

Министерство науки и высшего образования Российской Федерации  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ  
«МИФИ»

---

ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ  
КАФЕДРА №31 ПРИКЛАДНАЯ МАТЕМАТИКА

На правах рукописи  
УДК 517.9

МЕДВЕДЕВ ВИКТОР АЛЕКСАНДРОВИЧ

ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ СОЛИТОННЫХ РЕШЕНИЙ  
НЕЛИНЕЙНОГО УРАВНЕНИЯ ШРЕДИНГЕРА С ТРЕМЯ НЕЛИНЕЙНОСТЯМИ

Выпускная квалификационная работа магистра

Направление подготовки 01.04.02 «Прикладная математика и информатика»

Выпускная квалификационная  
работа защищена  
«\_\_\_\_\_» 20\_\_ г.  
Оценка \_\_\_\_\_  
Секретарь ГЭК \_\_\_\_\_ Чмыхов М.А.

г.Москва 2024

## **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к выпускной квалификационной работе на тему:

### **ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ СОЛИТОННЫХ РЕШЕНИЙ НЕЛИНЕЙНОГО УРАВНЕНИЯ ШРЁДИНГЕРА С ТРЕМЯ НЕЛИНЕЙНОСТЯМИ**

Студент–дипломник \_\_\_\_\_ Медведев В.А.

Руководитель проекта \_\_\_\_\_ д.ф.-м.н., профессор Кудряшов Н.А.

Рецензент \_\_\_\_\_ к.ф.-м.н., доцент Чмыхов М.А.

Зав. кафедрой №31 \_\_\_\_\_ д.ф.-м.н., профессор Кудряшов Н.А.

# Реферат

Отчет 64 страницы, 1 часть., 26 рисунков, 0 таблиц, 41 источник, 1 приложение.

Ключевые слова: обобщённое нелинейное уравнение Шрёдингера, псевдоспектральный метод Фурье, оптический солитон, численное моделирование, нелинейная оптика, нелинейные уравнения в частных производных

Объектом исследования является уравнение в частных производных, описывающее распространение импульсов в оптоволокне.

Цель работы - построить оптические солитоны для исследуемого уравнения, численно исследовать процессы их распространения, сравнить численные результаты с аналитическими, исследовать процессы взаимодействия оптических солитонов с возмущениями в начальных данных и исследовать влияние старших степеней нелинейности на решение классического нелинейного уравнения Шрёдингера.

Метод проведения работы: последовательное применение приёмов поиска частных решений уравнений в частных производных, численное исследование аналитических решений при помощи конечно-разностного метода и псевдоспектрального метода Фурье, математическое моделирование.

Результаты работы: для полученных аналитически оптических солитонов исследуемого уравнения определена область допустимых параметров математической модели. Представлена модификация конечно-разностного метода и псевдоспектрального метода Фурье для численного исследования процессов распространения импульсов в рамках исследуемой модели. Дано сравнение аналитического решения с результатами численных расчётов. Исследован процесс распространения оптического солитона исследуемого уравнения при возмущении начальных данных и в условиях случайного шума. Обсуждён факт численной устойчивости найденных аналитически решений. Проанализировано влияние нелинейных членов пятой и седьмой степеней в уравнении модели на процесс распространения уединенных волн нелинейного уравнения Шрёдингера. Обнаружено явление преобразования оптического солитона нелинейного уравнения Шрёдингера в устойчивый солитон для возмущённой нелинейной модели. Показано, при каких параметрах модели переход возможен. Изучены процессы столкновения солитонов нелинейного уравнения Шрёдингера при влиянии нелинейных членов пятой и седьмой степеней. Показано, что столкновения носят неупругий характер.

Область применения: проектирование оптоволоконных линий.

Экономическая эффективность или значимость работы: позволяет при определённой выбранной математической модели, описывающей физический процесс распространения импульсов в оптоволокне, подобрать параметры для построения устойчивых импульсов, пригодных для передачи сигналов.

# Оглавление

<b>Введение</b>	<b>5</b>
1    Роль солитонов в передаче сигналов . . . . .	5
2    Цели и задачи работы . . . . .	7
<b>1 Аналитический анализ</b>	<b>9</b>
1    Аналитическое решение для уравнения Шрёдингера с тремя нелинейностями . . . . .	9
2    Модификации численных методов для моделирования процессов, описываемых уравнением Шрёдингера с тремя нелинейностями . . . . .	14
<b>2 Численный анализ</b>	<b>17</b>
1    Сравнение и валидация численных методов решения задачи распространения оптических импульсов . . . . .	17
2    Применение метода Фурье для моделирования процесса распространения уединённой волны, описываемой уравнением Шрёдингера с тремя нелинейностями . . . . .	18
3    Моделирование распространения солитона при возмущении начальных условий . . . . .	20
4    Анализ влияния высших степеней нелинейности на распространение уединённых волн нелинейного уравнения Шрёдингера . . . . .	21
5    Столкновения солитонов в присутствии высших степеней нелинейности .	35
<b>3 Заключение</b>	<b>38</b>
<b>Приложение А - Коды программ, использованных в работе</b>	<b>40</b>
1    Функции, определяющие аналитические решения . . . . .	40
1.1    Солитон классического НУШ . . . . .	40
1.2    Солитон НУШ с нелинейностью 3 и 5 степеней . . . . .	40
1.3    Солитон НУШ с нелинейностью 3, 5 и 7 степеней . . . . .	41
2    Функции для расчёта интегралов (законов сохранения) . . . . .	43
2.1    Мощность . . . . .	43
2.2    Импульс . . . . .	43

3	Функция численного решения (CPU) . . . . .	43
3.1	Процедура фильтрации . . . . .	43
3.2	Основная функция . . . . .	44
4	Функция численного решения (GPU) . . . . .	47
4.1	CUDA-версии вспомогательных функций . . . . .	47
4.2	Основная функция . . . . .	49
5	Функции для подбора аналитического решения по численному . . . . .	54
5.1	Вспомогательные функции . . . . .	54
5.2	Основные функции . . . . .	54

# Введение

## 1 Роль солитонов в передаче сигналов

За последние пару десятилетий область телекоммуникаций претерпела существенную эволюцию благодаря впечатляющему прогрессу в разработке оптических волокон, оптических усилителей, передатчиков и приемников. В современной оптической системе связи линия передачи состоит из оптических волокон и усилителей, составляющих волновод [1]. Основная идея волновода состоит в том, чтобы направлять луч света, используя изменение показателя преломления в поперечном направлении, чтобы заставить свет проходить по четко определенному каналу. Зависимость показателя преломления от поперечного направления, направления, перпендикулярного тому, в котором распространяется волна, может быть непрерывной или дискретной. Но в любом из случаев показатель преломления максимальен в канале, по которому требуется направлять свет.

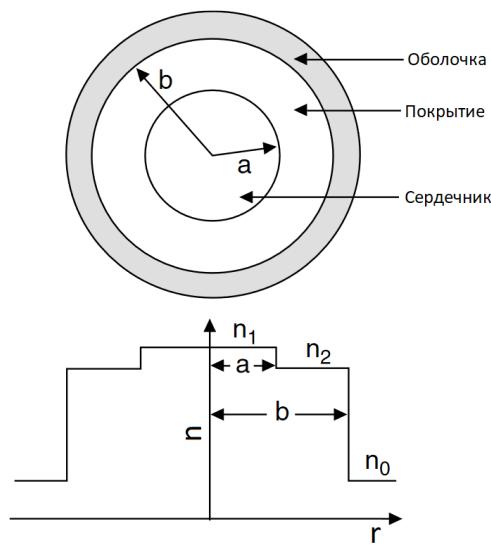


Рис. 1: Зависимость показателя преломления от радиуса поперёк оптического кабеля[2]

В 1973 году, Акира Хасегава и Фредерик Тапперт сформулировали условия существования световых оптических солитонов в одномодовых оптических волокнах и предположили возможность их использования при передаче сигналов.[3, 4]. В 1980 году

возможность использования световых солитонов в одномодовых оптических волокнах была экспериментально подтверждена Молленгаузом, Столеном и Гордоном [5, 6], что дало мощный толчок развитию нелинейной оптики [7, 8, 9, 10].

Хорошо известно, что нелинейное уравнение Шрёдингера (НУШ), которое включает дисперсию групповой скорости и самомодуляцию фазы, является основным нелинейным уравнением, используемым для описания динамики импульсов. Классическое НУШ имеет следующий вид:

$$iu_t + au_{xx} + |u|^2 u = 0, \quad (1)$$

где  $u(x, t)$  - комплексная функция,  $i^2 = -1$  и  $a$  - параметры модели. Это уравнение является параболическим, что обуславливает особенности применения численных и аналитических методов для его исследования. Солитоны НУШ имеют форму огибающей группы волн, что позволяет использовать их для описания групповых свойств волнового пакета.

Для увеличения пропускной способности оптических волокон современные системы передачи используют всё более короткие импульсы. Но при использовании коротковолновых импульсов возрастает влияние на процесс некоторых физических эффектов, таких как дисперсия высокого порядка [11, 12, 13, 14, 15], самофокусировка [16], и вынужденное комбинационное рассеяние [17, 18]. Для корректного описания поведения оптических импульсов в присутствии данных эффектов, классическое НУШ обобщается - в уравнение вводятся производные высших порядков, уточняющие процессы дисперсии, и нелинейные члены высших порядков, описывающие процессы дисперсии [19, 20, 21, 22, 23]. Как следствие, различные обобщения НУШ обширно исследуются математиками по всему миру. В качестве примера стоит упомянуть работы [24, 25, 26, 27].

Учет всех перечисленных физических факторов существенно усложняет аналитическое исследование процессов, описываемых предлагаемыми математическими моделями. Поэтому, с развитием вычислительных возможностей, для исследований все чаще используются численные методы.

К примеру, НУШ, как классическое уравнение нелинейной оптики, было численно исследовано в работах [28, 29]. Эти работы подтвердили неустойчивость постоянных решений и показали, что неустойчивость не растет неограниченно, как ожидалось в линейной теории. Также в этих работах выявлен тот факт, что решение периодически возвращается к исходным условиям. Это явление, известное как повторяемость, наблюдалось также в работах Ферми, Пасты и Улама. [30]. В дополнение к нелинейной оптике, НУШ находит применение в физике плазмы и теории невязкой жидкости.

Среди первых схем, использованных для численного анализа НУШ, упомянем конечно-разностный и псевдоспектральные методы [28, 31]. Среди более продвинутых работ, стоит упомянуть работы [32, 33, 34, 35, 36, 37, 38], где дискретизация простран-

ства осуществляется в основном с помощью методов конечных разностей или конечных элементов, а дискретизация времени осуществляется с использованием явных методов с переменным шагом [32, 33], метода расщепления [34], модифицированной схемы Кранка-Николсона [35] и прочих способов. Из-за хорошо изученных свойств, НУШ подходит для валидации новых разрабатываемых численных методов.

## 2 Цели и задачи работы

Несмотря на разнообразие предложенных математических моделей, вопрос о том, какая из них наиболее подходит для описания процессов распространения импульсов, остается открытым. В представленной квалификационной работе исследовано одно из обобщённых уравнений - нелинейное уравнение Шрёдингера с нелинейностями третьего, пятого и седьмого порядков, впервые представленное в работе [39]:

$$iu_t + au_{xx} + b_1|u|^2u + b_2|u|^4u + b_3|u|^6u = 0, \quad (2)$$

где  $u(x, t)$  - комплекснозначная функция,  $a, b_1, b_2$  и  $b_3$  - параметры модели. Для случая  $b_3 = 0$  исследование уравнения (2) представлено в книге [6].

Целями настоящей работы являются:

1. Аналитическое нахождение точного решения представленного уравнения;
2. Исследование факта устойчивости аналитического решения с помощью численного моделирования процессов распространения оптических импульсов;
3. Изучение влияния нелинейных членов высших степеней в исследуемой математической модели.

В работе решаются следующие задачи:

1. Построение аналитического решения исходного уравнения в виде уединённой волны с использованием метода прямых вычислений;
2. Определение области параметров модели, при которых построенное решение существует;
3. Модификация методов расщепления для численного решения задачи распространения оптических импульсов, описываемых математической моделью;
4. С целью установления факта устойчивости построенных решений - численное моделирование процессов:
  - (а) Распространения построенной уединённой волны;

- (b) Распространения построенных солитонов при наличии возмущений в начальных условиях;
  - (c) Распространения построенных солитонов в условиях случайного шума;
5. Определение влияния возмущающих нелинейных членов в нелинейном уравнении Шрёдингера.

Результаты работы представлены в следующем порядке. В разделе 1 главы 1 построено аналитическое решение для уравнения (2) в форме уединённой волны с использованием метода неявных функций и метода простейших уравнений [40]. Проиллюстрирована область значений параметров модели, при которых построенное решение существует. В разделе 2 главы 1 описаны модифицированные конечно-разностный и Фурье методы для моделирования процессов распространения оптических импульсов, описываемых уравнением (2). В разделе 1 главы 2 представлена валидация и сравнение реализованных численных схем решения задачи распространения оптических импульсов. В разделе 2 главы 2 представлено применение метода Фурье для моделирования процесса распространения уединенной волны, описываемого уравнением (2). В разделе 3 главы 2 смоделировано взаимодействие солитонного импульса с возмущением в начальном условии. В разделе также исследован процесс распространения импульса в условиях случайного шума в начальном условии и обсуждён факт устойчивости построенного оптического солитона. В разделе 4 главы 2 исследовано влияние более высоких степеней нелинейности на процесс распространения уединенной волны описываемой НУШ. В разделе 5 главы 2 смоделированы процессы столкновения солитонов уравнения НУШ при наличии членов с более высокой нелинейностью при различных параметрах модели.

Метод исследования - аналитический анализ, численный анализ.

# Глава 1

## Аналитический анализ

### 1 Аналитическое решение для уравнения Шрёдингера с тремя нелинейностями

С целью упростить уравнение (2), используем переход к безразмерным величинам в следующем виде:

$$\begin{cases} u(x, t) = c_u u'(x, t), \\ t = c_t t', \\ x = c_x x', \end{cases} \quad (1.1)$$

При этом уравнение (2) запишется следующим образом:

$$i \frac{c_u}{c_t} u'_{t'} + \frac{a c_u}{c_x^2} u'_{x' x'} + b_1 c_u^3 |u'|^2 u' \left( 1 + c_u^2 \frac{b_2}{b_1} |u'|^2 + c_u^4 \frac{b_3}{b_1} |u'|^4 \right) = 0. \quad (1.2)$$

Принимая

$$\begin{cases} c_u = b_1^{-1/3}, \\ c_t = b_1^{-1/3}, \\ c_x = \sqrt{a} b_1^{-1/6}, \end{cases} \quad (1.3)$$

Для уравнения (1.2) получим:

$$i u'_{t'} + u'_{x' x'} + |u'|^2 u' \left( 1 + \varepsilon_2 |u'|^2 + \varepsilon_3 |u'|^4 \right) = 0, \quad (1.4)$$

где

$$\begin{cases} \varepsilon_2 = b_1^{-4/3} b_2, \\ \varepsilon_3 = b_1^{-7/3} b_3. \end{cases} \quad (1.5)$$

Будем искать решения уравнения (1.4) в виде:

$$u'(x', t') = y(z) e^{i(kx' - \omega t' - \theta_0)}, \quad z = x' - c_0 t', \quad k, \omega, c_0, \theta_0 \in \mathbb{R}, \quad (1.6)$$

где  $y(z)$  - действительнозначная функция. Подставляя (1.6) в уравнение (1.4), получим переопределённую систему уравнений для  $y(z)$  в виде:

$$y_{zz} + \varepsilon_3 y^7 + \varepsilon_2 y^5 + y^3 + (\omega - k^2) y = 0, \quad (1.7)$$

$$(2k - c_0)y_z = 0. \quad (1.8)$$

уравнение (1.8) выполняется тождественно при  $c_0 = 2k$ . Уравнение (1.7) имеет первый интеграл:

$$y_z^2 + \frac{\varepsilon_3 y^8}{4} + \frac{\varepsilon_2 y^6}{3} + \frac{y^4}{2} + (\omega - k^2) y^2 = c_1. \quad (1.9)$$

Переходя к новой переменной  $y(z) = \sqrt{V(z)}$ , перепишем уравнение (1.9):

$$\frac{1}{4} V_z^2 + \frac{\varepsilon_3}{4} V^5 + \frac{\varepsilon_2}{3} V^4 + \frac{1}{2} V^3 + (\omega - k^2) V^2 - c_1 V = 0. \quad (1.10)$$

Используя метод неявных функций, будем искать решения уравнения (1.10) в виде  $V(z) = F(\xi)$ ,  $\xi = \psi(z)$ , полагая  $c_1 = 0$  и

$$\xi_z = \pm F(\xi), \quad (1.11)$$

что приводит к следующему уравнению:

$$F_\xi^2 + \varepsilon_3 F^3 + \frac{4}{3} \varepsilon_2 F^2 + 2F + 4(\omega - k^2) = 0. \quad (1.12)$$

Уравнение (1.12) может быть записано в виде:

$$\begin{aligned} \left[ \frac{d}{d\xi} \left( F + \frac{4\varepsilon_2}{9\varepsilon_3} \right) \right]^2 + \varepsilon_3 \left( F + \frac{4\varepsilon_2}{9\varepsilon_3} \right)^3 + \frac{2(27\varepsilon_3 - 8\varepsilon_2^2)}{27\varepsilon_3} \left( F + \frac{4\varepsilon_2}{9\varepsilon_3} \right) + \\ + \frac{128\varepsilon_2^3}{729\varepsilon_3^2} - \frac{8\varepsilon_2}{9\varepsilon_3} - 4k^2 + 4\omega = 0. \end{aligned} \quad (1.13)$$

Вводя следующие обозначения для постоянных величин:

$$\begin{aligned} g_2 &= \frac{64\varepsilon_2^2}{27\varepsilon_3^2} - \frac{8}{\varepsilon_3}, \\ g_3 &= \frac{512\varepsilon_2^3}{729\varepsilon_3^3} - \frac{32\varepsilon_2}{9\varepsilon_3^2} - \frac{16k^2}{\varepsilon_3} + \frac{16\omega}{\varepsilon_3}, \\ \psi &= -F - \frac{4\varepsilon_2}{9\varepsilon_3}, \end{aligned} \quad (1.14)$$

перепишем уравнение (1.13) в виде:

$$\left( \left( 2\varepsilon_3^{-1/2} \right) \psi_\xi \right)^2 = 4\psi^3 - g_2\psi - g_3. \quad (1.15)$$

Общее решение уравнения (1.15) может быть выражено через эллиптическую функцию Вейерштрасса, что позволяет записать:

$$F(\xi) = -\wp \left( \left[ \frac{1}{2} \sqrt{\varepsilon_3} (\xi - \xi_0) \right]; g_2; g_3 \right) - \frac{4\varepsilon_2}{9\varepsilon_3}. \quad (1.16)$$

Учитывая условие (1.11), возможно выразить  $\xi(z)$  в квадратурах:

$$z - z_0 = \pm \int \frac{d\xi}{F(\xi)} = \mp \int \frac{d\xi}{\wp \left( \left[ \frac{1}{2} \sqrt{\varepsilon_3} (\xi - \xi_0) \right]; g_2; g_3 \right) + \frac{4\varepsilon_2}{9\varepsilon_3}}, \quad (1.17)$$

однако, в общем случае, такой интеграл не может быть посчитан аналитически.

Стоит заметить, что для специального вида функции  $F(\xi)$  интеграл (1.17) может быть посчитан. Используя метод простейших уравнений [40], найдём решения (1.12) в виде:

$$F(\xi) = M_0 + M_1 Q(\xi) + M_2 Q^2(\xi), \quad (1.18)$$

где  $Q(\xi)$  - решение уравнения Риккати:

$$Q_\xi = \mu (Q^2 - Q), \quad (1.19)$$

имеющее следующий вид:

$$Q(\xi) = \frac{1}{1 + \exp(\mu(\xi - \xi_0))}. \quad (1.20)$$

Используя (1.19), и подставляя выражение (1.18) в уравнение (1.12), получим полином относительно  $Q(\xi)$ , равный нулю:

$$\begin{aligned} & (\varepsilon_3 M_2^3 + 4\mu^2 M_2^2) Q(\xi)^6 + (4\mu^2 M_1 M_2 - 8\mu^2 M_2^2 + 3\varepsilon_3 M_1 M_2^2) Q(\xi)^5 + \\ & + \left( \frac{4}{3} \varepsilon_2 M_2^2 - 8\mu^2 M_1 M_2 + \mu^2 M_1^2 + 4\mu^2 M_2^2 + 3\varepsilon_3 M_0 M_2^2 + 3\varepsilon_3 M_1^2 M_2 \right) Q(\xi)^4 + \\ & + \left( 4\mu^2 M_1 M_2 + \varepsilon_3 M_1^3 - 2\mu^2 M_1^2 + \frac{8}{3} \varepsilon_2 M_1 M_2 + 6\varepsilon_3 M_0 M_1 M_2 \right) Q(\xi)^3 + \\ & + \left( 2M_2 + \frac{4\varepsilon_2 M_1^2}{3} + \mu^2 M_1^2 + 3\varepsilon_3 M_0^2 M_2 + 3\varepsilon_3 M_0 M_1^2 + \frac{8}{3} \varepsilon_2 M_0 M_2 \right) Q(\xi)^2 + \\ & + \left( 2M_1 + 3\varepsilon_3 M_0^2 M_1 + \frac{8}{3} \varepsilon_2 M_0 M_1 \right) Q(\xi) + \\ & + \left( \frac{4}{3} \varepsilon_2 M_0^2 + 2M_0 + \varepsilon_3 M_0^3 - 4k^2 + 4\omega \right) = 0. \end{aligned} \quad (1.21)$$

Так как  $Q(\xi) \neq 0$ , коэффициенты полинома должны быть тождественно равны

нулю. Это приводит к следующим ограничениям на параметры модели:

$$\begin{cases} \omega - k^2 = -\frac{1}{12} \frac{M_0 M_1}{M_1 + 6M_0} - \frac{1}{6} M_0, \\ \mu = \pm \sqrt{\frac{M_1}{M_0(M_1 + 6M_0)}}, \\ M_2 = -M_1, \\ \varepsilon_2 = \frac{3}{4M_0} \left( \frac{M_1}{M_1 + 6M_0} - 2 \right), \\ \varepsilon_3 = \frac{4}{M_0(M_1 + 6M_0)}, \end{cases} \quad (1.22)$$

где  $M_0$  и  $M_1$  - произвольные константы. Уравнение (1.18) теперь записывается, как:

$$F(\xi) = M_0 + \frac{M_1}{1 + e^{\mu(\xi - \xi_0)}} - \frac{M_1}{(1 + e^{\mu(\xi - \xi_0)})^2}. \quad (1.23)$$

Для представленного вида  $F(\xi)$  следующее выражение может быть проинтегрировано:

$$\frac{d\xi}{F(\xi)} = dz, \quad (1.24)$$

тогда зависимость  $\xi$  и  $z$  описывается следующим образом:

$$z = z_0 + \frac{\xi}{M_0} + \frac{2M_1}{\mu M_0 \sqrt{4M_0 M_1 + M_1^2}} \operatorname{arctanh} \left( \frac{2e^{\mu(\xi - \xi_0)} M_0 + 2M_0 + M_1}{\sqrt{4M_0 M_1 + M_1^2}} \right). \quad (1.25)$$

Теперь решение уравнения (1.9) при  $c_1 = 0$  запишется в виде:

$$y(\xi) = \left[ M_0 + \frac{M_1}{1 + e^{\mu(\xi - \xi_0)}} - \frac{M_1}{(1 + e^{\mu(\xi - \xi_0)})^2} \right]^{\frac{1}{2}}, \quad (1.26)$$

где  $\xi(z)$  определяется неявно из (1.25), и выполнены ограничения на параметры модели (1.22).

Таким образом, решение уравнения (1.4) имеет вид:

$$u'(x', t') = y(z) e^{i(kx' - \omega t' - \theta_0)}, \quad z = x' - 2kt', \quad (1.27)$$

где  $k$  и  $\theta_0$  - произвольные постоянные.

Принимая во внимание, что  $z, \xi(z), y(\xi) \in \mathbb{R}$ , из выражений (1.25) и (1.26) следуют дополнительные ограничения на параметры  $M_0$  и  $M_1$  для существования найденного

решения:

$$\left| \frac{2e^{\mu(\xi-\xi_0)}M_0 + 2M_0 + M_1}{\sqrt{4M_0M_1 + M_1^2}} \right| < 1, \quad (1.28)$$

$$M_0 + \frac{M_1}{1 + e^{\mu(\xi-\xi_0)}} - \frac{M_1}{(1 + e^{\mu(\xi-\xi_0)})^2} \geq 0. \quad (1.29)$$

Данные ограничения удовлетворяются на ограниченном промежутке по переменной  $\xi$  из-за присутствия экспоненты в (1.28), что накладывает дополнительные ограничения при построении решения.

Условия (1.28) и (1.29) с учётом соотношений (1.22) удовлетворяются в следующей области параметров  $M_1$  и  $M_0$  (Рис. 1.1):

$$\begin{cases} M_0 < 0, \\ -4M_0 < M_1 < -6M_0. \end{cases} \quad (1.30)$$

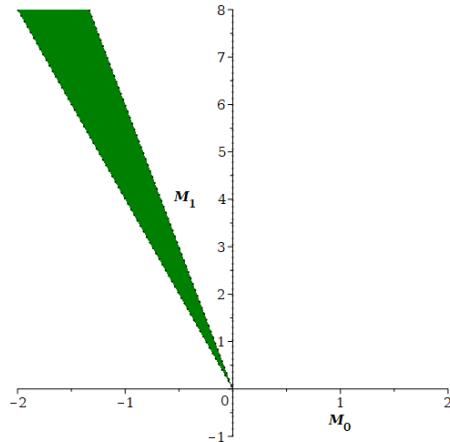


Рис. 1.1: Допустимые значения  $M_1$  и  $M_0$ .

Волновой профиль (1.27) при  $k = 1.6$ ,  $M_0 = -1.48$ ,  $M_1 = 6.16$ ,  $\varepsilon_2 = 2.16$ ,  $\varepsilon_3 = 0.99$  изображён на Рис. 1.2.

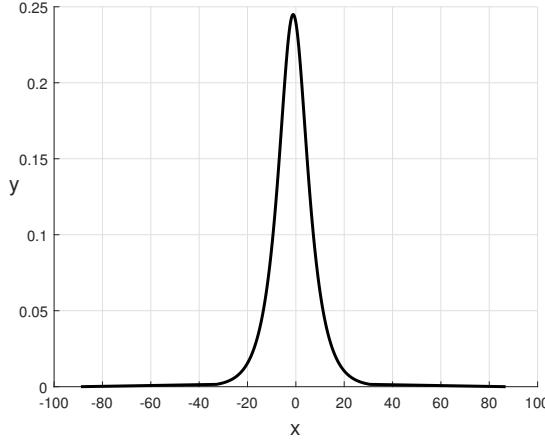


Рис. 1.2: Профиль уединённой волны при  $k = 1.6$ ,  $M_0 = -1.48$ ,  $M_1 = 6.16$ .

## 2 Модификации численных методов для моделирования процессов, описываемых уравнением Шрёдингера с тремя нелинейностями

Для численного решения задачи распространения оптических импульсов в рамках изучаемой модели требуется реализовать соответствующую численную схему. Рассмотрим, как это возможно сделать на примере метода расщепления. Семейство обобщённых уравнений Шрёдингера может быть записано в виде:

$$u_t = i\mathcal{L}[u] + i\mathcal{N}[u]u. \quad (1.31)$$

Операторы  $\mathcal{L}$  и  $\mathcal{N}$  соответствуют линейной и нелинейной частям уравнения. К примеру, при  $\mathcal{L}[u] \equiv au_{xx}$ ,  $\mathcal{N}[u] \equiv b_1|u|^2$  уравнение (1.31) представляет из себя нелинейное уравнение Шрёдингера (1).

Рассмотрим задачу распространения уединённых импульсов и объявим периодические граничные условия следующим образом:

$$\begin{cases} u\left(-\frac{L}{2}, t\right) = u\left(\frac{L}{2}, t\right), \\ \frac{\partial u}{\partial x}\left(-\frac{L}{2}, t\right) = \frac{\partial u}{\partial x}\left(\frac{L}{2}, t\right). \end{cases} \quad (1.32)$$

Предполагая  $x \in [-\frac{1}{2}L, \frac{1}{2}L]$ ,  $t \in [0, T]$ , разделим интервал по переменной  $x$  на  $N$  одинаковых частей с шагом

$$h = \frac{L}{N}. \quad (1.33)$$

Узлы координатной сетки определяются, как

$$x_j = jh, \quad j = -\frac{N}{2}, \dots, \frac{N}{2}. \quad (1.34)$$

Пусть  $\mathbf{U}^m$  - сеточная аппроксимация решения на  $m$ -ном временном слое, и  $\mathbf{V}^m$  - промежуточное решение. В таком случае, начальные условия задаются в  $\mathbf{U}^0$ . В общем виде, схема расщепления может быть записана следующим образом [1]:

$$\mathbf{U}^{m+1} = e^{i\tau\mathcal{L}} \mathbf{V}^m, \quad (1.35)$$

где

$$\mathbf{V}^m = e^{i\tau\mathcal{N}[\mathbf{U}^m]} \mathbf{U}^m. \quad (1.36)$$

В некоторых случаях, явный вид  $\mathcal{N}[u]$ , позволяет напрямую вычислить  $\mathbf{V}^m$ . Например, в случае уравнения (2) выражение (1.36) запишется следующим образом:

$$\mathbf{V}^m = e^{i\tau|\mathbf{U}^m|^2(1+\varepsilon_2|\mathbf{U}^m|^2+\varepsilon_3|\mathbf{U}^m|^4)} \mathbf{U}^m \quad (1.37)$$

В зависимости от метода аппроксимации оператора  $e^{i\tau\mathcal{L}}[u]$  возможно получить различные численные схемы для решения задачи распространения оптических импульсов, основанные на методе расщепления.

### Конечно-разностный метод

Для получения модификации конечно-разностного метода решения задачи распространения оптических импульсов, используем соотношение:

$$e^{i\tau\mathcal{L}} = (\mathcal{I} - \theta i\tau\mathcal{L})^{-1} (\mathcal{I} + (1 - \theta)i\tau\mathcal{L}), \quad (1.38)$$

где  $\mathcal{I}$  - тождественный оператор,  $\theta$  - параметр в пределах  $0 \leq \theta \leq 1$ . При этом выражение (1.35) запишется в виде:

$$(\mathcal{I} - \theta i\tau\mathcal{L}) \mathbf{U}^{m+1} = (\mathcal{I} + (1 - \theta)i\tau\mathcal{L}) \mathbf{V}^m. \quad (1.39)$$

Введём разностный оператор:

$$\mathcal{L}_h [\mathbf{U}_j] = (\mathbf{U}_{j+1} - 2\mathbf{U}_j + \mathbf{U}_{j-1}) / h^2, \quad j = -\frac{N}{2}, \dots, \frac{N}{2} - 1. \quad (1.40)$$

Тогда соотношение (1.39) запишется в матричной форме:

$$(I - ir\theta S) \mathbf{U}^{m+1} = (I + ir(1 - \theta S)) \mathbf{V}^m, \quad (1.41)$$

где  $I$  - тождественная матрица,  $r = \tau/h^2$ ,

$$\mathbf{U} = (U_{-N/2}, \dots, U_{N/2-1})^T, \quad (1.42)$$

$$S = \begin{pmatrix} -2 & 1 & . & . & 1 \\ 1 & -2 & 1 & . & . \\ . & . & . & . & . \\ . & . & 1 & -2 & 1 \\ 1 & . & . & -2 & 1 \end{pmatrix}. \quad (1.43)$$

Соотношения (1.37) и (1.41) полностью определяют конечно-разностный метод решения задачи распространения оптических импульсов в рамках модели (2).

### Псевдоспектральный метод Фурье

Альтернативный подход, приводящий к псевдоспектральному методу Фурье - использование дискретного преобразования Фурье для сеточной функции  $\mathbf{V}^m$  для определения решения на следующем временном слое:

$$\hat{\mathbf{V}}^m = \frac{h}{L} \exp(-i\boldsymbol{\mu}\mathbf{x}^T) \cdot \mathbf{V}^m, \quad (1.44)$$

где  $\hat{\mathbf{V}}^m$  - вектор коэффициентов Фурье,  $\boldsymbol{\mu} = (\mu_{-N/2}, \dots, \mu_{N/2-1})^T$  - вектор частот преобразования  $\mu_n = \frac{2\pi n}{L}$ ,  $\mathbf{x} = (x_{-N/2}, \dots, x_{N/2-1})^T$  - координаты точек сетки.

Воспользуемся соотношением между  $\hat{\mathbf{U}}^{m+1}$  и  $\hat{\mathbf{V}}^m$ ,

$$\hat{\mathbf{U}}^{m+1} = \exp(-i(\boldsymbol{\mu} \circ \boldsymbol{\mu})\tau) \circ \hat{\mathbf{V}}^m, \quad (1.45)$$

которое вытекает из уравнения (1.35) после подстановки в него вместо  $\mathbf{U}^{m+1}$  и  $\mathbf{V}^m$  соответствующих рядов Фурье. Под  $x \circ y$  понимается произведение по Адамару.

Решение на следующем временном слое восстанавливается с помощью обратного преобразования Фурье, используя (1.45):

$$\mathbf{U}^{m+1} = \exp(i\boldsymbol{\mu}\mathbf{x}^T) \cdot \hat{\mathbf{U}}^{m+1}. \quad (1.46)$$

Таким образом, соотношения (1.37) и (1.44-1.46) полностью определяют псевдоспектральный метод Фурье решения задачи распространения оптических импульсов в рамках модели (2).

# Глава 2

## Численный анализ

### 1 Сравнение и валидация численных методов решения задачи распространения оптических импульсов

Для проверки описанных в разделе 2 главы 1 численных схем проведём их валидацию и сравнение на примере моделирования процессов распространения уединённых волн, описываемых классическим НУШ. Рассмотрим (1.4) при  $\varepsilon_2 = 0$ ,  $\varepsilon_3 = 0$ :

$$iu_t + u_{xx} + |u|^2 u = 0. \quad (2.1)$$

Задача Коши для уравнения (2.1) решается методом обратной задачи рассеяния. Его решение в виде уединенной волны представлено в работе [39] и имеет вид:

$$u(x, t) = \frac{4(k^2 - \omega)}{2(k^2 - \omega)e^{-(x - c_0 t - z_0)\sqrt{(k^2 - \omega)}} + e^{(x - c_0 t - z_0)\sqrt{(k^2 - \omega)}}} \cdot e^{i(kx - \omega t - \theta_0)}, \quad (2.2)$$

где  $c_0 = 2k$  и  $k, \omega, z_0, \theta_0$  - произвольные константы. Начальное условие, относящееся к решению (2.2) представляется следующим образом:

$$u(x, 0) = \frac{4(k^2 - \omega)}{2(k^2 - \omega)e^{-(x - z_0)\sqrt{(k^2 - \omega)}} + e^{(x - z_0)\sqrt{(k^2 - \omega)}}} \cdot e^{i(kx - \theta_0)} \quad (2.3)$$

В качестве метрики на пространстве численных и соответствующих им аналитических решений на  $n$  временном слое будем использовать следующую величину:

$$\Delta\% = \frac{\max_{1 \leq m \leq N_x} \left( \left| U_{true}^{m,n} \right| - \left| U^{m,n} \right| \right)}{\max_{1 \leq m \leq N_x} \left| U_{true}^{m,n} \right|} \cdot 100\%, \quad (2.4)$$

где  $N_x = N + 1$  - количество x-узлов сетки на каждом временном слое согласно (1.34),  $U^{m,n}$  - численное решение,  $U_{true}^{m,n}$  - сеточная редукция соответствующего аналитического

решения. В качестве метрики для всего расчёта будем принимать максимальную из ошибок на временных слоях:

$$\Delta_{\%} = \max_{1 \leq n \leq N_t} \Delta_{\%}^n. \quad (2.5)$$

Величины  $\Delta_{\%}^n$  и  $\Delta_{\%}$  представляют из себя максимальное процентное отклонение по отношению к характерной амплитуде импульса для конкретного временного слоя и для всего решения, соответственно. Результаты моделирования процесса распространения уединённой волны (2.2) при помощи двух описанных в разделе 2 главы 1 методов представлены на Рис. 2.1.

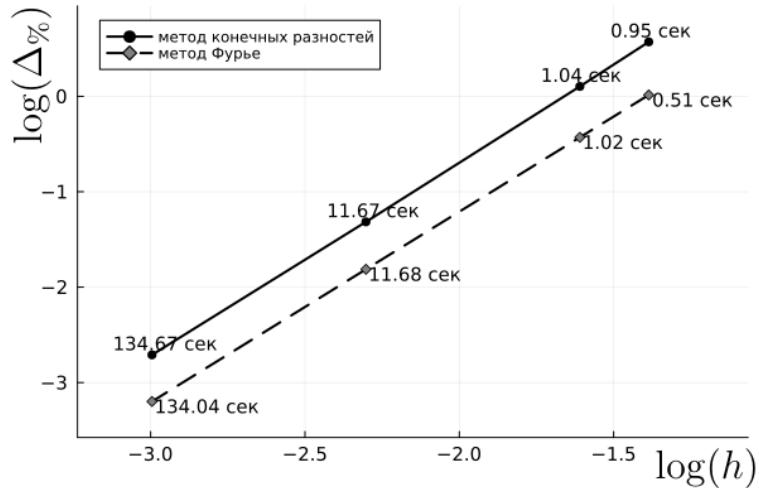


Рис. 2.1: Зависимости логарифма относительной ошибки от логарифма шага  $h$  по координате  $x$  при моделировании процесса распространения волны (2.2) с использованием конечно-разностного и Фурье методов с затраченным на расчёт временем при параметрах  $k = 0.15$ ,  $\omega = 0.4$ ,  $\theta_0 = 0$ ,  $z_0 = 0$ ,  $L = 160$ ,  $T = 25$ ,  $h = [0.25, 0.2, 0.1, 0.05]$ ,  $\tau = h^2$ .

Решения, полученные двумя схемами согласуются с аналитическим решением (2.2). Сеточная сходимость достигнута. Обе схемы имеют второй порядок аппроксимации по координате и первый порядок аппроксимации по времени. С точки зрения вычислительной сложности и эффективности обе схемы показали схожие результаты. Однако, в силу лучшей точности, в дальнейшей работе будут приводиться результаты, полученные псевдоспектральной схемой Фурье.

## 2 Применение метода Фурье для моделирования процесса распространения уединённой волны, описываемой уравнением Шрёдингера с тремя нелинейностями

Рассмотрим обобщённое уравнение (1.4). Смоделируем распространение уединённой волны (1.27), полученной в разделе 1 главы 1. Произведём расчёт для определён-

ных параметров  $M_0$  и  $M_1$ , удовлетворяющих условиям (1.30). Параметры  $\varepsilon_2, \varepsilon_3, \omega$  и  $\mu$  определяются формулами (1.22). Результат моделирования представлен на Рис.2.2. Аналитический и численно полученный профили в момент  $t = 16$  изображены на Рис. 2.2а.

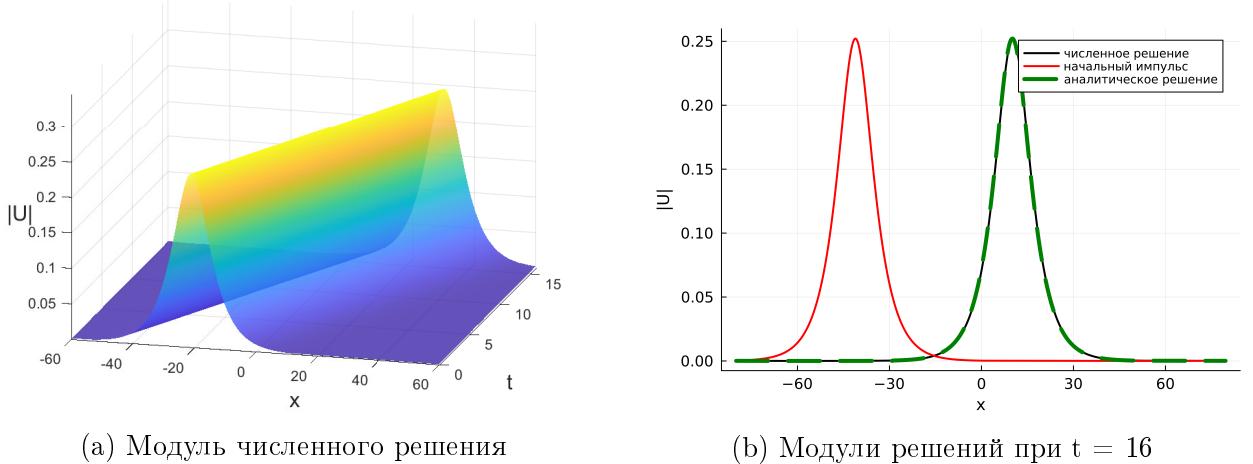


Рис. 2.2: Распространение уединённой волны (1.27) при параметрах  $L = 120$ ,  $T = 16$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $z_0 = -20$ ,  $k = 1.6$ ,  $M_0 = -1.48$ ,  $M_1 = 6.16$ ,  $\varepsilon_2 = 2.16$ ,  $\varepsilon_3 = 0.99$ .

Относительная погрешность расчета  $\Delta\%$  при заданных параметрах не превышает 0.1%. Зависимость относительной погрешности  $\Delta_{\%}^n$  от времени  $t = \tau n$  проиллюстрирована на Рис. 2.3а.

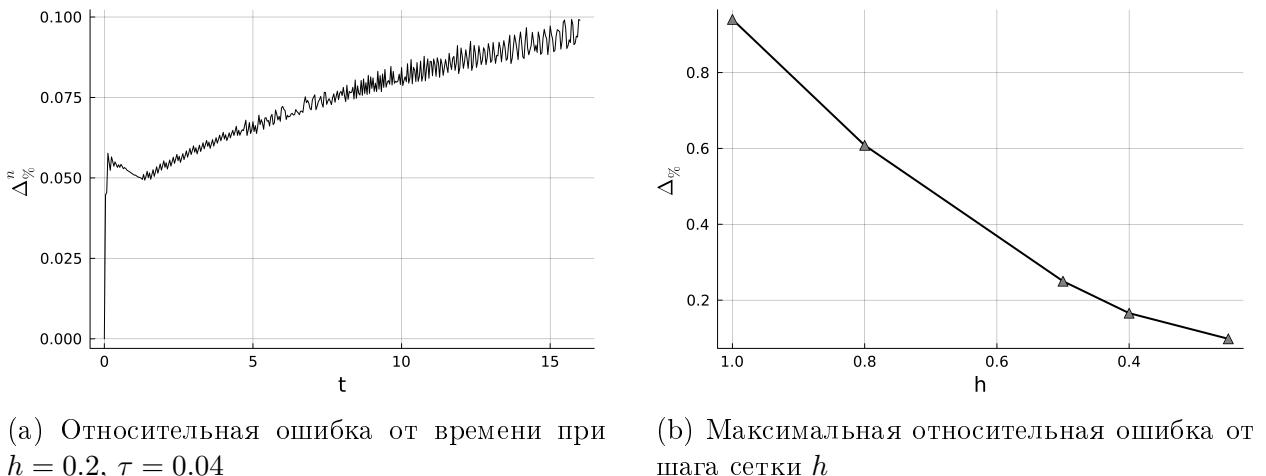


Рис. 2.3: Численные результаты при параметрах  $L = 160$ ,  $T = 16$ ,  $M_0 = -1.48$ ,  $M_1 = 6.16$ ,  $\varepsilon_2 = 2.16$ ,  $\varepsilon_3 = 0.99$ .

Сеточная сходимость достигнута (Рис. 2.3б), аналитическое решение совпало с численным. Следовательно, численная схема корректно описывает процесс распространения оптического импульса для обобщённого уравнения, и есть основание утверждать,

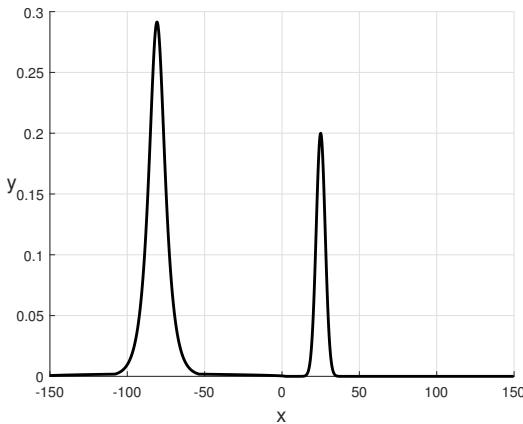
что аналитическое решение, построенное в разделе 1 главы 1, обладает солитонными свойствами.

### 3 Моделирование распространения солитона при возмущении начальных условий

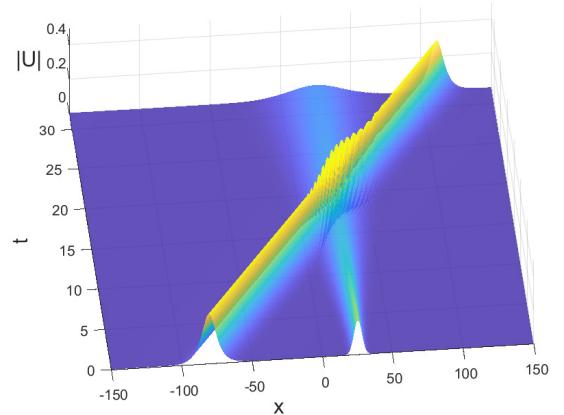
В рамках исследования устойчивости аналитического решения (1.27) проведём моделирование распространения импульса при возмущении в начальных условиях. Внёсём в начальное условие возмущение следующим образом:

$$u(x, 0) = y(\xi(x)) \cdot e^{i(kx - \theta_0)} + A e^{-\nu(x - x_0)^2}, \quad (2.6)$$

где  $A$  - амплитуда возмущения. Проведём моделирование для приведённого начального условия. Соответствующие численные результаты изображены на Рис. 2.4.



(a) Модуль начального условия (2.6)



(b) Модуль численного решения

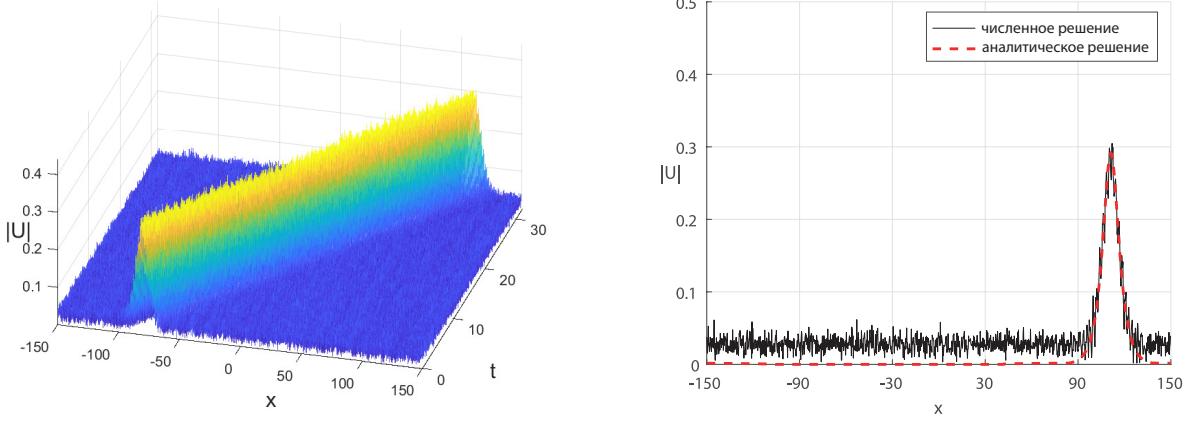
Рис. 2.4: Численные результаты при параметрах  $M_0 = -3$ ,  $M_1 = 12.34$ ,  $\varepsilon_2 = 1.04$ ,  $\varepsilon_3 = 0.23$ ,  $k = 3$ ,  $\xi_0 = 0$ ,  $z_0 = -80$ ,  $\theta_0 = 0$ ,  $L = 300$ ,  $T = 32$ ,  $h = 0.25$ ,  $\tau = 0.625$ ,  $A = 0.2$ ,  $\nu = 0.06$ ,  $x_0 = 25$ .

Проведённое моделирование позволяет сделать вывод, что солитон, заданный рассмотренными параметрами, взаимодействует с заданным возмущением, не распадаясь и не теряя способности к распространению. Профиль пульса восстанавливается после взаимодействия.

Также установлено, что солитон уравнения (1.4) устойчив при распространении в среде со случайным шумом следующего вида:

$$u(x, 0) = y(\xi(x)) \cdot e^{i(kx - \theta_0)} + A \cdot \text{rand}(x). \quad (2.7)$$

Результаты моделирования проиллюстрированы на Рис. 2.5.



(a) Модуль численного решения

(b) Профили решений при  $t = 32$ Рис. 2.5: Численные результаты при параметрах  $M_0 = -3$ ,  $M_1 = 12.34$ ,  $\varepsilon_2 = 1.04$ ,  $\varepsilon_3 = 0.23$ ,  $k = 3$ ,  $\xi_0 = 0$ ,  $z_0 = -80$ ,  $\theta_0 = 0$ ,  $L = 300$ ,  $T = 32$ ,  $h = 0.25$ ,  $\tau = 0.0625$ ,  $A = 0.05$ .

Моделирования, представленные в данном разделе, подтверждают стабильность солитонов, полученных в разделе 1 главы 1.

## 4 Анализ влияния высших степеней нелинейности на распространение уединённых волн нелинейного уравнения Шрёдингера

При решении физических задач классическое НУШ обобщается путём добавления в уравнение некоторых выражений, учитывающих определённые физические факторы. Поскольку реальные физические процессы могут протекать по более сложным законам, не всегда возможно заранее предугадать и учесть все необходимые уточняющие члены. В этом разделе мы исследуем влияние дополнительных нелинейных членов на решения НУШ.

Для изучения обсуждаемого влияния приведём два полезных уточнения:

1. Решение уравнения (1.4) при  $\varepsilon_3 = 0$  ранее уже было найдено в следующем виде:

$$u(x, t) = \left( \frac{4\mu e^{\sqrt{\mu}(x-2kt-z_0)}}{1 + 4e^{\sqrt{\mu}(x-2kt-z_0)} + (4 + 4\mu\nu)e^{2\sqrt{\mu}(x-2kt-z_0)}} \right)^{\frac{1}{2}} \cdot e^{i(kx-\omega t-\theta_0)}, \quad (2.8)$$

где  $\mu = 4(\omega - k^2)$ ,  $\nu = \frac{4\varepsilon_2}{3}$  и  $k$ ,  $\omega$ ,  $z_0$ ,  $\theta_0$  - произвольные константы. Заметим, что при  $\varepsilon_2 = 0$  решение (2.8) совпадает с решением (2.2).

2. Для проверки согласованности численно полученных результатов с физикой процесса возможно использовать законы сохранения. Модель (1.4) является консервацией

тивной. Следовательно, законы сохранения математически должны выполняться для всех моделирований с некоторой поправкой на численную погрешность. Повторяя алгоритм поиска из работы [41], первые два закона сохранения для мощности и переносимого импульса можно записать в следующем виде:

$$I_1 = \int_{-\infty}^{\infty} |u|^2 dx, \quad (2.9)$$

$$I_2 = -i \int_{-\infty}^{\infty} (u^* u_x - uu_x^*) dx. \quad (2.10)$$

Для сеточного численного решения на временном слое  $n$  воспользуемся следующими выражениями для аппроксимации значений интегралов (2.9) и (2.10):

$$I_{1,n}^{(h)} = \sum_{1 \leq m \leq N_x} |U^{m,n}|^2 dx, \quad (2.11)$$

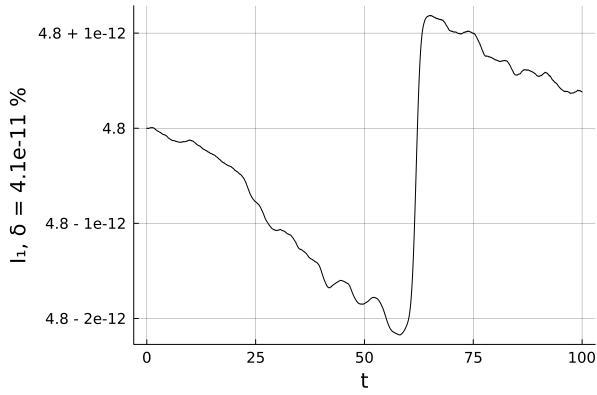
$$I_{2,n}^{(h)} = -i \sum_{1 \leq m \leq N_x} \left( (U^{m,n})^* \dot{U}^{m,n} - U^{m,n} (\dot{U}^{m,n})^* \right) dx, \quad (2.12)$$

где  $\dot{U}$  - узловая аппроксимация производной. Для этого будем использовать трехузловые формулы численного дифференцирования со вторым порядком аппроксимации:

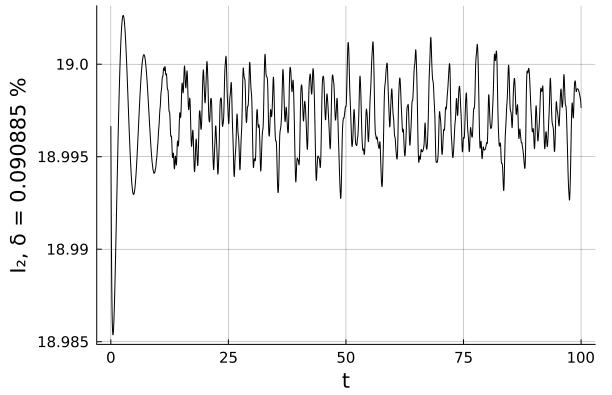
$$\begin{aligned} \dot{U}_0 &= \frac{-3U_0 + 4U_1 - U_2}{2h}, \\ \dot{U}_m &= \frac{U_{m+1} - U_{m-1}}{2h}, \\ \dot{U}_{N_x} &= \frac{U_{N_x-2} - 4U_{N_x-1} + 3U_{N_x}}{2h}. \end{aligned} \quad (2.13)$$

Исследуем влияние нелинейных выражений на распространение уединённой волны нелинейного уравнения Шрёдингера (2.2) в рамках обобщённой нелинейной модели, описываемой уравнением (1.4).

При  $\varepsilon_2 = 0, \varepsilon_3 = 0$  уединенная волна (2.2) является точным решением уравнения (1.4). Численное решение для начального условия (2.3) совпадает с аналитическим. Поведение законов сохранения (2.9) и (2.10) проиллюстрировано на Рис. 2.6, где  $\delta$  - относительная точность сохранения величины соответствующего интеграла.



(a) Зависимость  $I_1$  от времени.

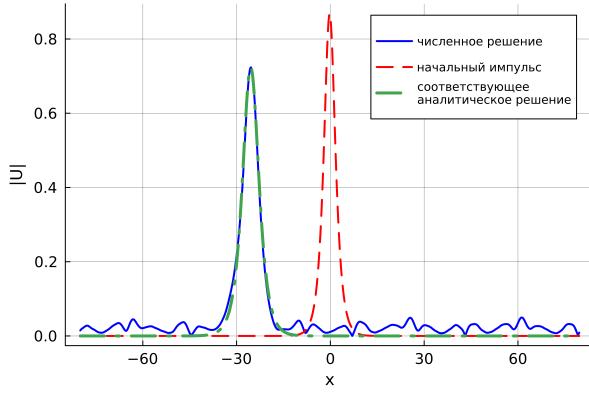


(b) Зависимость  $I_2$  от времени.

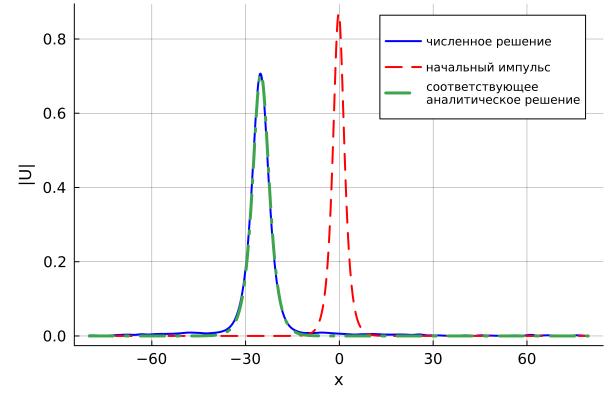
Рис. 2.6: Поведение законов сохранения при  $L = 100$ ,  $T = 100$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = 0$ ,  $\varepsilon_3 = 0$ ,  $\omega = 1.6$ ,  $k = 0.4$ .

В случае введения нелинейного члена при  $\varepsilon_2 \neq 0$ , импульс (2.3) претерпевает изменения в процессе распространения. Поскольку начальное условие не удовлетворяет уравнению модели (1.4), солитонные свойства импульса перестают выполняться, и его форма постепенно изменяется.

Излучение энергии при граничных периодических условиях нарушает изначальное предположение об уединённости импульса, создавая фон из-за периодических граничных условий. Для восстановления справедливости данного предположения, в процесс расчёта введена процедура фильтрации излучения в численном решении. За пределами характерной протяжённости импульса  $l_f$  каждые  $t_f$  единиц времени расчёта происходит уменьшение численного решения в  $k_f$  раз. При  $k_f = 1$  численное решение не изменяется. Подробный алгоритм фильтрации описан в приложении А в секции 3. Результат фильтрации проиллюстрирован на Рис.2.7. Алгоритм поддерживает задание динамического коэффициента  $k_f(t)$ , что позволяет отключить диссиацию излучения на произвольных временах моделирования.



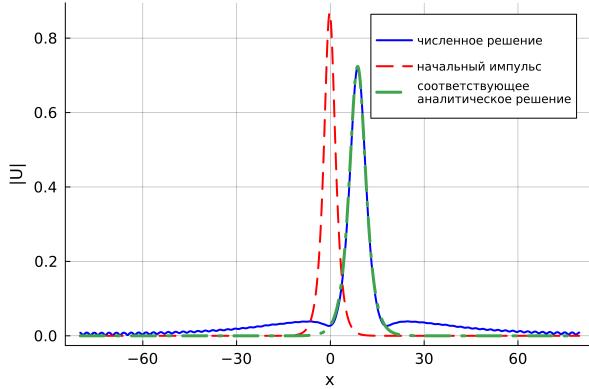
(a) Профиль решения при  $t = 450$  без применения фильтрации



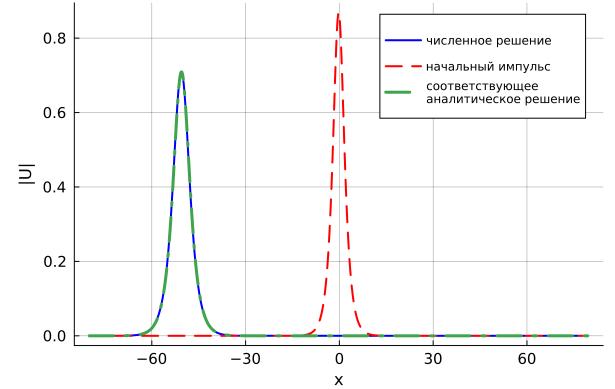
(b) Профиль решения при  $t = 450$  с применением фильтрации

Рис. 2.7: Численные результаты распространения импульса (2.3) при  $L = 160$ ,  $T = 450$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $l_f = 60$ ,  $t_f = 2$ ,  $k_f = 1.02$ ,  $\varepsilon_2 = -0.5$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Обнаружено, что форма импульса (2.3) в процессе расчёта становится устойчивой. По мере моделирования амплитуда солитона стремится к равновесному значению, а его форма - к соответствующему для данной амплитуды решению вида (2.8), являющемуся аналитическим для обобщённой модели при  $\varepsilon_2 \neq 0$ . Для расчёта, проиллюстрированного на Рис. 2.8 и Рис. 2.9 максимальная относительная ошибка между численным и аналитическим решением в конечный момент времени моделирования при  $T = 3500$  не превышает 0.05%.



(a) Профиль решения при  $t = 30$



(b) Профиль решения при  $t = 3500$

Рис. 2.8: Численные результаты распространения импульса (2.3) при  $L = 160$ ,  $T = 3500$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = -0.5$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Зависимость относительной ошибки между аналитическим (2.8) и численным решением от времени представлена на Рис. 2.9. Зависимость коэффициента фильтрации от времени также проиллюстрирована на графике. Коэффициент  $k_f(t)$  линейно убывает в процессе моделирования, и равен  $k_f = 1$  при  $t = 2500$ . Подбор аналитического решения происходит по максимуму модуля численного решения и параметру  $\varepsilon_2$ . Для

снижения сеточной ошибки при построении аналитического решения по численному, максимум модуля импульса определяется в результате решения задачи поиска глобального максимума В-сплайновой функции интерполяции, построенной для сеточного решения. Подробное описание алгоритма построения аналитического решения по сеточному представлено в приложении А в секции 5.

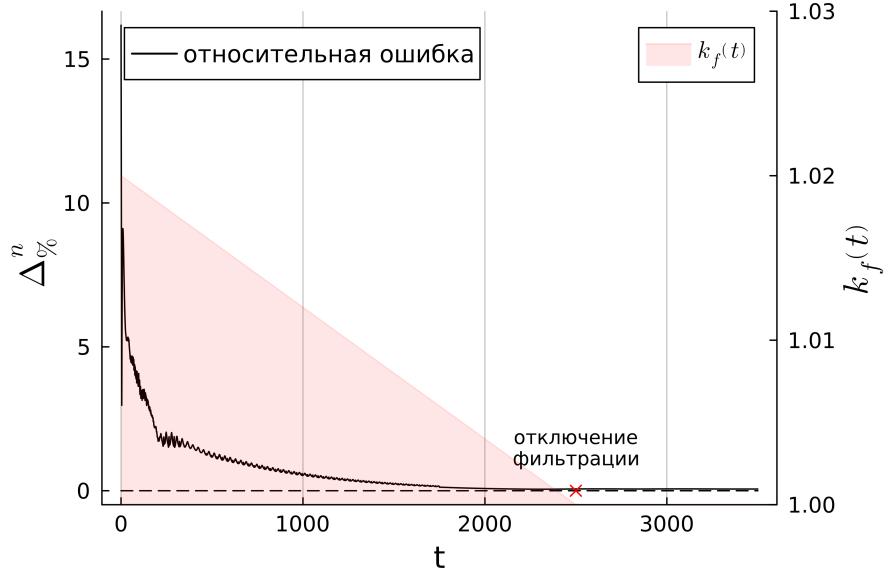
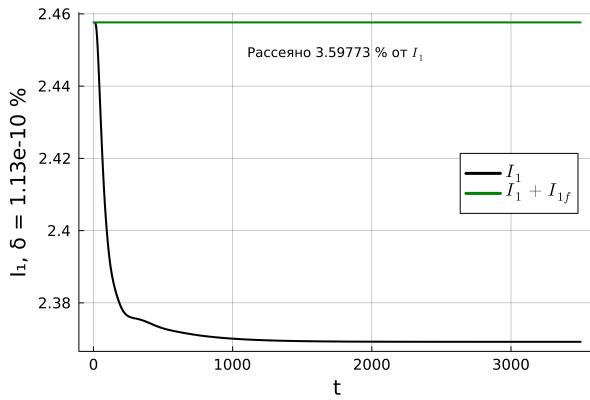
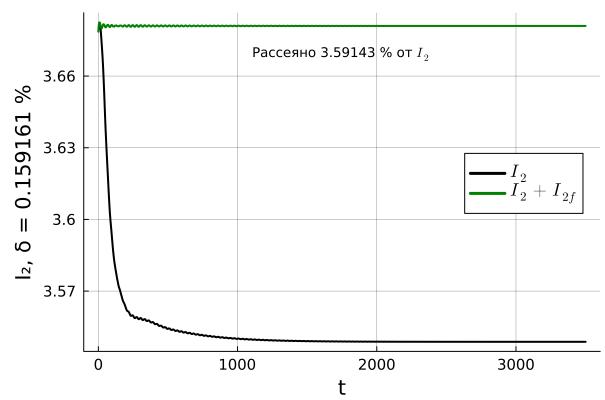


Рис. 2.9: Зависимости относительной ошибки и коэффициента фильтрации от времени при  $L = 160$ ,  $T = 3500$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = -0.5$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Поведение законов сохранения для представленного моделирования проиллюстрировано на Рис. 2.10. Солитоном потеряно 3.6% мощности и импульса, однако с учётом рассеянных составляющих оба интеграла сохраняются, что говорит о корректном применении процедуры фильтрации и выполнении законов сохранения. Величина  $\delta$  на Рис. 2.10 равна относительной точности, с которой в процессе всего моделирования сохраняются величины  $I_1 + I_{1f}$  и  $I_2 + I_{2f}$ , где  $I_{1f}$  и  $I_{2f}$  - рассеянные компоненты мощности и импульса.



(a) Зависимость  $I_1$  от времени.

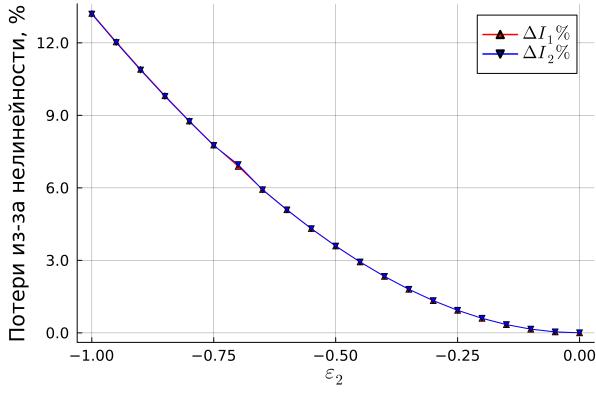


(b) Зависимость  $I_2$  от времени.

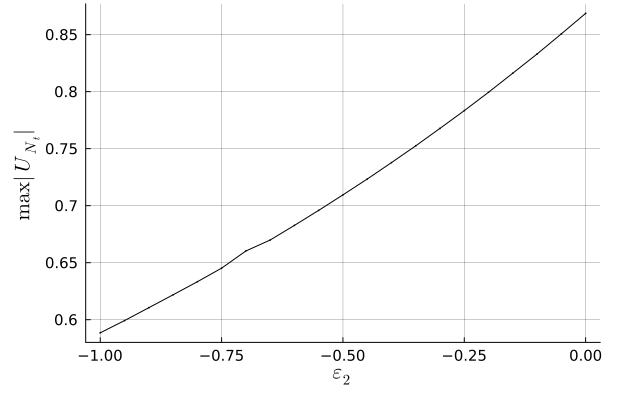
Рис. 2.10: Поведение законов сохранения при  $L = 160$ ,  $T = 1000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = -0.5$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Отметим, что более длительная фильтрация позволяет добиться на порядок меньшей ошибки между численным и аналитическим решением. Так, при времени окончания фильтрации  $t_{end,f} = T = 5000$  и линейном убывающем к единице законе изменения  $k_f(t)$  в конечный момент времени максимальная относительная разница аналитического и численного решений составляет 0.002%. При увеличении времени моделирования и конца фильтрации в два раза дополнительно рассеялось  $4 \times 10^{-4}\%$  от  $I_1$  и  $2 \times 10^{-4}\%$  от  $I_2$ , что говорит о том, что импульс во второй половине моделирования потерял в 8700 раз меньше мощности и в 17000 раз меньше импульса, чем в первой. Данное наблюдение позволяет заключить, что исходный импульс под влиянием нелинейных выражений в модели преобразуется в другой импульс меньшей амплитуды, совпадающей с аналитическим решением (2.8) уравнения (1.4) при  $\varepsilon_3 = 0$ . Данный процесс устанавливается со временем.

Выводы, полученные выше, справедливы для  $\varepsilon_2 < 0$ . Чем меньше при этом абсолютное значение параметра  $\varepsilon_2$ , тем меньше потерь импульсом на излучение, и тем меньше изменяется его амплитуда. Зависимости процентной величины потерь и амплитуды установившегося импульса от параметра нелинейности для отдельно взятых параметров модели и сетки проиллюстрирована на Рис. 2.11



(a) Процент потерь мощности и импульса

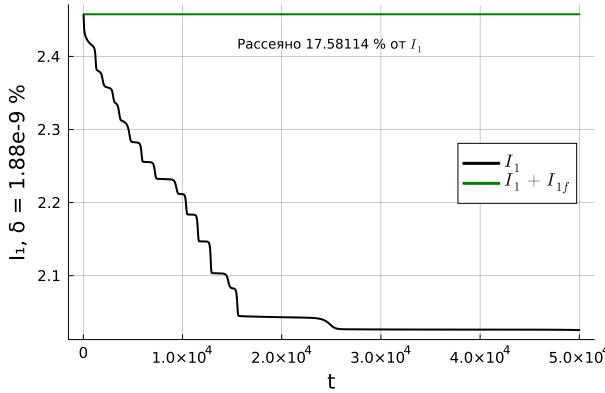


(b) Амплитуда решения в конечный момент

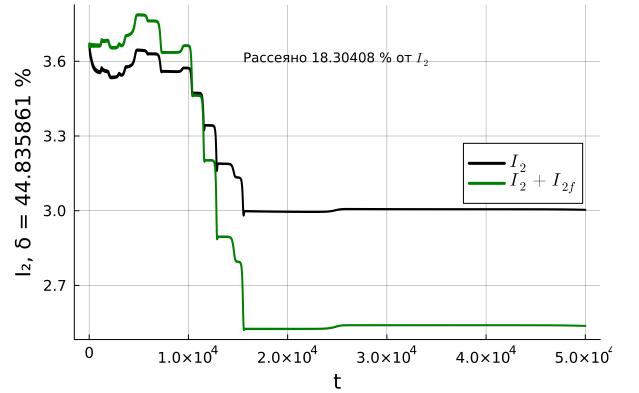
Рис. 2.11: Зависимости величин относительно  $\varepsilon_2$  при  $L = 160$ ,  $T = 3000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

В случае  $\varepsilon_2 > 0$  обнаружены следующее наблюдения. Амплитуда результирующего импульса в начале моделирования увеличивается. Потеря солитоном импульса и мощности происходит интенсивнее и дольше, чем в случае отрицательного  $\varepsilon_2$ , что приводит к необходимости фильтровать решение на больших временах и с большим коэффициентом  $k_f$ . В противном случае происходит зашумление численного решения.

Результаты моделирования для  $\varepsilon_2 = 0.5$  проиллюстрированы на Рис. 2.12 и Рис. 2.13. Мощность  $I_1$  сохраняется с учётом диссирированной в результате фильтрации. Поведение законов сохранения проиллюстрировано на Рис. 2.12. Суммарная величина  $I_2 + I_{2f}$  не сохраняется, изменившись на 44.8%. В конце моделирования при  $T = 50000$  солитоном потеряно 17.6% мощности и 18.3% импульса. В конечный момент моделирования ошибка между подобранным аналитическим и численным решением составляет 0.09%. Амплитуда импульса в конце моделирования колеблется, изменяясь в пределах 0.12% в зависимости от времени.

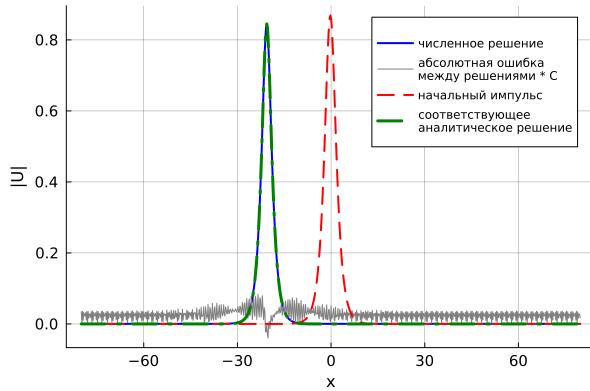


(a) Зависимость  $I_1$  от времени

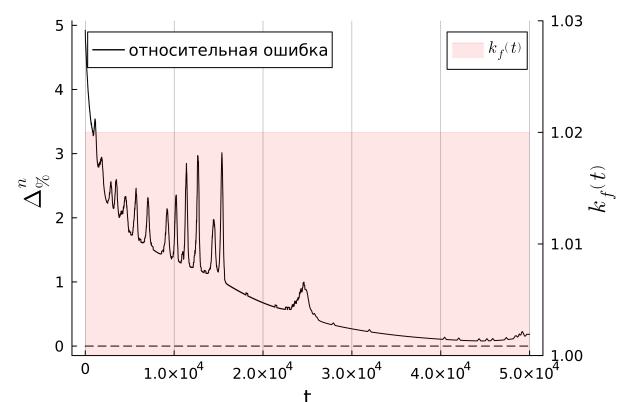


(b) Зависимость  $I_2$  от времени

Рис. 2.12: Поведение законов сохранения при  $L = 160$ ,  $T = 50000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = 0.5$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .



(a) Профиль решения при  $t = 50000$

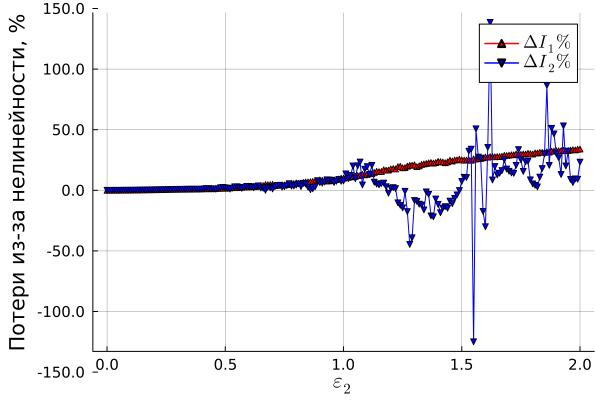


(b) Зависимости относительной ошибки и коэффициента фильтрации от времени

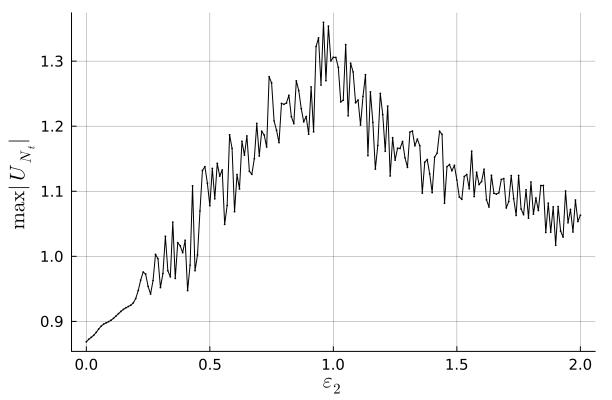
Рис. 2.13: Численные результаты при  $L = 160$ ,  $T = 50000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = 0.5$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Исходный солитон под воздействием степенного нелинейного члена с положительным коэффициентом претерпевает переходный процесс, который требует на порядок больше времени на установление, чем в случае  $\varepsilon_2 < 0$ , и происходит с более интенсивной потерей импульса и мощности. Учитывая, что при этом закон сохранения импульса не выполняется, моделирование процесса распространения уединённой волны в рамках модели с  $\varepsilon_2 > 0$  в общем случае не имеет физического смысла, однако полезно для понимания влияния нелинейных членов в уравнении модели.

В качестве дополнительного аргумента в пользу несостоятельности моделирования процессов распространения импульса при  $\varepsilon_2 > 0$  приведём зависимости конечной амплитуды импульса и его потерь в зависимости от  $\varepsilon_2 > 0$ . Результаты проиллюстрированы на Рис. 2.14. Зависимость перестаёт быть регулярной. При моделях нарушается сеточная сходимость.



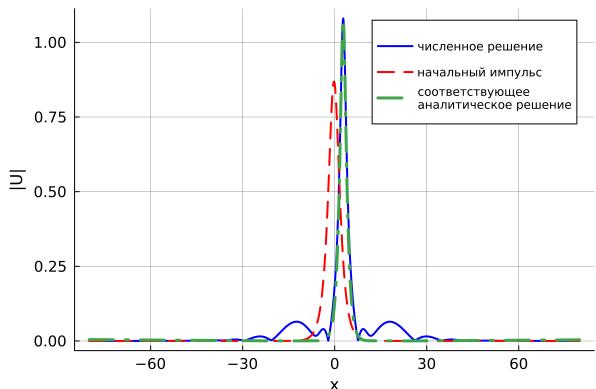
(a) Процент потерь мощности и импульса



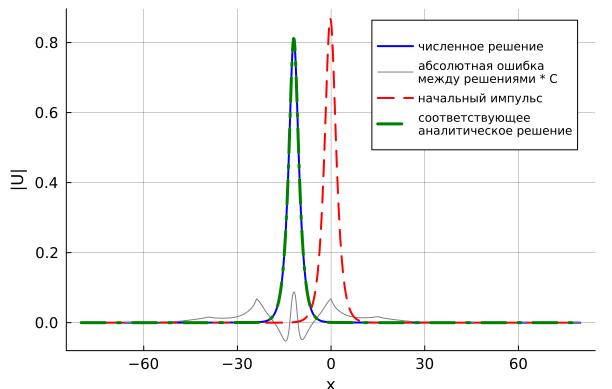
(b) Амплитуда решения в конечный момент

Рис. 2.14: Зависимости величин относительно  $\varepsilon_2$  при  $L = 160$ ,  $T = 1000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_3 = 0$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Рассмотрим более общий характер нелинейностей в модели при  $\varepsilon_3 \neq 0$ . Отметим, что аналитическое решение (1.27), построенное в разделе 1 главы 1 существует только для положительных  $\varepsilon_2$  и  $\varepsilon_3$  в силу ограничений и соотношений, использованных при его выводе. Более того, аналитическое решение существует только для тех  $\varepsilon_2$  и  $\varepsilon_3$ , по которым возможно найти параметры  $M_0$  и  $M_1$  из соотношений (1.22). В противном случае построить по данным параметрам нелинейности соответствующее аналитическое решение не получится. Проверим, возможен ли переход исходного импульса (2.2) под действием положительных параметров  $\varepsilon_2$  и  $\varepsilon_3$  к построенному решению (1.27). На Рис. 2.15 проиллюстрированы профили решения в различные моменты времени.



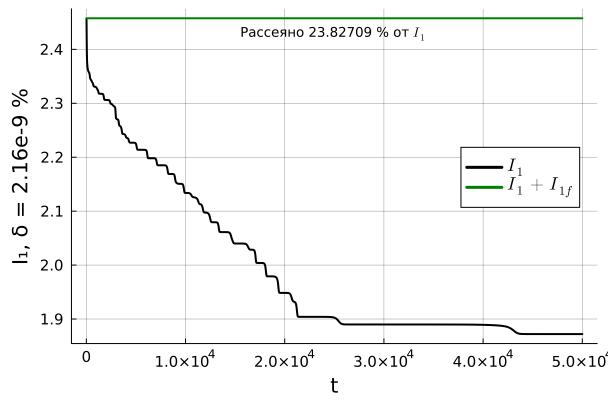
(a) Профиль решения при  $t = 10$



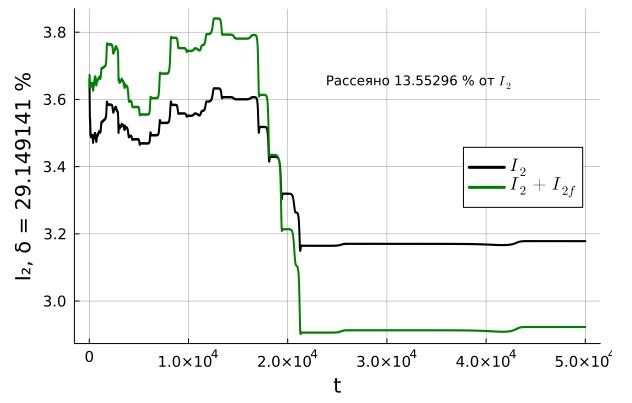
(b) Профиль решения при  $t = 50000$

Рис. 2.15: Численные результаты распространения импульса (2.3) при  $L = 160$ ,  $T = 50000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = 0.65$ ,  $\varepsilon_3 = 0.08$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

На Рис. 2.16 проиллюстрировано поведение законов сохранения для моделирования. Результаты аналогичны моделированию, проведённому для  $\varepsilon_2 = 0.5$  при  $\varepsilon_3 = 0$ . Закон сохранения импульса не выполняется. В конце моделирования по численному решению подобран солитон вида (1.27) с относительной разницей, равной 0.107%.



(a) Зависимость  $I_1$  от времени.



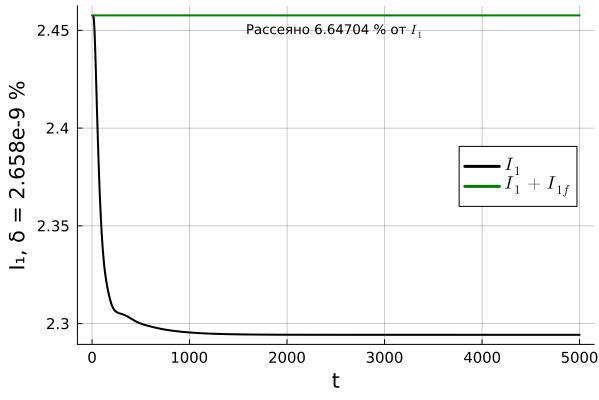
(b) Зависимость  $I_2$  от времени.

Рис. 2.16: Поведение законов сохранения при  $L = 160$ ,  $T = 50000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\varepsilon_2 = 0.65$ ,  $\varepsilon_3 = 0.08$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

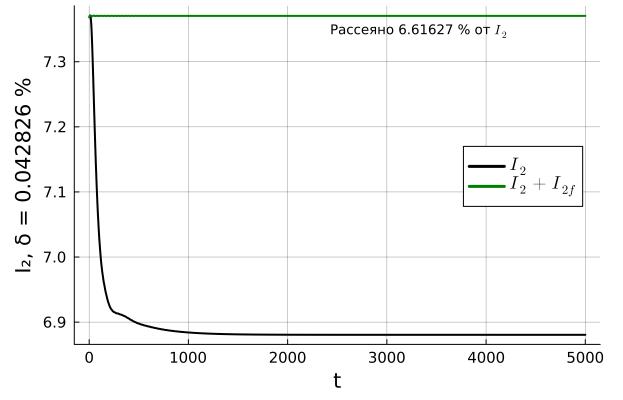
Колебания амплитуды импульса в конце моделирования происходят в переделах 0.5%, что позволяет сделать вывод, что для того же времени моделирования и параметров импульса, но больших положительных параметрах нелинейностей в модели, чем для расчёта, проиллюстрированного на Рис. 2.13 процесс перехода к концу моделирования установился хуже. Это согласуется с фактом, что для больших положительных коэффициентов  $\varepsilon_2$  и  $\varepsilon_3$  процесс устанавливается дольше.

Несмотря на то, что построить аналитическое решение с точностью порядка 0.1% по итоговому численному возможно, сделать вывод о сходимости к построенному решению в условиях нарушения законов сохранения не представляется возможным.

В противоположной ситуации, для отрицательных ненулевых  $\varepsilon_2$  и  $\varepsilon_3$ , в процессе моделирования наблюдается сходимость. Однако решение, построенное в разделе 1 главы 1 не существует для данных параметров. Алгоритм подбора аналитического решения по численному в данном случае не срабатывает, и сравнивать численное решение не с чем. Процесс перехода к моменту  $T = 5000$  установился с точностью  $3.7 \times 10^{-4}\%$ , законы сохранения выполняются, и процесс потери солитоном мощности и импульса прекратился. Зависимости интегралов от времени проиллюстрированы на Рис. 2.17.



(a) Зависимость  $I_1$  от времени.



(b) Зависимость  $I_2$  от времени.

Рис. 2.17: Поведение законов сохранения при  $L = 160$ ,  $T = 5000$ ,  $h = 0.1$ ,  $\tau = 0.02$ ,  $\varepsilon_2 = -0.65$ ,  $\varepsilon_3 = -0.08$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Несмотря на отсутствие аналитического решения, мы утверждаем, что сходимость численного решения достигнута. Законы сохранения выполняются, потери солитоном интегралов прекращаются - за вторую половину моделирования рассеяно в  $23 \times 10^3$  раз меньше  $I_1$ , чем в первую, и аналогично, в  $27 \times 10^3$  раз меньше  $I_2$ . Амплитуда численного решения устанавливается. Её поведение проиллюстрировано на Рис. 2.18.

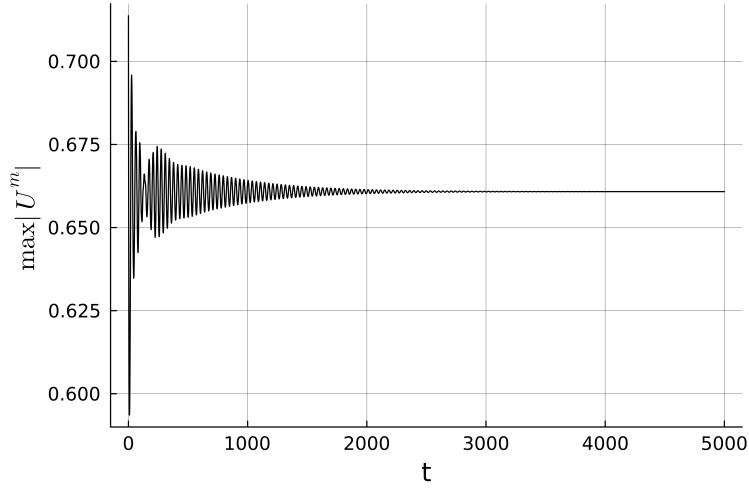


Рис. 2.18: Зависимость амплитуды численного решения от времени при  $L = 160$ ,  $T = 5000$ ,  $h = 0.1$ ,  $\tau = 0.01$ ,  $\varepsilon_2 = -0.65$ ,  $\varepsilon_3 = -0.08$ ,  $\omega = 0.4$ ,  $k = 0.15$ .

Определив влияние одновременно отрицательных или одновременно положительных  $\varepsilon_2$  и  $\varepsilon_3$ , возможно предположить, что комбинации в математической модели параметров разных знаков будет иметь усреднённые свойства, характерные для параметров каждого знака. Положительные значения стремятся нарушить сеточную сходимость и законы сохранения, привести к большим временам установления. Отрицательные значения, вероятно, нивелируют этот эффект.

Проиллюстрируем некоторые варианты поведения амплитуды импульса в зависимости от параметров нелинейностей на Рис. 2.19 и Рис. 2.20. Здесь  $\delta$  — процент, в пределах которого изменяется амплитуда импульса в конце временного промежутка моделирования. Величина  $\delta$  приведена только для условно устанавливающихся переходных процессов.

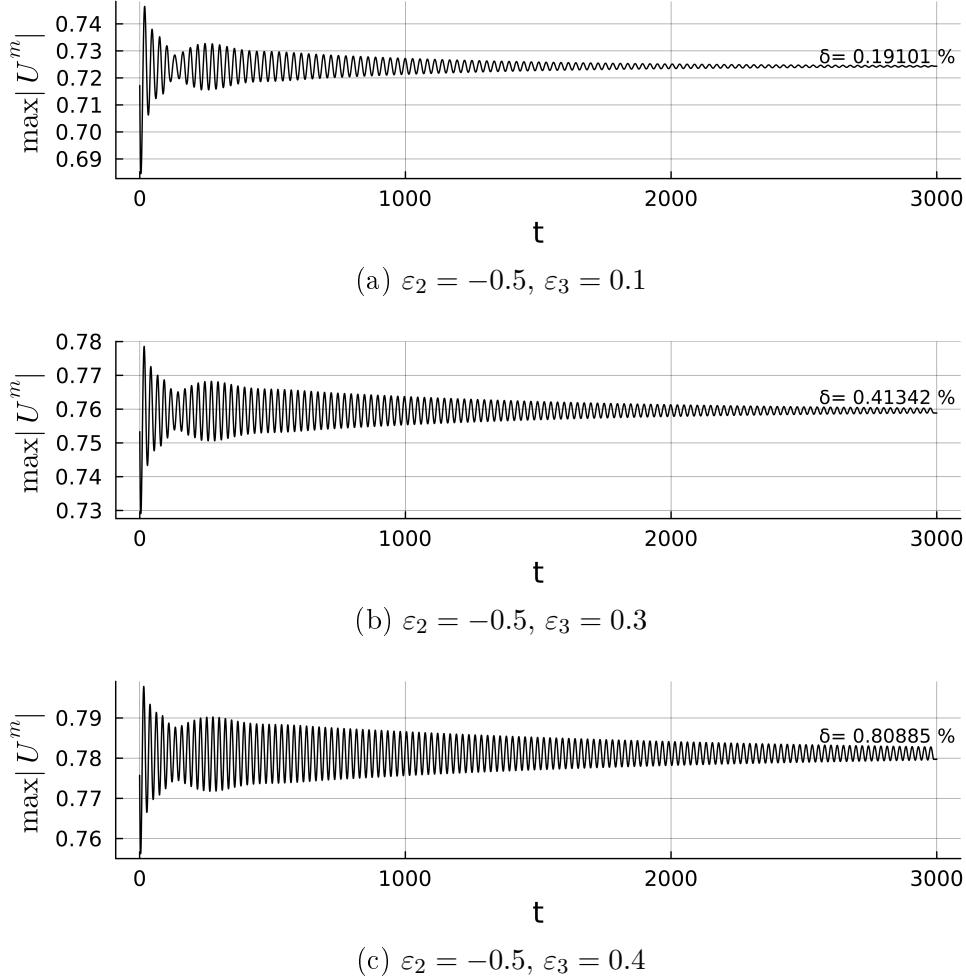
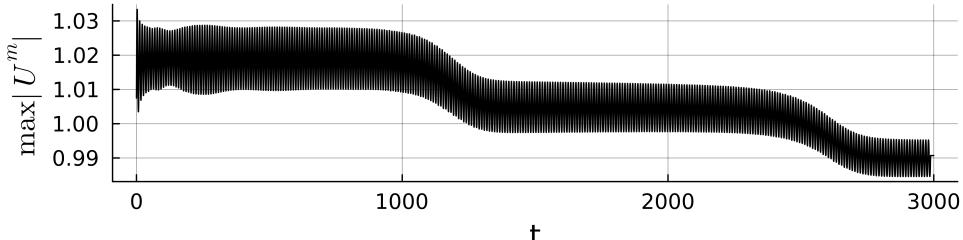
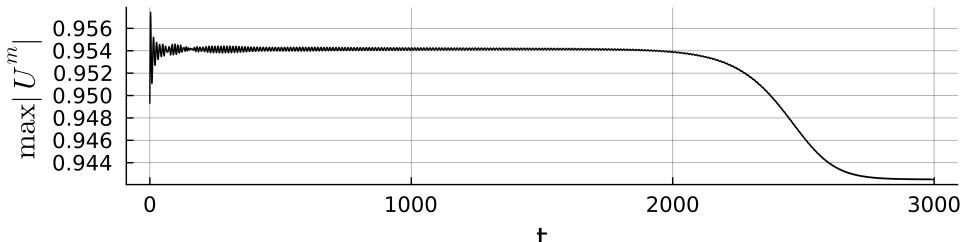


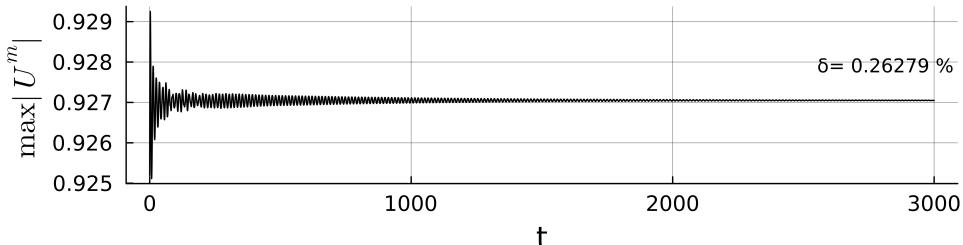
Рис. 2.19: Зависимость амплитуды импульса (2.3) при распространении .  $L = 160$ ,  $T = 3000$ ,  $h = 0.2$ ,  $\tau = 0.04$ ,  $\omega = 0.4$ ,  $k = 0.15$ .



(a)  $\varepsilon_2 = 0.5, \varepsilon_3 = -0.1$



(b)  $\varepsilon_2 = 0.5, \varepsilon_3 = -0.3$



(c)  $\varepsilon_2 = 0.5, \varepsilon_3 = -0.4$

Рис. 2.20: Зависимость амплитуды импульса (2.3) при распространении .  $L = 160, T = 3000, h = 0.2, \tau = 0.04, \omega = 0.4, k = 0.15$ .

Для случаев 2.20a, 2.20b законы сохранения нарушены. Для случая 2.20b - выполнены. Из приведённых результатов можно предположить, что установление при моделировании процесса распространения в рамках модели с нелинейными параметрами разных знаков возможно, причём независимо от того, какой из параметров положителен. Однако заметим, что в реальных физических приложениях коэффициенты в уравнении модели при высших нелинейных степенных членах уменьшаются по мере роста степени нелинейности.

С целью определения влияния нелинейных членов построим диаграмму в координатах  $\varepsilon_2, \varepsilon_3$ , на которой проиллюстрируем зависимость относительного процентного нарушения законов сохранения. Результаты, изображённые на Рис. 2.21, получены для различных комбинаций параметров  $\varepsilon_2$  и  $\varepsilon_3$ . Моделирования проведены на одинаковых сетках для одинаковых начальных условий.

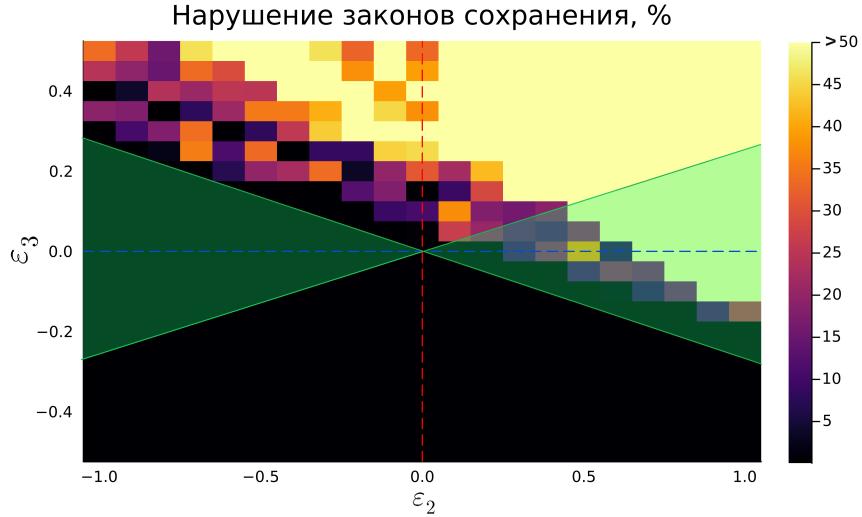


Рис. 2.21: Диаграмма влияния  $\varepsilon_2$  и  $\varepsilon_3$  на сохранение интегралов в процессе моделирования  $L = 120$ ,  $T = 2000$ ,  $h = 0.1$ ,  $\tau = 0.01$ ,  $\omega = 0.45$ ,  $k = 0.15$ .

Тёмные области на диаграмме иллюстрируют моделирования, для которых законы сохранения не нарушились. Светлые области соответствуют значениям параметров  $\varepsilon_2$  и  $\varepsilon_3$ , для которых законы сохранения нарушились. Цвет на Рис. 2.21 пропорционален погрешности, в пределах которой за время моделирования сохранялись величины  $I_1 + I_{1f}$  и  $I_2 + I_{2f}$ .

Для всех моделей, проведённых при положительных  $\varepsilon_2$  и  $\varepsilon_3$ , законы сохранения нарушены. Для всех моделей, проведённых при отрицательных  $\varepsilon_2$  и  $\varepsilon_3$ , законы сохранения выполняются. Для моделей в рамках математических моделей с разными знаками коэффициентов при нелинейных членах возможны различные варианты, в зависимости от модуля коэффициентов.

В реальных математических моделях коэффициент  $\varepsilon_3$  имеет меньший порядок, чем  $\varepsilon_2$ . Следовательно, существует связь вида

$$|\varepsilon_3| < C \cdot |\varepsilon_2|, \quad (2.14)$$

где  $C$  - некоторая положительная константа. Область, удовлетворяющая ограничению (2.14) для  $C = 0.28$  выделена на Рис. 2.21 зелёным. Из проведённого численного анализа эмпирически следует, что моделирование распространения импульса (2.3) в рамках модели (1.4) с отрицательным  $\varepsilon_2$  происходит с выполнением законов сохранения независимо от знака  $\varepsilon_3$ . Процессы распространения для положительного  $\varepsilon_2$  сопровождаются нарушением законов сохранения независимо от знака  $\varepsilon_3$ . Данный вывод справедлив для импульса (2.14) с рассмотренными параметрами  $\omega = 0.45$ ,  $k = 0.15$  в рамках моделирования на сетке с параметрами  $L = 120$ ,  $T = 2000$ ,  $h = 0.1$ ,  $\tau = 0.01$  при условии выполнения соотношения (2.14) между коэффициентами  $\varepsilon_2$  и  $\varepsilon_3$ .

Отметим, что исследуемая математическая модель является консервативной.

Следовательно, законы сохранения математически обязаны выполняться. Наблюдаемые нарушения законов и отсутствие сеточной сходимости для светлых областей на Рис. 2.21 говорит о несоответствии результатов данных моделирований реальной физике процесса. Нарушение законов сохранения происходит из-за численной аппроксимации производной для решения, которая вычисляется с существенными погрешностями для распавшегося и интерферирующего с самим собой импульса в силу существенной негладкости его профиля. Мы предполагаем, что распад импульса происходит по причине резкого накопления численных ошибок при определённых комбинациях параметров  $\varepsilon_2$  и  $\varepsilon_3$ , проиллюстрированных на диаграмме 2.21. В качестве подтверждения данного предположения выступает наблюдение: распад импульса происходит тем раньше, чем мельче разностная сетка для численного решения задачи.

Результаты, представленные в данном разделе, позволяют сделать вывод о том, что уединенные волны НУШ при распространении в среде с высшими нелинейными членами при определённых параметрах математической модели преобразуются в устойчивые солитоны обобщенной неинтегрируемой модели. Для рассмотренного импульса определены параметры математической модели, при которых возможен переход к устойчивому решению. Преобразование импульса происходит с выполнением законов сохранения для  $\varepsilon_2 < 0$ , и с нарушением при  $\varepsilon_2 > 0$  независимо от знака  $\varepsilon_3$  при соблюдении условия малости:  $|\varepsilon_3| < C \cdot |\varepsilon_2|$ , где  $C$  - некоторая константа. Для наблюдения перехода в процессе расчёта необходимо диссипировать излучение за пределами характерной протяжённости импульса.

Заметим, что математически, при равных абсолютных значениях  $\varepsilon_2$  и  $\varepsilon_3$ , большее влияние на процесс распространения оказывает коэффициент  $\varepsilon_3$ . В результате численного анализа установлено, что для некоторого  $C \in (0, 1)$  при  $\varepsilon_3 > -C \cdot \varepsilon_2$  численные ошибки резко возрастают по мере моделирования, импульс распадается, и законы сохранения нарушаются. При  $\varepsilon_3 < -C \cdot \varepsilon_2$  результаты моделирования согласуются с физическими законами.

## 5 Столкновения солитонов в присутствии высших степеней нелинейности

Известно, что решения интегрируемого нелинейного уравнения Шрёдингера взаимодействуют упруго, т.е. без обмена импульсом и энергией. При нарушении интегрируемости системы внешними возмущениями солитонные столкновения становятся неупругими. В этом разделе мы исследуем столкновения солитонов НУШ в среде, описываемой возмущённым уравнением (1.4).

Рассмотрим столкновения двух солитонов вида (2.3) с заданными параметрами  $k_1, k_2, \omega_1, \omega_2, z_{0,1}, z_{0,2}, \theta_{0,1}, \theta_{0,2}$ . Значения параметров  $\varepsilon_2$  и  $\varepsilon_3$  в данном случае влияют

на интенсивность обмена импульсом и энергией. Обнаружено, что итоговый характер взаимодействия солитонов зависит также от разности фаз в момент столкновения  $\Delta\theta = \theta_{0,1} - \theta_{0,2}$ .

Результаты моделирования для  $k_1 = -k_2$ ,  $\omega_1 = \omega_2$ ,  $z_{0,1} = -z_{0,2}$ ,  $\theta_{0,1} = \theta_{0,2} + \Delta\theta$  проиллюстрированы на Рис. 2.22 и Рис. 2.23.

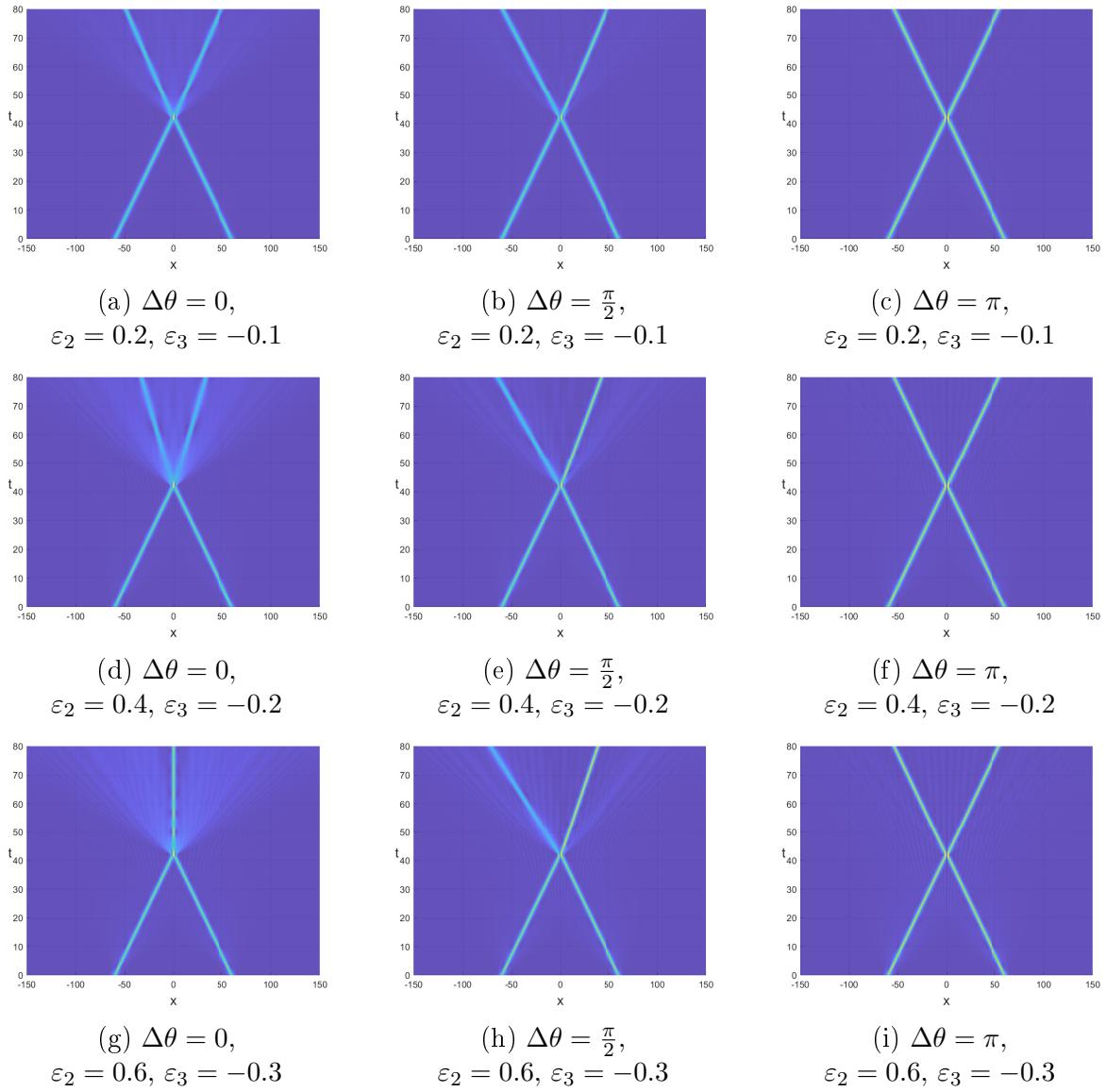


Рис. 2.22: Модуль численного решения при моделировании солитонных столкновений для  $k_1 = -k_2 = 0.7$ ,  $\omega_1 = \omega_2 = 0$ .

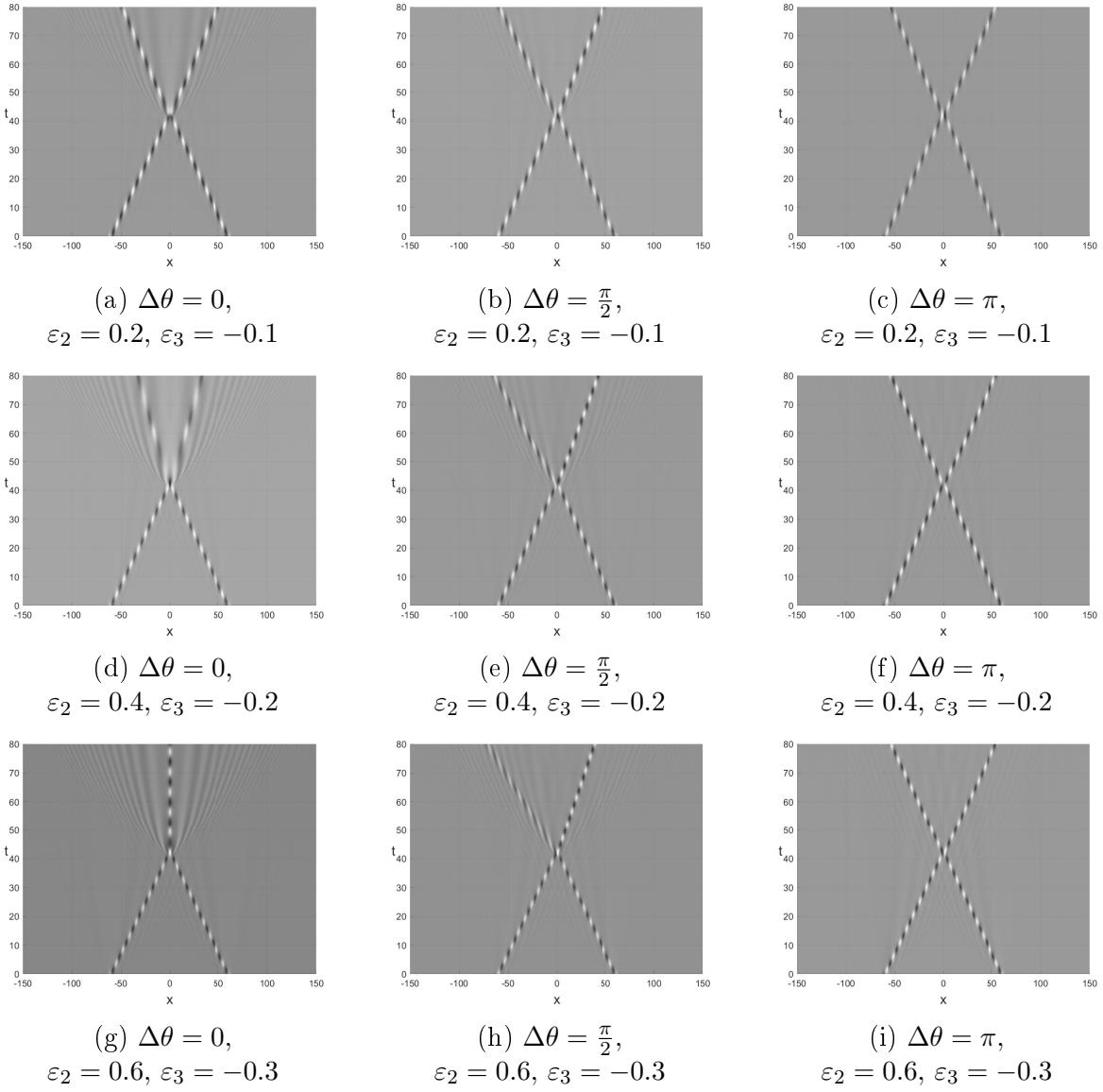


Рис. 2.23: Действительная часть численного решения при моделировании солитонных столкновений для  $k_1 = -k_2 = 0.7$ ,  $\omega_1 = \omega_2 = 0$ .

Результаты моделирования позволяют заключить, что столкновения солитонов НУШ в рамках математической модели, включающей нелинейные члены высшего порядка в зависимости от разности фаз в момент столкновения могут быть существенно неупругими. Вблизи  $\Delta\theta = \pi$  солитоны взаимодействуют наименее интенсивно. При параметре  $\Delta\theta \in (0, \pi)$  взаимодействие солитонов сопровождается обменом энергии и импульса. Когда разность фаз находится в некоторой окрестности нуля, при столкновении происходит потеря мощностей и импульсов сталкивающихся солитонов. Существуют пороговые параметры нелинейностей, при которых результирующий солитон после столкновения перестаёт распространяться. Помимо разности фаз, на определенный тип взаимодействия влияют значения параметров возмущения  $\varepsilon_2$  и  $\varepsilon_3$ . Чем больше модуль коэффициента при соответствующей степени нелинейности, тем более выражен неупругий характер взаимодействия.

# Глава 3

## Заключение

В данной работе проведено численное моделирование процессов распространения импульсов в нелинейной оптической среде с периодическими граничными условиями, описываемой обобщённым уравнением Шрёдингера (2) с нелинейными членами третьего, пятого и седьмого порядков. Получено аналитическое решение в виде уединенной волны (1.27) и условия ее существования. Представлена модификация конечно-разностного и Фурье методов для численного решения поставленной задачи. С численной точки зрения исследован процесс распространения аналитически полученного солитонного решения обобщённой модели. Проведено моделирование взаимодействия оптического солитона уравнения (2) с возмущением в начальных данных. Смоделировано распространение оптического импульса в среде со случайным шумом. Проанализировано влияние высших степеней нелинейности в математической модели на распространение уединенных волн нелинейного уравнения Шрёдингера. Проведено моделирование процессов столкновения солитонов в условиях наличия высших нелинейных членов.

Следующие результаты получены в результате исследования:

1. Уединённые волны уравнения Шрёдингера с нелинейными членами третьей, пятой и седьмой степеней распространяются устойчиво.
2. Оптические солитоны уравнения Шрёдингера с нелинейными членами третьей, пятой и седьмой степеней не распадаются при возмущениями начальных условий или при распространении в условиях случайного шума.
3. Определены параметры, для которых при распространении в оптической среде, описываемой математической моделью с нелинейными членами более высокого порядка солитоны НУШ преобразуются в солитоны, удовлетворяющие обобщённому уравнению.
4. В условиях наличия нелинейных членов высшего порядка столкновения солитонов НУШ происходят значительно неупруго. При определённых параметрах возмож-

но образование стоячих волн.

# Приложение А - Коды программ, использованных в работе

Для исследования процессов, обсуждённых в настоящей работе, создан проект на языке julia. Основные коды использованных функций приведены ниже.

## 1 Функции, определяющие аналитические решения

### 1.1 Солитон классического НУШ

```
1 function NSE_3_soliton(
2     x, t,
3     k::Real,
4     omega::Real,
5     theta_0::Real,
6     z_0::Real;
7     cycle::Bool = false,
8     L::Real = 0.0,
9 )
10    mu = (omega - k^2)
11    mu > 0.0 || throw(ArgumentError("mu <= 0. Некорректные k и omega."))
12    c = 2 * k
13    if cycle
14        if t > (L/2 + x)/c
15            t -= L/c * floor(1/2 + (c * t - x)/L)
16        end
17    end
18    Complex(
19        4 * (mu) / (
20            2 * mu * exp(-(x - z_0 - 2k * t) * sqrt(mu)) +
21            exp((x - z_0 - 2k * t) * sqrt(mu))
22        )
23    ) * exp(im * (k * x - theta_0 - omega * t))
24 end
```

### 1.2 Солитон НУШ с нелинейностью 3 и 5 степеней

```
1 function NSE_3_5_soliton(
2     x, t,
3     k::Real,
4     omega::Real,
5     epsilon_2::Real,
6     theta_0::Real,
7     z_0::Real;
```

```

8     cycle::Bool = false,
9     L::Real = 0.0,
10    )
11    mu = 4 * (omega - k^2)
12    mu > 0.0 || throw(ArgumentError("mu <= 0. Некорректные k и omega."))
13    c = 2 * k
14    if cycle
15        if t > (L/2 + x)/c
16            t -= L/c * floor(1/2 + (c * t - x)/L)
17        end
18    end
19    power = sqrt(mu) * (x - 2k * t - z_0)
20    v = 4/3 * epsilon_2
21    Complex(sqrt(
22        (4 * mu * exp(power))
23        / (1 + 4 * exp(power) + (4 + 4 * mu * v) * exp(2 * power)))
24        ) * exp(im * (k * x - theta_0 - omega * t))
25    end

```

### 1.3 Солитон НУШ с нелинейностью 3, 5 и 7 степеней

```

1 function evaluate_mu(M0, M1)
2     mu_a = sqrt(M1 / (M0 * (M1 + 6 * M0)))
3     mu_b = -mu_a
4     return mu_a
5 end
6 function xi_edges(mu, M0, M1, epsilon_left, epsilon_right, xi_0)
7     sqrt_expression = sqrt(4 * M0 * M1 + M1^2)
8     return (
9         log((sqrt_expression - 2 * M0 - M1) / (2 * M0))
10        / mu + xi_0 + epsilon_left,
11        log((-sqrt_expression - 2 * M0 - M1) / (2 * M0))
12        / mu + xi_0 - epsilon_right
13    )
14 end
15 function hyperbolic_space(a::Real, b::Real, N::Int; density::Real = 1.0)
16     N >= 2 || error("Количество точек меньше 2")
17     density > 0 || error("Параметр density должен быть положительным")
18     linear_space = range(-1.0, stop = 1.0, length = N)
19     hyperbolic_space = tanh.(linear_space * 2 * density) / tanh(2 * density)
20     return a .+ (b - a) .* (hyperbolic_space .+ 1) / 2
21 end
22 function evaluate_z(xi_vector, mu, M0, M1, z_0, xi_0)
23     sqrt_expression = sqrt(4 * M0 * M1 + M1^2)
24     _atanh_expression =
25         (2 * M0 * exp.(mu * (xi_vector .- xi_0)))
26         .+ 2 * M0 .+ M1) / sqrt_expression
27     v_clipped = [x > 1 ? 1 : x < -1 ? -1 : x for x in _atanh_expression]
28     atanh_expression = unique(v_clipped)
29     return z_0 .+ xi_vector / M0 + (2 * M1) / (mu * M0 * sqrt_expression) *
30         atanh.(atanh_expression)
31 end
32 function evaluate_y(xi_vector, mu, M0, M1, xi_0)
33     exp_expression = (1 .+ exp.(mu * (xi_vector .- xi_0)))
34
35     complex_y = sqrt.(Complex.(
36         M0 .+ M1 ./ exp_expression - M1 ./ (exp_expression .^ 2)
37     ))
38     all(imag(complex_y).==0) ||
39         @warn "При вычислении y(z) обнаружены и отброшены комплексные числа."

```

```

40     complex_y[imag(complex_y) .! = 0.0] .= Complex(0.0)
41     return Float64[real(x) for x in complex_y]
42 end
43 function precompile_NSE_3_5_7_soliton(
44     epsilon_2::Real,
45     epsilon_3::Real,
46     z_0::Real,
47     xi_0::Real,
48     L::Real;
49     use_M_values = false,
50     M0 = 0.0,
51     M1 = 0.0,
52 )
53     if !use_M_values
54         (M0, M1) = epsilon2_epsilon3_to_M0_M1(epsilon_2, epsilon_3)
55     end
56     println("M0 = ", round(M0, digits = 4), " M1 = ", round(M1, digits = 4))
57     mu = evaluate_mu(M0, M1)
58
59     iters, iters_limit, success, epsilon_left, epsilon_right = 0, 50, false, 0, 0
60     while !success && iters < iters_limit
61         iters += 1
62         (xi_left, xi_right)
63             = xi_edges(mu, M0, M1, epsilon_left, epsilon_right, xi_0)
64         global xi_vector = hyperbolic_space(xi_left, xi_right, 10^6; density = 1)
65         global z_vector = evaluate_z(xi_vector, mu, M0, M1, z_0, xi_0)
66         if z_vector[1]==-Inf
67             epsilon_left += eps()
68         end
69         if z_vector[end]==Inf
70             epsilon_right += eps()
71         end
72         success = !any(isinf, z_vector)
73         iters==iters_limit && error("Ошибка предкомпиляции решения: MaxIters")
74     end
75     maximum(z_vector) > L/2 ||
76         @warn "предкомпиляция решения: выход за правую границу. \
77             Будет использована экстраполяция"
78     minimum(z_vector) < -L/2 ||
79         @warn "предкомпиляция решения: выход за левую границу. \
80             Будет использована экстраполяция"
81
82     y_vector = evaluate_y(xi_vector, mu, M0, M1, xi_0)
83     interpolator = extrapolate(
84         interpolate((z_vector,), y_vector, Gridded(Linear())), Line()
85     )
86     return interpolator
87 end
88 function reduce_negative_values(x::Real)::Real
89     return x < 0.0 ? 0.0 : x
90 end
91 function reduce_negative_values(xs::AbstractVector{<:Real})::Vector{Real}
92     return [reduce_negative_values(x) for x in xs]
93 end
94 function NSE_3_5_7_soliton(
95     x,
96     t,
97     k::Real,
98     omega::Real,
99     theta_0::Real,

```

```

100     z_0::Real,
101     precompiled_data;
102     # Cycling parameters
103     cycle::Bool = false,
104     L::Real = 0.0,
105 )
106 y = precompiled_data
107 c = 2 * k
108 if cycle
109     if t > (L / 2 + x) / c
110         t -= L / c * floor(1 / 2 + (c * t - x) / L)
111     end
112 end
113 z = x - 2k * t - z_0
114
115 reduce_negative_values(y.(z)) * exp(
116     1im * (k * x - theta_0 - omega * t)
117 )
118 end

```

## 2 Функции для расчёта интегралов (законов сохранения)

### 2.1 Мощность

```

1 function integral_1(
2     U::Union{Vector{ComplexF64}, Vector{Float64}},
3     h::Float64,
4 )
5     return h * sum((abs.(U)).^2)
6 end

```

### 2.2 Импульс

```

1 function integral_2(
2     U::Union{Vector{ComplexF64}, Vector{Float64}},
3     h::Float64,
4 )
5     dU = [(U[2] - U[end]), (U[3:end] - U[1:end - 2])..., (U[1] - U[end - 1])]./(2.0 * h)
6     dU_dx = dU./(ones(length(U)) * h)
7
8     return real(
9         -1im * h * sum(
10            dU_dx .* conj(U) .- U .* conj(dU_dx)
11        )
12    )
13 end

```

## 3 Функция численного решения (CPU)

### 3.1 Процедура фильтрации

```

1 function filtration(
2     U::Union{Vector{ComplexF64}, Vector{Float64}},
3     h::Float64,
4     factor::Float64,

```

```

5      l_nominal::Float64,
6  )
7  delta = trunc(Int, l_nominal / h / 2) # полуширота в ячейках сетки
8  N = size(U)[1]
9  != 1 || throw(ArgumentError(
10    "Размерность вектора U равна 1. Рассмотрите transpose(U)."
11  ))
12
13  i_center = argmax(abs.(U))
14  i_left = i_center - delta
15  i_right = i_center + delta
16
17  i_left = i_left < 1 ? i_left + N : i_left
18  i_right = i_right > N ? i_right - N : i_right
19
20  I1 = integral_1(U, h)
21  I2 = integral_2(U, h)
22
23  if i_right < i_left
24    U[i_right:i_left] /= factor
25  else
26    U[1:i_left] /= factor
27    U[i_right:end] /= factor
28  end
29
30  return U, (I1 - integral_1(U, h), I2 - integral_2(U, h))
31 end

```

## 3.2 Основная функция

```

1  """
2   -2   1   0   0   1
3   1   -2   1   0   0
4   0   1   -2   1   0
5   0   0   1   -2   1
6   1   0   0   1   -2
7  """
8  function create_FD_matrix(n)
9    mat = zeros(Int, n, n)
10
11   for i = 1:n
12     mat[i, i] = -2
13     if i > 1
14       mat[i, i - 1] = 1
15     end
16     if i < n
17       mat[i, i + 1] = 1
18     end
19   end
20
21   mat[1, n] = 1
22   mat[n, 1] = 1
23
24   return mat
25 end
26 """
27 По начальному условию и заданным промежуткам решает начально-краевую
28 задачу Фурье или конечно-разностным методом.
29
30 Опционально:

```

```

31      - сравнивает численное решение с заданным аналитическим;
32      - вычисляет интегралы в процессе моделирования;
33      - реализует алгоритм фильтрации излучения;
34      - сохраняет решение в заданные моменты времени.
35  """
36  function solve(
37      tspan::Tuple{Real, Real},
38      xspan::Tuple{Real, Real},
39      tau::Real,
40      h::Real,
41      initial_function;
42      method::String = "fourier",
43      epsilon_2::Real = 0.0,
44      epsilon_3::Real = 0.0,
45      # filtration parameters
46      filtration_flag::Bool = false,
47      filtration_time::Real = 10.0,
48      filtration_factor::Union{Real, Function} = 1.0,
49      filtration_end_t::Real = tspan[2],
50      l_nominal::Real = 100.0,
51      # tolerance calculations
52      tolerance_flag = false,
53      analytical_solution = [],
54      # record integrals
55      integrals_flag = false,
56      # times of interest
57      capture_times = [],
58  )
59  theta = 0.5
60  L = xspan[2] - xspan[1]
61  T = tspan[2] - tspan[1]
62  N = Int(L / h)
63  N_x = N + 1
64  N_t = Int(round(T / tau + 1, digits = 1))
65
66  j = range(-N / 2, stop = N / 2, length = N_x)
67  x = collect(j .* h)[1:end - 1]
68  t = range(tspan[1], tspan[2], length = N_t)
69
70  if method == "fourier"
71      mun = collect(2 * pi / L .* range(-N / 2, stop = N / 2 - 1, length = N))
72      direct = (h / L .* exp.(-1im .* mun * x'))
73      inverse = exp.(1im .* mun * x')
74      fourier_ratio = exp.(-1im .* mun.^2 . * tau)
75      M = inverse * (fourier_ratio .* direct)
76  else # method == "finite_difference"
77      S = create_FD_matrix(N)
78      I = Diagonal(ones(N))
79      r = tau/h^2
80      M = (I - 1im * r * theta * S)^-1 * (I + 1im * r * (1 - theta) * S)
81  end
82
83  U = initial_function.(x)
84
85  (sum(abs.(U) .> abs(U[1])) * sum(abs.(U) .> abs(U[end]))) != 0 || 
86  throw(AssertionError("Начальный импульс за пределами x-интервала."))
87
88  epsilon_tail = 1e-5
89  (abs(U[1]) < epsilon_tail) & (abs(U[end]) < epsilon_tail) ||
90  @warn "Хвосты солитона превысили величину $epsilon_tail со значением \

```

```

91     $(max(abs(U[1]), abs(U[end]))). Попробуйте расширить промежуток по x."
92
93 if filtration_flag
94     t_slider = filtration_time
95     I1_dissipated = zeros(N_t)
96     I2_dissipated = zeros(N_t)
97 end
98 if tolerance_flag
99     tolerance = zeros(N_t)
100 end
101 if integrals_flag
102     I_1 = zeros(N_t)
103     I_2 = zeros(N_t)
104     I_1[1] = integral_1(U,h)
105     I_2[1] = integral_2(U,h)
106 end
107 capture_times_flag = false
108 if ~isempty(capture_times)
109     (any(capture_times .> tspan[2]) || any(capture_times .< tspan[1])) &&
110         @error "capture_times содержит элемент за пределами времен моделирования"
111     capture_times_flag = true
112     push!(capture_times, Inf)
113     capture_times_slider = 1
114     t_capture_entry = capture_times[capture_times_slider]
115     U_set = []
116 end
117 @showprogress for i in 1:N_t
118     if capture_times_flag && t_capture_entry! = Inf
119         t_current = (i - 1) * tau
120         if t_current >= t_capture_entry
121             push!(U_set, U)
122             capture_times_slider += 1
123             t_capture_entry = capture_times[capture_times_slider]
124         end
125     end
126     if tolerance_flag
127         if isa(analytical_solution, Real) # calculate the "pike" tolerance
128             tolerance[i] = (
129                 maximum(abs.(U)) - analytical_solution
130             ) / maximum(abs.(U)) * 100
131         else
132             # Percentage tolerance
133             tolerance[i] = maximum(
134                 abs.((abs.(analytical_solution.(x, (i - 1) * tau)) - abs.(U)))
135             ) / maximum(abs.(analytical_solution.(x, (i - 1) * tau))) * 100
136         end
137     end
138     if filtration_flag && (i * tau <= filtration_end_t)
139         if i * tau >= t_slider
140             t_slider += filtration_time
141             U, (power_I1, power_I2) = filtration(
142                 U,
143                 h,
144                 isa(filtration_factor, Function) ?
145                     filtration_factor(i * tau) : filtration_factor,
146                     l_nominal,
147             )
148             I1_dissipated[i] = power_I1
149             I2_dissipated[i] = power_I2
150         end

```

```

151     end
152
153     V = exp.(  

154         1im * tau* (  

155             (abs.(U)).^2 + epsilon_2 * (abs.(U)).^4 + epsilon_3 * (abs.(U)).^6  

156         )  

157     ). * U  

158     U = M * V
159
160     if integrals_flag  

161         I_1[i] = integral_1(U,h)  

162         I_2[i] = integral_2(U,h)
163     end
164 end
165 return(  

166     x,  

167     t,  

168     capture_times_flag ? U_set : U,  

169     filtration_flag ?  

170         (cumsum(I1_dissipated), cumsum(I2_dissipated))  

171         :  

172         (nothing, nothing),
173     tolerance_flag ? tolerance : nothing,  

174     integrals_flag ? (I_1, I_2) : (nothing, nothing),
175 )
176 end

```

## 4 Функция численного решения (GPU)

### 4.1 CUDA-версии вспомогательных функций

```

1 function cuda_calculate_V_and_multiplicate_kernel!(  

2     M, U, V, tau, epsilon_2, epsilon_3, y
3 )
4     i = threadIdx().x + (blockIdx().x - 1) * blockDim().x
5     if i <= length(U)
6         abs_U = abs(U[i])
7         phase_factor = exp(1im * tau * (
8             abs_U^2 + epsilon_2 * abs_U^4 + epsilon_3 * abs_U^6
9         ))
10        V[i] = phase_factor * U[i]
11    end
12    if i <= length(y)
13        sum = 0.0 + 0.0im
14        for j in axes(M, 2)
15            sum += M[i, j] * V[j]
16        end
17        y[i] = sum
18    end
19    return
20 end
21 function cuda_calculate_V_and_multiplicate(
22     M::CuArray{ComplexF64, 2},
23     U::CuArray{ComplexF64, 1},
24     tau::Float64,
25     epsilon_2::Float64,
26     epsilon_3::Float64,
27 )
28     m, _ = size(M)

```

```

29     V = CUDA.fill(0.0 + 0.0im, length(U))
30     y = CUDA.fill(0.0 + 0.0im, m)
31     @cuda threads = 256 blocks = ceil(Int, max(length(U), m)/256)
32         cuda_calculate_V_and_multiplicate_kernel!(M, U, V, tau, epsilon_2, epsilon_3, y)
33     return y
34 end
35 function cuda_maximum(U::CuArray{ComplexF64, 1})
36     abs_U = abs.(U)
37     return CUDA.reduce(max, abs_U)
38 end
39 function cuda_filtration(
40     U::Union{CuArray{ComplexF64, 1}, CuArray{Float64, 1}},
41     h::Float64,
42     factor::Float64,
43     l_nominal::Float64,
44 )
45     delta = trunc(Int, l_nominal / h / 2)
46     N = size(U, 1)
47     N != 1 || throw(BoundsError("Размерность вектора U равна 1. \
48             Рассмотрите transpose(U)."))
49
50     i_center = CUDA.argmax(abs.(U))
51     i_left = i_center - delta
52     i_right = i_center + delta
53
54     i_left = i_left < 1 ? i_left + N : i_left
55     i_right = i_right > N ? i_right - N : i_right
56
57     I1 = cuda_integral_1(U, h)
58     I2 = cuda_integral_2(U, h)
59
60     if i_right < i_left
61         U[i_right:i_left] .= U[i_right:i_left] ./ factor
62     else
63         U[1:i_left] .= U[1:i_left] ./ factor
64         U[i_right:end] .= U[i_right:end] ./ factor
65     end
66
67     return U, (I1 - cuda_integral_1(U, h), I2 - cuda_integral_2(U, h))
68 end
69 function derivative_kernel!(U, h, dU)
70     i = threadIdx().x + (blockIdx().x - 1) * blockDim().x
71     if i <= length(U)
72         if i == 1
73             dU[i] = (U[2] - U[end]) / (2.0 * h)
74         elseif i == length(U)
75             dU[i] = (U[1] - U[end - 1]) / (2.0 * h)
76         else
77             dU[i] = (U[i + 1] - U[i - 1]) / (2.0 * h)
78         end
79     end
80     return
81 end
82 function integral_1_kernel!(U, h, result)
83     i = threadIdx().x + (blockIdx().x - 1) * blockDim().x
84     if i <= length(U)
85         result[i] = h * abs(U[i])^2
86     end
87     return
88 end

```

```

89  function integral_2_kernel!(U, h, dU_dx, result)
90      i = threadIdx().x + (blockIdx().x - 1) * blockDim().x
91      if i <= length(U)
92          dU_dx[i] = dU_dx[i] / h
93          result[i] = real(-1im * h * (dU_dx[i] * conj(U[i]) - U[i] * conj(dU_dx[i])))
94      end
95      return
96  end
97  function cuda_integral_1(U::CuArray{ComplexF64}, h::Float64)
98      result = CUDA.fill(0.0, length(U))
99
100     @cuda threads = 256 blocks = ceil(Int, length(U)/256) integral_1_kernel!(U, h, result)
101
102     return sum(result)
103 end
104 function cuda_integral_2(U::CuArray{ComplexF64}, h::Float64)
105     dU = CUDA.fill(0.0 + 0.0im, length(U))
106     dU_dx = CUDA.fill(0.0 + 0.0im, length(U))
107     result = CUDA.fill(0.0, length(U))
108
109     @cuda threads = 256 blocks = ceil(Int, length(U)/256)
110         derivative_kernel!(U, h, dU)
111     @cuda threads = 256 blocks = ceil(Int, length(U)/256)
112         integral_2_kernel!(U, h, dU, result)
113
114     return sum(result)
115 end

```

## 4.2 Основная функция

```

1  function prepare_filtration_args(filtration_args, N_t)
2      @unpack filtration_flag, filtration_tau, filtration_factor,
3          filtration_end_t, filtration_l_nominal = filtration_args
4      return(
5          filtration_flag ? filtration_tau : nothing,
6          filtration_flag ? filtration_factor : nothing,
7          filtration_flag ? filtration_end_t : nothing,
8          filtration_flag ? [filtration_tau] : nothing,
9          filtration_flag ? filtration_l_nominal : nothing,
10         filtration_flag ? zeros(N_t) : nothing,
11         filtration_flag ? zeros(N_t) : nothing,
12     )
13 end
14 function prepare_capture_times_args(capture_times, tspan, N)
15     (any(capture_times.>tspan[2]) || any(capture_times.<tspan[1])) &&
16         @error "capture_times содержит элемент за пределами времен моделирования"
17     capture_times_flag = ~isempty(capture_times)
18     return(
19         capture_times_flag,
20         capture_times_flag ?
21             [Vector{ComplexF64}(undef, N) for _ in 1:length(capture_times)] : nothing,
22         capture_times_flag ? [1] : nothing,
23         capture_times_flag ? [capture_times..., Inf] : nothing,
24         capture_times_flag ? [capture_times[1]] : nothing,
25     )
26 end
27 function cuda_cycle_iteration!(
28     i::Int,
29     h::Real,
30     tau::Real,

```

```

31     cuda_M::CuArray{ComplexF64, 2, CUDA.Mem.DeviceBuffer},
32     cuda_U::CuArray{ComplexF64, 1, CUDA.Mem.DeviceBuffer},
33     epsilon_2::Real,
34     epsilon_3::Real,
35     # Return solution at times given
36     capture_times_flag::Bool,
37     U_set::Union{Vector{Vector{ComplexF64}}, Nothing},
38     capture_times_index_slider::Union{Vector{Int}, Nothing},
39     capture_times::Union{Vector{<:Real}, Nothing},
40     t_capture_entry::Union{Vector{<:Real}, Nothing},
41     # Return integrals
42     integrals_flag::Bool,
43     I_1::Union{Vector{<:Real}, Nothing},
44     I_2::Union{Vector{<:Real}, Nothing},
45     # Return pulse maximum
46     pulse_maximum_flag::Bool,
47     pulse_maximum::Union{Vector{<:Real}, Nothing},
48     filtration_flag::Union{Bool, Nothing},
49     filtration_tau::Union{Real, Nothing},
50     filtration_factor::Union{Real, Function, Nothing},
51     filtration_end_t::Union{Real, Nothing},
52     filtration_t_slider::Union{Vector{<:Real}, Nothing},
53     filtration_l_nominal::Union{Real, Nothing},
54     I1_dissipated::Union{Vector{<:Real}, Nothing},
55     I2_dissipated::Union{Vector{<:Real}, Nothing},
56     # Live plot
57     live_plot_solution_flag::Bool,
58     live_plot_tau::Real,
59     live_plot_t_slider::Union{Vector{<:Real}, Nothing},
60     live_t_observable::Union{Nothing, Observable{Float64}},
61     observable_U::Union{Nothing, Observable{T}},
62     io,
63 ) where T <: Vector{<:Real}
64     t_current = (i - 1) * tau
65     if live_plot_solution_flag && (t_current ? live_plot_t_slider[1])
66         observable_U[] = abs.(Array(cuda_U))
67         live_t_observable[] = t_current
68         recordframe!(io)
69         live_plot_t_slider[1] += live_plot_tau
70     end
71     if capture_times_flag && t_capture_entry[1] ? Inf
72         if t_current ? t_capture_entry[1]
73             U_set[capture_times_index_slider[1]] = Array(cuda_U)
74             capture_times_index_slider[1] += 1
75             t_capture_entry[1] = capture_times[capture_times_index_slider[1]]
76         end
77     end
78     if pulse_maximum_flag
79         pulse_maximum[i] = cuda_maximum(cuda_U)
80     end
81     if filtration_flag && (t_current <= filtration_end_t)
82         if t_current ? filtration_t_slider[1]
83             filtration_t_slider[1] += filtration_tau
84             cuda_U, (power_I1, power_I2) = cuda_filtration(
85                 cuda_U,
86                 h,
87                 isa(filtration_factor, Function) ?
88                     filtration_factor(t_current) : filtration_factor,
89                     filtration_l_nominal,
90             )

```

```

91         I1_dissipated[i] = power_I1
92         I2_dissipated[i] = power_I2
93     end
94 end
95 if integrals_flag
96     I_1[i] = cuda_integral_1(cuda_U,h)
97     I_2[i] = cuda_integral_2(cuda_U,h)
98 end
99 cuda_U[1:end] = cuda_calculate_V_and_multiplicate(
100     cuda_M, cuda_U, tau, epsilon_2, epsilon_3
101 )
102 end
103 """
104 По начальному условию и заданным промежуткам решает начально-краевую задачу Фурье или
105 конечно-разностным методом. Реализация на CUDA.
106
107 Опционально:
108     - вычисляет интегралы в процессе моделирования;
109     - реализует алгоритм фильтрации излучения;
110     - сохраняет решение в заданные моменты времени;
111     - сохраняет зависимость максимума решения от шага;
112     - выводит и/или сохраняет результат в формате mp4.
113 """
114 function cuda_solve(
115     tspan::Tuple{Real, Real},
116     xspan::Tuple{Real, Real},
117     tau::Real,
118     h::Real,
119     initial_function;
120     method::String = "fourier",
121     epsilon_2::Real = 0.0,
122     epsilon_3::Real = 0.0,
123     # filtration parameters
124     filtration_args::NamedTuple = (
125         filtration_flag = false,
126         filtration_tau = 10.0,
127         filtration_factor = 1.0,
128         filtration_end_t = tspan[2],
129         filtration_l_nominal = 100.0,
130     ),
131     # pulse_maximum calculations
132     pulse_maximum_flag = false,
133     # record integrals
134     integrals_flag = false,
135     # times of interest
136     capture_times = [],
137     # live plot
138     live_plot_solution_flag = false,
139     live_plot_tau = 1.0,
140     # omit progress bar
141     omit_progress_flag = false,
142 )
143     @unpack filtration_flag, filtration_tau, filtration_factor,
144         filtration_end_t, filtration_l_nominal = filtration_args
145     theta = 0.5
146     L = xspan[2] - xspan[1]
147     T = tspan[2] - tspan[1]
148     N = Int(L / h)
149     N_x = N + 1
150     N_t = Int(round(T / tau + 1, digits = 1))

```

```

151
152     j = range(-N / 2, stop = N / 2 , length = N_x)
153     x = collect(j .* h)[1:end - 1]
154     t = range(tspan[1], tspan[2], length = N_t)
155
156     if method == "fourier"
157         mun = collect(2 * pi / L .* range(-N / 2, stop = N / 2 - 1, length = N))
158         direct = (h / L .* exp.(-1im .* mun * x'))
159         inverse = exp.(1im .* mun * x')
160         fourier_ratio = exp.(-1im.* mun.^2 . * tau)
161         M = inverse * (fourier_ratio .* direct)
162     else # method == "finite_difference"
163         S = create_FD_matrix(N)
164         I = Diagonal(ones(N))
165         r = tau/h^2
166         M = (I - 1im * r * theta * S)^-1 * (I + 1im * r * (1 - theta) * S)
167     end
168
169     U = initial_function.(x)
170
171     (sum(abs.(U) .> abs(U[1])) * sum(abs.(U) .> abs(U[end])) != 0) ||
172         @warn "Начальный импульс за пределами x-интервала."
173
174     epsilon_tail = 1e-5
175     (abs(U[1]) < epsilon_tail) & (abs(U[end]) < epsilon_tail) ||
176     @warn "Хвосты солитона превысили величину $epsilon_tail со значением \
177     $(max(abs(U[1]), abs(U[end]))). Попробуйте расширить промежуток по x."
178
179     cuda_U = CuArray(U)
180     cuda_M = CuArray(M)
181
182     filtration_tau, filtration_factor, filtration_end_t, filtration_t_slider,
183         filtration_l_nominal, I1_dissipated, I2_dissipated =
184         prepare_filtration_args(filtration_args, N_t)
185
186     capture_times_flag, U_set, capture_times_index_slider, capture_times,
187         t_capture_entry = prepare_capture_times_args(capture_times, tspan, N)
188
189     pulse_maximum = pulse_maximum_flag ? zeros(N_t) : nothing
190
191     I_1 = I_2 = integrals_flag ? zeros(N_t) : nothing
192
193     if live_plot_solution_flag
194         observable_U = Observable(abs.(U))
195         observable_x = Observable(x)
196         live_plot_t_slider = [0.0]
197         live_t_observable = Observable(live_plot_t_slider[1])
198         live_plot = Figure()
199         live_axis = Axis(
200             live_plot[1, 1], limits = (minimum(x), maximum(x), 0.0, 1.0)
201         )
202         lines!(live_axis, observable_x, observable_U)
203         display(live_plot)
204     end
205
206     iteration_args = (
207         h,
208         tau,
209         cuda_M,
210         cuda_U,

```

```

211     epsilon_2,
212     epsilon_3,
213     capture_times_flag,
214     U_set,
215     capture_times_index_slider,
216     capture_times,
217     t_capture_entry,
218     integrals_flag,
219     I_1,
220     I_2,
221     pulse_maximum_flag,
222     pulse_maximum,
223     filtration_flag,
224     filtration_tau,
225     filtration_factor,
226     filtration_end_t,
227     filtration_t_slider,
228     filtration_l_nominal,
229     I1_dissipated,
230     I2_dissipated,
231     live_plot_solution_flag,
232     live_plot_tau,
233     live_plot_solution_flag ? live_plot_t_slider : [0.0],
234     live_plot_solution_flag ? live_t_observable : nothing,
235     live_plot_solution_flag ? observable_U : nothing,
236 )
237
238 if live_plot_solution_flag
239     GLMakie.record(live_plot, "output.mp4", framerate = 30) do io
240     if omit_progress_flag
241         for i in 1:N_t
242             cuda_cycle_iteration!(i, iteration_args..., io)
243         end
244     else
245         @showprogress for i in 1:N_t
246             cuda_cycle_iteration!(i, iteration_args..., io)
247         end
248     end
249     end
250 else
251     io = nothing
252     if omit_progress_flag
253         for i in 1:N_t
254             cuda_cycle_iteration!(i, iteration_args..., io)
255         end
256     else
257         @showprogress for i in 1:N_t
258             cuda_cycle_iteration!(i, iteration_args..., io)
259         end
260     end
261 end
262
263 return(
264     x,
265     t,
266     capture_times_flag ? U_set : Array(cuda_U),
267     filtration_flag ?
268         (cumsum(I1_dissipated), cumsum(I2_dissipated))
269         :
270         (zeros(N_t), zeros(N_t)),

```

```

271     pulse_maximum_flag ? pulse_maximum : nothing,
272     integrals_flag ? (I_1, I_2) : (nothing, nothing),
273   )
274 end

```

## 5 Функции для подбора аналитического решения по численному

### 5.1 Вспомогательные функции

```

1  function find_threshold(v::Vector{Float64}, N::Int)
2    sorted_v = sort(v, rev = true)
3    N <= length(v) ||
4      throw(ArgumentError("N = $N не может быть больше длины вектора $(length(v))"))
5    threshold_value = sorted_v[N]
6    max_value = maximum(v)
7    return max_value - threshold_value
8  end
9  function shift_pulse_to_center(
10    x::Vector{Float64},
11    y::Vector{Float64};
12    y_threshold::Float64 = 0.0
13  )
14    length(x) == length(y) ||
15      throw(ArgumentError("Вектора x и y должны иметь одинаковую размерность"))
16    max_index = argmax(y)
17    n_circ_shift = -max_index + cld(length(x), 2) + 1
18    y_centered = circshift(y, n_circ_shift)
19    filter_indices = findall(y_centered .? y_threshold)
20    x_filtered = x[filter_indices]
21    y_filtered = y_centered[filter_indices]
22    return x_filtered, y_filtered, n_circ_shift
23  end
24  function M0_M1_to_epsilon_2_epsilon_3(M0::Real, M1::Real)
25    epsilon_2 = 3.0 / (4.0 * M0) * (M1 / (M1 + 6 * M0) - 2.0)
26    epsilon_3 = 4.0 / (M0 * (M1 + 6 * M0))
27    return (epsilon_2, epsilon_3)
28  end
29  function epsilon_2_epsilon_3_to_M0_M1(epsilon_2::Real, epsilon_3::Real)
30    M0_a = (-4 * epsilon_2 - sqrt(2) * sqrt(8 * epsilon_2^2 - 27 * epsilon_3)) /
31      (9 * epsilon_3)
32    M1_a = (4 * sqrt(2) * sqrt(8 * epsilon_2^2 - 27 * epsilon_3)) / (3 * epsilon_3)
33
34    M0_b = (-4 * epsilon_2 + sqrt(2) * sqrt(8 * epsilon_2^2 - 27 * epsilon_3)) /
35      (9 * epsilon_3)
36    M1_b = -(4 * sqrt(2) * sqrt(8 * epsilon_2^2 - 27 * epsilon_3)) / (3 * epsilon_3)
37
38    (M0, M1) = M1_a > 0.0 ? (M0_a, M1_a) : (M0_b, M1_b)
39    return (M0, M1)
40  end

```

### 5.2 Основные функции

```

1  function construct_approximate_NSE_3_5_solution(
2    x_grid::Vector{<:Real},
3    h::Real,
4    U::Union{Vector{ComplexF64}, Vector{<:Real}},
5    epsilon_2::Real,

```

```

6      L::Real;
7      debug_flag = false,
8  )
9      abs_U = abs.(U)
10     debug_flag && println("Индекс максимума исходного решения: $(argmax(abs_U))")
11
12     # Точное определение максимума (снижение сеточной погрешности
13     # B-Сплайновой интерполяцией)
14     N_interpolation_points = 12
15     bottom_gap = find_threshold(abs_U, N_interpolation_points)
16     x_reduced, y_reduced, n_circ_shift = shift_pulse_to_center(
17         x_grid, abs_U; y_threshold = maximum(abs_U) - bottom_gap
18     )
19
20     debug_flag && println("Для центрирования импульс будет сдвинут \
21         на $n_circ_shift узлов")
22
23     x_range = round(x_reduced[1], digits = 10):h:round(x_reduced[end], digits = 10)
24     spline_interpolator = cubic_spline_interpolation((x_range), y_reduced)
25     minimization_function = xi -> -spline_interpolator(xi)
26     result = optimize(minimization_function, minimum(x_reduced), maximum(x_reduced))
27     x_max, y_max = result.minimizer, -result.minimum
28
29     debug_flag && println("Солитон передвинут. Считается, что его \
30         интерполированный максимум находится в точке x = $x_max")
31     debug_flag && println("Интерполированный максимум = $y_max")
32
33     k = 0.0
34     possible_? = 2 / 3 * (2 * epsilon_2 * y_max^4 + 3 * y_max^2)
35     _possible_? = possible_? / 4
36     possible_? = _possible_? + k^2
37
38     shifted_grid_solution_3_5 = abs.(NSE_3_5_soliton.
39         (x_grid, 0.0, k, possible_?, epsilon_2, 0.0, 0.0)
40     )
41
42     # x-coordinate correction
43     bottom_gap_shifted = find_threshold(
44         shifted_grid_solution_3_5, N_interpolation_points
45     )
46     x_reduced, y_reduced, _ = shift_pulse_to_center(
47         x_grid,
48         shifted_grid_solution_3_5;
49         y_threshold = maximum(shifted_grid_solution_3_5) - bottom_gap_shifted
50     )
51     x_range = round(x_reduced[1], digits = 10):h:round(x_reduced[end], digits = 10)
52     spline_interpolator = cubic_spline_interpolation((x_range), y_reduced)
53     minimization_function = xi -> -spline_interpolator(xi)
54     result = optimize(minimization_function, minimum(x_reduced), maximum(x_reduced))
55     x_max_shifted, _ = result.minimizer, -result.minimum
56     _z_0 = -(x_max_shifted - x_max)
57
58     debug_flag && println("Солитон построен, но теперь его интерполированный \
59         максимум находится в точке x = $x_max_shifted. Переносим на $_z_0")
60
61     possible_solution_3_5 = x -> NSE_3_5_soliton.(
62         x, 0.0, k, possible_?, epsilon_2, 0.0, _z_0
63     )
64     deshifted_grid_solution_3_5 = abs.(possible_solution_3_5.(x_grid))
65     if debug_flag

```

```

66     # Проверяем что передвинули куда надо
67     bottom_gap_deshifted = find_threshold(
68         deshifted_grid_solution_3_5, N_interpolation_points
69     )
70     x_reduced, y_reduced, _ = shift_pulse_to_center(
71         x_grid,
72         deshifted_grid_solution_3_5;
73         y_threshold = maximum(deshifted_grid_solution_3_5) - bottom_gap_deshifted
74     )
75     x_range = round(x_reduced[1], digits = 10):h:round(x_reduced[end], digits = 10)
76     spline_interpolator = cubic_spline_interpolation((x_range), y_reduced)
77     minimization_function = xi -> -spline_interpolator(xi)
78     result = optimize(
79         minimization_function, minimum(x_reduced), maximum(x_reduced)
80     )
81     x_max_deshifted, _ = result.minimizer, -result.minimum
82     println("Солитон перестроен, и теперь его интерполированный максимум \
83     находится в точке x = $x_max_deshifted")
84 end
85 return circshift(
86     possible_solution_3_5.(x_grid),
87     argmax(abs_U) - argmax(deshifted_grid_solution_3_5)
88 )
89 end
90 function construct_approximate_NSE_3_5_7_solution(
91     x_grid::Vector{<:Real},
92     h::Real,
93     abs_U::Vector{<:Real},
94     epsilon_2::Real,
95     L::Real;
96     use_M_values = false,
97     M0::Real = 0.0,
98     use_M1_value = false,
99     M1::Real = 0.0,
100    debug_flag = false,
101    warn_ignore = false,
102 )
103
104     debug_flag && println("Индекс максимума исходного решения: $(argmax(abs_U))")
105     _, y_shifted, n_circ_shift = shift_pulse_to_center(x_grid, abs_U)
106     debug_flag && println("Для центрирования импульс будет сдвинут\
107     на $n_circ_shift узлов")
108
109     # Точное определение максимума (снижение сеточной погрешности
110     # B-Сплайновой интерполяцией)
111     N_interpolation_points = 12
112     bottom_gap = find_threshold(abs_U, N_interpolation_points)
113     x_reduced, y_reduced, _ = shift_pulse_to_center(
114         x_grid, abs_U; y_threshold = maximum(abs_U) - bottom_gap
115     )
116     x_range = round(x_reduced[1], digits = 10):h:round(x_reduced[end], digits = 10)
117     spline_interpolator = cubic_spline_interpolation((x_range), y_reduced)
118     minimization_function = xi -> -spline_interpolator(xi)
119     result = optimize(minimization_function, minimum(x_reduced), maximum(x_reduced))
120     x_max, y_max = result.minimizer, -result.minimum
121     debug_flag && println("Солитон передвинут. Считается, что его \
122     интерполированный максимум находится в точке x = $x_max")
123     debug_flag && println("Интерполированный максимум = $y_max")
124     if use_M_values
125         if ~use_M1_value

```

```

126      M1 = 4 * (y_max^2 - MO)
127  end
128  debug_flag && println("M1 = $M1")
129  _epsilon_2, _epsilon? = 0.0, 0.0
130 else
131  D = 2 * sqrt(9 + 4 * epsilon_2^2 * y_max^4 - 6 * y_max^2 * epsilon_2)
132  b = -6 - 4 * epsilon_2 * y_max^2
133  MO = (b - D) / (4 * epsilon_2)
134  M1 = 4 * (y_max^2 - MO)
135  _epsilon_2, _epsilon? = MO_M1_to_epsilon2_epsilon3(MO, M1)
136  debug_flag && println("Параметры подобраны для epsilon_2 = ",
137  _epsilon_2, ", epsilon? = ", _epsilon?)
138 end
139
140 interpolator = precompile_NSE_3_5_7_soliton(
141  _epsilon_2, _epsilon?, 0.0, 0.0, L;
142  use_M_values = use_M_values, M_0 = MO, M_1 = M1, warn_ignore = warn_ignore
143 )
144 _possible_solution_3_5_7 = (x) -> NSE_3_5_7_soliton(
145  x, 0.0, 0.0, 0.0, 0.0, 0.0, interpolator
146 )
147
148 # x-coordinate correction
149 shifted_grid_solution_3_5_7 = abs._possible_solution_3_5_7.(x_grid)
150 bottom_gap_shifted = find_threshold(
151  shifted_grid_solution_3_5_7, N_interpolation_points
152 )
153 x_reduced, y_reduced, _ = shift_pulse_to_center(
154  x_grid,
155  shifted_grid_solution_3_5_7;
156  y_threshold = maximum(shifted_grid_solution_3_5_7) - bottom_gap_shifted
157 )
158 x_range = round(x_reduced[1], digits = 10):h:round(x_reduced[end], digits = 10)
159 spline_interpolator = cubic_spline_interpolation((x_range), y_reduced)
160 minimization_function = xi -> -spline_interpolator(xi)
161 result = optimize(minimization_function, minimum(x_reduced), maximum(x_reduced))
162 x_max_shifted, _ = result.minimizer, -result.minimum
163 z_0 = -(x_max_shifted - x_max)
164
165 debug_flag && println("Солитон построен, но теперь его интерполированный\
166 максимум находится в точке x = $x_max_shifted. Переносим на $z_0")
167
168 interpolator = precompile_NSE_3_5_7_soliton(
169  _epsilon_2, _epsilon?, 0.0, 0.0, L;
170  use_M_values = use_M_values, M_0 = MO, M_1 = M1, warn_ignore = warn_ignore
171 )
172 possible_solution_3_5_7 = (x) -> NSE_3_5_7_soliton(
173  x, 0.0, 0.0, 0.0, 0.0, z_0, interpolator
174 )
175 deshifted_grid_solution_3_5_7 = abs.(possible_solution_3_5_7.(x_grid))
176
177 if debug_flag
178  # Проверяем что передвинули куда надо
179  bottom_gap_deshifted = find_threshold(
180  deshifted_grid_solution_3_5_7,
181  N_interpolation_points
182 )
183  x_reduced, y_reduced, _ = shift_pulse_to_center(
184  x_grid,
185  deshifted_grid_solution_3_5_7;

```

```

186         y_threshold = maximum(deshifted_grid_solution_3_5_7) - bottom_gap_deshifted
187     )
188     x_range = round(x_reduced[1], digits = 10):h:round(x_reduced[end], digits = 10)
189     spline_interpolator = cubic_spline_interpolation((x_range), y_reduced)
190     minimization_function = xi -> -spline_interpolator(xi)
191     result = optimize(minimization_function, minimum(x_reduced), maximum(x_reduced))
192     x_max_deshifted, _ = result.minimizer, -result.minimum
193     println("Солитон перестроен, и теперь его интерполированный максимум\
194             находится в точке x = $x_max_deshifted")
195 end
196
197 return circshift(
198     possible_solution_3_5_7.(x_grid),
199     argmax(abs_U) - argmax(deshifted_grid_solution_3_5_7)
200 )
201 #-argmax(deshifted_grid_solution_3_5_7)
202 end
203 function argmin_observed_change(old_index, vector; debug_flag = false)
204     new_index = argmin(vector)
205     debug_flag && println("был индекс $old_index, стал индекс $new_index")
206     return new_index, old_index == new_index
207 end
208 # Улучшенная по точности, но более медленная функция подбора аналитического решения.
209 function construct_best_approximate_NSE_3_5_7_solution(
210     x_grid::Vector{<:Real},
211     h::Real,
212     abs_U::Vector{<:Real},
213     L::Real,
214     MO_initial;
215     MO_gap = 1.0,
216     MO_h = 0.5,
217     debug_flag = false,
218 )
219     y_max = maximum(abs_U)
220     MO_vector_length = div(MO_gap, MO_h)
221     construction_function = (MO, M1) -> abs.(construct_approximate_NSE_3_5_7_solution(
222         x_grid, h, abs_U, 0.0, L;
223         use_M_values = true, MO = MO, use_M1_value = true, M1 = M1, warn_ignore = true,
224     ))
225     error_function = soliton -> relative_error_to_amplitude(abs_U, soliton)
226     middle_index = Int(div(MO_gap, MO_h) + 1)
227
228     error_argmin = 0
229     solutions_with_same_amplitude = Vector{Float64}([])
230
231     for _ in 1:6 # Целевое количество шагов дихотомии
232         stop_flag = false
233         while ~stop_flag
234             MO_vector = round.(collect(
235                 MO_initial - MO_vector_length*MO_h:MO_h:MO_initial + MO_vector_length * MO_h
236             ), digits = 10)
237
238             debug_flag && println("Массив поиска: ", MO_vector)
239
240             MO_vector = filter(x -> x < 0, MO_vector)
241             M1_vector = 4 .* (y_max^2 .- MO_vector)
242             solutions_with_same_amplitude = construction_function.(
243                 MO_vector, M1_vector
244             )
245             errors = error_function.(solutions_with_same_amplitude)

```

```

246     error_argmin, stop_flag = argmin_observed_change(
247         middle_index, errors; debug_flag
248     )
249
250     debug_flag && println(errors)
251     stop_flag || (
252         MO_initial = MO_vector[error_argmin];
253         debug_flag && println("Останавливается рано. Перемещаемся на $MO_initial")
254     )
255     stop_flag && (
256         MO_initial = MO_vector[error_argmin];
257         MO_h /= MO_vector_length;
258         debug_flag && println("Достигли равновесия, уменьшаем шаг.\\"/>
259             Теперь MO_initial = $MO_initial, MO_h = $MO_h")
260     )
261     end
262 end
263 return solutions_with_same_amplitude[error_argmin]
264 end

```

# Литература

- [1] J. A. C. Weideman and B. M. Herbst. Split-step methods for the solution of the nonlinear Schrödinger equation. *SIAM Journal on Numerical Analysis*, 23:485–507, 6 1986.
- [2] N.A. Kudryashov. Highly dispersive solitary wave solutions of perturbed nonlinear Schrödinger equations. *Applied Mathematics and Computation*, 371, 4 2020.
- [3] Akira Hasegawa and Frederick Tappert. Transmission of stationary nonlinear optical pulses in dispersive dielectric fibers. I. Anomalous dispersion. *Applied Physics Letters*, 23:142–144, 8 1973.
- [4] Akira Hasegawa and Frederick Tappert. Transmission of stationary nonlinear optical pulses in dispersive dielectric fibers. II. Normal dispersion. *Applied Physics Letters*, 23:171–172, 8 1973.
- [5] Govind P. Agrawal. *Nonlinear Fiber Optics*. Springer Berlin Heidelberg, 2000.
- [6] Yuri Kivshar and Govind Agrawal. *Optical Solitons: From Fibers to Photonic Crystals*. Academic Press, 2 2003.
- [7] A.C. Newell. *Solitons in Mathematics and Physics*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1985.
- [8] Igor S. Aranson and Lorenz Kramer. The world of the complex ginzburg-landau equation. *Rev. Mod. Phys.*, 74:99–143, Feb 2002.
- [9] M Ablowitz and H Sigur. *Solitary Waves and the Inverse Problem Method [Russian translation]*. Mir, Moscow, 1982.
- [10] Yuri Kivshar. Dynamics of solitons in nearly integrable systems. *Reviews of Modern Physics - REV MOD PHYS*, 61:763–915, 2 1989.
- [11] Anjan Biswas, Mehmet Ekici, Abdullah Sonmezoglu, and Ali Saleh Alshomrani. Highly dispersive optical solitons in absence of self-phase modulation by F-expansion. *Optik*, 187:258–271, 6 2019.
- [12] Nikolay A. Kudryashov. General solution of the traveling wave reduction for the perturbed Chen-Lee-Liu equation. *Optik*, 186:339–349, 6 2019.

- [13] Hamood Ur Rehman, Naeem Ullah, and M.A. Imran. Highly dispersive optical solitons using Kudryashov's method. *Optik*, 199:163349, 12 2019.
- [14] Russell W. Kohl, Anjan Biswas, Mehmet Ekici, Qin Zhou, Salam Khan, Ali S. Alshomrani, and Milivoj R. Belic. Highly dispersive optical soliton perturbation with Kerr law by semi-inverse variational principle. *Optik*, 199:163226, 12 2019.
- [15] Anjan Biswas, Mehmet Ekici, Abdullah Sonmezoglu, and Milivoj R. Belic. Highly dispersive optical solitons in absence of self-phase modulation by Jacobi's elliptic function expansion. *Optik*, 189:109–120, 7 2019.
- [16] Ivan M. Uzunov, Todor N. Arabadzhiev, and Svetoslav G. Nikolov. Self-steepening and intrapulse Raman scattering in the presence of nonlinear gain and its saturation. *Optik*, 271:170137, 2022.
- [17] Chong Wang, Ying Wang, Shenghan Wang, Chenglin Sun, and Zhiwei Men. Cascaded amplification via three-beam double stimulated Raman scattering in benzene. *Journal of Molecular Liquids*, 368:120667, 2022.
- [18] E.M. Gromov and B.A. Malomed. Damped solitons in an extended nonlinear Schrödinger equation with a spatial stimulated Raman scattering and decreasing dispersion. *Optics Communications*, 320:88–93, 2014.
- [19] Nikolay A. Kudryashov. On traveling wave solutions of the Kundu-Eckhaus equation. *Optik*, 224:165500, 12 2020.
- [20] Russell W. Kohl, Anjan Biswas, Mehmet Ekici, Qin Zhou, Salam Khan, Ali S. Alshomrani, and Milivoj R. Belic. Highly dispersive optical soliton perturbation with cubic-quintic-septic refractive index by semi-inverse variational principle. *Optik*, 199:163322, 12 2019.
- [21] Nikolay A. Kudryashov. Construction of nonlinear differential equations for description of propagation pulses in optical fiber. *Optik*, 192:162964, 9 2019.
- [22] Nikolay A. Kudryashov. Solitary and periodic waves of the hierarchy for propagation pulse in optical fiber. *Optik*, 194:163060, 10 2019.
- [23] Anjan Biswas, Mehmet Ekici, Abdullah Sonmezoglu, and Milivoj R. Belic. Highly dispersive optical solitons with non-local nonlinearity by exp-function. *Optik*, 186:288–292, 6 2019.
- [24] Nikolay A. Kudryashov and Dariya V. Safonova. Painleve analysis and traveling wave solutions of the sixth order differential equation with non-local nonlinearity. *Optik*, 244:167586, 10 2021.

- [25] Nikolay A. Kudryashov. General solution of traveling wave reduction for the Kundu-Mukherjee-Naskar model. *Optik*, 186:22–27, 6 2019.
- [26] AI Maimistov, AM Basharov, AI Maimistov, and AM Basharov. Optical solitons in fibers. *Nonlinear Optical Waves*, pages 303–435, 1999.
- [27] A.T. Filippov. *Many-sided Soliton*. Nauka, Moscow, 1990.
- [28] Bruce M Lake, Henry C Yuen, Harald Rungaldier, and Warren E Ferguson. Nonlinear deep-water waves: theory and experiment. Part 2. Evolution of a continuous wave train. *Journal of Fluid Mechanics*, 83:49–74, 1977.
- [29] Henry C. Yuen and Warren E. Ferguson. Relationship between Benjamin-Feir instability and recurrence in the nonlinear Schrödinger equation. *Physics of Fluids*, 21:1275, 1978.
- [30] E. Fermi, P Pasta, S Ulam, and M Tsingou. Studies of nonlinear problems I. Los Alamos National Laboratory (LANL), 5 1955.
- [31] Ronald H Hardin and F. D. Tappert. Application of the split-step Fourier method to the numerical solution of nonlinear and variable coefficient wave equations. *Siam Review*, 15:423, 1973.
- [32] J.M Sanz-Serna. An explicit finite-difference scheme with exact conservation properties. *Journal of Computational Physics*, 47:199–210, 8 1982.
- [33] J M Sanz-Serna and V S Manoranjan. A method for the integration in time of certain partial differential equations. *Journal of Computational Physics*, 52:273–289, 1983.
- [34] M Delfour, M Fortin, and G Payr. Finite-difference solutions of a non-linear Schrödinger equation. *Journal of Computational Physics*, 44:277–288, 12 1981.
- [35] D.F. Griffiths, A.R. Mitchell, and J.Li. Morris. A numerical study of the nonlinear Schrödinger equation. *Computer Methods in Applied Mechanics and Engineering*, 45:177–215, 9 1984.
- [36] J. M. Sanz-Serna. Methods for the numerical solution of the nonlinear Schrödinger equation. *Mathematics of Computation*, 43:21–27, 1984.
- [37] A Mitchell and J Morris. A self adaptive difference scheme for the nonlinear Schrödinger equation. *Arab Gulf Journal of Scientific Research*, pages 461–472, 1983.
- [38] Hongyu Qin, Fengyan Wu, and Deng Ding. A linearized compact ADI numerical method for the two-dimensional nonlinear delayed Schrödinger equation. *Applied Mathematics and Computation*, 412:126580, 2022.

- [39] Nikolay A. Kudryashov. Method for finding optical solitons of generalized nonlinear Schrödinger equations. *Optik*, 261:169163, 7 2022.
- [40] Nikolai A. Kudryashov. Simplest equation method to look for exact solutions of nonlinear differential equations. *Chaos, Solitons & Fractals*, 24:1217–1231, 6 2005.
- [41] Nikolay A. Kudryashov and Daniil R. Nifontov. Conservation laws and hamiltonians of the mathematical model with unrestricted dispersion and polynomial nonlinearity. *Chaos, Solitons & Fractals*, 175:114076, 10 2023.