Praca Dyplomowa Inżynierska

Adrian Rostek
205860

Wykorzystanie technologii webowych i języka Python do stworzenia aplikacji edukacyjnej z mechaniki kwantowej

Utilizing web technologies and Python language to create quantum physics educational application

Praca dyplomowa na kierunku:
Informatyka

Praca wykonana pod kierunkiem
dr. Andrzeja Zembrzuskiego
Instytut Informatyki Technicznej
Katedra Systemów Informatycznych

Warszawa, rok 2024

SZKOŁA GŁÓWNA GOSPODARSTWA WIEJSKIEGO

Wydział Zastosowań Informatyki i Matematyki

Oświadczenie Promotora pracy

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia tej pracy w postępowaniu o nadanie tytułu zawodowego.

Data ..... Podpis promotora .....

Oświadczenie autora pracy

Świadom/a odpowiedzialności prawnej, w tym odpowiedzialności karnej za złożenie fałszywego oświadczenia, oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. 2019 poz. 1231 z późn. zm.)

Oświadczam, że przedstawiona praca nie była wcześniej podstawą żadnej procedury związanej z nadaniem dyplomu lub uzyskaniem tytułu zawodowego.

Oświadczam, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną. Przyjmuję do wiadomości, że praca dyplomowa poddana zostanie procedurze antyplagiatowej.

Data ..... Podpis autora pracy .....

Streszczenie

Wykorzystanie technologii webowych i języka Python do stworzenia aplikacji edukacyjnej z mechaniki kwantowej

Celem niniejszej pracy było stworzenie aplikacji edukacyjnej do nauki zagadnień mechaniki kwantowej. Zagadnienia podzielone zostały na sześć rozdziałów, w których za pomocą interaktywnych wizualizacji, użytkownik może nauczyć się o istocie i zachowaniu funkcji falowej. Aplikacja przy użyciu frameworka Tauri, stworzona została z wykorzystaniem technologii webowych - języków HTML, CSS i TypeScript, ale korzysta również z języka Python do wykonywania obliczeń fizycznych.

Słowa kluczowe – Edukacja, Fizyka kwantowa, Funkcja falowa, Wizualizacja

Summary

Utilizing web technologies and Python language to create quantum physics educational application

The goal of this thesis was to create and educational app for learning quantum mechanics. Material was divided into six chapters in which a user can learn about importance and behaviour of wave function via interactive visualizations. Application was created with the use of Tauri framework and web technologies - HTML, CSS and TypeScript languages but it also uses Python langauge for physics calculations.

Keywords – Education, Quantum physics, Wave function, Visualization

Spis treści

1 Wstęp **9**

1.1 Cel i motywacja pracy 9

1.2 Tematyka i struktura pracy 9

2 Wykorzystane technologie **11**

2.1 Popularne technologie webowe – HTML, CSS i TypeScript 11

2.2 Biblioteki Chart.js i MathJax 11

2.3 Język Python 12

2.4 Framework Tauri 13

3 Podstawy teoretyczne **14**

3.1 Zagadnienia matematyki wyższej 14

3.2 Falowa natura materii 16

3.3 Równanie Schrödingera 16

3.4 Znajdowanie równania funkcji falowej 17

3.5 Cząstka swobodna 17

3.6 Nieskończona studnia potencjału 18

3.7 Skończona studnia potencjału 18

3.8 Próg potencjału 19

3.9 Bariera potencjału 21

4 Budowa i struktura aplikacji **22**

4.1 Struktura aplikacji 22

4.2 Interfejs 22

4.3 Typescript i Chart.js 22

4.4 Obliczenia fizyczne w Pythonie 26

4.5 Interfejs pomiędzy językiem TypeScript i Pythonem 28

4.6 Wdrożenie i dystrybucja aplikacji 30

4.7 Problemy, ograniczenia i możliwości rozwoju 31

5 Interfejs aplikacji **33**

5.1 Ekran główny i nawigacja 33

5.2 Interaktywna wizualizacja 34

5.3 Narrator 37

6 Podsumowanie i wnioski **39**

7 Bibliografia **40**

1 Wstęp

Stworzenie mechaniki kwantowej okazało się niezwykle istotne dla dzisiejszej cywilizacji. Zawdzięczamy jej m.in. tranzystory oraz reaktory jądrowe, bez których ciężko wyobrazić sobie dzisiejszą energetykę [1]. Mimo to jest to bardzo nieintuicyjny i przez większość ludzi niezrozumiały dział fizyki [2]. Na temat ten napisane zostały liczne publikacje [2] [3] [4], jednak profesjonalny język i matematyka wyższa mogą sprawić dużo trudności w zrozumieniu nawet podstawowych konceptów tej teorii.

1.1 Cel i motywacja pracy

Celem pracy jest stworzenie aplikacji, która ma ułatwić naukę zagadnień z zakresu mechaniki kwantowej. Zagadnienia przedstawiane są w interaktywny sposób celem podtrzymania uwagi i zainteresowania tym nietrywialnym tematem. Osiągnięte to zostało poprzez wykorzystanie szeregu prezentacji wizualnych, na których efekt końcowy bezpośredni wpływ ma użytkownik. Zastosowany został również samouczek, który te efekty odpowiednio tłumaczy.

Osobiście temat mechaniki kwantowej uważam za niezwykle ciekawy, więc napisanie tej pracy motywowane jest chęcią poszerzenia swojej wiedzy w tym obszarze, jak i zastosowaniu nabytej wiedzy informatycznej w stworzeniu praktycznego narzędzia. Za interesujące również uważam symulację funkcji falowej w przeciwieństwie do przypatrywania się statycznym jej wykresom na papierze czy w plikach pdf. Aplikacja kierowana jest do osób chcących nauczyć się wstępnych zagadnień mechaniki kwantowej, jednak bez konieczności sięgania po profesjonalną literaturę. Do pełnego zrozumienia wszystkich zagadnień potrzebna jest znajomość podstaw matematyki wyższej, jednak nawet bez tej wiedzy użytkownik może wynieść z aplikacji dużo nowych informacji. Może ona być więc użyteczna nie tylko dla studentów fizyki, ale też osób ciekawiących się tym tematem.

1.2 Tematyka i struktura pracy

Aplikacja przytacza kontekst historyczny dziedziny fizyki, jaką jest mechanika kwantowa, jak i opisuje korpuskularno-falową naturę cząstek. Główna część jednak skupia się na typowych rozwiązaniach równania Schrödingera niezależnego od czasu, a dokładniej dla:

- cząstki swobodnej,
- nieskończonej studni potencjału,
- skończonej studni potencjału,
- progu potencjału,
- bariery potencjału.

Poprzez te przypadki wytłumaczone jest skwantowanie stanów energetycznych, czyli przyjmowanie przez cząstkę, zamiast dowolnych, pewnych dyskretnych wartości energii. Wyjaśnione jest też zjawisko tunelowe, czyli przejście cząstki przez barierę potencjału o większej wysokości niż energia tej cząstki. Są to często omawiane zagadnienia w podręcznikach wprowadzających do mechaniki kwantowej [2] [3] [4], ponieważ przypadki te dobrze obrazują istotę funkcji falowej.

W rozdziale drugim opisane zostały zastosowane technologie, charakterystyka ich działania oraz wytłumaczone zostało, czym motywowany był wybór akurat ich do stworzenia aplikacji.

W rozdziale trzecim szerzej wyjaśniony został dokładny zakres zagadnień zawartych w aplikacji oraz uzasadniony został powód upraszczania niektórych z nich i poświęcanie większej uwagi na pozostałe.

Rozdział czwarty skupia się na technicznych aspektach budowy aplikacji, omawia szczegóły implementacji wymaganych rozwiązań i problemy z tym związane. Omówione również zostały ograniczenia zastosowanych technologii, jak i otwartość aplikacji na rozwój.

W rozdziale piątym zaprezentowane są zrzuty ekranu z działania aplikacji, wytłumaczona zostaje budowa i działanie interfejsu oraz jak spełnione zostało wstępne wymaganie aplikacji, czyli ułatwianie nauki.

Pracę kończy rozdział zawierający podsumowanie i wnioski.

2 Wykorzystane technologie

2.1 Popularne technologie webowe – HTML, CSS i TypeScript

Dzięki swojej popularności, technologie webowe wydały mi się najlepszym wyborem to stworzenia interfejsu aplikacji. HTML i CSS dają niezwykle dużo swobody i możliwości w tworzeniu interfejsów oraz dzięki ich popularności znaleźć można dużo literatury na ich temat [10] [11]. Nie ustępuje im w tym aspekcie język TypeScript, który jest nadzbiorem języka JavaScript - najpopularniejszego, według wielu źródeł, języka programowania. Dodatkowo TypeScript umożliwia łatwą modyfikację i dostęp do elementów z plików html, dzięki czemu w prosty sposób można tworzyć dynamiczne aplikacje webowe. Niewątpliwym atutem tego języka jest również duża liczba dostępnych frameworków [6] [7] [8] [9] ułatwiających pracę z tym językiem, przy tworzeniu aplikacji. Najważniejszą jednak cechą języka TypeScript jest, że w przeciwieństwie do popularnego języka JavaScript, stosuje on silne typowanie, co pozwala na uniknięcie błędów w typach używanych zmiennych [12].

2.2 Biblioteki Chart.js i MathJax

Chart.js jest aktualnie najpopularniejszą biblioteką do tworzenia wykresów w języku JavaScript [14], z bogatą dokumentacją i liczbą funkcji. Co było dla mnie najważniejsze przy jej wyborze, to szybka i czytelna konfiguracja wykresów liniowych. Biblioteka daje wiele możliwości dostosowywania wyglądu generowanych wykresów jak i ich dynamicznej edycji po wygenerowaniu.

Kolejnym ważnym aspektem jest dostępność wbudowanych w bibliotekę typów, co umożliwia stosowanie jej w języku TypeScript z wykorzystaniem zalet, jakie wprowadza on do języka JavaScript. We wstępnych wersjach aplikacji zastosowałem bibliotekę Plotly, jednak typy, pomimo że dla niej dostępne, nie są w pełni dopracowane, co wymuszało wyłączenie reguły no-explicit-any w konfiguracji języka TypeScript dla projektu. W konsekwencji tworzyło to sytuację, w której istniało ryzyko tworzenia kodu o słabym typowaniu, co w dłuższej perspektywie mogłoby sprzyjać pojawianiu się w kodzie błędów związanych z typami.

Drugą użytą biblioteką, jednak nie stosowaną bezpośrednio w kodzie TypeScript, a w plikach HTML, jest MathJax. Umożliwia ona umieszczanie zaawansowanych formuł matematycznych na stronach pisanych w języku HTML [15], co było konieczne dla poruszanych w aplikacji zagadnień. Formuły umieszczane są pomiędzy znakami $$ $$ lub \(\), a ich składnia, pomimo że trochę odmienna, jest bardzo zbliżona do popularnego LaTeX'a. Wszelkie odstępstwa od LaTeX'a można szybko zweryfikować dzięki szczegółowej dokumentacji. Istotny jest również fakt, że nie jest wymagane umieszczanie całej biblioteki lokalnie w projekcie, co zajmowałoby dosyć dużo miejsca. Zamiast tego wystarczy umieścić odpowiednią adnotację w pliku HTML, aby pobrać niezbędne zasoby z internetu.

2.3 Język Python

Python, pomimo że znacząco wolniejszy od wielu innych języków programowania (str. 2 w [13]), posiada bogatą kolekcję bibliotek do przetwarzania danych i obliczeń matematycznych m.in. NumPy, SciPy i SymPy. Biblioteki te, najczęściej napisane w C lub Fortranie, są już z kolei znacznie szybsze od czystego Pythona (str. 2 w [13]). Charakterystyczna jest też jego przejrzysta składnia, zawierająca dużo lukru składniowego (str. 418 w [13]).

Głównymi powodami, dla których użyłem języka Pythona, jest właśnie dostępność bibliotek, a szczególnie SymPy oraz wbudowana obsługa liczb zespolonych. Co prawda do rachunku na liczbach zespolonych wystarczyłby, już użyty w projekcie, język TypeScript, jednak składnia Pythona jest w tym zastosowaniu znacznie bardziej zwięzła. Biblioteka SymPy z kolei nie została wykorzystana w żadnym miejscu w implementacji aplikacji, więc może się wydać nietypowym argumentem za wyborem języka, jednak tłumaczę ten wybór dokładniej w kolejnym akapicie.

SymPy jest biblioteką umożliwiającą obliczenia w matematyce symbolicznej [18], co najważniejsze pozwala na znajdowanie analitycznych rozwiązań dla równań różniczkowych. W mojej opinii umożliwia to łatwiejszy rozwój aplikacji i wzbogacenie jej o bardziej zaawansowane przypadki ruchu cząstki w przestrzeni.

Dostępność Matplotlib i podobnych bibliotek szkicujących umożliwia rysowanie wykresów do wizualizacji danych, wymagając przy tym bardzo małej ilości kodu, co może okazać się niezwykle przydatne przy sprawdzaniu poprawności implementacji obliczeń bardziej złożonych wzorów matematycznych. Przy pisaniu pracy niejednokrotnie z takiej możliwości skorzystałem, na co najczęściej potrzebne były 3-4 linie kodu.

2.4 Framework Tauri

Za pomocą Tauri możliwe jest tworzenie aplikacji desktopowych, przy jednoczesnym wykorzystaniu technologii webowych. W przeciwieństwie do rozwiązań takich jak Electron, aplikacje tworzone przy użyciu Tauri zajmują znacznie mniej miejsca oraz potrzebują mniej zasobów komputera do działania. Dodatkowym atutem jest możliwość korzystania z interpretera Pythona, zainstalowanego na komputerze użytkownika. Standardowe aplikacje webowe w przeglądarce również umożliwiają wykonywanie skryptów Pythona, za pomocą interpretera skompilowanego do WebAssembly [29], języka programowania obsługiwanego przez popularne przeglądarki [16]. Jest to niestety rozwiązanie stosunkowo nowe i we wczesnej fazie rozwoju.

3 Podstawy teoretyczne

3.1 Zagadnienia matematyki wyższej

Niewątpliwie matematykę, a szczególnie rachunek różniczkowy, można nazwać językiem fizyki. Do opisu zawartych w pracy zagadnień fizycznych wymagana jest pewna znajomość zagadnień matematyki wyższej.

Liczby zespolone są nadzbiorem liczb rzeczywistych, wzbogaconym o jednostkę urojoną i [19] definiowaną jako:

$$i^2 = -1. \tag{3.1}$$

Dowolną liczbę zespoloną możemy więc zapisać w postaci algebraicznej, będącej sumą części rzeczywistej i urojonej, tj. będącej rzeczywistą wielokrotnością jednostki urojonej:

$$z = a + bi, \tag{3.2}$$

gdzie $z \in \mathbb{C}$, $a \in \mathbb{R}$, $b \in \mathbb{R}$.

Do graficznego opisu liczby zespolonej poza osią rzeczywistą, potrzebna jest dodatkowa, pionowa oś urojona. Liczba zespolona może być przedstawiona jako punkt o współrzędnych a i b w kartezjańskim układzie współrzędnych. Moduł liczby z definiowany jako odległość tego punktu od środka układu współrzędnych można więc obliczyć, stosując twierdzenie Pitagorasa:

$$|z| = \sqrt{a^2 + b^2}, \tag{3.3}$$

gdzie:
$z = a + bi$,
$z \in \mathbb{C}$, $a \in \mathbb{R}$, $b \in \mathbb{R}$.

Inną postacią liczby zespolonej jest postać wykładnicza, przedstawiana jako:

$$z = |z|e^{i\phi}, \tag{3.4}$$

gdzie:

$z \in \mathbb{C}$,

e – podstawa logarytmu naturalnego,

i – jednostka urojona,

ϕ – kąt pomiędzy osią części rzeczywistej, a ramieniem poprowadzonym ze środka układu współrzędnych do liczby z.

Szczegóły funkcji falowej ψ wytłumaczone są w dalszej części pracy, jednak na szczególną uwagę zasługuje wartość kwadratu jej modułu, określająca gęstość prawdopodobieństwa znalezienia cząstki w danym położeniu. Oznacza to, że prawdopodobieństwo znalezienia cząstki w jednowymiarowym obszarze $[x_1, x_2]$ będziemy wyznaczać jako:

$$P(X) = \int_{x_2}^{x_1} |\psi(x)|^2 dx, \tag{3.5}$$

gdzie:

X – zdarzenie losowe, że cząstka znajduje się w obszarze,

x_1, x_2 – początek i koniec obszaru.

Równanie różniczkowe to równanie funkcyjne zawierające pochodne nieznanej funkcji. Znajdowanie rozwiązań takich równań znacząco wykracza poza wiedzę wymaganą do zrozumienia omawianych zagadnień fizycznych, więc nie będzie do tego przywiązywana szczególna uwaga. Wybrany zakres zagadnień wymaga jedynie zapamiętania rozwiązania jednego rodzaju równań:

$$f''(x) + k^2 f(x) = 0 \iff f(x) = Ae^{ikx} + Be^{-ikx}, \tag{3.6}$$

gdzie:

$f(x)$ *i* $f''(x)$ – funkcja i jej druga pochodna względem zmiennej x,

$k \in \mathbb{C}$,

$A \in \mathbb{C}, B \in \mathbb{C}$ – stałe wyznaczane z warunków brzegowych.

Warto zaznaczyć, że pomimo tego, że zagadnienia te są niezbędne do pełnego zrozumienia mechaniki kwantowej, nawet bez tej wiedzy użytkownik aplikacji może dowiedzieć się o tym, jak nieintuicyjnie zachowują się cząstki.

3.2 Falowa natura materii

Wyjaśnione przez Alberta Einsteina zjawisko fotoelektryczne ukazało, że światło zachowuje się nie tylko jak fala, ale też jak cząstką. Nośnik światła przekazywał energię porcjami – kwantami przez co wprowadzone zostało pojęcie fotonu jako cząstki światła [2] [3] [4] [20].

Jako konsekwencja tego odkrycia Louis de Broglie zapostulował, że cząstki materii, takie jak elektron, muszą podzielać falowe zachowanie fotonów. Korzystając z pracy Einsteina określił długość tych fal [21], zwanych falami materii, wzorem:

$$\lambda = \frac{h}{p}, \tag{3.7}$$

gdzie:

λ – długość fali materii cząstki,

h – stała Plancka,

p – pęd cząstki.

Dzięki falom materii możemy wyjaśnić m.in. sposób rozchodzenia się elektronów np. przez siatkę dyfrakcyjną, co jest niemożliwe przy przyjęciu elektronów jako kul lub punktów w przestrzeni.

Dokładniejszego opisu tych fal dokonał Erwin Schrödinger, proponując równanie funkcji falowej ψ o zespolonym zbiorze wartości.

3.3 Równanie Schrödingera

Funkcja falowa stanowi fundament zagadnień poruszanych w aplikacji. Wynika to z faktu, że jest ona niezbędna do opisu ruchu dowolnej cząstki. Erwin Schrödinger zawarł funkcję falową w równaniu nazwanym od jego nazwiska równaniem Schrödingera [22]. Jeżeli skupimy się na odizolowanych układach fizycznych, tj. takich, które nie oddziałują z otoczeniem i ich energia jest stała, rozwiązać można równanie Schrödingera niezależne od czasu o postaci:

$$-\frac{\hbar}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x) = E\psi(x), \tag{3.8}$$

gdzie:

$\hbar$ – zredukowana stała Plancka,

$\psi(x)$ – zespolona funkcja falowa zależna od położenia x,

E – energia całkowita ciała,

$V(x)$ – potencjał w położeniu x,

m – masa ciała.

Zespolona wartość funkcji falowej $\psi(x)$ nie posiada fizycznej interpretacji, do tego potrzebny jest kwadrat jej modułu, który określa gęstość prawdopodobieństwa znalezienia cząstki w położeniu x.

3.4 Znajdowanie równania funkcji falowej

Określenie położenia cząstki wymaga od nas rozwiązania równania Schrödingera, w którym będziemy mieli określoną funkcję $V(x)$. Rozwiązanie szczegółowe tego równania spełniać musi warunki początkowe, mogące wynikać m.in. z normowania gęstości prawdopodobieństwa. Poza warunkami początkowymi, konieczne będzie też spełnienie warunków brzegowych tj. ciągłości $\psi(x)$ oraz, w zagadnieniach bez nieskończonego potencjału, ciągłości jej pochodnej. Ważne jest również, że tylko skończone wartości $\psi(x)$ mają fizyczny sens [2].

3.5 Cząstka swobodna

Zanim zajmiemy się bardziej złożonymi przykładami, sprawdźmy jaką postać przyjmuje ψ dla cząstki swobodnej, tj. o zerowej energii potencjalnej $V(x)$, która porusza się w stronę dodatnich wartości x. Otrzymujemy równanie

$$\frac{d^2\psi}{dx^2} + k^2\psi = 0, \tag{3.9}$$

gdzie

$$k = \frac{1}{\hbar}\sqrt{2mE}. \tag{3.10}$$

Rozwiązanie ogólne ma postać

$$\psi(x) = Ae^{ikx} + Be^{-ikx}, \tag{3.11}$$

gdzie stała B musi być równa 0, ponieważ drugi wyraz opisuje falę poruszającą się w stronę ujemnych wartości x. Finalnie równanie przyjmuje postać:

$$\psi(x) = Ae^{ikx}, \tag{3.12}$$

w którym stałą A można wyznaczyć, jeśli chciałoby się znormalizować gęstość prawdopodobieństwa $|\psi|^2$, na określonym obszarze. Widać jednocześnie, że gęstość prawdopodobieństwa jest jednolita w całym takim obszarze.

3.6 Nieskończona studnia potencjału

Zaczynając od przypadku cząstki poruszającej się w nieskończonej studni potencjału o szerokości a, dzielimy zagadnienie na trzy obszary. Otrzymujemy obszar I, gdzie x$\in \left(-\infty, -\frac{a}{2}\right\rangle$, obszar II, gdzie x$\in \left\langle -\frac{a}{2}, \frac{a}{2}\right)$ oraz obszar III, gdzie x$\in \left\langle \frac{a}{2}, +\infty\right)$. W obszarach I i III potencjał jest nieskończony, więc jeśli $\psi \neq 0$, to $\frac{d^2\psi}{dx^2}$ musi być nieskończone, jednak nie ma to fizycznego sensu, więc ψ musi być równe 0 w tych obszarach. Dla obszaru II otrzymujemy równanie 3.9 oraz rozwiązanie 3.11. Na potrzebę tego przypadku rozwiązanie przekształcamy na równanie

$$\psi(x) = A\sin(kx) + B\cos(kx), \tag{3.13}$$

mając na uwadze, że współczynniki A i B nie są równe tym z równania 3.11. Ciągłość ψ pomiędzy obszarami jest możliwa tylko wtedy, gdy funkcja ta będzie równa 0 na granicach studni, co jest możliwe tylko dla $A = 0$ lub $B = 0$, a szerokość studni a jest całkowitą wielokrotnością długości tej fali. Zależność tę można zapisać jako

$$ka = n\pi, \text{ gdzie } n \in \{1,2,3...\}. \tag{3.14}$$

Rozwiązaniem jest więc

$$\begin{aligned} \psi &= Asin(kx) \quad \text{dla} \quad n \in \{2,4,6...\}, \\ \psi &= Bcos(kx) \quad \text{dla} \quad n \in \{1,3,5...\}. \end{aligned} \tag{3.15}$$

3.7 Skończona studnia potencjału

W przypadku studni skończonej postępujemy bardzo podobnie, dzieląc zagadnienie na trzy obszary, jednak przyjmujemy, że potencjał wynosi 0 dla obszarów I i III oraz $-V_0$ dla obszaru II. Jako W oznaczymy energię wiązania, tj. dodatnią wartość wymaganą do uwolnienia cząstki ze studni. W obszarze II równanie Schrödingera znowu przyjmie postać 3.9 z rozwiązaniem 3.15 z tą różnicą, że:

$$k = \frac{1}{\hbar}\sqrt{2m(V_0 - W)}. \tag{3.16}$$

Dla obszarów I i III równanie zapiszemy jako

$$\frac{d^2\psi}{dx^2} - \gamma^2\psi = 0, \tag{3.17}$$

gdzie

$$\gamma = \frac{1}{\hbar}\sqrt{2mW}, \tag{3.18}$$

co daje rozwiązanie ogólne

$$\psi_{I,III} = Ce^{\gamma x} + De^{-\gamma x}. \tag{3.19}$$

Celem znalezienia dozwolonych wartości k dopasowujemy na granicach obszarów pochodne logarytmiczne:

$$\frac{1}{\psi_I}\frac{d\psi_I}{dx} = \frac{1}{\psi_{II}}\frac{d\psi_{II}}{dx} \text{ w } x = -\frac{a}{2} \tag{3.20}$$

oraz

$$\frac{1}{\psi_{II}}\frac{d\psi_{II}}{dx} = \frac{1}{\psi_{III}}\frac{d\psi_{III}}{dx} \text{ w } x = \frac{a}{2}. \tag{3.21}$$

Z obydwu równań otrzymujemy te same wnioski:

$$\begin{aligned} \gamma &= k\cot\left(-\frac{ka}{2}\right) \text{ dla } B = 0, \\ \gamma &= -k\tan\left(-\frac{ka}{2}\right) \text{ dla } A = 0. \end{aligned} \tag{3.22}$$

Są to warunki konieczne do spełnienia, aby otrzymać poprawny wzór funkcji ψ. Zarówno we wzorach na k jak i γ znajduje się wartość energii wiązania W, więc zagadnienie sprowadza się do znalezienia takich wartości W, aby powyższe warunki były spełnione. Nie jest to problem dający się w łatwy sposób rozwiązać analitycznie, jednak pokazuje, że dla skończonej studni też istnieją pewne dyskretne stany energetyczne cząstki. W rozdziale czwartym szerzej opisałem znajdowanie tych wartości metodą numeryczną.

3.8 Próg potencjału

Prostokątnym progiem potencjału nazwiemy sytuację, w której w punkcie wartość energii potencjalnej rośnie z 0 do jakiejś wartości oznaczonej V_0. Przyjmując, że próg potencjału znajduje się w punkcie $x = 0$, możemy zapisać:

$$\begin{aligned} V(x) &= 0 \text{ dla } x \in (-\infty, 0), \\ V(x) &= V_0 \text{ dla } x \in \langle 0, +\infty). \end{aligned} \tag{3.23}$$

Tym razem zagadnienie dwa obszary - obszar I, gdzie $V(x) = 0$ oraz obszar II, gdzie $V(x) = V_0$. W obu tych obszarach możemy policzyć liczbę falową k jako:

$$k_I = \frac{1}{\hbar}\sqrt{2mE}, \tag{3.24}$$

$$k_{II} = \frac{1}{\hbar}\sqrt{2m(E - V_0)}. \tag{3.25}$$

Niech w stronę progu porusza się strumień cząstek o energii całkowitej $E > V_0$, opisany wyrażeniem $e^{ik_I x}$. Warunkiem koniecznym do spełnienia równania Schrödingera, jest założenie, że część cząstek odbije się od progu, a część przejdzie za niego. Funkcje falowe dla tych obszarów możemy więc zapisać jako:

$$\begin{aligned} \psi_I(x) &= e^{ik_I x} + Re^{-ik_I x}, \\ \psi_{II}(x) &= Te^{ik_{II} x}. \end{aligned} \tag{3.26}$$

Z warunków brzegowych, czyli ciągłości ψ i jej pochodnej po x w punkcie $x = 0$, możemy wyznaczyć współczynniki R i T, rozwiązując otrzymany układ równań:

$$\begin{cases} 1 + R = T, \\ k_I - k_I R = k_{II} T, \end{cases} \tag{3.27}$$

z którego otrzymujemy rozwiązanie

$$\begin{cases} T = \frac{2k_I}{k_I + k_{II}}, \\ R = \frac{k_I - k_{II}}{k_I + k_{II}}. \end{cases} \tag{3.28}$$

W przypadku, gdy $E < V_0$ podchodzimy do rozwiązania tego problemu tak samo, z tą tylko różnicą, że k_{II} przyjmuje wartość urojoną. Z tego tez powodu, możemy zapisać ψ_{II} bez jednostki urojonej w wykładniku liczby e, czyli

$$\psi_{II}(x) = Te^{\kappa x}, \tag{3.29}$$

gdzie $\kappa = -|k_{II}|$. Pomimo, że T też jest teraz urojone, w funkcji pojawia się rzeczywista funkcja wykładnicza, która szybko dąży do 0 dla x zmierzającego do nieskończoności. Zobaczyć więc można, że $|\psi|^2$ za barierą jest dodatnie, a co za tym idzie szansa na znalezienie cząstki za barierą jest niezerowa, chociaż szybko maleje w miarę oddalania się od niej. Cząstka może się w ten sposób pojawić w obszarze wzbronionym klasycznie.

3.9 Bariera potencjału

Jako ostatni omawiany jest przypadek bariery potencjału, w którym energia potencjalna zmienia się jak w progu potencjału, jednak przyjmujemy, że w punkcie $x = a$ spada poniżej V_0. Mamy więc trzy obszary o różnych energiach potencjalnych i tak jak w przypadku studni potencjału będziemy je numerować I, II i III. Zaczynając od sytuacji, gdy energia cząstki $E > V_0$ i pamiętając z rozwiązywania przypadku progu potencjału o tym, że część cząstek odbije się w miejscach zmiany energii potencjalnej, możemy zapisać funkcje falowe jako:

$$\begin{aligned} \psi_I(x) &= e^{ik_I x} + Re^{-ik_I x}, \\ \psi_{II}(x) &= Ce^{ik_{II} x} + De^{-ik_{II} x}. \\ \psi_{III}(x) &= Te^{ik_{III} x}. \end{aligned} \tag{3.30}$$

Znowu wymagamy, aby funkcja falowa i jej pochodna po x były ciągłe, tylko tym razem zarówno w punkcie $x = 0$ jak i $x = a$. Możemy to zapisać jako

$$\begin{cases} 1 + R = C + D \\ k_I - k_I R = k_{II} C - k_{II} D \\ Ce^{ik_{II} a} + De^{-ik_{II} a} = Te^{ik_{III} a} \\ k_{II} Ce^{ik_{II} a} - k_{II} De^{-ik_{II} a} = k_{III} Te^{ik_{III} a}. \end{cases} \tag{3.31}$$

Rozwiązując powyższy układ równań, możemy znaleźć wartości wszystkich stałych, z których wartość T umożliwi nam obliczenie gęstości prawdopodobieństwa $|T|^2$, mówiącej jaka część cząstek przejdzie przez barierę potencjału.

Jeśli będziemy chcieli rozważyć przypadek, kiedy energia cząstki $E < V_0$, postąpimy dokładnie tak samo, tylko wielkość k_{II} przyjmie wartość urojoną. Gęstość prawdopodobieństwa $|T|^2$ i w tym przypadku jest dodatnia, a co za tym idzie istnieje niezerowe prawdopodobieństwo przejścia cząstek przez barierę, pomimo energii zbyt małej, według fizyki klasycznej, na jej przekroczenie.

4 Budowa i struktura aplikacji

4.1 Struktura aplikacji

W celu uporządkowania struktury plików, pliki html, css, ts, py zostały umieszczone w oddzielnych folderach, poświęconych danemu rodzajowi plików. Fonty i ikony znajdziemy w folderze assets. Ważniejszy od umieszczenia plików jest jednak fakt, że framework Tauri domyślnie do kompilacji aplikacji używa tylko plików webowych (html, css, ts), a skrypty Pythona nie są dołączane. W celu dołączenia plików Pythona do gotowej aplikacji, w pliku konfiguracyjnym tauri.conf.json włączone zostało API fs, a folder ze skryptami Pythona oznaczony został jako folder z zasobami.

4.2 Interfejs

Aplikacja składa się z wielu widoków, każdy napisany jako oddzielny plik html. W każdym z tych plików, poza plikiem index.html będącym widokiem ekranu głównego, zaimportowana jest biblioteka MathJax.js z zewnętrznego serwera. Ponieważ widoki są mocno zbliżone wyglądem, korzystają one z tego samego pliku arkusza stylów styles.css. Dodatkowo ponieważ użyte wizualizacje, opisane szerzej w rozdziale 5.2, wymagają różnych elementów do interakcji, dla każdej z nich został stworzony oddzielny, krótki plik css. W plikach css użyta została metodologia BEM [23] do nazywania klas.

4.3 Typescript i Chart.js

Aplikacja zawiera wiele różnych wykresów, na których dynamicznie muszą być pokazywane i chowane odpowiednie elementy. Pomimo że wykresy wyglądają podobnie, to ciężko znaleźć cechy wspólne dla ich wszystkich, z tego powodu nie próbowałem stosować programowania obiektowego przy ich tworzeniu. Każdy wykres podzieliłem na trzy pliki: chart.ts, data.ts, options.ts.

Zaczynając od ostatniego, plik options.ts zawiera jedną stałą typu ChartOptions, służącą do konfiguracji wykresu tworzonego za pomocą biblioteki Chart.js. Zawiera on m.in. opisy osi, zakres osi, kolor użytego fontu, ale wyłącza on też kilka domyślnych ustawień, takich

jak siatka na wykresie oraz wyświetlanie współrzędnych punktów, co można zobaczyć na rysunku 4.1.

```
import { ChartOptions } from "chart.js/auto"

export const options: ChartOptions = {
  scales: {
    x: {
      title: {
        display: true,
        text: "X",
        color: "#FFFFFF",
        align: "center",
        font: {
          size: 24,
        },
      },
      type: "linear",
      min: -3e-9,
      max: 3e-9,
      ticks: {
        color: "#FFFFFF",
        font: {
          size: 18,
        },
        display: true,
        callback: (val, _) => {
          if (val === -1e-9) return "-a/2"
          if (val === 1e-9) return "a/2"
          return ""
        },
      },
      grid: {
        display: false,
      },
      border: {
        color: "#FFFFFF",
      },
    },
    y: {
      title: {
        display: true,
        text: "E",
        color: "#FFFFFF",
        align: "center",
        font: {
          size: 24,
        },
      },
      max: 1.5,
      min: -1.5,
      ticks: {
        color: "#FFFFFF",
```

Rysunek 4.1. Początek pliku options.ts dla wykresu skończonej studni potencjału

W pliku data.ts znajdziemy jedną stałą, widoczną na rysunku 4.2, która określa ile szkiców funkcji ma pojawić się na wykresie, jak i współrzędne punktów. Dodatkowo znajdziemy w tym pliku kolory wykresów oraz ich opis w legendzie.

```
export const data = {
  datasets: [
    {
      label: "Energia potencjalna",
      fill: false,
      borderColor: "rgb(50,50,200)",
      backgroundColor: "rgb(50,50,200)",
      tension: 0,
      data: [
        { x: -5e-9, y: 0 },
        { x: -1e-9, y: 0 },
        { x: -1e-9, y: -1 },
        { x: 1e-9, y: -1 },
        { x: 1e-9, y: 0 },
        { x: 5e-9, y: 0 },
      ],
    },
    {
      label: "Energia całkowita",
      fill: false,
      borderColor: "rgb(20, 255, 20)",
      backgroundColor: "rgb(20, 255, 20)",
      tension: 0,
      data: [],
    },
    {
      label: "ψ",
      fill: false,
      borderColor: "rgb(226,47,47)",
      backgroundColor: "rgb(226,47,47)",
      tension: 0.01,
      borderWidth: 5,
      data: [],
    },
    {
      label: "|ψ|²",
      fill: false,
      borderColor: "#FF8800",
      backgroundColor: "#FF8800",
      tension: 0.01,
      borderWidth: 5,
      data: [],
    },
  ],
}
```

Rysunek 4.2. Plik data.ts dla wykresu skończonej studni potencjału

W pliku chart.js importowane są stałe z dwóch poprzednich plików, w celu utworzenia wykresu, który to proces widać na rysunku 4.3.

```
export var quantumFiWellChart: Chart
;(async function () {
  const config: ChartConfiguration = {
    type: "line",
    data: data,
    options: options,
  }

  const chartElement: HTMLCanvasElement = <HTMLCanvasElement>(
    document.getElementsByClassName("quantum-fi-well-chart")[0]
  )

  quantumFiWellChart = new Chart(chartElement, config)
  await calcData()
})()
```

Rysunek 4.3. Tworzenie wykresu skończonej studni potencjału

Poza tym znajduje się tutaj cała logika wykresu, czyli wybór elementu canvas, do wyświetlenia wykresu, ale co ważniejsze wywoływanie funkcji, do pozyskania danych do naszkicowania (rysunek 4.4). Proces ten został szerzej opisany w rozdziale 4.5.

```
async function calcData() {
  await fiWellData(
    (x0, x1, re, psiSq, E) => {
      En = E
      scaleDown(En, 1.6e-19)
      Ren = re
      Psi_sqn = psiSq
      setRange(".quantum-total-energy", 0, En.length - 1)
      updateChart()
    },
    -3e-9,
    3e-9,
    2e-9, //4
    1.6e-19
  )
}
```

Rysunek 4.4. Wywołanie funkcji do obliczeń fizycznych i przedstawienia ich na wykresie

Dodatkowo w pliku chart.js znajdują się wszelkie ustawienia do reagowania na przyciski i suwaki w aplikacji, przy pomocy wbudowanego wzorca projektowego obserwatora, co przedstawiłem na rysunkach 4.5 i 4.6.

```
function updateChart() {
  const i = vValue(sliderName)
  quantumFiWellChart.data!.datasets[1]!.data = En[i]
  quantumFiWellChart.data!.datasets[2]!.data = Ren[i]
  quantumFiWellChart.data!.datasets[3]!.data = Psi_sqn[i]
  quantumFiWellChart.update("show")
}
```

Rysunek 4.5. Funkcja do aktualizacji wykresu po wprowadzeniu zmian

```
const energySlider = <HTMLInputElement>document.querySelector(".quantum-total-energy")

const info = document.querySelector(".info__val")

energySlider.addEventListener("input", () => {
  updateChart()
  info!.innerHTML = String(vValue(sliderName) + 1)
})
```

Rysunek 4.6. Kod odpowiedzialny za wywołanie aktualizacji wykresu po przesunięciu suwaka

W projekcie znaleźć też można kilka plików, które ciężko przypisać do jakiejś konkretnej grupy. W pliku physics-constants.ts znajdują się wartości stałych fizycznych użytych w projekcie. Obecny jest również plik python-version-check.ts, odpowiedzialny za sprawdzanie obecności i wersji zainstalowanego interpretera Pythona oraz biblioteki NumPy.

4.4 Obliczenia fizyczne w Pythonie

Implementacja obliczeń zagadnień cząstki swobodnej, nieskończonej studni potencjału i progu potencjału, wymaga jedynie przepisania równań 3.11, 3.15 i 3.26 do Pythona, co nie wymaga dłuższego tłumaczenia.

Sytuacja jest już odmienna dla pozostałych zagadnień, zaczynając od skończonej studni potencjału. Istnieją w niej bowiem, tak jak w studni nieskończonej, skwantowane stany energii dla cząstki związanej, jednak wyznaczenie ich nie wyraża się prostym wzorem. Jak jest wyjaśnione w [24], na podstawie wzorów 3.16, 3.18, 3.22 możemy zagadnienie skończonej studni, sprowadzić do jednej zmiennej v :

$$\sqrt{u_0^2 - v^2} = \begin{cases} v\tan v \\ -v\cot v \end{cases}, \tag{4.1}$$

gdzie:

$$u_0^2 = \frac{ma^2 V_0}{2\hbar^2},$$

$$v = \frac{ka}{2}.$$

Liczbę rozwiązań równania 4.1 możemy wyznaczyć jako:

$$N = \left\lceil \frac{2u_0}{\pi} \right\rceil, \tag{4.2}$$

a każdego z nich należy szukać w przedziałach

$$v_i \in \left\langle \frac{\pi}{2}(i-1),\ \frac{\pi}{2}i \right) \ \text{ dla } \ i = 1,2,...N \tag{4.3}$$

Ze względu na szybkość początkowo pomyślałem o zastosowaniu metody Newtona [25], jednak metoda ta nie daje możliwości znalezienia miejsca zerowego w zadanym przedziale. Użyłem więc metody bisekcji [26], co przedstawiłem na rysunku 4.7, a różnica czasu w wykonywaniu tych dwóch metod okazała się w moim przypadku marginalna.

```
def _zero(x1, x2, u0_2, even):
    eps = abs(x2-x1)/1000

    while True:
        s = (x1+x2)/2

        if not even:
            y = s*tan(s) - sqrt(u0_2-s**2)
        else:
            y = -s/tan(s) - sqrt(u0_2-s**2)

        if abs(y) ⩽ eps:
            return s

        if y < 0:
            x1 = s
        else:
            x2 = s
```

Rysunek 4.7. Implementacja metody bisekcji

Warto jednocześnie zaznaczyć, że wyjątkowej uwagi wymaga znalezienie ostatniego rozwiązania tj. v_N. Prawa granica przedziału 4.3 może być na tyle oddalona od rozwiązania, że wartość ze środka tego przedziału, którą w pierwszej iteracji wybiera algorytm bisekcji, wyniesie więcej niż u_0, co spowoduje że w równaniu 4.1 pierwiastkowana będzie liczba ujemna. Ponieważ szukane rozwiązania muszą być wartościami rzeczywistymi, przy ostatniej iteracji prawą granicę przedziału, w którym szukane jest miejsce zerowe, ustalam jako u_0, co można zobaczyć na rysunku 4.8.

```python
def _possible_v(a, m, V0, h_):
    u0_2 = (m*a**2)/(2*h_**2)*V0
    u0 = sqrt(u0_2)

    max_n = ceil(2*u0/pi)

    v = []

    for n in range(1, max_n):
        v_min = pi/2*(n-1)
        v_max = pi/2*n

        v.append(_zero(v_min, v_max, u0_2, n % 2 == 0))

    # last solution
    v_min = pi/2*(max_n-1)
    v_max = u0
    v.append(_zero(v_min, v_max, u0_2, max_n % 2 == 0))

    return v
```

Rysunek 4.8. Funkcja znajdująca wszystkie wartości v

Kolejnym przypadkiem, wymagającym dodatkowej pracy, jest obliczenie stałych z układu równań 3.31 dla bariery potencjału. Pomimo że jest to układ liniowy, mnogość wyrazów oraz dziedzina liczb zespolonych powodują, że wygodniej było mi go rozwiązać przy pomocy biblioteki NumPy, co przedstawiłem na rysunku 4.9

```python
A = [
    [1, -1, -1, 0],
    [-k1, -k2, k2, 0],
    [0, e**(1j*k2*a), e**(-1j*k2*a), -e**(1j*k3*a)],
    [0, k2*e**(1j*k2*a), -k2*e**(-1j*k2*a), -k3*e**(1j*k3*a)]
]
b = [-1, -k1, 0, 0]
R, C, D, T = linalg.solve(A, b)
```

Rysunek 4.9. Implementacja numerycznego rozwiązania układu równań 3.31

4.5 Interfejs pomiędzy językiem TypeScript i Pythonem

Języki programowania tak odmienne jak TypeScript i Python uruchamiane są w zupełnie innych środowiskach. Framework Tauri daje tutaj wyjątkową możliwość, do obsługi języka TypeScript przy użyciu WebView [27] i Pythona przy wykorzystaniu lokalnie zainstalowanego interpretera [28].

Problemem jest jednak przesyłanie danych między tymi językami, o czym nie myślano przy ich tworzeniu. Zastosowałem więc serializację danych do formatu JSON, który jest tekstową reprezentacją obiektu w języku TypeScript. Wyniki obliczeń fizycznych są więc serializowane w Pythonie jedną z metod z rysunku 4.10, czego przykład widać na rysunku 4.11. Dalej wyniki wysyłane są jako wyjście w powłoce systemowej i przekazywane przez Tauri do skryptu napisanego w języku TypeScript, co można zobaczyć na rysunku 4.12.

```python
from json import dumps


def to_json(x: list[float], **y: list[float]) -> str:
    result = {}

    for key, val in y.items():
        data = [{"x": x0, "y": y0} for x0, y0 in zip(x, val)]
        result[key] = data
    return dumps(result)


def to_json_bulk(x: list[float], E: list[float], **y: list[list[float]]) -> str:
    result = {}
    result["E"] = [[{"x": x[0], "y": En}, {"x": x[-1], "y": En}] for En in E]

    for key, val in y.items():
        data = [[{"x": x0, "y": y0} for x0, y0 in zip(x, yn)] for yn in val]
        result[key] = data
    return dumps(result)
```

Rysunek 4.10. Kod z pliku json_export.py

```python
from numpy import linspace
from math import sqrt, sin, cos
from json_export import to_json
from sys import argv


def free_particle(x0, x1, m, v, h_, samples=1000):

    x = linspace(x0, x1, samples)
    # A^2 * (x1-x0) = 1
    A = sqrt(1/abs(x1-x0))
    k = 1/h_ * m * v
    re = [A*cos(xn*k) for xn in x]
    im = [A*sin(xn*k) for xn in x]
    # A = 1 ⇒ A^2 = A = 1
    psi_sq = [A for _ in x]

    return to_json(
        x,
        re=re,
        im=im,
        psi_sq=psi_sq
    )


print(free_particle(float(argv[1]), float(argv[2]), float(
    argv[3]), float(argv[4]), float(argv[5])))
```

Rysunek 4.11. Wykorzystanie funkcji to_json w pliku free_particle.py

```
const command = new Command("run-python", [
  "python/free_particle.py",
  String(x0),
  String(x1),
  String(m),
  String(v),
  String(h_),
])

command.stdout.on("data", (text) => {
  const data: Record<string, Point[]> = JSON.parse(text)

  draw(x0, x1, data["re"], data["im"], data["psi_sq"])
})

command.stderr.on("data", (text) => console.log(text))

await command.spawn()
```

Rysunek 4.12. Kod z pliku free-particle.ts

4.6 Wdrożenie i dystrybucja aplikacji

Aplikacja została stworzona przy użyciu frameworka Tauri, który umożliwia kompilację do formatów .msi i .exe na system Windows, .deb i .AppImage na system GNU/Linux. Możliwa jest też kompilacja na system macOS na architekturę procesora x86 i ARM, jednak nie sprawdzałem tych opcji, ponieważ nie mam dostępu do takich urządzeń.

Udostępnione przeze mnie zostały formaty .msi, który w przeciwieństwie do .exe jest dedykowany dla instalatorów programów na system Windows oraz .deb, ponieważ jest to format wspierany przez największe dystrybucje systemu GNU/Linux, czyli Debian i Ubuntu. Dodatkowo format .AppImage powodował problem szerzej opisany w kolejnym rozdziale.

Framework Tauri sam kopiuje do wersji skompilowanej niezbędne pliki .html, .css .ts oraz wszelkie obrazy i fonty użyte w plikach .css, jednak użyte skrypty .py muszą być specjalnie oznaczone jako zasób aplikacji w pliku konfiguracyjnym tauri.conf.json. Ponieważ twórcy frameworka Tauri stworzyli go z myślą o bezpieczeństwie [17], w pliku konfiguracyjnym zadeklarować, żeby aplikacja miała możliwość wykonywania poleceń systemowych, do uruchomienia interpretera Pythona oraz możliwość odczytu plików, do użycia plików ze skryptami Pythona.

Ponieważ obliczenia matematyczne wykonywane są przy użyciu języka Python, niezbędny do działania aplikacji jest zainstalowany interpreter Pythona 3.11 lub nowszy z zainstalowanym modułem NumPy. Zdecydowałem się nie dołączać interpretera do instalatora mojej aplikacji, ponieważ znacząco zwiększyłoby to rozmiar finalnego pliku.

4.7 Problemy, ograniczenia i możliwości rozwoju

Jedno z ważniejszych ograniczenie mojej aplikacji leży w interfejsie pomiędzy językiem TypeScript i Pythonem. O ile wykonanie samych obliczeń fizycznych w Pythonie liczyć można w milisekundach, to proces zserializowania danych, czyli zamiany ich na tekst, przekazania ich do odczytu i deserializacji, czyli zamiany tekstu na obiekt, w języku TypeScript wydłuża ten proces do kilkuset milisekund, zwykle co najmniej 200. Jest to czas niezauważalny, jeżeli potrzebne jest wykonanie obliczeń raz lub stosunkowo rzadko, co ma miejsce w aplikacji, jednak częste odświeżanie wyświetlanych danych byłoby po prostu niemożliwe do zrealizowania przy zastosowanym rozwiązaniu. Należy jednocześnie zwrócić uwagę, że przy bardziej rozbudowanych scenariuszach fizycznych czas przekazywania danych może okazać się nieznaczący, jeśli samo wykonywanie obliczeń fizycznych trwać znacznie dłużej. Dokładnie takie przypadki miałem na względzie, decydując o zastosowaniu Pythona w projekcie i jest to kierunek, w którym aplikacja mogłaby zostać w przyszłości rozwinięta.

Kolejną rzeczą, która ogranicza aplikację jest forma dystrybucji. Na przestrzeni lat aplikacje desktopowe ustępowały popularnością aplikacjom webowym. W wyniku takich zmian użytkownicy przyzwyczajeni do korzystania z przeglądarki internetowej mogą nie chcieć poświęcić czasu na pobranie i zainstalowanie mojej aplikacji. Niestety jednak, pomimo że jest to już możliwe, wykonywanie skryptów Pythona w przeglądarce jest dalej w fazie rozwoju [29], co wymusza taką formę aplikacji.

Z pozoru największym problemem, jaki napotkałem, był fakt, że aplikacja po skompilowaniu do formatu .AppImage nie była w stanie poprawnie wywoływać skryptów Pythona. Jak jednak powiedziałem, problem tylko z pozoru wydał się krytyczny, a to przez fakt, że formaty .deb i .msi nie sprawiały żadnych problemów. Wciąż zaintrygowany tym nieoczekiwanym zachowaniem, zgłosiłem się przez komunikator Discord o pomoc do zespołu wsparcia frameworka Tauri. Okazało się, że nie jest to spodziewane zachowanie i najprawdopodobniej napotkałem błąd w samym frameworku. Na prośbę zespołu wsparcia zamieściłem repozytorium z minimalnym kodem niezbędnym do odtworzenia tego błędu [30].

Ważnym aspektem, który niekiedy spowalniał tworzenie pracy, jest fakt że framework Tauri, wydany w 2022 roku [31], jest stosunkowo nową technologią. Pomimo że dostępną jest dokumentacja, to potrafi ona zawierać braki lub niedopowiedzenia, a szukanie szczegółowych odpowiedzi może stanowić wyzwanie, ponieważ nie istnieje na ten temat znacząca ilość literatury.

Rzeczą którą niewątpliwie można poprawić, jednak nie jest ona moim zdaniem kluczowa,

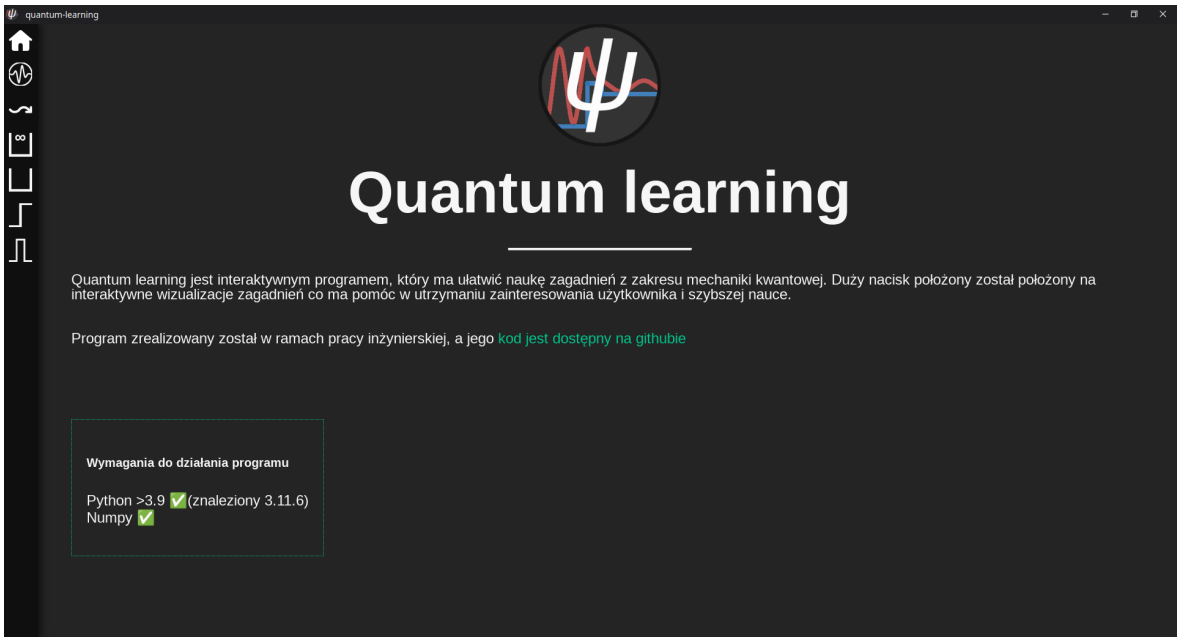
jest weryfikacja instalacji interpretera języka Python. Aktualnie aplikacja nie weryfikuje czy interpreter ten jest rzeczywiście zainstalowany oraz czy zainstalowany jest moduł NumPy. Dodanie takiej funkcjonalności w przyszłości niewątpliwie pomogłoby, w szukaniu problemów w razie nieprawidłowego działania aplikacji.

5 Interfejs aplikacji

Przy projektowaniu interfejsu najbardziej zależało mi na przejrzystości i intuicyjności. Były to wskaźniki dla mnie najważniejsze, ponieważ chciałem, aby użytkownik mógł szybko przystąpić do korzystania z aplikacji i nauki trudnej dziedziny, jaką jest mechanika kwantowa. Dzięki prostemu interfejsowi w aplikacji nie musiałem udzielać instrukcji, jak ją nawigować i z niej korzystać, co mogłoby odstraszyć potencjalnego użytkownika.

5.1 Ekran główny i nawigacja

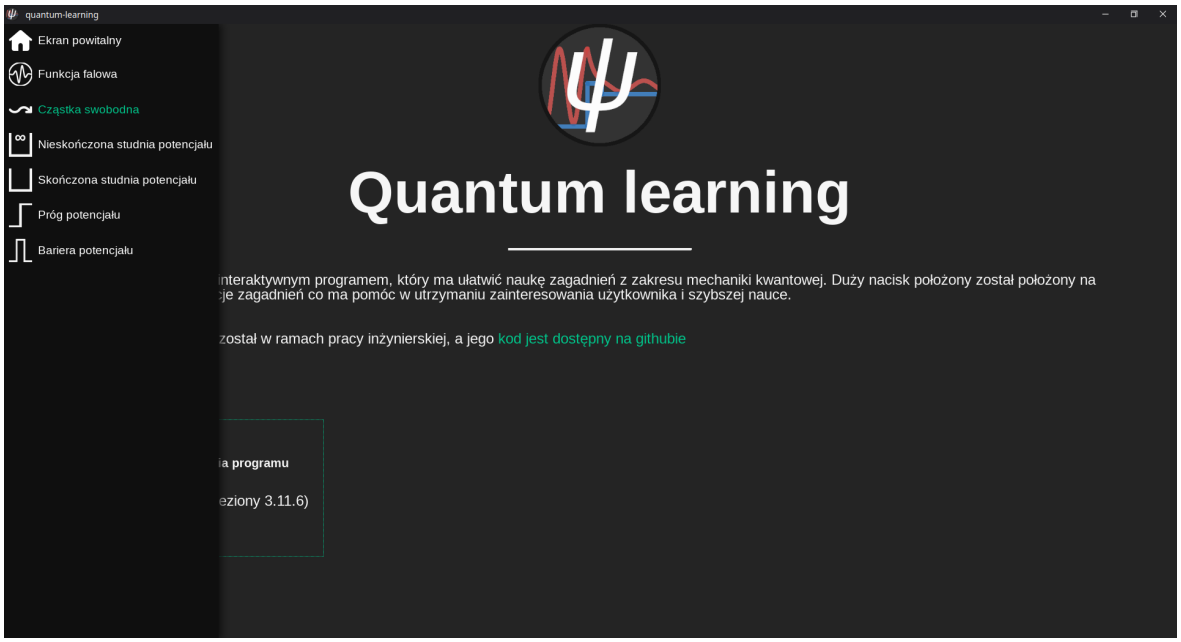
Po włączeniu aplikacji jako pierwszy zobaczymy ekran główny widoczny na rysunku 5.1. Umieściłem na nim krótką informację o celu stworzenia aplikacji i jej zawartości, a także zamieściłem link do kodu aplikacji. Poniżej tych informacji zamieściłem również element pokazujący, czy użytkownik ma zainstalowany interpreter Pythona w odpowiedniej wersji oraz bibliotekę NumPy.



Rysunek 5.1. Ekran główny

Do nawigacji po rozdziałach w aplikacji użytkownik może użyć menu po lewej stronie. Nieaktywne menu składa się tylko z prostych ikon, jednak po najechaniu kursorem pokazują się pełne nazwy rozdziałów co widać na rysunku 5.2. Nazwa rozdziału po najechaniu

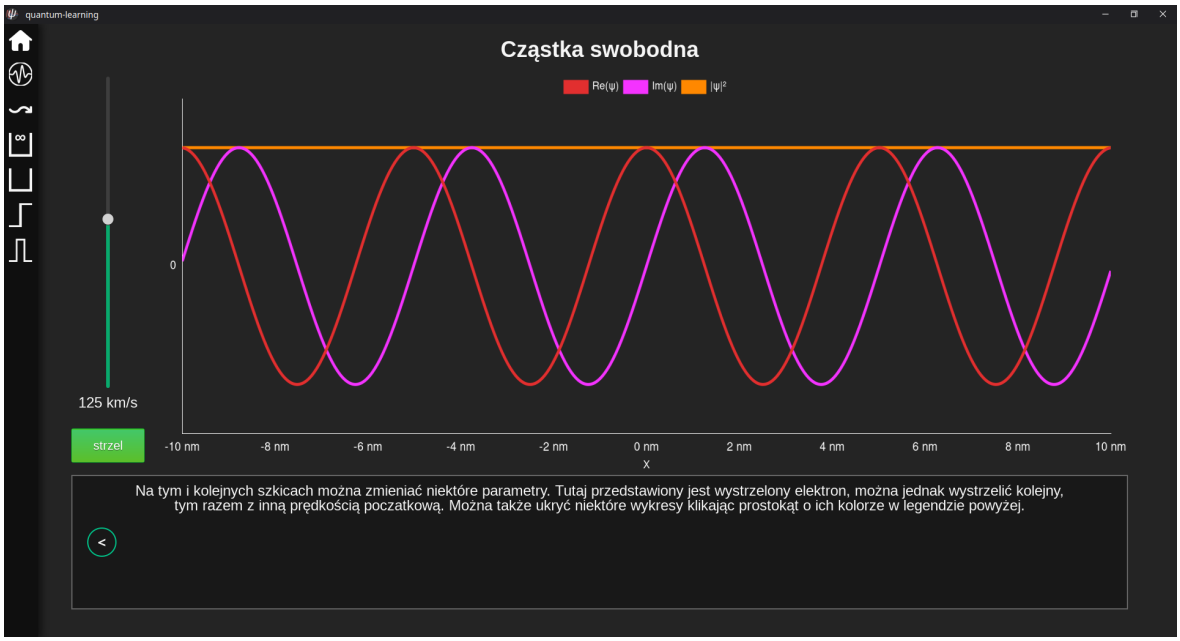
kursorem podświetla się innym kolorem, a jej kliknięcie przenosi użytkownika do danego tematu.



Rysunek 5.2. Rozwinięte menu do nawigacji

5.2 Interaktywna wizualizacja

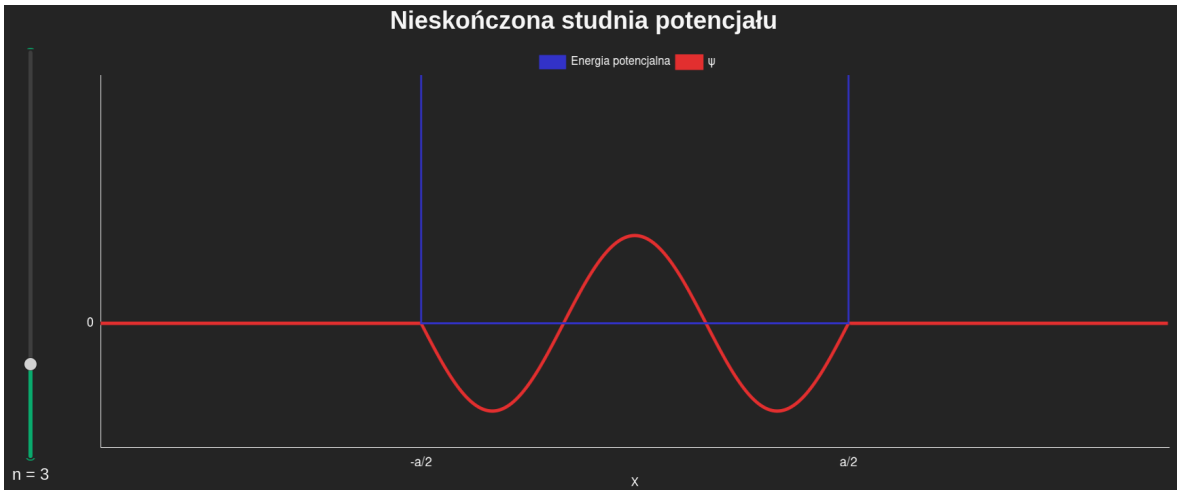
Najwięcej uwagi w aplikacji przykuwają interaktywne wizualizacje omawianych zagadnień, co można zobaczyć na rysunku 5.3. Do każdego z rozwiązań równania Schrodingera przytoczonego w rozdziale 1.2 przygotowana została wizualizacja, przedstawiająca zachowanie funkcji falowej. W każdej z tych wizualizacji użytkownik ma możliwość wprowadzenia zmian niektórych parametrów oraz podglądu, jak zmieni się funkcja falowa w reakcji na te zmiany.



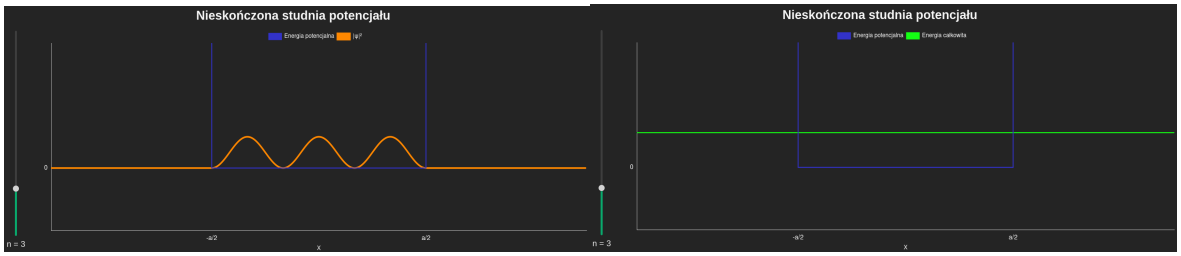
Rysunek 5.3. Widok rozdziału cząstka swobodna

Zaczynając od przypadku cząstki swobodnej, widocznego na rysunku 5.3, naszkicowana została funkcja falowa wraz z kwadratem jej modułu, czyli gęstością prawdopodobieństwa znalezienia cząstki w danym obszarze. Ponieważ funkcja falowa przyjmuje wartości zespolone, oddzielnie zostały naszkicowane część rzeczywista i urojona. Po lewej stronie znajduję się suwak, za pomocą którego użytkownik może wystrzelić elektron o prędkości z przedziału 20 km/s - 250 km/s i zobaczyć, jak zmieni się kształt funkcji falowej. Amplituda funkcji falowej została ustalona jako 1, co dla opisanej na osi x skali oznacza, że gęstość prawdopodobieństwa $|\psi|^2$ jest nieprawidłowo unormowana. Zdecydowałem się na taki zabieg, aby uniknąć znacznych rozbieżności między wartościami ψ oraz $|\psi|^2$ co utrudniłoby przedstawienie ich na tym samym wykresie. Dodatkowo poprawna normalizacja nie była konieczna do opisu tego przypadku.

Kolejna wizualizacja poświęcona jest cząstce w nieskończonej studni potencjału. Zgodnie ze wzorem 3.14 wiadomo, że cząstka przyjmuje dyskretne wartości energii, więc dołączony został suwak do zmiany numeru stanu energetycznego n. W przypadku studni potencjału funkcja falowa przyjmuje wartości czysto rzeczywiste lub czysto urojone nie było więc sensu rozdzielać tych wartości w legendzie i przedstawione one są zbiorczo jako ψ, co widać na rysunku 5.4. Dodatkowo ukazana została wartość energii każdego stanu.



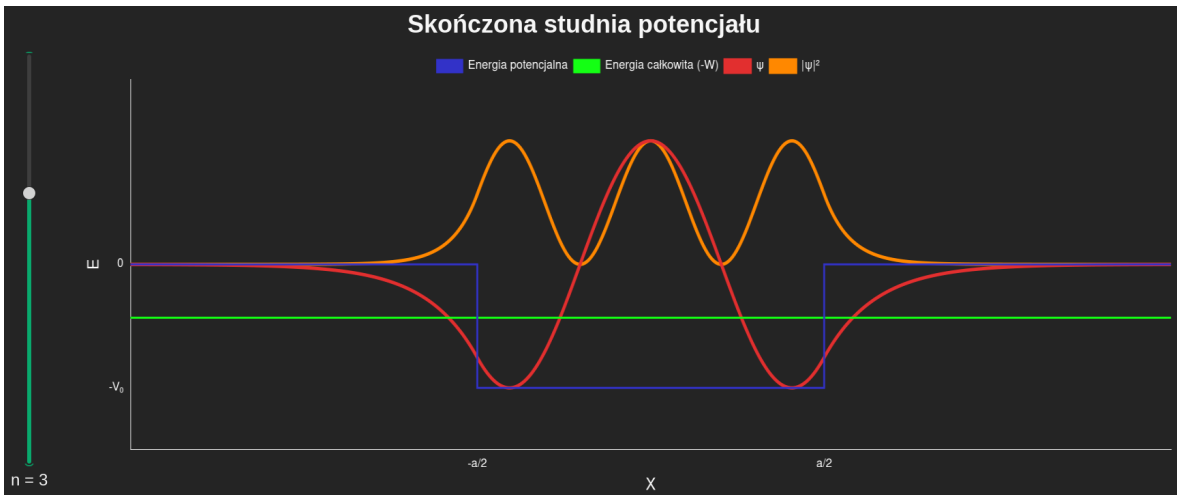
(a) ψ



(b) $|\psi|^2$ **(c)** Energia

Rysunek 5.4. Wizualizacja z rozdziału nieskończona studnia potencjału

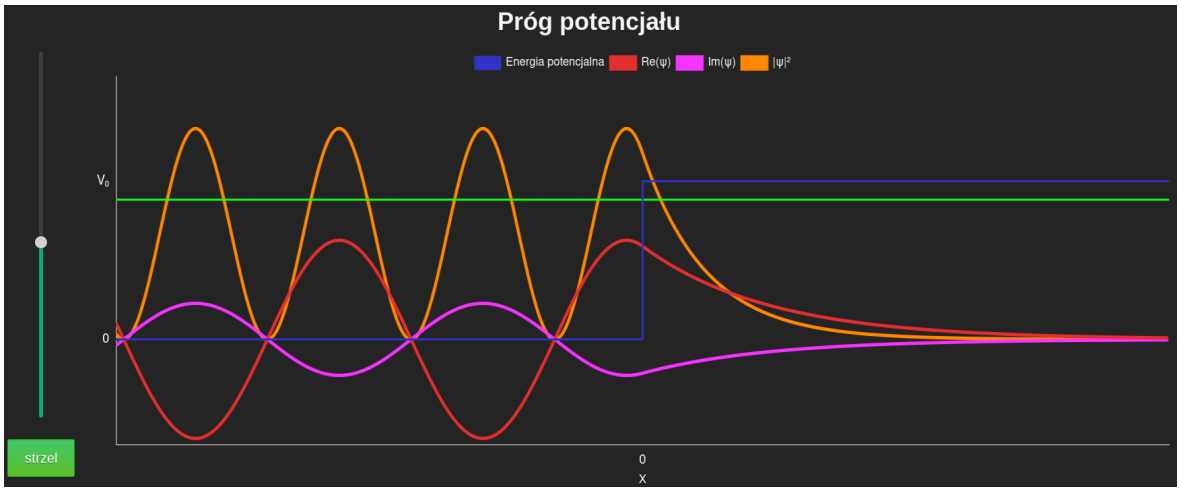
Bardzo podobnie prezentuje się wizualizacja skończonej studni potencjału przedstawiony na rysunku 5.5, jednak w tym przypadku zdecydowałem się nanieść wszystkie wartości na jeden wykres. Przez ograniczoną głębokość studni wartości energii nie wykraczają ponad wykres, w przeciwieństwie do większych wartości n w studni nieskończonej.



Rysunek 5.5. Wizualizacja z rodziału skończona studnia potencjału

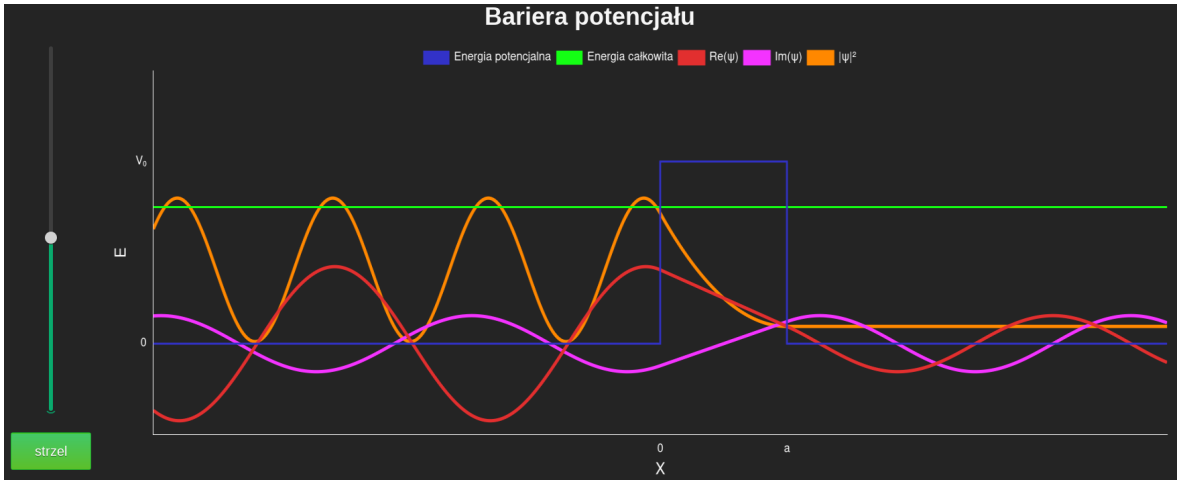
Dla progu potencjału, tak jak w przypadku cząstki swobodnej, oddzielnie naszkicowane zostały części rzeczywista i urojona funkcji falowej. Do dyspozycji użytkownika jest suwak

widoczny na rysunku 5.6, służący do zmiany energii wystrzelonej cząstki. Szkic wartości tej energii oznaczony został kolorem zielonym i reaguje swoją pozycją na zmiany wprowadzone suwakiem. Suwak uniemożliwia ustawienie przypadku, gdy $E = V_0$, ponieważ wymagałby on dodatkowej implementacji, a i tak jest on najczęściej pomijany w podręcznikach [2] [3] [4].



Rysunek 5.6. Wizualizacja z rozdziału próg potencjału

Dosyć zbliżona do wizualizacji progu potencjału jest zasada działania wizualizacji bariery potencjału, widoczna na rysunku 5.7. Tak samo dostępny jest suwak do zmiany wartości wystrzelonej cząstki, uniemożliwiający ustawienie $E = V_0$.

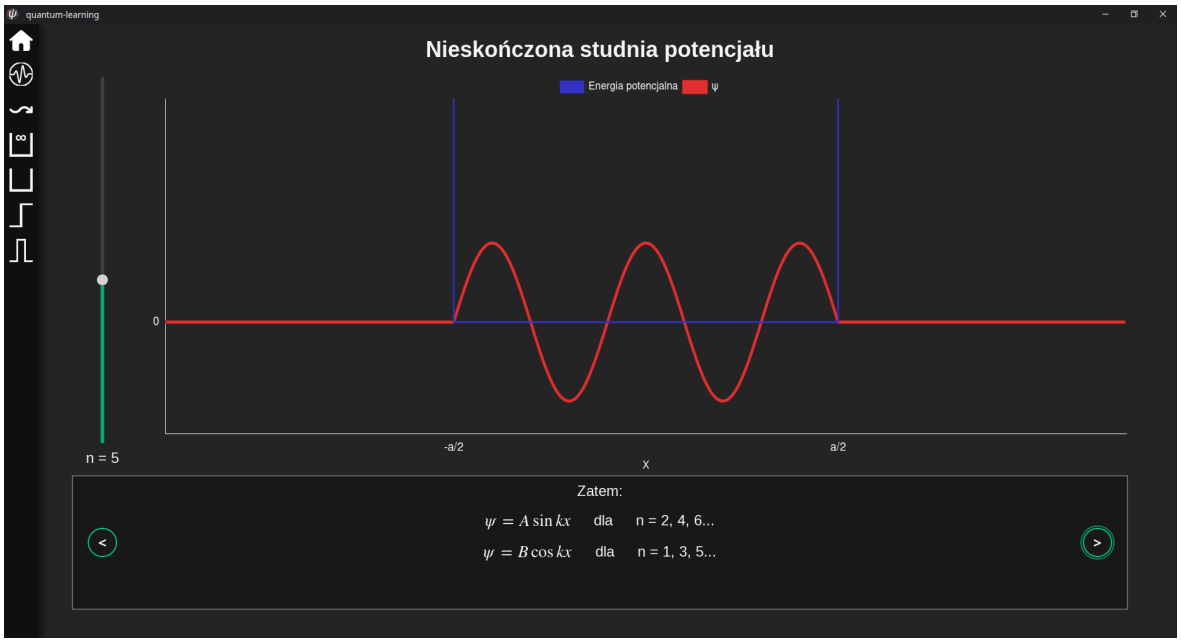


Rysunek 5.7. Wizualizacja z rozdziału bariera potencjału

5.3 Narrator

Zastosowane wizualizacje, pomimo że interesujące, wymagają opisu do wyjaśnienia i pełnego ich zrozumienia. Nie chciałem użytkownika przytłoczyć duża ilością tekstu oraz chcia-

łem, aby wizualizacje stanowiły centralny punkt aplikacji. Jako rozwiązanie tych problemów, pod każdym szkicem zamieściłem okno, w którym tłumaczone są kolejne kroki, do znalezienia równania funkcji falowej, dla danego scenariusza. Okno to, widoczne na rysunku 5.8, składa się również z przycisków do nawigacji po kolejnych krokach. Dodatkowo wizualizacja nie jest niezależna od takiego narratora, a zamiast tego pokazywane są użytkownikowi elementy, które są akurat tłumaczone. Widać to zachowanie na rysunku 5.8, gdzie przy tłumaczeniu wykresu funkcji falowej, tylko ta funkcja zostaje akurat pokazana.



Rysunek 5.8. Funkcja falowa w nieskończonej studni potencjału

6 Podsumowanie i wnioski

Patrząc na stworzoną aplikację, uważam że początkowy cel został zrealizowany. Aplikacja zawiera typowe rozwiązania równania Schrodingera, a przy tym przedstawia te rozwiązania w ciekawy i przykuwający uwagę sposób. Można jej użyć do łatwiejszej nauki trudnych problemów mechaniki kwantowej, jednocześnie przedstawiając je w interesujący sposób. Należy oczywiście pamiętać, że aplikacja nie przedstawia każdego szczegółu zagadnień, a jedynie zbiera najważniejsze informacje na dane tematy.

Dzięki zastosowanym technologiom możliwa jest rozbudowa aplikacji o kolejne przykłady. HTML, CSS, TypeScript i Python to bardzo popularne języki, dzięki czemu wiele osób poza mną byłoby w stanie dodać inne scenariusze do rozwiązania równania Schrodingera. Python daje też dużo elastyczności do wykonywania obliczeń fizycznych i symbolicznych, więc można niewątpliwe wyjść poza tematy opracowane przeze mnie.

Najbardziej ograniczającą cechą aplikacji, o której nie można zapomnieć przy planowaniu jej rozwoju, jest stosunkowo długi czas przekazywania obliczonych danych pomiędzy językami TypeScript i Python. Mając to na uwadze, tworzone wizualizacje nie mogą być zbyt dynamiczne, albo odbywać się w czasie rzeczywistym. Bardzo dobrze jednak sprawdzą się wizualizacje mniej zmienne, ale wymagające więcej czasu na obliczenie.

7 Bibliografia

[1] Statystyki eurostat na temat energetyki jądrowej, *https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Nuclear_energy_statistics* (ostatni dostęp 28.04.2024)

[2] M.R. Wehre, H.A. Enge, J.A. Richards, *Wstęp do fizyki atomowej*, Państwowe Wydawnictwo Naukowe, Warszawa 1983. ISBN 83-01-02200-0

[3] L.D. Landau, E.M. Lifszyc, *Mechanika kwantowa Teoria nierelatywistyczna* Państwowe Wydawnictwo Naukowe, Warszawa 1979. ISBN 83-01-00501-7

[4] R. Eisberg, R. Resnick, *Fizyka kwantowa atomów, cząsteczek, ciał stałych, jąder i cząstek elementarnych* Państwowe Wydawnictwo Naukowe, Warszawa 1983. ISBN 83-01-04350-1

[5] David Flanagan, *JavaScript: The Definitive Guide. Master the World's Most-Used Programming Language. 7th Edition*, O'Reilly Media, 2020. ISBN 978-14-919-5198-9

[6] Strona biblioteki React, *https://react.dev/* (ostatni dostęp 9.05.2024)

[7] Strona frameworka Angular, *https://angular.io/* (ostatni dostęp 9.05.2024)

[8] Strona frameworka Vue.js, *https://vuejs.org/* (ostatni dostęp 9.05.2024)

[9] Strona frameworka Svelte, *https://svelte.dev/* (ostatni dostęp 9.05.2024)

[10] Dokumentacja języka HTML, *https://developer.mozilla.org/en-US/docs/Web/HTML* (ostatni dostęp 9.05.2024)

[11] Dokumentacja języka CSS, *https://developer.mozilla.org/en-US/docs/Web/CSS* (ostatni dostęp 9.05.2024)

[12] Typy w języku TypeScript, *https://www.typescriptlang.org/docs/handbook/2/basic-types.html* (ostatni dostęp 9.05.2024)

[13] Wes McKinney, *Python for Data Analysis* O'Reilly Media, 2012. ISBN 078-1-449-31979-3

[14] Dokumentacja Chart.js, *https://www.chartjs.org/docs/latest/* (ostatni dostep 30.04.2024)

[15] Dokumentacja MathJax, *https://docs.mathjax.org/en/latest/* (ostatni dostep 30.04.2024)

[16] Strona główna WebAssembly, *https://webassembly.org/* (ostatni dostęp 9.05.2024)

[17] Prezentacja Tauri, *https://tauri.app/about/intro* (ostatni dostęp 28.04.2024)

[18] Dokumentacja SymPy, *https://docs.sympy.org/latest/index.html* (ostatni dostęp 13.01.2024)

[19] Liczby zespolone, *https://en.wikipedia.org/wiki/Complex_number* (ostatni dostęp 09.05.2024)

[20] Gilbert Lewis, *https://pl.wikipedia.org/wiki/Gilbert_Lewis* (ostatni dostęp 15.01.2024)

[21] Fale materii, *https://pl.wikipedia.org/wiki/Fale_materii* (ostatni dostęp 15.01.2024)

[22] Równanie Schrödingera, *https://pl.wikipedia.org/wiki/Równanie_Schrödingera* (ostatni dostęp 15.01.2024)

[23] Metodologia BEM, *https://getbem.com/naming/* (ostatni dostęp 11.05.2024)

[24] Studnia kwantowa, *https://en.wikipedia.org/wiki/Finite_potential_well* (ostatni dostęp 15.01.2024)

[25] Metoda Newtona, *https://pl.wikipedia.org/wiki/Metoda_Newtona* (ostatni dostęp 15.01.2024)

[26] Metoda równego podziału, *https://pl.wikipedia.org/wiki/Metoda_równego_podziału* (ostatni dostep 15.01.2024)

[27] Architektura Tauri, *https://tauri.app/v1/references/architecture/* (ostatni dostęp 28.04.2024)

[28] Dokumentacja API shell w Tauri, *https://tauri.app/v1/api/js/shell/* (ostatni dostęp 28.04.2024)

[29] Repozytorium Pyodide, *https://github.com/pyodide/pyodide* (ostatni dostęp 28.04.2024)

[30] Repozytorium do odtworzenia błędu w kompilacji aplikacji Tauri do AppImage, *https://github.com/xAdiro/tauri-appimage-python-error* (ostatni dostęp 28.04.2024)

[31] Plan działania Tauri 2.0, *https://beta.tauri.app/blog/roadmap-to-tauri-2-0/* (ostatni dostęp 28.04.2024)

Wyrażam zgodę na udostępnienie mojej pracy w czytelniach Biblioteki SGGW w tym w Archiwum Prac Dyplomowych SGGW.

. . . . . .
(czytelny podpis autora pracy)