Quantum Mechanical Scattering at Arbitrary Potentials in 1 and 2 Dimensions by Numerically Solving the Lippmann-Schwinger Equation

by

Dominic Hirtler

Bachelor thesis

A thesis submitted in partial fulfillment for the degree of Bachelor Of Science

Supervisor: Assoc.-Prof. Dr. Peter Puschnig

University of Graz

Institute of Physics

Department of Theoretical Physics

Graz, July 2019

Quantum Mechanical Scattering at Arbitrary Potentials

Abstract

In quantum mechanics, a particle is defined by its wave function. The Schrödinger equation describes, how this wave function behaves under the influence of a potential energy. Here, one distinguishes between bound solutions which lead to discrete energy levels and unbound, scattering solutions which generally exhibit a continuous energy spectrum. For the latter case, if one wants to know the state of such a particle, one needs to solve the Schrödinger equation for the given potential. For non-trivial external potentials, this proves quite difficult. However, this equation can be rewritten into an integral equation — the so called Lippmann-Schwinger equation — and then solved numerically to get an approximate solution. In this thesis, one such implementation gets discussed in the 1-dimensional and 2-dimensional case for a few chosen potentials.

Contents

1 Introduction and Derivation of Lippmann-Schwinger Equation

In quantum mechanics, a particle is defined by its state, which is described as the ket-vector $|\psi\rangle$ in Hilbert-space. Multiplying from the left with the bra-vector $\langle \boldsymbol{r}|$, gives the wave function corresponding to this state $|\psi\rangle$ in position space:

$$\langle \boldsymbol{r}|\psi\rangle = \psi(\boldsymbol{r}) \tag{1}$$

The stationary behavior of the state $|\psi\rangle$ in some potential $V = V(\boldsymbol{r})$ is described by the time-independent Schrödinger equation

$$\hat{H}\,|\psi\rangle = E\,|\psi\rangle, \tag{2}$$

with the Hamiltonian $\hat{H} = \hat{H}_0 + V(\boldsymbol{r})$, where $\hat{H}_0$ is the Hamiltonian of a free particle. This equation can be rearranged to assume the following form:

$$(E - \hat{H}_0)\,|\psi\rangle = V\,|\psi\rangle. \tag{3}$$

By splitting the state $|\psi\rangle$ into an eigenstate of the free Hamiltonian $|\phi\rangle$ and a scattered state $|\psi_s\rangle$ via $|\psi\rangle = |\phi\rangle + |\psi_s\rangle$, and inserting this into equation 3, one obtains

$$(E - \hat{H}_0)\,|\psi_s\rangle = V\,|\psi\rangle. \tag{4}$$

Here the property that $|\phi\rangle$ is an eigenstate of $\hat{H}_0$, and therefore satisfies the free particle Schrödinger equation $(E - \hat{H}_0)\,|\phi\rangle = 0$, has been used.

Multiplying both sides from the left with $(E - \hat{H}_0)^{-1}$ leads to

$$|\psi_s\rangle = (E - \hat{H}_0)^{-1}\,V\,|\psi\rangle, \tag{5}$$

where $|\psi_s\rangle = |\psi\rangle - |\phi\rangle$ can be used to get

$$|\psi\rangle = |\phi\rangle + (E - \hat{H}_0)^{-1} V |\psi\rangle, \tag{6}$$

which is known as the Lippmann-Schwinger equation. However, as E is an eigenvalue of $\hat{H}_0$, $(E - \hat{H}_0)$ is singular and therefore not invertable. To circumvent this, the operator gets a slight complex offset, which will approach 0 in the end. Therefore, a mathematically more consistent version of equation 6 would be

$$|\psi\rangle = |\phi\rangle + \lim_{\varepsilon \to 0} (E - \hat{H}_0 + i\varepsilon)^{-1} V |\psi\rangle. \tag{7}$$

This form of the Lippmann-Schwinger equation uses kets to represent the quantum mechanical states. However, it is a lot more intuitive to choose a basis and represent the state in it. For this thesis, position representation has been chosen. To transform equation 7, one firstly inserts a full basis of position eigenstates:

$$|\psi\rangle = |\phi\rangle + \int \mathrm{d}^n r' \, |\boldsymbol{r'}\rangle\langle\boldsymbol{r'}| \lim_{\varepsilon \to 0} (E - \hat{H}_0 + i\varepsilon)^{-1} V |\psi\rangle. \tag{8}$$

Multiplying this equation from the left with position eigenstates $\langle \boldsymbol{r}|$, one obtains

$$\langle \boldsymbol{r}|\psi\rangle = \langle \boldsymbol{r}|\phi\rangle + \int \mathrm{d}^n r' \, \langle \boldsymbol{r}|\boldsymbol{r'}\rangle \lim_{\varepsilon \to 0} (E - \hat{H}_0 + i\varepsilon)^{-1} V \langle \boldsymbol{r'}|\psi\rangle. \tag{9}$$

Using relation 1 and the fact that $\langle \boldsymbol{r'}|\boldsymbol{r}\rangle = \delta(\boldsymbol{r} - \boldsymbol{r'})$, where δ is the Dirac delta function, equation 9 assumes its position representation:

$$\psi(\boldsymbol{r}) = \phi(\boldsymbol{r}) + \int \mathrm{d}^n r' \, \delta(\boldsymbol{r} - \boldsymbol{r'}) \lim_{\varepsilon \to 0} (E - \hat{H}_0 + i\varepsilon)^{-1} V \psi(\boldsymbol{r'}). \tag{10}$$

Defining the Green's function $G(\boldsymbol{r})$ as

$$\lim_{\varepsilon \to 0} (E - \hat{H}_0 + i\varepsilon)\, G(\boldsymbol{r} - \boldsymbol{r'}) = \delta(\boldsymbol{r} - \boldsymbol{r'}), \tag{11}$$

one can substitute G into equation 10 to get the integral form of the Lippmann-Schwinger equation:

$$\psi(\boldsymbol{r}) = \phi(\boldsymbol{r}) + \int \mathrm{d}^n r'\, G(\boldsymbol{r} - \boldsymbol{r'})\, V(\boldsymbol{r'})\, \psi(\boldsymbol{r'}) \tag{12}$$

For this integral to converge, $V(\boldsymbol{r'})$ needs to approach 0 outside a finite area fast enough, such that

$$\lim_{|\boldsymbol{r'}| \to \infty} |\boldsymbol{r'}|\ V(\boldsymbol{r'}) = 0. \tag{13}$$

2 1-Dimensional Case

In the 1-dimensional analysis, the Lippmann-Schwinger equation only depends on a scalar position r, which must not be confused with the absolute value of the vector $\boldsymbol{r}$. This means, that the generally 3-dimensional integral transforms into a 1-dimensional one.

$$\psi(r) = \phi(r) + \int \mathrm{d}r'\, G(r, r')\, V(r)\, \psi(r') \tag{14}$$

For a single dimension, the free wave function $\phi(r)$ — meaning the eigenfunction to the free Hamilton operator in 1 dimension — and the corresponding energy eigenvalue E are given by

$$\phi(r) = e^{ikr} \qquad \text{and} \qquad E = \frac{k^2}{2}. \tag{15}$$

Here, as well as the rest of this thesis, natural units are used to keep notation concise. [1]

2.1 Derivation 1d-Green's Function

The biggest problem of the integral form of the Lippmann-Schwinger equation is finding the corresponding Green's function. To simplify the notation, the limits $-\infty$ and $+\infty$ are omitted in the following.

Starting from the 1-dimensional single particle Schrödinger equation,

$$\left[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + V(r)\right]\,\psi_k(r) = E_k\,\psi_k(r), \tag{16}$$

one can insert the scattered energy eigenvalues given in equation 15. Rearranging equation 16 then yields

$$\left[\frac{\mathrm{d}^2}{\mathrm{d}r^2} + k^2\right]\,\psi_k(r) = 2\,V(r)\,\psi_k(r). \tag{17}$$

Now the left side can be interpreted as some operator $\hat{H}'$ acting on $\psi_k(r)$, where

$$\hat{H}' = \frac{\mathrm{d}^2}{\mathrm{d}r^2} + k^2 \tag{18}$$

The Green's function $G(r-r')$ of this operator $\hat{H}'$, which is also the Green's function of the normal Hamiltonian $\hat{H}$, is then defined as

$$\left[\frac{\mathrm{d}^2}{\mathrm{d}r^2} + k^2\right]\,G(r-r') = \delta(r-r'). \tag{19}$$

Again, a small imaginary shift is added to avoid the singularity of the operator. For simplicity, the limit is not written explicitly in front of every

equation.

$$\left[\frac{\mathrm{d}^2}{\mathrm{d}r^2} + k^2 + i\varepsilon\right] G(r - r') = \delta(r - r'). \tag{20}$$

Introducing the concept of the Fourier transform from the spatial r-space to the conjugate wave vector k-space, we define $\widetilde{G}(k')$ by

$$G(r - r') = \frac{1}{\sqrt{2\pi}} \int \mathrm{d}k'\, e^{ik'(r-r')}\, \widetilde{G}(k'). \tag{21}$$

Knowing that the Fourier transform of the delta function is $\frac{1}{\sqrt{2\pi}}$, one can write

$$\delta(r - r') = \frac{1}{\sqrt{2\pi}} \int \mathrm{d}k'\, e^{ik'(r-r')}\, \frac{1}{\sqrt{2\pi}} \tag{22}$$

Inserting both of these identities into equation 20, one gets

$$\left[\frac{\mathrm{d}^2}{\mathrm{d}r^2} + k^2 + i\varepsilon\right] \frac{1}{\sqrt{2\pi}} \int \mathrm{d}k'\, e^{ik'(r-r')}\, \widetilde{G}(k') = \frac{1}{\sqrt{2\pi}} \int \mathrm{d}k'\, e^{ik'(r-r')}\, \frac{1}{\sqrt{2\pi}} \tag{23}$$

$$\frac{1}{\sqrt{2\pi}} \int \mathrm{d}k' \left[k^2 - k'^2 + i\varepsilon\right] e^{ik'(r-r')}\, \widetilde{G}(k') = \frac{1}{\sqrt{2\pi}} \int \mathrm{d}k'\, e^{ik'(r-r')}\, \frac{1}{\sqrt{2\pi}} \tag{24}$$

As the integration with respect to k' and the derivative with respect to r are independent from one another and $e^{ik'(r-r')}\, \widetilde{G}(k')$ as well as $\frac{\mathrm{d}}{\mathrm{d}r}\left(e^{ik'(r-r')}\, \widetilde{G}(k')\right)$ are assumed to be continuous everywhere, the Leibniz rule for integration allows the exchange of the two operations.

Applying the Fourier transform to both sides yields an expression for $\widetilde{G}(k')$:

$$\left[k^2 - k'^2 + i\varepsilon\right] \widetilde{G}(k') = \frac{1}{\sqrt{2\pi}} \tag{25}$$

$$\widetilde{G}(k') = \frac{1}{\sqrt{2\pi}}\, \frac{1}{k^2 - k'^2 + i\varepsilon} \tag{26}$$

To get $G(r-r')$, the inverse Fourier transform, equation 21, gets used.

$$G(r-r') = \frac{1}{2\pi}\int \mathrm{d}k' \, \frac{e^{ik'(r-r')}}{k^2 - k'^2 + i\varepsilon} \tag{27}$$

$$G(r-r') = -\frac{1}{2\pi}\int \mathrm{d}k' \, \frac{e^{ik'(r-r')}}{k'^2 - k^2 - i\varepsilon} \tag{28}$$

Now one can split the denominator into two factors and use the taylor expansion to approximate the square root:

$$\sqrt{k^2 + i\varepsilon} \approx k + \frac{i\varepsilon}{2k} =: k + i\delta \tag{29}$$

This approximation is valid since $\varepsilon \ll k$. Here $\delta = \frac{\varepsilon}{2k}$ was used and replaces the entity that will approach 0 in the end.

Using this, equation 28 will read

$$G(r-r') = -\frac{1}{2\pi}\int \mathrm{d}k' \, \frac{e^{ik'(r-r')}}{[k' - (k+i\delta)]\,[k' + (k+i\delta)]} \tag{30}$$

For further analysis, the problem will be split into two possibilities: $r - r' > 0$ and $r - r' < 0$. Introducing a simpler, special case of the Theorem of Residues, Cauchy's Integral Theorem,

$$\oint_C \mathrm{d}z \, \frac{1}{z - z_0} f(z) = \pm 2\pi i f(z_0), \tag{31}$$

where $f(z)$ has no singularity in z_0 and C denotes some curve in the complex plane which encloses the singularity z_0, one can calculate the integral in equation 30. The sign of the right hand side depends on the mathematical direction of the curve. [4]

To do this, the purely real integral in equation 30 needs to be turned into an integral along some closed contour. For this to work, the contour needs to close the current path of integration without contributing to the value of

the integral. In case of $r - r' > 0$, this can be done by looping around from a far positive real k' over the upper complex plane to a negative real k'. To see that this additional path does not change the integral, the exponential function can be split into a purely real dampening term and a complex phase:

$$e^{ik'(r-r')} = e^{i\,\mathrm{Re}\,k'(r-r')}\, e^{-\mathrm{Im}\,k'(r-r')} \tag{32}$$

The absolute value of the phase is always 1, and for the case of $r - r' > 0$ and a contour far in the upper complex plane, the real exponential function will vanish. Therefore, the numerator is bounded and the denominator will go to infinity for high values of k'. As a result, the integrand vanishes for the part added to the path of integration.

In equation 30 the singularities can be easily recognized as being at $k' = k + i\delta$ and at $k' = -k - i\delta$. As k is real and δ is positive, one singularity lies in the upper half of the complex plane, the other in the lower half. Therefore, the contour described in the last paragraph encloses only a single singularity: the one at $k' = k + i\delta$. This in turn means, that Cauchy's Integral Theorem can be used to evaluate the integral:

$$G(r - r') = -\frac{1}{2\pi}\left[2\pi i\,\frac{e^{i(k+i\delta)(r-r')}}{(k + i\delta) + (k + i\delta)}\right] \tag{33}$$

$$= -\frac{i}{2}\,\frac{e^{i(k+i\delta)(r-r')}}{k + i\delta} \xrightarrow{\delta\to 0} -\frac{i}{2}\,\frac{e^{ik(r-r')}}{k} \tag{34}$$

The same calculations can be done for $r - r' < 0$. In this case, the contour will loop via the negative imaginary plane back from the end to the beginning of the prior integration path. In the decomposition of the exponential function in equation 32, $\mathrm{Im}\,k' < 0$ and as $r - r' < 0$, this term vanishes again. Therefore this contour also does not contribute to the integral. In this contour, the singularity at $k' = -k - i\delta$ is the only one inside the curve and as a result, Cauchy's Integral Theorem can be applied again. However, in this case the direction of the contour is mathematically negative and therefore

the negative sign needs to be used in the theorem.

$$G(r-r') = -\frac{1}{2\pi}\left[-2\pi i\,\frac{e^{i(-k-i\delta)(r-r')}}{(-k-i\delta)-(k+i\delta)}\right] \tag{35}$$

$$= -\frac{i}{2}\,\frac{e^{-i(k+i\delta)(r-r')}}{k+i\delta} \xrightarrow{\delta\to 0} -\frac{i}{2}\,\frac{e^{-ik(r-r')}}{k} \tag{36}$$

As the two cases differ only by the sign of $r-r'$, the minus sign in equation 36 can be included in the bracket and the two solutions can be fused together to get one solution for all values of r:

$$G(r-r') = -\frac{i}{2}\,\frac{e^{ik|r-r'|}}{k} \tag{37}$$

Further detail, as well as the main source of this information can be found in *Griffiths: Introduction to Quantum Mechanics* [2], as well as *Baym: Lectures on Quantum Mechanics* [1].

2.2 Transformation for Numerical Evaluation

To implement a numerical evaluation of the Lippmann-Schwinger equation (eq. 12), firstly, the integral needs to be transformed into some discrete sum, which the computer can handle. In this case, the trapizoidal rule was chosen, as it converges sufficiently fast and is still quite easy to implement. [6] The trapizoidal rule estimates the integral by splitting the domain into N equidistant sections and approximating the area in each section with the area of a trapezoid. Figure 1 demonstrates this slicing of the domain:

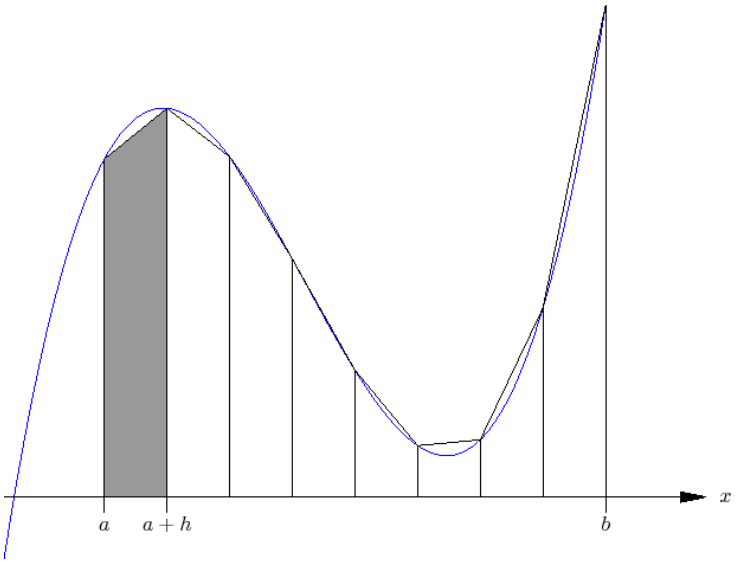

Fig. 1: Estimating an integral using the trapezoidal rule. Here, the domain of the integral is defined as the interval $[a, b]$ and h denotes the width of each slice. Taken from: [3]

Given a functional I, which, acting on a function f, calculates the integral of f from a to b:

$$I[f] = \int_a^b \mathrm{d}x\, f(x), \tag{38}$$

one can use the linearity of I to split the domain $[a, b]$ into N equidistant intervals to get

$$I[f] = \sum_{n=0}^{N-1} \int_{a+nh}^{a+(n+1)h} \mathrm{d}x\, f(x). \tag{39}$$

Now, using the formula for the area of a trapezoid, I can be approximated by

$$I_N[f] = \frac{h}{2} \sum_{n=0}^{N-1} \left[f(a+nh) + f(a+(n+1)h) \right]. \tag{40}$$

The bigger the number of subdivisions N, the better the approximation. The error of this method scales as $\mathcal{O}(h^2)$ for the entire interval $[a, b]$. For the limit of $N \to \infty$, both functionals are equal:

$$\lim_{N\to\infty} I_N[f] = I[f]. \tag{41}$$

Taking the 1-dimensional variant of the Lippmann-Schwinger equation (eq. 14), the infinite bounds of the integral need to be transformed to some finite values a and b. In this case, the infinite integral can be approximated by just starting the integration path at some low value a and stopping at some high value b. This is justified, if the potential $V(r)$ quickly vanishes near one of these bounds, and the other parts of the integrand stay bounded. Under these assumptions, the integral outside the interval $[a, b]$ is negligibly small. So by setting a and b to some value distant from the significant parts of the potential, the infinite integration bounds can be approximated finite:

$$\psi(r) = \phi(r) + \int_{-\infty}^{\infty} \mathrm{d}r'\, G(r, r')\, V(r)\, \psi(r') \tag{42}$$

$$\approx \phi(r) + \int_{a}^{b} \mathrm{d}r'\, G(r, r')\, V(r)\, \psi(r') \tag{43}$$

To allow numerical calculations, the position r will be discretized to a vector r_j, where $j \in \mathbb{N}$ indexes the various positions, where r is defined. In the case of this numerical evaluation, this also gets chosen to be the same interval $[a, b]$ as the domain of r', but this is not strictly necessary. For notational convenience, some abbreviations will be used in the next equations: $r' = a + jh$ will be replaced by r_j, and following this, for some function $F(r_j)$, the shortcut F_j will be used. This last substitution will also be expanded to functions depending on several variables: $F(r_j, r'_k) = F_{jk}$. Using this notation and implementing the discretization discussed in equation 40 with some large N, one gets

$$\psi_l \approx \phi_l + \frac{h}{2} \sum_{n=0}^{N-1} \left[G_{ln}\, V_n\, \psi_n + G_{l(n+1)}\, V_{n+1}\, \psi_{n+1} \right]. \tag{44}$$

For the sake of simplicity, N is assumed to be large enough for the difference between the real integral and the approximation to be negligibly small. Therefore '=' will be written instead of '≈'.

Summing over n yields every $(G_{ln}V_n\psi_n)$-term twice, except the first $(n = 0)$ and the last $(n = N)$. Excluding these two terms from the sum, the equation can be modified to read

$$\psi_l = \phi_l + \frac{h}{2} G_{l0} V_0 \psi_0 + \frac{h}{2} G_{lN} V_N \psi_N + h \sum_{n=1}^{N-1} G_{ln} V_n \psi_n. \tag{45}$$

After this discretization, G_{ln} can be represented by a matrix, where l indexes the rows and contributes the dependence of G on r_l, and n indexes the columns and contributes the dependence on r'_n. The multiplication of G_{ln} with V_n and ψ_n needs to be interpreted element-wise. This multiplication of G with V yields a matrix again. So, by scaling this matrix by h, weighing the first and last column by $\frac{1}{2}$ and calling it K, equation 45 can be written quite compactly:

$$\psi_l = \phi_l + \sum_{n=0}^{N} K_{ln} \psi_n. \tag{46}$$

Bringing the sum to the left side of the equation, and multiplying ψ_l with the identity matrix δ_{ln} yields:

$$\sum_{n=0}^{N} \delta_{ln} \psi_n - \sum_{n=0}^{N} K_{ln} \psi_n = \phi_l \tag{47}$$

Here, $\sum_n \delta_{ln}\psi_n = \psi_l$ was used to insert the identity matrix. Now, as the sums are finite, they can be joined and the common factor of ψ_n can be separated.

$$\sum_{n=0}^{N} [\delta_{ln} - K_{ln}] \psi_n = \phi_l \tag{48}$$

Introducing yet another matrix $A_{ln} = \delta_{ln} - K_{ln}$, one can see, that this equation corresponds to a system of linear equations with the coefficient matrix

A_{ln} , the unknown vector ψ_n and the inhomogeneity ϕ_l.

$$\sum_{n=0}^{N} A_{ln}\psi_n = \phi_l \tag{49}$$

This system of linear equations can then be solved quite easily using a computer. As this derivation shows, the solution to this system of equations then also gives an approximation for the solution of the 1-dimensional Lippmann-Schwinger equation, which is the original problem.

2.3 Implementation in Python

For all numerical evaluations in this thesis, Python 3.7.3 will be used. To make repeated calculations easier and the whole program more user friendly, the user input and customization options have been put into a separate file independent of the actual code. In this input file, the box dimensions $[a, b]$, the number of slices N, the wave number of the incoming wave k and the potential can be specified.

```
a = -1.5; # left border of simulation box
b = 1.5; # right border
N = 160; # number of subdivisions
pot_id = 'square';  # id of potential to use
k_vec = numpy.linspace(0.1, 4.5, 4.5 / 0.1);
    # wave number for incoming wave
```

Here, `pot_id` refers to an identifier, which corresponds to a below defined potential. A dictionary then links the ID to the function pointer. For a rectangular potential, one could define the function like this:

```
def Square(x):
    L = 2;
    V0 = 1;
    z = numpy.zeros(x.shape);
    z[abs(x) <= L/2] = -V0;
    return z;
```

To implement equation 49 and therefore the solution of the Lippmann-Schwinger equation, the matrix A_{ln} needs to be calculated. As A_{ln} depends on K_{ln} ,

which in turn is a function of G_{ln} and V_n, these also need to be implemented. Firstly, the program creates two identical spatial vectors with the parameters a, b and N specified in the input file. It then uses numpys `meshgrid` function to get two matrices with all possible combinations of two positions. For each combination, the program then calculates the potential V_n and the Green's function matrix G_{ln}, which in turn are used to calculate K_{ln} and then A_{ln}.

```
def G(x, xp): # 1 dimensional Greens function
    return -1j / k * numpy.exp(1j * k * abs(x - xp));

K = numpy.zeros((N+1, N+1), dtype = numpy.complex_)
    # setup K(x, t) matrix
[xx, tt] = numpy.meshgrid(x, t); # get all (x, t) combinations
K = (G(xx, tt) * V(tt)).transpose(); # calculate values of K
A = numpy.identity(N+1) - h * K;
```

Having evaluated the matrix A_{ln}, it is now possible to solve the system of linear equations. For this purpose, numpys `linalg.solve` function has been chosen because it has performed best in the timed tests. Performance will be discussed more extensively in section 4.

```
phi = numpy.exp(1j * k * x);  # inhomogenous function phi

psi = numpy.linalg.solve(A, phi); # solve system of equations
```

2.4 Results

Using matplotlib, this — generally complex — vector ψ_n can now be plotted. However, plotting the real and imaginary part is not very insightful, so the absolute value $|\psi_n|$ will be displayed. The following image uses a square potential well, which has a value of -1 inside the interval $[-1, 1]$ and 0 outside. Furthermore, the number of subdivisions was chosen to be $N = 320$, and the wave number k was varied in the interval $[0.5, 4.5]$.

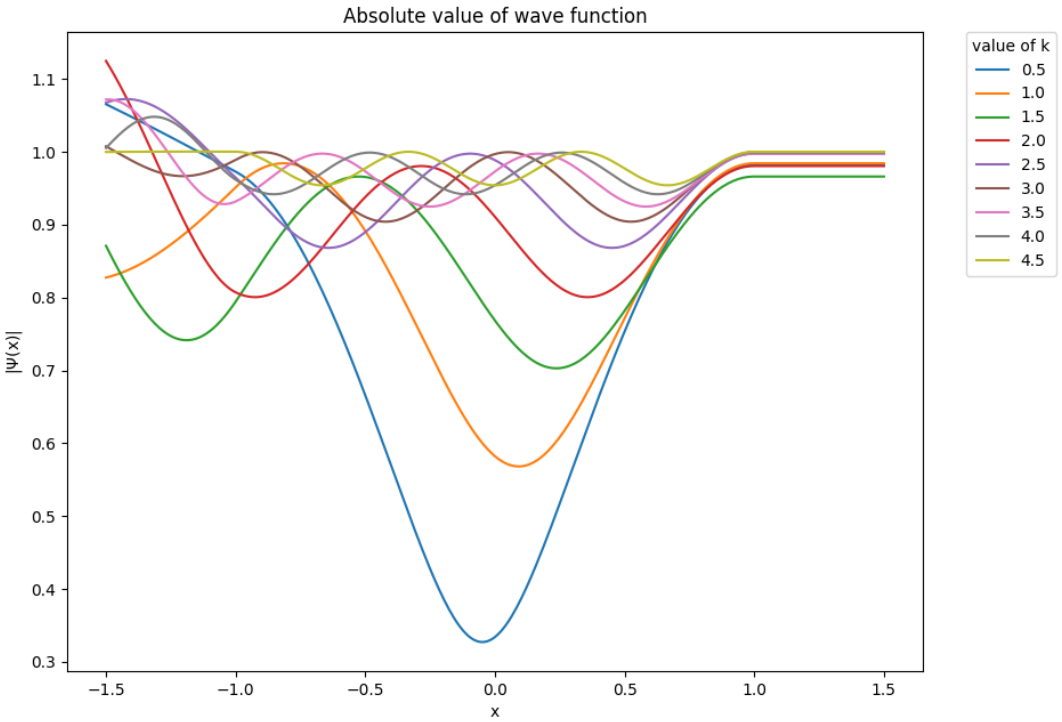

Fig. 2: Absolute value of wave function $|\psi|$ as a function of position x. This example uses the square potential well. ($N = 320$)

As one can see, the lower the wave number k, the more the wave function gets damped. This is expected, as for a lower k, the reflection coefficient is generally higher and therefore also the reduction of the incoming wave. To the right side of the barrier, no reflected waves are present, so no interference occurs. This results in the absolute value of the wave function ψ_n being constant. Of course, it still oscillates in the real and imaginary parts, but these are just rotations in the complex plane without changes in the magnitude.

Analagously to the reflection coefficient, the transmission coefficient T of the passing wave depends on k. For the case of a square well potential, the transmission coefficient $T(E)$ can be calculated analytically, which can be

used to test the numerical implementation.

$$T(E) = |\psi(b)|^2 = \left[1 + \frac{V_0^2 \, \sin(qL)}{4\,E\,(E+V_0)}\right]^{-1} \tag{50}$$

$$\text{where} \quad E = \frac{k^2}{2}, \quad q = \sqrt{2\,(E+V_0)} \quad \text{and} \quad L = b - a. \tag{51}$$

By the definition of the transmission coefficient given in the first part of equation 50, the numerically calculated wave function ψ_n will give values for T. As $|\psi_n|$ is constant for all positions between the end of the potential and the right border b, T will be the same value regardless of the point chosen, given it is somewhere inside this interval. In the numerical calculations of this thesis, the point b will be chosen nonetheless.

In the following picture, this transmission coefficient T is displayed as a function of k.

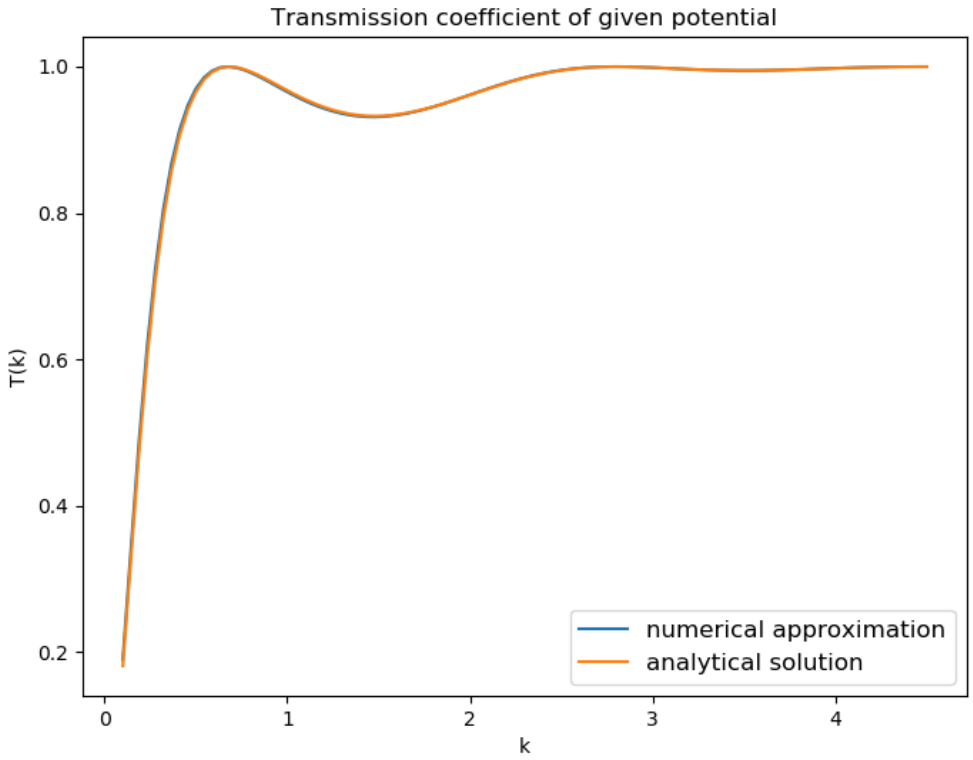

Fig. 3: Transmission coefficient T as a function of wave number k.
Here 100 equidistant values for k in the interval $(0, 4.5]$ were chosen.

Figure 3 shows clearly, that the values from the numerical solution to the Lippmann-Schwinger equation match the analytical form perfectly. It should be noted that the numerical evaluation of the Lippmann-Schwinger equation breaks down for small values of k. In particular, the special case $k = 0$ is ill-defined in the numerical approach and must be avoided.

3 2-Dimensional Case

For the 2-dimensional analysis, the Lippmann-Schwinger equation depends on a 2-dimensional position vector $\boldsymbol{r} = (x, y)$. In this form, the equation

assumes the following form:

$$\psi(x,y) = \phi(x,y) + \int \mathrm{d}x'\,\mathrm{d}y'\, G(x,y;x',y')\, V(x',y')\, \psi(x',y') \tag{52}$$

Here, $\phi(x,y)$ denotes the wave function of the free Hamilton operator in 2-dimensions:

$$\phi(x,y) = e^{i\boldsymbol{k}\boldsymbol{r}} = e^{i(k_x x + k_y y)} \qquad \text{with corresponding} \qquad E = \frac{|\boldsymbol{k}|^2}{2} = \frac{k^2}{2} \tag{53}$$

For the sake of simplicity, the vector (x,y) will be denoted as $\boldsymbol{r}$ and the vector (k_x, k_y) as $\boldsymbol{k}$. These must not be confused with the 3-dimensional $\boldsymbol{r}$ and $\boldsymbol{k}$ used in the general derivation of the Lippmann-Schwinger equation.

3.1 Derivation 2d-Green's Function

The beginning of the derivation of the 2-dimensional Green's function is analogous to the 1-dimensional variant. However, later on, they start to differ drastically. First of, the 2-dimensional Schrödinger equation is needed:

$$\left[\frac{1}{2}\Delta + E\right] \psi_{\boldsymbol{k_0}}(\boldsymbol{r}) = V(\boldsymbol{r})\, \psi_{\boldsymbol{k_0}}(\boldsymbol{r}) \tag{54}$$

In this equation $\Delta = \frac{\mathrm{d}^2}{\mathrm{d}x^2} + \frac{\mathrm{d}^2}{\mathrm{d}y^2}$ denotes the Laplace operator. Using the energy from equation 53, equation 54 now becomes

$$\left[\Delta + k_0^2\right] \psi_{\boldsymbol{k_0}}(\boldsymbol{r}) = 2\, V(\boldsymbol{r})\, \psi_{\boldsymbol{k_0}}(\boldsymbol{r}). \tag{55}$$

Applying the definition of the Green's function, one gets

$$\left[\Delta + k_0^2 + i\varepsilon\right] G(\boldsymbol{r}, \boldsymbol{r'}) = \delta(\boldsymbol{r} - \boldsymbol{r'}), \tag{56}$$

where again a small imaginary shift was introduced to avoid the singularity.

By again using the 2-dimensional Fourier transform

$$G(\boldsymbol{k}, \boldsymbol{r'}) = \frac{1}{2\pi} \int \mathrm{d}^2 r \, e^{-i\boldsymbol{kr}} \, G(\boldsymbol{r}, \boldsymbol{r'}) \tag{57}$$

$$G(\boldsymbol{r}, \boldsymbol{r'}) = \frac{1}{2\pi} \int \mathrm{d}^2 k \, e^{i\boldsymbol{kr}} \, G(\boldsymbol{k}, \boldsymbol{r'}) \tag{58}$$

equation 56 can be rewritten as

$$\left[\Delta + k_0^2 + i\varepsilon\right] \frac{1}{2\pi} \int \mathrm{d}^2 k \, e^{i\boldsymbol{kr}} \, G(\boldsymbol{k}, \boldsymbol{r'}) = \delta(\boldsymbol{r} - \boldsymbol{r'}). \tag{59}$$

Again, using the Leibniz rule for integration, this equation transforms to

$$\frac{1}{2\pi} \int \mathrm{d}^2 k \, \left[\Delta + k_0^2 + i\varepsilon\right] e^{i\boldsymbol{kr}} \, G(\boldsymbol{k}, \boldsymbol{r'}) = \delta(\boldsymbol{r} - \boldsymbol{r'}) \tag{60}$$

$$\frac{1}{2\pi} \int \mathrm{d}^2 k \, \left[-k^2 + k_0^2 + i\varepsilon\right] e^{i\boldsymbol{kr}} \, G(\boldsymbol{k}, \boldsymbol{r'}) = \delta(\boldsymbol{r} - \boldsymbol{r'}) \tag{61}$$

Applying the Fourier transform on both sides and dividing by $[-k^2 + k_0^2 + i\varepsilon]$ yields a formula for $G(\boldsymbol{k}, \boldsymbol{r'})$:

$$G(\boldsymbol{k}, \boldsymbol{r'}) = -\frac{1}{2\pi} \frac{e^{-i\boldsymbol{kr'}}}{k^2 - k_0^2 - i\varepsilon} \tag{62}$$

Using the inverse Fourier transform, $G(\boldsymbol{r}, \boldsymbol{r'})$ can be found:

$$G(\boldsymbol{r}, \boldsymbol{r'}) = -\frac{1}{(2\pi)^2} \int \mathrm{d}^2 k \, \frac{e^{-i\boldsymbol{k}(\boldsymbol{r}-\boldsymbol{r'})}}{k^2 - k_0^2 - i\varepsilon} = G(\boldsymbol{r} - \boldsymbol{r'}) \tag{63}$$

To get rid of the $(\boldsymbol{r} - \boldsymbol{r'})$ term and to separate the two integrals, a transformation of variables will be made. A new position variable $\boldsymbol{u} = \boldsymbol{r} - \boldsymbol{r'} = -u\,\hat{\boldsymbol{e}}_{\boldsymbol{u}}$ with a perpendicular coordinate $\hat{\boldsymbol{e}}_{\boldsymbol{w}}$ gets introduced. Here, $\hat{\boldsymbol{e}}_{\boldsymbol{u}}$ and $\hat{\boldsymbol{e}}_{\boldsymbol{w}}$ represent unit vectors in the direction of $\boldsymbol{u}$ and orthogonal to it respectively. In this system of coordinates, the wave vector $\boldsymbol{k}$ can be written as $\boldsymbol{k} = k_u\,\hat{\boldsymbol{e}}_{\boldsymbol{u}} + k_w\,\hat{\boldsymbol{e}}_{\boldsymbol{w}}$. This representation of $\boldsymbol{k}$ allows the simplification $\boldsymbol{ku} = (k_u\,\hat{\boldsymbol{e}}_{\boldsymbol{u}} + k_w\,\hat{\boldsymbol{e}}_{\boldsymbol{w}})\,(-u\,\hat{\boldsymbol{e}}_{\boldsymbol{u}}) =$

$-k_u\, u$ due to the orthogonality relation $\hat{\boldsymbol{e}}_{\boldsymbol{w}}\, \hat{\boldsymbol{e}}_{\boldsymbol{u}} = 0$ and normalization criterium $\hat{\boldsymbol{e}}_{\boldsymbol{w}}\, \hat{\boldsymbol{e}}_{\boldsymbol{w}} = \hat{\boldsymbol{e}}_{\boldsymbol{u}}\, \hat{\boldsymbol{e}}_{\boldsymbol{u}} = 1$. In this notation $k^2 = |\boldsymbol{k}|^2 = k_u^2 + k_w^2$. Applying this coordinate transformation to equation 63 yields

$$G(\boldsymbol{u}) = -\frac{1}{(2\pi)^2} \int \mathrm{d}k_u\, \mathrm{d}k_w\, \frac{e^{-ik_u u}}{k_w^2 + k_u^2 - k_0^2 - i\varepsilon}. \tag{64}$$

As the numerator doesn't depend on k_w, it can be put in front of the integral with respect to k_w. The denominator can now be factored using $a^2 + b^2 = (a + ib)(a - ib)$.

$$G(\boldsymbol{u}) = -\frac{1}{(2\pi)^2} \int \mathrm{d}k_u\, e^{-ik_u u} \int \mathrm{d}k_w\, \frac{1}{k_w^2 + k_u^2 - k_0^2 - i\varepsilon} \tag{65}$$

$$= -\frac{1}{(2\pi)^2} \int \mathrm{d}k_u\, e^{-ik_u u} ... \tag{66}$$

$$... \int \mathrm{d}k_w \frac{1}{(k_w + i\sqrt{k_u^2 - k_0^2 - i\varepsilon})(k_w - i\sqrt{k_u^2 - k_0^2 - i\varepsilon})} \tag{67}$$

In the second integral — the one with respect to k_w — the part $\sqrt{k_u^2 - k_0^2 - i\varepsilon}$ acts like a constant and will be replaced by z_0. For the sake of simple notation, k_w will be replaced by z, and the integral with respect to k_w examined alone:

$$\int \mathrm{d}k_w \frac{1}{(k_w + i\sqrt{k_u^2 - k_0^2 - i\varepsilon})(k_w - i\sqrt{k_u^2 - k_0^2 - i\varepsilon})} = ... \tag{68}$$

$$... = \int \mathrm{d}z\, \frac{1}{(z + iz_0)(z - iz_0)} \tag{69}$$

Again, having a similar problem as in equation 30, the Theorem of Residues (eq. 31) will be used to manage the integral over the singularities. For $k_u > k_0$, z_0 is real, otherwise z_0 is imaginary. As z_0 is in the limit a root of a real number, z_0 cannot have both a real and an imaginary part. Depending on this, the poles are either purely real or imaginary. If the poles are purely real, the same path as for equation 30 can be used. Otherwise, the curve will also be the same, except it will not need to curve around the singularities, as they will not lie on the path of integration anymore. In both cases, only one singularity resides within the curve and the Theorem of

Residues will give the same value for the integral.

$$\int \mathrm{d}z \, \frac{1}{(z+iz_0)(z-iz_0)} = \oint_C \mathrm{d}z \, \frac{1}{z-iz_0} \, \frac{1}{z+iz_0} \tag{70}$$

$$= 2\pi i \frac{1}{iz_0+iz_0} = \frac{\pi}{z_0} \tag{71}$$

Resubstituting z_0 and inserting this result into equation 67 will give

$$G(\boldsymbol{u}) = -\frac{1}{(2\pi)^2} \int \mathrm{d}k_u \, e^{-ik_u u} \, \frac{\pi}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} \tag{72}$$

For taming this integral, one can split it into a real and an imaginary part. To do this, the denominator needs to be identified of being purely real or purely imaginary. This only depends on the magnitude of k_u and k_0: If $|k_u| > |k_0|$, then it is real, if $|k_u| < |k_0|$, then it is imaginary. As k_0 is a fixed constant and k_u is the variable of integration, the integral, which has limits $(-\infty, \infty)$, can be split into a part, where $|k_u| > |k_0|$ and one where $|k_u| < |k_0|$.

$$G(\boldsymbol{u}) = -\frac{1}{(2\pi)^2} \int \mathrm{d}k_u \, e^{-ik_u u} \, \frac{\pi}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} \tag{73}$$

$$= -\frac{1}{(2\pi)^2} \left[\int\limits_{|k_u|>|k_0|} \mathrm{d}k_u \, \frac{\pi \, e^{-ik_u u}}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} + \int\limits_{|k_u|<|k_0|} \mathrm{d}k_u \, \frac{\pi \, e^{-ik_u u}}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} \right] \tag{74}$$

$$= -\frac{1}{(2\pi)^2} \left[\int\limits_{|k_u|>|k_0|} \mathrm{d}k_u \, \frac{\pi \, e^{-ik_u u}}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} + \int\limits_{|k_u|<|k_0|} \mathrm{d}k_u \, \frac{i\pi \, e^{-ik_u u}}{\sqrt{k_0^2 - k_u^2 + i\varepsilon}} \right] \tag{75}$$

Now the denominators are purely real. Furthermore, they are even functions with respect to k_u, as they only depend on k_u^2. Using Euler's identity, $e^{i\varphi} = \cos\varphi + i\sin\varphi$, the numerator can be separated into a real and imaginary part. Using the linearity of the integral, the $(\cos k_u u + i \sin k_u u)$ sum can be split into separate integrals. In case of the numerator containing $\sin k_u u$, which is an odd function, the integrand is a product of and odd and

an even function, which is an odd function. As the integral spans a symmetric domain, the positive and negative parts of the domain of the integral cancel and the integral vanishes. Therefore only the parts with $\cos k_u u$ remain. As $\cos k_u u$ is an even function and the integrand then is a product of two even functions, the whole integrand is an even function. Again, the domain of the integral is symmetric, so the contribution of the positive and negative part of the domain are equal. Therefore the domain can be confined to the positive part and the result multiplied by 2. For the sake of readability, $G(\boldsymbol{u})$ will be split into the outer and inner integral.

$$G(\boldsymbol{u}) = -\frac{1}{(2\pi)^2}\,\left[G_o(\boldsymbol{u}) + G_i(\boldsymbol{u})\right] \tag{76}$$

Just coping with the outer integral for now, one gets:

$$\begin{aligned}
G_o(\boldsymbol{u}) &= \int\limits_{|k_u|>|k_0|} \mathrm{d}k_u\, \frac{\pi\, e^{-ik_u u}}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} && (77)\\
&= \int\limits_{|k_u|>|k_0|} \mathrm{d}k_u\, \frac{\pi\,(\cos k_u u + i \sin k_u u)}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} && (78)\\
&= \int\limits_{|k_u|>|k_0|} \mathrm{d}k_u\, \frac{\pi\,\cos k_u u}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} + \int\limits_{|k_u|>|k_0|} \mathrm{d}k_u\, \frac{\pi\, i \sin k_u u}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} && (79)\\
&= \int\limits_{|k_u|>|k_0|} \mathrm{d}k_u\, \frac{\pi\,\cos k_u u}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} && (80)\\
&= 2\int\limits_{k_0}^{\infty} \mathrm{d}k_u\, \frac{\pi\,\cos k_u u}{\sqrt{k_u^2 - k_0^2 - i\varepsilon}} && (81)
\end{aligned}$$

Analogously the inner integral $G_i(\boldsymbol{u})$ can be simplified to

$$G_i(\boldsymbol{u}) = 2\int\limits_{0}^{k_0} \mathrm{d}k_u\, \frac{i\pi\,\cos k_u u}{\sqrt{k_0^2 - k_u^2 + i\varepsilon}}\ . \tag{82}$$

Continuing with the outer integral $G_o(\boldsymbol{u})$, a substitution $s = \frac{k_u}{k_0}$ transforms

it into a standard integral

$$G_o(\boldsymbol{u}) = 2\pi \int_1^\infty k_0 \,\mathrm{d}s \, \frac{\cos k_0 us}{\sqrt{(k_0 s)^2 - k_0^2 - i\varepsilon}} \tag{83}$$

$$\xrightarrow{\varepsilon\to 0} 2\pi \int_1^\infty k_0 \,\mathrm{d}s \, \frac{\cos k_0 us}{\sqrt{(k_0 s)^2 - k_0^2}} = 2\pi \int_1^\infty \mathrm{d}s \, \frac{\cos k_0 us}{\sqrt{s^2 - 1}} \tag{84}$$

Here the limit has been applied as the singularity is now at the margin of the integration interval, where it does not contribute to its value. This special integral as a function of $k_0 u$ is given by the 0th order Bessel function of second kind $Y_0(k_0 u)$. As only the numerical evaluation of this integral counts, and there are a lot of python packages which can deal with Bessel functions, this function will not be evaluated further.

$$G_o(\boldsymbol{u}) = 2\pi \int_1^\infty \mathrm{d}s \, \frac{\cos k_0 us}{\sqrt{s^2 - 1}} = 2\pi \left[-\frac{\pi}{2} Y_0(k_0 u) \right] = -\pi^2 \, Y_0(k_0 u) \tag{85}$$

Returning to the inner integral, a substitution of $k_u = k_0 \cos\vartheta$ is helpful.

$$G_i(\boldsymbol{u}) = 2\pi i \int_{\frac{\pi}{2}}^0 -k_0 \sin\vartheta \mathrm{d}\vartheta \, \frac{\cos\left(k_0 u \cos\vartheta\right)}{\sqrt{k_0^2 - k_0^2 \cos^2\vartheta + i\varepsilon}} \tag{86}$$

$$\xrightarrow{\varepsilon\to 0} 2\pi i \int_{\frac{\pi}{2}}^0 -k_0 \sin\vartheta \mathrm{d}\vartheta \, \frac{\cos\left(k_0 u \cos\vartheta\right)}{\sqrt{k_0^2 - k_0^2 \cos^2\vartheta}} \tag{87}$$

$$= 2\pi i \int_0^{\frac{\pi}{2}} \mathrm{d}\vartheta \, \cos\left(k_0 u \cos\vartheta\right) \tag{88}$$

This, again, is a standard integral whose value is given by a Bessel function.

This time, it's the 0th order Bessel function of first kind $J_0(k_0 u)$.

$$G_i(\boldsymbol{u}) = 2\pi i \int_0^{\frac{\pi}{2}} \mathrm{d}\vartheta \, \cos\left(k_0 u \cos\vartheta\right) = 2\pi i \left[\frac{\pi}{2} J_0(k_0 u)\right] = i\pi^2 J_0(k_0 u) \tag{89}$$

Combining these forms of the inner (eq. 89) and outer (eq. 85) integral and replacing k_0 by k, the Green's function can be calculated as given in equation 76:

$$G(\boldsymbol{u}) = -\frac{1}{(2\pi)^2} \left[-\pi^2 Y_0(ku) + i\pi^2 J_0(ku)\right] \tag{90}$$

$$= -\frac{1}{4} \left[-Y_0(ku) + iJ_0(ku)\right] \tag{91}$$

$$= -\frac{i}{4} H_0^{(1)}(ku) \tag{92}$$

Here, a new function $H_0^{(1)}(z) \equiv J_0(z) + iY_0(z)$, the 0th order Hankel function of first kind, has been introduced. Resubstituting $\boldsymbol{u} = \boldsymbol{r} - \boldsymbol{r'}$ and therefore $u = |\boldsymbol{r} - \boldsymbol{r'}|$ will yield the Green's function for the 2-dimensional Lippmann-Schwinger equation as a function of positions $\boldsymbol{r}$ and $\boldsymbol{r'}$.

$$G(\boldsymbol{r}, \boldsymbol{r'}) = -\frac{i}{4} H_0^{(1)}(k\,|\boldsymbol{r} - \boldsymbol{r'}|) \tag{93}$$

Similar to the 1-dimensional case, $G(\boldsymbol{r}, \boldsymbol{r'})$ is only a function of the distance between its two position arguments. As a result, it is also often written as $G(\boldsymbol{r} - \boldsymbol{r'})$. Moreover it only depends on the magnitude k of the wave vector, as defined in equation 53.

3.2 Transformation for Numerical Evaluation

For transforming the 2-dimensional Lippmann-Schwinger equation (eq. 52) into a form, which can be solved easily numerically, the integral will again be turned into a sum. To allow for a more straight-forward discretization, we make use of the midpoint-rule which exhibits the same accuracy as the

trapizoidal rule (eq. 40 and 41 as well as fig. 1), but treats all points, including the end points, identically. Similar to the trapezoidal rule, instead of approximating the area under the curve with trapezoids, in the rectangular midpiont rule, rectangles will be used. For the height of the rectangles, the point in the middle of the interval in the domain of the function will serve, hence the name *midpoint* rule.

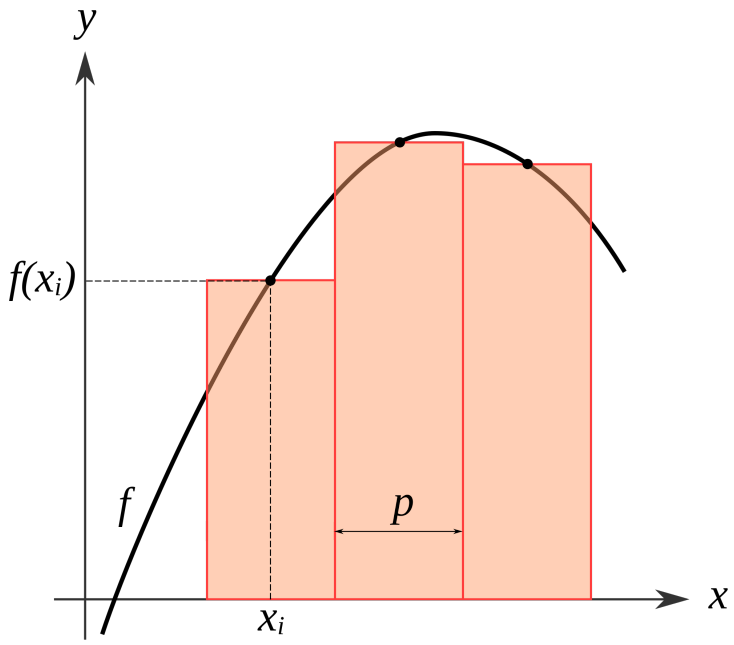

Fig. 4: Illustration of the midpoint rule.
Here x_i denotes the center point and p the width of the intervals (equivalent to h in the rest of this thesis).
Taken from: [5], slightly modified.

Taking the same functional I as before in section 2.2, equation 38 and 39, it can be approximated with the rectangular rule as follows:

$$I_N[f] = h \sum_{n=0}^{N-1} f\left(a + \left(n + \frac{1}{2}\right) h\right) \tag{94}$$

Again, for higher and higher N, $I_N[f]$ approaches $I[f]$.

$$\lim_{N\to\infty} I_N[f] = I[f] \tag{95}$$

For adapting this integration method to 2 dimensions, an 2-d analog to I will be defined:

$$I'[f] = \int_{a_y}^{b_y} \int_{a_x}^{b_x} \mathrm{d}x \, \mathrm{d}y \, f(x, y) \tag{96}$$

By switching from 1 to 2 dimensions, these rectangles become cuboids, whose height equals the value of the function in the center of the top and bottom face. Splitting the total integration period $[a_x, b_x]$ on the x-axis into N_x equidistant parts of width h_x and $[a_y, b_y]$ on the y-axis into N_y equidistant parts of width h_y, the rectangular midpoint rule gives the following approximation for $I'[f]$:

$$I'_{N_x,N_y}[f] = h_x \, h_y \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} f\left(a_x + \left(n_x + \frac{1}{2}\right) h_x, a_y + \left(n_y + \frac{1}{2}\right) h_y\right) \tag{97}$$

As these two sums are finite and the two indices n_x and n_y are completely independent of each other, the two sums can be merged into a single one, whose index n ranges from 0 to $N - 1$, where $N = N_x \, N_y$. To keep the notation more compact, $f(x_n, y_n)$ will be denoted as f_n, where the dependence of x and y on n are given by

$$x_n = a_x + \left((n \bmod N_x) + \frac{1}{2}\right) h_x \tag{98}$$

$$y_n = a_y + \left(\frac{n - (n \bmod N_x)}{N_x} + \frac{1}{2}\right) h_y \tag{99}$$

Here, 'mod' symbolizes the mathematical modulo operation, which returns the remainder of a division.

Using this notation, equation 97 can be rewritten as

$$I'_N[f] = h_x \, h_y \sum_{n=0}^{N-1} f_n \tag{100}$$

Applying this approximation to the 2-dimensional Lippmann-Schwinger equation (eq. 52), its discretized version turns into

$$\psi_l = \phi_l + h_x\, h_y \sum_{n=0}^{N-1} G_{ln}\, V_n\, \psi_n. \tag{101}$$

All indices in this equation are linear indices, which represent positions in the (x, y)-plane.

Similar to the 1-dimensional case, a matrix $K_{ln} = h_x\, h_y\, G_{ln}\, V_n$ can be defined. Rearranging equation 101 yields

$$\sum_{n=0}^{N} (\delta_{ln} - K_{ln})\ \psi_n = \phi_l \tag{102}$$

and with $A_{nm} = \delta_{nm} - K_{nm}$ the standard form of a system of linear equations can be obtained:

$$\sum_{n=0}^{N} A_{ln}\, \psi_n = \phi_l \tag{103}$$

3.3 Implementation in Python

The core of the Python implementation of the 2-d model is similar to the 1-d variant. However, as the increased processor and memory demands strongly limit the capabilities of the 2-dimensional version, more emphasis has been put on computational efficiency when writing the code. For the sake of saving memory, a lot of readability of the code is sacrificed, as all big matrices are stored in one and the same space, referenced by one and the same pointer.

Identical to the 1-dimensional version, a separate file was used for the input. For the integrals to be evaluated for $x \in [x1, x2]$ and $y \in [y1, y2]$, the following code was used:

```
x1 = -2;
x2 = 2;
y1 = -2;
y2 = 2;
Nx = 40; # number of points in x direction
Ny = 40  # number of points in y direction
pot_id = 'Trench'; # type of potential to use
angle = numpy.zeros((16, 1)); # angle between k and x-axis
k_mag = numpy.linspace(1, 16, 16); # magnitude of k vector
```

Instead of defining the wave vector $\boldsymbol{k}$ by its Cartesian components in x and y-direction, $\boldsymbol{k}$ can be fed into the program in polar coordinates. The reason for this is the ability to separately change the magnitude and the angle of the vector very easily.

In the main code, the $\boldsymbol{x}$ and $\boldsymbol{y}$ vectors get defined and with meshgrid and scipys `cdist`, the distance between $\boldsymbol{r}$ and $\boldsymbol{r'}$ gets calculated for every combination of 2 points.

```
X, Y = numpy.meshgrid(x, y);  # get all (x, y) combinations
eps = 1e-4; # for avoiding 0 in hankel function
coords = numpy.array([numpy.ravel(X),
        numpy.ravel(Y)]).transpose();
S = scipy.spatial.distance.cdist(coords, coords + eps,
        'euclidean') + 0j;
```

Here, a tiny shift `eps` to the second point has been introduced. The idea behind this is to avoid the singularity in the Hankel function later on. As this shift is negligiby small compared to the other values for the distance, it can be ignored in the endresult. However, to keep the code consistent with the formula, this `eps`-shift is added to all coordinates, over which the integral is defined.

Furthermore, a complex `+ 0j` is added. Reason for this is to convert the datatype of `S` to be complex, which in turn has been done to be able to store other (complex) matrices, which will be derived from `S`, in the same space `S` refers to.

The Hankel function is calculated using `scipy.special.hankel1`, which is the most suitable tested variant. Further detail concerning the performance of various parts of the code can be found in section 4.

```
scipy.special.hankel1(0, k*S, out=S);
S *= -1j / 4;
```

At this point, `S` contains the Green's function. To get the coefficient matrix of the system of linear equations, the same algorithm as in the 1-dimensional case is used. However, it has been rewritten in a different style to allow reusing the same storage space `S` reserves.

```
h = (x[1] - x[0]) * (y[1] - y[0]);
S *= numpy.ravel(V(X + eps, Y + eps));
S *= -h;
S += numpy.identity(N);
psi = numpy.linalg.solve(S, phi);
```

Here $h = h_x h_y$ is calculated directly using adjacent points of the grid of positions. In the end, again `numpy.linalg.solve` has been used to solve the system of linear equations.

3.4 Results

As in the 1-dimensional case, matplotlib is used to plot the results. All shown wave functions are not normalized, as only their behavior inside the simulation box is known.

For the first demonstration, a simple cuboid potential trench has been chosen. It spans all y-values and has a depth of $V_0 = -1$ for all $x \in [-0.5, 0.5]$. Figure 5 displays $V(x, y)$ in the simulation box $[-2, 2] \times [-2, 2]$.

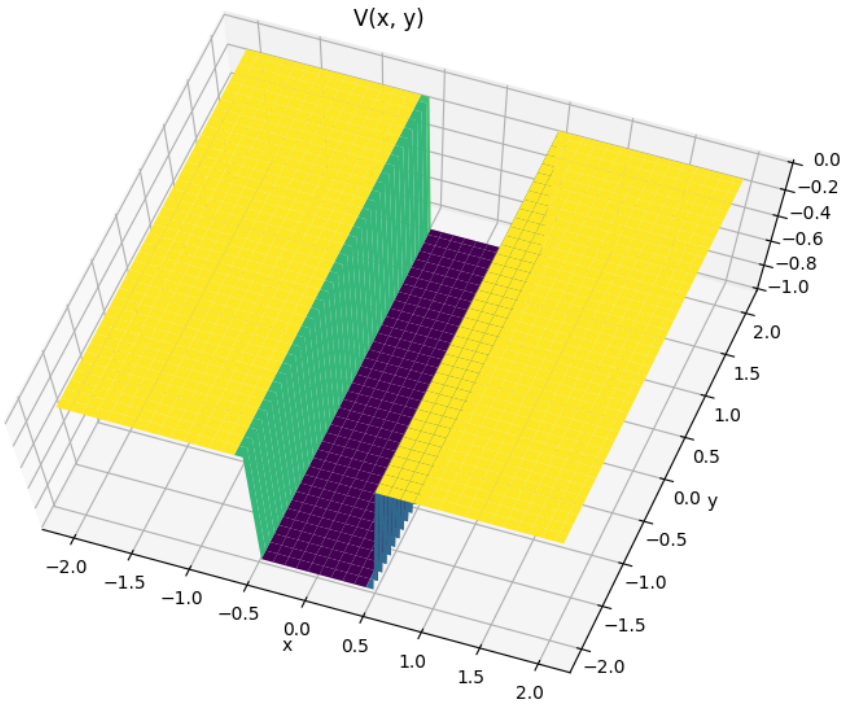

Fig. 5: Potential $V(x, y)$ used for the first simulation. $N_x = N_y = 120$

Setting $\boldsymbol{k} = 8\,\hat{\boldsymbol{e}}_{\boldsymbol{x}}$ and keeping $N_x = N_y = 120$, the the following square of the absolute value of the wave function $|\psi(x, y)|^2$ has been calculated:

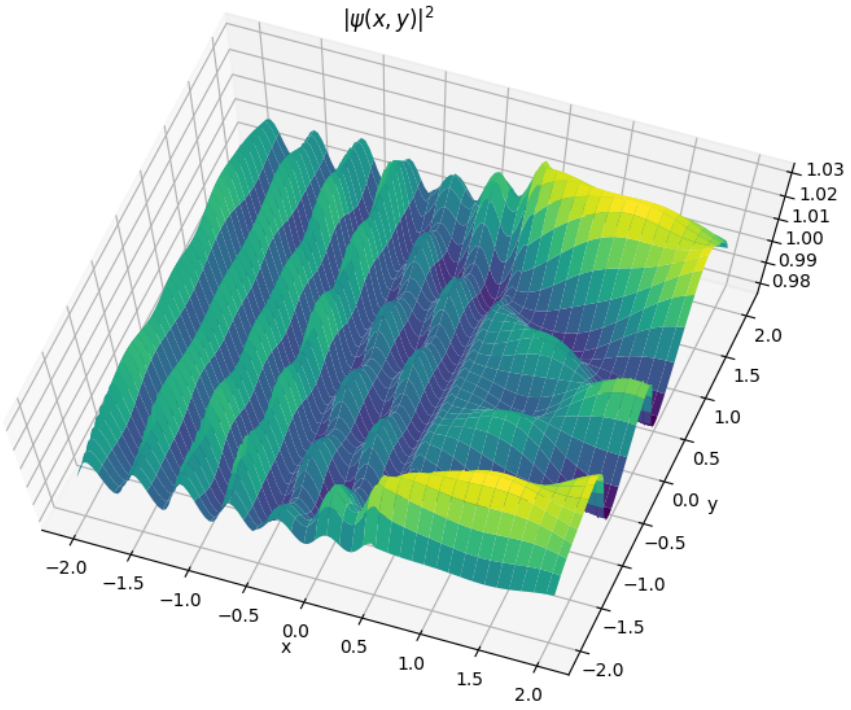

Fig. 6: Absolute value of wave function $|\psi(x,y)|^2$.
$\boldsymbol{k} = 8\,\hat{\boldsymbol{e}}_{\boldsymbol{x}}$ and $N_x = N_y = 120$. V as shown in figure 5.
Lighter colors (greenish yellow) depict higher values, darker ones (blueish turquoise) lower values. Not normalized.

Repeating this with $\boldsymbol{k} = 16\,\hat{\boldsymbol{e}}_{\boldsymbol{x}}$, a similar result can be obtained:

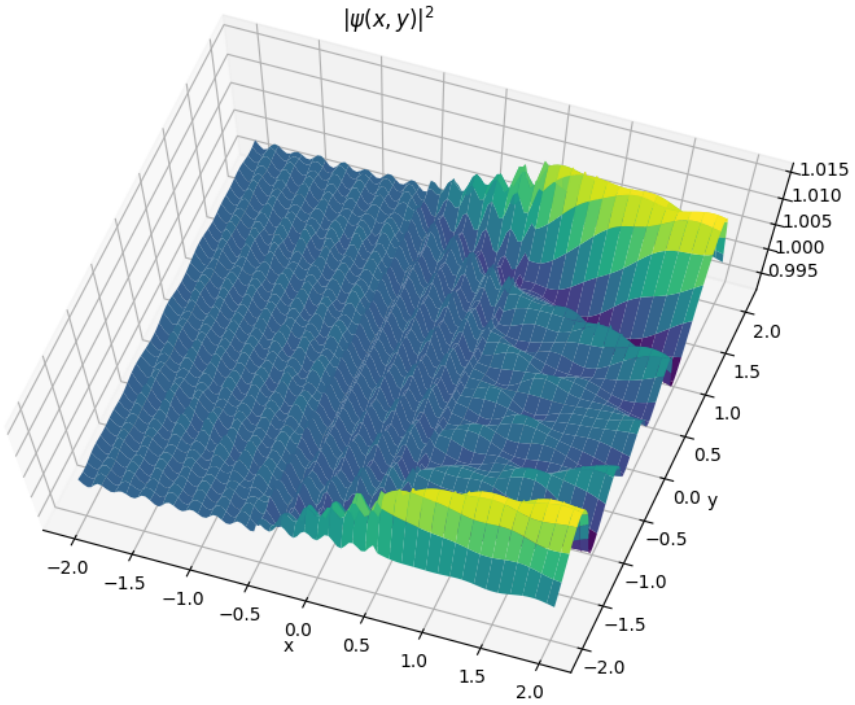

Fig. 7: Absolute value of wave function $|\psi(x,y)|^2$.
$\boldsymbol{k} = 16\,\hat{\boldsymbol{e}}_{\boldsymbol{x}}$ and $N_x = N_y = 120$. V as shown in figure 5.
Lighter colors (greenish yellow) depict higher values, darker ones (blueish turquoise) lower values. Not normalized.

The first thing that stands out in these images is the high amplitude of the wave function for positive x and $|y| > 1$. These mountains start at the corners of the potential well at $x = -0.5$ and move slowly inward. The source of these peaks can be interpreted as the wave function, which diffracts around the edges at $y = \pm 2$ and bends around the potential. This could happen because the simulation only takes the potential inside this $[-2, 2] \times [-2, 2]$ chamber into account, whereas the Lippmann-Schwinger equation describes the behavior of the wave function on the whole $\mathbb{R}^2$-plane. Further evidence for this explanation is provided by the angle, at which these extrema move towards the center. For $k = 8$, they close faster and reach almost to $y = \pm 0.5$ at the end of the simulation box, while for $k = 16$, they only reach to about $y = \pm 1.0$. This would be expected from the two wave functions with different energies.

Furthermore, one can see that the squared amplitude of the wave function

is slightly higher inside the trench (for $x \in [-0.5, 0.5]$). This corresponds to a higher probability amplitude in an area of lower energy. The difference is only this small because the energy of the wave is given by $E = \frac{k^2}{2}$ (see equation 53) which is a lot bigger than the depth of the potential.

Without going into detail, but just to demonstrate the possibility to use different potentials, a double slit potential will be simulated. It will have a small barrier with two small slits in the center. The following picture should again clarify the details.

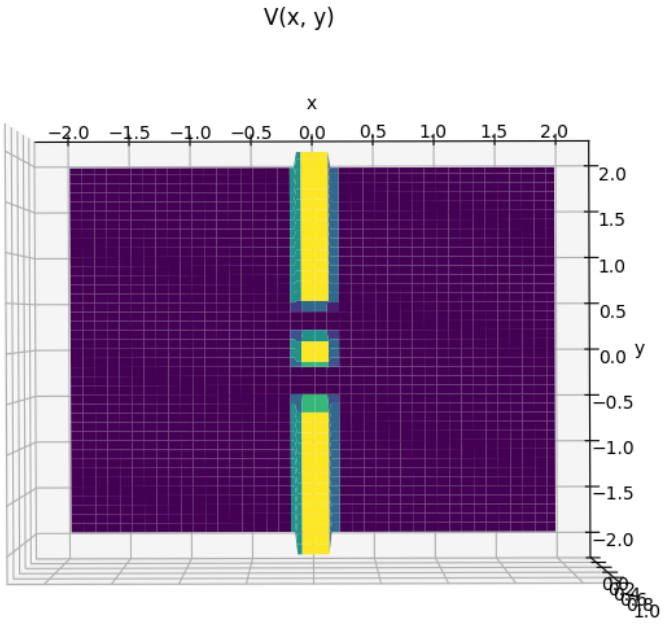

Fig. 8: Potential used for the double slit experiment.
In the purple area $V = 0$, everywhere else $V = 1$. $N_x = N_y = 120$.

For $\boldsymbol{k} = 16\,\hat{\boldsymbol{e}}_{\boldsymbol{x}}$, the following wave function has been computed. As in the example above, only the square of the absolute value of the wave function will be shown.

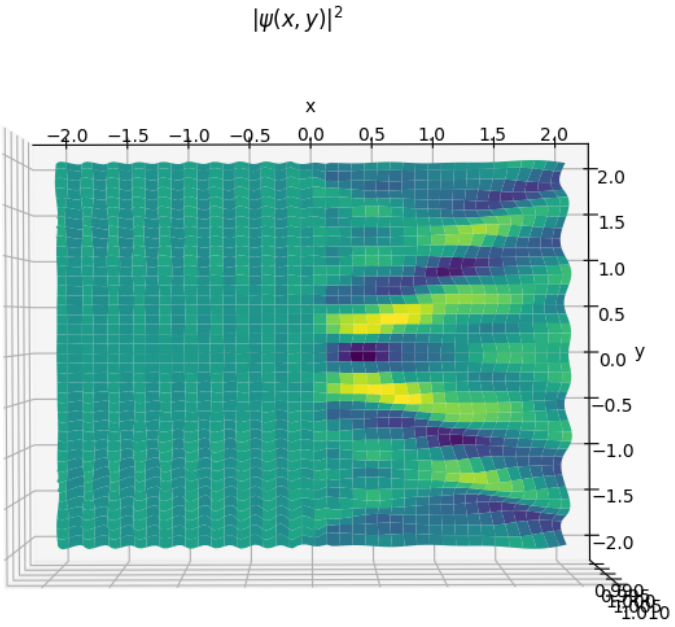

Fig. 9: $|\psi(x,y)|^2$ for the double slit potential displayed in figure 8.
$N_x = N_y = 120$, $\boldsymbol{k} = 16\,\hat{\boldsymbol{e}}_{\boldsymbol{x}}$.

In this picture the incoming parallel wave fronts on the left are clearly visible, as well as the parts of the wave function, that pass through the two slits. Identical to the cuboid trench potential shown before, the wave function leaks around the barrier and interferes with the parts of the wave function, that pass through the slits.

4 Performance and Improvements

As the 2-dimensional variant consumes a lot of computational power, the code has been optimized for speed and memory efficiency. Several tests and improvements have already been made, however, the end product is still definitely not perfect. For timing the code, a custom primitive timer class was written. It allows setting markers in the code and outputs the time passed between these markers using pythons `time` package.

The first test tries to improve the speed for the calculations of the distances between all points on the plane. Prior to using scipys `cdist`, a fourfold nested loop has been used. As the following output of the `timer` class shows, `cdist` proves far superior.

```
### Analysis of computational times: ###

                  4 loop:    113.303951 seconds
            cdist mgrid:      0.000136 seconds
          arrayification:     0.000054 seconds
              cdist calc:     0.179445 seconds
                printing:     0.197794 seconds

##########################
```

In this output, `4 loop` counts the time it takes for the 4 nested `for`-loops to calculate all the distances. `cdist mgrid` times the creation of the position matrices by numpys `meshgrid`. `arrayification` is the part, where the meshgrid-matrices get converted into 1-dimensional numpy arrays. `cdist calc` then calculates all possible distances and `printing` prints the results and checks if `cdist` and the `4 loop` variant both yield the same result. This test uses $N_x = N_y = 80$.

A similar test was performed for different variants of calculating the Hankel-function. Apart from `scipy.special.hankel1` also the variant with $J_0 + iY_0$ was implemented, where `scipy.special.jv` and `scipy.special.yn` were used to calculate J_0 and Y_0. As scipy is not the only package, which provides integrated bessel functions, also `mpmath.hankel1` was tried. This package, however, proved immensely slow compared to the scipy variants.

```
### Analysis of computational times: ###

          scipy hankel1:      0.002666 seconds
          scipy jv + yn:      0.001937 seconds
        mpmath hankel1:       4.239912 seconds

########################
```

In this test, 10^4 random real numbers in the interval $(0, 1)$ were fed in the three functions. It is plain to see, that mpmaths variant takes about 2000 times as long as the scipy methods. Even though the $J_0 + iY_0$ variant is a bit faster than the pure $H_0^{(1)}$ version, the `hankel1` function was prefered. Reason

for this is the ability to easily overwrite the input matrix with the output matrix of the `hankel1` function to save memory. This was crucial, as these were the biggest matrices in the whole program and needed several GiB of memory.

All these tests have been performed in separate programs to isolate the functions, which have been focused on. To conclude the digression on performance, the main program, which solves the Lippmann-Schwinger equation, will be analyzed. The following results are taken from the calculation of the double slit simulation with $N_x = N_y = 120$. Only the potential $V(x, y)$ and $|\psi(x,y)|^2$ are being plotted.

```
### Analysis of computational times: ###

                Beginning:      0.000261 seconds
          potential plot:       0.190887 seconds
        ynumerical init:        0.000017 seconds
   call of lse function:        0.000633 seconds
      spatial distance:         2.370444 seconds
                hankel:        80.085719 seconds
     setup of A matrix:         1.753537 seconds
     sys of eq solving:        67.217227 seconds
          reshape, ret:         0.078901 seconds
          plots at end:         0.068395 seconds

#########################

Total time needed: 151.766021 seconds
```

In this list, `Beginning` describes the pulling of the necessary information from the input file and setting up basic variables. As the label suggests, `potential plot` times the plotting of the potential $V(x, y)$. `ynumerical init` only takes the initialization of the final output matrix for the wave function into account, while `call of lse function` is the function call of the function, which solves the Lippmann-Schwinger equation with the given parameters. `spacial distance` is the time `cdist` took for calculating all possible distances and `hankel` the time it took for evaluating the hankel function for all these distances. In `setup of A matrix` the matrix A_{ln} is calculated and `sys of eq solving` times the solving of the system of linear equations. `reshape, ret` reforms the 1-dimensional wave function vector into the 2-dimensional matrix, which is then returned to the main program. In `plots at end` various plots can be specified to be drawn. In this example, however, only 1 plot has been ordered: $|\psi(x,y)|^2$.

As these numbers clearly suggest, the Hankel function and the solving of the system of linear equations are the only big hits on computation time. All other parts are negligible in comparison. Therefore, for improving the runtime of this code, one should focus on accelerating these two parts. Memory consumption wise, this program needs about 6.2 GiB for the 120x120 resolution and about 11.5 GiB for 140x140. A higher resolution has not been possible to test because of these high demands.

5 Summary

The Lippmann-Schwinger equation, which has been derived from the Schrödinger equation in section 1, describes the behavior of quantum mechanical waves in the presence of some potential. By means of numerical integration, this integral equation can be refactored into a system of linear equations and solved numerically (sec. 2.2 and sec. 3.2). In section 2.3 an implementation of this numerical way of solving the 1-dimensional variant is discussed. Similarly, the 2-dimensional implementation is described in section 3.3.

To sum up, the resulting wave functions behave as expected from quantum mechanical waves in their specific environment. This in turn suggests, that the program works as intended.

References

[1] Gordon Baym. *Lectures on Quantum Mechanics*. 4th ed. Reading, Massachusetts 01867, United States of America: W. A. Benjamin, Inc., 1969. ISBN: 0-805-30667-1.

[2] David J. Griffiths. *Introduction to Quantum Mechanics*. 1st ed. Upper Saddle River, New Jersey 07458, United States of America: Prentice Hall, Inc., 1995. ISBN: 0-13-124405-1.

[3] Klaus Höllig and Jörg Hörner. *Trapez Regel*. German. 2017. URL: https://mo.mathematik.uni-stuttgart.de/kurse/kurs5/seite20.html (visited on 06/23/2019).

[4] Klaus Jänich. *Funktionentheorie. Eine Einführung*. German. 6th ed. Springer, 2004. ISBN: 978-3-540-20392-6.

[5] Scaler and Cdang. 2009. URL: https://commons.wikimedia.org/wiki/File:Integration_num_rectangles_notation.svg (visited on 06/30/2019).

[6] Benjamin A. Stickler and Ewald Schachinger. "Numerical Integration". In: *Basic Concepts in Computational Physics*. Cham: Springer International Publishing, 2014, pp. 29–50. ISBN: 978-3-319-02435-6. DOI: 10.1007/978-3-319-02435-6_3. URL: https://doi.org/10.1007/978-3-319-02435-6_3.