{ij}^{n+1}+u_{i-1,j}^{n+1})\\&-r(u_{i,j+1}^{n+1}-2u_{ij}^{n+1}+u_{i,j-1}^{n+1})+\frac{i\Delta t}{2}v_{ij}u_{ij}^{n+1}\\&+r(u_{i+1,j}^{n}-2u_{ij}^{n}+u_{i-1,j}^{n})\\&+r(u_{i,j+1}^{n}-2u_{ij}^{n}+u_{i,j-1}^{n})-\frac{i\Delta t}{2}v_{ij}u_{ij}^{n}.\end{aligned} \tag{20}$$

Lastly we collect the terms at the same time step at either side of the equation, resulting in (15):

$$\begin{aligned}&u_{ij}^{n+1}-r\left[u_{i+1,j}^{n+1}-2u_{ij}^{n+1}+u_{i-1,j}^{n+1}\right]\\&\quad-r\left[u_{i,j+1}^{n+1}-2u_{ij}^{n+1}+u_{i,j-1}^{n+1}\right]+\frac{i\Delta t}{2}v_{ij}u_{ij}^{n+1}\\&=u_{ij}^{n}+r\left[u_{i+1,j}^{n}-2u_{ij}^{n}+u_{i-1,j}^{n}\right]\\&\quad+r\left[u_{i,j+1}^{n}-2u_{ij}^{n}+u_{i,j-1}^{n}\right]-\frac{i\Delta t}{2}v_{ij}u_{ij}^{n}.\end{aligned}$$

APPENDIX B: WRITING THE DISCTRETIZED SCHRÖDINGER EQUATION AS A MATRIX EQUATION

Here we show how to write (15) as the matrix equation (16).

We start by collecting the u_{ij} terms on each side of (15), resulting in

$$\begin{aligned} &u_{ij}^{n+1}\left(1+4r+\frac{i\Delta t}{2}v_{ij}\right) \\ &-r\left[u_{i+1,j}^{n+1}+u_{i-1,j}^{n+1}+u_{i,j+1}^{n+1}+u_{i,j-1}^{n+1}\right] \\ &=u_{ij}^{n}\left(1-4r-\frac{i\Delta t}{2}v_{ij}\right) \\ &+r\left[u_{i+1,j}^{n}+u_{i-1,j}^{n}+u_{i,j+1}^{n}+u_{i,j-1}^{n}\right]. \end{aligned} \tag{21}$$

By introducing

$$\begin{aligned} a_{ij}&=1+4r+\frac{i\Delta t}{2}v_{ij}, \\ b_{ij}&=1-4r-\frac{i\Delta t}{2}v_{ij}, \end{aligned}$$

we can rewrite (21) as

$$\begin{aligned} &a_{ij}u_{ij}^{n+1}-r\left[u_{i+1,j}^{n+1}+u_{i-1,j}^{n+1}+u_{i,j+1}^{n+1}+u_{i,j-1}^{n+1}\right] \\ &=b_{ij}u_{ij}^{n}+r\left[u_{i+1,j}^{n}+u_{i-1,j}^{n}+u_{i,j+1}^{n}+u_{i,j-1}^{n}\right]. \end{aligned} \tag{22}$$

For the M^2 different points u_{ij} in the xy-plane, (22) will consist of M^2 equations with M^2 variables. To write this as a matrix equation it is useful to collect all the variables, i.e. all the points, in a column vector. We will do this by stacking the columns of the matrix u^n on top of each other to get a vector $\vec{U}^n$:

$$u^n=\begin{bmatrix} u_{0,0}^n & u_{0,1}^n & \cdots & u_{0,M-2}^n & u_{0,M-1}^n \\ u_{1,0}^n & u_{1,1}^n & \cdots & u_{1,M-2}^n & u_{1,M-1}^n \\ u_{2,0}^n & u_{2,1}^n & \cdots & u_{2,M-2}^n & u_{2,M-1}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{M-2,0}^n & u_{M-2,1}^n & \cdots & u_{M-2,M-2}^n & u_{M-2,M-1}^n \\ u_{M-1,0}^n & u_{M-1,1}^n & \cdots & u_{M-1,M-2}^n & u_{M-1,M-1}^n \end{bmatrix}$$

$$\rightarrow \vec{U}^n=\begin{bmatrix} u_{0,0}^n \\ u_{1,0}^n \\ \vdots \\ u_{M-2,M-1}^n \\ u_{M-1,M-1}^n \end{bmatrix}.$$

To relate a position ij in u^n to a position K in $\vec{U}^n$ we set

$$K=i+jM,$$

which means that we can rewrite $\vec{U}^n$ as

$$\vec{U}^n=\begin{bmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{M^2-2}^n \\ u_{M^2-1}^n \end{bmatrix}.$$

We can also rewrite (22) as

$$\begin{aligned} &a_K u_K^{n+1}-r\left[u_{K+1}^{n+1}+u_{K-1}^{n+1}+u_{K+M}^{n+1}+u_{K-M}^{n+1}\right] \\ &=b_{ij}u_K^n+r\left[u_{K+1}^n+u_{K-1}^n+u_{K+M}^n+u_{K-M}^n\right]. \end{aligned} \tag{23}$$

We will now try to see how the right hand side of this equation will look like as a matrix $\mathbf{B}$. Finding the matrix for the left hand side will be completely analogous. We start by showing a random row at row number S:

$$\mathbf{B}=\begin{array}{c} \begin{matrix} S-M & & S-1 & S & S+1 & & S+M & \end{matrix} \\ \begin{matrix} \downarrow & & \downarrow & \downarrow & \downarrow & & \downarrow & \end{matrix} \\ \begin{bmatrix} \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \\ \cdots & r & \cdots & r & b_S & r & \cdots & r & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array},$$

where we have indicated the column number found from (23) above.

Adding the next rows will then look like:

$$\mathbf{B} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & r & 0 & 0 & \cdots & r & b_S & r & 0 & 0 & \cdots & r & 0 & 0 & \cdots \\ \cdots & 0 & r & 0 & \cdots & 0 & r & b_{S+1} & r & 0 & \cdots & 0 & r & 0 & \cdots \\ \cdots & 0 & 0 & r & \cdots & 0 & 0 & r & b_{S+2} & r & \cdots & 0 & 0 & r & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix},$$

where we have used colors to indicate the pattern which is starting to form.

It seems apparent that the full matrix will be on the form

$$\mathbf{B} = \begin{bmatrix} & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{bmatrix},$$

where the black line is the main diagonal with values b_K, the red lines are the sub and super diagonal with values r and the blue lines are the "band" diagonals with values r. All other elements are zero.

This matrix will be of size $M^2 \times M^2$ representing the M^2 variables. The left hand side of equation (23) can now be written as $\mathbf{B}\vec{U}^n$.

However, as we are using Dirichlet boundary conditions, all the boundary points in the xy-plane will be equal to zero:

$$u^n = \begin{bmatrix} u^n_{0,0} & u^n_{0,1} & \cdots & u^n_{0,M-2} & u^n_{0,M-1} \\ u^n_{1,0} & u^n_{1,1} & \cdots & u^n_{1,M-2} & u^n_{1,M-1} \\ u^n_{2,0} & u^n_{2,1} & \cdots & u^n_{2,M-2} & u^n_{2,M-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u^n_{M-2,0} & u^n_{M-2,1} & \cdots & u^n_{M-2,M-2} & u^n_{M-2,M-1} \\ u^n_{M-1,0} & u^n_{M-1,1} & \cdots & u^n_{M-1,M-2} & u^n_{M-1,M-1} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & u^n_{1,1} & \cdots & u^n_{1,M-2} & 0 \\ 0 & u^n_{2,1} & \cdots & u^n_{2,M-2} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & u^n_{M-2,1} & \cdots & u^n_{M-2,M-2} & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

By introducing $d = M - 2$, we then only have d^2 unknowns.

Instead of using $\vec{U}^n$ which contains all points of u^n, we can then introduce a vector $\vec{u}^n$ which only contains the internal points:

$$\vec{u}^n = \begin{bmatrix} u^n_{1,1} \\ u^n_{2,1} \\ \vdots \\ u^n_{M-2,M-2} \end{bmatrix}.$$

We now introduce

$$k = (i-1) + (j-1)(M-2) = (i-1) + (j-1)d, \quad (24)$$

which means that we can rewrite $\vec{u}^n$ as

$$\vec{u}^n = \begin{bmatrix} u^n_0 \\ u^n_1 \\ \vdots \\ u^n_{d^2-1} \end{bmatrix}.$$

By removing the boundary points in $\vec{U}^n$, we also have to remove the rows and columns in $\mathbf{B}$ related to the boundary points. By sectioning $\mathbf{B}$ in boundary-parts and internal-parts we will get subsections looking like

$$\mathbf{B} = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & \bullet & \bullet & & & 0 & 0 & \bullet & & & & & \cdots \\ \cdots & \bullet & \ddots & \ddots & & 0 & 0 & & \ddots & & & & \cdots \\ \cdots & & \ddots & \ddots & \bullet & 0 & 0 & & & \ddots & & & \cdots \\ \cdots & & & \bullet & \bullet & 0 & 0 & & & & & \bullet & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \cdots & \bullet & & & & 0 & 0 & \bullet & \bullet & & & & \cdots \\ \cdots & & \ddots & & & 0 & 0 & \ddots & & \ddots & & & \cdots \\ \cdots & & & \ddots & & 0 & 0 & & \ddots & \ddots & \bullet & & \cdots \\ \cdots & & & & \bullet & 0 & 0 & & & \bullet & \bullet & & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where the zeros are explicitly shown on the rows and columns of the boundary points. These sections are two elements wide because of the zero-elements on top and bottom of each column in the matrix u^n. All elements not on these rows and columns, and not on the diagonals, are also zero. We have used bullet points to indicate the diagonal nonzero elements.

By removing these we get subsections looking like

$$\mathbf{B} = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & \bullet & \bullet & & & \bullet & & & & \cdots \\ \cdots & \bullet & \ddots & \ddots & & & \ddots & & & \cdots \\ \cdots & & \ddots & \ddots & \bullet & & & \ddots & & \cdots \\ \cdots & & & \bullet & \bullet & & & & \bullet & \cdots \\ \cdots & \bullet & & & & \bullet & \bullet & & & \cdots \\ \cdots & & \ddots & & & \bullet & \ddots & \ddots & & \cdots \\ \cdots & & & \ddots & & & \ddots & \ddots & \bullet & \cdots \\ \cdots & & & & \bullet & & & \bullet & \bullet & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where the main diagonal elements (black dots) are now found as b_k (not b_K). We note that $\mathbf{B}$ is still a quite sparse matrix.

As we have removed the boundary points, the left hand side of equation (23) can now be written as $\mathbf{B}\vec{u}^n$ (instead of $\mathbf{B}\vec{U}^n$). We can in similar way construct a matrix $\mathbf{A}$ for the right hand side, resulting in the matrix equation (16)

$$\mathbf{A}\vec{u}^{n+1} = \mathbf{B}\vec{u}^n.$$