МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет *компьютерных наук*

Кафедра *цифровых технологий*

Использование метода перевала в нестационарных задачах квантовой механики

ВКР *Бакалаврская работа*
02.03.01 Математика и компьютерные науки
Распределенные системы и искусственный интеллект

Допущено к защите в ГЭК

Зав. кафедрой _____ *С.Д. Кургалин, д. ф.-м. н., профессор* __ . __ . 2017
Обучающийся _____ *А.А. Махно, 4 курс, д/о*
Руководитель _____ *А.В. Флегель, к. ф.-м. н., доцент*

Воронеж 2017

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных наук
Кафедра цифровых технологий

УТВЕРЖДАЮ
заведующий кафедрой
_____ Кургалин С.Д.,
подпись *расшифровка подписи*
__ . __ . 2017

ЗАДАНИЕ
НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
ОБУЧАЮЩЕГОСЯ МАХНО АРТЕМА АНДРЕЕВИЧА
фамилия, имя, отчество

1. Тема работы «Использование метода перевала в нестационарных задачах квантовой механики», утверждена решением ученого совета факультета компьютерных наук от ___.___.20___
2. Направление подготовки / специальность 02.03.01 Математика и компьютерные науки
шифр, наименование
3. Срок сдачи студентом законченной работы ___.___.20___
4. Календарный план: (строится в соответствии со структурой ВКР)

№	Структура ВКР	Сроки выполнения	Примечание
1	Введение	26.10.2016	
2	Глава 1. Постановка задачи	30.11.2016	
3	Глава 2. Описание метода перевала	16.01.2017	
4	Глава 3. Методы вычисления функции M	21.03.2017	
5	Глава 4. Анализ результатов	12.04.2017	
6	Заключение	3.05.2017	

Обучающийся _____ Махно А.А.
подпись *расшифровка подписи*

Руководители _____ Флегель А.В.
подпись *расшифровка подписи*

Реферат

Бакалаврская работа 48 с., 17 рисунков

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, КВАНТОВАЯ МЕХАНИКА, МЕТОД ПЕРЕВАЛА, ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ.

Объектом исследования являются интегралы от быстро осциллирующих функций, возникающие в квантово-механических задачах моделирования взаимодействия атома с лазерным импульсом.

Цель работы – получение аналитических формул для решений нестационарных уравнений квантовой механики с использованием метода перевала, а также выявление границ их применимости.

В процессе выполнения работы была получена аналитическая формула перевальной оценки интеграла от быстро осциллирующей функции, интенсивность осцилляций которой определяется параметрами задачи. Для получения точного результата для интеграла разработан численный метод расчета с использованием библиотек численного интегрирования.

В результате работы была выявлена область параметров задачи, для которой наблюдается хорошее согласие полученной аналитической оценки с исходным интегралом. Проанализированы причины нарушения согласия вне указанной области параметров. Приведенный в работе анализ может рассматриваться как одно из обоснований приближенных аналитических квазиклассических оценок в задачах квантовой механики.

Результаты работы могут использоваться при математическом моделировании взаимодействия атомов и молекул с интенсивными электромагнитными полями, а также в других задачах теоретической физики.

Содержание

Введение

В последние годы активно развиваются высокопроизводительные вычисления с использованием массивных вычислительных кластеров и суперкомпьютеров. Область научных и технических задач, решаемых на параллельных многопроцессорных системах, стремительно расширяется и включает в себя сложные многочастичные динамические задачи, решение которых еще недавно казалось невозможным. Тем не менее, математическое и компьютерное моделирование некоторых экспериментально наблюдаемых процессов сталкивается с существенными сложностями, обусловленными, с одной стороны, все еще недостаточной мощностью вычислительных ресурсов, с другой – принципиальной невозможностью исключить ошибки численного решения даже при использовании наиболее продвинутых алгоритмов расчетов.

Примером такой задачи является описание взаимодействия атомов и молекул с интенсивным низкочастотным лазерным полем, влияние которого приводит к появлению ряда нелинейных многофотонных явлений, таких как, генерация высоких гармоник лазерного поля (атом излучает фотон с энергией, в десятки или сотни раз превышающей энергию фотона поля накачки) или надпороговая ионизация атомов (электрон вырывается из связанного состояния, поглощая десятки или сотни фотонов лазерного поля). Численное решение нестационарного уравнения Шредингера, являющегося математической (квантовомеханической) моделью атома в лазерном поле, является крайне сложной задачей даже в рамках одноэлектронной модели, в которой взаимодействие активного электрона с атомным остовом описывается модельным потенциалом. Сложность этой задачи связана с экспоненциальным ростом размерности задачи (из-за роста числа узлов сетки четырехмерного пространства $\mathbf{r} \otimes t$) при увеличении интенсивности лазерного поля или его длины волны. В связи с этим актуальным остается использование, во-первых, модельных подходов к описанию физики задачи, снижающих размерность исходных уравнений, и, во-вторых, приближенных методов вычислений, позволяющих получить простые аналитические результаты, удовлетворительно описывающие интересующие процессы как качественно, так и количественно. Ценность таких аналитических формул заключается также и в том, что они дают простое физическое объяснение изучаемого явления, что

более трудно извлечь из ресурсозатратных вычислений.

Одним из приближенных подходов для получения аналитического результата для квантовой амплитуды многофотонного процесса в интенсивном низкочастотном лазерном поле является квазиклассический подход, основанный на использовании метода перевала. **Целью** настоящей работы является получение перевальной оценки для волновой функции атомного электрона в лазерном поле и проверка применимости такой оценки путем сравнения с точным результатом численного расчета. При формулировании проблемы, будут получены основные соотношения для волновой функции квазистационарного состояния электрона в короткодействующем потенциале и выделены временные интегралы для ключевых элементов модели. При анализе этих интегралов будут рассмотрены решения уравнений на перевальные точки подынтегральных выражений и проанализирован их вклад. Для получения точного численного результата в работе используется алгоритм на базе быстрого преобразования Фурье для интегрирования медленно затухающей осциллирующей функции.

Текст работы организован следующим образом: постановка задачи представлена в главе 1; в главе 2 описан принцип метода перевала; в главе 3 получена аналитическая формула для поставленной задачи, и анализируется метод численного интегрирования исходной функции; в главе 5 проведено сравнение численного и аналитического решения; наконец, в заключении приводятся основные результаты работы.

1 Постановка задачи

В квантовой механике одноэлектронная математическая модель атома, подверженного воздействию лазерного импульса, описывается нестационарным уравнением Шредингера для электронной волновой функции $\Psi(\mathbf{r}, t)$:

$$\left[i\hbar\frac{\partial}{\partial t} + \frac{\hbar^2}{2m}\nabla^2 - U(r) - V(\mathbf{r}, t) \right] \Psi(\mathbf{r}, t) = 0, \tag{1.1}$$

где $U(r)$ – потенциал взаимодействия электрона с атомным остовом, $V(\mathbf{r}, t) = |e|(\mathbf{F}(t) \cdot \mathbf{r})$ – взаимодействие электрона (с зарядом e и массой m) с лазерным импульсом, электрическое поле которого описывается вектором напряженности $\mathbf{F}(t)$.[1] Пусть электрон-атомное взаимодействия моделируется короткодействующим потенциалом $U(r)$, поддерживающим одно связанное сферически-симметричное состояние $\Psi_0(\mathbf{r}, t)$ с энергией $E_0 = -\hbar^2\kappa^2/(2m)$.

Для больших r, вне области действия потенциала $U(r)$ волновая функция $\Psi(\mathbf{r}, t)$ ведет себя как суперпозиция расходящихся волн, соответствующих потоку уходящих на бесконечность частиц с энергиями $E = \mathcal{E}(F, \omega) + n\hbar\omega$, где $\mathcal{E}(F, \omega)$ – положение атомного уровня в лазерном поле (т.е. энергия эволюционировавшего в поле $\mathbf{F}(t)$ состояния Ψ_0), $\hbar\omega$ – энергия лазерного фотона, соответствующая несущей частоте ω.[2] Функция $\Psi(\mathbf{r}, t)$ с требуемым асимптотическим поведением при $r \to \infty$ может быть выражена через нестационарную функцию Грина $G(\mathbf{r}, t; \mathbf{r}', t')$ для свободного электрона в поле $\mathbf{F}(t)$:

$$\Psi(\mathbf{r}, t) = -\frac{2\pi\hbar^2}{m\kappa} \int_{-\infty}^{t} e^{-i\epsilon t'/\hbar} G(\mathbf{r}, t; 0, t') f(t') dt', \tag{1.2}$$

где $f(t)$ – неизвестная функция, которую необходимо определить из граничного условия для $\Psi(\mathbf{r}, t)$ при $r \to 0$, ϵ – квазиэнергия состояния $\Psi(\mathbf{r}, t)$, переходящая в E_0 при выключении лазерного поля.[3] Функция Грина $G(\mathbf{r}, t; \mathbf{r}', t')$ удовлетворяет уравнению

$$\left[i\hbar\frac{\partial}{\partial t} + \frac{\hbar^2}{2m}\nabla^2 - V(\mathbf{r}, t) \right] G(\mathbf{r}, t; \mathbf{r}', t') = \delta(\mathbf{r} - \mathbf{r}')\delta(t - t'), \tag{1.3}$$

и имеет следующий вид:

$$G(\mathbf{r}, t; \mathbf{r}', t') = -\theta(t - t')\frac{i}{\hbar} \left[\frac{m}{2\pi i\hbar(t - t')} \right]^{3/2} e^{iS(\mathbf{r}, t; \mathbf{r}', t')/\hbar}, \tag{1.4}$$

где $\theta(x)$ – функция Хевисайда, и S – классическое действие для электрона в лазерном поле $\mathbf{F}(t)$:

$$S(\mathbf{r},t;\mathbf{r}',t') = \frac{m}{2(t-t')}\left(\mathbf{r}-\mathbf{r}'-\frac{e}{mc}\int_{t'}^{t}\mathbf{A}(\tau)d\tau\right)^2 - \frac{e^2}{2mc^2}\int_{t'}^{t}\mathbf{A}^2(\tau)d\tau - \frac{e}{c}[\mathbf{r}\mathbf{A}(t)-\mathbf{r}'\mathbf{A}(t')]. \quad (1.5)$$

В (1.5) $\mathbf{A}(t)$ – векторный потенциал лазерного поля,

$$\mathbf{F}(t) = -\frac{1}{c}\frac{\partial\mathbf{A}(t)}{\partial t}. \quad (1.6)$$

При $r \to 0$ функция $\Psi(\mathbf{r},t)$ удовлетворяет следующему краевому условию:

$$\Psi(\mathbf{r},t) \simeq \left(\frac{1}{r}+B(\epsilon)\right)f(t)e^{-i\epsilon t/\hbar}, \quad (1.7)$$

где явный вид зависимости $B(\epsilon)$ определяется характером взаимодействия $U(r)$. Например, для потенциала нулевого радиуса $B = -\kappa = -\sqrt{2m|E_0|}/\hbar =$ const.

Сшивая решение (1.2) с краевым поведением (1.7), получим систему уравнений для коэффициентов Фурье функции $f(t)$,

$$f_n = \frac{1}{\mathcal{T}}\int_0^{\mathcal{T}} f(t)e^{in\omega_\tau t}dt, \quad \omega_\tau = \frac{2\pi}{\mathcal{T}}, \quad (1.8)$$

где $\mathcal{T}$ – временной интервал между последовательными импульсами лазерного поля (рассматривается бесконечная периодическая последовательность импульсов).[4] Результирующая система уравнений для f_n имеет вид:

$$\sum_{n=-\infty}^{\infty}\mathcal{R}(\epsilon+n\hbar\omega_\tau)f_n e^{-in\omega_\tau t} = \sum_{m=-\infty}^{\infty}e^{-im\omega_\tau t}f_m\mathcal{M}(\epsilon+m\hbar\omega_\tau,t), \quad (1.9)$$

где

$$\mathcal{R}(E) = B(E) - i\sqrt{2mE}/\hbar, \quad (1.10)$$

$$\mathcal{M}(\epsilon,t) = \sqrt{\frac{m}{2\pi i\hbar}}\int_0^{\infty}\frac{e^{i\epsilon\tau/\hbar}}{\tau^{3/2}}[e^{iS(t,t-\tau)/\hbar}-1]d\tau.[5] \quad (1.11)$$

В данной работе мы анализируем ключевой элемент системы (1.9) – матричный элемент $\mathcal{M}$. Используя метод перевала, будет получена аналити-

ческая оценка для интеграла в (1.11), применимость которой будет проверена численным сравнением с точным результатом для $\mathcal{M}(\epsilon, t)$. В дальнейшем будет использоваться атомная система единиц, в которой $|e| = m = \hbar = 1$.

Явный вид напряженности электрического поля лазерного импульса $\mathbf{F}(t) = \mathbf{e}_z\, F(t)$ задается следующей функцией:

$$S(t, t') = -\frac{1}{2}\int_{t'}^{t} [\alpha(\epsilon, t, t')]^2 d\epsilon, \tag{1.12}$$

где

$$\alpha(\epsilon, t, t') = A(\epsilon) - \frac{1}{(t - t')}\int_{t'}^{t} A(\tau) d\tau, \tag{1.13}$$

$$A(t) = -\int_{-\infty}^{t} F(\tau) d\tau. \tag{1.14}$$

Функции $F(t)$ и $A(t)$ представлены на рисунке (1.1)

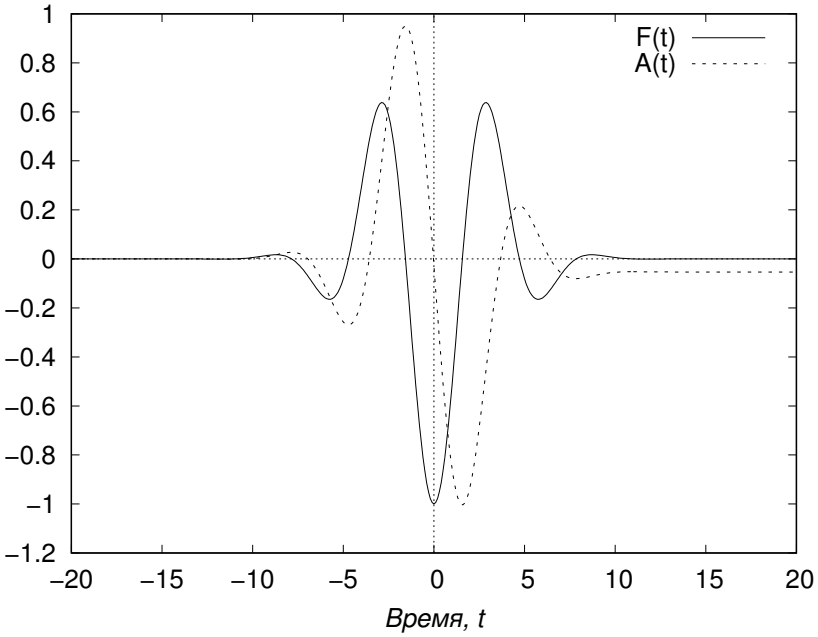

Рисунок 1.1 – Функции $F(t)$ и $A(t)$

2 Описание метода перевала

Метод перевала применяется для оценки при больших значениях параметра λ контурных интегралов вида

$$I(\lambda) = \int_C \phi(t) e^{\lambda f(z)} dz, \tag{2.1}$$

где $f(z)$ и $\phi(z)$ функции, аналитические вдоль линии интегрирования C. Интегралами вида (2.1) представляются многие специальные функции, решения дифференциальных уравнений, как обыкновенных, так и с частными производными. Эти интегралы часто встречаются при решении различных задач физики.[6]

Рассмотрим частный случай, а именно - действительные интегралы вида

$$I(\lambda) = \int_a^b \phi(t) e^{\lambda f(t)} dt. \tag{2.2}$$

Этот случай был рассмотрен Лапласом.

Предположим, что $f(t)$ имеет на отрезке (a, b) один резко выраженный максимум. Чем больше значение параметра λ, тем резче выражается этот максимум, и поэтому ясно, что при больших λ основной вклад в значение интеграла дает окрестность точки максимума.

В основе этого метода лежит лемма:

Лемма : Пусть дан интеграл

$$I(\lambda) = \int_0^a \phi(t) e^{-\lambda t^\alpha} dt \quad (0 < a \leq \infty, \alpha > 0),$$

где $\phi(t)$ при $|t| < 2h$ представляется сходящимся рядом

$$\phi(t) = t^\beta (c_0 + c_1 t + \cdots + c_n t^n + \dots), \ \beta > -1,$$

причем $\int_0^a |\phi(t)| e^{-\lambda_0 t^\alpha} dt \leq M$ для некоторого λ_0. Тогда имеет место асимптотическое разложение

$$I(\lambda) \sim \sum_{n=0}^{\infty} \frac{c_n}{\alpha} \Gamma\left(\frac{\beta + n + 1}{\alpha}\right) \lambda^{-\frac{\beta+n+1}{\alpha}}, \tag{2.3}$$

где Γ - гамма-функция Эйлера.

К доказанной лемме сводится оценка интеграла (2.2).

Теорема 1. Пусть интеграл (2.2) абсолютно сходится для некоторого $\lambda = \lambda_0$, т. е.

$$\int_a^b |\phi(t)| e^{\lambda_0 f(t)} dt \leq M,$$

и $f(t)$ достигает своего наибольшего значения во внутренней точке t_0 отрезка $(, b)$, в окрестности $|t - t_0| < \delta$ которой $f(t)$ представляется рядом

$$f(t) = f(t_0) + a_2(t - t_0)^2 + \cdots + a_n(t - t_0)^n + \ldots \quad (a_2 < 0),$$

причем существует $h > 0$ такое, что вне этой окрестности $f(t_0)$—$f(t) > h$. Пусть еще функция $t = \psi(\tau)$ определяется в окрестности точки $\tau = 0$ из уравнения $f(t_0)$—$f(t) = \tau^2$, причем в этой окрестности

$$\phi[\psi(\tau)]\psi'(\tau) = \sum_{n=0}^{\infty} c_n \tau^n. \tag{2.4}$$

Тогда интеграл (2.2) имеет асимптотическое разложение

$$I(\lambda) = \int_a^b \phi(t) e^{\lambda f(t)} dt \sim e^{\lambda} f(t_0) \sqrt{\frac{\pi}{\lambda}} \sum_{n=0}^{\infty} \frac{c_2 n}{\lambda^n} \frac{(2n)!}{4^n n!}.$$

Эта теорема относится к случаю, когда наибольшее значение $f(t)$ достигается во внутренней точке отрезка (a, b).

Теорема 2. Пусть интеграл (2.2) абсолютно сходится для некоторого $\lambda = \lambda_0$ (см теорему 1) и $f(t)$ достигает наибольшего значения в точке $t = a$, аналитична в этой точке ($f'(a) \neq 0$), и существует $h > 0$ такое, что $f(a) - f(t) > h$ вне некоторой окрестности точки a. пусть еще функция $t = \psi(t)$ определяется в окрестности точки $\tau = 0$ из уравнения $f(a) - f(t) = \tau$, причем в этой окрестности имеет место разложение (2.4). Тогда

$$I(\lambda) = \int_a^b \phi(t) e^{\lambda f(t)} dt \sim \frac{e^{f(a)}}{\lambda} \sum_{n=0}^{\infty} \frac{n! c_n}{\lambda^n}. \tag{2.5}$$

Суть метода перевала состоит в том, что при больших значениях параметра λ величина интеграла

$$I(\lambda) = \int_C \phi(t) e^{\lambda f(z)} dz$$

в основном определяется тем участком пути интегрирования C, на котором $|e^{\lambda f(z)}| = e^{\lambda \mathrm{Re} f(z)}$, т. е. $\mathrm{Re} f(z)$ велика по сравнению со значениями на остальной части . При этом интеграл оценивается тем легче, чем меньше этот участок и чем круче падает величина $\mathrm{Re} f(z)$ В соответствии со сказанным, при применении метода перевала стараются деформировать путь интегрирования С в наиболее удобный путь $\widetilde{C}$, пользуясь тем, что по теореме Коши такая деформация не меняет величины интеграла.[7]

Чтобы уяснить вопрос геометрически, положим $z = +iy$, и представим

$$u = \mathrm{Re} f(z),$$

как поверхность S в пространстве $(, y, u)$. Так как функция и гармоническая, то S не может иметь точек максимума и минимума, а точки, в которых $f'(z) = 0$, будут для нее точками перевала (седловыми точками, рисунок 2.1).

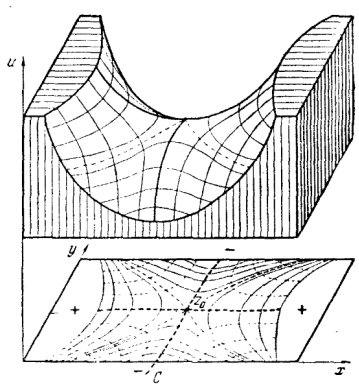

Рисунок 2.1 – Седловые точки

Наиболее удобный для оценки путь интегрирования $\widetilde{C}$, в каждой точке должен проходить в направлении наиболее быстрого изменения $\mathrm{Re} f(z)$, а так как функция f(z) аналитическая, то это направление должно совпадать с линией, на которой $\mathrm{Im} f(z) = \mathrm{const}$.

Также, новый контур $\widetilde{C}$ должен содержать точку z_0, в которой $\mathrm{Re} f(z)$ достигает наибольшего значения на $\widetilde{C}$. Покажем для этого случая, что $f'(z_0) = 0$, то есть точка линии $\mathrm{Im} f(z) = \mathrm{const}$, в которой $\mathrm{Re} f(z)$ достигает наибольшего значения, является точкой перевала.

Так и есть, ведь в точке z_0, которая является максимумом для $\mathrm{Re} f(z)$ производная $u = 0$ вдоль линии $\widetilde{C}$ должна быть равна 0, т. е. $\frac{\partial}{\partial s}\mathrm{Re} f(z) = 0$, а так как $\mathrm{Im} f(z) = \mathrm{const}$ на $\widetilde{C}$, то $\frac{\partial}{\partial s}\mathrm{Im} f(z) \equiv= 0$, а значит и

$$f'(z_0) = \frac{\partial}{\partial s}\mathrm{Re}f(z) + i\frac{\partial}{\partial s}\mathrm{Im}f(z) = 0.$$

Для метода перевала к интегралу (2.1) путь интегрирования следует деформировать в путь $\widetilde{C}$, проходящий через точку перевала z_0 и в окрестности этой точки идущий вдоль линии наибольшего ската $\mathrm{Im}f(z) = \mathrm{const}$ *(рисунок 2.1).*

Есть одно важное обстоятельство, обеспечивающее эффективность применения метода перевала: так как вдоль линии $\widetilde{C}$ имеем $\arg e^{f(z)} = \mathrm{Im}f(z) = \mathrm{const}$, то оценка интеграла (2.1) сводится к оценке интеграла от действительной функции, которая может быть проведена по методу Лапласа для интеграла вида (2.2). [8]

Именно это позволяет нам пользоваться полученными результатами теорем 1 и 2.

Рассмотрим случай, когда путь интегрирования можно деформировать в путь $\widetilde{C}$, проходящий через точку перевала z_0, где $f'(z_0) = 0$, $f''(z_0) \neq 0$, и в окрестности z_0 совпадающий с линией наибольшего ската $\mathrm{Im}f(z) = \mathrm{const}$, причем на $\widetilde{C}$ вне этой окрестности $\mathrm{Re}f(z) < \mathrm{Re}f(z_0) - h$ $(h > 0)$. Кроме того, предположим, что интеграл (2.1) абсолютно сходится для достаточно больших значений λ. Тогда образом, оценку интеграла можно провести на основании теоремы 1. Пусть $z = z(t)$ будет уравнение контура $\widetilde{C}$; Тогда,

$$I(\lambda) = \int_C \phi(z)e^{\lambda f(z)}dz = e^{\lambda i\mathrm{Im}f[z(t)]}\int_a^b \phi[z(t)]e^{\lambda \mathrm{Re}f[z(t)]}z'dt. \qquad (2.6)$$

и задача сводится к оценки интеграла вида 2.2 действительной области, разложение для которого уже было получено Лапласом, и имеет вид [9]:

$$I(\lambda) = \int_a^b \phi(t)e^{\lambda f(t)}dt \sim \frac{e^{f(a)}}{\lambda}\sum_{n=0}^{\infty}\frac{n!c_n}{\lambda^n}.$$

Выпишем первый член этого разложения. Обозначим $\phi[z(t)]z' = \widetilde{\phi}(t)$, $\mathrm{Re}f[z(t)] = \widetilde{f}(t)$ и тогда по формуле (2.5) получаем:

$$\int_a^b \widetilde{\phi}(t)e^{\lambda\widetilde{f}(t)}dt \sim e^{\lambda\widetilde{f}(t_0)}\sqrt{\frac{\pi}{\lambda}}\widetilde{c}_0, \qquad (2.7)$$

где $\widetilde{c}_0$ - свободный член в разложении функции $\widetilde{\phi}[\widetilde{\psi}(\tau)]\widetilde{\psi}'(\tau)$.

Имеем: $\widetilde{\phi}(t_0) = \phi(z_0)z'(t_0)$, и исходя из того, что $f[z(t)] = \mathrm{Re}f[z(t)] + i\mathrm{Im}f[z(t)] = \widetilde{f}(t) + \mathrm{const}$ вдоль $\widetilde{C}$, то

$$\widetilde{f}''(t_0) = \frac{d^2}{dt^2} f[z(t)] \mid_{t=t_0} = f''(z_0)z'^2(t_0),$$

причем $f'[z(t)]z''(t) = 0$ при $t = t_0$. Так как эта величина отрицательна, то представив $z'(t_0) = ke^{i\theta}$, можно записать ее в виде $\widetilde{f}'' = -|f''(z_0)|k^2$. Получаем, что

$$\widetilde{c}_0 = \widetilde{\phi}(t_0)\sqrt{-\frac{2}{\widetilde{f}''(z_0)}} = \phi(z_0)e^{i\theta}\sqrt{\frac{2}{|f''(z_0)|}}.$$

Подставим найденное значение в (2.7), а затем в (2.6), получаем искомую формулу

$$I(\lambda) \sim e^{\lambda f(z_0)}\sqrt{\frac{2\pi}{|f''(z_0)|}}\phi(z_0)e^{i\theta}\frac{1}{\sqrt{\lambda}}. \tag{2.8}$$

Как говорилось ранее, точка z_0 - это точка, где $\mathrm{Re}f(z)$ достигает своего максимального значения. В то же время совершенно обычная ситуация - когда на искомом контуре $\widetilde{C}$ имеется несколько точек перевала, в которых значения $\mathrm{Re}f(z)$ находятся вблизи к наибольшему, то следует взять сумму выражений (2.8) по всем этим точками.

Тот случай, когда контур интегрирования заканчивается в точке перевала z_0, аналогичным образом приводится к теореме 2.

Итак, мы получили рабочую формулу, подставляя в которую составляющие наших искомых функций $\phi(z)$ и $f(z)$, мы должны получать приближенные значения интеграла, когда $\lambda \to \infty$

В интеграле $\mathcal{M}(\epsilon, t)$ (1.11) роль большого параметра λ играет конструкция вида:

$$\lambda = \frac{1}{2\mathcal{T}}\int_{-\mathcal{T}/2}^{\mathcal{T}/2} A^2(\tau)d\tau. \tag{2.9}$$

3 Методы вычисления функции $\mathcal{M}(\epsilon, t)$

3.1 Аналитическое решение

Получим формулу оценки интеграла (1.11) при помощи метода перевала.

Множитель интеграла $\frac{1}{(t-t')^{3/2}}$ приводит к тому, что в случае, когда $t \to t'$ возникает бесконечность, которую необходимо учитывать. Конечно, в нашем случае, участок, на котором эта бесконечность возникает интересовать не должен, так как метод перевала применим для случая $\lambda \gg 1$, где λ из формулы (2.9).

Функция $S(t', t)$ при фиксированном t имеет вид:

Построим эту функцию

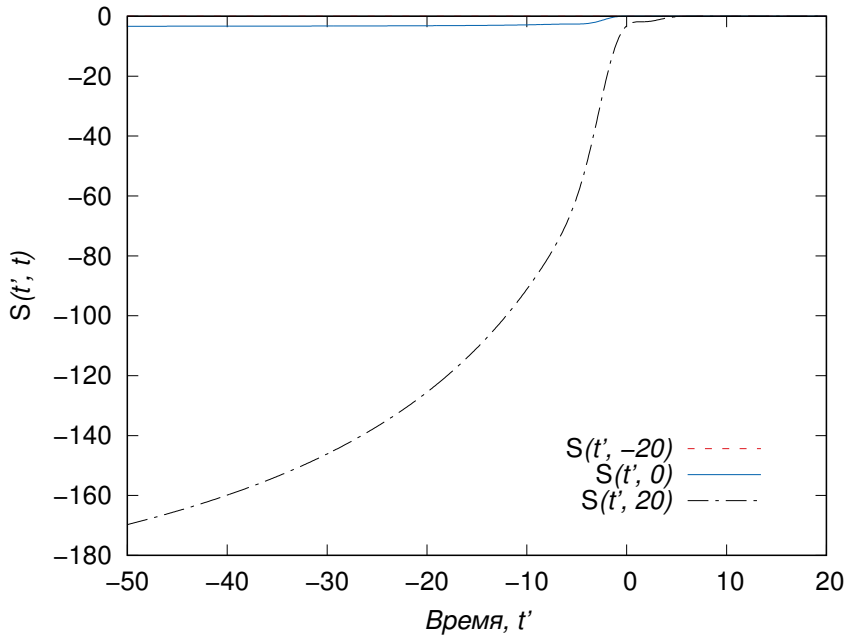

Рисунок 3.1 – Значение функции $S(t', t)$ для t=-20, 0 и 20

Из рисунка (3.1) видно, что большое значение функция S может принимать только в случае $|t - t'| \gg 1$. Несмотря на это следует рассмотреть случай $t = t'$ отдельно, в связи с тем, что вклад этой точки может оказывать большое влияние на результат. В этом случае ($t \to t'$) разложение функции S в ряд Тейлора принимает вид

$$S(\tau, t) \sim \frac{1}{24}\left(\frac{d}{dt}A(t)\right)^2 (t' - t)^3 + \frac{1}{24}\frac{d}{dt}A(t) \cdot \frac{d^2}{dt^2}A(t)(t' - t)^4 + ...$$

При втором приближении получаем

$$S(t', t) \sim \frac{1}{24} \left(\dot{A}(t)\right)^2 (t' - t)^3. \tag{3.1}$$

Подставляя это S в интеграл $\mathcal{M}$ и введем замену $t - t' = \tau$, получаем

$$\mathcal{M}_{\tau=0} = \frac{1}{\sqrt{2\pi i}} \int_0^\infty \frac{e^{i\epsilon\tau}}{\tau^{3/2}} \left(e^{-i\frac{\dot{A}(t)^2}{24}\tau^3} - 1\right) d\tau = \\ = \frac{1}{\sqrt{2\pi i}} \int_0^\infty e^{i\epsilon\tau}\tau^{3/2} d\tau \cdot \left(-i\frac{\dot{A}(t)^2}{24}\right)$$

Таким образом, вклад этой точки $\mathcal{M}_{\tau=0}$, при включении его в сумму учитывает элемент подынтегральной (-1), а значит при расчете остальных перевальных точек его можно не учитывать.

На рисунке 3.2 отображен график $\mathcal{M}_{\tau=0}$.

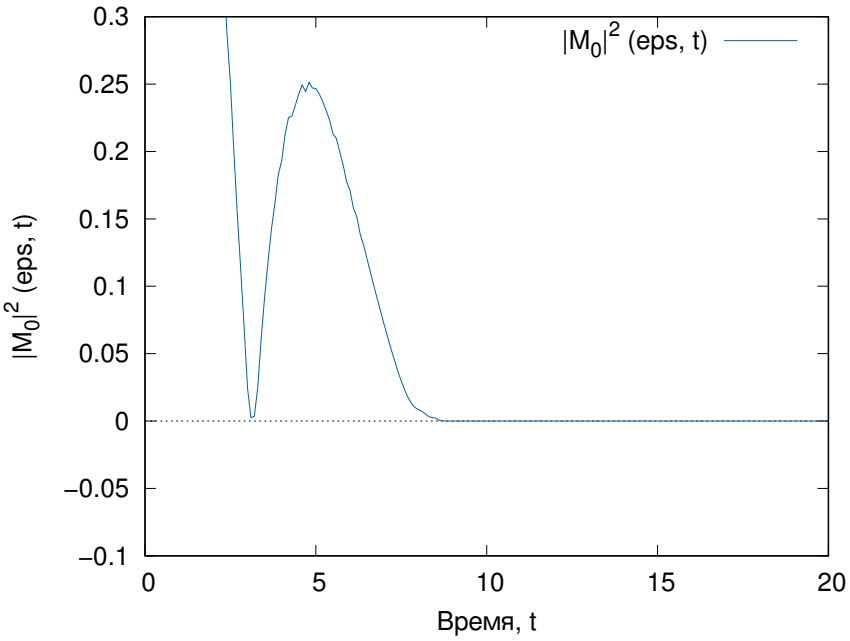

Рисунок 3.2 – Вклад интеграла $\mathcal{M}_{\tau=0}(\epsilon, t)$

Полученная функция, как видно из графика, очень быстро затухает. Поскольку при недостаточном отдалении от нуля ($t \approx 0$) даже для функции $S(t', t)$ малый размер функции не позволяет использовать перевальную оценку, то вкладом $\mathcal{M}_{\tau=0}(\epsilon, t)$ вблизи нуля, а также на остальном участке можно пренебречь (т. к. на интервале $t = (10, \infty)$ $\mathcal{M}_{\tau=0}(\epsilon, t) = 0$).

Тогда интеграл принимает вид:

$$\mathcal{M} \sim \int_{-\infty}^{t} \frac{1}{(t-t')^{3/2}} e^{i[\epsilon(t-t')+S(t,t')]} dt'.$$

После замены $t - t' = \tau$, получаем

$$\mathcal{M} \sim \int_{0}^{\infty} \frac{1}{\tau^{3/2}} e^{i[\epsilon\tau+S(t,t-\tau)]} d\tau.$$

Если обозначить $f(t,\tau) = \epsilon\tau + S(t, t-\tau)$ и $\phi(\tau) = \frac{1}{\tau^{3/2}}$, то получим формулу вида:

$$M \sim \int_{0}^{\infty} \phi(\tau) e^{if(t,\tau)} d\tau.$$

Которая совпадает с формулой для Метода Перевала, применимого для интегралов вида (2.1).

Для нашего случая приближение должно иметь вид:

$$M_0(\epsilon, t) \simeq \sqrt{\frac{1}{2\pi i}} \sum_{t0} e^{f(t,t_0)} \sqrt{-\frac{2}{\frac{\partial f(t,\tau)}{\partial \tau}|_{\tau=t_0}}} \phi(t_0),$$

где t_0 - корни уравнения $f'(t) = 0$.

Остается только получить формулу для $f''(t)$ и решить уравнение на стационарные точки.

Найдем теперь перевальные точки (t_0), дифференцируя по τ

$$\begin{gathered} f(t,\tau) = \epsilon\tau + S(t, t-\tau), \\ \frac{\partial f(t,\tau)}{\partial \tau} = \epsilon + \frac{\partial S'(t, t-\tau)}{\partial \tau}, \\ \frac{\partial f(t,\tau)}{\partial \tau} = 0 \Rightarrow \frac{\partial S'(t, t-\tau)}{\partial \tau} = -\epsilon. \end{gathered} \tag{3.2}$$

С введенной заменой S примет вид

$$S(t, t-\tau) = -\frac{1}{2m} \int_{t-\tau}^{t} \alpha(\epsilon, t, t-\tau)^2 d\epsilon$$

Дифференцируя $S(t,\tau)$ по τ, получаем

$$\frac{\partial S(t, t-\tau)}{\partial \tau} = -\frac{1}{2m} \alpha(t-\tau, t, t-\tau)^2.$$

Произведем обратную замену $t_0 = t - \tau$ и с учетом того, что $\frac{\partial S(t,t-\tau)}{\partial \tau} =$

$-\epsilon$ (3.2), получаем явный вид уравнения на стационарные точки:

$$\alpha^2(t_0, t, t_0) = 2\epsilon m. \tag{3.3}$$

Далее получим формулу для $\frac{\partial^2 f(t,\tau)}{\partial \tau^2}$:

$$\begin{aligned}\frac{\partial^2 f(t,\tau)}{\partial \tau^2} = -\frac{1}{2}\left[\alpha(t-\tau, t, t-\tau)^2\right]' = \\ = -\alpha(t_0, t, t_0)\alpha(t_0, t, t_0)'.\end{aligned}$$

Распишем эту функцию α:

$$\alpha(t_0, t, t_0) = \frac{|e|}{c}\left[A_\tau(t_0) - \frac{1}{(t-t_0)}\int_{t_0}^{t} A(\tau)d\tau\right].$$

Итого, получаем формулу для $f''(t,\tau)$ вида

$$\begin{aligned}\frac{\partial^2 f(t,\tau)}{\partial \tau^2} = |e|\alpha(t_0, t, t_0) \times \\ \times\left[F(t_0) - \tfrac{1}{(t-t_0)^2}\int_{t_0}^{t} A(\tau)d\tau + A(t_0)\tfrac{1}{t-t_0}\right].\end{aligned}$$

Обозначим $D = f''(t, t_0)$ и

$$\widetilde{S} = \epsilon\tau + S(t, t-\tau) = \epsilon(t-t_0) + S(t, t_0).$$

Подставляя в формулу для Метода перевала, получаем:

$$M_0(\epsilon, t) \simeq \sum_{t_0} \frac{e^{i\widetilde{S}(t,t_0)}}{\sqrt{D}(t-t_0)^{3/2}}. \tag{3.4}$$

Видно, что полученная формула включает в себя множество различных элементов, предполагающих численное интегрирование. Также перед ее вычислением необходимо решать уравнение на стационарные точки для каждого t. Рассмотрим сам процесс решения этих уравнений с точки зрения реализации в программе.

Решаем уравнение (3.3). если посмотреть на то, что из себя представляет это ϵ, можно заметить, что оно из формулы (1.9) относительно $\mathcal{M}$ может принимать различные значения, не только отрицательные , а также положи-

тельные, т. к.

$$\mathcal{M}(\epsilon + m\omega_\tau, t).$$

То есть ϵ, которое было представлено в формуле (1.11) это на самом деле конструкция вида $\epsilon + m\omega_\tau$. Тогда для случая $\epsilon > 0$.

Возникает 2 уравнения:

$$\alpha(t', t, t') - \sqrt{2\epsilon} = 0,$$
$$\alpha(t', t, t') + \sqrt{2\epsilon} = 0.$$

Объединяя множество решений (точек t') получаем искомые седловые точки (t_0).

Построим график функции $\alpha(t', t, t')$ на отрезке $t' = [-20..20]$ при $t = 0$ (рисунок 3.3):

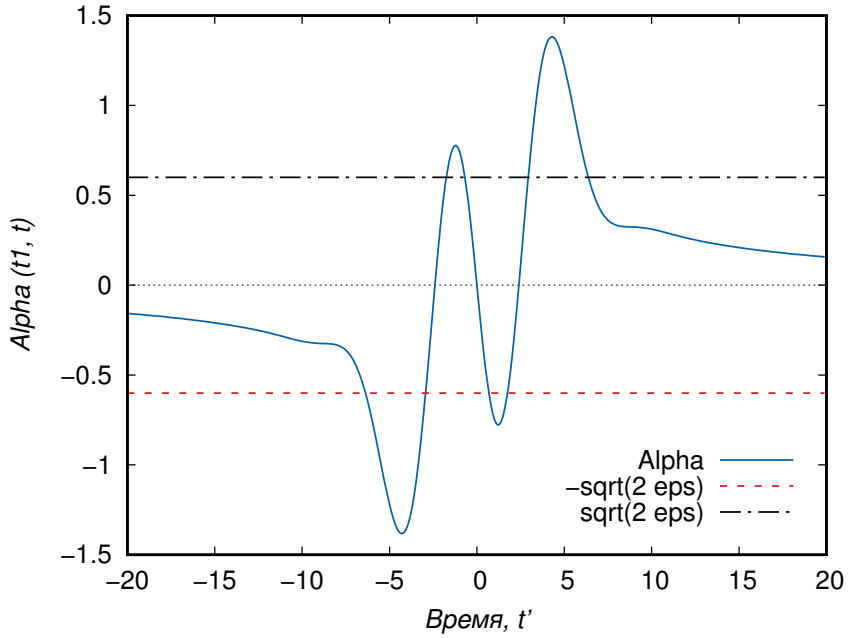

Рисунок 3.3 – Функция α при $t = 0$

Видно, что корней несколько и они практически симметричны относительно ветвей оси Oy. Важно, что в суммировании участвуют только решения на промежутке $t' = [-20, t]$. На графике показано, что не для каждого ϵ у нас найдутся решения в области действительных чисел. Максимальное значение ϵ при данных условиях будет около 0.6.

Корни этого уравнения могут быть, в общем случае, комплексные. Тогда необходимо решать систему ($t = const$):

$$\begin{cases} \text{Re } \alpha(x+iy,t,x+iy) = 2\epsilon \\ \text{Im } \alpha(x+iy,t,x+iy) = 0 \end{cases}$$

Здесь ϵ - любое действительное число. В этому случае потребуется производить интегрирование в области комплексных чисел, что вызывает определенные трудности. Проблема также возникает в поиске множества корней. Основным методом решения уравнений в нескольких измерениях является метод градиентного спуска, то есть должны быть описать также производные этих функций. Самым трудоемким в таком случае является именно поиск множества корней, так как возможна потеря некоторых из них во время перехода к следующей области поиска.

Вернемся к случаю $\epsilon > 0$. Если корни близки к пику функции $\alpha(t', t, t')$, то возможен случай слияний корней, который является еще одним ограничением использования перевальных оценок. Чтобы этого избежать, можно получить оценку не для двух точек, а для одной, которая находится между ними на равном удалении. Тогда разложение примет другой вид в связи с тем, что сделанное ранее предположение о равенстве нулю первой производной будет не верным. Оценка будет выражаться через функцию Эйри.[10] Поскольку явный вид ее объемный, приведен он не будет, но получение не вызывает трудностей.

На (рисунок 3.4) представлен график $\alpha(t', t, t')$ на отрезке $t' = [-20..20]$ при $t = 20$

Корней стало меньше, и они дальше друг от друга расположены. Но при предыдущем $\epsilon = 0.6$ (с рисунок 3.3) у нас для данных условиях не будет корней, то есть мы не сможем получить перевальную оценку для $t >> 0$. Поэтому для каждого конкретного случая следует следить за выбором ϵ и тем, существуют ли решения уравнения на моделируемом участке. Корни уравнений будут найдены методом Брента. [11]

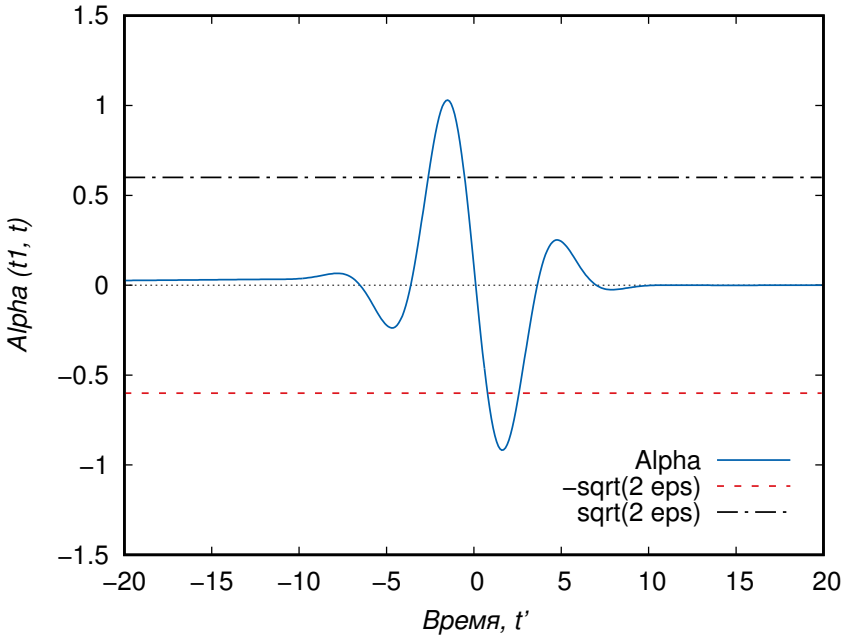

Рисунок 3.4 – Функция α при $t = 20$

3.2 Численное решение

Выполним численное интегрирование интеграла (1.11)

Заметим, что нижний предел интегрирования у нас $-\infty$. Очевидно, что с этими производить вычисления невозможно, но зато наша начальная функция под интегралом (1.14), имеющая вид (**??**) является затухающей. Тогда, решив уравнение

$$|e^{-\frac{t_b^2}{\alpha^2}}| < \epsilon,$$

на промежутке $[-\infty; 0]$, приближаясь слева, мы получим то самое значение t_b, при котором можно не учитывать отрезок интегрирования в связи с достижением необходимой точности.

Интеграл (1.14) примет вид

$$A(t) = -c \int_{t_b}^{t} F(\tau) d\tau.$$

Для вычисления этого интеграла воспользуемся методом интегрирования Гаусса. В этом методе точки интегрирования берутся с разными интервалами и при этом имеют различные веса ω_i, характеризующие их вклад в интеграл.

$$\int_{x_1}^{x_2} f(x) dx = \sum_{j=1}^{N} \omega_j f(x_j).$$

Метод Гаусса также может считать интегралы от неограниченных, быстро затухающий функция. Нам это не потребуется, но в связи с тем, что у нас этот интеграл (1.14) будет входить, далее, в подынтегральные функции, а также с необходимостью на каждом шаге пересчета матричного элемента, менять пределы интегрирования придется использовать адаптивные методы, позволяющие придерживаться заданной точности. Для этого воспользуемся библиотекой GNU GSL.

GNU GSL - Это библиотека, написанная на языке программирования C для численных вычислений в прикладной математике и науке.

Особенности GSL: написана полностью в Стандарте C (также применимое от C++) и основана на использовании заголовочных файлов, определяет новые типы, и структуры данных, у которых нет аналогов в Фортране или C. GSL использует порядок хранения данных на многомерных массивах,

который отличается от используемого Фортраном способом. Единственный способ, использование его в Фортране, состоит в том, чтобы записать собственные подпрограммы на C, чтобы обеспечить необходимое соответствие между типами данных, структурами и соглашениями хранения их для двух языков. Интерфейсный уровень не поставляется с библиотекой.

Самый главный ее плюс - скорость вычислений, а также относительная экономия памяти/очистка неиспользуемой. В ней реализовано огромное количество численных методов:

1. Базовые математические функции
2. Комплексные числа
3. Специальные функции
4. Вектора и матрицы
5. Комбинаторика
6. Сортировка
7. Линейная алгебра
8. БПФ
9. Численно интегрирование (основанное на QUADPACK)
10. Поиск корней уравнений и т. д.

Для нашей задачи важны только два алгоритма, а именно $gsl\_integration\_qags$ [12], позволяющий нам интегрировать функцию на отрезке с заданной точностью и $gsl\_root\_fsolver$, решающий уравнения методом Брента, не используя информацию о производной [13]

Стоит отменить, что для первого интеграла, получаемого библиотекой численно (1.14) GSL не смог добиться точности абсолютной ошибки выше 10^{-9}, а значит добиваться увеличения точности дальнейших расчетов не улучшит результат.

Далее, рассматривая функцию α (1.13), входящую в интеграл (1.12), имеет смысл перейти к интегралу $S(t', t)$ вида:

$$S(t,t') = \frac{1}{2}\int_{t'}^{t} \left(A(\epsilon) - A_1(t',t)\right)^2 d\epsilon, \tag{3.5}$$

где $A_1(t',t) = \frac{1}{t'-t}\int_{t'}^{t} A(\tau)d\tau$. Относительно внешнего интеграла по ϵ $A_1(t',t)$ остается постоянной, поэтому этот элемент можно посчитать только 1 раз для данных t', t. Этим преобразованием мы уменьшаем количество операций, но сложность для вычислений представляет именно последний интеграл M. Для его вычисления рассмотрим преобразование Фурье.

Преобразование Фурье анализирует функцию времени (сигнал) в частоты, которые составляют его.

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x\xi}\,dx$$

Когда функция и ее преобразование Фурье заменены дискретизированными дубликатами, ее называют дискретным преобразованием Фурье (DFT).

Дискретное преобразование Фурье преобразоует конечную последовательность равномерно распределенных выборок функции в последовательность эквивалентной длины выборки преобразования Фурье дискретного времени (DTFT), которое является комплексной функцией частоты. Интервал, в котором выполняется DTFT, является обратной величиной продолжительности входной последовательности. Обратное DFT - ряд Фурье, использующее выборки DTFT в качестве коэффициентов сложных синусоид на соответствующих частотах DTFT. Поэтому говорят, что DFT является представлением частотной области входной последовательности. Если исходная последовательность охватывает все ненулевые значения функции, ее DTFT непрерывный (и периодический). Если исходная последовательность - один цикл периодической функции, DFT обеспечивает все ненулевые значения одного цикла DTFT.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} =$$

$$= \sum_{n=0}^{N-1} x_n \cdot [\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)],$$

DFT стал оплотом числовых вычислений и необходимость быстрого

алгоритма для вычисления его привела к появлению Быстрого преобразования Фурье (FFT).

Алгоритм быстрого преобразования Фурье (FFT) вычисляет дискретное преобразование Фурье (DFT) последовательности или ее инверсию. Анализ Фурье преобразовывает сигнал (часто всего во времени или пространстве) к представлению в частотной области и наоборот. FFT быстро вычисляет такие преобразования, разлагая на множители матрицу DFT в продукт разреженных (главным образом нуль) факторов. В результате это приводит к уменьшению сложности вычислений DFT от $O(n^2)$, который возникает, при обычном применении DFT до $O(n \log n)$, где n является размером данных.

Быстрые преобразования Фурье широко используются для многих приложений в технике, науке и математике. Основные идеи были популяризированы в 1965, но некоторые алгоритмы были получены уже в 1805. В 1994, Gilbert Strang описал FFT как "самый важный числовой алгоритм нашего времени".

Заметим схожесть интеграла (1.11) с общим видом обратного преобразования Фурье

Введем замену $t - t' = \tau$, получаем

$$\mathcal{M}(t, \xi) = \int_0^\infty \frac{e^{i\epsilon\tau}}{\tau^{3/2}}(e^{i[S(t,t-\tau)} - 1)d\tau \tag{3.6}$$

$$\hat{F}(\xi) = \int_\infty^\infty e^{i\xi\tau} F(\tau)d\tau$$

Интегрирование можно ускорить, использовав не стандартные методы интегрирования, а используя дискретное преобразование Фурье.

Для этого воспользуемся библиотекой FFTW.

FFTW является библиотекой программного обеспечения для вычислений дискретных преобразований Фурье (DFTs).[14]

FFTW известен как самая быстрая реализация бесплатного программного обеспечения алгоритма Быстрого преобразования Фурье (FFT) (по регулярными сравнительными тестами). Этот алгоритм может выполнить дискретное преобразование Фурье от последовательности со сложным знаком произвольного размера, и размерности в $O(n \log n)$, время.

Реализовано это через поддержку множества различных алгоритмов, и выбор одного (определенное разложение преобразования в менее слож-

ные преобразования), который оценивают на оптимальность при аналогичных условиях. Библиотека работает лучше всего над массивами небольших размеров с небольшими простыми элементами, но также хорошо работает с массивами с большими начальными элементами, являющимися наихудшим случаем (сохраняя сложность $O(n, \log n)$). Разложение преобразовывает алгоритм дискретного преобразования Фурье в алгоритм с меньшим числом преобразований, (выбор идет среди нескольких вариантов Cooley–Tukey FFT алгоритмов), соответствующий различным факторизациям и/или различным образцам доступа к памяти, в то время как для больших размеров библиотека использует алгоритм Rader или Bluestein FFT. Как только преобразование было разбито в под-преобразования достаточно меньшей сложности, FFTW использует сложный разбор, разворачивал FFTs для этих небольших размеров, которые были произведены (во время компиляции, не во время выполнения) генерацией кода.

Для достаточно большого количества повторных преобразований выгодно сохранять результаты работы библиотеки для будущего использования алгоритмов на данном размере массива и платформе. Эти результаты, которые авторы именуют как ”мудрость могут быть сохранены в файле или другой последовательности для более позднего использования.

FFTW ограничил поддержку неисправных к текущему времени преобразований (использующих версию MPI). Поэтому периодически приходится пере-упорядочивать данные.

Именно скорость вычислений - главное что нам потребуется в нашей работе. И, несмотря на то, что повторные вычисления будут запущены на аналогичной архитектуре и, скорее всего, массиве аналогичного размера и вида, использовать сохраненные результаты компиляции прошлого запуска FFTW не имеет смысла, так как компиляция практически не занимает времени, а увеличение сложности вычислений ни коим образом не влияет на усложнение выходного кода библиотеки. Принципиальное усложнение вызывает только увеличение размера массива, но основную сложность вычислений составляет не само преобразование Фурье, а получение последовательности, над которой выполняется преобразование.

Исходя из формулы (3.6), если мы возьмем

$$F(\tau) = \frac{1}{\tau^{3/2}}(e^{i[+S(t,t-\tau)]} - 1),$$

то

$$\hat{F}(\psi) = \int_{-\infty}^{\infty} e^{i\phi\psi} F(\phi) d\phi.$$

А также если положить $\psi = \epsilon$, и использовать преобразование не отрезке от $[0, \infty]$, то $\hat{F}(0)$ будет равно

$$\hat{F}(\epsilon) = \mathcal{M}(t, \epsilon).$$

Для того, чтобы не подбирать шаг массива для преобразования Фурье, с целью точного попадания ϵ, можно использовать линейное приближение от соседних с ϵ точек. Например, если $x_1 < \epsilon < x_2$, то $\mathcal{M}(t, \epsilon)$ примет вид:

$$\mathcal{M}(t, \epsilon) = \frac{(\epsilon - x_1) \cdot (F(x_2) - F(x_1))}{x_2 - x_1} + F(x_1) \tag{3.7}$$

Выполняя лишнюю работу, так как считаем значение $\mathcal{M}$ для разных ϵ, используя этот способ, мы можем получить результат сразу для множества значений ϵ, не затрачивая на это много времени (поскольку само преобразование Фурье считается почти мгновенно).

Следует понимать, что опорных точек для интегрирования все равно остаться достаточно, чтобы была возможность уловить большую часть колебаний подынтегральной функции.

К тому же остается вопрос, каким же стоит брать конец отрезка интегрирования, ведь по изначальной формуле интегрировать нужно от 0 до ∞, но поскольку такой возможности нет, смоделируем различные отрезки для проверки результата.

На графике (3.5) изображены результаты численного интегрирования через FFTW при интегрировании на отрезках разной длинны (100 или 200).

Разница достаточно велика, значит следует брать число, намного больше 100. Построив больше графиков, был сделан вывод, что при использовании отрезка $[0, 350]$ получается очень хорошее отношение затраты/результат.

Видно также, что не только амплитуды у волн отличаются, но также и фаза. Это связано с природой преобразования Фурье, оптимальное использование которого возможно только в случае, если на границах массива, над которым производится преобразование , значение равны или очень близки ($f(x_0) \approx f(x_{end})$).

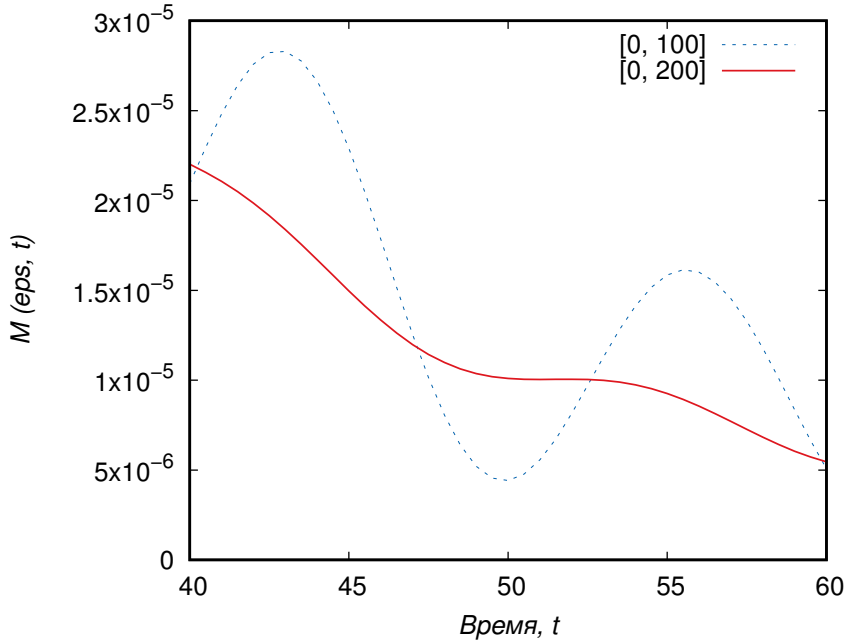

Рисунок 3.5 – Численное решения с использованием FFTW для разных отрезков

Так как в нашем случае в 0 значение функции равно нулю, а на конце отрезка близко к нулю, но не равно ему, то возникает бегущая волна.

Несмотря на то, что волна дает абсолютно малые значения, пренебрегать ими по порядку величины нельзя. Поскольку целью работы не является посмотреть на закономерность от ϵ, чтобы избежать лишних временных затрат будем использовать интегрирование по классическому определению интегральной суммы, то есть проводить интегрирования через площадь прямоугольника.[15]

Сравним с результатом, полученным через fftw (рисунок 3.6)

Результаты, полученные этим способом близки, но не совпадают. Это связанно с тем, что быстрое преобразование Фурье дает точные результаты для случаев, когда ϵ - целое число. Линейное приближение (3.7), полученное выше дает не самый лучший результат (в связи с величиной шага по ϵ). Его можно было бы улучшить, используя, например, сплайны второго порядка.

Несмотря на эти попытки увеличить производительность, процесс вычисления все еще занимает достаточно долгое время, даже при использовании библиотеки GSL. Теперь мы можем сравнить полученное численное решение с аналитическим.

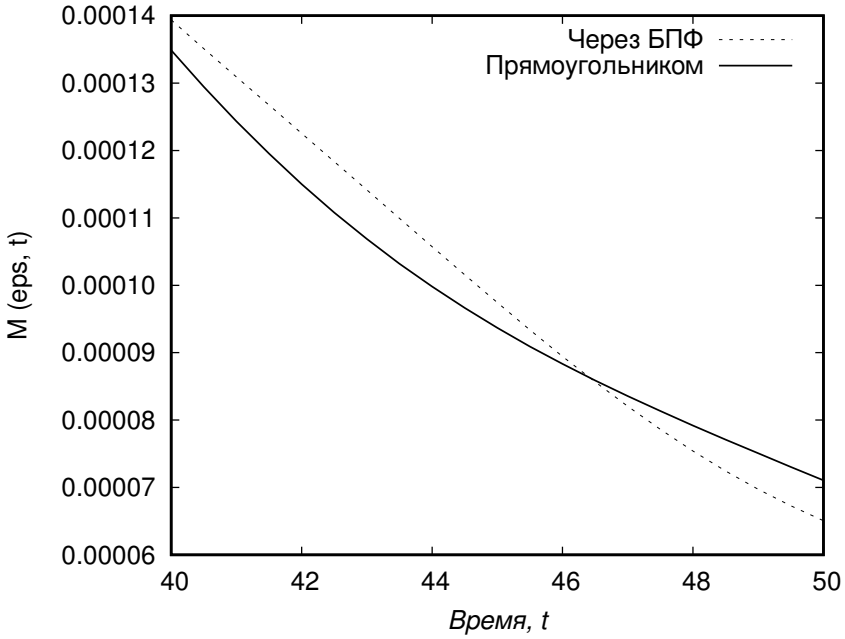

Рисунок 3.6 – Сравнение методов интегрирования

4 Анализ результатов

Теперь, когда мы разобрались по отдельности со способами решения задачи, необходимо сравнить полученные результаты. Расчет матричного элемента будем проводить на отрезке $t = [0, 70]$. Увеличить интервал нам не дает накапливающаяся численная ошибка . Значение функции $\mathcal{M}(\epsilon, t)$ на таком отдалении от 0 уже очень малые, порядка 10^{-7} и ошибка при суммирование 1024 точек с точностью получения каждой 10^{-9} начинает очень сильно влиять на результат.

Для простоты будет брать α, входящую в формулу (**??**) равной $\sqrt{20}$, а изменять будем другие параметры системы, такие, как ϵ, F_0 и ω .

Рассмотрим ситуацию, когда $\epsilon = 0.4,\ F_0 = 2,\ \omega = 0.8$. Для этих условий график корней примет такой вид (рисунок 4.1):

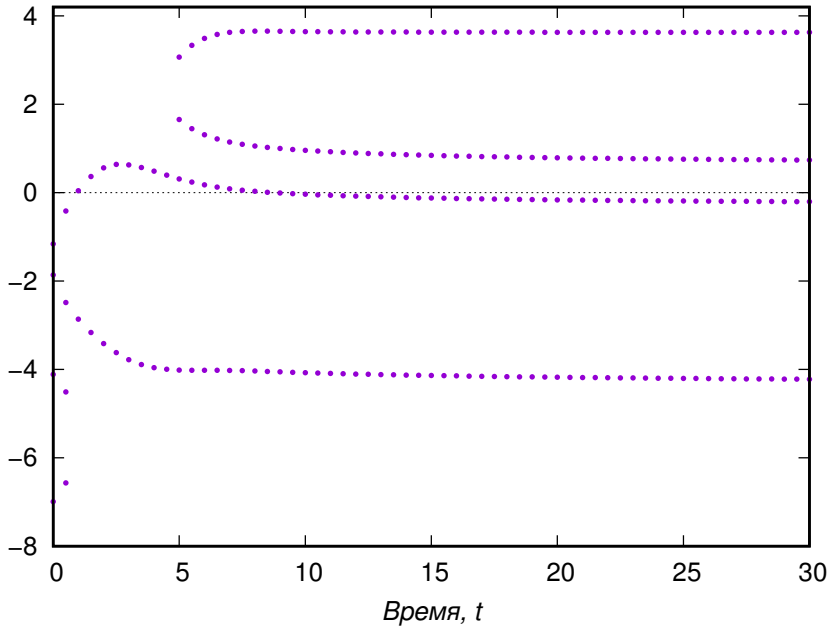

Рисунок 4.1 – Корни при $\epsilon = 0.4$, $\omega = 0.8$

Из графика видно, что новых корней при большом отдалении ($t \gg 0$) не возникает, а значит рассматривать отрезок такой длинны на появление новых корней не имеет смысла.

Чтобы убедится в этом построим график для других значений ϵ (рисунок 4.2):

Видно, что для квадратов ($\epsilon = 0.3$) коней больше, чем для остальных параметров, максимум 6, и последний появляется при $t \approx 11$. Подобная закономерность просматривалась на графике (рисунок 3.4), где уменьшение ϵ

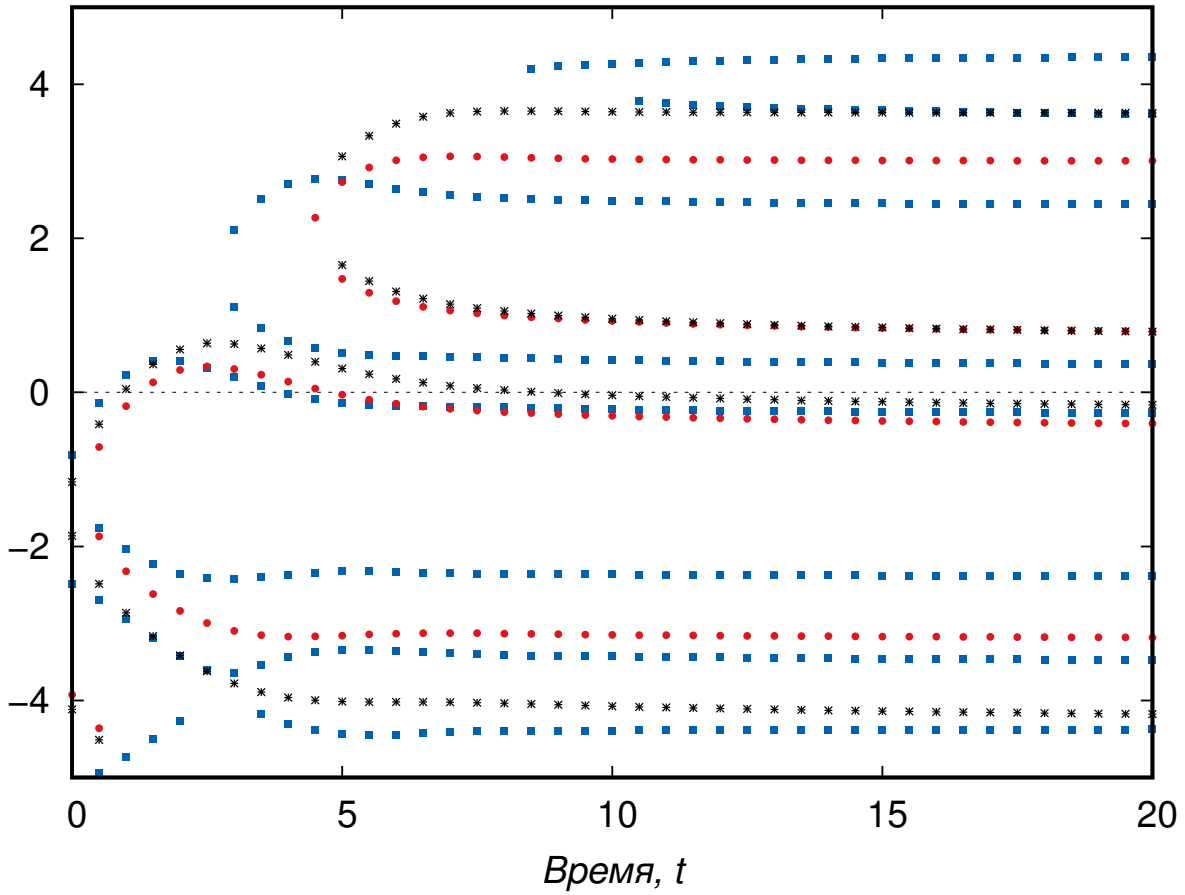

Рисунок 4.2 – Корни при $\epsilon = 0.3$ (квадраты), 0.35 (круги) и 0.4 (звездочки)

привело бы к увеличению числа корней.

На рисунках (4.3, 4.4 и 4.5) представлены графики сравнения решений для случая, когда $\epsilon = 0.4, F_0 = 2, \omega = 0.8$.

Исходя из рисунка (4.4) можно сказать, что пики на графике аналитического решения (рисунок 4.3) появляются в тех местах, где $t \approx t_0$ (t_0 - корень), в связи с тем, что множитель $\frac{1}{(t-t')^{3/2}}$ увеличивает значение функции на бесконечность. Вблизи этих пиков мы не можем получить хорошую оценку, но если $|t - t_0| \gg 1$, то аналитическая формула дает лучший результат, что видно на графике (рисунок 4.5).

На рисунках (4.6, 4.7 и 4.8) представлены графики сравнения решений для другого случая, когда $\epsilon = 0.4, F_0 = 2, \omega = 0.8$.

Полученный аналитический результат так же достаточно хорошо описывает матричный элемент $\mathcal{M}$, но как и говорилось до этого, точная оценка может быть получена не для всего временного промежутка времени. Сравнение графиков вблизи нуля показывает, что метод перевала не применим для случая $t \approx 0$, а также $t \approx t_0$. Стоит также отметить ожидаемый прирост

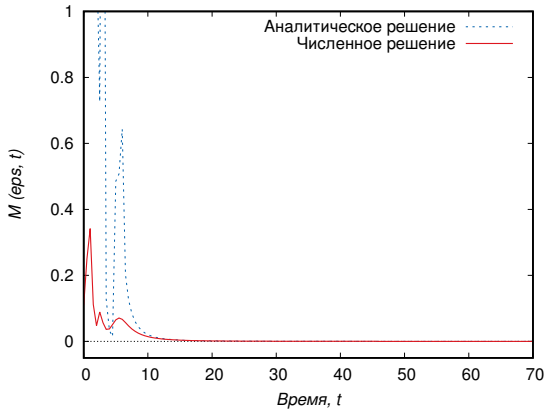

Рисунок 4.3 – Сравнение численного *(обычная линия)* и аналитического *(пунктирная)* решения. [0, 70]

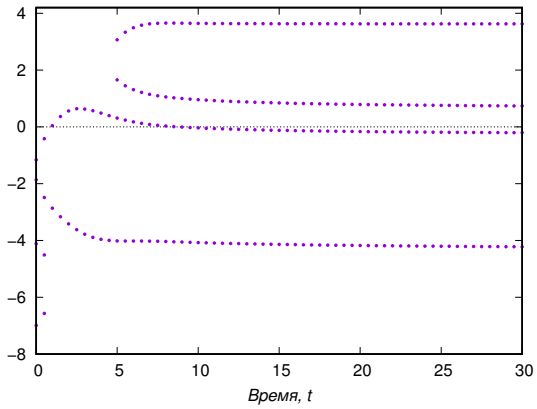

Рисунок 4.4 – Корни для аналитического решения

производительности. Ускорение действительно велико, аналитическая оценка получается быстрее примерно в 10 раз. Эту цифру можно увеличить, если использовать более быстрые алгоритмы для поиска корней уравнения, чем метод Брента.

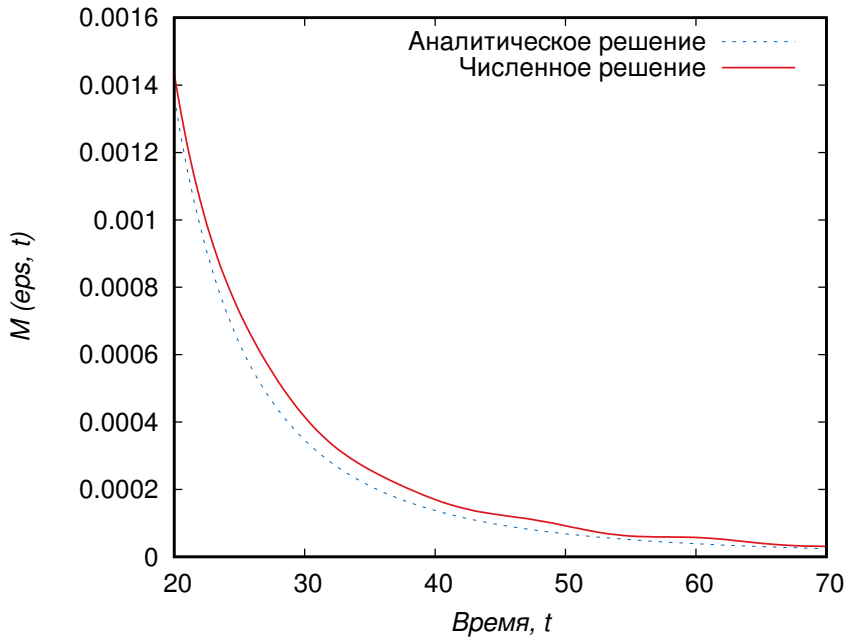

Рисунок 4.5 – Сравнение численного *(обычная линия)* и аналитического *(пунктирная)* решения. [20, 70]

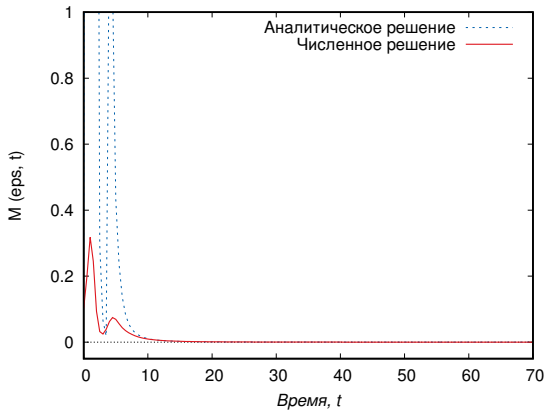

Рисунок 4.6 – Сравнение численного *(обычная линия)* и аналитического *(пунктирная)* решения

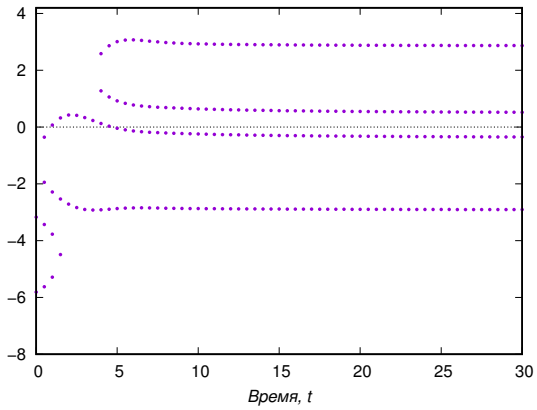

Рисунок 4.7 – Корни для аналитического решения

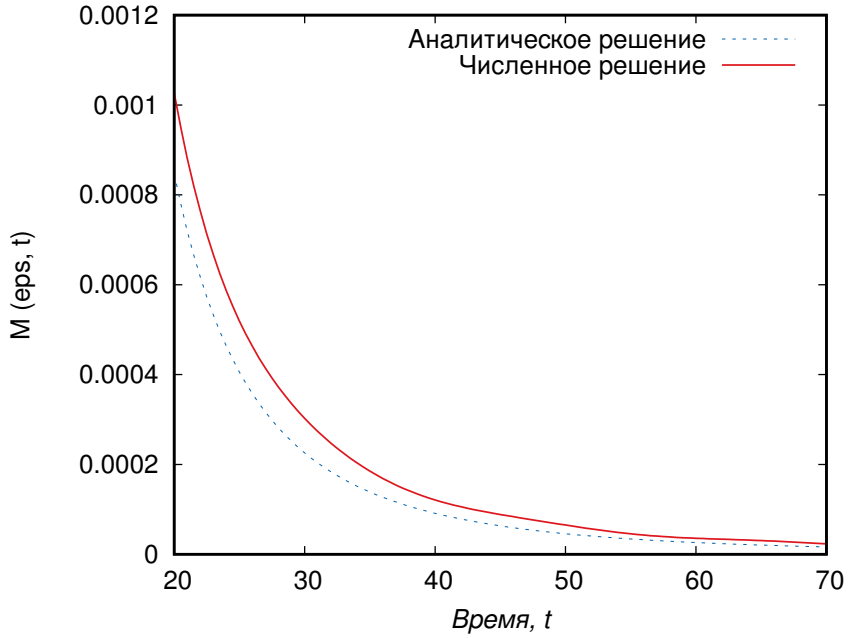

Рисунок 4.8 – Сравнение численного *(обычная линия)* и аналитического *(пунктирная)* решения

Заключение

В ходе работы была получена асимптотические оценки для матричного элемента $\mathcal{M}$, являющегося частью задачи моделирования атома под действием низко-частотного лазера, на основе метода перевала.

Также работы было представлено описание метод перевала, и способ применения к задаче. Затем была получена асимптотическая оценка как сумма по всем перевальным точкам и показано, почему точка $t = 0$ вносит очень малый вклад в искомый интеграл. Разработана и реализована программный модуль на языке C++, использующий библиотеки FFTW и GNU GSL для получения численной оценки и сравнения ее с полученной аналитической формулой. Также описаны плюсы и минусы использования быстрого преобразования Фурье, в качестве инструмента интегрирования.

Анализируя результаты можно сделать вывод, что данная оценка позволяет качественно оценить данный интеграл, при условии $t \gg 0$, а также $t \gg t_0$.

Главной проблемой при оценке является ограниченность ее применения. Она не может получить хорошую оценку для случая, когда $t \approx 0$, а также $t \approx t_0$, где t_0 - корни уравнения $\alpha^2(t', t, t') = 2\epsilon$. Зато для случая, когда получение численного результата начинает занимать большое количество времени, и возникают проблемы точности, полученная формула работает хорошо.

Можно было бы использовать для аналитических расчетов и аппроксимацию через функцию Эйри, для случая сливающихся корней, но в связи с тем, что корни близки только на том промежутке, на котором возникает сингулярность ($(t - t_0)^{-3/2}$), это не может принципиально улучшить оценку

Цель получить быструю качественную оценку была достигнута, ускорение составило порядка 10 раз. Причем можно сделать алгоритм еще быстрее, путем использования другого численного метода решения уравнения, так как почти все затраты ресурсов для получения оценки методом перевала занимает именно оно.

Список используемой литературы

1 Ionization by few-cycle pulses: Tracing the electron orbits [Text] / D. B. Milosevic, G.G. Paulus, D. Bauer, W. Becker // Phys. Rev. A. — 2005. — Vol. 71. — 061404.

2 Model-Independent Quantum Approach for Intense Laser Detachment of a Weakly Bound Electron [Text] / M. V. Frolov, N.L. Manankov, E.A. Pronin, A.F. Starace // Phys. Rev. Lett. — 2003. — Vol. 91. — 052003.

3 Frolov, M. V. Effective-range theory for an electron in a short-range potential and a laser field [Text] / M. V. Frolov, N.L. Manakov, A.F. Starace // Phys. Rev. A. — 2008. — Vol. 78. — 063418.

4 Accurate Retrieval of Structural Information from Laser-Induced Photoelectron and High-Order Harmonic Spectra by Few-Cycle Laser Pulses [Text] / T. Morishita, A.-T. Le, Z. Chen, C. D. Lin // Phys. Rev. Lett. — 2008. — Vol. 100. — 013903.

5 Analytic model for the description of above-threshold ionization by an intense short laser pulse [Text] / M.V. Frolov, D.V. Knyazeva, N.L. Manankov, Ji-Wei Geng // Phys Rev A. — 2014. — Vol. 89. — 063419.

6 М.А, Лаврентьев. Методы теории функций комплексного переменного [Текст] / Лаврентьев М.А, Шабат Б.В. — М. : Главная редакция физико-математической литературы изд-ва «Наука», 1973. — С. 736.

7 И.Г., Араманович. Уравнения математической физики [Текст] / Араманович И.Г., Левин В.И. — М. : Наука, 1969. — С. 288.

8 М.В., Федорюк. Метод перевала [Текст] / Федорюк М.В. — М. : Главная редакция физико-математической литературы изд-ва «Наука», 1977.

9 Wong, R. Asymptotic Approximations of Integrals [Text] / R. Wong. — NY. : SIAM, 2001. — P. 543.

10 М., Абрамовиц. Справочник по специальным функциям [Текст] / Абрамовиц М., Стиган И. — М. : Наука, 1979.

11 Ю.Ю., Тарасевич. Численные методы на Mathcad’е [Текст] / Тарасевич Ю.Ю. — Астрахань : Астраханский гос. пед. ун-т, 2000. — С. 70.

12 Galassi, Mark. GNU Reference Manual [Text]. — 2016. — [Сайт] URL: https://www.gnu.org/software/gsl/manual/html_node/ (дата обращения 03.01.17).

13 Galassi, Mark. GSL - GNU Scientific Library [Text]. — 2016. — [Сайт] URL: https://www.gnu.org/software/gsl/ (дата обращения 03.01.17).

14 Frigo, Matteo. FFTW - Home Page [Text]. — 2017. — [Сайт] URL: http://www.fftw.org/index.html (дата обращения 20.12.16).

15 Numerical Recipes in C [Text] / William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. — NY : Press Syndicate of the University of Cambridge, 2002. — P. 949.

Приложение А
Листинг программы

В данном приложении приведен текст программы, реализующей алгоритмы, описанные в данной работе.

```
#include <math.h>
#include <stdio.h>
#include <malloc.h>
#include <float.h>
#include <complex.h>
#include <fftw3.h>
#include <gsl/gsl_integration.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_roots.h>
#include <gsl/gsl_errno.h>

#define TBEGIN (-20.0)
#define TEND (20.0)
#define TSTEP (0.1)
#define EPS 0.5
#define H 1.054572e-27
#define PIM4 0.7511255444649425
#define PI 3.14159265359
#define OMEGA 1
#define F0 1
#define ALPHA 4.472135955
#define NPOINTS (1024.0)

double f(double x, void * params);
double A_evaluate(double t, void * params);
double A_integrate(double t1, double t);
double S_integrate_new(double tau, double t);
double S_integrate_func_new(double eps, void * params);
double Alpha2(double t1, double t);
double GetEvalSolveWithFFTW(double t);
double GetEvalSolveM_0(double t);
double GetEvalSolveWithoutFFTW(double t);
double GetAnalyticSolve(double t, int countOfRoots, double* roots);
int FindRoots(double t, double* roots);

int main()
{
  FILE *f;
  f = fopen("analytical.txt", "w");
  FILE *f1;
  f1 = fopen("numerical.txt", "w");
  FILE *f2;
```

```
43    f2 = fopen("roots.txt", "w");
44    double tBegin = TBEGIN, tEnd = TEND, tStep = TSTEP;
45    int steps = (int)(tEnd-tBegin)/tStep + 1;
46    double* m1 = (double*) malloc(sizeof(double) * steps);
47    double* m2 = (double*) malloc(sizeof(double) * steps);
48
49    for(int i = 0; i<steps; i++)
50    {
51      double t = tBegin + i*tStep;
52      printf("T = %f\n", t);
53      double roots[20];
54      int countOfRoots = FindRoots( tBegin + i*tStep , roots );
55
56      for(int j = 0; j<countOfRoots; j++)
57      {
58        fprintf(f2, "%3.10f %3.10f\n", t, roots[j]);
59      }
60
61      double m1 = GetAnalyticSolve( t , countOfRoots, roots );
62      double m2 = GetEvalSolveWithoutFFTW( t );
63      printf("M_analytic(eps, t) = %2.15f\n", m1);
64      printf("M_numerical(eps, t) = %2.15f\n", m2);
65      printf("M_0(eps, t) = %2.15f\n", m2);
66      fprintf(f, "%2.15f %2.15f\n", t , m1);
67      fprintf(f1, "%2.15f %2.15f\n", t, GetEvalSolveM_0( t ));
68    }
69
70    fclose(f);
71    fclose(f1);
72    fclose(f2);
73    return 0;
74  }
75
76
77  //Структура длячисленногорешенияуравнения
78  struct quadratic_params
79    {
80      double t;
81    };
82
83  struct S_integrate_params
84    {
85      double Aint;
86    };
87
88  //Структура длячисленногорешенияуравнения
89  double quadratic (double t0, void *params)
90  {
91    struct quadratic_params *p
```

```
92      = (struct quadratic_params *) params;
93
94    double t = p->t;
95
96    return Alpha2(t0, t)-sqrt(2*EPS);
97  }
98
99  double quadratic2 (double t0, void *params)
100 {
101   struct quadratic_params *p
102     = (struct quadratic_params *) params;
103
104   double t = p->t;
105
106   return Alpha2(t0, t)+sqrt(2*EPS);
107 }
108
109 int FindRoots(double t, double* roots)
110 {
111   gsl_function AlphaFunc;
112   const gsl_root_fsolver_type *T;
113   gsl_root_fsolver *s;
114   struct quadratic_params params = { t };
115   AlphaFunc.function = &quadratic;
116   AlphaFunc.params = &params;
117   T = gsl_root_fsolver_brent;
118   s = gsl_root_fsolver_alloc (T);
119   double h = 0.5;
120   double t1_hi = t, t1_lo = t - h;
121   int countOfRoots = 0;
122   while( t1_lo > -10 )
123   {
124     if ((quadratic(t1_lo, &params)) * (quadratic(t1_hi, &params)) < 0)
125     {
126       gsl_root_fsolver_set (s, &AlphaFunc, t1_lo, t1_hi);
127       int iter = 0;
128       int status = 0;
129       do
130       {
131         iter++;
132         status = gsl_root_fsolver_iterate (s);
133         roots[countOfRoots] = gsl_root_fsolver_root (s);
134         t1_lo = gsl_root_fsolver_x_lower (s);
135         t1_hi = gsl_root_fsolver_x_upper (s);
136         status = gsl_root_test_interval (t1_lo, t1_hi,
137                                          0, 0.001);
138       }
139       while (status == GSL_CONTINUE && iter < 1000);
140       countOfRoots++;
```

```
141         t1_hi = t1_lo;
142         t1_lo -= h;
143       }
144       t1_hi = t1_lo;
145       t1_lo -= h;
146     }
147
148
149     AlphaFunc.function = &quadratic2;
150     AlphaFunc.params = &params;
151     h = 0.5;
152     t1_hi = t, t1_lo = t - h;
153     while( t1_lo > -10 )
154     {
155       if ((quadratic2(t1_lo, &params)) * (quadratic2(t1_hi, &params)) < 0)
156       {
157         gsl_root_fsolver_set (s, &AlphaFunc, t1_lo, t1_hi);
158         int iter = 0;
159         int status = 0;
160         do
161         {
162           iter++;
163           status = gsl_root_fsolver_iterate (s);
164           roots[countOfRoots] = gsl_root_fsolver_root (s);
165           t1_lo = gsl_root_fsolver_x_lower (s);
166           t1_hi = gsl_root_fsolver_x_upper (s);
167           status = gsl_root_test_interval (t1_lo, t1_hi,
168                                            0, 0.001);
169         }
170         while (status == GSL_CONTINUE && iter < 1000);
171         countOfRoots++;
172         t1_hi = t1_lo;
173         t1_lo -= h;
174       }
175       t1_hi = t1_lo;
176       t1_lo -= h;
177     }
178     gsl_root_fsolver_free (s);
179     return countOfRoots;
180 }
181
182 double GetAnalyticSolve(double t, int countOfRoots, double* roots)
183 {
184   complex sum = 0;
185
186   int a = 0;
187
188   for(int i = 0; i<countOfRoots; i++)
189   {
```

```
190     double S_1 = S_integrate_new(roots[i], t) + EPS*(t-roots[i]);
191     double d = Alpha2(roots[i], t) * (( f(roots[i], 0) ) -
192       ( 1/( (roots[i] - t)*(roots[i] - t) ) *
193       A_integrate(roots[i], t) + A_evaluate(roots[i], 0) / (t - roots[i])));
194     sum += cexp(I * S_1)/(csqrt(d) * cpow( t - roots[i] , 3.0/2.0));
195   }
196   sum *= csqrt(2);
197   return creal(sum)*creal(sum) + cimag(sum)*cimag(sum);
198 }
199
200 double f (double x, void * params)
201 {
202   return -F0*exp(-(x*x)/(ALPHA*ALPHA))*cos(OMEGA*x);
203 }
204
205 //A какинтегралот    f
206 double A_evaluate(double t, void * params)
207 {
208   double t1 = -50;     // -\infty
209   double result;
210   double error;
211   double alpha = 1.0;
212   gsl_function F;
213   F.function = &f;
214   F.params = &alpha;
215   gsl_integration_workspace * w = gsl_integration_workspace_alloc (1000000);
216   gsl_integration_qags (&F, t1, t, 1e-7, 0, 1000000, w, &result, &error);
217   gsl_integration_workspace_free (w);
218   return result;
219 }
220
221 //A какинтегралот    A_eval
222 double A_integrate(double t1, double t)
223 {
224   double result;
225   double error;
226   double alpha = 1.0;
227   gsl_function F;
228   F.function = &A_evaluate;
229   F.params = &alpha;
230   gsl_integration_workspace * a = gsl_integration_workspace_alloc (10000);
231   gsl_integration_qags (&F, t1, t, 1e-7, 0, 10000, a, &result, &error);
232   gsl_integration_workspace_free (a);
233   return result;
234 }
235
236 //Под интеграломфункция  S_integrate_func
237 double S_integrate_new(double tau, double t)
238 {
```

```
239   if (tau ≥ t)
240   {
241     return 0;
242   }
243   double Aint = A_integrate(tau, t)/(t-tau);
244   struct S_integrate_params params = { Aint };
245   double result, error;
246   gsl_function F;
247   F.function = &S_integrate_func_new;
248   F.params = &params;
249   gsl_integration_workspace * b = gsl_integration_workspace_alloc (10000);
250   gsl_integration_qags (&F, tau, t, 1e-7, 0, 10000, b, &result, &error);
251   gsl_integration_workspace_free (b);
252   return -result/2.0;
253 }
254
255 double S_integrate_func_new(double eps, void * params)
256 {
257   struct S_integrate_params *p
258     = (struct S_integrate_params *) params;
259
260   double Aint = p->Aint;
261   double temp = A_evaluate(eps, 0) - Aint;
262   return temp*temp;
263 }
264
265 //Функция покоторойрешаетсяуравнение
266 double Alpha2(double t1, double t)
267 {
268   return A_evaluate(t1, 0) - A_integrate(t1, t) / (t - t1);
269 }
270
271 double S_integrate_func(double eps, void * params)
272 {
273   double temp = A_evaluate(eps, 0);
274   return temp*temp;
275 }
276
277 double GetEvalSolveWithoutFFTW(double t)
278 {
279   double tempEnd = 350;
280   double step = tempEnd / NPOINTS;
281   double tempPower;
282   int i = 0;
283   complex sum = 0;
284
285   for(int i = 1; i < (int)NPOINTS; i++)
286   {
287     sum += cexp(I * EPS * i*step)
```

```
288         / cpow(i*step, 3.0/2.0)
289         * (cexp(I * S_integrate_new(t-i*step, t)) - 1);
290     }
291
292     sum *= step;
293     sum *=1/csqrt(2*PI*I);
294     double result = creal(sum)*creal(sum) + cimag(sum)*cimag(sum);
295     return result;
296 }
297
298 double GetEvalSolveWithFFTW(double t)
299 {
300     fftw_complex *in, *out;
301       fftw_plan p;
302
303       in = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * NPOINTS);
304       out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * NPOINTS);
305
306     double tempEnd = 200;
307     double step = tempEnd / NPOINTS;
308     double tempPower;
309     int i = 0;
310
311     for(double tempT = 0; tempT < tempEnd; tempT+=step)
312     {
313       if(!tempT)
314       {
315         in[i] = 0;
316         i++;
317         continue;
318       }
319       in[i] = cexp(I * EPS * tempT)
320         / cpow(tempT, 3.0/2.0)
321         * (cexp(I * S_integrate_new(t-tempT, t)) - 1);
322       i++;
323
324     }
325
326     p = fftw_plan_dft_1d(NPOINTS, in, out, FFTW_FORWARD, FFTW_ESTIMATE);
327     fftw_execute(p);
328     out[0]*=1.0/NPOINTS;
329     out[0]*=tempEnd;
330     out[0]*=1/csqrt(2*PI*I);
331     double result = creal(out[0])*creal(out[0]) + cimag(out[0])*cimag(out[0]);
332     free(out);
333     fftw_destroy_plan(p);
334     return result;
335 }
336
```

```
337 double GetEvalSolveM_0(double t)
338 {
339   fftw_complex *in, *out;
340     fftw_plan p;
341 
342     in = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * NPOINTS);
343     out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * NPOINTS);
344 
345   double tempEnd = 90;
346   double step = tempEnd / NPOINTS;
347   double tempPower;
348 
349   int i = 0;
350   for(double tempT = 0; tempT < tempEnd; tempT+=step)
351   {
352     if(!tempT)
353     {
354       in[i] = 0;
355       i++;
356       continue;
357     }
358     double S_0=-f(t,0)*f(t,0)/24*tempT*tempT*tempT;
359     in[i] = cexp(I * EPS * tempT)
360       / cpow(tempT, 3.0/2.0)
361       * (cexp(I * S_0) - 1);
362     i++;
363   }
364 
365   p = fftw_plan_dft_1d(NPOINTS, in, out, FFTW_FORWARD, FFTW_ESTIMATE);
366 
367   fftw_execute(p);
368   out[0]*=1.0/NPOINTS;
369   out[0]*=tempEnd;
370   double result = creal(out[0])*creal(out[0]) + cimag(out[0])*cimag(out[0]);
371   free(out);
372   fftw_destroy_plan(p);
373   return result;
374 }
```