Wave Dynamics through Slit Potentials — A Memory-Efficient Stencil Scheme

Adrian Gundersen,[*] Casper André Johnsen,[†] and Victor Berge Johansen[‡]

(Dated: December 14, 2025)

We present a memory-efficient numerical scheme for solving the two-dimensional time-dependent Schrödinger equation (*TDSE*) for a Gaussian wave packet in a slit potential. The time evolution is done by a matrix-free Crank–Nicolson algorithm. By discretizing the equations, we rewrite the continuous problem as a matrix system. This is further reduced to arrays, which scale better and are more efficient and less memory-consuming. This is solved using a matrix-free red–black Gauss–Seidel solver. The scheme is unconditionally stable and, for our simulation parameters, conserves the discrete L^2-norm to within $\mathcal{O}(10^{-7})$. We validate the method on single-, double-, triple-, and eight-slit barriers. The simulations display expected qualitative behavior. We observe circular wavefronts and a Gaussian single-slit detection. This implies we are in the near-field, so Fraunhofer diffraction does not apply in our geometry. For multi-slit setups we observe an increasingly sharp interference pattern for an increasing number of slits. The scheme can easily be adapted and is a good framework for further computational studies of the time-dependent Schrödinger equation. All results can be reproduced by running the code at our GitHub repository [a].

I. INTRODUCTION

The double-slit experiment is a cornerstone in the study of waves, both light waves and matter waves. In 1801, a century before the birth of quantum mechanics, Thomas Young presented the idea [1]. Since then, multiple experiments have reproduced the results and studied interference, coherence, phase shifts, particle behavior, and more. The double-slit experiment has become a fundamental experiment that can be tweaked and varied to both confirm and disprove theories. In this paper we will consider a baseline setup and present a computational framework for studying the double-slit experiment. The aim is to develop and test a memory efficient Crank–Nicolson scheme for the 2D *TDSE* in slit potentials. We investigate how the scheme conserves probability over time and to what degree it shows qualitative and quantitative correspondence to observed experiments.

We start by introducing the physical model as well as notation. Our model consists of a Gaussian wave packet traveling through a double-slit barrier. By introducing a dimensionless model, we simplify the problem without loss of generality for easier simulation and implementation. Additionally, we discretize the problem in order to solve it numerically and to show that our algorithms converge and are stable.

As the discretization results in large grids, we propose a method of reducing large matrices to arrays that can be more efficiently solved using a red–black Gauss–Seidel algorithm. This is more efficient and less memory-consuming, especially as our grids grow large.

Using our algorithm we run simulations for free propagation, single slit, double-slit, triple slit, and eight slits. The results are used to study conservation of probability, propagation through space, and detection probability.

Our main contributions are a matrix-free Crank–Nicolson scheme solved with a red–black Gauss–Seidel iterator using much less memory than matrix alternatives. We also demonstrate qualitative behavior for slit potentials, providing a framework for simulating the time-dependent Schrödinger equation.

In Section II we introduce the theoretical groundwork for retrieving results and modeling the numerical scheme. Section III presents and discuss our numerical results for probability conservation, wave propagation through single- and double slits and probability detection for varying number of slits. Finally, our results are summarized in Section IV and we discuss how they could be improved. All results can be reproduced by running the code at our GitHub repository [a].

II. METHODS

A. Physical model

1. Linear system

The general formulation of the time-dependent Schrödinger equation, *TDSE*, is as follows:

$$\mathrm{i}\hbar\frac{d}{dt}\left|\mathbf{\Psi}\right\rangle = H\left|\mathbf{\Psi}\right\rangle,$$

where $|\Psi(t)\rangle$ is the state vector and H is the Hamiltonian operator. In two dimensions this can be written as

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

where V is the potential. We also require normalization with respect to the L^2-norm for our wave function mean-

[*] adriangg@uio.no
[†] casjoh@uio.no
[‡] victorbj@uio.no
[a] https://github.uio.no/adriangg/FYS3150

ing that

$$\langle \mathbf{\Psi} | \mathbf{\Psi} \rangle = \|\mathbf{\Psi}\|_2 = \int_\Omega d\mathbf{r}\, \mathbf{\Psi}^* \mathbf{\Psi} = 1$$

where Ω is the domain.

Note that the formulas used are extracted from [2]. However, we have made some notational changes in this paper as described in Table I.

TABLE I: Notational change from [2].

Quantity	**From [2]**	**To**
wave function at grid point	u_{ij}^n	v_{ij}^n
Dimensionless potential	$v(x,y)$	$\phi(x,y)$
timestep	Δt	τ
Stencil coefficients	a, b	α, β

2. *Dimensionless formulation*

In this project, we will use dimensionless variables and a linear time-independent potential giving us

$$u_t = \mathcal{L}u,$$

where

$$\mathcal{L} = \mathrm{i}\Big(\mathbf{\Delta} - \phi(x,y)\Big),$$

and

$$\mathbf{\Delta} \equiv \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right), \quad u_t \equiv \frac{\partial}{\partial t}u.$$

Here $\phi(x,y)$ denotes the dimensionless time-independent potential. Further, we will also assume that our wave functions are Gaussian wave packets described by initial state

$$u(x,y,t=0) = \exp\left[-\frac{(x-x_c)^2}{2\sigma_x^2} - \frac{(y-y_c)^2}{2\sigma_y^2} + \mathrm{i}p_x x + \mathrm{i}p_y y\right],$$

where (x_c, y_c) is the center of the wave packet and (p_x, p_y) is the momentum. The initial widths are described by σ. As we work in dimensionless units, we get that the momentum p is the same as the wavenumber k. This gives us the de Broglie wavelength:

$$\lambda_x = \frac{2\pi}{p_x},$$

where we neglect the wavelength in the y-direction as we will never impose a y-momentum.

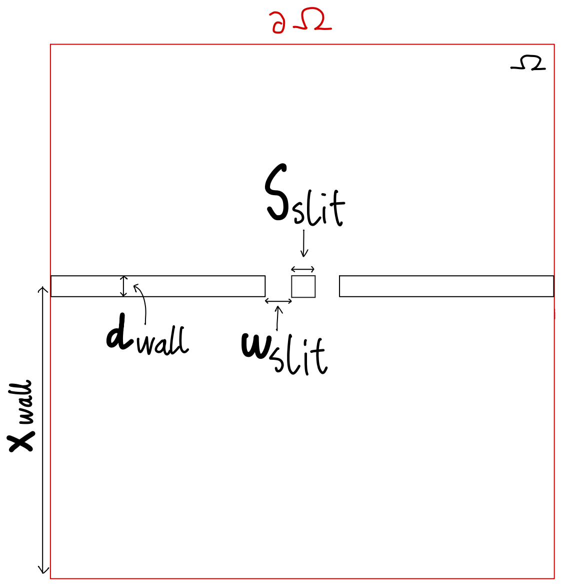

FIG. 1: Illustration of the setup. The red line marks the boundary $\partial\Omega$ of the domain Ω. The wall D_W is centered along the x-axis at x_{wall}, and d_{wall} describes the thickness. s_{slit} is the spacing between the slits and the slit aperture is ω_{slit}. $N_{\text{slits}} = 2$ is the number of slits, and for odd numbers an aperture is centered, for even a barrier is in the center.

We also impose Dirichlet boundary conditions, giving us the full system:

$$\begin{aligned} u_t &= \mathcal{L}u, \quad (x,y,t) \in \Omega \times (0,\infty), \\ u &= 0, \quad (x,y) \in \partial\Omega, \\ u(x,y,0) &= \exp\left[-\frac{(x-x_c)^2}{2\sigma_x^2} - \frac{(y-y_c)^2}{2\sigma_y^2} + ip_x x + ip_y y\right], \\ \|u\|_2 &= 1. \end{aligned}$$

Here $\Omega = (0,1) \times (0,1)$ denotes the dimensionless box with sides 1.0, with the boundary $\partial\Omega$. The box is illustrated in Figure 1.

3. *Potential & slits*

Our main focus will be when the potential ϕ is described by a barrier with slits, mainly a double-slit. Instead of creating a physical wall acting as a barrier, we will introduce a high potential within the wall and zero outside. We define the wall by the domain D_W

$$D_W = \left\{(x,y) \in \Omega : |x - x_{\text{wall}}| \leq \frac{d_{\text{wall}}}{2},\ y \notin S\right\},$$

where

$$S = \bigcup_{n=1}^{N_{\text{slits}}} \left(y_n - \frac{w_{\text{slit}}}{2}, y_n + \frac{w_{\text{slit}}}{2}\right),$$

where y_n is spaced evenly around the center with spacing s_{slit} between slits. This gives the potential function

$$\phi(x,y) = \begin{cases} V_0, & (x,y) \in \Omega \cap D_W, \\ 0, & (x,y) \in \Omega \setminus D_W. \end{cases}$$

We should also initialize the wave function to be zero within the wall, but we ignore this since the placement of the initial state is far enough from the wall making it negligible.

B. Discrete problem

1. Spatial Discretization

We now discretize our domain Ω into a grid $\Omega_h \subset \Omega$ defined as

$$\Omega_h = \big\{(x_i, y_j) : x_i = ih,\ y_j = jh,\ i,j \in [\![0, M-1]\!]\big\}.$$

Using this discretization on u, we seek an approximation

$$v_{ij}^n \approx u(x = ih, y = jh, t = n\tau),$$

where τ is the temporal discretization, this will be discussed further in Section II B 2. By the second order Centered Finite Difference in spatial coordinates:

$$\begin{aligned} \mathbf{\Delta} u(x_i, y_j, t_n) &\approx \mathbf{\Delta}_h v_{ij}^n \\ &= \frac{v_{i+1,j}^n - 2v_{ij}^n + v_{i-1,j}^n}{h^2} \\ &+ \frac{v_{i,j+1}^n - 2v_{ij}^n + v_{i,j-1}^n}{h^2}. \end{aligned}$$

This gives us the discrete operator L_h acting on $\mathbf{v}^n$ with matrix elements

$$(L_h \mathbf{v}^n)_{ij} = \mathrm{i}\Big(\mathbf{\Delta}_h v_{ij}^n - \phi_{ij} v_{ij}^n\Big).$$

To avoid storing a matrix for each timestep n, we can instead regard v^n as an M^2-dimensional vector by mapping

$$v_{ij}^n \to v_k^n, \qquad k = iM + j, \quad i,j \in [\![0, M-1]\!],$$

such that

$$\mathbf{v}^n = \begin{pmatrix} v_{11}^n \\ v_{12}^n \\ \vdots \\ v_{1M}^n \\ v_{2M}^n \\ \vdots \\ v_{MM}^n \end{pmatrix},$$

however, we will write v_{ij}^n keeping in mind that this is an element of a column-vector, not a matrix. As the potential $\phi(x,y)$ is known and time-independent we let $\phi : \Omega_h \to \mathbb{R}$ by $\phi(x_i, y_j)$. Explicitly, this is written as:

$$\phi_{ij} \equiv \phi(x_i, y_j) = \begin{cases} V_0, & (x_i, y_j) \in \Omega_h \cap D_W, \\ 0, & (x_i, y_j) \in \Omega_h \setminus D_W. \end{cases}$$

Further, we introduce the discrete inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle_h = \sum_{(x_i, y_j) \in \Omega_h} u_{ij}^* v_{ij},$$

with the associated norm

$$\|\mathbf{u}\|_{2,h}^2 = \langle \mathbf{u}, \mathbf{u} \rangle_h.$$

Note that we omit the standard factor h^2, interpreting $|v_{ij}|^2$ as a probability, not a probability density, we will discuss this further later.

2. Time evolution & Crank–Nicolson

To evolve the wave function we will use the trapezoidal rule:

$$\frac{v_{ij}^{n+1} - v_{ij}^n}{\tau} \approx \frac{1}{2}\left(\mathcal{L} u_{ij}^{n+1} + \mathcal{L} u_{ij}^n\right),$$

in the time interval

$$t_n = n\tau, \quad \tau = \frac{T}{N-1}, \quad n \in [\![0, N-1]\!].$$

For a linear equation

$$u_t = \mathcal{L} u$$

where $\mathcal{L}$ is a general, spatial linear operator. We apply the evolution to get

$$\frac{u^{n+1} - u^n}{\tau} = \frac{1}{2}\left(\mathcal{L} u^{n+1} + \mathcal{L} u^n\right).$$

This can be written as the Crank–Nicolson scheme:

$$\Big(I - \frac{\tau}{2}\mathcal{L}\Big) u^{n+1} = \Big(I + \frac{\tau}{2}\mathcal{L}\Big) u^n,$$

where I is the identity operator.

After discretizing the spatial operator, $\mathcal{L} \to L_h = \mathrm{i}\Big(\mathbf{\Delta}_h - \phi_{ij}\Big)$, and flattening the grid values v_{ij}^n, this becomes

$$A\,\mathbf{v}^{n+1} = B\,\mathbf{v}^n, \tag{1}$$

where

$$A = I - \frac{\tau}{2} L_h, \qquad B = I + \frac{\tau}{2} L_h.$$

Requiring us to solve for

$$\mathbf{v}^{n+1} = A^{-1} B\,\mathbf{v}^n.$$

Writing it out, we get

$$\begin{aligned} v_{ij}^{n+1} &- r\big[v_{i+1,j}^{n+1} - 2v_{ij}^{n+1} + v_{i-1,j}^{n+1}\big] \\ &- r\big[v_{i,j+1}^{n+1} - 2v_{ij}^{n+1} + v_{i,j-1}^{n+1}\big] + \frac{\mathrm{i}\tau}{2}\phi_{ij}v_{ij}^{n+1} \\ &= v_{ij}^{n} + r\big[v_{i+1,j}^{n} - 2v_{ij}^{n} + v_{i-1,j}^{n}\big] \\ &+ r\big[v_{i,j+1}^{n} - 2v_{ij}^{n} + v_{i,j-1}^{n}\big] - \frac{\mathrm{i}\tau}{2}\phi_{ij}v_{ij}^{n}, \end{aligned} \tag{2}$$

where $r \equiv \frac{\mathrm{i}\tau}{2h^2}$. Derivations can be found in Appendix A 1.

We also know that Crank–Nicolson is unconditionally stable for any $h, \tau > 0$ and converges as $h, \tau \to 0$.

Further, letting $S = A^{-1}B$, we can check if the probability is conserved over time. We do this by seeing if S is unitary.

$$S^\dagger = B^\dagger {A^{-1}}^\dagger = (I + \frac{\tau}{2}{L_h}^\dagger)(I - \frac{\tau}{2}L_h^\dagger)^{-1}.$$

As L is the discrete version of the operator $\mathcal{L} = \mathrm{i}\,(\boldsymbol{\Delta} - \phi)$, we see that $L_h^\dagger = -L_h$ (for real potentials ϕ). This gives us

$$\begin{aligned} S^\dagger S &= \left(I - \frac{\tau}{2}L_h\right)\left(I + \frac{\tau}{2}L_h\right)^{-1}\left(I - \frac{\tau}{2}L_h\right)^{-1}\left(I + \frac{\tau}{2}L_h\right) \\ &= I. \end{aligned}$$

meaning that the scheme is L^2-norm preserving and thus probability conserving. So we have the property that

$$\|\mathbf{v}^n\|_{2,h} = \|\mathbf{v}^0\|_{2,h} \quad \forall n \in [\![1, N-1]\!].$$

3. Gauss–Seidel iteration on stencils

As explained, the Crank–Nicolson scheme can be written as

$$A\mathbf{v}^{n+1} = B\mathbf{v}^n,$$

where A and B are large, but sparse matrices. We could assemble them and solve them as a large linear system, but this is unnecessary and will be memory-consuming. For large systems it will also most likely run significantly slower. We instead note that every row of A has the same structure. Each grid point has a stencil of 5 points; itself and its 4 nearest neighbors.

Starting from the fully discretized scheme, we get

$$\begin{aligned} v_{ij}^{n+1} &- r\big[v_{i+1,j}^{n+1} - 2v_{ij}^{n+1} + v_{i-1,j}^{n+1}\big] \\ &- r\big[v_{i,j+1}^{n+1} - 2v_{ij}^{n+1} + v_{i,j-1}^{n+1}\big] + \frac{\mathrm{i}\tau}{2}\phi_{ij}v_{ij}^{n+1} \\ &= \mathrm{RHS}_{ij}^n, \end{aligned}$$

where $\mathbf{RHS}^n$ is a vector with the right-hand side of equation (2). We can write this as

$$\alpha_{ij}\, v_{ij}^{n+1} + \beta\big(v_{i+1,j}^{n+1} + v_{i-1,j}^{n+1} + v_{i,j+1}^{n+1} + v_{i,j-1}^{n+1}\big) = \mathrm{RHS}_{ij}^n,$$

where

$$\alpha_{ij} = 1 + 4r + \frac{\mathrm{i}\tau}{2}\phi_{ij}, \qquad \beta = -r.$$

To solve this we use a red–black Gauss–Seidel iteration. We split $A = D + L + U$ where D is the diagonal and L, U are lower and upper parts. L is not the discrete linear operator L_h here. Each sweep then solves

$$(D + L)\mathbf{v}^{(k+1)} = \mathbf{b} - U\mathbf{v}^{(k)}, \tag{3}$$

where (k) denotes the iteration number, not the timestep. In stencil form this becomes, for each interior point (i, j),

$$v_{ij}^{\text{new}} = \frac{1}{\alpha_{ij}}\Big[\mathrm{RHS}_{ij}^n - \beta\big(v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1}\big)\Big],$$

where each point is taken from the current vector and then overwriting v_{ij} with v_{ij}^{new}.

We can then color the grid into red and black points, like a checkerboard. Then in each iteration we first update all the red points using the black points and the next iteration update all the black points.

We also precompute the coefficients α_{ij} and the constant β as they are not time-dependent. We then compute the right-hand side $\mathrm{RHS}_{ij}^n = B\mathbf{v^n}$ once per timestep. The iterations then sweep over the flattened discrete wave function v_{ij}^n without having to store large matrices.

Rewriting Equation (3), one obtains

$$\mathbf{v}^{k+1} = G\mathbf{v}^k + \mathbf{c}, \qquad G = -(D + L)^{-1}U.$$

We then know, from Golub and Van Loan [3], for the scheme to converge for any initial guess, the spectral radius must be $\rho(G) < 1$, giving the requirement

$$\rho(G) \le \|G\|_\infty \le \max_{i,j}\frac{4|\beta|}{|\alpha_{ij}|} < 1,$$

which comes from the matrix being strictly diagonally dominant. This inequality is satisfied if

$$|\alpha_{ij}| > 4|\beta| \quad \forall (i, j).$$

Since $|\beta| = |r| = \tau/(2h^2)$. In the free region where $\phi_{ij} = 0$, we have

$$\alpha_{ij} = 1 + 4r = 1 + 2\mathrm{i}\frac{\tau}{h^2},$$

so

$$|\alpha_{ij}| = \sqrt{1 + \left(\frac{2\tau}{h^2}\right)^2} > \frac{2\tau}{h^2} = 4|\beta|.$$

Hence the condition $|\alpha_{ij}| > 4|\beta|$ is always satisfied in the free region. Inside the barrier, where $\phi_{ij} \ge 0$, $|\alpha_{ij}|$ only increases further. So it still holds and also converges faster. This means that the Gauss–Seidel also converges unconditionally for our system, and converges faster for higher potential regions. It is also convergent for any

TDSE-scheme with a real potential meaning that our method can be adapted for other *TDSE*-systems.

In our algorithm we calculate for a given timestep n

$$\Delta_{\max}^{(k)} = \max_{i,j} \left| v_{ij}^{(k+1)} - v_{ij}^{(k)} \right|,$$

where k is the iterations. We declare convergence when

$$\Delta_{\max}^{(k)} < \epsilon,$$

where ϵ is our tolerance. We also impose a maximum number of iterations $m_{\max}$ to not have the program run too long if it converges slowly. [1]

4. *Nyquist wave number*

Recalling that the momentum operator acts as the wave number, $p = k$, we can use the Nyquist criterion to see that our grid spacing is large enough. On a grid with spacing h the largest resolvable wave number is the Nyquist wave number

$$k_{\mathrm{N}} = 2\pi \frac{M-1}{2} = \frac{\pi}{h},$$

for our dimensionless domain. So to avoid aliasing we require that the momentum in each direction is $|p| < k_{\mathrm{N}}$. We ensure this holds for all our simulations.

C. Measurement, probability, & normalization

At a given time t the probability of measuring the particle at a given (x, y)-position is determined by Born's rule:

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

We also know that the probability must be normalized. One way to do so, is to look at the probability density and require

$$\|\mathbf{\Psi}\|_2 = \int_\Omega d\mathbf{r}\, p(\mathbf{r}) = 1.$$

However, we have a discrete grid of points. So instead of taking the integral between points to find the probability we look at the probability for the particle to be in a discrete box in our grid. This gives the normalization constraint with respect to the discrete norm $\|\cdot\|_{2,h}$:

$$\|\mathbf{v}^n\|_{2,h} \equiv P(t) = \sum_{(x_i, y_j) \in \Omega_h} p(x_i, y_j; t_n = \tau n) = 1,$$

meaning that each p_{ij}^n is itself a discrete probability of measuring the particle at a discrete point $(x_i, y_j) \in \Omega_h$, and $P(t)$ is the total probability. We will also look at measurements for given x-values. This is as if we placed a detector at a fixed x-value x_D and measured at a given time t_D, using

$$p(y_j \mid x_D, t) = \frac{1}{\xi(x_D, t)} |v(x_D, y_j, t_n)|^2,$$

with normalization factor

$$\xi(x_D, t_n) = \sum_{j=0}^{M-1} \left| v(x_D, y_j, t_n) \right|^2. \tag{4}$$

Giving

$$\sum_{y_j \in \Omega_h} p(y_j \mid x = x_D; t_n = t_D) = 1.$$

Compared to real-world experiments, this corresponds to sending a single particle through the barrier multiple times and detecting it on a screen placed at x_D. Over time the particles build up a histogram along y.

1. *Interference*

Interference is a phenomenon where two or more coherent waves interact with each other causing a larger amplitude (constructive interference) or a decreased amplitude (destructive interference). This comes from a superposition of the waves at a given point.

For a single slit with aperture ω_{slit} we expect Fraunhofer diffraction with destructive interference occurring at

$$\omega_{\text{slit}} \sin\theta = \ell\lambda, \qquad \ell \in \mathbb{Z} \setminus \{0\}$$

where ℓ labels the diffraction minima and λ the wavelength [4]. If $\lambda/\omega_{\text{slit}} \approx 1$ the first minimum occurs on $\theta_1 \approx 90°$. In this case the transmitted wave acts like an approximate point source producing a circular wavefront. There would be no interference, and at the detection screen the wavefront collapses to a Gaussian shape.

By adding slits with spacing s_{slit}, each slit emits a coherent circular wavefront and these interfere. In the Fraunhofer limit this leads to standard multi-slit diffraction with intensity

$$I(\theta) \propto \left[\frac{\sin N_{\text{slits}}\gamma/2}{\sin\gamma/2}\right]^2,$$

where

$$\gamma = 2\pi s_{\text{slit}} \sin\theta/\lambda$$

is the phase difference between two neighbor slits, and N_{slits} is the number of slits. The principal maxima occur when

$$s_{\text{slit}} \sin\theta = \ell\lambda, \qquad \ell \in \mathbb{Z}, \tag{5}$$

[1] For further reading on solvers and convergence, read our report under `/Rapports/Rapport2` at our Github repository [a].

which means peak positions and pacing between are determined by slit spacing and wavelength, not N_{slits}. Increasing the number of slits creates more interfering path with phase difference γ to intersect, increasing the number of peaks. In addition, the intersecting phase differences makes the peaks sharper, and the minima broader.

Fraunhofer diffraction requires the detection screen to be in the far field, which is when the circular wavefront becomes approximately plane waves. The distance from the wall to the detection screen is $L_D = x_D - x_{\text{wall}}$. To observe Fraunhofer diffraction $\omega_{\text{slit}}^2/\lambda \ll L_D$ must be satisfied. In our simulation this is not quite the case, thus we expect mainly qualitative agreements.

D. Simulation

As mentioned, our algorithm to solve the system will be a Gauss–Seidel iterative solver. After precomputing the coefficients β, it will for each timestep assemble the right-hand side $\mathbf{RHS}^n$ and find an approximate solution. This is described in Algorithm 1.

Algorithm 1 Red–black Gauss–Seidel solver: $\mathbf{v}$ is the solution vector, **RHS** the right-hand side, $\mathbf{D}^{-1}$ the precomputed inverse diagonal, M the grid size, m the iteration counter, and ϵ the tolerance.

```
Precomputed:
**RHS**
$\beta = -\mathrm{i}\left(\frac{\tau}{2h^2}\right)$
$\mathbf{D}^{-1} = \mathbf{invD}$

Tolerance $\epsilon$ and a maximum number of iterations $m_{\max}$
Let $\mathbf{v}^0 = \mathbf{v}_{\text{curr}}$          ▷ Use current solution as initial guess
Let $\mathbf{v} = \mathbf{v}^0$          ▷ In-place updates (Gauss–Seidel)
Set $m = 0$          ▷ Iteration counter
while $m < m_{\max}$ do
    Let $\Delta_{\max} = 0$ ▷ Max pointwise update in this iteration

    for $c \in \{0, 1\}$ do          ▷ Red–black color sweep
        For all interior points $i, j \in [\![1, M-2]\!]$:
        If $(i + j) \bmod 2 \neq c$: continue          ▷ Skip opposite color
        $k = \mathrm{idx}(i, j)$          ▷ Flat index
        From latest state, compute sum of neighbors
            $\mathrm{sumN} = v\big(\mathrm{idx}(i+1, j)\big) + v\big(\mathrm{idx}(i-1, j)\big) + v\big(\mathrm{idx}(i, j+1)\big) + v\big(\mathrm{idx}(i, j-1)\big)$
        Update value at $k$ using Gauss–Seidel formula
            $v_k^{\text{new}} = D_k^{-1}\left(\mathrm{RHS}_k - \beta\,\mathrm{sumN}\right)$
        Compute pointwise change
            $\delta_k = \left|v_k^{\text{new}} - v_k\right|$
        If $\delta_k > \Delta_{\max}$ set $\Delta_{\max} = \delta_k$
        Overwrite current value
            $v_k \leftarrow v_k^{\text{new}}$

    if $\Delta_{\max} < \epsilon$ then
        break          ▷ Converged
    else
        $m \leftarrow m + 1$
```

1. *Theoretical memory saving and speedup*

We will not test directly the speedup nor the memory saving of our method compared to large matrix solvers. This is as there is no single alternative algorithm. However, for a full matrix system, one would have to construct two matrices A, B of size $|A| = |B| = \left((M-2)^2\right)^2$ as well as two state arrays of size $|v^n| = |v^{n+1}| = (M-2)^2$. In contrast, from Algorithm 1, one sees that we store the arrays of size $|\mathbf{RHS}^n| = |\mathbf{D}^{-1}| = |v^n| = |v^{n+1}| = (M-2)^2$. Assuming that each element occupies the same storage of one unit 1 u, the total memory requirement for a full matrix method is

$$|full = 2(M-2)^4 + 2(M-2)^2\,\mathrm{u},$$

whereas the red–black Gauss–Seidel implementation requires

$$M_{\mathrm{rb}} = 4(M-2)^2\,\mathrm{u}.$$

The ratio of memory usage is therefore

$$\frac{M_{\mathrm{full}}}{M_{\mathrm{rb}}} = \frac{2(M-2)^4 + 2(M-2)^2}{4(M-2)^2} = \frac{1}{2}(M-2)^2 + \frac{1}{2},$$

which grows as $\mathcal{O}(M^2)$. However, this is a naive approach and realistically the full matrix solver would utilize a sparse matrix method as `arma::sp_cx_mat`. This requires memory proportional to the number of non-zero entries. For a five-point stencil, this is $\approx 5(M-2)^2$ for both A, B, so,

$$M_{\mathrm{sp}} \approx \big(2\cdot 5 + 2\big)(M-2)^2\,\mathrm{u} = 12(M-2)^2\,\mathrm{u},$$

and the memory ratio becomes

$$\frac{M_{\mathrm{sp}}}{M_{\mathrm{rb}}} \approx \frac{12}{4} = 3.$$

There are also multiple solvers to solve the system. One could theoretically solve the system exactly, but this would be very slow for large systems, even though it would give more accurate solutions. Solving an `arma::sp_cx_mat` is also more tedious as we need to know which elements to choose and we need to pull from rows and columns which would most likely be slower than our array scheme.

Thus, without direct comparison, we can conclude that our method is an improvement in both memory and speed from the pure matrix [2].

E. Other tools

For plotting we used the Python library `matplotlib` [5], and `numpy` [6] for general calculations.

For parallelizing the Gauss–Seidel solver we used OpenMP [7].

For reading, writing, and parsing configuration and results files we used the `nlohmann::json` C++ library [8].

We also used OpenAI's ChatGPT as declared in Appendix B. This was primarily to reduce tedious tasks and polish language and structure.

[2] There exist alternative iterative solvers to our red–black Gauss–Seidel scheme which should converge faster. Gauss–Seidel is however easy to implement. We tested a Jacobi method, which in our experiments was slightly slower.

III. RESULTS AND DISCUSSION

TABLE II: Solver parameters.

Quantity	Value
Max iterations $m_{\max}$	2000
Tolerance ϵ	10^{-10}

Unless otherwise specified, we will use the parameters from Table II for our Gauss–Seidel solver.

A. Conservation of probability

We study how the simulated total probability $P(t) = \|\mathbf{v}^n\|_{2,h}$ evolves over time for free propagation and a double-slit potential. We are comparing $P(t)$ against the expected value of 1 for each timestep. We want to find the deviation because, in practice we compute $A\mathbf{v}^{n+1} = B\mathbf{v}^n$ approximately with red–black–Gauss–Seidel. The tolerance ϵ is not zero, meaning the algorithm runs until $\Delta_{max}^{(k)} < \epsilon$ or reaches m_{max}. This makes the solution not exact.

1. *Free propagation*

For the free propagation we will use the simulation parameters described in Table III.

TABLE III: Grid and simulation parameters for the free propagation.

Quantity	Value
Grid points per dimension M	200
Total simulation time T	0.008
Number of time points N	321
timestep τ	$2.5 \cdot 10^{-5}$
Initial packet center in x, x_c	0.25
Initial width in x, σ_x	0.05
Initial momentum in x-direction, p_x	200
Initial packet center in y, y_c	0.5
Initial width in y, σ_y	0.05
Initial momentum in y-direction, p_y	0
Potential height V_0	0

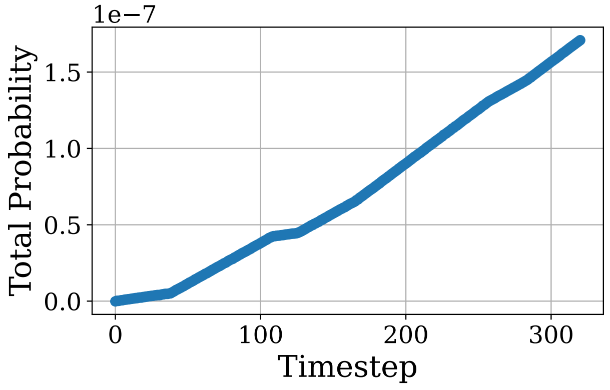

FIG. 2: Deviation of the total probability $P(t) - P(0)$ as a function of timestep for the free propagation.

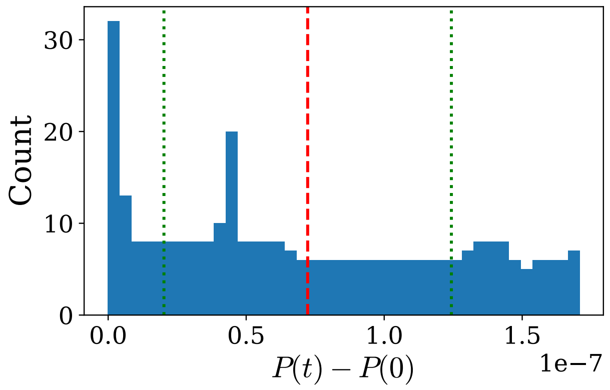

FIG. 3: Histogram of the probability deviations $P(t) - 1.0$ for the free propagation. Mean deviation $\langle P(t) - 1.0 \rangle \approx 7.23 \cdot 10^{-8}$ and standard deviation $\sigma \approx 5.21 \cdot 10^{-8}$.

Figure 2 shows the deviation $P(t) - 1.0$ for free propagation $V_0 = 0$. The deviation oscillates around a small negative number with mean $\langle P(t) - 1.0 \rangle \approx 7.23 \cdot 10^{-8}$ and standard deviation $\sigma \approx 5.21 \cdot 10^{-8}$, as seen in the histogram in Figure 3. From the figures, we see the maximum deviation is of order $\mathcal{O}(10^{-7})$, not too far from the tolerance $\epsilon = 10^{-10}$. We also observe a slight drift, however, as we are mainly interested in qualitative results this is not alarming. If it were crucial to have a higher accuracy the tolerance ϵ could be adjusted as needed. We could also study the convergence rate of the scheme to determine ϵ based on our desired accuracy.

2. Double-slit

For the double-slit, we use the same parameters as in Table III, but we now let $\sigma_y = 0.1$ and $V_0 = 10^{10}$.

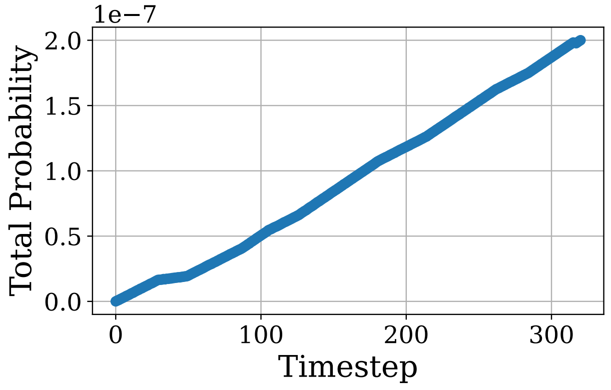

FIG. 4: Deviation of the total probability $P(t) - 1.0$ as a function of timestep for the double-slit simulation.

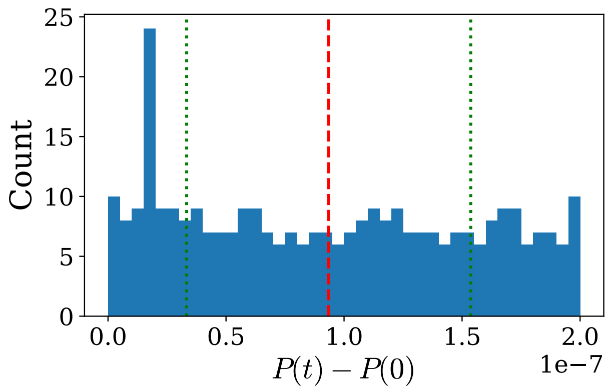

FIG. 5: Histogram of the probability deviations $P(t) - P(0)$ for the double-slit simulation. Mean deviation $\langle P(t) - P(0) \rangle \approx 9.35 \cdot 10^{-8}$ and standard deviation $\sigma \approx 6.02 \cdot 10^{-8}$.

For the double-slit in Figure 5 the highest absolute deviating values $P(t) - 1$ also stays within $\mathcal{O}(10^{-7})$. In Figure 4 the double-slit deviation increase slightly more than the free propagation within our time interval. We have not tested if this is statistically significant, but we assume it is caused by the introduced double barrier potential. This is as expected as $\alpha_{ij} = 1 + 4r + \frac{i\tau}{2}\phi_{ij}$. And with our convergence criterion $4|\beta|/|\alpha_{ij}| < 1$ no potential means a longer convergence time. So the without potential, the algorithm might reach max iterations before converging, meaning that we might need to change maximum number of iterations.

In both cases Crank–Nicolson with Gauss–Seidel has shown stability within reason. Choosing a lower tolerance ϵ and increasing the maximum Gauss–Seidel iterations $m_{max} = 2000$ could however improve the results. But choosing $\epsilon = 10^{-10}$ gave a balanced compromise between efficiency and precision. It is also worth noting that this is not the deviation of the exact wave function

u, but rather whether the probability is conserved.

One could also compare the conservation against other solvers. For instance an exact solver's inaccuracy is determined by floating point errors. We could also compare runtimes and memory usage for other schemes to test our hypothesis of a $\gtrsim 3\times$ speedup. We could also test for very large grids and consider GPU-parallelization if we need to store large arrays.

B. Wave propagation through double-slit

We now look at how the probability propagates through the double-slit and how the real and imaginary parts of the wave evolve. We used the parameters in Table IV and took snapshots at times $t_n = \{0,\, 0.001,\, 0.002\}$ which corresponds to total timesteps $N = \{0, 41, 81\}$. The plots display the square root of our values to avoid dominancy by high probability regions. So instead of plotting the probability distribution, we plot $|v_{ij}^n| = \sqrt{|v_{ij}^n|^2}$. In the `README.md` of the GitHub repository [a] there is an animation of the full propagation $|v_{ij}^n|^2$ for an extended time period.

TABLE IV: Grid, simulation and slit parameters.

Quantity	**Value**
Grid points per dimension M	200
Number of time points N	81
timestep τ	$2.5 \cdot 10^{-5}$
Initial packet center in x, x_c	0.25
Initial width in x, σ_x	0.05
Initial momentum in x-direction, p_x	200
Initial packet center in y, y_c	0.5
Initial momentum in y-direction, p_y	0
Initial width in y, σ_y	0.2
Potential height V_0	10^{10}
Wall center position x_{wall}	0.5
Wall thickness d_{wall}	0.02
Slit aperture w_{slit}	0.05
Number of slits N_{slits}	2
Slit spacing s_{slit}	0.05

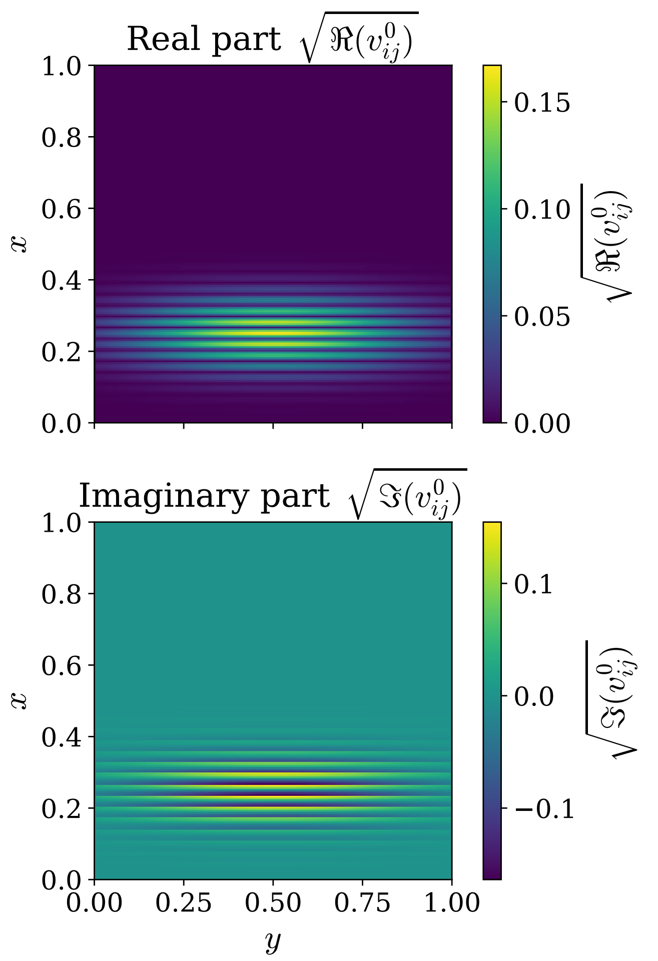

FIG. 6: The square root of the real and imaginary parts of the initial wave function v_{ij}^0. The upper figure displays the square root of the real part, $\sqrt{\mathrm{Re}(v_{ij}^0)}$, and the lower the imaginary $\sqrt{\mathrm{Im}(v_{ij}^0)}$. The colors display intensity.

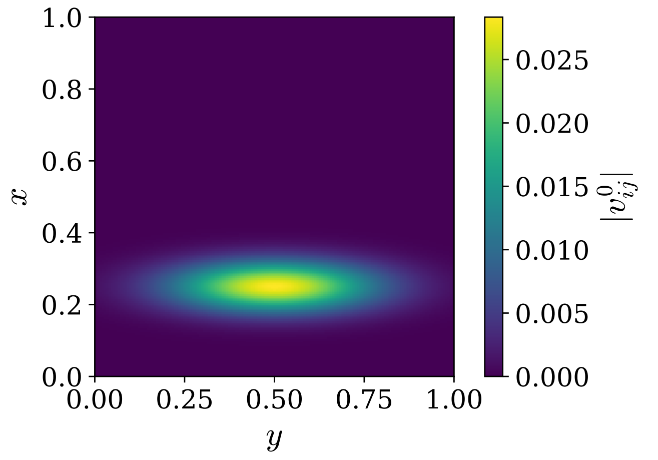

FIG. 7: Initial probability function $|v_{ij}^0|$ corresponding to the wave packet in Figure 6.

From Figure 7, we see that the wave packet's spatial spread is as we intended. It is broader in the y-direction since $\sigma_y > \sigma_x$, and it is located beneath the slits with center $(0.25, 0.5)$ as listed in Table IV. Figure 6 shows the initial wave packet is approximately a plane wave in the x-direction.

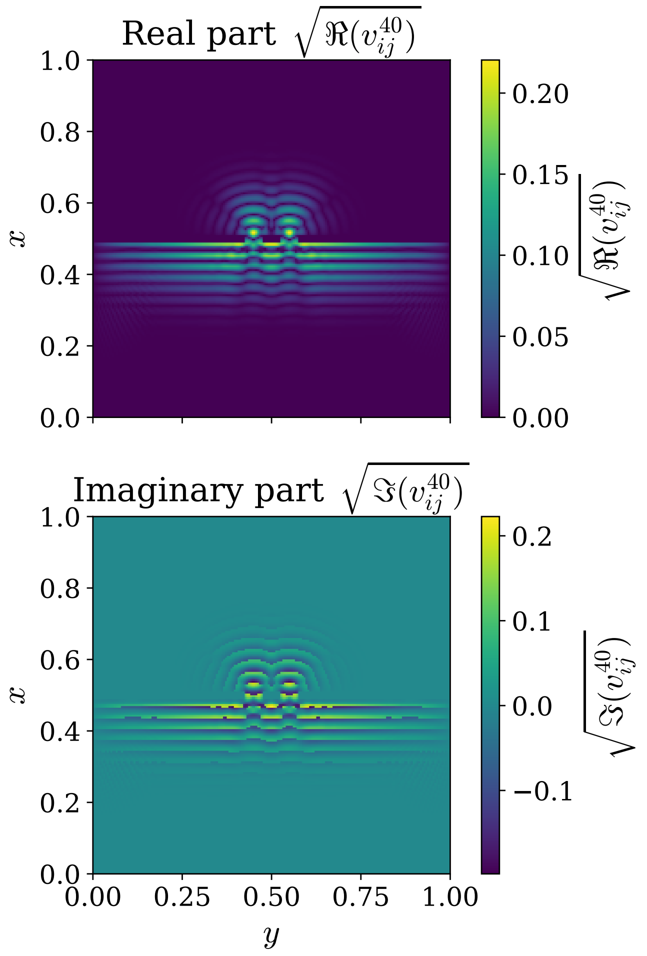

FIG. 8: The square root of the real and imaginary parts of the wave function v_{ij}^{40} at $n = 40 \implies t_n = 0.001$ after a short time evolution with the double-slit potential. The upper figure displays the square root of the real part, $\sqrt{\text{Re}(v_{ij}^{40})}$, and the lower the imaginary $\sqrt{\text{Im}(v_{ij}^{40})}$. The colors display intensity.

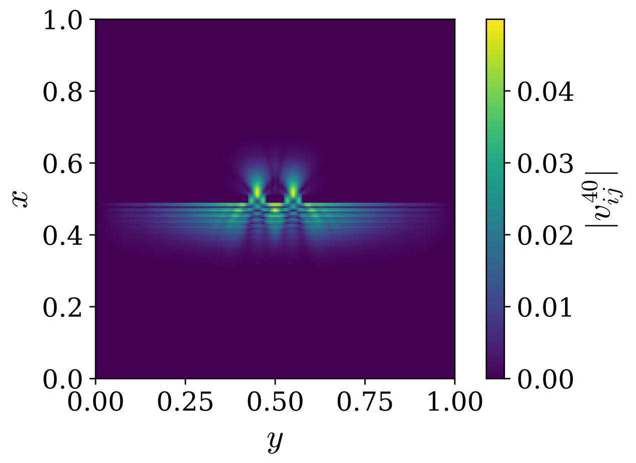

FIG. 9: Probability function $|v_{ij}^{40}|$ at $n = 40 \implies t_n = 0.001$ corresponding to Figure 8.

As the wave packet crosses the slits, we observe both transmission and reflection, as illustrated in Figure 8. To better see the wave behavior of the wave packet, we must look at the real and imaginary part of it in Figure 9. Further away from the slits, the reflected wave resembles a plane wave. The high potential acts like an infinite barrier, and the Dirichlet boundary condition leads to near total reflection for the part of the packet that does not crosses the slit. Closer to the wall, the wavefront is distorted by circular waves, creating regions of destructive interference. The parts of the wave packet that pass through the potential barrier act like two coherent circular waves in both the real and imaginary plane.

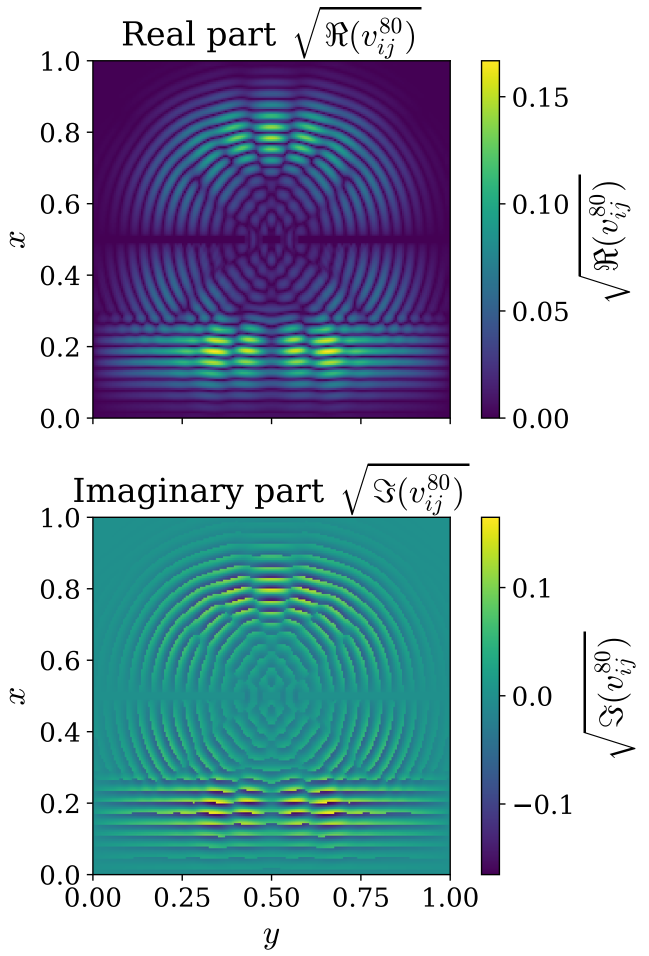

FIG. 10: The square root of the real and imaginary parts of the wave function v_{ij}^{80} at $n = 80 \implies t_n = 0.002$ at the final simulation time T. The upper figure displays the square root of the real part, $\sqrt{\mathrm{Re}(v_{ij}^{80})}$, and the lower the imaginary $\sqrt{\mathrm{Im}(v_{ij}^{80})}$. The colors display intensity.

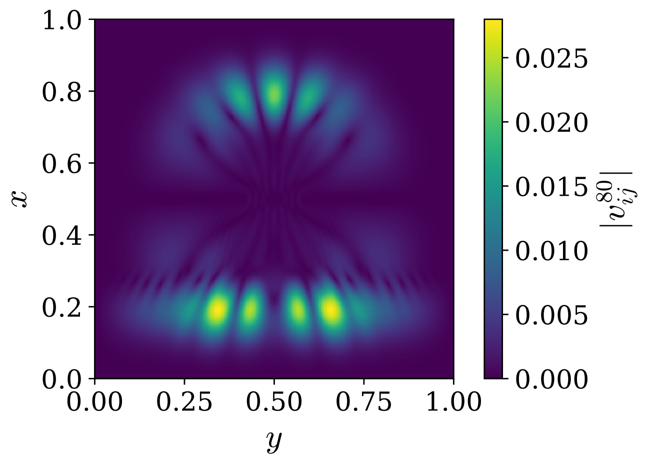

FIG. 11: Probability function $|v_{ij}^{80}|$ at $n = 80 \implies t_n = 0.002$ corresponding to Figure 10, showing the distribution after passing the double-slit.

After propagating to $t_n = 0.002$, the probability function in Figure 11 is developed into two interference patterns on either side of the barrier. On the upper side, the pattern is more circular and intense towards the center. The reflected side is more elongated, and forms two individual patterns pointing back to their respective slit. Each of the patterns has its maxima, because each slit acts as an individual transmitting source. This makes the reflected wave a superposition of transmitted and reflected waves interfering. The wave behavior more visible in the real and imaginary parts of Figure 10. Overall, the simulations follow the predicted behavior of the superposition principle.

1. *Single slit propagation*

We also perform the same simulations for a single slit setup. This will be discussed further later, but we present here the propagation through space.

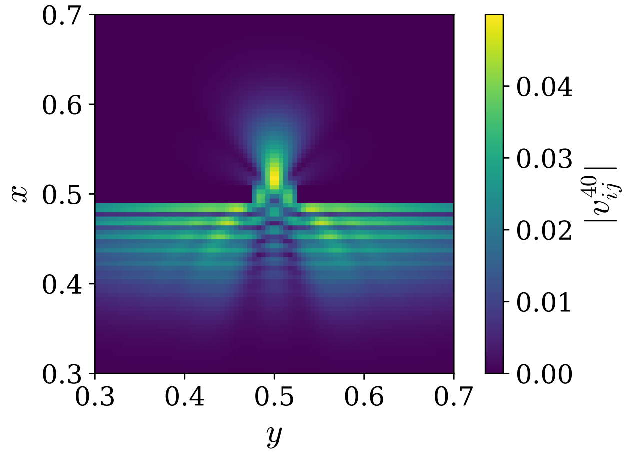

FIG. 12: Probability function $|v_{ij}^{40}|$ for single slit at $n = 40 \implies t_n = 0.001$ showing the as it passes through the slit. Zoomed in to better visualize over the slit opening.

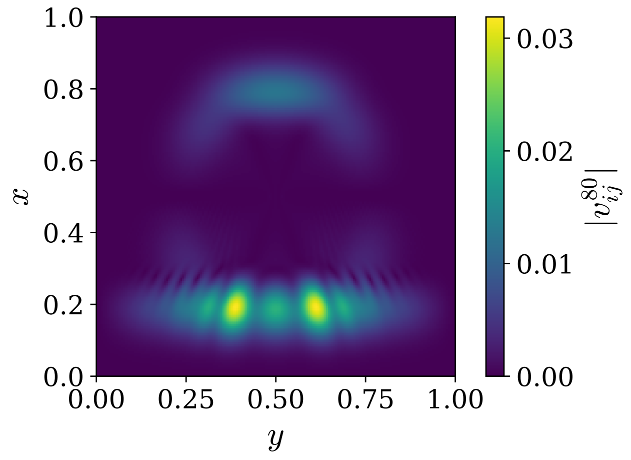

FIG. 13: Probability function $|v_{ij}^{80}|$ for single slit at $n = 80 \implies t_n = 0.002$ showing the distribution after it passed the slit.

C. Detection

We introduce a detector at $x_D = 0.8$ and measure $p(y \mid x = x_D; t_n = t_D)$ at fixed time $t_n = 0.002$. The simulations are run with the same parameters as in Table IV except from varying N_{slits}. The conditional probability distribution was normalized to one using $\xi(x_D, t_n)$ from Equation (4). This would represent running multiple real-world experiments for a single particle and constructing a histogram of the detections.

1. *Single slit*

We start by simulating a single slit.

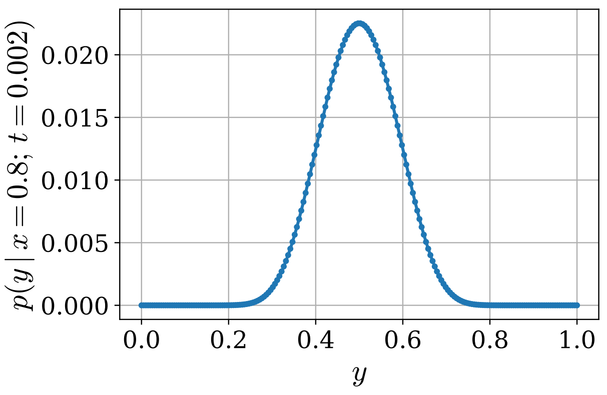

FIG. 14: Probability function for single slit at time $t_n = 0.002$, with a detector at $x_D = 0.8$.

With $p_x = 200$, the dimensionless wavelength is

$$\lambda = \frac{2\pi}{p_x} \approx 0.031,$$

while the slit aperture is $w_{\text{slit}} = 0.05$. So $\lambda/w_{\text{slit}} \approx 0.63$, so the first Fraunhofer minimum for an ideal single slit,

$$w_{\text{slit}} \sin\theta_1 = \lambda,$$

occurs at $\theta \approx 39°$. At our detector position $L_D = x_D - x_{\text{wall}} = 0.3$ this corresponds to

$$\Delta y \simeq L_D \tan\theta_1 \approx 0.24,$$

However, for the Fraunhofer regime, we require that

$$L_D \gg \frac{w_{\text{slit}}^2}{\lambda} \approx 0.08.$$

As our slit distance $L_D = 0.3$, this might not hold, which means we might not see what we expect.

The shape of the probability distribution in Figure 14 is Gaussian, which gives the impression of no interference. Since $\lambda/w_{\text{slit}} \neq 1$, the wavefront crossing the barrier cannot be perfectly transmitted circular waves, with first minima $\theta_1 = 90°$. By looking at the transmitted waves in Figure 12, there is a dimmer region on each side corresponding to an angle around 40°. In addition, Figure 13 is captured at the time $t_n = 0.002$ along the y-screen. A measurement at $x_D = 0.8$ captures the probability distribution without letting it spread out evenly to reach the Fraunhofer regime. We are instead capturing the tip of a curved wavefront, displaying a Gaussian wavefront.

2. *Double-slit*

We now look at the same double-slit experiment as we did before.

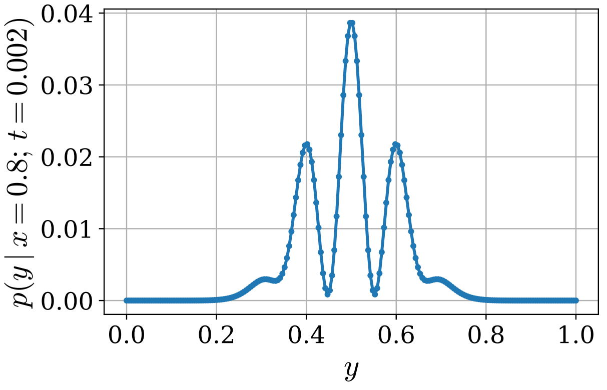

FIG. 15: As figure 14 with two slits.

For the double-slit in Figure 15 we clearly see interference patterns with a dominant central maximum, symmetric side lobes and minima. However, the structure differs from Fraunhofer's prediction in Equation (5). The first maxima appear to be more centralized than what is expected from Fraunhofer diffraction. Since we are in the near-field, Fraunhofer condition is not truly fulfilled, which is consistent with the curved wavefront in Figure 10. The curvature of the wavefront means the phase difference γ between the slits does not grow linearly along y at the detection line.

Nevertheless, the qualitative features of the double-slit pattern is present. A dominant central peak with symmetric minima and side lobes. They arise from the coherent superposition of two circular transmitted waves in the near-field.

3. Three slits

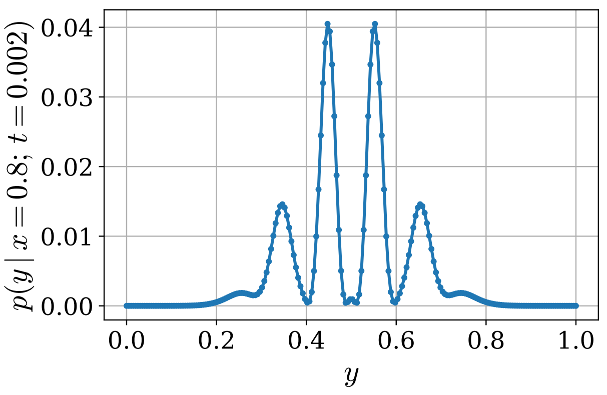

FIG. 16: As figure 14 with three slits.

4. Eight slits

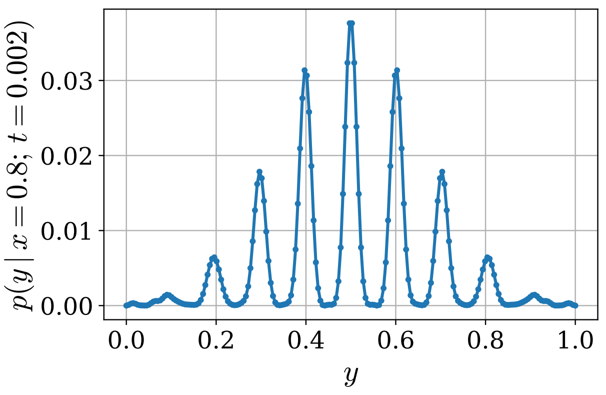

FIG. 17: As figure 14 with eight slits.

For three and eight slits, in Figure 16 and 17 the spacing between the peaks are approximately the same as for the double-slit. This is because the phase difference γ between the slits stays constant as N_{slits} increases. The number of peaks increase, as mentioned in section II C 1, because there are more sources emitting interference with the same phase difference. This makes them narrower with more spacing.

Another way of understanding sharper peaks, is the fact that the conditional probability $p(y \mid x_D, t_n)$ is normalized to one. To preserve the same area under the graph, increasing the number of peaks, forces them to become taller and narrower.

Overall, this indicates that Fraunhofer diffraction can give a qualitative understanding of the probability distribution, even though we are in the near-field.

IV. CONCLUSION

We have developed a matrix-free Crank–Nicolson scheme for the two-dimensional time-dependent Schrödinger equation. It is solved by a red–black Gauss–Seidel iterator over arrays instead of matrices, which saves memory usage and is designed to be more efficient, especially for larger grids. The scheme is stable and conserves the probability within $\mathcal{O}(10^{-7})$ for our simulation parameters.

The simulations reproduce expected qualitative behaviors observed for double-slit experiments. For a single slit we obtain a Gaussian detection profile indicating measurement in the near-field. Increasing the number of slits leads to interference with sharper patterns for larger number of slits.

Our scheme could be further developed to study quantitative behavior. By increasing the grid, adding more timesteps and reducing the tolerance of the solver one could more accurately represent the physical system. This is an advantage of our Gauss–Seidel non-matrix algorithm. In future work one could compare this to alternative solutions to determine whether it is ideal. One could also study the convergence of Gauss–Seidel for higher iterations. It is, however, a memory-efficient and scalable scheme for the TDSE in a slit potential. From unconditional convergence for any real potential, our system is a good framework that can be built upon for further studies of quantum mechanics, both qualitatively and quantitatively.

[1] Wikipedia contributors. Double-slit experiment. https://en.wikipedia.org/wiki/Double-slit_experiment, 2025. Accessed: 2025-12-09.

[2] Anders Kvellestad. Fys3150 project 5. https://anderkve.github.io/FYS3150/book/projects/project5.html, 2025. Accessed: 2025-11-20.

[3] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 3 edition, 1996.

[4] Arnt Inge Vistnes. *Physics of Oscillations and Waves.* Springer, 2 edition, 2018.

[5] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[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*, 585:357–362, 2020.

[7] OpenMP Architecture Review Board. *OpenMP Application Programming Interface*, version 5.2 edition, 2021. Accessed: 2025-10-14.

[8] Niels Lohmann. Json for modern c++. https://github.com/nlohmann/json. MIT License. Accessed: 2025-10-23.

Appendix A: Derivations

1. Crank–Nicolson Scheme

By substituting A and B into Equation (1) we get

$$\big(I - \tfrac{\tau}{2} L_h\big)\mathbf{v}^{n+1} = \big(I + \tfrac{\tau}{2} L_h\big)\mathbf{v}^{n},$$

where $L_h = \mathrm{i}\big(\boldsymbol{\Delta}_h - \phi_{ij}\big)$. This gives

$$v_{ij}^{n+1} - \frac{\mathrm{i}\tau}{2}\Big(\boldsymbol{\Delta}_h v_{ij}^{n+1} - \phi_{ij} v_{ij}^{n+1}\Big) = v_{ij}^{n} + \frac{\mathrm{i}\tau}{2}\Big(\boldsymbol{\Delta}_h v_{ij}^{n} - \phi_{ij} v_{ij}^{n}\Big).$$

Using the centered finite-difference approximation for the Laplacian,

$$\boldsymbol{\Delta}_h v_{ij}^{n} = \frac{v_{i+1,j}^{n} - 2v_{ij}^{n} + v_{i-1,j}^{n}}{h^2} + \frac{v_{i,j+1}^{n} - 2v_{ij}^{n} + v_{i,j-1}^{n}}{h^2},$$

we obtain

$$\begin{aligned} v_{ij}^{n+1} &- r\big[v_{i+1,j}^{n+1} - 2v_{ij}^{n+1} + v_{i-1,j}^{n+1}\big] \\ &- r\big[v_{i,j+1}^{n+1} - 2v_{ij}^{n+1} + v_{i,j-1}^{n+1}\big] + \frac{\mathrm{i}\tau}{2}\phi_{ij} v_{ij}^{n+1} \\ &= v_{ij}^{n} + r\big[v_{i+1,j}^{n} - 2v_{ij}^{n} + v_{i-1,j}^{n}\big] \\ &\quad + r\big[v_{i,j+1}^{n} - 2v_{ij}^{n} + v_{i,j-1}^{n}\big] - \frac{\mathrm{i}\tau}{2}\phi_{ij} v_{ij}^{n}, \end{aligned}$$

where $r \equiv \frac{\mathrm{i}\tau}{2h^2}$. This is the Crank–Nicolson stencil.

Appendix B: Declaration of Use of Generative AI

In this scientific work, generative artificial intelligence (AI) has been used. All data and personal information have been processed in accordance with the University of Oslo's regulations, and we, as the authors of the document, take full responsibility for its content, claims, and references. An overview of the use of generative AI is provided below.

Summary

- **Tool(s) used:** OpenAI ChatGPT (GPT-5), https://chatgpt.com/
- **Use:**

- Generating boilerplate code like plotting with `matplotlib`.
- Generating Makefiles.
- Formatting README.md.
- Checking language for clarity and grammar and general proofreading.
- Generating brief code documentation, comments, and variable names for better readability in adherence to conventions.
- Creating Tables in proper TEX-format.
- Brainstorming ideas for optimizing code for efficiency.
- Suggesting unused, fitting variable symbols.

— The end —