МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики, информатики и механики
Кафедра вычислительной математики и прикладных информационных технологий

ЛАБОРАТОРНАЯ РАБОТА №2
ЧИСЛЕННОЕ РЕШЕНИЕ СТАЦИОНАРНОГО УРАВНЕНИЯ ШРЁДИНГЕРА: ТЕОРИЯ ВОЗМУЩЕНИЙ

Направление: 01.04.02 – Прикладная математика и информатика
Выполнил: студент 11 группы 2 курса магистратуры
Крутько А.С.
Преподаватель: доктор физ.-мат. наук, профессор Тимошенко Ю.К.

Воронеж 2024

Содержание

1 Цели и задачи работы

1.1 Цель работы.

Целями лабораторной работы являются практическое освоение информации, полученной при изучении курса «Компьютерное моделирование в математической физике» по теме «Численное решение стационарного уравнения Шрёдингера», а также развитие алгоритмического мышления и приобретение опыта использования знаний и навыков по математике, численным методам и программированию для решения прикладных задач физико-технического характера.

1.2 Задачи работы:

Проблема: электрон находится в одномерной потенциальной яме с бесконечными стенками $U(x)$:

$$v(x) = \begin{cases} J_2(x), & x \in (-L, L), \\ \infty, & x \notin (-L, L), \end{cases}$$

Где $U(x) = v(x) * V_0$, $V_0 = 25$ эВ, $L = 3$ Å, $J_n(x)$ – функция Бесселя, $n = 2$.

1. Используя метод возмущений, найти энергию, нормированную волновую функцию для основного и 2-го возбужденного состояний и плотности вероятности. Энергию вычислять с учетом поправок до второго порядка включительно, а волновую функцию с учетом поправок первого порядка. Возмущенную систему смоделировать, создав в потенциальной функции $U(x)$ пик произвольной формы. Привести как числовые значения энергий, так и построить графики волновых функций и плотностей вероятности.

2. Вычислить для этих состояний квантовомеханические средние $\langle p(x) \rangle$ и $\langle p(x^2) \rangle$.

3. Сравнить результаты с данными, полученными методом пристрелки.

2 Одномерное стационарное уравнение Шрёдингера. Математический формализм. Общие свойства решений

Одномерное стационарное уравнение Шрёдингера [1]:

$$\hat{H}\psi(x) = E\psi(x), \tag{1}$$

где $\hat{H}$ – оператор Гамильтона, E – собственные значения энергии, $\psi(x)$ – волновая функция.

С математической точки зрения оно представляет собой задачу определения собственных значений E и собственных функций ψ оператора Гамильтона $\hat{H}$. Для частицы с массой m, находящейся в потенциальном поле $U(x)$, оператор Гамильтона имеет вид

$$\hat{H} = \hat{T} + U(x), \tag{2}$$

где оператор кинетической энергии

$$\hat{T} = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2}, \tag{3}$$

а $\hbar$ — постоянная Планка. Собственное значение оператора Гамильтона имеет смысл энергии соответствующей изолированной квантовой системы. Собственные функции называются волновыми функциями. Волновая функция однозначна и непрерывна во всём пространстве. Непрерывность волновой функции и её первой производной сохраняется и при обращении $U(x)$ в ∞ некоторой области пространства. В такую область частица вообще не может проникнуть, то есть в этой области, а также на её границе $\psi(x) = 0$.

Оценим нижнюю границу энергетического спектра. Пусть минимальное значение потенциальной функции равно $U_{\min}$. Очевидно, что $\langle T\rangle \geq 0$ и $\langle U\rangle \geq U_{\min}$. Потому из уравнения (1) следует, что:

$$E = \langle H\rangle = \int_{-\infty}^{+\infty} \psi^{\star}(x)\hat{H}\psi(x)\,dx = \langle T\rangle + \langle U\rangle > U_{\min}. \tag{4}$$

то есть, энергии всех состояний $> U_{min}$.

Особый практический интерес представляет случай, когда

$$\lim_{x\to\infty} U(x) = 0. \tag{5}$$

Потенциал такого типа называется также потенциальной ямой. Для данной $U(x)$ свойства решений уравнения Шрёдингера зависят от знака собственного значения E. Если $E < 0$. Частица с отрицательной энергией совершает финитное движение. Оператор Гамильтона имеет дискретный спектр, то есть собственные значения и соответствующие собственные функции можно снабдить номерами. При $E < 0$ уравнение (1) приобретает вид[1]:

$$\hat{H}\psi_k(x) = E_k\psi_k(x). \tag{6}$$

Квантовое состояние, обладающее наименьшей энергией, называется основным. Остальные состояния называют возбужденными состояниями. В силу линейности стационарного уравнения Шрёдингера, волновые функции математически определены с точностью до постоянного множителя. Однако, из физических соображений, волновые функции должны быть нормированы следующим образом:

$$\int_{-\infty}^{+\infty} |\psi_k(x)|^2, dx = 1. \tag{7}$$

В дальнейшем будет рассматриваться только дискретный спектр. При этом необходимо пользоваться **осцилляционной теоремой**.

Осцилляционная теорема. Упорядочим собственные значения оператора Гамильтона в порядке возрастания, нумеруя энергию основного состояния индексом "0": E_0, E_1, E_2 ..., E_k,.... Тогда волновая функция $\psi_k(x)$ будет иметь k узлов (то есть, пересечений с осью абсцисс). Исключения: области, в которых потенциальная функция бесконечна.

3 Теория возмущений. Алгоритм

К числу приближенных методов вычисления собственных значений и собственных функций оператора Гамильтона относится метод стационарных возмущений Релея-Шрёдингера[3], который мы далее будем просто называть «методом» или «теорией возмущений».

В рамках этого теоретического подхода предполагается, что оператор Гамильтона, чьи собственные значения и собственные функции требуется определить, может быть представлен в виде:

$$\hat{H} = \hat{H}^0 + \hat{V} \tag{8}$$

где $\hat{H}^0$ – гамильтониан идеализированной задачи, решение которой можно найти либо аналитически, либо относительно простым численным путем; – называется оператором возмущения или просто возмущением.

Оператором возмущения может быть либо часть гамильтониана, которая не учитывалась в идеализированной задаче, либо потенциальная энергия, связанная с наличием внешнего воздействия.

Идеализированную систему, которую описывает гамильтониан $\hat{H}^0$, называют «невозмущенной» системой, а систему с гамильтонианом $\hat{H}$ – «возмущенной» системой. В рамках теории возмущений удаётся получить формулы, определяющие энергии и волновые функции стационарных состояний через известные значения энергий $E_n^{(0)}$ и волновых функций $\Psi_n(0)$ невозмущенной системы.

Стационарные уравнения Шрёдингера для невозмущенной ив возмущенной систем [3] имеют вид:

$$\hat{H}^{(0)}\Psi_n^{(0)}(x) = E_n^{(0)}\Psi_n^{(0)}; \tag{9}$$

$$\hat{H}\Psi_n(x) = E_n\Psi_n(x). \tag{10}$$

В теории возмущений решения уравнения (10) ищутся в виде рядов:

$$\begin{aligned} E_n &= E_n^{(0)} + E_n^{(1)} + E_n^{(2)} + \cdots = \sum_{k=0}^{\infty} E_n^{(k)}, \\ \Psi_n(x) &= \Psi_n^{(0)}(x) + \Psi_n^{(1)}(x) + \Psi_n^{(2)}(x) + \cdots = \sum_{k=0}^{\infty} \Psi_n^{(k)}(x), \end{aligned} \tag{11}$$

где E_n^k, $\Psi_n^{(k)}$ – величины k-го порядка малости по возмущению $\hat{V}$, называемые k-ми поправками или поправками k-го порядка. Первые слагаемые рядов (11) определяются следующими формулами:

$$\begin{aligned} E_n^{(1)} &= V_{nn}, \\ E_n^{(2)} &= \sum_m{}' \frac{|V_{mn}|^2}{E_n^{(0)} - E_m^{(0)})}, \\ \Psi_n^{(0)}(x) &= \sum_m{}' \frac{V_{mn}}{E_n^{(0)} - E_m^{(0)}} \Psi_m^0(x), \end{aligned} \tag{12}$$

где

$$V_{mn} \equiv \langle m|V|n\rangle = \int_\Omega \Psi_m^{(0)\star}(x)\hat{V}\Psi_n^{(0)}(x)dx. \tag{13}$$

Штрих над знаком суммы означает пропуск слагаемого с $m = n$: $\sum'_m \equiv \sum_{m\neq n}$. Очевидно, что ряды в (11) сходятся, если выполняется равенство

$$|V_{mn}| \ll |E_n^{(0)} - E_m^{(0)}|. \tag{14}$$

Во многих случаях для решения задачи достаточно ограничиться вычислением энергии с учетом поправок до второго порядка включительно волновой функции с учетом поправок первого порядка[3]:

$$\begin{aligned} E_n &= E_n^{(0)} + V_{nn} + {\sum_m}' \frac{|V_{mn}|^2}{E_n^{(0)} - E_m^{(0)}}, \\ \Psi_n(x) &= \Psi_n^{(0)}(x) + {\sum_m}' \frac{V_{mn}}{E_n^{(0)} - E_m^{(0)}} \Psi_m^{(0)}(x). \end{aligned} \tag{15}$$

При вычислении второй поправки к энергии и первой поправки к волновой функции основного состояния возмущенной системы бесконечные суммы в (11) заменяются конечными. Для оценки корректного численного значения надо выполнить несколько расчетов соответствующей суммы для монотонно возрастающих величин. Если увеличение верхнего предела суммы, начиная с некоторого значения, не приводит к заметным изменениям суммы, то задача оценки значения верхнего значения суммы решена.

4 Программная реализация алгоритма

В Приложение представлена программа на языке Python 3.12[2], реализованная в среде разработки PyCharm Community Edition 2024.3.1, численного решения одномерного стационарного уравнения Шрёдингера для электрона в одномерной потенциальной яме. Программа реализует алгоритм теории возмущений, позволяющий находить собственные значения и соответствующие им волновые функции. Потенциальная функция (невозмущенная система) и параметры для нее соответствуют постановке задачи из первой главы. Энергия и длина ямы были переведены в атомные единицы Хартри (строки **X**–**X**).

В строках **X**–**X** реализован алгоритм пристрелки, который подробно разобран в лабораторной работе №1, в данной программе этот алгоритм используется для реализации невозмущенной системы и вычисления её решения (собственные значения и собственные функции оператора Гамильтона). Собственные значения и собственные функции невозмущенной системы будут использоваться для вычисления решения возмущенной системы. Путем компьютерного моделирования был вычислен верхний предел сумм (11). П программе за верхний предел сумм отвечает `k_max` в строке **X**, он равен количеству вычисленных энергий невозмущенной системы, для длинного варианта задачи достаточно было 14.

В строках **X**–**X** реализована потенциальная функция возмущенной системы, для этого был создан пик, который больше $\max(U(x))$ на отрезке $[2.5; 3.0]$

В строках **X**–**X** реализован оператор возмущения.

В строках **X**–**X** и **X**–**X** реализованы функции возвращающие энергии и волновые функции невозмущенной системы.

В строках **X**–**X** и **X**–**X** реализованы функции которые вычисляют матричный элемент оператора возмущения по невозмущенным системам (12).

В строках **X**–**X** реализована функция вычисляющая поправку второго порядка (11).

В строках **X**–**X** и **X**–**X** реализованы функции вычисляющие поправку первого порядка для волновой функции возмущенной системы.

В строках **X**–**X** реализована функция вычисляющая первое приближение волновой функции (15)

В строках **X**–**X** реализована функция с одним параметром `root` определяющий номер состояния возмущенной системы для которого требуется вычислить энергию и волновую функцию. В функции вычисляется энергия с учетом поправок до второго порядка включительно, и волновая функция с учетом поправок первого порядка. Также в функции реализован вывод графиков и запись данных в файл.

В строках **X**–**X** вызывается функция`result` для основного и второго возбужденного состояний возмущенной системы.

В строках **X**–**X** выводятся графики невозмущенной и возмущенной систем.

5 Результаты численных экспериментов

Ниже продемонстрированы результаты работы программного кода написанного на Python.

5.1 Иллюстрация работы программы

Потенциал из постановки задачи представлен на Рис. **??**

5.2 Значения искомых параметров

Ниже результаты численных экспериментов, полученных в результате работы программы выведены в таблицу:

Квантовомеханические средние $\langle p(x) \rangle$ и $\langle p(x^2) \rangle$ для основного, первого и второго возбужденного состояний:

Состояние	**Энергия, а.е.**	$\langle p(x) \rangle$	$\langle p(x^2) \rangle$
Основное	0.026451	$0.000000e+00$	$3.072237e-01$
1-е возбужденное	0.498856	$0.000000e+00$	$1.006510e+00$
2-е возбужденное	0.828367	$0.000000e+00$	$2.410640e+00$

6 Заключение

Таким образом, было получено численное решение для задачи о частице в одномерной квантовой яме с бесконечными стенками при помощи метода пристрелки. Были получены значения энергий и волновые функции основного и второго возбужденного состояний. Полученные волновые функции соответствуют осцилляционной теореме. Кроме того, для каждого состояния были вычисленные квантовомеханические средние $\langle p(x) \rangle, \langle p(x^2) \rangle$.

Приложение

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import eval_laguerre

def draw_potential_graph():
    n = 500
    c_energy = 27.212
    c_length = 0.5292
    v0 = 25.0 / c_energy
    l = 3.0 / c_length
    a, b = -l, l
    x = np.linspace(a - 0.01, b + 0.01, n)

    def u_func():
        u_val = np.zeros(n)
        for i in range(n):
            if np.abs(x[i]) <= l:
                u_val[i] = v0 * eval_laguerre(5, np.abs(x[i]))
            else:
                u_val[i] = l

        return u_val

    y = u_func()

    plt.plot(x, y, 'g-', linewidth=6.0, label="U(x)")
    plt.title(f"Potential function graph")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.grid(True)
    plt.legend()

    plt.savefig('Potential_func_graph.jpg')
    plt.show()


class Solver:
    # Params
    def __init__(self):
        self.U_min = -0.149124
        self.c_energy = 27.212
        self.c_length = 0.5292
        self.V0 = 25.0 / self.c_energy
        self.L = 3.0 / self.c_length
        self.A, self.B = -self.L, self.L
        self.n = 650
        self.h = (self.B - self.A) / (self.n - 1)
        self.c, self.W = self.h ** 2 / 12.0, 3.0
        self.Psi, self.Fi, self.X = np.zeros(self.n), np.zeros(self.n), np.
    linspace(self.A, self.B, self.n)
        self.r = (self.n - 1) // 2 - 80
        self.limit_value = 4.0

        self.d1, self.d2 = 1.e-09, 1.e-09
        self.tol = 1e-6

        self.E_min, self.E_max, self.step = self.U_min + 0.01, 2.0, 0.01

```

```
58
59      def u_func(self, x):
60          # Check if x is a scalar
61          if np.isscalar(x):
62              # x - scalar
63              return self.V0 * eval_laguerre(5, abs(x)) if abs(x) <= self.L
    else self.W
64          u_val = np.zeros(self.n)
65          for i in range(self.n):
66              if np.abs(x[i]) <= self.L:
67                  u_val[i] = self.V0 * eval_laguerre(5, np.abs(x[i]))
68              else:
69                  u_val[i] = self.L
70          return u_val
71
72      def q(self, e, x):
73          return 2.0 * (e - self.u_func(x))
74
75
76      @staticmethod
77      def derivative_func(y, h, m):
78          return (y[m - 2] - y[m + 2] + 8.0 * (y[m + 1] - y[m - 1])) / (12.0 *
    h)
79
80
81      def normalize_wave_function(self, y):
82          norm = np.sqrt(np.trapz(y ** 2, self.X))
83          return y / norm
84
85
86      @staticmethod
87      def mean_momentum(psi, x):
88          h_bar = 1.0
89          d_psi_dx = np.gradient(psi, x)
90          integrand = psi.conj() * d_psi_dx
91          mean_px = -1j * h_bar * np.trapz(integrand, x)
92          return mean_px.real
93
94
95      @staticmethod
96      def mean_square_momentum(psi, x):
97          h_bar = 1.0
98          d2_psi_dx2 = np.gradient(np.gradient(psi, x), x)
99          integrand = psi.conj() * d2_psi_dx2
100         mean_px2 = -h_bar**2 * np.trapz(integrand, x)
101         return mean_px2.real
102
103     def f_fun(self, e, n):
104         f = np.array([self.c * self.q(e, self.X[i]) for i in np.arange(n)])
105         self.Psi[0] = 0.0
106         self.Fi[n - 1] = 0.0
107         self.Psi[1] = self.d1
108         self.Fi[n - 2] = self.d2
109
110         for i in np.arange(1, n - 1, 1):
111             p1 = 2.0 * (1.0 - 5.0 * f[i]) * self.Psi[i]
112             p2 = (1.0 + f[i - 1]) * self.Psi[i - 1]
113             self.Psi[i + 1] = (p1 - p2) / (1.0 + f[i + 1])
114
115         for i in np.arange(n - 2, 0, -1):
```

```
            f1 = 2.0 * (1.0 - 5.0 * f[i]) * self.Fi[i]
            f2 = (1.0 + f[i + 1]) * self.Fi[i + 1]
            self.Fi[i - 1] = (f1 - f2) / (1.0 + f[i - 1])

        p1 = np.abs(self.Psi).max()
        p2 = np.abs(self.Psi).min()
        big = p1 if p1 > p2 else p2

        self.Psi[:] = self.Psi[:] / big

        coefficient = self.Psi[self.r] / self.Fi[self.r]
        self.Fi[:] = coefficient * self.Fi[:]

        return Solver.derivative_func(self.Psi, self.h, self.r) - Solver.
    derivative_func(self.Fi, self.h, self.r)

    def energy_scan(self, e_min, e_max, step):
        energies = []
        values = []
        e = e_min
        while e <= e_max:
            f_value = self.f_fun(e, self.n)
            energies.append(e)
            values.append(f_value)
            e += step
        return energies, values

    def find_exact_energies(self, e_min, e_max, step, tol):
        energies, values = self.energy_scan(e_min, e_max, step)
        exact_energies = []
        for i in range(1, len(values)):
            log1 = values[i] * values[i - 1] < 0.0
            log2 = np.abs(values[i] - values[i - 1]) < self.limit_value
            if log1 and log2:
                e1, e2 = energies[i - 1], energies[i]
                exact_energy = self.bisection_method(e1, e2, tol)
                self.f_fun(exact_energy, self.n)
                exact_energies.append(exact_energy)
        return exact_energies

    def bisection_method(self, e1, e2, tol):
        while abs(e2 - e1) > tol:
            e_mid = (e1 + e2) / 2.0
            f1, f2, f_mid = self.f_fun(e1, self.n), self.f_fun(e2, self.n),
    self.f_fun(e_mid, self.n)
            if f1 * f_mid < 0.0:
                e2 = e_mid
            else:
                e1 = e_mid
            if f2 * f_mid < 0.0:
                e1 = e_mid
            else:
                e2 = e_mid
        return (e1 + e2) / 2.0

    def plot_wave_functions(self, energies):
        for i, E in enumerate(energies):
            self.f_fun(E, self.n)
            psi_norm = self.normalize_wave_function(self.Psi.copy())
            fi_norm = self.normalize_wave_function(self.Fi.copy())
```

```
            mean_px = Solver.mean_momentum(fi_norm, self.X)
            mean_px2 = Solver.mean_square_momentum(fi_norm, self.X)
            file = open("result.txt", "w")
            file.close()
            file1 = open("result.txt", "a")
            print(f"Condition {i}: E = {E:.6f}, <p_x> = {mean_px:.6e}, <p_x^2> = {mean_px2:.6e}")
            print(f"Condition {i}: E = {E:.6f}, <p_x> = {mean_px:.6e}, <p_x^2> = {mean_px2:.6e}", file = file1)

            plt.scatter(self.X[self.r], psi_norm[self.r], color='red', s=50, zorder=5)  # Point at Psi
            plt.scatter(self.X[self.r], fi_norm[self.r], color='blue', s=50, zorder=5)  # Point at Fi
            plt.plot(self.X, [self.u_func(x) for x in self.X], 'g-', linewidth=6.0, label="U(x)")
            plt.plot(self.X, psi_norm, label=f"Normalized condition Psi {i}")
            plt.plot(self.X, fi_norm, '--', label=f"Normalized condition Phi {i}")
            plt.title(f"Condition {i} (Normalized) for E = {E:.4f}")
            plt.xlabel("X")
            plt.ylabel("Normalized wave functions")
            plt.grid(True)
            plt.legend()
            plt.savefig(f"Condition_{i}_(normalized).jpg", dpi=300)
            plt.show()


            prob_density_psi = psi_norm**2
            prob_density_fi = fi_norm**2
            plt.plot(self.X, [self.u_func(x) for x in self.X], 'g-', linewidth=6.0, label="U(x)")
            plt.plot(self.X, prob_density_psi, label=f"Probability density Psi condition {i+1}")
            plt.plot(self.X, prob_density_fi, '--', label=f"Probability Density Phi condition {i+1}")
            plt.title(f"Condition {i} - Probability density where E = {E:.4f}")
            plt.xlabel("X")
            plt.ylabel("Probability density")
            plt.grid(True)
            plt.legend()
            plt.savefig(f"Condition_{i}_(Probability_density).jpg", dpi=300)
            plt.show()


    def solve(self):
        e_min, e_max, step = self.U_min + 0.01, 3.0, 0.01
        exact_energies = self.find_exact_energies(e_min, e_max, step, self.tol)

        if len(exact_energies) == 0:
            print("Error: energies were not found.")
        else:
            print("Energies:")
            for i, E in enumerate(exact_energies):
                print(f"Condition {i}: Energy = {E:.6f}")

```

```
221                 self.plot_wave_functions(exact_energies)
```

Листинг 1: Код файла solver.py

Список литературы

[1] Тимошенко Ю.К. *Численное решение стационарного уравнения Шрёдингера.* Воронеж, 2019.

[2] Доля П.Г. *Введение в научный Python* Харьков: ХНУ, 2016. 265 с.

[3] Тимошенко Ю.К. *Численное решение стационарного уравнения Шрёдингера: теория возмущений. Учебное пособие* Воронеж: Научная книга, 2019. 14с.