

УЧРЕЖДЕНИЕ РОССИЙСКОЙ АКАДЕМИИ НАУК  
ИНСТИТУТ СИСТЕМ ОБРАБОТКИ ИЗОБРАЖЕНИЙ РАН

САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ  
УНИВЕРСИТЕТ

имени академика С. П. КОРОЛЁВА  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

*Н.Л. Казанский, П.Г. Серафимович, С.Н. Хонина*

**ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ  
ВЫЧИСЛЕНИЯ  
В ДИФРАКЦИОННОЙ НАНООПТИКЕ**

*Учебное пособие*

САМАРА  
2010

УЧРЕЖДЕНИЕ РОССИЙСКОЙ АКАДЕМИИ НАУК  
ИНСТИТУТ СИСТЕМ ОБРАБОТКИ ИЗОБРАЖЕНИЙ РАН

САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ  
имени академика С. П. КОРОЛЁВА  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**Н.Л. Казанский, П.Г. Серафимович, С.Н. Хонина**

**ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ  
В ДИФРАКЦИОННОЙ НАНООПТИКЕ**

*Учебное пособие*

САМАРА

2010

УДК 681.324.006.3

**Н.Л. Казанский, П.Г. Серафимович, С.Н. Хонина** Высокопроизводительные вычисления в дифракционной нанооптике. ИСОИ РАН. Самара. 2010, 157 с.

**ISBN 5-93673-021-9**

В учебном пособии изложены основы применения суперкомпьютера для решения задач нанооптики. Описаны способы удаленного доступа пользователя на суперкомпьютер. Рассмотрены этапы подготовки исполняемого файла параллельного приложения и запуск его на суперкомпьютере с помощью системы пакетной обработки заданий. Приведены примеры мониторинга запущенного задания.

Рассмотрены примеры решения вычислительных задач нанофотоники параллельным методом FDTD. Рассчитаны электромагнитные поля в прямом и изогнутом волноводах. Исследуются резонансные моды кольцевого диэлектрического резонатора. Моделируется дисперсия света в материале.

Также рассмотрено параллельное моделирование распространения световых полей с использованием дифракционных интегралов в рамках строгой теории дифракции с использованием высокопроизводительных вычислительных средств. Приведены краткие характеристики ряда известных программных продуктов для оптических приложений.

Учебное пособие предназначено для обеспечения учебного процесса базовой кафедры высокопроизводительных вычислений и ориентировано на массовую подготовку студентов в рамках программ подготовки бакалавров, магистров, слушателей ФПКП и других категорий, изучающих технологии параллельных вычислений. Учебное пособие может быть полезно также при проведении научных исследований, выполнении курсовых и дипломных проектов по физико-математическим и техническим направлениям подготовки.

Учебное пособие предназначено для студентов специальностей и направлений "Прикладная математика и информатика", "Прикладная математика и физика", а также аспирантов и докторантов, обучающихся по специальностям 01.04.05 «Оптика» и 05.13.18 «Математическое моделирование, численные методы и комплексы программ».

Печатается по решению ученого совета ИСОИ РАН.

Рецензенты: д. ф.-м. н., профессор В.В. Ивахник,  
д. ф.-м. н., доцент Д.Л. Головашкин.

© Казанский Н.Л., Серафимович П.Г., Хонина С.Н., 2010

© Институт систем обработки изображений РАН, 2010

© Самарский государственный аэрокосмический университет, 2010

**ISBN 5-93673-021-9**

Введение .....	6
<b>1. Выполнение вычислений на кластере .....</b>	<b>7</b>
<b>1.1 Способы доступа пользователя на кластер .....</b>	<b>7</b>
1.1.1 Удаленный доступ на кластер для компиляции и запуска расчетных программ .....	7
1.1.2 Удаленный доступ на кластер для копирования файлов между персональным компьютером пользователя и кластером .....	8
<b>1.2 Настроить окружение выполнения MPI программы .....</b>	<b>10</b>
1.2.1 Посмотреть текущее состояние .....	10
1.2.2 Посмотреть список возможных настроек .....	10
1.2.3 Установить окружение выполнения MPI программы .....	10
1.2.4 Перезагрузить программную оболочку .....	10
1.2.5 Проверить установленные настройки .....	11
<b>1.3 Подготовка исполняемого файла MPI приложения .....</b>	<b>11</b>
1.3.1 Редактирование текста программ пользователя .....	11
1.3.2 Создание ctf-файла .....	12
<b>1.4 Запустить MPI приложение на кластере .....</b>	<b>13</b>
1.4.1 Подготовка PBS-задания .....	13
1.4.2 Постановка PBS-задания в очередь на выполнение .....	13
1.4.3 Мониторинг запущенного задания .....	13
1.4.4 Состояние очереди заданий .....	13
1.4.5 Полная информация по заданию .....	13
1.4.6 Информация о состоянии очереди заданий от менеджера ресурсов .....	13
1.4.7 Полная информация по узлам кластера .....	13
<b>2. Примеры использования программного пакета Meep .....</b>	<b>14</b>
<b>2.1 Формат ctf-файла .....</b>	<b>14</b>
<b>2.2 Электромагнитное поле в волноводе .....</b>	<b>15</b>
2.2.1 Прямой волновод .....	15
2.2.2 Волновод с изгибом $90^\circ$ .....	19
2.2.3 Спектр пропускания через изогнутый волновод .....	23
<b>2.3 Моды кольцевого резонатора .....</b>	<b>29</b>
2.3.1 Исследование резонансных мод кольцевого резонатора в 2D- геометрии .....	29
2.3.2 Использование симметрии кольцевого резонатора .....	33
2.3.3 Кольцевой резонатор в цилиндрических координатах .....	33
<b>2.4 Дисперсия света в веществе .....</b>	<b>37</b>
<b>3. Моделирование распространения световых полей с     использованием дифракционных интегралов .....</b>	<b>42</b>
<b>3.1 Скалярные операторы распространения .....</b>	<b>43</b>
<b>3.2 Примеры расчета в рамках скалярной теории .....</b>	<b>44</b>

3.2.1 Дифракция плоской волны на круглой апертуре.....	44
3.2.2 Дифракция вихревого пучка на круглой апертуре.....	47
3.2.3 Дифракция плоской волны на бинарном осесимметричном аксиконе.....	48
<b>3.3 Векторные операторы распространения.....</b>	<b>49</b>
3.3.1 Векторное интегральное преобразование Релея-Зоммерфельда.....	49
3.3.2 Векторное представление через плоские волны.....	50
<b>3.4 Примеры расчета с использованием векторных интегралов.....</b>	<b>51</b>
3.4.1 Дифракция плоской волны на круглой апертуре.....	51
3.4.2 Дифракция вихревого пучка на круглой апертуре.....	52
<b>3.5 Векторное представление через плоские волны в модификации Мансуришура.....</b>	<b>54</b>
<b>3.6 Учёт в методе разложения по плоским волнам коэффициентов пропускания Френеля.....</b>	<b>56</b>
3.6.1 Представление через ТЕ- и ТМ-компоненты.....	56
3.6.2 Расчет коэффициентов Френеля.....	57
3.6.3 Дифракция плоской волны на бинарном би-аксиконе.....	59
<b>3.7 Описание программного обеспечения FieldPropagators.....</b>	<b>61</b>
3.7.1 Структура языка.....	61
3.7.1.1 Пакеты параметров.....	62
3.7.1.2 Пакеты несовместимых параметров.....	63
3.7.2 Варианты запуска программы.....	63
3.7.2.1 Запуск на расчет под Win*.....	63
3.7.2.2 Запуск на расчет под Linux.....	63
3.7.2.3 Запуск на расчет в фоновом режиме на кластере.....	63
3.7.2.4 Выдача справочной информации.....	64
3.7.3. Описание языка параметров.....	66
3.7.3.1 Правила задания параметров.....	66
3.7.3.2 Типы параметров.....	66
<b>3.8 Примеры использования программного обеспечения FieldPropagators.....</b>	<b>78</b>
3.8.1 Пример с одним пакет параметров.....	78
3.8.2 Пример с несколькими входными полями.....	79
3.8.3 Пример с несколькими вариантами расчета.....	81
3.8.4 Пример сравнения двух скалярных пропагаторов.....	83
3.8.5 Пример сравнения скалярного и векторного пропагаторов.....	84
3.8.6. Пример вывода топологии продольных сечений.....	86
<b>4 Программное обеспечение для оптических приложений.....</b>	<b>88</b>
<b>4.1 Краткий обзор программных продуктов.....</b>	<b>88</b>
<b>4.2 Программное обеспечение по компьютерной оптике.....</b>	<b>91</b>
<b>4.3 Программное обеспечение «SimuLight» для моделирования дифракционных оптических элементов.....</b>	<b>95</b>
4.3.1 Возможности «SimuLight» для моделирования оптических систем.....	96

4.3.2 Утилита Gds2Mask для преобразования из векторного многоуровневого формата GDS в набор бинарных GDS файлов....	99
<b>4.4 Программное обеспечение «TracePro».....</b>	<b>101</b>
4.4.1 Методы и алгоритмы, используемые «TracePro».....	101
4.4.2 Краткое изложение возможностей «TracePro».....	104
<b>Заключение к главе 4.....</b>	<b>111</b>
<b>Выводы.....</b>	<b>112</b>
<b>Приложение 1. Обзор необходимых команд Linux.....</b>	<b>113</b>
<b>Приложение 2. Примеры PBS скриптов.....</b>	<b>116</b>
<b>Приложение 3. Переменные окружения планировщика Torque.....</b>	<b>118</b>
<b>Список литературы.....</b>	<b>119</b>

## **Введение**

Данный учебный материал предназначен для студентов высших учебных заведений, специализирующихся в области прикладной математики, физики и информационных технологий.

Материал содержит инструкции по работе на суперкомпьютере СГАУ от компании ИР. Также в пособии рассмотрено применение одного из общепринятых методов численного моделирования в нанофотонике – метода FDTD (Finite-Difference Time Domain). Часть материала базируется на курсе лекций «Математические методы в нанофотонике» от Massachusetts Institute of Technology.

Особенностью данного учебного пособия является то, что основной акцент сделан на непосредственной работе на суперкомпьютере, что позволит студентам получить практические навыки параллельных вычислений на примерах задач нанофотоники. Тем не менее, прикладная направленность курса не помешает заинтересованным студентам более глубоко разобраться в аналитике на которой базируется курс [1-4].

В первой главе пособия описаны способы удаленного доступа пользователя на суперкомпьютер для компиляции и запуска расчетных программ пользователя. Также показано как настроить окружение выполнения параллельной программы. Рассмотрены шаги подготовки исполняемого файла параллельного приложения и запуск его на суперкомпьютере с помощью системы пакетной обработки заданий. Приведены примеры мониторинга запущенного задания.

Во второй главе рассмотрены три примера решения вычислительных задач нанофотоники методом FDTD.

В первом примере рассчитывается электромагнитное поле в прямом и изогнутом волноводе.

Второй пример иллюстрирует один из традиционных вариантов применения метода FDTD – расчет резонансных мод – их частот и скоростей затухания – для некоторой электромагнитной резонансной системы. В данном примере рассматривается нахождение резонансных мод кольцевого диэлектрического резонатора.

В третьем примере моделируется дисперсия света в материале. Сравниваются аналитические и рассчитанные методом FDTD действительные и мнимые части диэлектрической функции материала с зависимостью диэлектрической проницаемости от частоты.

В третьей главе, рассмотрено моделирование распространения световых полей с использованием дифракционных интегралов в рамках строгой теории дифракции, а также в различных приближениях и аппроксимациях, с использованием высокопроизводительных вычислительных средств.

В четвертой главе приведены краткие характеристики ряда известных программных продуктов для оптических приложений. Также дано подробное описание программного обеспечения, имеющегося в распоряжении на факультете информатики СГАУ и базовой кафедре высокопроизводительных вычислений ИСОИ РАН.

Авторы благодарят ведущего программиста ИСОИ РАН Сергея Геннадиевича Волоотовского за выполнение расчетов для 3-ей главы этого учебного пособия.

## 1. Выполнение вычислений на кластере

### 1.1 Способы доступа пользователя на кластер

#### 1.1.1 Удаленный доступ на кластер для компиляции и запуска расчетных программ

Пользователи операционной систем Linux могут воспользоваться стандартным ssh-клиентом. Пользователям Microsoft Windows рекомендуется использовать программу PuTTY (<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>)

При первом запуске программы в окне HostName набрать IP адрес кластера – 89.186.247.14. В поле Saved Sessions ввести любое удобное название (на рис. 1 выбрано название «HP»).

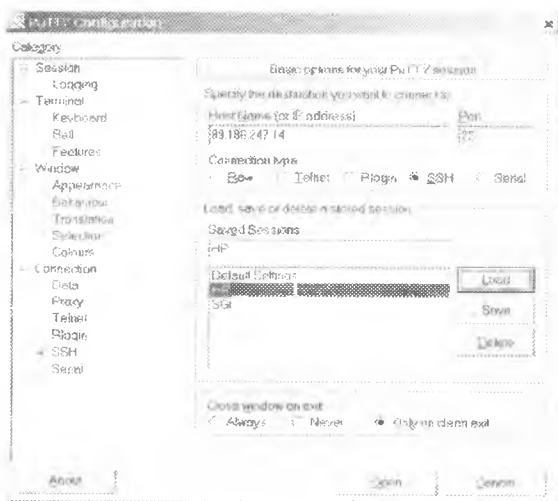


Рис. 1. Конфигурация программы PuTTY

В поле Category — Window — Translation — Character set translation on received data выбрать UTF-8 (см. рис. 2). Вернуться в окно Category — Session. Нажать кнопку «Save». Под именем HP будут сохранены настройки.

При последующих запусках PuTTY в окне конфигурации выбрать «HP» и щелкнуть кнопку “Load”. В окне Host Name появится IP адрес кластера – 89.186.247.14.

Нажать кнопку “Open”. Произойдет соединение с кластером, откроется окно терминала кластера. В этом окне сначала ввести имя пользователя, затем пароль.



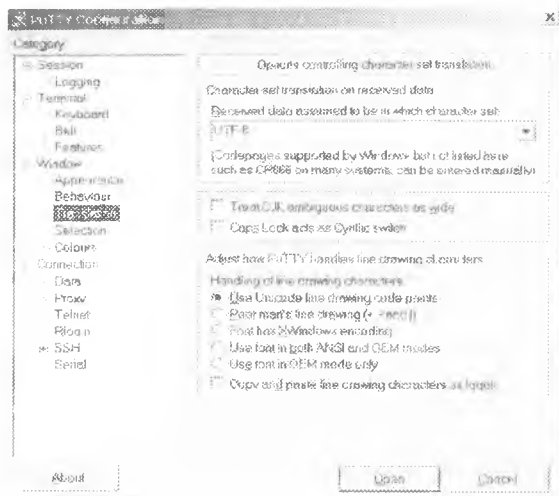


Рис. 2. Выбор таблицы кодировки символов

### 1.1.2 Удаленный доступ на кластер для копирования файлов между персональным компьютером пользователя и кластером

Сетевой файловый менеджер WinSCP может быть использован для файловых операций с удаленными и локальными файлами. Скачать программу WinSCP можно по адресу <http://winscp.net>.

При первом запуске программы WinSCP необходимо ввести в поле Host name IP адрес кластера – 89.186.247.14, а в поле User name имя пользователя (см. рис. 3). Поле Password можно не заполнять, его вы будете вводить при каждом следующем соединении. Введенные данные сохраняются кнопкой “Save”.

При последующих запусках программы окно WinSCP Login будет с заполненными личными данными (рис. 4).

Соединение начинается выбором пункта “Login”. В окне Server prompt надо ввести свой пароль (рис. 5). Для соединения с удаленной ЭВМ необходимо время. Поэтому окно программы WinSCP появится с некоторой задержкой. В зависимости от варианта, выбранного при установке программы, откроется окно, по внешнему виду сходное с Windows Explorer либо двухпанельное окно в стиле Total Commander. В этом окне будет отображена файловая система кластера и вы можете копировать файлы на кластер и обратно также, как это делается в Windows (рис. 6). Поддерживаются операции Drag and Drop.

Кроме того, программа WinSCP позволяет проводить другие операции с файлами: редактирование, переименование, удаление, изменение прав доступа и прочее.

Завершение работы с программой и прекращение связи проводится клавишей F10 или выбором в меню пункта Commands/Quit. Требуется подтвердить окончание работы в окне завершения.

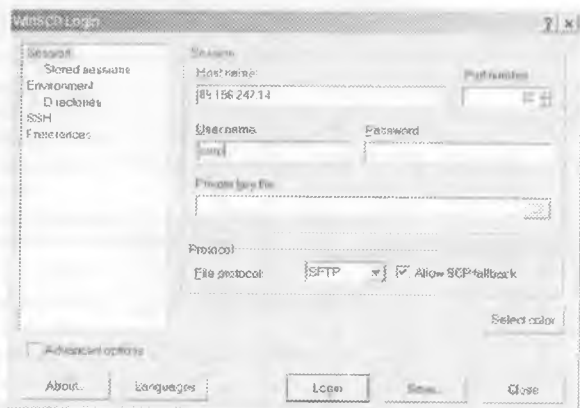


Рис. 3. Стартовое окно программы WinSCP при первом запуске

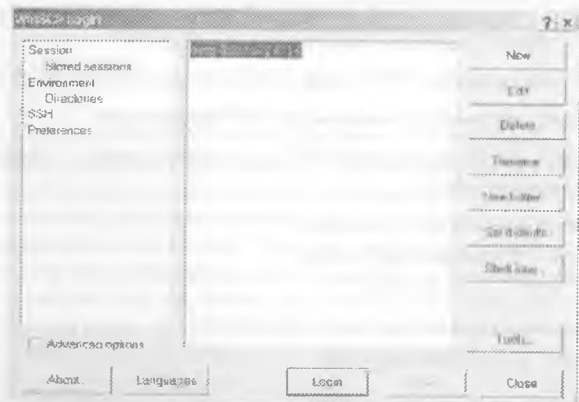


Рис. 4. Окно соединения

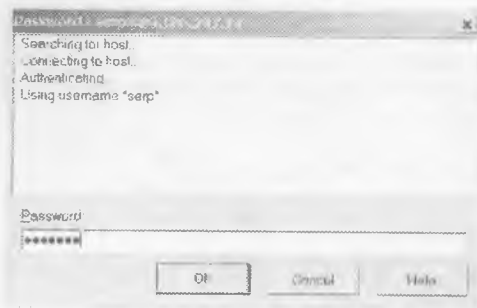


Рис. 5. Окно ввода пароля

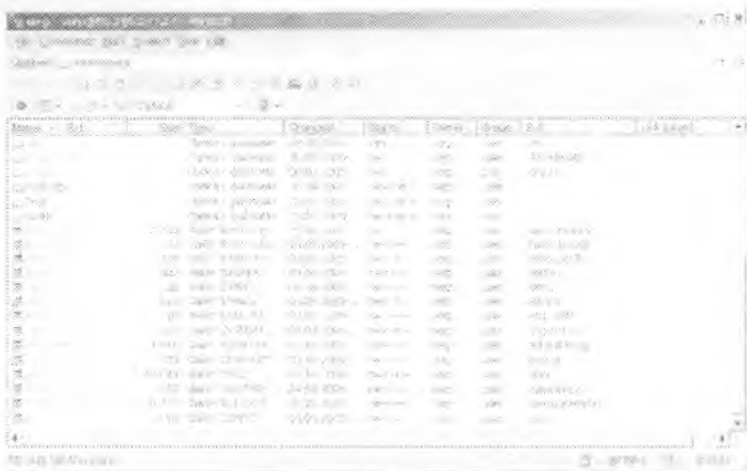


Рис. 6. Рабочее окно программы WinSCP

## 1.2 Настроить окружение выполнения MPI программы

Настроить окружение выполнения MPI программы можно с использованием команд утилиты Switcher.

### 1.2.1 Посмотреть текущее состояние

```
[tester@master ~]$ switcher mpi --show
user:default=lam-7.1.4 user:exists=1
```

### 1.2.2 Посмотреть список возможных настроек

```
[tester@master ~]$ switcher mpi --list
openmpi-1.2.4
mpich-ch_p4-gcc-1.2.7
lam-7.1.4
```

### 1.2.3 Установить окружение выполнения MPI программы

В примере будет установлено окружение mpich-ch\_p4-gcc-1.2.7

```
[tester@master ~]$ switcher mpi --add-attr default mpich-ch_p4-gcc-1.2.7
Warning: mpi:default already has a value:
lam-7.1.4
```

Replace old attribute value (y/N)? y

Attribute successfully set; new attribute setting will be effective for future shells

### 1.2.4 Перезагрузить программную оболочку

```
[tester@master ~]$ bash
```



Для запуска этой программы используется команда `ms`.

**Midnight Commander** — это файловый менеджер с текстовым интерфейсом. Его предназначение — упростить основные действия пользователя, связанные с управлением файлами. Принцип работы **Midnight Commander** такой же, как и у **Far Manager**, **TotalCommander**, или **Norton Commander**. Экран состоит из двух панелей, в которых отображается список файлов и каталогов в выбранных каталогах, и пользователь может выполнять некоторый набор действий над этими файлами. В нижней части экрана расположена командная строка и панель горячих клавиш F1-F10. Можно вызвать верхнее меню, нажав клавишу F9. Одна из панелей всегда является активной, а в ней курсор установлен на активный файл. Пользователь может выполнять действия либо с активным файлом или каталогом, либо групповые операции со всеми объектами активной панели. Также доступны некоторые общие операции: поиск файлов, помощь по работе с ОС, выполнение команд операционной системы и т.д.

Для редактирования файла необходимо клавишами управления курсором выбрать нужный файл, и нажать клавишу F4. Запустится редактор текстовых файлов, выйти из которого можно, нажав клавишу F10 либо Esc. Чтобы сохранить изменения, необходимо нажать клавишу F2, либо выбрать нужный вариант при выходе из редактора.

Для создания нового файла нужно нажать Shift+F4 (при нажатой клавише Shift нажмите клавишу F4). Откроется окно редактора, и при сохранении изменений программа предложит ввести имя сохраняемого файла.

### *1.3.2 Создание ctf-файла*

Одной из целей данного пособия является обучение решению задач нанофотоники на кластере. При решении таких задач используется теория электромагнетизма. Один из наиболее распространенных вычислительных аппаратов в классическом электромагнетизме — это метод FDTD (Finite-Difference Time Domain), который делит пространство и время на регулярную сетку и моделирует временную эволюцию уравнений Максвелла. Это пособие использует реализацию метода FDTD с открытым исходным кодом Meep, доступную онлайн [5]. Meep является программным пакетом с множеством функциональных возможностей. В Meep можно, например, моделировать произвольные анизотропные материалы, накладывать различные граничные условия, использовать декартовы (1D/2D/3D) и цилиндрические координаты.

Для формирования задания в пакете Meep используется скриптовый язык. Программы на этом языке содержатся в файлах с расширением `ctl`. Вот, например, как выглядит фрагмент `ctl`-файла, задающий геометрию кольцевого резонатора, который мы будем рассматривать в последующих разделах:

```
(set! geometry (list  
  (make cylinder (center 0 0) (height infinity)  
  (radius (+ r w)) (material (make dielectric (index n))))  
  (make cylinder (center 0 0) (height infinity)  
  (radius r) (material air))))
```

#### 1.4 Запустить MPI приложение на кластере

Выделение ресурсов и запуск приложений на кластере обеспечивает система пакетной обработки заданий Torque и менеджер ресурсов MAUI. Поэтому для того чтобы запустить MPI приложение на кластере необходимо выполнить следующее:

- подготовить PBS-задание
- поставить PBS-задание в очередь на выполнение.

##### 1.4.1 Подготовка PBS-задания

PBS-задание это некоторый скрипт написанный на языке командного интерпретатора, который содержит как директивы для самой системы пакетной обработки на выделение ресурсов, так и директивы для запуска задачи пользователя.

Пример. Для запуска MPI программы PBS- задание может быть оформлено следующим образом:

```
#PBS -l walltime=00:30:00,nodes=2:ppn=8
#PBS -q workq@master
#PBS -N job_name
#PBS -o /home/tester/out
#PBS -e /home/tester/err
#!/bin/sh
cd /home/student/Work/Mcep/Example
/usr/mp/intel/openmpi-1.2.8/bin/mpirun -np 16 -hostfile
/home/student/cluster_ns /usr/local/bin/mcep-mpi example.ctl
```

Значение параметров PBS задания смотрите в **Приложении 2** данного пособия.

##### 1.4.2 Постановка PBS-задания в очередь на выполнение

После того как PBS задание готово, его необходимо поставить в очередь на выполнение командой

```
qsub [имя PBS-задания]
[tester@master ~]$ qsub job_hpl
```

##### 1.4.3 Мониторинг запущенного задания

Мониторинг очереди заданий может быть выполнен с использованием терминальных команд системы пакетной обработки Torque или менеджера ресурсов MAUI.

##### 1.4.4 Состояние очереди заданий

```
[tester@master ~]$ qstat
```

##### 1.4.5 Полная информация по заданию

```
qstat -f [Job ID]
```

##### 1.4.6 Информация о состоянии очереди заданий от менеджера ресурсов

```
[tester@master ~]$ /opt/maui/bin/showq
```

##### 1.4.7 Полная информация по узлам кластера

```
[tester@master ~]$ pbsnodes
```

## 2. Примеры использования программного пакета Meep

Теория метода FDTD (Finite-Difference Time Domain) подробно описана в целом ряде работ, см., например, [6-8]. Перейдем сразу к вопросам практического использования метода FDTD для решения задач нанофотоники. Как было показано в предыдущем разделе для расчетов используется параллельная вычислительная среда — кластер.

В этом разделе мы разберем несколько примеров, которые иллюстрируют способы расчета электромагнитных полей, спектров пропускания и отражения, а также резонансных мод в Meep. Для ускорения расчетов все примеры используют двухмерную геометрию, которой достаточно для демонстрации основных особенностей рассматриваемых моделей. Естественным образом эти примеры обобщаются в Meep на 3D геометрию.

Для описания моделей в Meep имеются два способа. Во-первых, простой скриптовый язык Scheme. Во-вторых, более гибкий язык программирования C++. Мы будем использовать язык Scheme, который из-за своей простоты используется большинством пользователей Meep.

Выходные данные Meep записываются в формате HDF5. Чтобы преобразовать выходные файлы формата HDF5 в изображения полей, мы будем использовать бесплатные утилиты из набора h5utils. Однако возможно использовать любую другую программу, поддерживающую чтение HDF5 файлов, например, Matlab.

### 2.1 Формат ctf-файла

Использование Meep тесно связано с управляющим файлом, который имеет расширение «ctl», т.е. называется, например, example.ctl. Ctf-файл определяет изучаемую геометрию, источники тока, вычисляемую мощность и все остальное, касающееся наших расчетов. Как было уже сказано, ctf-файл является программой, написанной на скриптовом языке. Это означает, что он может содержать многое, начиная от простой последовательности команд для настройки геометрии и заканчивая полноценной программой, включающей ввод данных пользователем, циклы и все, что еще может понадобиться.

Тем не менее, простые вещи остаются простыми (не обязательно быть опытным программистом), но даже в них вы по достоинству оцените гибкость языка Scheme. Например, вы можете вводить данные в любом порядке, без учета пробелов, вставлять комментарии где вам угодно, опускать значения, когда доступны подходящие значения по умолчанию и т.д.

Команды ctf-файла используют библиотеку libctl – набора утилит, которые в свою очередь написаны на языке Scheme. Таким образом, синтаксис возможных команд основывается на трех уровнях:

- **Scheme**, удобный язык программирования, который был разработан в MIT и обладает простым синтаксисом: все выражения имеют вид (*функция аргументы...*). Scheme запускается в интерпретаторе GNU Guile. Не нужно хорошо знать Scheme для создания простейшего ctf-файла; однако данный язык всегда в нем присутствует; вы можете узнать больше, перейдя по ссылке [9].

- **libctl**, библиотека, расширяющая возможности интерпретатора Guile для упрощения взаимодействия Scheme с программным обеспечением для научных вычислений. Libctl устанавливает основной стиль интерфейса и определяет ряд полезных функций (например, оптимизацию с несколькими переменными, численное интегрирование и т.д.) [10].
- **Meep**, который описывает элементы интерфейса, являющиеся определяющими для использования метода FDTD. Мы сосредоточимся на этом уровне синтаксиса команд.

На данном этапе полезно обратиться к руководству по libctl и обрести начальное понимание интерфейса прежде, чем перейти к вещам, характерным именно для Meep.

Обычно программу Meep можно вызвать, запустив в командной строке Unix (далее обозначается строкой `unix%`) или в скрипте PBS-задания, например, команду:

```
unix% meep example.ctl >& example.out
```

Данная команда считывает `ctl`-файл `example.ctl`, выполняет его и выводит результат вычислений в файл `example.out`. Если же вызвать `meep` без параметров, то он перейдет в интерактивный режим, в котором можно ввести команду и сразу же увидеть результат. Если вы сделаете это сейчас, то вы можете вставлять команды из данного руководства и смотреть, что они делают.

## 2.2 Электромагнитное поле в волноводе

В качестве первого примера рассмотрим конфигурацию поля, вызванного источником незатухающих колебаний сначала - в прямом, а затем - в изогнутом волноводе. Ширина волновода  $1$ , а  $\epsilon = 12$  (без дисперсии). Единицу длины выберем такую, чтобы ширина волновода равнялась  $1$ ; и, исходя из этого, определим остальные величины.

### 2.2.1 Прямой волновод

Прежде чем определять геометрию структуры, необходимо задать рабочую область. Поместим в один конец волновода источник и будем наблюдать распространение волны по волноводу (в направлении оси  $x$ ). Будем использовать ячейку длиной  $16$  по оси  $X$ , чтобы было некоторое расстояние для распространения волноводной моды. В направлении оси  $Y$  нам нужно пространство, достаточное для того, чтобы границы не влияли на форму волноводной моды. Поэтому положим размер в направлении  $Y$  равным  $8$ . Зададим эти размеры в нашем `ctl`-файле с помощью переменной `geometry-lattice`.

```
(set! geometry-lattice (make lattice (size 16 8 no-size)))
```

Здесь `set!` – это команда языка Scheme для определения значения вводимой величины. Последний параметр `no-size` показывает, что у вычисляемой области нет размера по оси  $z$ , т.е. что она является двумерной.



Теперь можно добавить волновод. Обычно в Meep моделируемую структуру определяет список геометрических объектов, хранящихся в переменной геометрии:

```
(set! geometry (list
  (make block (center 0 0) (size infinity 1 infinity)
    (material (make dielectric (epsilon 12))))))
```

Волновод представляет собой блок (параллелепипед) размера  $\infty \times 1 \times \infty$ .  $\epsilon = 12$ , волновод центрирован в точке с координатами (0,0) (положение центра рабочей области). По умолчанию в любом месте, где нет никаких объектов, находится воздух ( $\epsilon = 1$ ), хотя это можно изменить, устанавливая значение переменной *default-material*. Получившаяся структура показана на рис. 8.

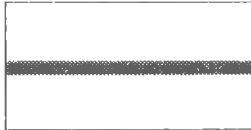


Рис. 8 Геометрия прямого волновода

Теперь, когда у нас есть структура, необходимо определить источники излучения, которые установлены списком *sources*. Простейшим является добавление точечного источника *Jz*.

```
(set! sources (list
  (make source
    (src (make continuous-src (frequency 0.15)))
    (component Ez)
    (center -7 0))))
```

Здесь задана частота источника 0,15 и определен объект *continuous-src*, который представляет собой синусоидальную экспоненту  $\exp(-i\omega t)$  с постоянной частотой. Этот источник (по умолчанию) запускается в момент времени  $t=0$ . Отметим, что в Meep единица частоты соответствует значению  $2\pi$ , что эквивалентно обратной величине длины волны в вакууме. Т.е. 0,15 соответствует длине волны в вакууме около  $1 / 0.15 = 6.67$  или длине волны 2 в материале где  $\epsilon = 12$ . Таким образом, наш волновод имеет ширину в половину длины волны, что, как мы надеемся, должно обеспечить его одномодовость. (В действительности задача о сокращении количества возможных мод в данном волноводе до одной моды аналитически разрешима; решение соответствует частоте  $1/2\sqrt{11}$  или примерно 0.15076.) Чтобы задать *Jz*, необходимо определить составляющую *Ez* (однако, например, для определения магнитного потока следует задать *Nx*, *Ny* и/или *Nz*). Источник тока расположен в точке (-7;0). Это на единицу длины правее левого края области (необходимо оставить некоторое пространство между источником и границами, чтобы избежать влияния граничных условий).

Чтобы еще сократить влияние границ расчетной области на результаты, добавим поглощающие слои вокруг всех границ нашей области. В Meep поглощающие слои реализуются с помощью техники *идеально согласованных слоев* (PML – Perfect Matched Layer). По сути они являются не граничными условиями, а фиктивным поглощающим материалом вокруг расчетной области. Чтобы добавить вдоль всех границ области поглощающий слой толщиной 1 следует выполнить команду:

```
(set! pml-layers (list (make pml (thickness 1.0))))
```

PML-слои представляют собой список объектов. У вас может быть только 1 pml объект, если вы хотите создать PML-слой с определенной стороны области. Например, команда

```
(make pml (thickness 1.0) (direction X) (side High))
```

добавит PML-слой только в положительном направлении оси x перпендикулярно к ней. Отметим важный момент: PML-слой находится внутри клетки, покрывая любые объекты внутри неё. В нашем случае PML-слой покрывает волновод, это и требуется, чтобы PML-слои поглощали волноводные моды. Конечная ширина PML важна для уменьшения повторных отражений.

Meep будет дискретизировать структуру по пространству и времени. Дискретизация определяется одной переменной: разрешением, которая задает число пикселей на единицу длины. Установим разрешение 10, что соответствует примерно 67 пикселей/длину волны или 20 пикселей/длину волны в материале с наибольшей диэлектрической проницаемостью для рассматриваемой геометрической структуры. В диэлектрике разумным является разрешение хотя бы в 8 пикселей/длину волны. Заданное нами разрешение даст нам расчетную область размером 160×80 гочек.

```
(set! resolution 10)
```

Теперь мы готовы начать моделирование. Сделаем это, вызвав команду *run-until*. Первым аргументом является время моделирования, последующие аргументы определяют, например, компоненты электромагнитных полей, которые необходимо вывести или другие виды анализа на каждом шаге выполнения по времени:

```
(run-until 200  
  (at-beginning output-epsilon)  
  (at-end output-efield-z))
```

Здесь мы выводим диэлектрическую функцию  $\epsilon$  и компоненту электрического поля  $E_z$ . Эти функции вывода вложены в операторы *at-beginning* и *at-end*, смысл которых соответствует их названиям. В противном случае вывод производился бы на каждом шаге. В Meep имеются другие подобные функции, изменяющие содержимое выводимых данных. Также вы можете написать свои

собственные функции; кроме того, вы можете проводить любые вычисления или выводить любые значения в любое время в процессе выполнения и даже изменять параметры моделирования в то время как он запущен.

Прогон нашей задачи должен занимать порядка нескольких секунд. Если вы запустили его в интерактивном режиме, то 2 выходных файла будут называться `eps-000000.00.h5` и `ez-000200.00.h5` (заметьте, что имена файлов включают время, в которое они были созданы). Если вы запустили `tutorial.ctf` file, то выходные файлы будут называться `tutorial-eps-000000.00.h5` и `tutorial-ez-000200.00.h5`. В любом случае теперь мы можем эти файлы проанализировать и визуализировать с помощью большого количества программ, поддерживающих формат HDF5, включая беслинейные утилиты `h5utils`. В частности, программу `h5topng` для преобразования HDF5-файлов в изображения PNG.

```
unix% h5topng -S3 eps-000000.00.h5
```

эта команда создаст файл `eps-000000.00.png`; здесь `-S3` увеличивает масштаб изображения в 3 раза. В данном случае ширина изображения будет около 450 отсчетов). Именно этой командой было создано изображение диэлектрика на рис. 8. Однако, куда более содержательным является изображение рассчитанного электромагнитного поля:

```
unix% h5topng -S3 -Zc dkbluered -a yarg -A eps-000000.00.h5 ez-000200.00.h5
```

Здесь `-Zc dkbluered` устанавливает цветовую градацию от темно-синего (отрицательное значение) до темно-красного (положительное). Белый цвет соответствует 0. Параметры `-a/-A` накладывают диэлектрическую функцию светло-серыми контурами. Результатом является изображение на рис. 9.

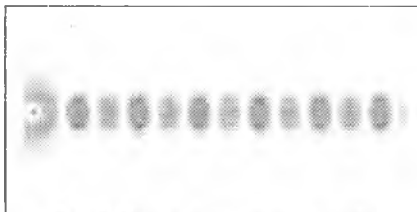


Рис. 9 Поле в прямом волноводе

На рисунке видно, что источник возбудил волноводную моду, а также излучающие поля, распространяющиеся в стороны от волновода. На границах благодаря наличию PML слоев поле быстро обращается в ноль. Если внимательно посмотреть на изображение, то можно увидеть кое-что ещё: в правой части изображение «пятнистое». Это наблюдается потому, что резко включив источник в момент времени ( $t=0$ ) мы вызвали высокочастотные компоненты (моды очень высоких порядков), но не подождали достаточно времени чтобы они исчезли; в следующем разделе мы избавимся от таких мод, включая источник плавно.

### 2.2.2 Волновод с изгибом 90°

Сейчас мы начнем новое моделирование, в котором рассмотрим поля в изогнутом волноводе, а так же изменим некоторые детали процесса моделирования. Если вы запустили Меср в интерактивном режиме, то необходимо сначала избавиться от старой структуры и полей, чтобы Меср заново рассчитал их:

```
(reset-mesr)
```

Несколько расширим размеры расчетной области для изогнутого волновода:

```
(set! geometry-lattice (make lattice (size 16 16 no-size)))
```

```
(set! geometry (list  
  (make block (center -2 -3.5) (size 12 1 infinity)  
    (material (make dielectric (epsilon 12))))  
  (make block (center 3.5 2) (size 1 12 infinity)  
    (material (make dielectric (epsilon 12))))))
```

```
(set! pml-layers (list (make pml (thickness 1.0))))  
(set! resolution 10)
```

Отметим, что сейчас моделируемая структура содержит два блока, смещенных относительно центра для того, чтобы создать показанную на рисунке 10 структуру. Как показано на рис. 10, начало системы координат (0,0) находится в центре рабочей области. Ось  $Y$  в утилите *h5topng* направлена вниз, поэтому центр блока размером  $12 \times 1$  находится в точке  $(-2, -3,5)$ . Зеленый отрезок показывает положение источника при  $x=-7$ .

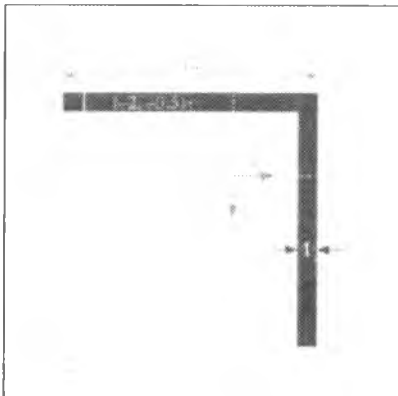


Рис. 10 Изогнутый волновод

Также необходимо переместить источник в точку с координатой  $y=-3,5$ , чтобы он оставался внутри волновода. На этом этапе внесем несколько изменений.

Во-первых, точечный источник недостаточно хорошо подходит для работы с модами волновода - поэтому мы преобразуем в источник-отрезок той же длины, что и волновод, добавив в источник параметр *size*. Во-вторых, вместо того, чтобы мгновенно включать источник в момент времени  $t=0$ , что приводило к появлению искажающих высоких частот, мы будем делать это медленно в течение времени, пропорционального 20 (параметр *width*) временным единицам, что соответствует немногим более, чем трем периодам. Наконец, просто для того, чтобы поупражняться, мы зададим длину волны (в вакууме) вместо частоты. Будем использовать такую длину волны, чтобы ширина волновода была в 2 раза меньше.

```
(set! sources (list
  (make source
    (src (make continuous-src
      (wavelength (* 2 (sqrt 12)))) (width 20)))
  (component Ez)
  (center -7 -3.5) (size 0 1))))
```

Наконец, мы начинаем моделирование. Однако, вместо того, чтобы запустить *output-efield-z* лишь в конце моделирования, мы будем делать это каждые 0.6 единиц времени (порядка 10 раз за период), что соответствует коду (*at every 0.6 output-efield-z*). Сама по себе данная команда будет создавать отдельные файлы для различных промежутков времени. Однако мы вместо этого используем другую функцию Месер, чтобы вывести все в один 3-мерный файл HDF5, где третьим измерением является время:

```
(run-until 200
  (at-beginning output-epsilon)
  (to-appended "ez" (at-every 0.6 output-efield-z)))
```

здесь "ez" определяет название выходного файла, который будет назван ez.h5, если вы запустили Месер в интерактивном режиме, или получит приставку, совпадающую с именем ctl-файла (например, tutorial-ez.h5 для tutorial.ctl). Если посмотреть параметры этого файла утилитой *h5ls*, то получим следующее:

```
unix% h5ls ez.h5
ez          Dataset {161, 161, 330/Inf}
```

т.е. файл содержит единственный набор данных ez, являющийся массивом  $161 \times 161 \times 330$ ; последнее измерение – это время. Получившийся файл достаточно большой, 69 МВ; позже мы увидим способы уменьшить размер – в случае, когда нас будут интересовать только изображения. Сейчас у нас несколько вариантов, как вывести поля. Чтобы вывести единичный срез по времени, можно воспользоваться командой *h5topng* как и прежде, но с дополнительной опцией *-t* чтобы установить момент времени. Например, *h5topng -t 329* выведет последний интервал времени, подобно предыдущему разделу. Вместо этого создадим анимацию поля как функцию времени. Во-первых, необходимо создать изображение на всех временных срезах:

```
unix% h5topng -t 0:329 -R -Zc dkbluered -a yarg -A eps-000000.00.h5
ez.h5
```

Эта команда подобна встречавшейся ранее, только с двумя новыми опциями: `-t 0:329` выводит изображение по всем временным индексам с 0 по 329, т.е. во всем промежутке времени; флаг `-R` указывает `h5topng` использовать последовательную цветовую шкалу для каждого изображения (вместо того, чтобы формировать ее независимо для каждого изображения). Затем преобразуем эти изображения в анимацию определенного формата. Для этого используем программу `convert` пакета `ImageMagick` (хотя есть и другое программное обеспечение, выполняющее то же самое).

```
unix% convert ez.t*.png ez.gif
```

Теперь результаты расчета представлены в анимированном формате GIF. Один кадр анимации представлен на рис. 11.

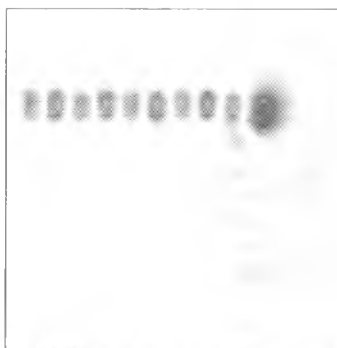


Рис. 11 Поле в изогнутом волноводе

Очевидно, что для данной частоты и заданной структуры прошедшее по волноводу через область изгиба излучение довольно мало, ясно видны большие отражение и потери на рассеяние.

Выше мы выводили порцию 2D-данных каждые 0.6 единиц времени, получив в итоге файл объемом 69 Мб. Для современных вычислительных устройств этот размер файла не представляет неудобств, но нетрудно представить, каким стал бы объем выходного файла в случае 3D-моделирования или даже более масштабного 2D – он легко бы дошел до нескольких гигабайт, что не только расточительно по использованию памяти, но и медленно по времени обработки. Вместо этого есть способ проводить вывод данных более эффективно, если иметь четкое представление о требуемом результате.

В вышеприведенном примере всё, что требовалось, чтобы создать видеоролик – это изображения поля, соответствующие каждому заданному моменту времени. Хранить такие изображения намного более выгодно, чем непосредственно массивы чисел. Чтобы воспользоваться этим, в Мсер предусмотрена возможность выводить данные не только в HDF5-файлы, но и в изображения форм-

мата PNG. В частности, вместо использованной выше команды **output – efield – z** можно написать **output – png Ez “–Zc dkbluered”**, где Ez – выходная компонента поля, а “–Zc dkbluered” – опции для h5topng ( программа, используемая для создания файлов изображений ). И, наконец:

```
( run until 200 ( at-every 0.6 ( output – png Ez “–Zc bluered” ) ) )
```

– выводит каждые 0.6 единиц времени PNG-файл, из которых далее можно собрать видеоряд с помощью команды **convert**. Получившийся ролик будет подобен уже рассмотренному примеру, но будут и различия, обусловленные заданием цветовой шкалы. Ранее была использована опция **–R** утилиты **h5topng** задающая равномерную шкалу цвета для всех изображений, построенную на максимальном/минимальном значениях полей за ВСЕ временные отсчеты. Однако, здесь подобное невозможно, поскольку изображения выводятся еще до того, как становятся известными значения полей в еще не прошедшие отсчеты времени. Соответственно по команде **output – png** цветовая шкала формируется по максимальному и минимальному значениям полей за все ПРОШЕДШИЕ отсчеты времени. Поэтому шкала цвета будет медленно «наращиваться» по мере запуска источника излучения.

В результате работы приведенных выше команд будет создано огромное число .png- файлов. Будет нежелательно, если все они заполнят рабочую директорию. Чтобы избежать этого, воспользуемся следующей командой перед запуском **run-until** :

```
( use – output – directory )
```

В результате все выходные файлы ( .h5 , .png и проч. ) будут помещены в созданную поддиректорию, по умолчанию названную *filename-out* , что соответствует имени *ctl* файла *filename.ctl* .

Но как быть, если требуется вывести срез  $x \times t$  для фиксированного значения  $y$  (например,  $y = -3.5$ )? Для этого мы можем воспользоваться другим мощным средством вывода в Meer – пакет позволяет выводить лишь подмножество области вычислений. Это реализуется с помощью функции **in – volume** , которая, подобно **at – every** и **to – appended**, способна влиять на другие функции вывода данных. В частности, можно запустить процедуру:

```
( run – until 200  
  ( to – appended “ez – slice”  
    ( at every 0.6  
      ( in - volume ( volume ( center 0 -3.5 ) ( size 16 0 ) )  
        output – efield – z ) ) ) )
```

Первый аргумент для **in – volume** – значение объема, заданное через ( volume ( center ... ) ( size ... ) ). Оно применяется во всех вложенных функциях. ( обратите вниманис на то что **to – append**, **at – every**, **in – volume** выполняются в совокупности, не зависимо от их порядка. В результате создается выходной

файл *ez-slice.h5* , в котором содержится набор данных размеров  $162 \times 330$  соответствующий требуемому срезу  $x \times t$ .

### 2.2.3 Спектр пропускания через изогнутый волновод

Выше мы построили картину поля для прохождения света через изогнутый волновод. Хотя цель и была достигнута, но мы не получили количественных данных. Хотелось бы точно знать, какая мощность у прошедшего через изгиб излучения, какая – у отраженного и, наконец, сколько энергии было рассеяно. Как решить такую задачу?

Основные шаги следующие. Сначала мы сохраняем значения рассчитанных полей и их Фурье-образы в некоторой области и далее по ним вычислим поток электромагнитной энергии как функцию частоты  $\omega$ . Более того, мы получим полный спектр пропускания, запустив модель лишь однажды. Для этого потребуются построить Фурье-образ отклика системы на короткий импульс. По с целью нормирования значения прошедшего излучения ( т. е. чтобы рассчитать прошедшее излучение как долю от падающего ), нам потребуются два запуска модели: один с изгибом волновода, второй – без него.

В этом примере управляющий *ctl*-файл будет более сложным, чем в ранее рассмотренных примерах. Поэтому будет определено удобнее оформить его именно как отдельный файл, а не вводить все команды в диалоговом режиме. Рекомендуем просмотреть файл *bend-flux.ctl* , прилагающийся к пакету Meep – он находится в папке *examples/* .

В рассмотренных примерах мы всякий раз однозначно задавали все параметры – размер области, ширину волновода и т. д. Однако, для серьезных задач такой подход неэффективен – часто требуется рассмотреть несколько различных значений подобных параметров. К примеру, может потребоваться изменить размеры рабочей области. Тогда потребуется определить их:

**(define-param sx 16) ; размер области в направлении X**  
**(define-param sy 32) ; размер области в направлении Y**  
**(set! geometry-lattice (make lattice (size sx sy no-size)))**

Обратите внимание на точку с запятой « ; » - после этого символа следуют комментарии, текст которых не исполняется программой. *Define – param* – средство библиотеки *libctl* , позволяющее определить параметры, значения которых могут быть изменены в командной строке. Теперь мы можем, например, запустить команду

```
meep sx=17 tut-wvg-bend-trans.ctl
```

меняющую значение размера расчетной области по оси X на 17 без внесения каких-либо изменений в *ctl*-файл. Кроме этого определим еще два параметра: ширину волновода и «прокладку» (*padding*) - интервала между ним и границей рабочей области.

**(define-param pad 4) ; дистанция между волноводом и границей области**



**(define-param w 1) ; ширина волновода**

Чтобы задать позицию волновода, потребуется выполнить арифметические вычисления. К примеру, координата  $y$  горизонтального волновода определяется выражением  $-0.5 * (sy - w - 2 * pad)$ . По крайней мере, на языке программирования Си выражение бы выглядело подобным образом. Но на языке Scheme действие  $1 + 2$  записывается как  $(+ 1 2)$  и т. д. соответственно, центры горизонтального и вертикального волноводов определяются следующим образом:

**(define wvg-ycen (\* -0.5 (- sy w (\* 2 pad)))) ; y – центр гор. волновода**  
**(define wvg-xcen (\* 0.5 (- sx w (\* 2 pad)))) ; x – центр верт. волновода**

Теперь, как и в предыдущих примерах, нам необходимо задать геометрию системы. Однако, в этот раз это предстоит сделать дважды: для изогнутого, а также для прямого волновода, необходимого для нормировки. Можно было бы сделать это, создав два отдельных `ctl`-файла, но этот путь неудобен. Вместо этого мы определим дополнительный параметр *no-bend?*, который будет принимать значение *true* для прямого волновода и *false* – для изогнутого.

**(define-param no-bend? false) ; true – прямой волновод; false – изогнутый**

Далее мы определяем геометрию системы, разделяя процедуру на две ветви с помощью оператора `if`, синтаксис которого на Scheme имеет вид: **(if predicate? if-true if-false)**.

```
(set! geometry
  (if no-bend?
    (list
      (make block
        (center 0 wvg-ycen)
        (size infinity w infinity)
        (material (make dielectric (epsilon 12))))))
    (list
      (make block
        (center (* -0.5 pad) wvg-ycen)
        (size (- sx pad) w infinity)
        (material (make dielectric (epsilon 12))))
      (make block
        (center wvg-xcen (* 0.5 pad))
        (size w (- sy pad) infinity)
        (material (make dielectric (epsilon 12))))))
```

Таким образом, если значение *no-bend?* = *true*, то будет создан единственный блок материала – для случая прямого волновода. Иначе же будут созданы оба блока – для изогнутого волновода. Источник излучения теперь задается командой *gaussian-src* вместо использованного ранее *continuous-src*. При

этом в качестве параметров задается центральная частота и ширина полосы частот ( ширина Гауссова спектра ), которые определяются, как и ранее, посредством команды *define – param*.

```
(define-param fcen 0.15) ; pulse center frequency
(define-param df 0.1) ; ширина импульса ( по частоте )
(set! sources (list
  (make source
    (src (make gaussian-src (frequency fcen) (fwidth df)))
    (component Ez)
    (center (+ 1 (* -0.5 sx)) wvg-ycen)
    (size 0 w))))
```

Обратите внимание на то, каким образом используются параметры *wvg-ycen* и *w* : если изменить геометрические размеры, то теперь управляющий файл перестроится автоматически. Граничные условия и разрешения задаются так же, как и раньше, кроме того, что теперь мы используем команду *set – param!* , благодаря чему становится возможно переопределить разрешение из командной строки.

```
(set! pml-layers (list (make pml (thickness 1.0))))
(set-param! resolution 10)
```

Наконец, нам осталось определить область, в которой Meep предстоит вычислять спектр потока, а также частоты, на которых это требуется сделать. ( этот шаг необходимо выполнить после описания геометрии системы, источников излучения, разрешения, и т.д., поскольку все параметры поля инициализируются в процессе формирования поверхностей для вычисления потока).

```
(define-param nfreq 100) ; количество частот для расчета потока
(define trans ; прошедший поток
  (add-flux fcen df nfreq
    (if no-bend?
      (make flux-region
        (center (- (/ sx 2) 1.5) wvg-ycen) (size 0 (* w 2)))
        (make flux-region
          (center wvg-xcen (- (/ sy 2) 1.5)) (size (* w 2) 0))))))
(define refl ; reflected flux
  (add-flux fcen df nfreq
    (make flux-region
      (center (+ (* -0.5 sx) 1.5) wvg-ycen) (size 0 (* w 2)))))
```

Потоки рассчитываются через отрезки длиной вдвое шире волновода, расположенные в его начале или конце. Отметим, что эти отрезки удалены на 1 от границ области – поэтому они не входят в область поглощения PML. Здесь снова возможно два случая: пропущенный поток вычисляется либо в правой, либо в нижней части рабочей области, в зависимости от того, прямой волновод или изогнутый.

В данном примере потоки будут рассчитываться для 100 ( значение `nfreq` ) частот, значение центральной из которых = `fcen`, т.е. от `fcen - df / 2` до `fcen + df / 2`. Таким образом, вычисляются лишь частоты из диапазона нашего импульса. Это играет важную роль, поскольку далеко за пределами диапазона частот импульса источника спектральная энергия достигает настолько малых значений, что вычисление потоков теряет смысл.

Теперь встает необходимость разделить падающее и отраженное излучение. В Meep это производится путем сохранения Фурье-образов полей при запуске модели, предназначенном для нормировки ( когда `no-bend?` = `true` ) и последующей загрузки их инвертированных (умноженных на -1) значений перед последующими запусками модели. Это позволяет вычесть Фурье-образ падающего поля из образа рассеянного. По логике, мы могли бы проделать это после очередного запуска модели, но в действительности оказывается более удобно сначала вычесть значения падающего поля и лишь затем работать с Фурье-образами. Для выполнения всего описанного достаточно запустить лишь 2 команды: `save -flux` ( после нормировочного прогона модели ) и `load -minus -flux` ( перед последующими запусками ). Следующий код вызывает эти команды:

```
(if (not no-bend?) (load-minus-flux "refl-flux" refl))
(run-sources+ 500 (at-beginning output-epsilon))
(if no-bend? (save-flux "refl-flux" refl))
```

Здесь использован файл под названием `refl-flux.h5` или, в действительности, `bend-flux-refl-flux.h5` ( здесь имя `ctl` – файла использовано как префикс ) – он служит для хранения и загрузки Фурье-образов полей на отрезках, на которых вычисляется поток. Команда `run - sources+ 500` запускает модель на время, пока включен гауссовский источник излучения (он отключится автоматически, как только затухание составит несколько средних квадратических отклонений) + 500 единиц времени.

Почему работа модели не прекращается вместе с выключением излучателя? Дело в том, что необходимо дать импульсу время полностью распространиться по рабочей области. Более того, необходимое для этого время достаточно трудно предсказать, если вы имеете дело со сложными структурами. Это связано с возможными резонансными явлениями, из-за которых излучение источника прерывисто изменяется долгое время. По этой причине, удобно задать время запуска по-другому: вместо того чтобы использовать определенное значение, потребуем, чтобы затухание величины квадрата амплитуды поля  $|E_z|^2$  на конце волновода достигло определенной доли ( к примеру,  $1 / 1000$  ) от ее максимального значения. Это осуществляется посредством следующей процедуры:

```
(run-sources+
(stop-when-fields-decayed 50 Ez
(if no-bend?
(vector3 (- (/ sx 2) 1.5) wvg-ycen)
(vector3 wvg-xcen (- (/ sy 2) 1.5)))
1e-3))
```

Здесь *stop-when-fields-decayed* принимает четыре аргумента: (*stop-when-fields-decayed dT component pt decay-by*). После прекращения излучения источников работа модели продолжается дополнительно *dT* единиц времени всякий раз, когда значение величины

$|component|^2$  в заданной точке *pt* не уменьшилось как минимум на долю *decay-by* от ее максимального значения за *dT* предыдущих отсчетов. В приведенном примере *dT* = 50, *component* – *Ez*, рабочая точка – центр поверхности расчета потока на конце волновода и значение *decay-by* = 0.001. Соответственно, модель будет работать дополнительно 50 единиц времени до тех пор, пока затухание величины квадрата амплитуды излучения не составит 1/1000 от ее максимального значения. Этого должно оказаться достаточно для гарантии сходимости Фурье-преобразований. И наконец, необходимо вывести значения потоков:

**(display-fluxes trans refl)**

На выходе программы будет нечто подобное:

```
flux1:, 0.1, 7.91772317108475e-7, -3.16449591437196e-7
flux1:, 0.101010101010101, 1.18410865137737e-6, -4.85527604203706e-7
flux1:, 0.102020202020202, 1.77218779386503e-6, -7.37944901819701e-7
flux1:, 0.103030303030303, 2.63090852112034e-6, -1.11118350510327e-6
flux1:, ...
```

Здесь данные разделены запятыми – в таком формате они легко могут быть импортированы в любую программу табличных вычислений или средство построения графиков ( к примеру, Matlab ). Первый столбец соответствует значениям частоты, второй – мощность прошедшего излучения и третий – мощность отраженного излучения.

Далее нам необходимо *дважды* запустить модель: в одном случае значение переменной *no-bend?* = *true*, в другом = *false* ( последнее принимается по умолчанию ):

```
unix% meep no-bend?=true bend-flux.cfl | tee bend0.out
unix% meep bend-flux.cfl | tee bend.out
```

( команда *tee* – удобное средство в Unix , позволяющее как сохранить выходной файл, так и вывести на экран его содержимое. Это дает возможность отслеживать ход процессов при запуске модели ). Теперь нам необходимо поместить строки, начинающиеся с *flux1* , в отдельный файл, чтобы иметь возможность импортировать его в средство построения графиков:

```
unix% grep flux1: bend0.out > bend0.dat
unix% grep flux1: bend.out > bend.dat
```

теперь мы можем импортировать их в Matlab с помощью команды *dload* и вывести графическое представление результатов (рис. 12).

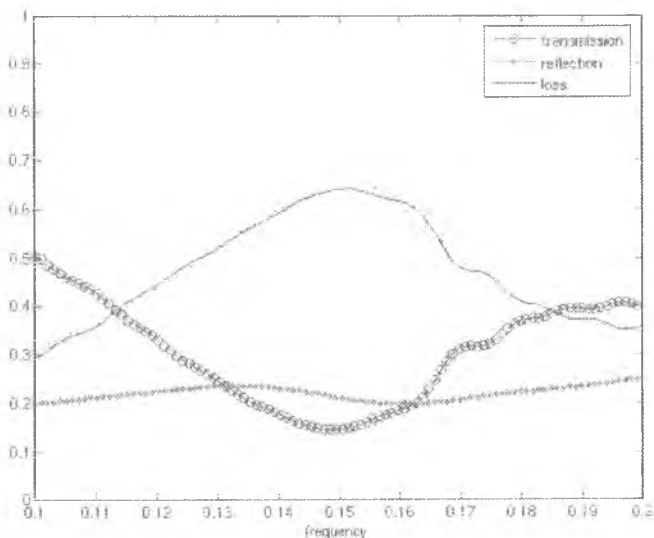


Рис. 12 Отражение и пропускание изогнутого волновода

Рассмотрим приведенный график. Transmission соответствует прошедшему потоку (второй столбец в файле *bend.dat*), деленному на падающий поток (второй столбец в *bend0.dat*). Таким образом, мы рассматриваем долю прошедшего излучения. Reflection соответствует отраженному излучению (третий столбец из *bend.dat*), деленному на падающий поток (опять же второй столбец из *bend0.dat*); здесь необходимо домножить значение на  $-1$ , поскольку в Meep поток всегда вычисляется в положительном направлении координат по умолчанию, а здесь рассматривается поток в направлении  $-x$ . И наконец, *loss* – потери – составляют просто значение  $(1 - \text{transmission} - \text{reflection})$ .

Теперь необходимо проверить сходимость результата. Для этого необходимо увеличить разрешение и размер рабочей области и заметить, насколько существенно меняются значения. В данном случае проведем лишь удвоение размера рабочей области:

```
unix% meep sx=32 sy=64 no-bend?=true bend-flux.ctl |tee bend0-big.out
unix% meep sx=32 sy=64 bend-flux.ctl |tee bend-big.out
```

здесь снова необходимо дважды запустить модель, чтобы достичь необходимой нормировки. Результаты можно видеть на графике выше в виде пунктирных линий. Видно, что значения для потерь и пропускания изменились лишь незначительно, возможно, это произошло в результате интерференции света, испускаемого непосредственно источником, и света, распространяющегося вокруг волновода.

## 2.3 Моды кольцевого резонатора

Как уже указывалось, одним из распространенных вариантов применения FDTD метода является расчет резонансных мод – их частот и скоростей затухания – для некоторой электромагнитной резонансной системы. В данном разделе будет дан пример соответствующего расчета для, вероятно, самого компактного из возможных диэлектрических резонаторов – кольцевого резонатора. Он представляет собой обыкновенный волновод, изогнутый в кольцо. Этот пример можно также найти в файле *examples/ring.cml*, который содержится в пакете Meep. Отметим, что из-за цилиндрической симметрии такой системы значительно эффективнее строить геометрию модели в цилиндрических координатах, однако для наглядности мы начнем с более привычной 2D – геометрии.

### 2.3.1 Исследование резонансных мод кольцевого резонатора в 2D-геометрии

Прежде всего, определим несколько параметров для описания геометрии системы, для того чтобы с легкостью вносить в нее изменения:

**(define-param n 3.4) ; индекс рефракции волновода**

**(define-param w 1) ; ширина волновода**

**(define-param r 1) ; внутренний радиус кольца**

**(define-param pad 4) ; расстояние между волноводом и границей PML-слоя**

**(define-param dpml 2) ; толщина PML-слоя**

**(define sxy (\* 2 (+ r w pad dpml))) ; размер области**

**(set! geometry-lattice (make lattice (size sxy sxy no-size)))**

Здесь используются относительные единицы измерения геометрии и префиксная (польская) нотация для написания арифметических выражений. Meep дает возможность описывать цилиндры, сферы, эллипсоиды и конусы, наряду с возможностью задания диэлектрической функции пользователем. В данном примере мы будем использовать два цилиндрических объекта – один внутри другого:

**(set! geometry (list**

**(make cylinder (center 0 0) (height infinity)**

**(radius (+ r w)) (material (make dielectric (index n))))**

**(make cylinder (center 0 0) (height infinity)**

**(radius r) (material air))))**

**(set! pml-layers (list (make pml (thickness dpml))))**

**(set-param! resolution 10)**

Каждый последующий объект в списке получает приоритет перед предыдущим (располагается поверх него). Таким образом второй воздушный ( $\epsilon = 1$ ) цилиндр вырезает круговое отверстие в большем цилиндре, формируя кольцо толщины  $w$ . Поскольку невозможно узнать частоты, соответствующие модам,

заблаговременно, то далее в систему подается широкий Гауссов импульс, который возбуждает все (ТМ-поляризованные) моды в выбранной полосе частот:

```
(define-param fcen 0.15) ; центральная частота импульса
(define-param df 0.1) ; ширина импульса ( по частоте )
(set! sources (list
  (make source
    (src (make gaussian-src (frequency fcen) (fwidth df)))
    (component Ez) (center (+ r 0.1) 0))))
```

Теперь все готово для запуска модели. Основная идея в том, чтобы расчетный метод работал, пока не затухнет излучение источника, и еще некоторое время после этого. В течение этого времени после затухания источника мы выполним необходимую обработку полученных данных с помощью пакета *harminv*. В результате этой обработки будут получены значения частот и скоростей затухания возбужденных мод:

```
(run-sources+ 300
  (at-beginning output-epsilon)
  (after-sources (harminv Ez (vector3 (+ r 0.1)) fcen df)))
```

Обработка сигнала выполняется посредством функции *harminv*, принимающей 4 аргумента: компонента поля ( здесь *Ez* ), положение рабочей точки ( здесь  $(r + 0.1, 0)$  ) и диапазон частот, заданный через центральную частоту и ширину полосы ( здесь то же, что и для импульса источника ). Обратите внимание на то, что команда *harminv* заключена в блок *(after-sources ...)*, поскольку анализ частот производится в системе без излучающих источников ( их наличие исказило бы результаты ). По окончании вычислений *harminv* выводит на экран набор строк ( начинающихся с *harminv0* : для удобства работы Unix-команды *grep* ), в которых перечисляются найденные частоты:

```
harminv0: frequency, imag. freq., Q, |amp|, amplitude, error
harminv0: 0.118101575043663, -7.31885828253851e-4, 80.683059081382,
0.00341388964904578, -0.00305022905294175-0.00153321402956404i,
1.02581433904604e-5
harminv0: 0.147162555528154, -2.32636643253225e-4, 316.29272471914,
0.0286457663908165, 0.0193127882016469-0.0211564681361413i,
7.32532621851082e-7
harminv0: 0.175246750722663, -5.22349801171605e-5, 1677.48461212767,
0.00721133215656089, -8.12770506086109e-4-0.00716538314235085i,
1.82066436470489e-7
```

Шесть столбцов разделены запятыми, что упрощает импорт полученных данных в другие программы. Смысл данных, выведенных в эти столбцы, заключается в следующем. *Harminv* проводит анализ полей  $f(t)$  в заданной точке и представляет их как совокупность мод ( принадлежащих заданной полосе частот):

$$f(t) = \sum_n a_n e^{-i\omega_n t}$$

где  $a_n$  – комплексные амплитуды,  $\omega_n$  – комплексные частоты. Все 6 столбцов связаны с этими величинами: первый столбец содержит действительную часть  $\omega_n$ , выраженную в привычных нам единицах  $2\pi c$ , второй столбец содержит мнимую часть  $\omega_n$  – отрицательную величину, соответствующую экспоненциальному затуханию. Для резонаторов скорость затухания чаще выражается как безразмерное «время жизни»  $Q$ , определяемое как

$$Q = \frac{\text{Re } \omega}{-2\text{Im } \omega}$$

( $Q$  равняется числу оптических периодов, за которые мощность излучения падает в  $\exp(-2\pi)$  раз, а величина  $1/Q$  – относительная ширина полосы частот на половине значения максимума резонансного пика в Фурье-области). Это значение содержится в третьем столбце выходного файла. Четвертый и пятый столбцы содержат абсолютное значение  $|a_n|$  и комплексную величину амплитуд  $a_n$ . Последний столбец содержит примерное значение ошибки вычисления частот – как действительной, так и мнимой их частей. Если величина ошибки значительно превосходит, к примеру, мнимую часть, то нельзя полагаться на точность полученного  $Q$ . Отметим, что данная величина ошибки обусловлена неточностью обработки сигнала процедурой *harminv*, она не имеет никакого отношения к погрешностям, вызванным конечным разрешением, конечным размером ячейки дискретизации и т. д.

Возникает вопрос: на какое время необходимо продлить вычисления после прекращения излучения источников, чтобы произвести необходимый расчет частот? В традиционном Фурье-анализе это время пропорционально требуемому разрешению по частоте. Однако, использование *harminv* сокращает длину необходимого временного отрезка. В уже рассмотренном примере анализируются 3 моды. У последней из них  $Q = 1677$  – это означает, что мода затухает за примерно 2000 периодов или  $2000 / 0.175 = 10000$  единиц времени. Но анализ был проведен лишь на приблизительно 300 временных единиц и предполагаемая неточность в измерении частоты составила  $10^{-7}$  (как будет показано ниже, действительная величина ошибки будет здесь иметь порядок  $10^{-6}$ ). В общем случае следует увеличивать время прогона модели для достижения большей точности и для увеличения значений  $Q$ , но нет нужды делать его очень большим. Отметим, что, исходя из предыдущего опыта, удаётся получать моды с  $Q = 10^9$ , проводя расчеты для лишь 200 периодов. В данном случае, в заданной полосе частот были найдены три моды – на частотах 0.118, 0.147, 0.175 со значениями  $Q$  соответственно 81, 316, 16777. Это соответствует теоретическим оценкам, согласно которым значение  $Q$  кольцевого резонатора растет экспоненциально с ростом произведения  $\omega$  на радиус резонатора. Теперь предположим, что требуется построить картину электромагнитных полей соответствующим этим модам. Это не составляет проблемы – достаточно перезапустить модель с узкополосным источником излучения на каждой моде и после этого вывести результат.

В частности, чтобы вывести значение поля в конце, можно добавить аргумент (*at - end output - efield -z*) к функции *run-sources+*. Однако в этом случае может возникнуть проблема. Если нам не повезет, то в момент вывода значение



поля  $E_z$  может быть близко к 0, т.к. вся энергия в этот момент времени сосредоточена в магнитном поле. Тогда полученная картина будет малоинформативна. Вместо этого по завершении работы модели будем выводить 20 снимков картины поля за полный период  $1/f_{\text{ср}}$ . С этой целью добавим команду:

```
(run-until (/ 1 fср) (at-every (/ 1 fср 20) output-efield-z))
```

Теперь можно получить требуемые моды, запустив программу:

```
unix% meep fср=0.118 df=0.01 ring.ctl
```

после каждого цикла выполнения команд будем конвертировать картину поля в PNG-изображения и далее – в анимированный GIF-файл:

```
unix% h5topng -RZc dklbluered -C ring-eps-000000.00.h5 ring-ez-*.h5  
unix% convert ring-ez-*.png ring-ez-0.118.gif
```

На рис. 13 приведены полученные при этом изображения резонансных мод, соответствующие, слева направо, частотам 0.118, 0.147 и 0.175. на них отчетливо видно радиально распространяющееся излучение, приводящее к потерям.

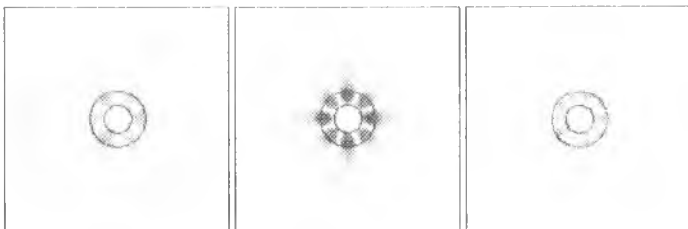


Рис. 13 Моды кольцевого резонатора

Отметим, что при прогоне модели с узкополосным источником *harminv* выдает несколько иные значения частот и  $Q$ , а также меньший порядок ошибки. В этом нет ничего удивительного, поскольку при возбуждении одной единственной моды сигнал получается более чистый, т.е. более удобный для точного анализа. К примеру, приведем результаты анализа для случая узкополосного источника на моде  $\omega = 0.175$ :

```
harminv0:, 0.175247426698716, -5.20844416909221e-5, 1682.33949533974,  
0.185515412838043, 0.127625313330642-0.13463932485617i,  
7.35320734698267e-12
```

Эти значения отличаются от предыдущих приблизительно на  $0.000001$  ( $10^{-6}$ ). При этом различие больше для значений  $Q$ , что естественно: небольшая абсолютная погрешность в  $\omega$  приводит к большей относительной погрешности в значении мнимой частоты, которое само по себе мало.

### 2.3.2 Использование симметрии кольцевого резонатора.

В рассматриваемом случае имеет место зеркальная симметрия, характерная как для источника излучения, так и для структуры. И ее можно использовать в целях ускорения вычислений. А именно, мы лишь добавим во входной файл одну строку после определения источников ( *sources* ), оставив все остальное без изменений:

```
(set! symmetries (list (make mirror-sym (direction Y))))
```

По этой команде Меер вводит в системы зеркальную симметрию, плоскость которой проходит через начало координат и перпендикулярно оси *y*. При этом Меер не проверяет, действительно ли в системе есть заданная симметрия. Таким образом, следует определять ее лишь в том случае, когда и источники, и сама структура действительно обладают данной симметрией.

Все прочее в модели остается неизменным: также возможно получать значения полей в любой точке, в выходном файле по-прежнему содержится результат обработки всей области кольца, и выходные данные *harminv* тоже неизменны. Однако, промежуточные вычисления, проводимые Меер, теперь затрагивают лишь половину структуры и поэтому занимают примерное вдвое меньшее время.

### 2.3.3 Кольцевой резонатор в цилиндрических координатах

Итак, мы вычислили моды кольцевого резонатора, выполнив двухмерное моделирование. Теперь мы будем моделировать ту же структуру в цилиндрических координатах, используя тот факт, что у системы есть непрерывная вращательная симметрия. Также можно использовать файл *ring-cyl.cil.*, который содержится в пакете Меер.

Начнем, как обычно, с задания параметров задачи теми же значениями, что и в 2-мерном моделировании.

```
(define-param n 3.4) ; index of waveguide  
(define-param w 1) ; width of waveguide  
(define-param r 1) ; inner radius of ring  
(define-param pad 4) ; padding between waveguide and edge of PML  
(define-param dpml 2) ; thickness of PML
```

Теперь определим размерность и объем рабочей области:

```
(define sr (+ r w pad dpml)) ; радиальный размер ( ячейка от 0 до sr )  
(set! dimensions CYLINDRICAL)  
(set! geometry-lattice (make lattice (size sr no-size no-size)))
```

Таким образом, мы задаем параметр размерностей как ЦИЛИНДРИЧЕСКИЙ. Это означает, что вместо координат (*x*, *y*, *z*) все векторы будут представлены в виде (*r*, $\phi$ ,*z*). Вычисляемая ячейка в направлении *r* имеет размер *sr* = *r* + *w* + *pad* + *dpml*, и (по умолчанию) принимает значения от 0 до *sr*, а не от  $-sr/2$  до *sr/2*, как было бы в любом другом измерении. Отметим, что размер

вдоль оси  $z$  имеет значение  $no-size$ , т.к. задача двумерная. Переменная  $\phi$  также безразмерная ( $no-size$ ), что соответствует непрерывной вращательной симметрии (конечный размер  $\phi$  соответствовал бы дискретной вращательной симметрии, но в данный момент она не поддерживается Meep).

Известно, что в системах с непрерывной вращательной симметрией аналогично теореме Блоха угловая зависимость полей всегда может быть представлена в виде  $\exp(im\phi)$ , где  $m$  – некоторое целое число. Meep использует этот факт для аналитического рассмотрения угловой зависимости, где  $m$  определяется входной переменной  $m$  (пока установим её равной 3).

```
(set-param! m 3)
```

Таким образом, мы фактически используем одномерную геометрию, в которой Meep нужно дискретизировать только направление  $z$ . Поэтому вычисления будут выполнены быстрее, чем предыдущее 2-мерное моделирование.

Теперь геометрия задана единственным объектом – блоком. Важно понять, что блоком он является только в цилиндрических координатах, на самом деле это круглое кольцо:

```
(set! geometry (list  
  (make block (center (+ r (/ w 2))) (size w infinity infinity)  
    (material (make dielectric (index n))))))  
(set! pml-layers (list (make pml (thickness dpml))))  
(set-param! resolution 10)
```

Мы добавили поглощающие электромагнитное излучение PML (Perfect Matched Layer) слои со “всех” сторон. Однако, Meep замечает, что в направлении  $z$  толщина моделируемого объекта равна бесконечности, поэтому автоматически делает границу периодичной без PML.

Остальные команды `ctl`-файла представляют собой почти то же самое, что в предыдущем 2-хмерном моделировании. Поместим точечный Гауссовский точечный источник на оси  $z$  чтобы вызывать ТМ колебания определенной центральной частоты и ширины полосы:

```
(define-param fcen 0.15) ; центральная частота импульса  
(define-param df 0.1) ; ширина импульса (по частоте)  
(set! sources (list  
  (make source  
    (src (make gaussian-src (frequency fcen) (fwidth df))  
      (component Ez) (center (+ r 0.1) 0 0))))
```

Отметим, что на самом деле из-за цилиндрической симметрии это не “точечный” источник. Это “кольцевой” источник с зависимостью от  $\phi$  в виде  $\exp(im\phi)$ . Наконец, как и ранее, мы запускаем моделирование до момента выключения источника и, затем, ждем еще 200 временных единиц: в течение которых мы изучаем поле  $E_z$  (с помощью `harminv`) в заданной точке, чтобы получить значения частот и скоростей затухания волн.

```
(run-sources+ 200 (after-sources (harminv Ez (vector3 (+ r 0.1)) fcen df)))
```

В самом конце мы также выведем поля в течение одного временного периода, чтобы сделать анимацию и т.д. Значения одного поля будут записаны в одномерный массив данных (вдоль направления  $r$ ), поэтому для большей наглядности мы используем команду *to-appended*, чтобы соединить все эти данные в один файл HDF5 и получить 2-мерный массив  $r \times t$ . Также мы будем использовать команду *in-volume*, чтобы задать больший объем выходных данных, чем рабочая область; в частности, мы выведем данные от  $-sr$  до  $sr$  вдоль направления  $r$ . Меер автоматически выведет значения поля для отрицательных  $r$  исходя из симметрии отражения.

```
(run-until (/ 1 fcen)
  (in-volume (volume (center 0) (size (* 2 sr)))
    (at-beginning output-epsilon)
    (to-appended "ez"
      (at-every (/ 1 fcen 20) output-cfield-z))))
```

Теперь мы готовы начать моделирование. Напомним, что при 2-мерном вычислении у нас получились три волны: одна при  $\omega = 0.11785$ ;  $Q = 77$  и  $m = 3$ ; другая при  $\omega = 0.14687$ ;  $Q = 351$  и  $m = 4$ ; последняя при  $\omega = 0.17501$ ;  $Q = 1630$  и  $m = 5$ . Теперь мы должны получить те же моды (с небольшими различиями из-за разрешения); кроме того, придется запустить три расчета, по одному на каждое значение  $m$  (Это будет намного быстрее, чем раньше, поскольку моделирование 1-мерное, а не 2- мерное).

В итоге, мы запустим:

```
unix% meep m=3 ring-cyl.ctl | grep harminv
unix% meep m=4 ring-cyl.ctl | grep harminv
unix% meep m=5 ring-cyl.ctl | grep harminv
```

что даст в результате:

```
harminv0:, frequency, imag. freq., Q, |amp|, amplitude, error
harminv0:, 0.11834848194079, -6.80930025762674e-4, 86.9020879261668,
0.257477542991357, -0.234862526831655-0.105519091330034i,
2.6465657298186e-10
harminv0:, 0.147555705534808, -1.91078761299536e-4, 386.112262114517,
1.93737432741834, 1.35411847722594+1.38556213652616i, 2.73521325130449e-
11
harminv0:, 0.175944214054996, -4.82976799119763e-5, 1821.45616907125,
0.45258172336278, -0.107884449861492-0.439535165601237i,
1.2656772930993e-10
```

Действительно, очень похоже на 2-хмерное моделирование: расхождения частот в пределах 1%. Значения  $Q$  (время жизни) различаются на большую величину (хотя и они довольно близки друг к другу).

Так что же точнее, 2-мерное или цилиндрическое моделирование? На этот вопрос можно ответить, повысив разрешение для обоих случаев и проанализировав сходимость результата. Остановимся на моде при  $m=4$ . Для цилиндрического моделирования, если удвоить разрешение до 20, то получим  $\omega = 0.14748$  и  $Q = 383$ . Для 2-мерного моделирования, если удвоить разрешение до 20, то получим  $\omega = 0.14733$  и  $Q = 321$ . Похоже, что значения частот сходятся, при этом, цилиндрическое моделирование точнее. Этого можно было ожидать, учитывая, что направление  $\phi$  описывается аналитически. Но значения  $Q$  становятся более удаленными. В чем же дело?

У данной проблемы две причины. **Во-первых**, в процессе обработки сигнала для определения  $Q$  при двумерном моделировании есть некоторая ошибка: как показано в колонке «его», погрешность для двумерного моделирования 4с-7, в то время, как для цилиндрического 3е-11. Можно снизить ошибку, проведя моделирование для источника с более узкой шириной полосы, который возбуждает лишь одну волну и дает более четкий сигнал, или проведя исследования в течении более долгого времени, чем 200 единиц. Выполнив это, мы обнаружим, для двумерного моделирования значение  $Q$  при разрешении 20 действительно должно быть равным  $Q = 343$ . **Во-вторых**, поглощающие PML слои созданы для поглощения плоских волн на плоских поверхностях; здесь же у нас цилиндрический PML слой. Из-за этого при цилиндрическом моделировании наблюдается большее число отражение от PML, что можно исправить, выставив для PML больший радиус (т.е. используя большее значение  $pad$ ) и/или увеличив толщину PML (увеличивая  $d_{pml}$ ). Для цилиндрического моделирования при разрешении 20 если увеличить  $d_{pml}$  до 16, то получим  $Q=342$ , что куда лучше согласуется с 2-мерными расчетами (а если увеличить  $d_{pml}$  до 32, то также получится  $Q=342$ ; таким образом, эти значения сходятся).

Этот анализ иллюстрирует то, что при использовании метода FDTD необходимо контролировать одновременно несколько параметров (разрешение, время моделирования, толщина PML и т.д.), чтобы гарантировать сходимость результатов во временной области.

Наконец, мы можем получить изображения полей. Так как, согласно *harminv*, для каждого значения  $m$  мы возбуждаем одну волну, то не требуется использовать узкополосный источник. Однако мы сделали это для того, чтобы напомнить общую процедуру, например, для  $\omega = 0.118$   $m = 3$ :

```
unix% meep m=3 fcen=0.118 df=0.01 ring-cyl.ctl
unix% h5topng -S 2 -Zc dkbluered -C ring-cyl-eps-001200.00.h5 ring-cyl-
ez.h5
```

Отметим, что в результате работы команды *to-appended* файл *ing-cyl-ez.h5* представляет собой массив  $160 \times 18$ , соответствующий  $r \times t$  слою. Повторив это для всех трех волн, получим изображения полей (рис. 14).

Поскольку мы рассматриваем слои с  $\phi = 0$ , визуальное различие между полями с различными значениями  $m$  куда меньше, чем при 2-мерном моделировании. Очевидно то, что с ростом частоты волны все больше ограничиваются волноводом, а излучающее поле (изогнутые волны на слое, направленные парежу) становится меньше, как и предполагалось.

$E_z$  при моде  $\omega = 0.118 \quad m = 3$



$E_z$  при моде  $\omega = 0.148 \quad m = 4$



$E_z$  при моде  $\omega = 0.176 \quad m = 5$



Рис. 14 Моды кольцевого резонатора в цилиндрических координатах

## 2.4 Дисперсия света в веществе

В этом разделе рассмотрим оптическое моделирование материала с зависимостью диэлектрической проницаемости от частоты  $\epsilon = \epsilon(\omega)$ . В частности, мы промоделируем однородную среду, состоящую из такого вещества. Файл *material-dispersion.cil*, содержащий данный пример, прилагается к пакету Меер. Из дисперсионной зависимости  $\omega(k)$  рассчитаем численную зависимость  $\epsilon(\omega)$ , воспользовавшись формулой

$$\epsilon(\omega) = \left( \frac{ct}{v(\omega)} \right)^2$$

В дальнейшем мы проведем сравнение значений  $\epsilon(\omega)$ , заданных аналитически, с вычисленными по этой формуле.

Поскольку рассматривается однородная среда, то рабочая область может быть нулевого размера (т.е. содержать лишь 1 пиксель). В ней будут использоваться периодические по Блоху граничные условия для задания волнового вектора  $k$ .

```
(set! geometry-lattice (make lattice (size no-size no-size no-size)))  
(set-param! resolution 20)
```

Затем заполним пространство дисперсионным материалом:

```
(set! default-material  
  (make dielectric (epsilon 2.25)  
    (E-polarizations  
      (make polarizability  
        (omega 1.1) (gamma 1e-5) (sigma 0.5))  
      (make polarizability  
        (omega 0.5) (gamma 0.1) (sigma 2e-5))  
    )))
```

Это соответствует диэлектрической функции

$$\epsilon(\omega) = 2.25 + \frac{1.1^2 \cdot 0.5}{1.1^2 - \omega^2 - i\omega \cdot 10^{-5}} + \frac{0.5^2 \cdot 2 \cdot 10^{-5}}{0.5^2 - \omega^2 - i\omega \cdot 0.1}$$

Важно обратить внимание на то, что в Меер единицей измерения частоты  $\omega$  является  $2\pi c/a$ , поэтому в действительности приведенное выражение представляет собой следующее:

$$\epsilon(f) = 2.2 + i \frac{1^2 \cdot 0.5}{(f - f_0)^2 + \gamma^2/4} + \frac{0.7^2 \cdot 2 \cdot 10^{-3}}{0.7^2 - f^2 - (f \cdot 0.1) \cdot 2\pi}$$

На рис. 15 приведены графики действительной и мнимой частей заданной диэлектрической функции:

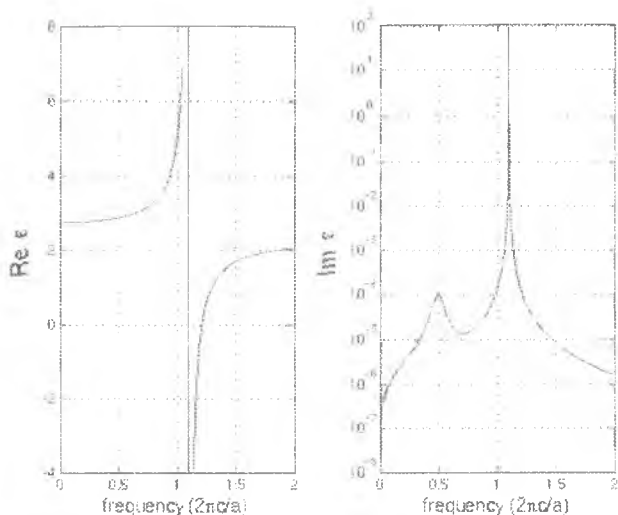


Рис. 15 Действительная и мнимая части рассматриваемой диэлектрической функции

Можно видеть, что резонанс при значении  $\omega = 1.1$  вызывает значительные изменения как в действительной, так и мнимой частях функции в окрестности этой частоты. Фактически, имеется диапазон частот от 1.1 до 1.2161, при которых значение  $\epsilon$  отрицательно. В этом интервале не существует ни одной распространяющейся волны (моды). Мы имеем здесь дело с подобием фотонной запрещенной зоны, связанной с поляритонным резонансом в веществе.

С другой стороны, резонанс при значении  $\omega = 0.5$ , из-за малости параметра  $\sigma$  приводит к лишь небольшим изменениям в действительной части  $\epsilon$ . И, тем не менее, в мнимой части при этом наблюдается отчетливый пик, соответствующий пику резонансного поглощения.

Теперь перейдем к определению оставшейся части модели. Зададим широкополосный ТМ-поляризованный Гауссов источник; зададим ряд значений  $k$ , для которых будем рассчитывать  $\omega(k)$  и вычислим соответствующие частоты с помощью функции `run - k - points`:

```
(define-param fcen 1.0)
(define-param df 2.0)
(set! sources (list (make source
```

```
(src (make gaussian-src (frequency fcen) (fwidth df)))  
(component Ez) (center 0 0 0))))
```

```
(define-param kmin 0.3)  
(define-param kmax 2.2)  
(define-param k-interp 99)  
(define kpts (interpolate k-interp (list (vector3 kmin) (vector3 kmax))))
```

```
(define all-freqs (run-k-points 200 kpts)) ; перечень списков частот.
```

Функция *run-k-points* возвращает список списков частот – один список значений ( комплексных ) частот для каждого значения  $k$  – который сохраняется в переменной *all-freqs*. Далее этот список вводится в цикл, чтобы вывести на экран полученные значения  $\epsilon$  ( используется соотношение  $(ck / \omega)^2$ , см. выше ). Для этого воспользуемся функцией *map* языка Scheme, которая применяет заданную операцию ( функцию ) ко всем элементам списка ( или списков ). А поскольку мы работаем со списком списков, то потребуется дважды использовать эту функцию:

```
(map (lambda (kx fs)  
      (map (lambda (f)  
            (print "eps:", " (real-part f) ", " (imag-part f)  
              ", " (sqrt (/ kx f)) "\n")  
            fs))  
      (map vector3-x kpts) all-freqs)
```

В качестве альтернативы можно просто передать все данные в Matlab или какую-либо электронную таблицу ( spreadsheet ) и далее там проводить вычисления. После запуска программы посредством команды

```
unix% meep material-dispersion.ctf | tee material-dispersion.out
```

мы можем просмотреть полученные значения частот и диэлектрической функции с помощью команды *gter* и построить график их зависимости. В первую очередь, построим дисперсионное соотношение  $\omega(k)$  (для действительной части  $\omega$ ) (рис. 16).

Здесь красные кружки соответствуют значениям, рассчитанным Meep, тогда как голубая линия – аналитическая зонная диаграмма по заданным значениям  $\epsilon(\omega)$ . Нетрудно видеть, что для каждого значения  $k$  получены 2 области, разделенные поляритонным (polaritonic) промежутком (желтая область).

Аналогичным образом определяются аналитическая и вычисленная программой Meep действительные части диэлектрической функции (рис. 17).

Отметим отличное соответствие аналитических (голубая линия) и просчитанных программой (красные кружки) данных. Однако, подобный график для мнимой части выглядит менее однозначно (рис. 18).

Здесь, опять же, голубая линия, построенная по результатам приведенных выше аналитических вычислений, а красные кружки – численные значения,



полученные Меер. Но почему нет полного совпадения, как на предыдущем графике? Причиной является не ошибка вычислений Меер или какая-либо численная погрешность. Дело в том, что мы пытаемся сравнить несравнимое.

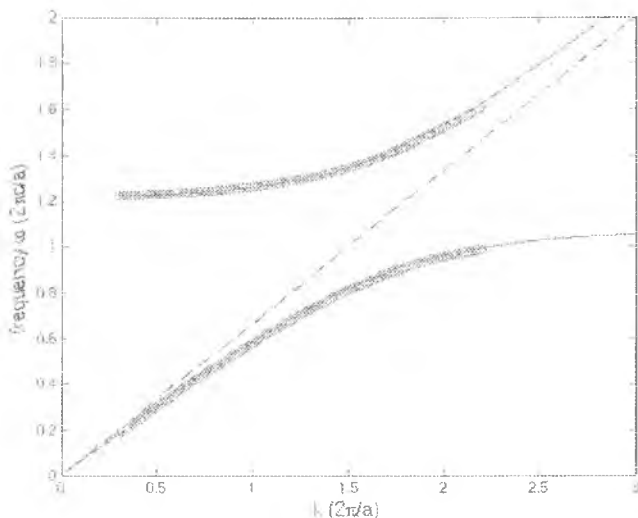


Рис. 16 Дисперсионное соотношение  $\text{Re}[\omega(k)]$

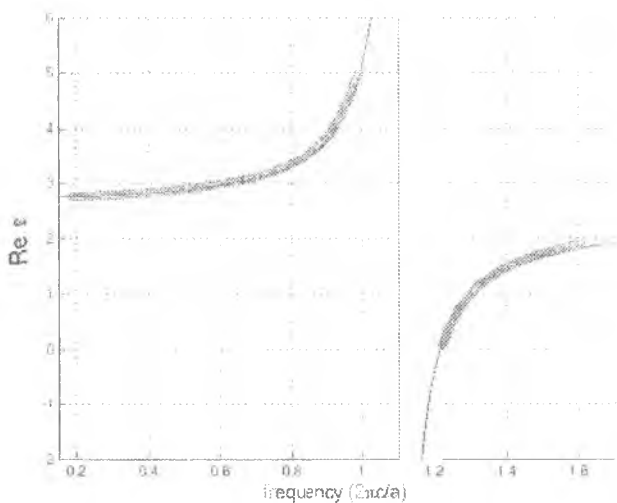


Рис. 17 Действительная часть диэлектрической функции  $\epsilon(\omega)$

Голубая линия соответствует аналитическому расчету  $\epsilon(\omega)$ , проведенному для действительной частоты  $\omega$  (что соответствует решениям с комплексным

значением волнового вектора  $k$ ), тогда как в Меер  $\epsilon$  вычислялось при комплексном  $\omega$  и действительном  $k$ . Таким образом, чтобы провести верное сопоставление, требуется подставить комплексное значение  $\omega$ , использованное Меер, в аналитическую формулу для  $\epsilon(\omega)$ . Результат отражен на графике в виде зеленых линий, которые лежат практически поверх красных кружков.

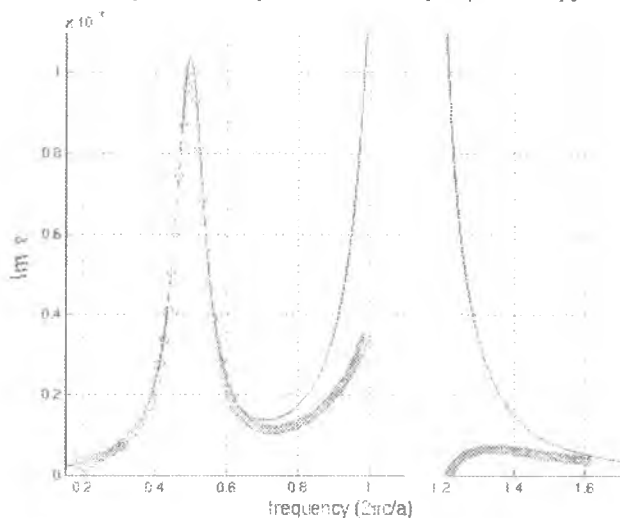


Рис. 18 Мнимая часть диэлектрической функции  $\epsilon(\omega)$

Почему, тем не менее, был получен такой превосходный результат при сопоставлении действительных частей  $\epsilon$ ? Причина кроется в том, что как при действительных, так и при мнимых  $\omega$  значения  $\epsilon(\omega)$  связаны между собой аналитическими свойствами величины  $\epsilon$ . В частности, поскольку  $\epsilon$  - аналитическая функция на действительной оси  $\omega$ , то введение *малой* мнимой части в  $\omega$ , проделанное здесь, не внесло больших поправок (отклонения незначительны при всех значениях  $k$ ) в значение  $\epsilon$ . Значительным оказалось лишь изменение мнимой части  $\epsilon$ .

### 3. Моделирование распространения световых полей с использованием дифракционных интегралов

В связи с уменьшением размеров оптических устройств большое внимание в последнее время уделяется описанию непараксиального распространения световых полей и разработке алгоритмов моделирования такого распространения.

Дифракция на круглой апертуре плоской волны является тестовым примером для разрабатываемых алгоритмов как случай, имеющий аналитическое решение на оптической оси.

Непараксиальная скалярная модель, основанная на теории Рэлея-Зоммерфельда [1], позволяет получать согласующиеся с экспериментами результаты на очень близких расстояниях от апертуры. В частности, для случая дифракции на цилиндрических тонких фазовых объектах (что практически соответствует скалярной задаче дифракции на щели) было показано, что в зависимости от фазового набега различные типы дифракционных интегралов – Кирхгофа, Рэлея-Зоммерфельда I или II типа – позволяют получить корректные результаты на расстоянии всего полдлины волны от объекта. На расстоянии более двух длин волн все типы интегралов дают одинаковый результат.

Однако при уменьшении поперечных размеров светового поля (или его характерных деталей) до размеров порядка длины волны также необходимо учитывать векторный характер светового поля. В этом случае применяют векторный вариант интегралов Рэлея-Зоммерфельда или метод разложения по плоским волнам.

При численной реализации интегралов Рэлея-Зоммерфельда сложно воспользоваться какой-либо симметрией световых полей, поэтому для упрощения расчетов или получения аналитических выражений приходится прибегать к различным аппроксимациям. Однако, как показывают расчеты, использование даже непараксиальных аппроксимаций в ближней зоне является некорректным, поэтому для расчетов в этой зоне необходимо использовать точные операторы распространения, а для ускорения расчетов применять высокопроизводительные технологии.

При использовании метода разложения по плоским волнам возможно применение алгоритма быстрого преобразования Фурье (БПФ), что значительно сокращает время расчета. Однако использование алгоритма БПФ имеет свои недостатки, связанные с фиксированной дискретностью сигналов на входе и выходе, а также возможностью вычислять только поперечные распределения. При необходимости получения продольных (вдоль оптической оси) распределений использовать алгоритм, основанный на БПФ, нецелесообразно: зависимость метрических размеров от  $z$  требует пересчета (интерполяции) отсчетов в каждой плоскости. А при необходимости получения значений на некоторой кривой, не лежащей в поперечной плоскости, преимущество БПФ по скорости теряется.

В данном пособии предлагается к рассмотрению программное обеспечение **FieldPropagators**, предназначенное для расчета поля в заданных областях различными операторами распространения в пространстве с постоянным показателем преломления.

Для удобства использования на кластере программа выполнена как консольное приложение. Входные данные задаются в текстовом файле на языке

описания задания. При запуске программы без параметров выдается краткое описание допустимых параметров.

В результате работы программы формируются:

- файл протокола (содержит входные параметры, имена выходных файлов и дополнительную информацию),
- выходные файлы изображений (в формате BMP и DAT),
- выходные файлы графиков (в текстовом формате),
- сообщения об ошибках, если таковые были.

Программное обеспечение **FieldPropagators** организовано так, что его удобно использовать для сравнения действия различных операторов распространения. Это позволит студентам оценить границы применимости различных приближений и аппроксимаций, используемых при моделировании дифракционных процессов.

### 3.1 Скалярные операторы распространения

В данном разделе рассматриваются скалярные операторы распространения, которые описывают распространение монохроматического поля с длиной волны  $\lambda$ , заданного в плоскости  $z=0$  комплексным распределением  $E_0(x, y)$  в пространстве с постоянным показателем преломления  $n$ .

Везде далее волновое число определяется по формуле:

$$k = \frac{\omega}{c} \sqrt{\mu\epsilon} = \frac{2\pi}{\lambda} n, \quad (1)$$

где  $\omega$  – частота излучения,  $c$  – скорость света в вакууме,  $\epsilon$  и  $\mu$  диэлектрическая и магнитная проницаемости среды, соответственно.

**Преобразование Фурье** соответствует распределению в фокальной плоскости сферической линзы:

$$G(u, v) = \frac{k}{f} \iint_{\Sigma} E_0(x, y) \exp\left[-\frac{ik}{f}(xu + yv)\right] dx dy \quad (2)$$

где  $f$  – фокусное расстояние,  $\Sigma$  – область задания входного поля.

**Преобразование Френеля** описывает параксиальное (близкое к оптической оси) распространение на расстояние  $z$ :

$$G(u, v, z) = -\frac{ik}{2\pi z} \exp(ikz) \iint_{\Sigma} E_0(x, y) \exp\left[\frac{ik}{2z}((x-u)^2 + (y-v)^2)\right] dx dy \quad (3)$$

где  $z$  – расстояние от входной плоскости.

**Интегральное преобразование Релея-Зоммерфельда первого типа** точно (без каких-либо аппроксимаций) описывает распространение на расстояние  $z$ :

$$E(u, v, z) = -\frac{z}{2\pi} \iint_{\Sigma} E_0(x, y) \frac{e^{ikt}}{\ell^2} \left( ik - \frac{1}{\ell} \right) dx dy, \quad (4)$$

где  $\ell = \sqrt{(u-x)^2 + (v-y)^2 + z^2}$

**Разложение по плоским волнам** точно (без каких-либо аппроксимаций) описывает распространение на расстояние  $z$ :

$$E(x, y, z) = \frac{1}{\lambda^2} \iint_{\Sigma} F(\xi, \eta) \exp\left[ikz\sqrt{1-(\xi^2 + \eta^2)}\right] \exp[ik(\xi x + \eta y)] d\xi d\eta, \quad (5)$$

$$\text{где } F(\xi, \eta) = \iint_{\Sigma} E_0(x, y) \exp[-ik(\xi x + \eta y)] dx dy \quad (6)$$

где  $F(\xi, \eta)$  – Фурье-спектр входного электрического поля  $E_0(x, y)$ , определенно-го в области  $\Sigma$ .

Спектр  $F(\xi, \eta)$  рассматривается в области  $\Sigma_s$ :  $\sigma_1 \leq \sqrt{\xi^2 + \eta^2} \leq \sigma_2$ .

Распространяющимся волнам соответствуют пространственные частоты, расположенные в круге радиусом  $\sigma_0 \leq 1$ . Чтобы учесть также и затухающие волны, вносящие свой вклад на расстояниях меньше длины волны, необходимо увеличивать радиус учитываемых пространственных частот до некоторого значения  $\sigma_1 > 1$ , зависящего от расстояния  $z$  от апертуры. При  $\sigma_1 = 0, \sigma_2 = 1$  учитываются только распространяющиеся волны, при  $\sigma_1 = 1, \sigma_2 > 1$  учитываются только затухающие волны.

Если значения пространственной переменной лежат в диапазоне  $x \in [-R_x, R_x]$  и дискретизация в объектной плоскости:  $h_x = \frac{2R_x}{N}$ , то с учетом условия Найквиста в спектральной плоскости частоты лежат в диапазоне  $\xi \in [-R_\xi, R_\xi]$ ,  $R_\xi = \frac{\lambda}{2h_x}$ .

Поэтому  $\max|\xi| = R_\xi$  и  $\max|\eta| = R_\eta$ . Если этот диапазон частот оказывается меньше необходимого, т.е.  $\sqrt{R_\xi^2 + R_\eta^2} < \sigma_2$ , то необходимо уменьшать шаг дискретизации входного поля.

Выражения (4) и (5)-(6) можно свести друг к другу, однако в последнем случае возможно применение различных быстрых алгоритмов, в том числе БПФ, что значительно сокращает время расчета, несмотря на удвоенное по сравнению (4) интегрирование.

Применение алгоритма БПФ имеет свои недостатки, связанные с фиксированной дискретностью сигналов на входе и выходе, а также возможностью вычислять только поперечные распределения. Кроме того, в работе [10] было указано на проблемы, возникающие при наличии наклона или смещения падающего пучка, для решения которых требуются дополнительные преобразования.

## 3.2 Примеры расчета в рамках скалярной теории

### 3.2.1 Дифракция плоской волны на круглой апертуре

Дифракция на круглой апертуре плоской волны может использоваться для начальной оценки точности рассматриваемых в данном пособии алгоритмов.

При дифракции плоской волны на круглой апертуре в скалярном случае известно аналитическое решение для значений интенсивности вдоль оптической оси:

$$I(0, 0, z) = 1 + \frac{z^2}{r_0^2 + z^2} - \frac{2z}{\sqrt{r_0^2 + z^2}} \cos\left[k\left(\sqrt{r_0^2 + z^2} - z\right)\right], \quad (7)$$

где  $r_0$  – радиус ограничивающей апертуры.

Для сравнения численных результатов с аналитическим выражением (7) были выполнены расчеты осевой интенсивности плоской волны, прошедшей через круглое отверстие радиусом  $r_0=10\lambda$ , на различных расстояниях с помощью интегрального преобразования Рэлея-Зоммерфельда (4) и разложения по плоским волнам (5), (6) при  $m=0$ .

Для получения значений на оптической оси или в продольном сечении применение алгоритма БПФ неэффективно, т.к. для этого потребуется вычислить столько поперечных сечений, сколько желательно получить значений вдоль оси распространения пучка.

На рис. 19 приведены сравнительные результаты вычисления осевой интенсивности в ближней зоне дифракции, из которых видно, что результаты обоих методов практически полностью совпадают для расстояний больших длины волны. Однако на расстояниях меньших длины волны метод разложения по плоским волнам несколько отличается от аналитического выражения, особенно при  $z < 0,1\lambda$ . По формулам (5), (6) для получения корректных результатов при  $z=0,1\lambda$  необходимо учитывать пространственные частоты вплоть до  $\sigma_0=8$ , а при  $z=0,01\lambda$  д.б.  $\sigma_0 > 79$ , что резко увеличивает временные затраты, т.к. в этом случае необходимо также увеличивать дискретизацию входного поля.

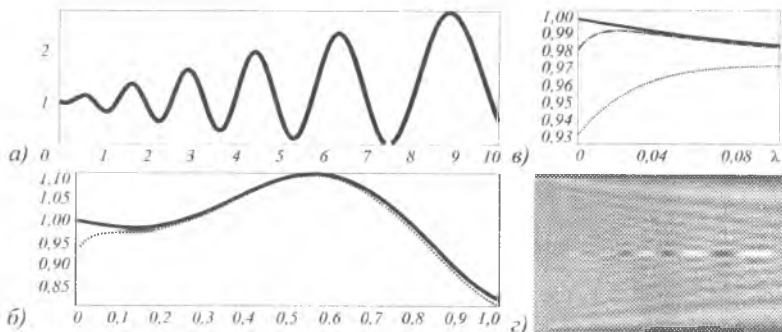


Рис. 19 Сравнение распределения осевой интенсивности, рассчитанной с помощью (1.1) (пунктирная линия) и (1.6) (точечная линия), с аналитическим выражением (1.3) (черный цвет сплошная линия): (а) на отрезке  $z \in [0,1\lambda, 10\lambda]$  при  $\sigma_0=3$ , (б) на отрезке  $z \in [0,01\lambda, 1\lambda]$  при  $\sigma_0=5$  и (в) на отрезке  $z \in [0,01\lambda, 0,1\lambda]$  при  $\sigma_0=5$  (точечная линия) и  $\sigma_0=20$  (штрихпунктирная линия); (г) фокусирующее поведение круглой апертуры

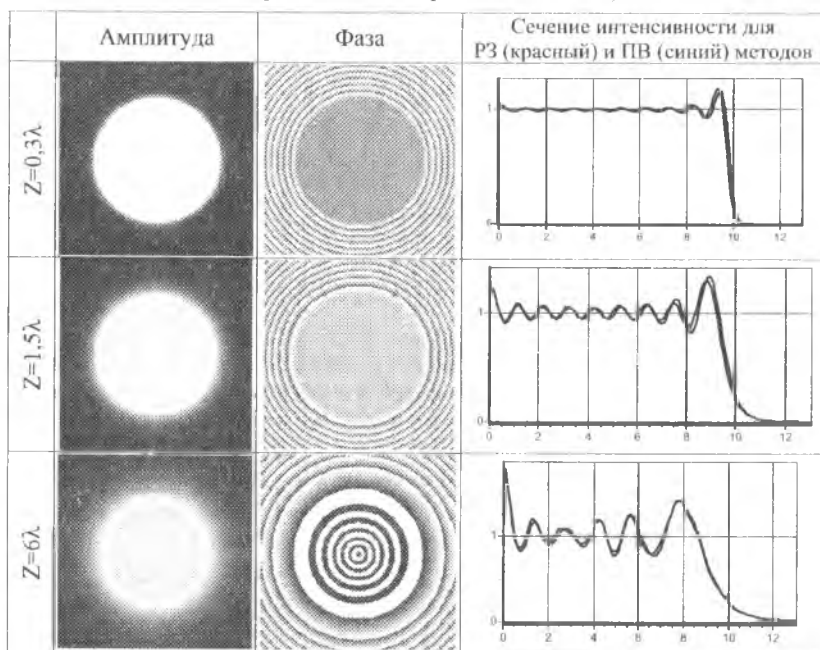
Для получения результатов, приведенных на рис. 19 в верхний предел учитываемых спектральных частот был принят  $\sigma_0=20$ , что в совокупности с уменьшением шага дискретизации привело к увеличению расчетного времени и превысило время прямого расчета по формуле (4). Заметим, что для получения корректных результатов в выражении (4) при  $z < 0,1\lambda$  понадобилась на порядок меньшая степень дискретизации, чем при использовании формул (5), (6). Таким образом, в данном случае прямой расчет по формуле (4) оказывается предпочтительным.

На рис. 19г показано внесосевое продольное распределение интенсивности при дифракции плоской волны на круглой апертуре в области  $z \in [0, 1\lambda, 10\lambda]$ ,  $x \in [-10\lambda, 10\lambda]$ , рассчитанное по формулам (5), (6). На расстояниях, больших одной десятой длины волны, алгоритм разложения по плоским волнам, учитывающий радиальную симметрию задачи, полностью совпадает с аналитическим решением и значительно опережает по скорости алгоритм.

Радиальная реализация метода разложения ПВ удобнее БПФ. Удобство заключается в том, что данная реализация позволяет с произвольной дискретизацией (не связанной с бинарной системой) получать распределения на любых поверхностях за малое время. Недостатком является требование наличия радиальной симметрии или вихревой угловой зависимости входного пучка.

Более детально внесосевое распределение интенсивности в поперечных направлениях распространения плоскостях на различных расстояниях от апертуры показано в Табл. 1. Среднеквадратичное отклонение в расчетах, получаемых двумя реализованными методами составляет менее 5%.

Таблица 1. Распределение интенсивности в поперечных направлениях распространения плоскостях на различных расстояниях от апертуры (поперечная область расчета  $30 \times 30 \lambda^2$ )



Таким образом, каждый из методов и их реализаций обладает достоинствами и недостатками. Относительно медленный алгоритм прямого расчета интеграла РЗ играет важную роль на очень близких расстояниях от апертуры, а также как

универсальный по отношению к типу падающей волны и форме апертуры тестовый инструмент. Метод разложения ПВ, реализованный через БПФ является самым быстрым, хотя и требующим много памяти, и также универсален. Радиальная реализация метода разложения ПВ характеризуется высокой скоростью расчета, очень экономична в требованиях к свободной памяти и удобна для расчетов распределений на любых поверхностях, но круг решаемых задач ограничен.

### 3.2.2 Дифракция вихревого пучка на круглой апертуре.

Рассмотрим дифракцию вихревого пучка на круглой диафрагме радиуса  $r_0$ :

$$E_0(r, \varphi) = E_0(r) \exp(im\varphi) \text{circ}(r/r_0), \quad (8)$$

где  $(r, \varphi)$  – полярные координаты во входной плоскости,  $\text{circ}(r/r_0) = \begin{cases} 1, & r \leq r_0, \\ 0, & r > r_0. \end{cases}$

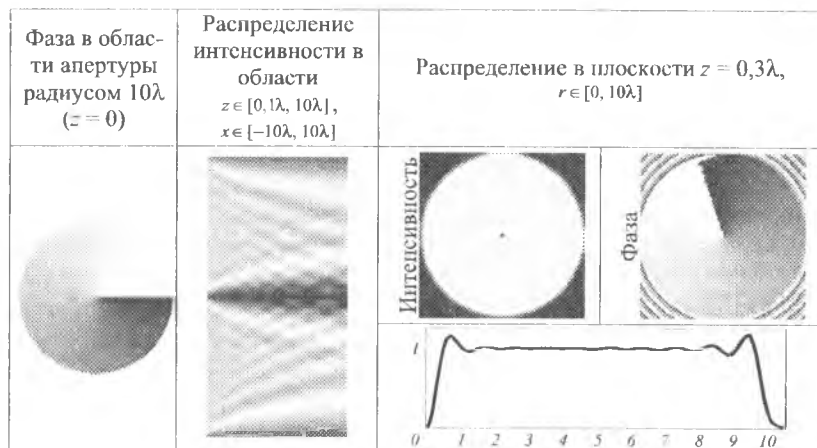
Как правило, дифракция на спиральной фазовой пластинке рассматривается в приближении тонкого оптического элемента, что фактически эквивалентно дифракции пучка с вихревой фазовой составляющей на апертуре того же радиуса соответствующего дифракционного оптического элемента.

При дифракции вихревого пучка с постоянной амплитудой на микроапертуре получить удобные аналитические выражения не удаётся.

В скалярной модели на оптической оси в этом случае будет нулевое значение интенсивности.

В табл. 2 показаны результаты моделирования в рамках скалярной непараксиальной модели дифракции вихревого пучка, имеющего порядок вихря  $m = 1$ , с постоянной амплитудой на круглой апертуре радиусом  $10\lambda$  в области  $z \in [0, 1\lambda, 10\lambda]$ ,  $x \in [-10\lambda, 10\lambda]$ .

Таблица 2. Распределение в ближней зоне дифракции на круглой апертуре вихревого пучка ( $m=1$ ) с постоянной амплитудой



На графике в табл.2 приведены сечения интенсивности в плоскости  $z = 0,3\lambda$ , полученные при использовании формулы (4) (сплошная линия) и (5), (6) (точечная линия).



### 3.2.3 Дифракция плоской волны на бинарном осесимметричном аксиконе.

Фазовый дифракционный аксикон имеет комплексную функцию пропускания следующего вида:

$$\tau(r) = \exp(ik\alpha_0 r), \quad (9)$$

где  $k = 2\pi/\lambda$  – волновое число,  $\lambda$  – длина волны,  $\alpha_0$  – параметр аксикона, определяющий числовую апертуру аксикона  $\alpha_0 = NA$ .

Рассмотрим высокоапертурный ( $\alpha_0 = 1$ ) бинарный микроаксикон вида:

$$\tau_c(r) = \exp\{i \arg[\cos(kr)]\}, \quad (10)$$

который будет использоваться для преобразования излучения в дальнем инфракрасном диапазоне  $\lambda = 10,6$  мкм, радиусом  $R = 2,25\lambda = 23,85$  мкм (см. рис. 20а).

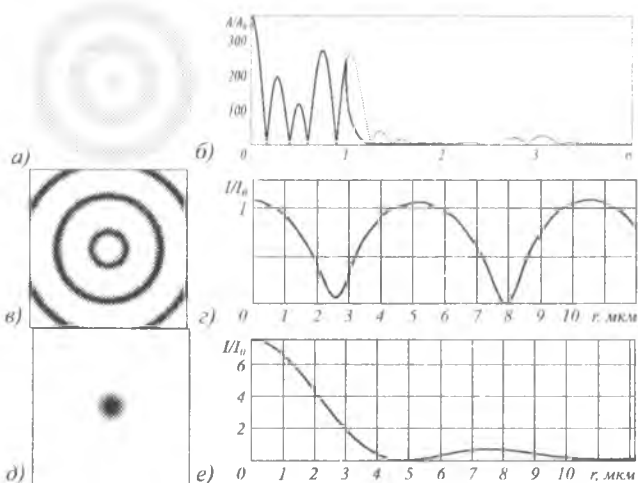


Рис. 20 Моделирование для аксикона (а) с помощью выражений (11)–(12): радиальное сечение амплитуды в спектральной плоскости, соответствующее распространению входного поля на расстоянии  $z = 0,5$  мкм (точечная линия) и  $z = 7$  мкм (сплошная линия) (б). поперечное распределение интенсивности в плоскости  $z = 0,5$  мкм (в), радиальное сечение интенсивности в плоскости  $z = 0,5$  мкм (з), поперечное распределение интенсивности в плоскости  $z = 7$  мкм (д), радиальное сечение интенсивности в плоскости  $z = 7$  мкм (е)

Картина дифракции ограниченной радиусом  $R$  плоской волны в приближении тонкого оптического элемента в рамках в скалярной непараксиальной теории описывается выражением:

$$E(\rho, z) = k^2 \int_0^{\alpha_0} P(\sigma) \exp(ikz\sqrt{1-\sigma^2}) J_0(k\sigma\rho) \sigma d\sigma, \quad (11)$$

где

$$P(\sigma) = \int_0^R \tau(r) J_0(k\sigma r) r dr, \quad (12)$$

пространственный спектр,  $\sigma_0$  – радиус учитываемых пространственных частот.

На рис. 20б показано радиальное сечение амплитуды в спектральной плоскости (12), соответствующее расстоянию от плоскости аксикона (10)  $z=0,5$  мкм (точечная линия) и  $z=7$  мкм (сплошная линия). Как видно из графиков, в первом случае необходимо учитывать более высокие пространственные частоты.

На рис. 20в и 20д показаны поперечные распределения интенсивности в плоскостях  $z=0,5$  мкм и  $z=7$  мкм, соответственно. Видно, что в непосредственной близости от оптического элемента (рис. 20в) распределение интенсивности повторяет структуру элемента.

Таким образом, в скалярном случае полная ширина центрального светового пятна по полуспаду интенсивности (FWHM), формируемого высокоапертурным бинарным микроаксиконом (10), на расстоянии  $z = 0,5$  мкм ( $\approx 0,05\lambda$ ) составляет 3,63 мкм ( $\approx 0,34\lambda$ ), а площадь по полуспаду интенсивности (НМА) равна 10,35 мкм<sup>2</sup> ( $\approx 0,092\lambda^2$ ). На расстоянии  $z = 7$  мкм ( $\approx 0,66\lambda$ ) FWHM = 4,44 мкм ( $\approx 0,42\lambda$ ), а НМА = 15,48 мкм<sup>2</sup> ( $\approx 0,138\lambda^2$ ).

Хотя приведенные выше результаты говорят о преодолении дифракционного предела, который соответствует FWHM = 0,51 $\lambda$ , минимум для бесселевых пучков (FWHM = 0,36 $\lambda$ ) удастся достичь лишь в непосредственной близости к поверхности элемента – в зоне затухающих волн.

### 3.3 Векторные операторы распространения.

В данном разделе рассматриваются векторные операторы распространения, которые необходимо использовать при уменьшении поперечных размеров светового поля (или его характерных деталей) до размеров порядка длины волны.

#### 3.3.1 Векторное интегральное преобразование Рэлея-Зоммерфельда

Интегральная теорема Рэлея-Зоммерфельда первого типа в векторной форме записывается следующим образом:

$$E_x(u, v, z) = -\frac{z}{2\pi} \iint_{\Sigma} E_{0x}(x, y) \frac{e^{ik\ell}}{\ell^2} \left( ik - \frac{1}{\ell} \right) dx dy, \quad (13a)$$

$$E_y(u, v, z) = -\frac{z}{2\pi} \iint_{\Sigma} E_{0y}(x, y) \frac{e^{ik\ell}}{\ell^2} \left( ik - \frac{1}{\ell} \right) dx dy, \quad (13б)$$

$$E_z(u, v, z) = \frac{1}{2\pi} \iint_{\Sigma} [E_{0x}(x, y)(u-x) + E_{0y}(x, y)(v-y)] \frac{e^{ik\ell}}{\ell^2} \left( ik - \frac{1}{\ell} \right) dx dy, \quad (13в)$$

где  $\ell = \sqrt{(u-x)^2 + (v-y)^2 + z^2}$ ,  $E_{0x}(x, y)$ ,  $E_{0y}(x, y)$  – тангенциальные компоненты входного электрического поля, которые также могут быть заданы через коэффициенты поляризации:

$$\begin{pmatrix} E_{0x}(x, y) \\ E_{0y}(x, y) \end{pmatrix} = E_0(x, y) \begin{pmatrix} a_x \\ a_y \end{pmatrix}.$$

Во всех этих интегралах присутствует выражение для сферической волны  $\exp(ik\ell)/\ell$ . Как правило, это не позволяет аналитически вычислить интегралы (13). На первый взгляд можно использовать разложение сферической волны в ряд по функциям Бесселя, но замена подынтегрального выражения на беско-

нечный ряд, включающий спецфункции, не облегчает численный расчёт. Таким образом, для прямых вычислений по формулам (13) либо нужно использовать алгоритмы быстрого расчета, либо привлекать высокопроизводительные вычислительные средства.

### 3.3.2 Векторное представление через плоские волны

Вычисление с помощью выражений (13) является очень ресурсозатратным, а какие-либо аппроксимации расстояния  $\ell$  приводят к существенным погрешностям в ближней зоне дифракции.

Векторный вариант оператора распространения, использующего разложение по плоским волнам позволяет учесть радиальную симметрию решаемых задач, а также использовать алгоритм быстрого преобразования Фурье.

В этом случае все компоненты электромагнитного поля на расстоянии  $z$  вычисляются с помощью следующих выражений:

$$\mathbf{E}(x, y, z) = \begin{bmatrix} E_x(x, y, z) \\ E_y(x, y, z) \\ E_z(x, y, z) \end{bmatrix} = \frac{1}{\lambda^2} \iint_{\Sigma} \mathbf{S}_E(\xi, \eta) \begin{bmatrix} F_x(\xi, \eta) \\ F_y(\xi, \eta) \end{bmatrix} \exp\left[ ikz\sqrt{1-(\xi^2+\eta^2)} \right] \exp[ik(\xi x + \eta y)] d\xi d\eta, \quad (14)$$

$$\mathbf{H}(x, y, z) = \begin{bmatrix} H_x(x, y, z) \\ H_y(x, y, z) \\ H_z(x, y, z) \end{bmatrix} = \frac{1}{\lambda} \iint_{\Sigma} \mathbf{S}_H(\xi, \eta) \begin{bmatrix} F_x(\xi, \eta) \\ F_y(\xi, \eta) \end{bmatrix} \exp\left[ ikz\sqrt{1-(\xi^2+\eta^2)} \right] \exp[ik(\xi x + \eta y)] d\xi d\eta, \quad (15)$$

$$\begin{bmatrix} F_x(\xi, \eta) \\ F_y(\xi, \eta) \end{bmatrix} = \iint_{\Sigma} \begin{bmatrix} E_x(x, y, 0) \\ E_y(x, y, 0) \end{bmatrix} \exp[-ik(\xi x + \eta y)] dx dy, \quad (16)$$

где  $F_x(\xi, \eta)$  и  $F_y(\xi, \eta)$  - спектры тангенциальных компонент входного электрического поля  $E_x(x, y, 0)$ ,  $E_y(x, y, 0)$  определенного в области  $\Sigma$ .

Матрицы поляризационного преобразования для электрических и магнитных компонент имеют следующий вид:

$$\mathbf{S}_E(\xi, \eta) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \frac{\xi}{\sqrt{1-(\xi^2+\eta^2)}} & \frac{\eta}{\sqrt{1-(\xi^2+\eta^2)}} \end{bmatrix}, \quad (17)$$

$$\mathbf{S}_H(\xi, \eta) = -\sqrt{\frac{\epsilon}{\mu}} \begin{bmatrix} \frac{\xi\eta}{\sqrt{1-(\xi^2+\eta^2)}} & \frac{1-\xi^2}{\sqrt{1-(\xi^2+\eta^2)}} \\ \frac{1-\eta^2}{\sqrt{1-(\xi^2+\eta^2)}} & \frac{\xi\eta}{\sqrt{1-(\xi^2+\eta^2)}} \\ \xi & -\eta \end{bmatrix}, \quad (18)$$

### 3.4 Примеры расчета с использованием векторных интегралов

#### 3.4.1 Дифракция плоской волны на круглой апертуре

В разделе 3.2 было показано, что при радиусе апертуры  $10\lambda$  в скалярном случае рассмотренные алгоритмы РЗ и ПВ практически полностью совпадают. При уменьшении размеров апертуры до сравнимых с длиной волны необходимо учитывать векторный характер светового поля.

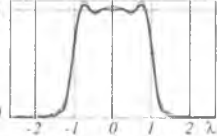
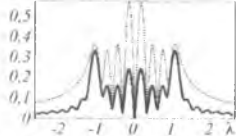
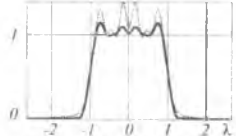
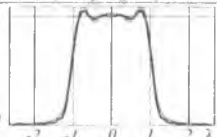
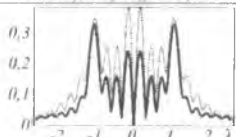
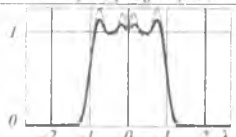
Как следует из раздела 3.3, при линейной  $x$ -поляризации падающей на микроапертуру плоской волны дополнительный вклад в суммарную интенсивность, кроме  $x$ -компоненты, будет вносить продольная компонента. При вычислении этой компоненты каждый из рассматриваемых алгоритмов также имеет свои особенности. Для оценки точности получаемых результатов в данном разделе используется алгоритм FDTD.

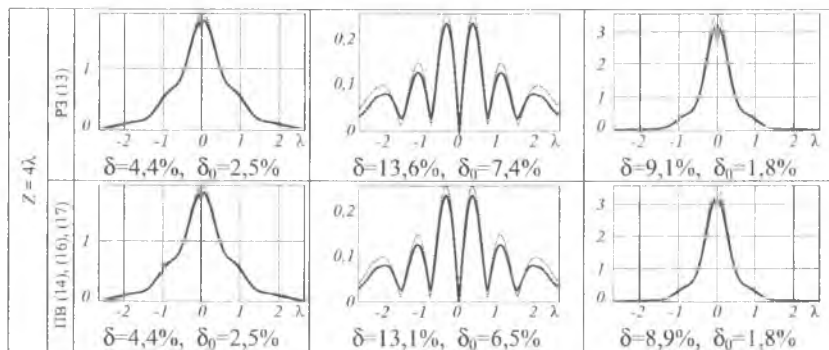
Чтобы согласовать результаты, полученные с помощью интегральных методов, не учитывающих материал и толщину экрана с отверстием, при использовании конечно-разностного временного метода рассматривалось распространение радиально-ограниченной плоской волны в свободном пространстве.

Параметры расчёта при использовании алгоритма FDTD, реализованного в программном продукте R-Soft: длина волны  $\lambda = 532$  нм, радиус, ограничивающий плоскую волну на входе  $r_0 = 2\lambda$ , размер расчётной области  $x \in [-5\lambda, 5\lambda]$ ,  $y \in [-5\lambda, 5\lambda]$ ,  $z \in [0, 5\lambda]$ . Толщина PML  $-\lambda$ , шаг дискретизации по пространству:  $\lambda/15$ , шаг дискретизации по времени:  $\lambda/(30c)$  ( $c$  – скорость света в вакууме), положение плоскости регистрации:  $z = 0,3\lambda$  ( $\approx 160$  нм) и  $z = 4\lambda$  ( $\approx 2$  мкм).

В табл. 3 приведены сравнительные результаты расчёта дифракции плоской волны при линейной  $x$ -поляризации с помощью метода FDTD, дифракционных интегралов РЗ (13) и разложения по плоским волнам (14), (16), (17).

Таблица 3. Сравнительные результаты расчёта дифракции плоской волны при линейной  $x$ -поляризации с помощью метода FDTD и дифракционных интегралов на расстоянии  $z = 0,3\lambda$  и  $z = 4\lambda$

Плоскость	Метод	Сечение амплитуды $x$ -компоненты, $ E_x $	Сечение амплитуды $z$ -компоненты, $ E_z $	Сечение суммарной интенсивности, $ E ^2$
$Z = 0,3\lambda$	РЗ (13)	 $\delta = 3,4\%$ , $\delta_0 = 2,9\%$	 $\delta = 52,3\%$ , $\delta_0 = 26,5\%$	 $\delta = 11,4\%$ , $\delta_0 = 7,5\%$
	ПВ (14), (16), (17)	 $\delta = 3,4\%$ , $\delta_0 = 2,9\%$	 $\delta = 33,1\%$ , $\delta_0 = 17,3\%$	 $\delta = 6,9\%$ , $\delta_0 = 4,9\%$



Для алгоритма ПВ на расстоянии  $z = 0,3\lambda$  от апертуры учитывались спектральные частоты в радиусе  $\sigma_0 = 3$  и применялся обход особенности на основе формулы Ньютона-Лейбница, а на расстоянии  $z = 4\lambda$  от апертуры полагалось  $\sigma_0 = 1$ . В табл. 3 приведены сравнительные графики и среднеквадратичные погрешности  $\delta$  сечений полученной интенсивности для каждого метода: точечной линией показан график, полученный с помощью дифракционных интегралов, а сплошной – полученный с помощью метода FDTD.

Как следует из приведённых в табл. 3 результатов, основную погрешность в расчётах вносит именно продольная компонента. Результаты для  $x$ -компоненты (что соответствует скалярному случаю) совпадают для обоих алгоритмов и незначительно отличаются от предсказанных FDTD.

Погрешность в вычислении  $z$ -компоненты растёт с уменьшением расстояния до апертуры. При этом алгоритм ПВ (14), (16), (17) демонстрирует лучшие результаты, чем алгоритм РЗ, на расстояниях, меньших длины волны.

При удалении от апертуры всего на несколько длин волн результаты этих двух алгоритмов фактически совпадают (табл. 3 для  $z = 4\lambda$ ) и отличаются от полученных с помощью FDTD только масштабно, т.е. рассматриваемые в данной работе алгоритмы позволяют получать правильное распределение, но несколько завышенное по амплитуде. Среднеквадратичное отклонение приведённых к одному масштабу распределений  $\delta_0$  составляет в обоих случаях менее 2% для суммарной интенсивности.

Такая амплитудная «завышенность» может быть связана с тем, что рассмотренные методы РЗ и ПВ не подразумевают наличия  $y$ -компоненты в дифракционной картине изначально  $x$ -поляризованного поля. Метод же FDTD показывает, что эта компонента присутствует, хотя её доля энергии в суммарном электрическом поле невелика.

### 3.4.2 Дифракция вихревого пучка на круглой апертуре.

При дифракции линейно-поляризованной вдоль оси  $x$  плоской волны с вихревой сингулярностью  $m=1$  на круглой апертуре выражения (14), (16), (17) приводят к следующим формулам:

$$E_x(\rho, \theta, z) = -\frac{i^2 k \exp(ikR)}{R^2} \exp(i\theta) \int_0^{\rho} \exp\left(ik \frac{r^2}{2R}\right) J_1\left(\frac{kr\rho}{R}\right) r dr, \quad (19a)$$

$$E_y(\rho, \theta, z) = 0, \quad (19b)$$

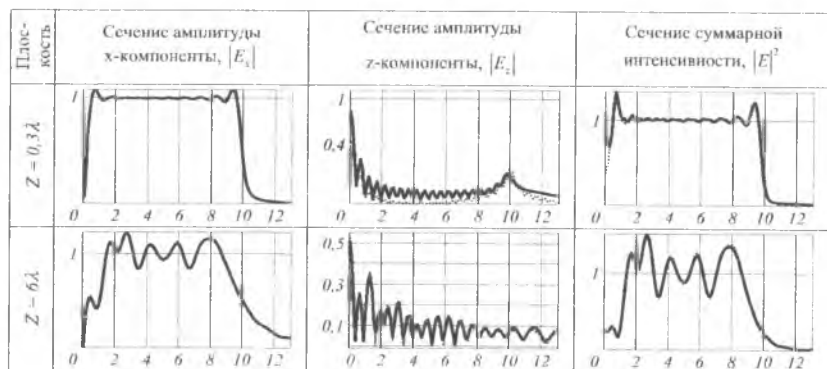
$$E_z(\rho, \theta, z) = -\frac{k \exp(ikR)}{R^2} \exp(i\theta) \left\{ \rho \cos\theta \int_0^{\rho} \exp\left(ik \frac{r^2}{2R}\right) J_1\left(\frac{kr\rho}{R}\right) r dr - \right. \\ \left. - i \int_0^{\rho} \exp\left(ik \frac{r^2}{2R}\right) \left\{ e^{i\theta} J_2\left(\frac{kr\rho}{R}\right) - \left\{ -e^{-i\theta} J_0\left(\frac{kr\rho}{R}\right) \right\} r^2 dr \right\} \right\}. \quad (19b)$$

Таким образом, при  $|m|=1$  для линейной  $x$ -поляризации легко видеть, что суммарная интенсивность на оптической оси не будет равна нулю, как это предсказывает скалярная теория. Осевые значения поперечной компоненты действительно обнуляются, но продольная компонента остаётся ненулевой:

$$E_z(0, 0, z) = -\frac{ik \exp(ikz)}{2z^2} \int_0^{\rho} \exp\left(ik \frac{r^2}{2z}\right) r^2 dr. \quad (20)$$

Из табл. 4 видно, что при расстояниях от апертуры более пяти длин волн оба метода полностью совпадают, в том числе по внеосевым распределениям. В таблице 2 сплошной линией показаны графики для алгоритма РЗ (13), а точечной – для алгоритма ПВ (14), (16), (17).

Таблица 4. Сравнительные результаты вычислений дифракционных интегралов для дифракции плоской волны с вихревой фазовой сингулярностью  $m = 1$  при линейной  $x$ -поляризации



Также различие двух рассмотренных методов может усугубляться наличием в (17) особенности при вычислении  $z$ -компоненты, возникающей при  $\sigma \rightarrow 1$ , т.е. очень близко к плоскости апертуры. Таким образом, проблемы с расчётом продольной компоненты, обнаруженные для плоской волны, также проявляются в случае вихревых пучков.

### 3.5 Векторное представление через плоские волны в модификации Мансурипура

Возможно также спектральное разложение падающей линсйно-поляризованной волны как прохождения через виртуальные призмы, которые меняют направление вектора распространения волны. На основе такого подхода получается иная, чем в (17), матрица преобразования спектральных координат:

$$M_E(\xi, \eta) = \begin{bmatrix} 1 - \frac{\xi^2}{1 + \sqrt{1 - (\xi^2 + \eta^2)}} & -\frac{\xi\eta}{1 + \sqrt{1 - (\xi^2 + \eta^2)}} \\ -\frac{\xi\eta}{1 + \sqrt{1 - (\xi^2 + \eta^2)}} & 1 - \frac{\eta^2}{1 + \sqrt{1 - (\xi^2 + \eta^2)}} \\ -\xi & -\eta \end{bmatrix}. \quad (21)$$

Интересно, что матрица Мансурипура предсказывает появление  $y$ -компоненты при распространении, даже если изначально волна была полностью  $x$ -поляризованной. Достоинство матрицы (21) также в отсутствии каких-либо особенностей.

Чтобы оценить корректность всех рассмотренных выше алгоритмов в ближней зоне дифракции, было проведено сравнение с результатами алгоритма FDTD.

Параметры расчёта при использовании алгоритма FDTD, реализованного в программном продукте R-Soft: длина волны  $\lambda = 532$  нм, радиус круглой апертуры  $r_0 = 2\lambda$ , порядок вихря  $m = 1$ , размер расчётной области  $x \in [-5\lambda, 5\lambda]$ ,  $y \in [-5\lambda, 5\lambda]$ ,  $z \in [0, 5\lambda]$ . Толщина PML —  $\lambda$ , шаг дискретизации по пространству:  $\lambda/15$ , шаг дискретизации по времени:  $\lambda/(30c)$ ,  $c$  — скорость света, положение плоскости экрана для измерений:  $z = 0,3\lambda$  ( $\approx 160$  нм) и  $z = 4\lambda$  ( $\approx 2$  мкм).

Сравнительные результаты расчёта дифракции плоской волны с вихревой фазовой сингулярностью  $m = 1$  при линейной  $x$ -поляризации на расстоянии  $z = 4\lambda$  от апертуры показали, что наименьшее отличие от результатов, предсказываемых FDTD, демонстрирует алгоритм ПВ с модификацией Мансурипура. В этом случае наблюдается не только наименьшее среднеквадратичное отклонение (менее 1%), но и полное качественное согласование — присутствует  $y$ -компонента.

Метод Мансурипура, являясь очень удобным для алгоритмизации, позволяет получать результаты, практически совпадающие с расчётами FDTD, кроме области очень близкой к апертуре. В этой области, как видно из табл. 5, имеется определённое отличие во внеосевом распределении, хотя погрешность остаётся минимальной из всех исследуемых интегральных методов.

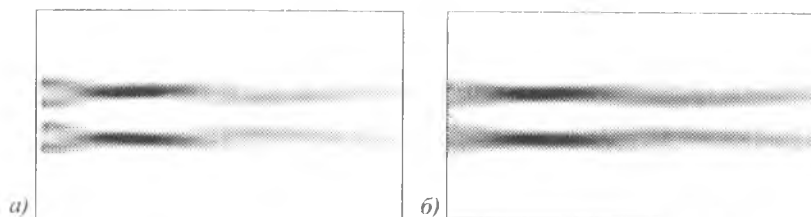
В табл. 5 показаны результаты работы рассматриваемых алгоритмов для области, очень близкой к апертуре, а именно, на расстоянии  $z = 0,3\lambda$  от неё: с помощью метода FDTD (первая строка), дифракционных интегралов Релея-Зоммерфельда (13) (вторая строка), разложения по плоским волнам (14), (16), (17) (третья строка) и в модификации Мансурипура (14), (16), (21) (третья строка). Несмотря на рост погрешности в ближней к элементу области (до десятка процентов), имеется хорошее качественное согласование результатов, особенно для метода ПВ. На графиках также точечной линией показаны результаты,

полученные с помощью дифракционных интегралов, а сплошной – полученные с помощью метода FDTD.

Таблица 5. Сравнительные результаты расчёта дифракции плоской волны с вихревой фазовой сингулярностью  $m = 1$  при линейной  $x$ -поляризации с помощью метода FDTD и дифракционных интегралов на расстоянии  $Z = 0,3\lambda$  (160 нм)

Метод, время расчёта	$ E_x $	$ E_y $	$ E_z $	$ E ^2$
FDTD				
ГЗ (13)		Отсутствует		 $\delta = 16,5\%$ , $\delta_0 = 12,2\%$
ПВ (14), (16), (17)		Отсутствует		 $\delta = 11,3\%$ , $\delta_0 = 6,9\%$
Мансурипур (14), (16), (21)				 $\delta = 6,9\%$ , $\delta_0 = 6,5\%$

На рис. 4 показано распределение интенсивности вихревого пучка в плоскости  $XZ$  при  $y = 0$  в области  $x \in [-5\lambda, 5\lambda]$ ,  $z \in [0, 5\lambda]$ , рассчитанное методом FDTD и ПВ с матрицей Мансурипура. Как видно, полученные картины вполне согласованы друг с другом. При этом хорошо видна структура теневой области вихревого пучка: размер световой воронки то увеличивается, то уменьшается при распространении пучка, и он может быть достаточно мал (диаметр между максимумами  $1,7\lambda$  на расстоянии  $z = 3\lambda$ ).





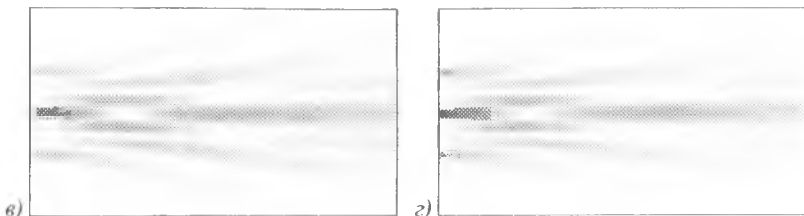


Рис.4. Распределение обшей интенсивности (а), (б) и амплитуды продольной компоненты (в), (з) вихревого пучка в плоскости  $yz$  при  $y=0$  в области  $x \in [-5\lambda, 5\lambda]$ ,  $z \in [0, 5\lambda]$ , рассчитанное методом FDTD (а), (в) и ПВ с матрицей Мансурпура (14), (16), (21) (б), (з)

### 3.6 Учёт в методе разложения по плоским волнам коэффициентов пропускания Френеля

#### 3.6.1 Представление через TE- и TM-компоненты

Запишем метод разложения по плоским волнам в общем виде в декартовых координатах, причём представим электрический вектор поля через сумму TE- и TM-поляризованных полей:

$$\mathbf{E}(u, v, z) = \mathbf{E}_{TE}(u, v, z) + \mathbf{E}_{TM}(u, v, z), \quad (22)$$

$$\mathbf{E}_{TE}(u, v, z) = \iint_{\xi, \eta} \frac{1}{\xi^2 + \eta^2} \begin{bmatrix} \eta^2 & -\xi\eta \\ -\xi\eta & \xi^2 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} F_x(\xi, \eta) \\ F_y(\xi, \eta) \end{pmatrix} \exp[ikz\sqrt{1-(\xi^2 + \eta^2)}] \exp[ik(\xi u + \eta v)] d\xi d\eta, \quad (23)$$

$$\mathbf{E}_{TM}(u, v, z) = \iint_{\xi, \eta} \frac{1}{\xi^2 + \eta^2} \begin{bmatrix} \xi^2 & \xi\eta \\ \xi\eta & \eta^2 \\ \frac{(\xi^2 + \eta^2)\xi}{\sqrt{1-\xi^2 - \eta^2}} & \frac{(\xi^2 + \eta^2)\eta}{\sqrt{1-\xi^2 - \eta^2}} \end{bmatrix} \begin{pmatrix} F_x(\xi, \eta) \\ F_y(\xi, \eta) \end{pmatrix} \exp[ikz\sqrt{1-(\xi^2 + \eta^2)}] \exp[ik(\xi u + \eta v)] d\xi d\eta, \quad (24)$$

Очевидно, что в свободном пространстве, выражения (22)-(24) записываются в классическом виде (14), (16), (17).

Однако при прохождении через границу двух сред вектора напряженности электрического поля волны, поляризованной перпендикулярно и параллельно плоскости падения, имеют различные коэффициенты пропускания Френеля. С учётом этого факта матрица (17) будет выглядеть следующим образом:

$$\mathbf{S}_k(\xi, \eta) = \frac{1}{\xi^2 + \eta^2} \begin{bmatrix} t_p \xi^2 + t_s \eta^2 & \xi\eta(t_p - t_s) \\ \xi\eta(t_p - t_s) & t_s \xi^2 + t_p \eta^2 \\ \frac{t_p \xi(\xi^2 + \eta^2)}{\sqrt{1-(\xi^2 + \eta^2)}} & \frac{t_p \eta(\xi^2 + \eta^2)}{\sqrt{1-(\xi^2 + \eta^2)}} \end{bmatrix}, \quad (25)$$

где  $t_s, t_p$  – коэффициенты пропускания Френеля для ТЕ- и ТМ-компонент, соответственно.

В модификации Мансурипура вид матрицы преобразования будет следующий:

$$M_E(\xi, \eta) = \frac{1}{(\xi^2 + \eta^2)} \begin{bmatrix} t_p \gamma \xi^2 + t_s \eta^2 & \xi \eta (t_p \gamma - t_s) \\ \xi \eta (t_p \gamma - t_s) & t_s \xi^2 + t_p \eta^2 \\ -t_p \xi (\xi^2 + \eta^2) & -t_s \eta (\xi^2 + \eta^2) \end{bmatrix}, \quad \gamma = \sqrt{1 - (\xi^2 + \eta^2)}. \quad (26)$$

Заметим, что при  $\gamma=1$  (соответствует нулевой пространственной частоте) (25) сводится к (17) только для поперечных компонент. Причём для продольной компоненты имеет место особенность при  $\gamma \rightarrow 0$  (для высоких значений пространственных частот), которая формально приводит к неограниченному росту продольной компоненты. В модификации Мансурипура такая особенность отсутствует.

### 3.6.2 Расчет коэффициентов Френеля

Коэффициенты Френеля необходимо учитывать в случае, когда во входной плоскости имеется оптический элемент. Обычно коэффициенты рассчитываются исходя из формы элемента. Однако при использовании разложения по плоским волнам знание формы не требуется (она может быть определена с точностью до длины волны по создаваемому элементом распределению фазы). Достаточно только знать показатели преломления элемента и среды.

Коэффициенты Френеля вычисляются по формулам:

$$t_s = \frac{2n_1 \cos \beta_i}{n_1 \cos \beta_i + n_2 \cos \beta_t}, \quad t_p = \frac{2n_1 \cos \beta_i}{n_2 \cos \beta_i + n_1 \cos \beta_t}, \quad (27)$$

где  $n_1, n_2$  – показатели преломления соответственно элемента и среды,  $\beta_i$  – угол падения,  $\beta_t$  – угол преломления. Они связаны законом преломления:

$$n_1 \cos \beta_i = n_2 \cos \beta_t, \quad (28)$$

и соотношением, позволяющим обойтись без знания формы элемента:

$$\cos(\beta_i - \beta_t) = \gamma \quad (29)$$

Из равенств (28) и (29) получаем значения квадратов косинусов углов

$$\cos^2(\beta_i) = \frac{(n_1 - \gamma n_2)^2}{n_1^2 + n_2^2 - 2\gamma n_1 n_2}, \quad \cos^2(\beta_t) = \frac{(n_2 - \gamma n_1)^2}{n_1^2 + n_2^2 - 2\gamma n_1 n_2}, \quad (30)$$

Если подставить выражения (30) в (27), то получим следующие результаты:

$$t_s = \frac{2}{1 + \frac{n_2 |n_1 - \gamma n_1|}{n_1 |n_1 - \gamma n_2|}}, \quad t_p = \frac{2}{n_2 + \frac{n_2 - \gamma n_1}{|n_1 - \gamma n_2|}} \quad (31)$$

В таком виде коэффициенты пропускания Френеля удобно использовать только для распространяющихся волн, когда  $\gamma$  является вещественным. Для заглушающих волн  $\gamma$  является мнимым и действие операции взятия модуля приводит к неверным результатам. Поэтому избавимся от знака модуля.

Пусть  $\gamma$  является вещественным,  $0 \leq \gamma \leq 1$ , примем также, что  $n_1 > n_2$ . В этом случае  $n_1 - \gamma n_2 > 0$ , поэтому модуль этой величины совпадает с ней самой, а

$$|n_2 - \gamma_1| = \begin{cases} (n_2 - \gamma_1), & \gamma < \frac{n_2}{n_1}, \\ -(n_2 - \gamma_1), & \text{иначе.} \end{cases} \quad (32)$$

Таким образом, вместо (31) можно записать:

$$t_s = \begin{cases} \frac{2n_1(n_1 - \gamma_2)}{n_1^2 + n_2^2 - 2\gamma_1 n_2}, & \gamma < \frac{n_2}{n_1}, \\ \frac{2n_1(n_1 - \gamma_2)}{n_1^2 - n_2^2}, & \text{иначе.} \end{cases} \quad t_r = \begin{cases} \frac{2n_1(n_1 - \gamma_2)}{2n_1 n_2 - \gamma(n_1^2 + n_2^2)}, & \gamma < \frac{n_2}{n_1}, \\ \frac{2n_1(n_1 - \gamma_2)}{\gamma(n_1^2 - n_2^2)}, & \text{иначе.} \end{cases} \quad (33)$$

Выражения (33) также можно использовать для учёта затухающих волн. Причём при  $\gamma < n_2/n_1$  имеет место непрерывность перехода от вещественной области к мнимой.

Выражения (33) соответствуют прохождению только через рельефную сторону оптического элемента, т.е. из среды оптического элемента в окружающую среду. Учтем далее также вход падающего излучения в оптический элемент.

Можно показать [11], что соотношение между коэффициентами пропускания и отражения имеет вид:

$$r^2 + t^2 \frac{N_2 \cos \beta_2}{N_1 \cos \beta_1} = 1 \quad (34)$$

где  $t$  – коэффициент пропускания, а  $r$  – коэффициент отражения,  $N_1$  и  $N_2$  – показатели преломления соответственно первой (откуда идут лучи) и второй среды, а  $\beta_1$  и  $\beta_2$  – соответственно углы падения в первой среде и выхода во вторую.

Соотношение (34) означает, что для вычисляемых по (33) коэффициентов пропускания Френеля необходимо выполнять коррекцию:

$$t^c = t \sqrt{\frac{N_2 \cos \beta_2}{N_1 \cos \beta_1}} = t A_c \quad (35)$$

Рассмотрим теперь с учётом (35) прохождение излучения через обе стороны оптического элемента с показателем преломления  $n_1$ , размещённого в среде с показателем преломления  $n_2$ . Будем считать, что излучение падает на плоскую сторону оптического элемента.

Используя выражения (33), в которых меняем местами  $n_1$  и  $n_2$ , и, учитывая равенство нулю углов падения и преломления на плоской стороне элемента, получаем

$$t_s^1 = t_p^1 = \frac{2n_2}{n_1 + n_2} \quad (36)$$

Коррекция по формуле (41), в которой  $N_1 = n_2$ ,  $N_2 = n_1$ ,  $\beta_1 = \beta_2 = 0$ ,  $A_c = \sqrt{n_1/n_2}$  приводит к следующему виду корректных коэффициентов пропускания:

$$t_s^{1c} = t_p^{1c} = \frac{2\sqrt{n_1 n_2}}{n_1 + n_2} \quad (37)$$

При прохождении через вторую (рельефную сторону) оптического элемента в (41)  $N_1 = n_1$ ,  $N_2 = n_2$ ,  $\beta_1 = \beta_1$ ,  $\beta_2 = \beta_1$  и коррекционный коэффициент равен

$$A_c = \sqrt{\frac{n_2 \cos \beta_1}{n_1 \cos \beta_1}} \quad (38)$$

С учетом выше приведенных рассуждений для косинусов преломленного и падающего углов, выражение (38) можно записать следующим образом:

$$A_i = \begin{cases} \sqrt{\frac{n_2 |n_2 - \gamma_{i1}|}{n_1 (n_1 - \gamma_{i2})}}, & \gamma - \text{вещественная,} \\ \sqrt{\frac{n_2 (n_2 - \gamma_{i1})}{n_1 (n_1 - \gamma_{i2})}}, & \gamma - \text{мнимая.} \end{cases} \quad (39)$$

Итоговый коэффициент пропускания равен произведению коэффициентов пропускания для плоской и рельефной сторон.

### 3.6.3 Дифракция плоской волны на бинарном би-аксиконе

В работе [14] было показано, что аподизация функции пропускания высокоапертурной фокусирующей системы фазовыми линейными скачками позволяет уменьшить размер фокального пятна при линейной поляризации. Рассмотрим аналогичный подход для дифракционного аксикона (9).

В этом случае входное поле будет иметь следующий вид (рис. 5а):

$$E_0(x, y) = E_0(r, \varphi) = \begin{cases} \exp\{i \arg[\cos(kr) \cos \varphi]\}, & r \leq R, \\ 0, & r > R, \end{cases} \quad (40)$$

необладающий осевой симметрией, поэтому для расчётов на основе разложения по плоским волнам необходимо воспользоваться выражениями (14)-(16) с соответствующими матрицами. Для ускорения вычислений в поперечной плоскости можно применить быстрое преобразование Фурье (БПФ), но для продольных распределений этот алгоритм затруднительно применять из-за необходимости менять в каждой плоскости шаг дискретизации.

На рис. 21 и в Таблице 6 приведены сравнительные результаты моделирования с помощью метода разложения по плоским волнам (14), (16) с учётом коэффициентов Френеля для стандартной матрицы (25) и матрицы Мансурипура (26), а также FDTD. В последнем случае был использован симметричный аксикон в условиях падения на него пучка с равномерной амплитудой и фазой, имеющий линейный скачок перпендикулярно оси поляризации.

В данном случае можно говорить только о качественном совпадении результатов, полученных с помощью интегральных методов и разностного. Причем метод разложения по плоским волнам в модификации Мансурипура обеспечивает корректные результаты на расстояниях более длины волны от оптического элемента (см. рис 21б), использование классической матрицы (25) позволяет достаточно близкие к FDTD результаты в непосредственной близости к элементу.

На расстоянии  $z = 7 \text{ мкм}$  ( $\approx 0,66\lambda$ ) поперечный размер центрального светового пятна вдоль оси  $x$  составляет 110 полуспаду интенсивности  $\text{FWHM}(-) = 0,46\lambda$  для метода FDTD и  $\text{FWHM}(-) = 0,45\lambda$  для метода разложения по плоским волнам в модификации Мансурипура, что достаточно близко к результатам, предсказанным скалярной теорией ( $\text{FWHM} = 0,42\lambda$ ) и почти в два раза меньше, чем при использовании радиально-симметричного аксикона. Такой эффект был достигнут за счет формирования на оптической оси распределения, в основном состоящего из продольной компоненты.

В непосредственной близости к оптическому элементу (в плоскости  $z = 0,5 \text{ мкм} \approx 0,05\lambda$ ), как показывают результаты применения метода FDTD, формируется очень компактное круглое световое пятно с  $\text{FWHM} = 0,32\lambda$  и  $\text{HMA} = 0,08\lambda^2$ . Такой же горизонтальный размер получается при использовании классической матрицы (25), хотя световое пятно в этом случае вытянутое.

Из рис. 22 видно, что протяженный фокус с компактным световым пятном сохраняется на расстоянии примерно две длины волны от оптического элемента, что соответствует радиусу биаксикона.

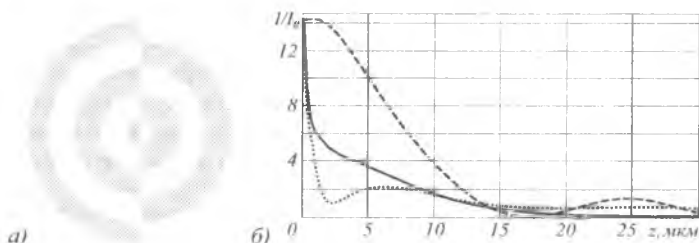


Рис. 21. Результаты моделирования для фазового биаксикона (а): осевое распределение интенсивности (б), полученное с помощью выражений (14), (16), (25) (пунктирная линия), (14), (16), (26) (точечная линия) и метода FDTD (сплошная линия)

Таблица 6. Результаты моделирования дифракции на бинарном биаксиконе ограниченной плоской волны с линейной  $x$ -поляризацией на расстоянии  $z = 7 \text{ мкм}$  от элемента, размер изображений  $50 \text{ мкм} \times 50 \text{ мкм}$

	Выражения (19), (20), (22)	Выражения (19), (20), (28)	Метод FDTD
Амплитуда $ E_x $			
Амплитуда $ E_y $			
Амплитуда $ E_z $			

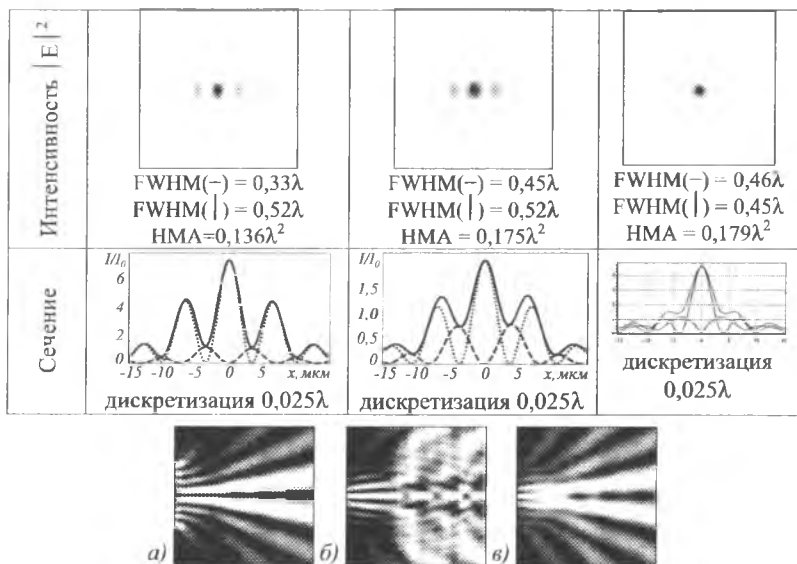


Рис. 22. Картина дифракции на биаксиконе (топология), рассчитанная с использованием (14), (16), (25) в области  $z \in [0,1 \text{ мкм}, 100 \text{ мкм}]$ ,  $x \in [-50 \text{ мкм}, 50 \text{ мкм}]$ : интенсивность для  $x$ -компоненты (а),  $z$ -компоненты (б), суммарная (в)

### 3.7 Описание программного обеспечения *FieldPropagators*.

#### 3.7.1 Структура языка

Можно выделить следующие типы параметров

- параметры задания одного или нескольких вариантов расчета,
- параметры задания одного или нескольких пропагаторов (операторов распространения),
- параметры задания одного или нескольких входных полей,
- параметры общие для всех пакетов.

Строка задания вариант расчета имеет следующий формат:

**target** = <наименованиеВариантаРасчета> <параметры> [ : <суффикс> ]

В файле параметров должен быть задан один или несколько вариантов расчета. Если задано более одного варианта расчета, то они должны иметь различный суффикс. В файле параметров может быть не более одного варианта расчета без суффикса.

Строка задания пропагатора имеет следующий формат:

**propagator** = <имяПропагатора> <параметры> [ : <суффикс> ]

В файле параметров может быть задан один или несколько пропагаторов. Если задано более одного пропагатора, то они должны иметь различный суф-

фикс (<postfix>). В файле параметров может быть не более одного пропегатора без суффикса. Для вывода графика или изображений входного поля пропегатор не нужен. В этом случае можно не задавать пропегатор.

Пропегаторы могут быть векторными или скалярными.

Строка задания входного поля имеет следующий формат:

**objectField** = <имяТипаПоля> <параметры> [ : <суффикс> ]

В файле параметров должно быть задано одно или несколько входных полей. Если задано более одного входного поля, то они должны иметь различный суффикс (<postfix>). В файле параметров может быть не более одного входного поля без суффикса.

Входные поля могут быть одного из трех типов:

- поля для скалярных пропегаторов ( имеют одну компоненту ),
- поля для векторных пропегаторов, заданные одной компонентой и коэффициентами поляризации,
- поля для векторных пропегаторов, заданные двумя компонентами.

Возможность множественного задания пропегаторов, входных полей и вариантов расчета позволяет проще и быстрее решать многие задачи. Например, для сравнения работы разных пропегаторов достаточно задать более одного пропегатора. Или для одного поля и одного пропегатора задать несколько вариантов расчета. Или сравнить результат от нескольких входных полей.

#### 3.7.1.1 Пакеты параметров

В случае задания нескольких пропегаторов, входных полей и вариантов расчета выполняется формирование пакетов параметров. В каждом пакете параметров лишь одно исходное поле, один вариант расчета и не более одного пропегатора (если нужен). Таким образом, при обработке задания на расчет формируются и обрабатываются пакеты параметров.

Существуют параметры общие для всех пакетов. К ним относятся параметры задания волны, показателя преломления элемента, задания префикса (приставки) всех выходных файлов и другие.

Для того чтобы различать файлы, формируемые при выполнении задания, каждому пакету приписывается имя. Имя пакета формируются из:

- приставки для всех файлов (задается в **result.filePrefix**),
- суффикса поля (может отсутствовать),
- суффикса пропегатора (может отсутствовать),
- суффикса варианта расчета (может отсутствовать).

Суффиксы дополняются последовательно, отделяясь от предыдущей части подчеркиванием.

При выполнении для каждого пакета в файл протокола выводятся:

- строка имени пакета,
- параметры пакета,
- характеристики входного поля ( энергия, минимальное и максимальное значения интенсивности ),
- список имен сформированных файлов.

В зависимости от варианта расчета для каждого пакета формируется один или более файлов. Имена всех файлов пакета начинаются с имени пакета.

### 3.7.1.2 Пакеты несовместимых параметров

В некоторых случаях могут сформироваться пакеты несовместимых параметров.

Можно выделить два типа таких пакетов:

- несоответствие входного поля и пропегатора ( векторное поле для скалярного пропегатора или наоборот ),
- вывод графиков или изображений спектра для пропегатора, в котором нет спектра.

Это не считается ошибкой. В этом случае просто выдается сообщение о несоответствии и на этом обработка пакета параметров заканчивается.

### 3.7.2 Варианты запуска программы.

Существует различные варианты запуска программы:

- запуск на расчет на персональном компьютере под Win\*,
- запуск на расчет на персональном компьютере под Linux,
- запуск на расчет в фоновом режиме на кластере.
- для выдачи краткой информации,

#### 3.7.2.1 Запуск на расчет под Win\*.

Для запуска на расчет надо запустить программу, указав имя файла параметров:

Например,

**FieldPropagators pars.\_pars**

#### 3.7.2.2 Запуск на расчет под Linux.

Для запуска на расчет надо запустить программу, указав имя файла параметров:

Например,

**.\FieldPropagators pars.\_pars**

#### 3.7.2.3 Запуск на расчет в фоновом режиме на кластере.

Для запуска на расчет в фоновом режиме необходимо сформировать командный файл для диспетчера задач и запустить процесс командой **qsub comFile**

При этом в командном файле в строке запуска программы надо указать имя файла параметров.

Командный файл comFile :

```
#PBS -l nodes=14:ppn=8
# Script for running MPI sample program on Any processors on cluster
cd $PBS_O_WORKDIR
echo #PBS -o $DIR/stdout.log
echo "Current working directory is `pwd`"
echo "Node file: $PBS_NODEFILE :"
```



```

echo "_____ "
cat $PBS_NODEFILE
echo "_____ "
NUM_PROCS="/bin/awk 'END {print NR}' $PBS_NODEFILE"
echo "Running on $NUM_PROCS processors."

echo "Starting run at: `date`"
# /usr/mp/intel/openmpi-1.2.8/bin/mpirun -np 16 -hostfile /home/sv/cluster_ns ./FieldPropagators
p_pars
/usr/mp/intel/openmpi-1.2.8/bin/mpirun -np $NUM_PROCS -hostfile $PBS_NODEFILE
./FieldPropagators p_pars
echo "Job finished at: `date`"

```

При этом в первой строке указывается число узлов (14 в данном примере) и число процессоров на каждом узле (8). Максимальные значения этих параметров зависят от конфигурации конкретного кластера. Необязательно использовать все узлы кластера, т.е. если на кластере есть 15 узлов можно задать для программы от 1 до 15 узлов.

### 3.7.2.4 Выдача справочной информации

Если запустить программу без параметров, то на консоль будет выдано краткое описание языка задания параметров:

```

main properties
wavef.length = <realValue>
mediaRefractiveIndex = <complexValue>
result.filePrefix = <stringValue>
result.isTrace = <booleanValue={ false | true }>

additional output ( optional )
result.additionalOutput = amplitude
result.additionalOutput = intensitySquare
result.additionalOutput = topologyForPlanImage

one or more of
target = objectRadialGraph <angleInGrad> <r1> <r2> <pointN> | : <postfix> |
target = objectImage | : <postfix> |
target = objectField | : <postfix> |
target = spectrumRadialGraph <angleInGrad> <r1> <r2> <pointN> | : <postfix> |
target = spectrumImage | : <postfix> |
target = weightedSpectrumRadialGraph <angleInGrad> <r1> <r2> <pointN> <w> | : <postfix> |
target = weightedSpectrumImage <w> | : <postfix> |
target = radialGrpAccrossOz <angleInGrad> <r1> <r2> <pointN> <w> | : <postfix> |
target = graphAccrossOz <x1> <y1> <x2> <y2> <pointN> <w> | : <postfix> |
target = imageAccrossOz <xSize> <xN> <ySize> <yN> <w> | : <postfix> |
target = fieldAccrossOz <xSize> <xN> <ySize> <yN> <w> | : <postfix> |
target = graphAlongOz <x> <y> <z1> <z2> <zN> | : <postfix> |
target = imageAlongOzByXY <x1> <y1> <x2> <y2> <xyN> <z1> <z2> <zN> | : <postfix> |
target = imageAlongOzByR <angleInGrad> <r1> <r2> <rN> <z1> <z2> <zN> | : <postfix> |
target = timeGraphOfCrossSections <xSize> <xN> <xN_step> <ySize> <yN> <yN_step> <w> <graph-
PointN> | : <postfix> |

one or more of
propagator = scalarFourier | : postfix |
propagator = scalarFresnel | : postfix |
propagator = scalarByPlaneWaveExpansion <yN> <yHS> <xN> <xHS> <oat> | : postfix |
propagator = scalarRayleighSommerfeld | : postfix |
propagator = vectorRayleighSommerfeld | : postfix |
propagator = vectorByPlaneWaveExpansionMansuripur <yN> <yHS> <xN> <xHS> <oat> | : postfix |
propagator = vectorByPlaneWaveExpansionMansuripur+FresnelCoeff <yN> <yHS> <xN> <xHS>
<oat> <n> <correctoinType> | : postfix |

```

```

propagator = vectorByPlaneWaveExpansionStratton <yN> <yHS> <xN> <xHS> <oat> <delta> | :
    postfix |
propagator = vectorByPlaneWaveExpansionStratton+FresnelCoeff <yN> <yHS> <xN> <xHS> <oat>
    <n> <correctoinType> <delta> | : postfix |
    <yHS> <xHS> - halfwidthes
    <oat> = { pointwise | cellwise } - objectValueApproximationType
    <n> - elementRefractiveIndex ( complex )
    <correctoinType> = { no reliefSide planeSide bothSides }

one ore more objectField from
objectField = fieldForScalarPropagator <xR> <xN> <yR> <yN> <fName> | : <postfix> |
objectField = fieldForVectorPropagatorWithPolarizationCoef <xR> <xN> <yR> <yN> <fName> <cx>
    <cy> | : <postfix> |
objectField = fieldForVectorPropagatorAsTwoPolarizationField <xR> <xN> <yR> <yN> <fNameX>
    <fNameY> | : <postfix> |
    <xR> <yR> - metricalHalfSideSizes
    <xN> <yN> - pixelSizes
    <fName> <fNameX> <fNameY> - fieldName

simpleElementField list ( with parNames and comments after '#' )
<fieldName> = ConstantVortical <angularOrder>
<fieldName> = SuperGaussVortical <s> <p> <m> # exp( -(r/s)^p ) * exp( i*m*r )
<fieldName> = GaussLaguerre <gaussianRadius> <radialOrder> <angularOrder>
<fieldName> = Bessel <alfa> <angularOrder>
<fieldName> = Zernike <radialOrder> <angularOrder> <maxR>
<fieldName> = GaussHermite <gaussianRadius> <horizontalOrder> <verticalOrder>
<fieldName> = RingSet { : <complexCoef> <ringExternalRadius> }
<fieldName> = FieldInPolarCoordsFromFile <fileName>
<fieldName> = CartesianFieldFromFile <fileName>
<fieldName> = FieldBy1woCrossedGraphs <horGraphFileName> <verGraphFileName>
<fieldName> = RandomPhase
<fieldName> = RadialPower <a> <b> # a*r^b, a-complex, b-real
<fieldName> = ExponentOfRadialPower <a> <b> # exp( a*r^b ), a-complex, b-real
<fieldName> = ExponentOfRadialLogarithm <a> <b> # exp( a*ln(|b*r|) ), a-complex, b-real
<fieldName> = DecartPower <a> <b> <c> <d> # a*x^b + c*y^d, a,c-complex, b,d-real
<fieldName> = ExponentOfDecartPower <a> <b> <c> # exp( a*x^b*y^c ), a-complex, b,c-real
<fieldName> = ExponentOfDecartLogarithm <a> <b> <c> <d> # exp( a*ln(|b*x|) + c*ln(|d*y|) ),
    a,c-complex, b,d-real

ElementFieldFunction list ( with parNames and comments after '#' )
<fieldName> = SumOf( { : <name> <complexCoef> }
<fieldName> = ProductOf( { <name> }
<fieldName> = ExponentOf( <c> <name> # exp( c*f(..) ) c-complex
<fieldName> = PowOf( <a> <b> <name> # a*( f(..)^b ) a-complex, b-real
<fieldName> = KinofomOf( <name>
<fieldName> = MultiLevelKinofomOf( <levelNumber> <name>
<fieldName> = KinofomBinarizationBy1.levelOf( <level> <name>
    <level> - realValue from [0, 1]
<fieldName> = PhaseOf( <name>
<fieldName> = AmplitudeOf( <name>
<fieldName> = DecartTransformOf <a> <n> <b> <c> <m> <d> <g> <h> <name> # F( a*x^n+b,
    c*y^m+d )*g+h, a,b,c,d-real, g,h-complex, n,m-int
<fieldName> = RadialTransformOf <a> <b> <c> <g> <h> <name> # F( a*r^b+c )*g+h, a,b,c-real, g,h-
    complex
<fieldName> = RingCompositeOf { : <ci> <name> <externalR> } # ci*Fi( r ), ci-complex
    if <name> = unused, value is equal to <ci>
<fieldName> = RotationOf <angleInGrad> <name>
<fieldName> = UpDownCompositeOf <UpName> <DownName>
<fieldName> = QuadrantCompositeOf <UpRightName> <UpLeftName> <DownLeftName> <Down-
    RightName>
<fieldName> = KinofomWithBinaryCodingByGratings <period> <fillFactor> <name>

```

### 3.7.3. Описание языка параметров.

#### 3.7.3.1 Правила задания параметров.

1. Любая информация после знака # воспринимается как комментарий.
2. Каждый параметр задается в отдельной строке.
3. Все параметры задаются в формате

**<имяПараметра> = <строкаЗначения>**

При этом формат и семантика <строкиЗначения> определяется семантикой для <имяПараметра>.

4. Строки параметров в файле параметров могут располагаться в любом порядке.

5. Действительное число может задаваться в экспоненциальном формате (например, 1.5e-6).

6. Комплексные числа задаются без пробелов в одном из форматов:

- <действительнаяЧасть><знак><мнимаяЧасть>\*i ( например, 2-1\*i 5.5-3.3\*i ),

- <действительнаяЧасть> ( например, -2 5.5 ),

- <мнимаяЧасть>\*i ( например, 1\*i -3.3\*i ).

7. Лишние параметры игнорируются.

#### 3.7.3.2 Типы параметров.

Можно выделить следующие типы параметров:

- общие для всех пакетов,
- параметры задания одного или нескольких вариантов расчета,
- параметры задания одного или нескольких пропегаторов,
- параметры задания одного или нескольких входных полей.

А. Общие для всех пакетов параметры

Общие параметры едины для всех пакетов параметров (см. пакеты параметров).

К ним относятся:

**waveLength = <realValue>**

**mediaRefractiveIndex = <complexValue>**

**result.filePrefix = <stringValue>**

**result.isTrace = <booleanValue={ false | true }>**

и параметры дополнительного вывода

**result.additionalOutput = amplitude**

**result.additionalOutput = intensitySquare**

**result.additionalOutput = topologyForPlanImage**

**mpi.isParallelByPoint = true**

Первые два параметра не требуют пояснений.

Все выходные файлы будут начинаться со строки, заданной в result.filePrefix.

Так если <stringValue> задано как adb, то файл протокола будет иметь имя 'adb\_info'

и имена всех выдаваемых файлов будут начинаться со строки `adb_` (например, `adb_int.bmp`).

Параметр **result.isTrace** определяет необходимость выдачи на консоль строки выполнения. При запуске на кластере не имеет смысла, т.к. вывод на консоль буферизуется. По умолчанию имеет значение `false`.

Примеры.

```
waveLength = 2
waveLength = 5.7
waveLength = 1e-6
mediaRefractiveIndex = 1.5
mediaRefractiveIndex = 2+.1*i
result.filePrefix = a
result.filePrefix = results_for
result.filePrefix = adb
result.isTrace = false
result.isTrace = true
```

Каждый из этих параметров (кроме **result.isTrace**) обязан быть задан один раз в файле параметров. Т.е. наличие двух строк задания параметра или отсутствие задания (например, для **waveLength**) вызовет ошибку. Диагностика о ней будет выдана в файл протокола.

При выводе изображений выводятся изображения интенсивности и фазы. Если требуется выводить изображение амплитуды, то в файл параметров надо добавить строку

```
result.additionalOutput = amplitude
```

Если требуется выводить изображение квадрата интенсивности, то в файл параметров надо добавить строку

```
result.additionalOutput = intensitySquare
```

Если в файл параметров добавить строку

```
result.additionalOutput = topologyForPlanImage
```

то для всех продольных изображений ( кроме изображений фазы ) будут сформированы и изображения с поиском экстремума поперек оси OZ.

Параметр **mpi.isParallelByPoint** имеет смысл применять только при исследованиях времени расчета поперечных или продольных сечений поля. По умолчанию имеет значение `true`. В этом случае в данные рассчитанные в не главном процессе передаются по одной точке. Если этому параметру задать значение `false`, то данные передаются по строкам для поперечных сечений и по столбцам для продольных (чтобы минимизировать время расчета).

Б. Параметры задания вариантов расчета.

В файле параметров должен быть один или несколько вариантов расчета. Если задано более одного варианта расчета, то они должны иметь различные суффиксы ( `<postfix>` ). В файле параметров может быть не более одного варианта расчета без суффикса. Далее идут описания параметров для каждого вариантов расчета.

**ЗАМЕЧАНИЕ:** при задании параметров радиальных графиков и изображений, если задать значение радиуса отрицательным, то будет взята точка с положительным радиусом и противоположным углом (смещенным на  $\pi$ )

**ЗАМЕЧАНИЕ:** при выводе изображений выводятся изображения интенсивности, фазы и других, заданных в параметрах дополнительного вывода; при этом имя файла интенсивности имеет вид \*\_int.bmp, фазы - \*\_pha.bmp, амплитуды - \*\_amp.bmp, топологии интенсивности - \*\_int\_topo.bmp, топологии амплитуды - \*\_amp\_topo.bmp

**ЗАМЕЧАНИЕ:** структура входного поля зависит от типа пропагатора; для скалярных пропагаторов поле имеет одну компоненту, для векторных две поперечные компоненты поляризации (вдоль OX и вдоль OY); поэтому при формировании графиков или изображений входного поля для скалярного поля будет сформирован один график или изображения одного поля, а для векторного – графики или изображения полей компонент, при этом для X-ой компоненты к имени файла добавляется суффикс ‘\_x’, для Y-вой – ‘\_y’; со спектрами – аналогично.

**ЗАМЕЧАНИЕ:** для скалярных пропагаторов поле после элемента имеет одну компоненту, для векторных три компоненты электрического поля (X, Y и Z); поэтому при формировании графиков или изображений поля для скалярного поля будет сформирован один график или изображения одной компоненты поля, а для векторного – графики или изображения полей компонент, при этом для X-ой компоненты к имени файла добавляется суффикс ‘\_x’, для Y-вой – ‘\_y’, для Z-вой – ‘\_z’, для интенсивности поперечного поля – ‘\_1’, для интенсивности поля – ‘\_’.

**target = objectRadialGraph <angleInGrad> <r1> <r2> <pointN> | : <postfix> |**

радиальный график исходного поля, где  
<angleInGrad> - угол в градусах,  
<r1> - начальный радиус,  
<r2> - конечный радиус  
<pointN> - число точек графика

**target = objectImage | : <postfix> |**

изображения входного поля  
нет параметров

**target = fieldImage | : <postfix> |**

входное поле в файл ( файлы )  
нет параметров

**target = spectrumRadialGraph <angleInGrad> <r1> <r2> <pointN> | : <postfix> |**

радиальный график спектра ( только для пропагаторов, имеющих спектр )  
<angleInGrad> - угол в градусах, где  
<r1> - начальный радиус,

<r2> - конечный радиус  
<pointN> - число точек графика

**target = spectrumImage [ : <postfix> ]**  
изображения спектра ( только для пропагаторов, имеющих спектр ),  
нет параметров

**target = weightedSpectrumRadialGraph <angleInGrad> <r1> <r2> <pointN>  
<z> [ : <postfix> ]**

радиальный график взвешенного спектра (только для пропагаторов, имеющих спектр), где

<angleInGrad> - угол в градусах,  
<r1> - начальный радиус,  
<r2> - конечный радиус  
<pointN> - число точек графика  
<z> - значение координаты Z

**target = weightedSpectrumImage <z> [ : <postfix> ]**

изображения взвешенного спектра ( только для пропагаторов, имеющих спектр ), где

<z> - значение координаты Z

**target = radialGrapAccrossOz <angleInGrad> <r1> <r2> <pointN> <z> [ :  
<postfix> ]**

радиальный график в поперечном сечении, где

<angleInGrad> - угол в градусах,  
<r1> - начальный радиус,  
<r2> - конечный радиус  
<pointN> - число точек графика  
<z> - значение координаты Z

**target = grapAccrossOz <x1> <y1> <x2> <y2> <pointN> <z> [ : <postfix> ]**

график в поперечном сечении, где

<x1> <y1> - координаты начальной точки  
<x2> <y2> - координаты конечной точки  
<pointN> - число точек графика  
<z> - значение координаты Z

**target = imageAccrossOz <xSize> <xN> <ySize> <yN> <z> [ : <postfix> ]**

изображения поперечного сечения ( центрального ), где

<xSize> <xN> - метрический и пиксельный размер вдоль оси OX  
<ySize> <yN> - метрический и пиксельный размер вдоль оси OY  
<z> - значение координаты Z

**target = fieldAccrossOz <xSize> <xN> <ySize> <yN> <z> [ : <postfix> ]**

поперечное сечение поля в файл, где

<xSize> <xN> - метрический и пиксельный размер вдоль оси OX

<ySize> <yN> - метрический и пиксельный размер вдоль оси OY  
<z> - значение координаты Z

**target = graphAlongOz <x> <y> <z1> <z2> <zN> [ : <postfix> ]**  
продольный график ( на отрезке параллельном оси OZ ), где  
<x> <y> - координаты точки в поперечном сечении  
<z1> <z2> - координаты начальной и конечной точек  
<zN> - число точек графика

**target = imageAlongOzByXY <x1> <y1> <x2> <y2> <xyN> <z1> <z2> <zN> [ : <postfix> ]**  
изображения продольного сечения, где  
<x1> <y1> <x2> <y2> - координаты концов отрезка в поперечном сечении  
<xyN> - число пикселей в поперечном сечении ( по вертикали )  
<z1> <z2> - координаты концов отрезка в продольном сечении  
<zN> - число пикселей в продольном сечении ( по горизонтали )

**target = imageAlongOzByR <angleInGrad> <r1> <r2> <rN> <z1> <z2> <zN> [ : <postfix> ]**  
изображения продольного радиального сечения, где  
<angleInGrad> - угол в градусах  
<r1> <r2> - координаты концов отрезка в поперечном сечении  
<rN> - число пикселей в поперечном сечении ( по вертикали )  
<z1> <z2> - координаты концов отрезка в продольном сечении  
<zN> - число пикселей в продольном сечении ( по горизонтали )

**target = timeGraphOfCrossSections <xSize> <xN> <xN\_step> <ySize> <yN> <yN\_step> <z> <graphPointN> [ : <postfix> ]**  
график времени вычисления поперечных сечений изменяемых размеров,  
где  
<xSize> <xN> <xN\_step> - метрический и пиксельный размер вдоль оси OX и приращение пиксельного размера,  
<ySize> <yN> <yN\_step> - метрический и пиксельный размер вдоль оси OY и приращение пиксельного размера,  
<z> - значение координаты Z  
<graphPointN> - число точек графика

этот вариант используется совместно с параметром **mpi.isParallelByPoint**, который и определяет вариант расчета и передачи данных поля ( по точкам или по строкам/столбцам ).

В. Параметры задания пропатора ( пропаторов ).

В файле параметров может быть задан один или несколько пропаторов. Если задано более одного пропатора, то они должны иметь различные суффиксы ( <postfix> ). В файле параметров может быть не более одного пропатора без суффикса. Если в файле параметров заданы только варианты вывода входного поля (нет вариантов расчета поля или вывода спектра), то задание

пропагатора не требуется. Далее идут описания параметров для каждого пропагатора.

**ЗАМЕЧАНИЕ:** для пропагаторов, имеющих спектр, задаются следующие параметры:

<yN> <yHS> - пиксельный и метрический размер по вертикали  
<xN> <xHS> - пиксельный и метрический размер по горизонтали  
<oat> - вариант аппроксимации значений исходного поля, может принимать одно из значений pointwise или cellwise, поточечной или ступенчатой аппроксимации соответственно

**propagator = scalarFourier [ : postfix ]**  
скалярный пропагатор Фурье

**propagator = scalarFresnel [ : postfix ]**  
скалярный пропагатор Френеля

**propagator = scalarByPlaneWaveExpansion <yN> <yHS> <xN> <xHS> <oat> [ : postfix ]**  
скалярный пропагатор разложения по плоским волнам  
<yN> <yHS> <xN> <xHS> <oat> - параметры спектра

**propagator = scalarRayleighSommerfeld [ : postfix ]**  
скалярный пропагатор Рэллея-Зоммерфельда

**propagator = vectorRayleighSommerfeld [ : postfix ]**  
векторный пропагатор Рэллея-Зоммерфельда

**propagator = vectorByPlaneWaveExpansionMansuripur <yN> <yHS> <xN> <xHS> <oat> [ : postfix ]**  
векторный пропагатор Мансирипура ( разложения по плоским волнам )  
<yN> <yHS> <xN> <xHS> <oat> - параметры спектра

**propagator = vectorByPlaneWaveExpansionMansuripur+FresnelCoeff <yN> <yHS> <xN> <xHS> <oat> <n> <correctoinType> [ : postfix ]**  
векторный пропагатор Мансирипура с коэффициентами Френеля  
<yN> <yHS> <xN> <xHS> <oat> - параметры спектра  
<n> - комплекснозначный показатель преломления элемента  
<correctoinType> - тип коррекции, одно из значений { no reliefSide planeSide bothSides }, которые соответствуют вариантам: без учета коррекции коэффициентов Френеля, учет прохождения только через рельефную сторону оптического элемента, учет прохождения только через плоскую сторону оптического элемента и учет прохождения через обе стороны оптического элемента.

**propagator = vectorByPlaneWaveExpansionStratton <yN> <yHS> <xN> <xHS> <oat> <delta> [ : postfix ]**  
векторный пропагатор Страттона ( разложения по плоским волнам )  
<yN> <yHS> <xN> <xHS> <oat> - параметры спектра



<delta> - параметр, определяющий размер игнорируемой области в синк-  
тральной плоскости в районе частот, близких к единице (область наличия осо-  
бенности).

```
propagator = vectorByPlaneWaveExpansionStratton+FresnelCoeff <yN>
<yHS> <xN> <xHS> <oat> <n> <correctoinType> <delta> | :
postfix ]
```

Г. Параметры задания одного или нескольких входных полей.

В файле параметров должно быть задано одно или несколько входных по-  
лей. Если задано более одного входного поля, то они должны иметь различные  
суффиксы (<postfix>). В файле параметров может быть не более одного вход-  
ного поля без суффикса. Далее идут описания параметров для каждого входно-  
го поля.

Можно задавать три варианта входных полей:

- поле для скалярных пропагаторов ( задается одной компонентой ),
  - поле для векторных пропагаторов ( задается одной компонентой и коэф-  
фициентами поляризации ),
  - поле для векторных пропагаторов ( задается двумя компонентами поля ).
- Если будет нарушено соответствие входного поля и пропагатора ( напри-  
мер,

скалярный пропагатор и векторное поле ), то выдается сообщение, но про-  
грамма не прерывается по ошибке.

Можно задать только центрированные поля ( ось OZ проходит через  
центр поля ).

Строки задания входных полей имеют следующую структуру:

```
objectField = fieldForScalarPropagator <xR> <xN> <yR> <yN> <fName> | :
<postfix> ]
```

```
objectField = fieldForVectorPropagatorWithPolarizationCoef <xR> <xN>
<yR> <yN> <fName> <cx> <cy> [ : <postfix> ]
```

```
objectField = fieldForVectorPropagatorAsTwoPolarizationField <xR> <xN>
<yR> <yN> <fNameX> <fNameY> | : <postfix> ]
```

где

- <xR> <yR> - метрические полуширины области определения поля
- <xN> <yN> - пиксельные размеры поля
- <fName> <fNameX> <fNameY> - имена полей

Строка задания поля имеет следующую структуру:

```
<имяПоля> = <гмиПоля> [ <параметры> ]
```

Поля формируются из простых полей и полей, заданных как функция или  
суперпозиция других полей.

Например.

```
objectField = fieldForScalarPropagator 1 100 1 100 a1
objectField = fieldForScalarPropagator 1 100 1 100 a2 : a2
objectField = fieldForScalarPropagator 1 100 1 100 a3 : aa3
# <fieldName> = RandomPhase
a1 = RandomPhase
```

```
# <fieldName> = RadialComplexPower <a> <b> # a*r^b, a-complex, b-real
a2 = RadialComplexPower 3 2
# <fieldName> = UpDownCompositeOf <UpName> <DownName>
a3 = UpDownCompositeOf a1 a2
```

Строки комментариев выделены серым цветом. Они приведены для пояснения семантики параметров.

В данном примере первое поле ( без суффикса ) имеет случайную фазу при постоянной амплитуде. Второе поле имеет суффикс a2 и определяется как радиальная функция  $3*r^2$ . Третье поле ( суффикс aa3 ) составное: в верхней половине случайная фаза, а в нижней - радиальная функция. Задание второго поля демонстрирует также, что суффикс поля и имя функции могут совпадать.

*Структура файла графиков ( \*.gr ).*

Данный формат предназначен для сохранения одного или нескольких графиков

в одном файле. Каждый график сопровождается строкой комментария. Можно сохранять комплекснозначный график.

<строкаКомментария>

<строкаОписания>

<строкаОднойТочки>

...

<строкаОднойТочки>

<строкаОписания> := <числоТочек> [ <индексЦвета> ]

<строкаОднойТочки> := <xValue> <yValue>

<xValue> := <действительноеЧисло>

<yValue> := <realValue> [ <imagePartOfComplexValue> ]

<realValue> := <действительноеЧисло>

## Примеры.

График содержит один отрезок,  $x_1=0, y_1=1, x_2=2, y_2=4$

```
-----
example: line
2
0 1
2 4
-----
```

График содержит один отрезок,  $x_1=0, y_1=-1e+27, x_2=2, y_2=4.66$

```
-----
example: line
2
-0.1 -1e+27
2.2 4.66
-----
```

График содержит 3 точки:  $x_1=5, y_1=1, x_2=2, y_2=4, x_3=0, y_3=5$

```
-----
example: line
2
5 1
-----
```

2 4  
0 5

-----  
Два графика в файле

Первый график содержит один отрезок,  $x1=0, y1=1, x2=2, y2=4$ .

Второй график содержит один отрезок,  $x1=10, y1=1, x2=12, y2=4$ .

-----  
line 1  
2  
0 1  
2 4  
line 1  
2  
10 1  
12 4  
-----

Комплекснозначный график содержит 3 точки:

$x1=0, y1=Complex(1, 2), x2=2, y2=Complex(4, 3), x1=5, y1=Complex(5, 4)$

-----  
example: line  
2  
0 1 2  
2 4 3  
5 5 4  
-----

Комплекснозначный график содержит 3 точки:

$x1=0, y1=Complex(1, 0), x2=2, y2=Complex(4, 3), x1=5, y1=Complex(5, 0)$

-----  
example: line  
2  
0 1  
2 4 3  
5 5  
-----

*Список базовых функций.*

**<fieldName> = ConstantVortical <angularOrder>**

Вихревое поле с постоянной амплитудой

$$\Psi_m(\varphi) = \exp(im\varphi)$$

$m$  - <angularOrder> - порядок вихря (действительное)

**<fieldName> = SuperGaussVortical <s> <p> <m> #  $\exp(-(r/s)^p) * \exp(i*m*\varphi)$**

Вихревое поле с супергауссовой амплитудой

$$\Psi_m(r, \varphi) = \exp\left(-\left(\frac{r}{\sigma}\right)^p\right) \exp(im\varphi)$$

$\sigma$  - <s> - гауссовый радиус (действительное)

$p$  - <p> - экспоненциальная степень (действительное)

$m$  - <m> порядок вихря (действительное)

**<fieldName> = GaussLaguerre <gaussianRadius> <radialOrder> <angularOrder>**

Моды Гаусса-Лагерра:

$$\Psi_{nm}(r, \varphi) = \frac{1}{\sigma} \sqrt{\frac{n!}{\pi(n+|m|)!}} \exp\left(-\frac{r^2}{2\sigma^2}\right) \left(\frac{r}{\sigma}\right)^{|m|} L_n^{(|m|)}\left(\frac{r^2}{\sigma^2}\right) \exp(im\varphi), \quad n \geq 0$$

$\sigma$  - <gaussianRadius> - гауссовый радиус (действительное)

$n$  - <radialOrder> - радиальный индекс (целое, положительное)

$m$  - <angularOrder> - угловой индекс (целое)

**<fieldName> = Bessel <alfa> <angularOrder>**

Моды Бесселя:

$$\Psi_{\alpha m}(r, \phi) = \frac{(-i)^m}{r_0 \sqrt{\pi}} J_m(\alpha r) \exp(im\phi)$$

$\alpha$  - <alfa> - масштабный радиальный индекс (действительное)

$m$  - <angularOrder> - угловой индекс (целое)

**<fieldName> = Zernike <radialOrder> <angularOrder> <maxR>**

Функции Цернике:

$$\Psi_{nm}(r, \varphi) = \frac{1}{r_0} \sqrt{\frac{n+1}{\pi}} R_n^{(|m|)}(r/\sigma) \exp(im\varphi), \quad n \geq 0, \quad |m| \leq n, \quad (n-m) - \text{четное},$$

$n$  - <radialOrder> - радиальный индекс (целое, положительное)

$m$  - <angularOrder> - угловой индекс (целое)

$\sigma$  - <maxR> (действительное)

**<fieldName> = GaussHermite <gaussianRadius> <horizontalOrder> <verticalOrder>**

Моды Гаусса-Эрмита:

$$\Psi_{nm}(x, y) = \frac{1}{\sigma} \sqrt{\frac{(-i)^{n+m}}{2^{n+m} n! m!}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) H_n\left(\frac{x}{\sigma}\right) H_m\left(\frac{y}{\sigma}\right), \quad n, m \geq 0$$

$\sigma$  - <gaussianRadius> - гауссовый радиус (действительное)

$n$  - <horizontalOrder> - индекс по горизонтали (целое, положительное)

$m$  - <verticalOrder> - индекс по вертикали (целое, положительное)

**<fieldName> = RingSet { : <complexCocf> <ringExternalRadius> }**

Кусочно-постоянная радиальная функция:

$$g(r) = \begin{cases} a_1, & r \in [0, r_1] \\ a_2, & r \in [r_1, r_2] \\ \dots \end{cases}$$

**<fieldName> = FieldInPolarCoordsFromFile <fileName>**

Поле в полярных координатах, считанное из файла специального формата.

**<fieldName> = CartesianFieldFromFile <fileName>**

Поле в декартовых координатах, считанное из файла специального формата.

**<fieldName> = FieldByTwoCrossedGraphs <horGraphFileName> <verGraphFileName>**

Поле в декартовых координатах, созданное как произведение одномерных распределений по координатам  $x$  и  $y$ , считанных из соответствующих файлов.

**<fieldName> = RandomPhase**

Случайная фаза

**<fieldName> = RadialPower <a> <b> #  $a^*r^b$ , a-complex, b-real**

$$g(r) = ar^b$$

$a$  – <complexCoeff> (комплексное)

$b$  – <power> (действительное)

**<fieldName> = ExponentOfRadialPower <a> <b> #  $\exp(a^*r^b)$ , a-complex, b-real**

$$g(r) = \exp(ar^b),$$

$a$  – <complexCoeff> (комплексное)

$b$  – <Power> (действительное)

**<fieldName> = ExponentOfRadialLogarithm <a> <b> #  $\exp(a^*\ln(|b*r|))$ , a-complex, b-real**

$$g(r) = \exp(a \ln |br|),$$

$a$  – <a> (комплексное)

$b$  – <b> (действительное)

**<fieldName> = DecartPower <a> <b> <c> <d> #  $a^*x^b + c^*y^d$ , a,c-complex, b,d-real**

$$g(x, y) = ax^b + cy^d$$

$a$  – <a> (комплексное)

$b$  – <b> (действительное)

$c$  – <c> (комплексное)

$d$  – <d> (действительное)

**<fieldName> = ExponentOfDecartPower <a> <b> <c> #  $\exp(a^*x^b + c^*y^d)$ , a-complex, b,c-real**

$$g(x, y) = \exp(ax^b + cy^d),$$

$a$  – <a> (комплексное)

$b$  – <b> (действительное)

$c$  – <c> (действительное)

**<fieldName> = ExponentOfDecartLogarithm <a> <b> <c> <d> #  $\exp(a^*\ln(|b*x|) + c^*\ln(|d*y|))$ , a,c-complex, b,d-real**

$$g(x, y) = \exp(a \ln|hx| + c \ln|\phi|),$$

$a$  – <a> (комплексное)

$b$  – <b> (действительное)

$c$  – <c> (комплексное)

$d$  – <d> (действительное)

*Список операций над функциями.*

<fieldName> = **SumOf()** { : <name> <complexCoef> }

- сумма с комплексными коэффициентами;

<fieldName> = **ProductOf()** { <name> }

- произведение;

<fieldName> = **ExponentOf()** <c> <name> #  $\exp(c * f(..))$  c-complex

- экспонента с комплексным коэффициентом

<fieldName> = **PowOf()** <a> <b> <name> #  $a * (f(..)^b)$  a-complex, b-real

- степень с комплексным коэффициентом

<fieldName> = **KinoformOf()** <name>

- создание киноформа: амплитуда заменяется на единичную;

<fieldName> = **MultiLevelKinoformOf()** <levelNumber> <name>

- квантование фазы для нескольких уровней:

$\varphi \in [\pi/2; 3\pi/2]$  заменяется на  $\pi$ , остальные на 0;

<fieldName> = **KinoformBinarizationByLevelOf()** <level> <name>

<level> - realValue from [0, 1]

- создание бинарного киноформа: амплитуда заменяется на единичную, а фаза обинаривается по заданному уровню в долях от  $2\pi$ ;

<fieldName> = **PhaseOf()** <name>

- вычисление фазы поля  $g(x) = \arg[f(x)]$

<fieldName> = **AmplitudeOf()** <name>

- создание амплитудной функции, фаза обнуляется

<fieldName> = **DecartTransformOf** <a> <n> <b> <c> <m> <d> <g> <h>

<name>

#  $F(a * x^n + b, c * y^m + d) * g + h$ , a,b,c,d-real, g,h-complex, n,m-int

Декартовое преобразование вида  $g(x) = f(\alpha x^n + b, \gamma y^m + d)g + h$ ,

<fieldName> = **RadialTransformOf** <a> <b> <c> <g> <h> <name>

#  $F(a * r^b + c) * g + h$ , a,b,c-real, g,h-complex

Радиальное преобразование вида  $g(r) = f(ar^b + c)g + h$ ,

<fieldName> = RingCompositeOf { : <ci> <name> <externalR> }  
# ci\*Fi( r ), ci-complex, if <name> = unused, value is equal to <ci>

Набор функций, расположенных в последовательных вложенных кольцах со соответствующими комплексными множителями

<fieldName> = RotationOf <angleInGrad> <name>

Поворот двумерного распределения на заданный угол

<fieldName> = UpDownCompositeOf <UpName> <DownName>

Поле, составленное из двух половинок, в каждой из которых своя функция

<fieldName> = QuadrantCompositeOf <UpRightName> <UpLeftName>  
<DownLeftName> <DownRightName>

Поле, составленное из четырех квадрантов, в каждом из которых своя функция

### 3.8 Примеры использования программного обеспечения FieldPropagators

В каждом примере дается его описание, приводятся тексты файла параметров и файла протокола.

В файлы параметров добавлены строки комментариев из справочной информации, которые выделены серым цветом.

#### 3.8.1 Пример с одним пакет параметров

Демонстрирует вариант параметров, содержащих один пакет без использования суффиксов.

Файл входных параметров ( строки комментариев выделены серым цветом ):

waveLength = 1

mediaRefractiveIndex = 1.5

result.filePrefix = a

result.isTrace = false

# propagator = scalarFresnel | : postfix |

propagator = scalarFresnel

# objectField = fieldForScalarPropagator <xS> <xN> <yS> <yN> <fName> | : <postfix> |

objectField = fieldForScalarPropagator 2 101 2 101 a

# <fieldName> = ConstantVortical <angularOrder>

a = ConstantVortical 2

# target = fieldImageAccrossOz <xSize> <xN> <ySize> <yN> <z> | : <postfix> |

target = fieldImageAccrossOz 2 101 2 101 5

Файл протокола имеет имя a\_info и содержит:

D:\work\cpp\field\try\FieldPropagators.exe

```
propertyFileName = p_pars
```

строка имени программы и строка, содержащая имя файла параметров, далее идут строки введенных параметров, лишние параметры в этом списке отсутствуют,

в начале каждой строки выводится индекс этой строки в файле параметров

```
----- Input parameters: <index> <propName> = <string>
```

```
1 wavelength = 1
2 mediaRefractiveIndex = 1.5
3 result.filePrefix = a
4 result.isTrace = false
7 propagator = scalarFresnel
10 objectField = fieldForScalarPropagator 2 101 2 101 a
13 a = ConstantVortical 2
16 target = fieldImageAccrossOz 2 101 2 101 5
```

далее идут параметры пакета ( общие параметры не выводятся повторно ) в начале каждой строки выводится индекс этой строки в файле параметров

```
-- prefix = a
10 objectField = fieldForScalarPropagator 2 101 2 101 a
7 propagator = scalarFresnel
16 target = fieldImageAccrossOz 2 101 2 101 5
13 a = ConstantVortical 2
```

далее идут параметры входного поля: энергия, минимальное и максимальное значения

```
<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
a |ConstantVortical| e = 16. min = 1. max = 1.
```

далее перечислены имена сформированных файлов и время выполнения пакета, для изображения интенсивности выдаются характеристики ( максимальное значение, энергия, отношение энергии в изображении к энергии входного поля и отношение энергии к площади изображения )

```
a_int.bmp formed maxV = 0.879317 e = 1.521399 e/e0 = 0.095087 e/s = 0.38035
a_pha.bmp formed
-- Computation time: 25.516000 sec
```

далее идет время работы программы

```
Computation time: 25.531000 sec
```

### 3.8.2 Пример с несколькими входными полями

Демонстрирует построение сложного поля с выводом всех промежуточных полей.

Файл входных параметров ( строки комментариев выделены серым цветом ):

```
wavelength = 1
mediaRefractiveIndex = 1.5
result.filePrefix = a
result.isTrace = false
```

```
# propagator = scalarFresnel | : postfix |
propagator = scalarFresnel
```

```
# objectField = fieldForScalarPropagator <xS> <xN> <yS> <yN> <fName> [ : <postfix> ]
```



```

objectField = fieldForScalarPropagator 1 100 1 100 a1 : a1
objectField = fieldForScalarPropagator 1 100 1 100 a2 : a2
objectField = fieldForScalarPropagator 1 100 1 100 a3 : a3
# <fieldName> = RandomPhase
a1 = RandomPhase
# <fieldName> = RadialComplexPower <a> <b> # a*r^b, a-complex, b-real
a2 = RadialComplexPower 3 2
# <fieldName> = UpDownCompositeOf <UpName> <DownName>
a3 = UpDownCompositeOf a1 a2

# target = objectImage
target = objectImage

```

В данном примере также демонстрируется, что имена полей и строки суффиксов могут совпадать ( a2 ). В этом примере задание пропегатора не требуется, т.к. задан только вывод входного поля. Поэтому строка задания пропегатора игнорируется ( не выводится в списке введенных параметров ).

Файл протокола имеет имя a\_info и содержит:

```

D:\work\cpp\field\try\FieldPropagators.exe
propertyFileName = p_pars
----- Input parameters: <index> <propName> = <string>
1 waveLength = 1
2 mediaRefractiveIndex = 1.5
3 result.filePrefix = a
4 result.isTrace = false
10 objectField = fieldForScalarPropagator 1 100 1 100 a1 : a1
11 objectField = fieldForScalarPropagator 1 100 1 100 a2 : a2
12 objectField = fieldForScalarPropagator 1 100 1 100 a3 : a3
14 a1 = RandomPhase
16 a2 = RadialComplexPower 3 2
18 a3 = UpDownCompositeOf a1 a2
21 target = objectImage

```

далее идет строка, содержащая имя первого пакета, параметры пакета, характеристики входного поля, имена сформированных файлов и время выполнения пакета параметров

```

- prefix = a_a1
10 objectField = fieldForScalarPropagator 1 100 1 100 a1 : a1
7 propagator = scalarFresnel
21 target = objectImage
14 a1 = RandomPhase

<name> |<type>| e = <energy> min = <intensityMin> max = <intensityMax>
a1 |RandomPhase| e = 4. min = 1. max = 1.

a_a1_int.bmp formed maxV = 1. e = 4. e/e0 = 1. e/s = 1.
a_a1 pha.bmp formed
- Computation time: 0.016000 sec

```

далее идет информация для второго пакета параметров

```

- prefix = a_a2
11 objectField = fieldForScalarPropagator 1 100 1 100 a2 : a2
7 propagator = scalarFresnel
21 target = objectImage
16 a2 = RadialComplexPower 3 2

<name> |<type>| e = <energy> min = <intensityMin> max = <intensityMax>
a2 |RadialComplexPower| e = 22.41306 min = 3.747673e-07 max = 36.

```

```

a_a2_int.bmp formed maxV = 36. e = 22.41306 e/e0 = 1. e/s = 5.603265
a_a2 pha.bmp formed
- Computation time: 0.015000 sec

```

далее идет информация для третьего пакета параметров

```

- prefix = a_a3
12 objectField = fieldForScalarPropagator 1 100 1 100 a3 : a3
7 propagator = scalarFresnel
21 target = objectImage
18 a3 = UpDownCompositeOf a1 a2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
a3 [UpDownCompositeOf] e = 13.20653 min = 3.747673e-07 max = 36.
a1 [RandomPhase] e = 2. min = 1. max = 1.
a2 [RadialComplexPower] e = 11.20653 min = 3.747673e-07 max = 36.

```

т.к. поле составное, то выдается информация, как для всего поля, так и для полей, из которых оно состоит

```

a_a3_int.bmp formed maxV = 36. e = 13.20653 e/e0 = 1. e/s = 3.301632
a_a3 pha.bmp formed
- Computation time: 0.016000 sec

```

```

Computation time: 0.063000 sec

```

### 3.8.3 Пример с несколькими вариантами расчета

Данный пример показывает наиболее часто используемую возможность – задание нескольких вариантов расчета.

Файл входных параметров ( строки комментариев выделены серым цветом ):

```

waveLength = 10.6
mediaRefractiveIndex = 1.5
result.filePrefix = a
result.isTrace = true
result.additionalOutput = amplitude

# target = objectImage [ : <postfix> |
target = objectImage : obj
# target = radialGrapAccrossOz <angleInGrad> <r1> <r2> <pointN> <z> [ : <postfix> |
target = radialGrapAccrossOz 0 -2 2 101 5 : z5
# target = imageAccrossOz <xSize> <xN> <ySize> <yN> <z> [ : <postfix> |
target = imageAccrossOz 2 101 2 101 5 : z05
target = imageAccrossOz 2 101 2 101 10 : z10

# propagator = scalarRayleighSommerfeld [ : postfix |
propagator = scalarRayleighSommerfeld

# objectField = fieldForScalarPropagator <xR> <xN> <yR> <yN> <fName> [ : <postfix> |
objectField = fieldForScalarPropagator 5 31 5 31 fun

# <fieldName> = ConstantVortical <angularOrder>
fun = ConstantVortical 2

```

В данном примере заданы :

- вывод изображений входного поля ( target с суффиксом obj ),

- вывод графика поля на расстоянии  $z = 5$  ( суффикс z5 ),
- вывод изображений поля при  $z = 5$  ( суффикс z05 ),
- вывод изображений поля при  $z = 10$  ( суффикс z10 ).

Кроме этого задан дополнительный вывод изображений амплитуды (`result.additionalOutput = amplitude`).

Файл протокола имеет имя `a_info` и содержит:

```
D:\work\cpp\try\fieldPropagators.exe
propertyFileName = p_pars
----- Input parameters: <index> <propName> = <string>
1 waveLength = 10.6
2 mediaRefractiveIndex = 1.5
3 result.filePrefix = a
4 result.isTrace = true
5 result.additionalOutput = amplitude
9 target = objectImage : obj
11 target = radialGrapAccrossOz 0 -2 2 101 5 : z5
13 target = imageAccrossOz 2 101 2 101 5 : z05
14 target = imageAccrossOz 2 101 2 101 10 : z10
17 propagator = scalarRayleighSommerfeld
21 objectField = fieldForScalarPropagator 5 31 5 31 fun
24 fun = ConstantVertical 2

-- prefix = a_obj
21 objectField = fieldForScalarPropagator 5 31 5 31 fun
9 target = objectImage : obj
24 fun = ConstantVertical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun [ConstantVertical] e = 100. min = 1. max = 1.

a_obj_int.bmp formed maxV = 1. e = 100. e/e0 = 1. e/s = 1.
a_obj_amp.bmp formed
a_obj_pha.bmp formed
-- Computation time: 0.000000 sec

-- prefix = a_z5
21 objectField = fieldForScalarPropagator 5 31 5 31 fun
17 propagator = scalarRayleighSommerfeld
11 target = radialGrapAccrossOz 0 -2 2 101 5 : z5
24 fun = ConstantVertical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun [ConstantVertical] e = 100. min = 1. max = 1.

a_z5.gr formed
-- Computation time: 0.078000 sec

-- prefix = a_z05
21 objectField = fieldForScalarPropagator 5 31 5 31 fun
17 propagator = scalarRayleighSommerfeld
13 target = imageAccrossOz 2 101 2 101 5 : z05
24 fun = ConstantVertical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun [ConstantVertical] e = 100. min = 1. max = 1.

a_z05_int.bmp formed maxV = 0.013752 e = 0.007853 e/e0 = 0.000079 e/s = 0.001963
a_z05_amp.bmp formed
```

```

a_z05_pha.bmp formed
-- Computation time: 7.953000 sec

-- prefix = a_z10
21 objectField = fieldForScalarPropagator 5 31 5 31 fun
17 propagator = scalarRayleighSommerfeld
14 target = imageAccrossOz 2 101 2 101 10 : z10
24 fun = ConstantVortical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun [ConstantVortical] e = 100. min = 1. max = 1.

a_z10_int.bmp formed maxV = 0.001359 e = 0.000682 e/e0 = 6.815931e-06 e/s = 0.00017
a_z10_amp.bmp formed
a_z10_pha.bmp formed
-- Computation time: 7.937000 sec

Computation time: 15.984000 sec

```

### 3.8.4 Пример сравнения двух скалярных пропагаторов

В данном примере рассчитываются графики поля двумя пропагаторами.

Файл входных параметров ( строки комментариев выделены серым цветом ):

```

waveLength = 10.6
mediaRefractiveIndex = 1.5
result.filePrefix = a
result.isTrace = true

# target = radialGrapAccrossOz <angleInGrad> <r1> <r2> <pointN> <z> [ : <postfix> ]
target = radialGrapAccrossOz 0 -2 2 101 5

# propagator = scalarRayleighSommerfeld [ : postfix ]
propagator = scalarRayleighSommerfeld : rs

# propagator = scalarFresnel [ : postfix ]
propagator = scalarFresnel : fr

# objectField = fieldForScalarPropagator <xR> <xN> <yR> <yN> <fName> [ : <postfix> ]
objectField = fieldForScalarPropagator 5 101 5 101 fun

# <fieldName> = ConstantVortical <angularOrder>
fun = ConstantVortical 2

```

Т.к. заданы два пропагатора, то при их задании выгодно задавать суффиксы.

Файл протокола имеет имя a\_info и содержит:

```

D:\work\cpp\try\FieldPropagators.exe
propertyFileName = p_pars
----- Input parameters: <index> <propName> = <string>
1 waveLength = 10.6
2 mediaRefractiveIndex = 1.5
3 result.filePrefix = a
4 result.isTrace = true
8 target = radialGrapAccrossOz 0 -2 2 101 5
11 propagator = scalarRayleighSommerfeld : rs
14 propagator = scalarFresnel : fr
17 objectField = fieldForScalarPropagator 5 101 5 101 fun
20 fun = ConstantVortical 2

```

```

-- prefix = a_rs
17 objectField = fieldForScalarPropagator 5 101 5 101 fun
11 propagator = scalarRayleighSommerfeld : rs
8 target = radialGrapAccrossOz 0 -2 2 101 5
20 fun = ConstantVortical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun [ConstantVortical] e = 100. min = 1. max = 1.

a_rs.gr formed
-- Computation time: 0.750000 sec

-- prefix = a_fr
17 objectField = fieldForScalarPropagator 5 101 5 101 fun
14 propagator = scalarFresnel : fr
8 target = radialGrapAccrossOz 0 -2 2 101 5
20 fun = ConstantVortical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun [ConstantVortical] e = 100. min = 1. max = 1.

a_fr.gr formed
-- Computation time: 0.281000 sec

Computation time: 1.047000 sec

```

### 3.8.5 Пример сравнения скалярного и векторного пропагаторов.

Файл входных параметров ( строки комментариев выделены серым цветом ):

```

waveLength = 10.6
mediaRefractiveIndex = 1.5
result.filePrefix = a
result.isTrace = true

# target = radialGrapAccrossOz <angleInGrad> <r1> <r2> <pointN> <z> | : <postfix> |
target = radialGrapAccrossOz 0 -2 2 101 5

# propagator = scalarRayleighSommerfeld | : postfix |
propagator = scalarRayleighSommerfeld : sRS

# propagator = vectorRayleighSommerfeld | : postfix |
propagator = vectorRayleighSommerfeld : vRS

# objectField = fieldForScalarPropagator <xR> <xN> <yR> <yN> <fName> | : <postfix> |
objectField = fieldForScalarPropagator 5 101 5 101 fun : sf

# objectField = fieldForVectorPropagatorWithPolarizationCoef <xR> <xN> <yR> <yN> <fName>
<cx> <cy> | : <postfix> |
objectField = fieldForVectorPropagatorWithPolarizationCoef 5 101 5 101 fun 1. 0. : vf

# <fieldName> = ConstantVortical <angularOrder>
fun = ConstantVortical 2

```

Т.к. для скалярного и векторного пропагаторов необходимо задавать разные входные поля, поэтому заданы два варианта поля. В данном примере видно, что для пакетов с несовместимыми параметрами просто выдается сообщение об этом и никаких файлов не формируется.

Следует обратить внимание, что для скалярного пропагатора выдается один график, а для векторного – комплект графиков: по компонентам (`a_vf_vRS_x.gr`, `a_vf_vRS_y.gr`, `a_vf_vRS_z.gr`), интенсивности поперечного поля (`a_vf_vRS_t.gr`) и интенсивности всего поля (`a_vf_vRS_*.gr`).

Файл протокола имеет имя `a_info` и содержит:

```
D:\work\cpp\try\fieldPropagators.exe
propertyFileName = p_pars
----- Input parameters: <index> <propName> = <string>
1 wavelength = 10.6
2 mediaRefractiveIndex = 1.5
3 result.fileName = a
4 result.isTrace = true
7 target = radialGrpAccrossOz 0 -2 2 101 5
10 propagator = scalarRayleighSommerfeld : sRS
13 propagator = vectorRayleighSommerfeld : vRS
16 objectField = fieldForScalarPropagator 5 101 5 101 fun : sf
19 objectField = fieldForVectorPropagatorWithPolarizationCoef 5 101 5 101 fun 1.0. : vf
22 fun = ConstantVortical 2

-- prefix = a_sf_sRS
16 objectField = fieldForScalarPropagator 5 101 5 101 fun : sf
10 propagator = scalarRayleighSommerfeld : sRS
7 target = radialGrpAccrossOz 0 -2 2 101 5
22 fun = ConstantVortical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun |ConstantVortical| e = 100. min = 1. max = 1.

a_sf_sRS.gr formed
-- Computation time: 0.765000 sec

-- prefix = a_sf_vRS
16 objectField = fieldForScalarPropagator 5 101 5 101 fun : sf
13 propagator = vectorRayleighSommerfeld : vRS
7 target = radialGrpAccrossOz 0 -2 2 101 5
22 fun = ConstantVortical 2

-- incompatible type of field and propagator
-- Computation time: 0.000000 sec

-- prefix = a_vf_sRS
19 objectField = fieldForVectorPropagatorWithPolarizationCoef 5 101 5 101 fun 1.0. : vf
10 propagator = scalarRayleighSommerfeld : sRS
7 target = radialGrpAccrossOz 0 -2 2 101 5
22 fun = ConstantVortical 2

-- incompatible type of field and propagator
-- Computation time: 0.000000 sec

-- prefix = a_vf_vRS
19 objectField = fieldForVectorPropagatorWithPolarizationCoef 5 101 5 101 fun 1.0. : vf
13 propagator = vectorRayleighSommerfeld : vRS
7 target = radialGrpAccrossOz 0 -2 2 101 5
22 fun = ConstantVortical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun |ConstantVortical| e = 100. min = 1. max = 1.

a_vf_vRS_x.gr formed
```

```

a_vf_vRS__y.gr formed
a_vf_vRS__z.gr formed
a_vf_vRS__t.gr formed
a_vf_vRS___.gr formed
– Computation time: 1.313000 sec

```

Computation time: 2.078000 sec

### 3.8.6. Пример вывода топологии продольных сечений.

Файл входных параметров ( строки комментариев выделены серым цветом ):

```

waveLength = 10.6
mediaRefractiveIndex = 1.5
result.filePrefix = a
result.additionalOutput = topologyForPlanImage
result.isTrace = true

# target = imageAlongOzByR <angleInGrad> <r1> <r2> <rN> <z1> <z2> <zN> [: <postfix> |
target = imageAlongOzByR 0 -10 10 101 2 7 101

# propagator = scalarRayleighSommerfeld [: postfix |
propagator = scalarRayleighSommerfeld

# objectField = fieldForScalarPropagator <xR> <xN> <yR> <yN> <fName> [: <postfix> |
objectField = fieldForScalarPropagator 5 101 5 101 fun

# <fieldName> = ConstantVortical <angularOrder>
fun = ConstantVortical 2

```

В данном примере, что для файла a\_int.bmp файл топологии имеет имя a\_int\_topo.bmp. Для изображения фазы изображение топологии не формируется.

Файл протокола имеет имя a\_info и содержит:

```

D:\work\cpp\try\FieldPropagators.exe
propertyFileName = p_pars
----- Input parameters: <index> <propName> = <string>
1 waveLength = 10.6
2 mediaRefractiveIndex = 1.5
3 result.filePrefix = a
4 result.additionalOutput = topologyForPlanImage
5 result.isTrace = true
8 target = imageAlongOzByR 0 -10 10 101 2 7 101
11 propagator = scalarRayleighSommerfeld
14 objectField = fieldForScalarPropagator 5 101 5 101 fun
17 fun = ConstantVortical 2

-- prefix = a
14 objectField = fieldForScalarPropagator 5 101 5 101 fun
11 propagator = scalarRayleighSommerfeld
8 target = imageAlongOzByR 0 -10 10 101 2 7 101
17 fun = ConstantVortical 2

<name> [<type>] e = <energy> min = <intensityMin> max = <intensityMax>
fun [ConstantVortical] e = 100. min = 1. max = 1.

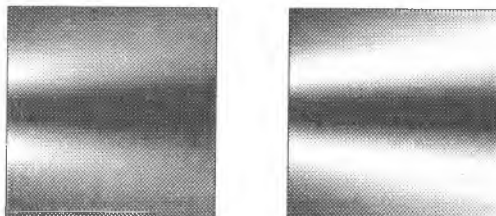
a_int.bmp formed maxV = 0.53096
a_int_topo.bmp formed

```

a pha.bmp formed  
-- Computation time: 76.797000 sec

Computation time: 76.813000 sec

На рис.23 приведены изображения для файла a\_int.bmp ( слева ) и файла a\_int\_toro.bmp ( справа ). Видно, что на втором изображении лучше видна топология распространения света.



*Рис. 23. Иллюстрация топологии.*



#### 4 Программное обеспечение для оптических приложений

Математическое моделирование и вычислительный эксперимент [12-17] являются важными этапами научной и инженерной деятельности.

Имеется достаточно большое количество программных продуктов, предназначенных для моделирования оптических систем. Все программные средства, используемые для проектирования оптических устройств, в той или иной степени обладают возможностями для моделирования конструируемых оптических схем. В зависимости от своих расчетных возможностей и наполнения баз данных, содержащих программное описание выпускаемых мировыми производителями оптических элементов и источников света, существующие коммерческие программы сильно отличаются по цене. В частности, ряд программных продуктов (CODE V, OSLO), позволяющих проектировать только изображающие оптические системы, например, объективы фотоаппаратов. Наиболее мощные программные комплексы (TracePro Expert, ASAP) позволяют проектировать не только изображающие, но и освещающие, фокусирующие и другие оптические системы, однако стоимость таких программных продуктов доходит до 40 тысяч долларов США (ASAP).

##### 4.1 Краткий обзор программных продуктов

В данном параграфе (Табл. 7) приведены краткие характеристики ряда известных программных продуктов, а в последующих параграфах дано более подробное описание программного обеспечения, имеющегося в распоряжении Института компьютерных исследований Самарского государственного аэрокосмического университета (СГАУ).

Таблица 7 Список некоторых программных продуктов

Название	Описание	Производитель
OPAL	Одна из самых известных российских программ для расчета и оптимизации оптических систем.	Санкт-Петербургский государственный институт информационных технологий, точной механики и оптики (технический университет). <a href="http://aco.ifmo.ru/">http://aco.ifmo.ru/</a>
SARO	Программа для расчета и оптимизации оптических систем.	Всероссийский научный центр «Государственный оптический институт им. С.Н.Вавилова» (Санкт-Петербург)
ZEMAX, LensVIEW  (ZELUM, ZEBASE)	ZEMAX - самая известная и популярная на сегодняшний день программа для расчета оптики на персональных компьютерах. Эта программа позволяет анализировать оптические системы на основе последовательного или непоследо-	Компания Focus Software, Inc. Доступны демонстрационные версии: ZEMAX, ZELUM и LensVIEW. <a href="http://www.focus-software.com/">http://www.focus-software.com/</a> <a href="http://www.optima.co.uk/">http://www.optima.co.uk/</a> Модули ZEBASE и LensVIEW позволяют работать с самой обширной библиотекой оптических систем, которая создана и постоянно

	<p>вательного расчета лучей, выполнять глобальную и локальную оптимизацию параметров оптической системы. Программа обладает всеми необходимыми возможностями, позволяющими проектировать современные оптические системы.</p>	<p>обновляется специалистами компании. Дополнительный программный модуль ZELUM помогает разработать осветительную оптическую систему.</p>
<b>CODE V, LightTools</b>	<p>Программа CODE V, разрабатываемая уже на протяжении нескольких десятилетий, обладает самым обширным арсеналом средств для синтеза, анализа и оптимизации оптических систем. Представляет интерес и относительно новый программный продукт LightTools, созданный специалистами этой компании.</p>	<p>Optical Research Associates (ORA) является лидером в области компьютерного проектирования оптических систем. <a href="http://www.opticalres.com/">http://www.opticalres.com/</a></p>
<b>OSLO</b>	<p>OSLO - одна из старейших программ для проектирования оптических систем. Кроме информации о возможностях программы на сайте можно найти учебные материалы по расчету оптики с использованием OSLO. Доступна бесплатная версия OSLO LT.</p>	<p>Фирма Sinclair Optics. <a href="http://www.sinopt.com/">http://www.sinopt.com/</a> Права на распространение и дальнейшее развитие программы переданы Lambda Research Corporation.</p>
<b>TracePro</b>	<p>TracePro - программа расчета лучей с учетом поглощения, отражения, преломления, рассеяния и дифракции света при распространении через оптическую систему.</p>	<p>Компания Lambda Research Corporation предлагает программное обеспечение для проектирования и анализа работы оптических систем. <a href="http://www.lambdares.com/">http://www.lambdares.com/</a></p>
<b>OSD Synosys</b>		<p><a href="http://www.gwi.net/osd">http://www.gwi.net/osd</a></p>
<b>ASAP, ReflectorCAD, APART</b>	<p>Программа ASAP (Advanced Systems Analysis Program) широко используется для проектирования сложных изображающих и осветительных систем спе-</p>	<p>Компания Breault Research Organization, известная также как BRO, является авторитетным разработчиком оптического программного обеспечения. <a href="http://www.breaut.com">http://www.breaut.com</a></p>

	<p>циального назначения. Программа APART предназначена для анализа паразитных засветок. Компания также предлагает специализированный программный продукт ReflectorCAD для конструирования отражателей.</p>	<p>На сайте доступна демонстрационная версия программы <a href="#">ReflectorCAD</a>. На сайте компании работает <a href="#">электронная библиотека источников освещения</a> (BRO Light Source Library), которая содержит конструктивные параметры и математические модели всевозможных источников оптического излучения.</p>
<p><b>PARAXIA-Plus, OPTEC, SIGRAPH-OPTIK/CAOS</b></p>	<p>PARAXIA-Plus - программа для проектирования лазерных систем и моделирования распространения пучков в лазерных резонаторах, OPTEC - программа для проектирования оптических систем и расчета лучей, SIGRAPH-OPTIK/CAOS - программа для проектирования интегральной оптики.</p>	<p>Компания SCIOPT Enterprises. <a href="http://www.sciopt.com">http://www.sciopt.com</a> Компания предлагает широкий спектр оптических программных продуктов, в том числе: PARAXIA-Plus, OPTEC, SIGRAPH-OPTIK/CAOS</p>
<p><b>OptiCAD</b></p>	<p>Трассировка лучей через стандартные или определяемые пользователем компоненты. Позволяет импортировать IGES, STL объекты, существует механизм перевода объектов из ZEMAX в Opticad.</p>	<p>Opticad Corp., США. <a href="http://www.landform.com/opticad">www.landform.com/opticad</a> На сайте производителя доступны для скачивания Демо и полная (!) рабочая версия. Однако без HASP она не работает.</p>
<p><b>OptisWorks</b></p>	<p>Программа для расчета и оптимизации оптических систем, интегрированная в SolidWorks.</p>	<p><a href="http://www.optics.fr/software">http://www.optics.fr/software</a></p>
<p><b>QuickDOE, IterDOE, IterMODE</b></p>	<p>Программное обеспечение по компьютерной оптике</p>	<p>Институт систем обработки изображений РАН (Самара). <a href="http://www.ipsi.smr.ru">www.ipsi.smr.ru</a></p>
<p><b>SimuLight</b></p>	<p>Программное обеспечение для моделирования дифракционных оптических элементов</p>	<p>Институт систем обработки изображений РАН (Самара). <a href="http://www.ipsi.smr.ru">www.ipsi.smr.ru</a></p>

По функциональности оптические программы можно разделить на несколько групп:

- Инструменты для проектирования и расчета детерминированных оптических систем (обычно используют «Straight ray analysis» - анализ методом прямых лучей). К ним относятся объективы фотоаппаратов, телескопы, би

нокли и т.д. Здесь может быть предсказана траектория каждого светового луча. Одновременно осуществляется управление любым световым лучом.

- Программы для расчета оптических систем, где предсказать ход луча невозможно. В английской терминологии этот процесс называется «Nonsequential ray tracing» - непоследовательная трассировка лучей.
- Программы для проектирования светотехники, понимаемой как недетерминированная оптическая система (светильники, автомобильные фары, и фонари).
- Средства для обработки результатов оптических измерений, подготовки математических моделей источников света.

Первый и последний пункты интересуют весьма узкий круг специалистов, требуя сугубо специальных знаний. Поэтому рассмотрим следующие программы: «TracePro» - разработку фирмы «Lambda Research Corporation» (США), и программное обеспечение по компьютерной оптике.

Рассматриваемое программное обеспечение разработано, как правило, для использования в операционных системах семейства Windows. Учитывая то, что на большинстве современных суперкомпьютеров установлены операционные системы семейства Linux, необходимо портирование оптического программного обеспечения на суперкомпьютер. Одним из способов такого портирования может быть использование виртуальной машины, например, VirtualBox. В этом случае виртуальные машины устанавливаются на вычислительные узлы суперкомпьютера. Затем внутри виртуальной машины устанавливается операционная система Windows. И уже затем устанавливается необходимое оптическое программное обеспечение.

#### *4.2 Программное обеспечение по компьютерной оптике*

**Программное обеспечение «QUICK-DOE».** Для моделирования работы дифракционных оптических элементов в «QUICK-DOE» в приближении Френеля реализован дифракционный метод расчета, основанный на быстром преобразовании Фурье (БПФ) [3,18-20], а для более сложных оптических схем – метод расчета трассировки лучей. Ниже дано краткое описание других возможностей программного обеспечения (рис. 24)[3].

Это программное обеспечение включает реализацию аналитических методов расчета матриц фотошаблонов, описывающих фазовые функции ДОО, и методов кодирования функций пропускания оптических элементов. Программное обеспечение также оснащено средствами формирования файлов для вывода на растровые фотопостроители, расчета цифровых голограмм, выполнения ряда вспомогательных функций. В частности, к вспомогательным операциям относятся преобразование форматов файлов, визуализация маски файла, моделирование работы ДОО на основе БПФ, описание и подготовка составного изображения для вывода на фотоплоттер. Программное обеспечение предназначено для использования физиками-оптиками, проектировщиками дифракционной оптики и программистами, разрабатывающими программы расчета ДОО.

«QUICK-DOE» содержит программы расчета следующих аналитически описанных ДОО:

- различных вариантов дифракционных линз (радиальная, бифокальная, многофокусная, цилиндрическая, скрещенные цилиндрические, с повышенной глубиной фокуса [3]);
- фокусаторов (в кольцо, в крест, в контур квадрата, в отрезок в фокальной плоскости, в прямоугольную область – см. [3]);
- компенсаторов для преобразования сферического волнового фронта в волновой фронт с осесимметричной поверхностью вращения второго порядка (включая параболоид, эллипсоид и гиперболоид – см. [3]);
- оптических фильтров (Карунсна-Лоева базисные функции, фильтр Винера, регуляризирующий фильтр Тихонова – см. [3]);
- ДОЭ, согласованных с модами (моды Гаусса-Эрмита и Гаусса-Лагерра – см. [3]) когерентного излучения (моданов).

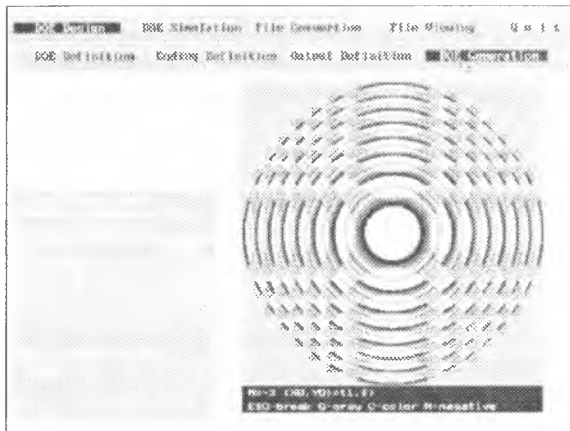


Рис. 24 Экран программного обеспечения «QUICK-DOE».

Список рассчитываемых ДОЭ может быть дополнен пользовательскими ДОЭ. Разработан и реализован несложный метод включения новых оптических элементов. Для введения нового элемента необходимо отредактировать программу проведения диалога для ввода требуемых параметров и написать программу расчета поля (фазы или амплитуды) в точке.

Программное обеспечение содержит различные варианты кодирования фазы и амплитуды:

- кодирование фазы по модулю  $2\pi$ ,
- кодирование из фазы в амплитуду с наложением несущей частоты,
- кодирование из амплитуды в фазу со значениями  $\{0, \pi\}$ ,
- кодирование из амплитуды в фазу с наложением несущей частоты методом Кирка-Джонса [2, 3],
- кодирование методами Ломана и Ли [3],
- кодирование из амплитуды в амплитуду с наложением несущей частоты,
- преобразование амплитуды по абсолютной величине (амплитуда),
- преобразование амплитуды в интенсивность.

Все преобразования могут быть выполнены не только для существующих элементов, но и для элементов, включенных позже пользователем.

При расчете элемента могут быть получены полутоновые или бинарные маски. Маска может быть записана в файл в следующих вариантах:

- каждый отсчет как действительное число,
- байт на отсчет,
- бит на отсчет (только для бинарных масок).

Если нужен набор из нескольких бинарных масок (2 - 4), то можно рассчитать их последовательно или записать полутоновую маску в файл и затем получить все требуемые бинарные маски, используя преобразование. При расчете маски ДОЭ и изменении формата файла маски ДОЭ выполняется синхронная визуализация.

При работе «QUICK-DOE» поддерживается внутренний (растрового типа) формат файлов, записанный построчно с комментариями, содержащимися в заголовке и в окончании файла. Для стыковки с другим программным обеспечением предусмотрены перекодировщики в TIFF-формат байт/отсчет и обратно, и в поле байт/отсчет с построчной организацией. Для облегчения работы пользователя с программным обеспечением имеется удобная оболочка.

#### Программное обеспечение «Iter-DOE».

В случае необходимости, рассчитанные с помощью «QUICK-DOE» фазовые функции оптических элементов могут быть оптимизированы с помощью программного обеспечения, реализующего итерационные методы расчета на основе БПФ. Такое программное обеспечение используется также, когда нельзя аналитически получить и запрограммировать фазовую функцию ДОЭ.

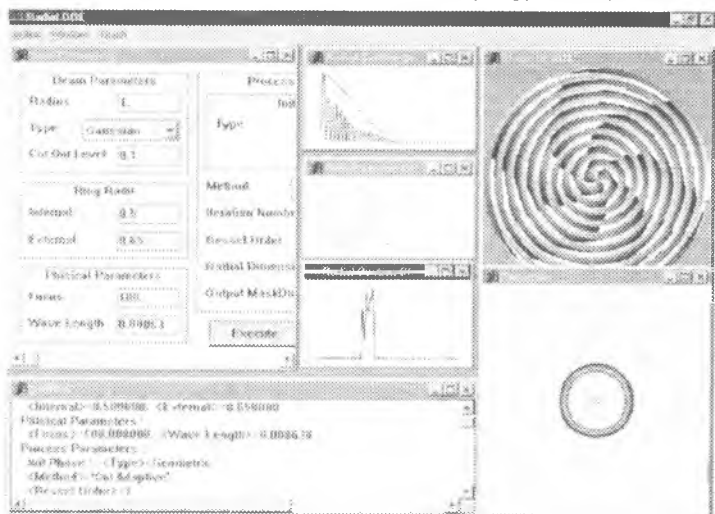


Рис. 25 Экран программного обеспечения «Iter-DOE (на примере расчета фазового ДОЭ, фокусирующего в кольцо).

На рис. 25 представлен экран программного обеспечения «Iter-DOE», предназначенного для расчета фазовых оптических элементов (килоформов, фокусирующих ДОЭ, многофокусных линз, аксионов, компенсаторов) на основе быстрых интегральных преобразований.

Многооконный интерфейс позволяет наглядно представить все изменения фазовой функции и качества работы ДОЭ в процессе итерационного расчета оптического элемента.

**Программное обеспечение «Iter-MODE».** На рис. 26 представлен экран другого программного обеспечения, основанного на итерационных методах расчета ДОЭ. Это - программное обеспечение «Iter-MODE», предназначенное для расчета дифракционных оптических элементов, формирующих моды Гаусса-Лагерра, Гаусса-Эрмита и Бесселя на основе алгоритмов аппроксимации комплексной функции конечным числом базисных функций. Многооконный интерфейс позволяет наглядно представить вид фазовой маски рассчитываемого 9-модового пространственного фильтра и результат моделирования воздействия этого фильтра на освещающий пучок с соответствующим модовым составом.

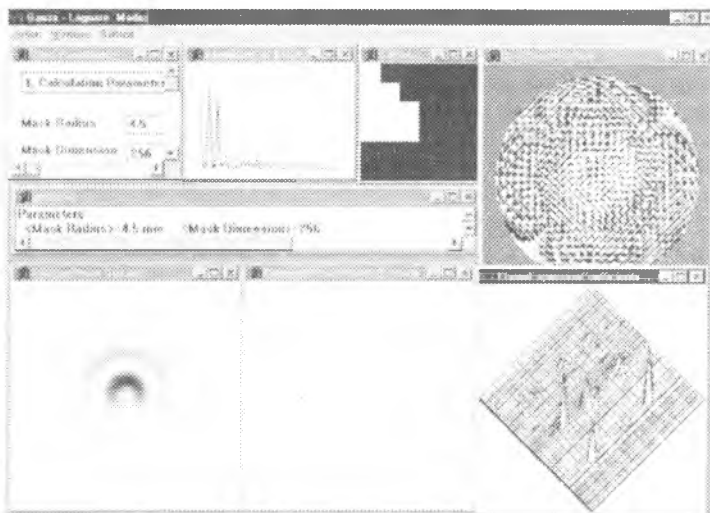


Рис. 26. Экран программного обеспечения «Iter-MODE» (на примере расчета ДОЭ, формирующего многомодовый пучок Гаусса-Лагерра).

**Программное обеспечение «GRATING SOLVER».** Если в процессе итерационного расчета выясняется, что итерационными методами в приближении Френеля-Кирхгофа невозможно добиться решения поставленной задачи, то приходится использовать более точные методы электромагнитного подхода [3]. Эти методы приходится применять также в случаях, когда нарушаются условия приближения Френеля. Программное обеспечение «GRATING SOLVER» (Рис. 27) предназначено для расчета и моделирования дифракционных решеток в рамках электромагнитной теории.

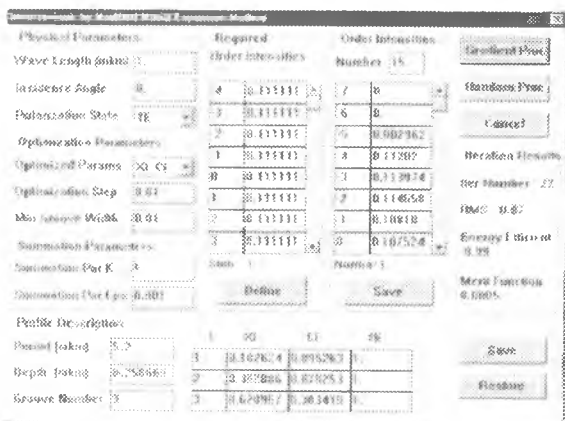


Рис. 27 Экран программного обеспечения «GRATING SOLVER» (на примере расчета 9-порядковой решетки с 3 штрихами).

Это программное обеспечение охватывает:

- моделирование отражающих и пропускающих дифракционных решеток с непрерывным и бинарным профилями рельефа;
- градиентные методы и стохастические алгоритмы решения обратной задачи расчета бинарного профиля из условия формирования заданных порядков дифракции;
- градиентные методы решения обратной задачи расчета непрерывного отражающего профиля в приближении Релея.

На Рис. 27 представлен экран программного обеспечения «GRATING SOLVER» (на примере расчета 9-порядковой решетки с 3 штрихами). Метод расчета основан на оптимизации координат ( $x$ ,  $c$ ,  $h$ ) штрихов профиля из условия минимизации функции невязки расчетных и требуемых интенсивностей порядков.

#### 4.3 Программное обеспечение «SimuLight» для моделирования дифракционных оптических элементов

Программное обеспечение «SimuLight» предназначено для моделирования дифракционных оптических элементов (ДОЭ) и оптических систем, состоящих из нескольких ДОЭ. Программное обеспечение также обеспечивает средства для преобразования файлов форматов микроэлектроники: из векторного многоуровневого формата GDS в набор бинарных GDS файлов. Программное обеспечение предусматривает работу на операционных системах семейства Windows. Рабочий язык интерфейса – английский.

Моделирование работы ДОЭ в приближении скалярной теории дифракции реализовано на основе специальных методов расчета, использующих последние достижения в области численных методов. Программное обеспечение «SimuLight» включает приложение «SimuLight», предназначенное для моделирования оптических систем, а утилита Gds2Mask предназначена для преобразо-



вания файлов из векторного многоуровневого формата GDS в набор бинарных GDS файлов.

#### 4.3.1 Возможности «SimuLight» для моделирования оптических систем

**Цель:** моделирование оптических систем с последующей визуализацией результата моделирования. Оптическая система включает источник когерентного излучения и один или несколько ДОЭ. Предполагается, что источник когерентного излучения и дифракционные оптические элементы расположены на одной оптической оси  $OZ$ . Для моделирования процесса распространения света используется скалярная теория дифракции (интегральное преобразование Кирхгофа). При этом предполагается, что минимальный характеристический размер микрорельефа ДОЭ [3] больше, чем длина волны источника света.

Визуализируются распределения интенсивности и фазы:

- непосредственно от источника света,
  - освещающего пучка в плоскости непосредственно перед ДОЭ,
  - прошедшего через ДОЭ пучка сразу после ДОЭ,
  - в выбранной области наблюдения,
- а также - фазовая функция ДОЭ.

Выбранная область наблюдения определяется как прямоугольник в плоскости, параллельной одной из базовых плоскостей ( $XY$ ,  $ZY$ ,  $ZX$ ). При этом стороны прямоугольника параллельны базовым осям координат ( $OX$ ,  $OY$ ,  $OZ$ ).

#### **Описание:**

Данное приложение реализовано как *MDI*-приложение. Окно задачи состоит из двух частей. Верхняя часть содержит таблицу с описаниями объектов. Нижняя часть содержит изображения описанных в верхней части таблицы объектов. Окно задачи также включает в себя строку состояний (в нижней части окна).

В данном приложении доступны следующие типы объектов:

- когерентный источник света (освещающий пучок - «*Beam*»);
- дифракционный оптический элемент («*DOE*»);
- область визуализации в плоскости («*Rectangle across OZ*»);
- область визуализации в плоскости  $ZX$  («*Rectangle across OY*»);
- область визуализации в плоскости  $ZY$  («*Rectangle across OX*»).

**Замечание:** область визуализации в плоскости  $XY$  имеет общую оптическую ось (ось  $OZ$ ) с источником света и ДОЭ.

**Замечание:** В описываемом программном обеспечении фаза ДОЭ описывается с помощью *htr*-файла. При этом предполагается, что горизонтальный и вертикальный размеры файла в пикселах одинаковы.

Таблица объектов содержит следующие столбцы:

- тип объекта, название столбца «*Type*»;
- плоскость визуализации ( $XY$ ,  $ZX$ ,  $ZY$ ), название столбца «*Plane*»;
- отклонение плоскости наблюдения от точки с координатами  $(0,0,0)$ , название столбца «*Offset*», название координаты ( $x$ ,  $y$  или  $z$ ) соответствует направлению сдвига;
- физические размеры области визуализации, название столбца «*Physical Sizes*», первое значение соответствует оси ординат выбранной плоскости ( $x$  –

для плоскости  $XY$ ,  $z$  – для  $ZX$  или  $ZY$ ), второе значение соответствует оси абсцисс;

- размеры отсчетов в области визуализации;
- другие параметры.

Строка состояний включает актуальную текущую информацию о задаче:

- значение длины волны;
- число обрабатываемых объектов;
- текущий процент обработки объекта;
- обозначение обрабатываемого объекта.

Обработка объектов реализована в фоновом режиме, чтобы предотвратить зависание компьютера. Имеются отдельные окна для каждого объекта. Эти окна содержат визуализируемые изображения. Обозначение объекта приводится в заголовке окна.

#### **Параметры объекта:**

В настоящее время реализованы следующие типы освещающих пучков:

- Гауссов пучок;
- плоский пучок;
- круглый равномерный пучок;
- пучок, интенсивность и фаза которого определяются из файла.

*Замечание: список параметров пучка зависит от выбранного типа пучка.*

*Замечание: задаваемое сечение пучка всегда имеет нулевую координату ( $z = 0$ ).*

#### **Параметры Гауссова пучка:**

- размеры апертуры (в мм), название параметра «Aperture Size»;
- радиус перетяжки Гауссова пучка, название параметра «Gaussian Radius»;

- размер апертуры в пикселях, название параметра «Pixel Number per Side»;

#### **Параметры равномерного пучка:**

- размеры апертуры (в мм), название параметра «Aperture Size»;
- размер апертуры в пикселях, название параметра «Pixel Number per Side»;

Side»;

#### **Параметры равномерного кольцевого пучка:**

- размеры апертуры (в мм), название параметра «Aperture Size»;
- размер апертуры в пикселях, название параметра «Pixel Number per Side»;

Side»;

**Параметры пучка, значения интенсивности и фазы которого задаются из файлов:**

- размеры апертуры (в мм), название параметра «Aperture Size»;
- имя файла, содержащего значения интенсивности;
- имя файла, содержащего значения фазы.

*Замечание: файлы, содержащие значения интенсивности и фазы, должны быть представлены в формате `htr` и иметь одинаковые размеры в пикселях. Предполагается, что вертикальный и горизонтальный размеры апертуры в пикселях одинаковы.*

#### **Параметры ДОЭ:**

- положение вдоль оси  $OZ$  (мм);

- размеры апертуры ДОО (в мм), название параметра «Aperture Size»;
- имя файла, содержащего значения фазовой функции ДОО.

**Параметры области визуализации (прямоугольная область, перпендикулярная оси  $OZ$  - «Rectangle across  $OZ$ ):**

- положение вдоль оси  $OZ$  (мм);
- размеры области (в мм), название параметра «Aperture Size»;
- размеры области визуализации в пикселях, название параметра «Pixel Number per Side»;

**Параметры области визуализации (прямоугольная область, перпендикулярная оси  $OY$  - «Rectangle across  $OY$ ):**

- положение вдоль оси  $OY$  (мм);
- размеры области вдоль оси  $OX$  (в мм),
- количество отсчетов в области визуализации вдоль оси  $OX$ ,
- начальное (минимальное) значение координаты  $z$  (в мм),
- конечное (максимальное) значение координаты  $z$  (в мм),
- количество отсчетов в области визуализации вдоль оси  $OZ$ .

**Параметры области визуализации (прямоугольная область, перпендикулярная оси  $OX$  - «Rectangle across  $OX$ ):**

- положение вдоль оси  $OX$  (мм);
- размеры области вдоль оси  $OY$  (в мм),
- количество отсчетов в области визуализации вдоль оси  $OY$ ,
- начальное (минимальное) значение координаты  $z$  (в мм),
- конечное (максимальное) значение координаты  $z$  (в мм),
- количество отсчетов в области визуализации вдоль оси  $OZ$ .

#### Основные операции, выполняемые приложением:

<i>Название операции</i>	<i>Элемент меню</i>	<i>Клавиатура, действия</i>	<i>Результат</i>
Изменение параметров системы	Action-> Change System Parameters	Ctrl-S	Диалоговое окно для ввода длины волны
Добавление пучка	Action-> Add Beam	Ctrl-B	Диалоговое окно для ввода параметров пучка
Добавление фазы ДОО	Action-> Add Phase DOE	Ctrl-D	Диалоговое окно для ввода параметров ДОО
Добавление прямоугольной области вдоль оси $OZ$	Action-> Add Rectangle across $OZ$	Ctrl-Z	Диалоговое окно для ввода параметров области
Добавление прямоугольной области вдоль оси $OY$	Action-> Add Rectangle across $OY$	Ctrl-Y	Диалоговое окно для ввода параметров области
Добавление прямоугольной области вдоль оси $OX$	Action-> Add Rectangle across $OX$	Ctrl-X	Диалоговое окно для ввода параметров области

Выбор существующего объекта	-	Использование стрелок вверх / вниз; выбор мышкой в таблице; выбор мышкой окна	Описание выбранного объекта помечается; активизируется окно объекта.
Изменение текущего объекта	Action->Modify Current Item	Ctrl-M, Enter	Соответствующее диалоговое окно
Удаление текущего объекта	Action->Remove Current Item	Ctrl-R, Del, Ctrl-Del	Удаление объекта
Конец сеанса	Action-> Exit	Ctrl-E	Окно задачи закрывается без сохранения результата
Выбор изображения для визуализации из отмеченного файла	Pictures->Show*, *={Intensity, Phase, Phase before, DOE Phase, Phase After}	Alt – стрелка влево, Alt – стрелка направо, выбор мышкой в списке окон	Изображение меняется на новое

#### 4.3.2 Утилита Gds2Mask для преобразования из векторного многоуровневого формата GDS в набор бинарных GDS файлов

Утилита Gds2Mask предназначена для преобразования из векторного многоуровневого формата GDS (формат версий 0, 3, 4, 5 или 600) в набор бинарных GDS файлов, соответствующих различным уровням исходного GDS-файла. Утилита реализована в среде компилятора Borland C++. Имеется проверка на корректность исходных данных.

Формат командной строки:

**Gds2Mask.exe <source GDS-file name> <wavelength> <refractive index>**

**<source GDS-file name>** - имя исходного многоуровневого gds-файла.

Имя файла должно быть введено с расширением .GDS или без него;

**<wavelength>** - длина волны освещающего пучка для ДОО (в микронах);

**<refractive index>** - показатель преломления подложки ДОО.

Исходный GDS-файл может содержать следующие GDS-коды:

**HEADER;**  
**ENDLIB;**  
**ENDSTR;**  
**ENDEL;**  
**BOUNDARY;**  
**PATH;**  
**SREF;**  
**AREF;**  
**TEXT;**  
**LAYER;**

**DATATYPE;**  
**PATHTYPE;**  
**GENERATIONS;**  
**UNITS;**  
**LIBNAME;**  
**XY;**  
**BGNLIB;**  
**STRNAME;**  
**SNAME.**

Выходные файлы: DEPTH.TXT, MASK0.GDS, MASK1.GDS, ..., MASKN.GDS, где MASKM.GDS – бинарный файл формата GDS, соответствующий M-тому слою исходного многоуровневого файла GDS-формата.

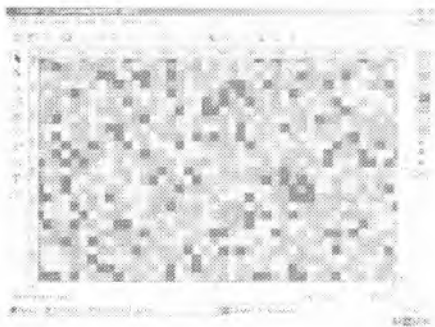
DEPTH.TXT – файл типа «txt», содержащий значения глубин травления для каждого уровня микрорельефа.

### **Пример:**

Многоуровневый файл relief.gds, содержащий рассчитанные значения микрорельефа ДООЭ, спроектированного для работы с гелий-неоновым (He-Ne) лазером (длина волны 0,633 мкм), необходимо преобразовать в набор бинарных файлов gds-формата, описывающих отдельные уровни исходного файла. Показатель преломления материала – 1,5. Для запуска преобразования необходимо ввести следующую командную строку:

```
C:\UTILITY\ gds2mask 0.633 1.5
```

В результате мы получим сгенерированные файлы DEPTH.TXT, MASK0.GDS, MASK1.GDS, MASK2.GDS, MASK3.GDS, MASK4.GDS, MASK5.GDS, MASK6.GDS, MASK7.GDS в той же самой директории. На рис. 27 показан экран программного обеспечения CleWin во время визуализации исходного GDS-файла. На рис. 28 показан экран программного обеспечения CleWin во время визуализации бинарного GDS-файла MASK1.GDS.



*Рис. 27 Экран специализированного программного обеспечения во время визуализации исходного многоуровневого GDS-файла*

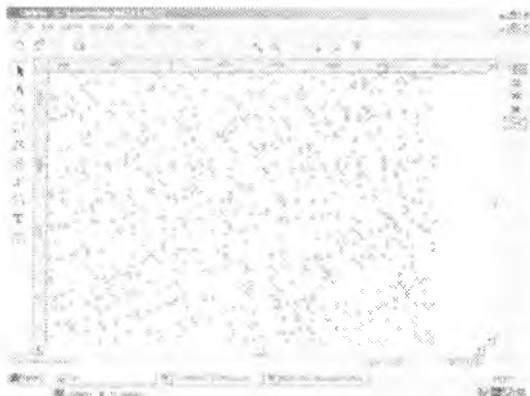


Рис. 28 Экран специализированного программного обеспечения во время визуализации бинарного GDS-файла MASK1.GDS

Файл типа «txt» Depth.txt, сгенерированный в той же директории, содержит следующие значения:

Etching Depths for Masks (глубина травления для масок):

```
mask (маска) N 0 (глубина) depth=0.000000
mask (маска) N 1 (глубина) depth=0.157500
mask (маска) N 2 (глубина) depth=0.315000
mask (маска) N 3 (глубина) depth=0.472500
mask (маска) N 4 (глубина) depth=0.630000
mask (маска) N 5 (глубина) depth=0.787500
mask (маска) N 6 (глубина) depth=0.945000
mask (маска) N 7 (глубина) depth=1.102500
```

#### 4.4 Программное обеспечение «TracePro»

##### 4.4.1 Методы и алгоритмы, используемые «TracePro»

В данном разделе (4.4) будет рассмотрен пример программного обеспечения, предназначенного для моделирования оптических процессов - «TracePro». Акцент сделан на анализе возможностей, наиболее востребованных в инженерной практике.

Кратко рассмотрим историю развития программ, связанных с моделированием светотехнических устройств. Первоначальная цель такого моделирования состояла в расчете освещенности сцен. При этом главным критерием была визуальная достоверность полученной сцены. Реальные оптические процессы тогда мало интересовали разработчиков. Одной из первых стала локальная модель освещения, в которой учитывались эффекты зеркального и — на примитивном уровне — диффузного отражения. Кроме того, вводилось так называемое фоновое освещение, учитывающее наличие излучения, порождаемого «неформализуемыми» источниками. Следующим шагом было появление модели

локального затенения, одной из особенностей которого является интерполяция освещенности по поверхности объектов, а также наличие ряда моделей характеристик поверхности.

Важным этапом стало появления метода «Ray tracing» (Трассировка лучей), который заключается в отслеживании траекторий лучей и моделировании того, как каждый луч взаимодействует со встречающимися на его пути телами и поверхностями. Этот подход позволяет ввести в рассмотрение эффекты преломления, затенения, многократного зеркального отражения. Исторически первым был метод Forward ray tracing (Прямая трассировка), когда лучи, как это и происходит в реальности, испускаются источником света, а затем последовательно взаимодействуют с объектами сцены. Именно этот алгоритм стал базой для развития светотехнических программ, и именно тогда произошло их выделение в самостоятельную ветвь программного обеспечения. Программные продукты для «эстетической» визуализации (рендеринга) стали развиваться отдельно. Взаимодействие между этими направлениями происходило практически только на уровне идей, но предназначение программ и, что немаловажно, клиентская база, были различны.

Круг алгоритмов, которые используются в обеих группах программ, включает в себя метод «Distribute raytracing» (Распределенная трассировка лучей) или, в терминах «TracePro» — «Ray splitting» (Расщепление луча). Смысл его в том, что при взаимодействии с поверхностью или средой луч может расщепляться на несколько, причем отдельные лучи несут зеркально отраженную, рассеянную, преломленную и т.д. компоненты светового потока. Это является базой для корректного описания эффектов диффузного отражения и рассеяния в материале.

Еще одной общей идеей стал алгоритм «Backward ray tracing» (Обратная трассировка лучей) или, как он называется в «TracePro» — «Reverse raytracing». Он базируется на том, что лучи испускаются объектами, освещенность которых интересует наблюдателя, после чего прослеживаются их пути к источнику. В «TracePro» данный алгоритм появился только в 2004 году.

Число трассируемых лучей является ключевым параметром, влияющим на точность анализа. В методе прямой трассировки лучи испускаются источниками света, которые в рассматриваемом программном обеспечении могут быть поверхностными, или же назначаться на сетке. В первом случае, независимо от того, какой тип распределения лучей в пространстве выбирается пользователем, координаты точек, из которых испускается луч, выбираются программой случайным образом. Если используется пространственное распределение, которое не является равномерным, то направления также случайны (хотя, разумеется, подчиняются некоторой функции распределения). Этими процессами управляет метод Монте-Карло, функция которого в данной ситуации генерировать точки старта и направления. Если же алгоритм трассировки имитирует рассеяние света при взаимодействии с поверхностью или в объеме, то здесь также используется генератор случайных чисел. В первой ситуации, в зависимости от соотношения между долями отраженного, прошедшего и рассеянного света формируется луч, который, может имитировать как отраженный, преломленный, так и рассеянный свет. В последнем случае направление луча назначается программой исходя из заданного закона распределения рассеянной энергии, сформулированного с позиции теории вероятности. При описании объемного рассеяния используемые модели подразуме-

вают изменение направления луча (направление выбирается статистически исходя из назначенного закона распределения) при прохождении им некоторого расстояния. Здесь также может имитироваться расщепление: в зависимости от соотношения между долями отраженного, прошедшего и рассеянного света формируется луч, который, может имитировать как отраженный, преломленный, так и рассеянный свет.

В качестве источников света может использоваться совокупность лучей, исходящих из вершин некоторой сетки. Она может быть равномерной (прямоугольной или круговой), а может быть назначена статистически или с учетом статистического отклонения от «правильной» формы. Возникает вопрос: для чего введены эти усложнения. Во-первых, взаимодействие света со средой имеет статистическую природу, поскольку оптические характеристики среды также имеют вероятностный характер: параметры шероховатости поверхности, а также отклонения ее формы, оптическая неоднородность прозрачной среды, например, локальная анизотропия пластмасс, наличие включений и т.д. Реальная атмосфера в силу различия температуры, запыленности, движения потоков с отличающейся плотностью, не является оптически-однородной. Поэтому идеализацию оптических характеристик объектов необходимо компенсировать вводом статистической компоненты в законы излучения и распространения света. Кроме того, вывод изображения на экран монитора, представляющий собой совокупность пикселей, также порождает «Aliasing» (Алиасинг) — зубчатый характер контуров, а также сопровождается появлением повторяющихся структур на однородном, или плавно изменяющемся в действительности изображении.

Алгоритм рекурсивной вокселизации, предназначенный для ускоренной идентификации объектов, встречающихся на пути луча, в «TracePro» был введен в 2003 году. В то же время простая вокселизация — разбиение пространства на виртуальные параллелепипеды с установлением связи между объектами сцены и параллелепипедами, в которые они могут входить, в этой программе существовал изначально.

Алгоритм, который не используется в программах светотехники, но является вполне традиционным для процедур рендеринга — метод «Radiosity» (Метод излучательности), предполагающий разбиение поверхностей на участки с последующим анализом взаимодействия участков, описывающих все объекты сцены, между собой. Многие идеи данного алгоритма перекликаются с методом конечных элементов (МКЭ). Надо сказать, что теоретическая база для моделирования теплопереноса излучением, использованная в МКЭ, имеет много общего с данным алгоритмом. Последние реализации программ рендеринга содержат опции, позволяющие, в какой-то степени, объединять два алгоритма: радиосити и трассировки. Эти гибридные алгоритмы носят название «Photon Maps» (Метод фотонных карт).

Для увеличения визуального правдоподобия в алгоритмах рендеринга, основанных на трассировке лучей, в дополнение к используемым BRDF и BTDF — функциям, описывающим процесс рассеяния при отражении и прохождении через поверхность, разработан метод BSSRDF («Bidirectional Surface Scattering Reflection Distribution Function» — Двухнаправленная функция распределения отражения рассеянного света). Эта модель описывает поведение луча, попавшего на поверхность тела, обладающего объемным рассеянием.



При этом часть светового потока, в силу объемного рассеяния, возвращается обратно к поверхности, а затем — в среду, из которой этот луч пришел. То есть, по сути, объединяются эффекты, которые в «TracePro» обрабатываются принципиально различными инструментами. Для имитации сцены, когда преследуется цель повысить визуальное правдоподобие, объединение эффектов объемного и поверхностного рассеяния более результативно, по крайней мере, с точки зрения производительности вычислений (возможность прохождения света через тело с такой поверхностью отсутствует в принципе).

Как видно из краткого обзора, имеется достаточно число точек соприкосновения двух направлений компьютерного моделирования при визуализации. Собственно, последние релизы «TracePro», а также других конкурирующих продуктов, включают в себя модули рендеринга (они поставляются опционно). Суть соответствующего приложения «TracePro» в том, что в качестве исходной информации оно использует результаты расчета базовой программы, после чего изображение формируется с учетом условностей, которые порождают ощущение визуального правдоподобия.

Самое важное отличие программ светотехнического анализа от программ визуализации: «TracePro» и родственные продукты — полноценные члены семейства CAD/CAE. В качестве геометрического ядра они имеют одну из известных систем: ACIS («TracePro»), Parasolid, математику Catia V5 и т.д. Соответственно доступны (иногда за дополнительную плату) интерфейсы через форматы STEP и IGES, в то время как организация взаимодействия программ "эстетического" рендеринга и CAD-систем весьма щекотливый вопрос.

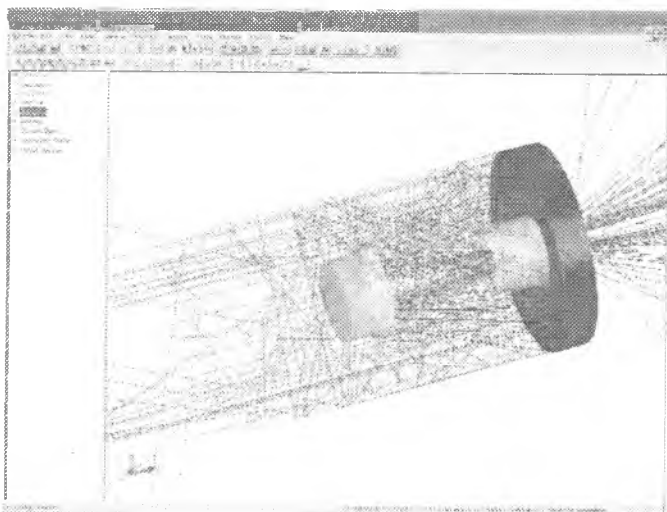
Упомянем еще два типа программ, оперирующих с оптическими явлениями. Первые — программы детерминированной оптики, то есть не предполагающие статистического характера распределения света. Эти продукты предназначены для проектирования оптических приборов: объективов, телескопов и т.д. Примером данной программы является «OSLO», которая, так же, как и «TracePro» разрабатывается фирмой «Lambda Research». Вторая группа программ — инструменты проектирования, но, на сей раз, в светотехнике. Одна из них — «ReflectorCAD».

#### *4.4.2 Краткое изложение возможностей «TracePro»*

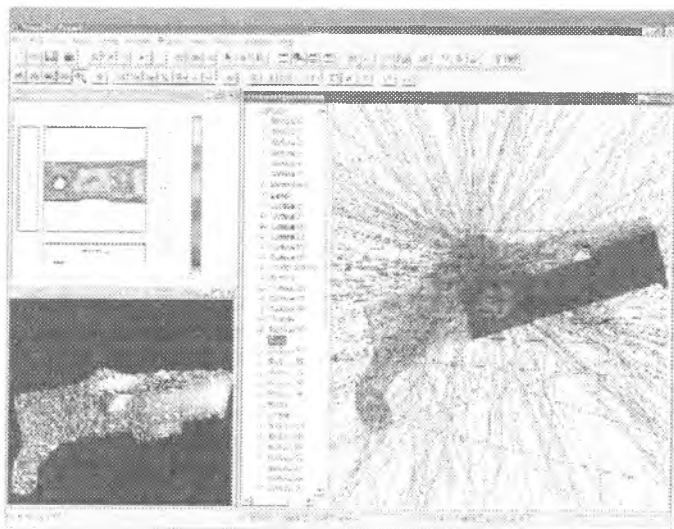
«TracePro» представляет собой уникальный по функциональности инструмент, который можно использовать в широком спектре приложений и характеризуется разработчиком как «интегрированная система оптического анализа и твердотельного моделирования» (Рис. 29 и 30).

«TracePro» имеет следующие возможности для анализа освещения, расчета осветительных приборов и источников света:

- программа имитирует работу ламп накаливания, светодиодов, флуоресцентных ламп, разнообразных рефлекторов и линзовых систем;
- каталог ламп содержит более 200 стандартных объектов, включая каталоги «OSRAM» и «Phillips»;
- «TracePro» обеспечивает импорт источников, представленных распределением интенсивности света;
- возможен экспорт результатов в стандартах «IES» и «Eulumdat».



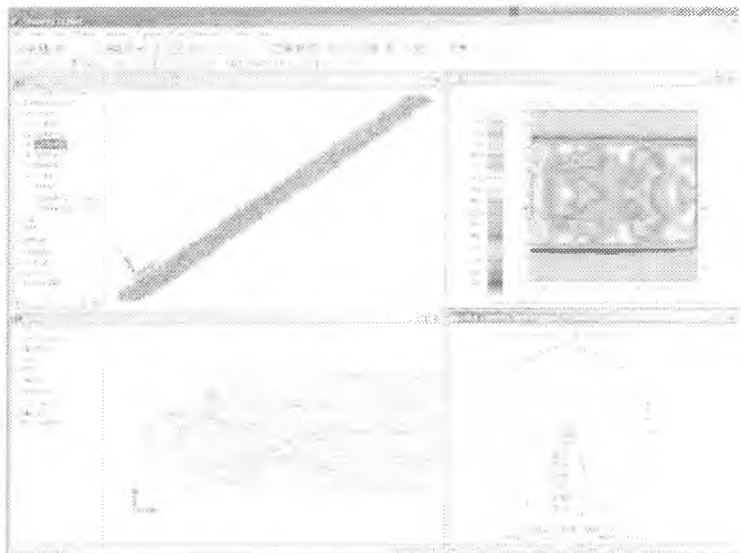
*Рис. 29 «TracePro» с успехом решает задачи трассировки лучей в произвольной оптической системе. Анализ распространения света с учетом отражения, рассеяния и поглощения выполняется с максимальной наглядностью*



*Рис. 30 Виртуальные прототипы световодов могут быть рассчитаны с получением фактического распределения света, плоского и пространственного отображения освещенности и яркости, а также качественного рендеринга сцены*

**«ТгасеПро»** предоставляет следующие возможности для моделирования дисплейных систем, включая жидкокристаллические мониторы (LCD):

- программа позволяет рассчитывать дисплеи с задней и передней подсветкой, например, дисплеи компьютеров и мобильных телефонов;
- имеются интегрированные в программу средства для моделирования работы LCD систем и световодов;
- алгоритм «RepTile™» является уникальным средством моделирования устройств, содержащих миллионы повторяющихся объектов, с учетом яркости и рассеяния (см. Рис. 31);



*Рис. 31 Алгоритм «RepTile» применительно к задним фонарям даёт возможность полноценного анализа сложной конструкции, содержащей LED панель. Приведённая модель содержит семь миллионов рассеивающих точек. Для её расчета потребовалось не более десяти минут*

- моделирование поляризации с учетом векторов Стокса и матриц Мюллера в зависимости от угла падения, длины волны и температуры;
- используя «Bitmap», можно создать источник с полной информацией о распределении энергии по длинам волн - этот источник можно использовать в расчетах для анализа искажений и качества цветопередачи;
- диаграммы яркости и освещенности обеспечивают наглядное отображение результатов.

**Возможности «ТгасеПро» для систем машинного зрения:**

- посредством «ТгасеПро» можно создавать изделия, предназначенные для дефектоскопии, робототехники, сканирования и т.д.;
- полноценное моделирование эффектов рассеяния, с имитацией анизотропии свойств, а также асимметричной BSDF.

## **Возможности и средства «TracePro» для решения задач автомобилестроения:**

- анализ паразитной засветки, в частности, на дисплеях панелей приборов и на лобовых стеклах, используя алгоритм обратной трассировки;
- обширный каталог автомобильных ламп, а также специальные возможности для создания фасетных отражателей;
- инструменты для генерации массивов линзовых объектов в совокупности с фасетными отражателями, используемые применительно к задним фонарям;
- интегрированные IGES и STEP трансляторы для обмена с универсальными CAD-системами;
- инструменты для моделирования линз Френеля;
- инструменты для создания моделей практически любых типов источников, а также импорта источников из каталогов фирмы «Lambda Research» или библиотек сторонних производителей.

## **Возможности и средства «TracePro» для решения задач аэрокосмической промышленности и оборонной техники:**

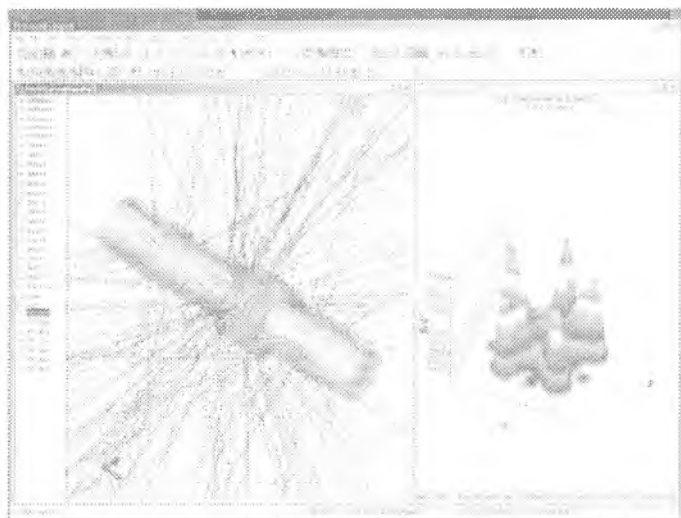
- эффективный анализ CHMSL систем, которые содержат массивы элементов, состоящих из линз Френеля и жидкокристаллических источников, посредством алгоритма RepTile;
- использование выборки по значимости для оптимальной трассировки рассеянного света по направлению к заданным датчикам;
- наличие моделей асимметричного рассеяния, соответствующего оптически-анизотропным поверхностям; учет зависимости параметров рассеяния от направления падения, длины волны и температуры;
- моделирование расщепления лучей при взаимодействии с поверхностью на пять составляющих: пропущенную, поглощенную, отраженную, пропущенную и отраженную с рассеянием;
- эффекты дифракции на апертурах и в коронографах, учет краевых эффектов на экранах;
- получение результатов для произвольных объектов: тел и поверхностей с учетом распределения по длинам волн и с анализом долей попавшей, поглощенной и потерянной энергии;
- доступ к траектории произвольного луча, включая длину траектории, величину переносимого светового потока, информацию о поляризации; выделение совокупности лучей, попавших на поверхность;
- получение информации о распределении энергии в потоке с произвольным спектром излучения.

### **«TracePro» - для медицинской техники:**

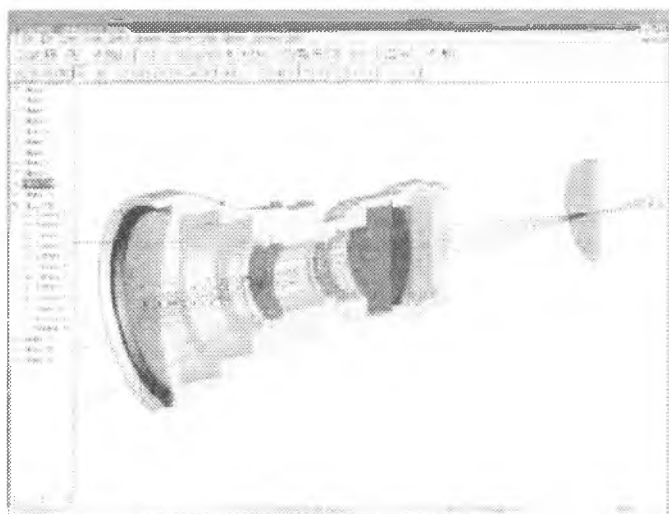
- моделирование взаимодействия излучения с кожей;
- наличие библиотеки характеристик компонентов биологических тканей.

### **Лазеры:**

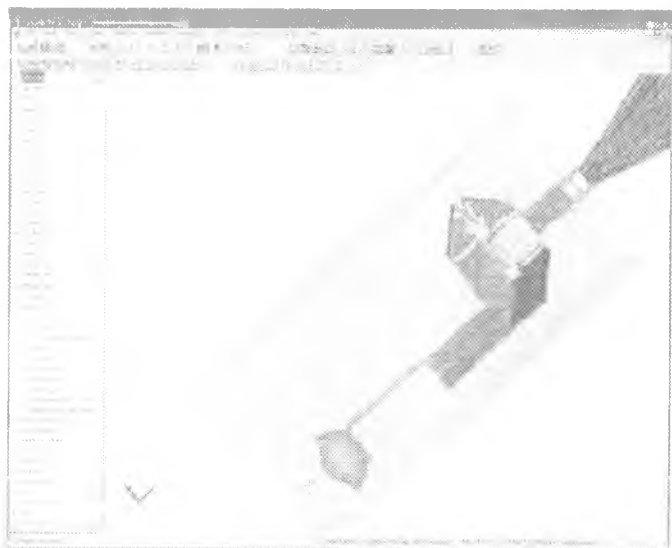
- трассировка потока в резонаторах и отображение этого на экране;
- экспорт информации о распределении потока в специализированные приложения для расчёта лазеров (см. рис. 32);
- идентификация рассеянной компоненты энергии, включая процессы в кристаллах АИГ, в частицах воды и в лазерных диодах;
- назначение множественных источников, включая анизотропно-излучающие поверхности (см. рис. 33-38);
- учет поляризации посредством векторов Стокса.



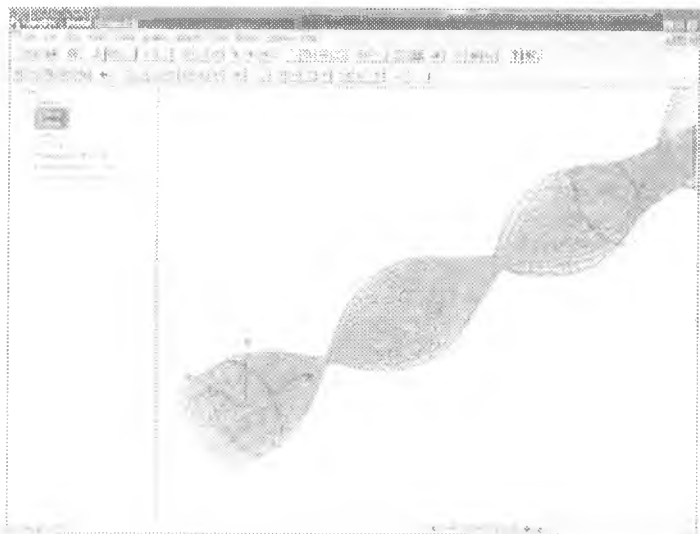
*Рис. 32 Анализ прохождения светового потока в объёме тела. Программа позволяет рассчитывать объёмное распределение потока энергии в лазерных резонаторах, выделять величину попавшей, отраженной и поглощенной энергии*



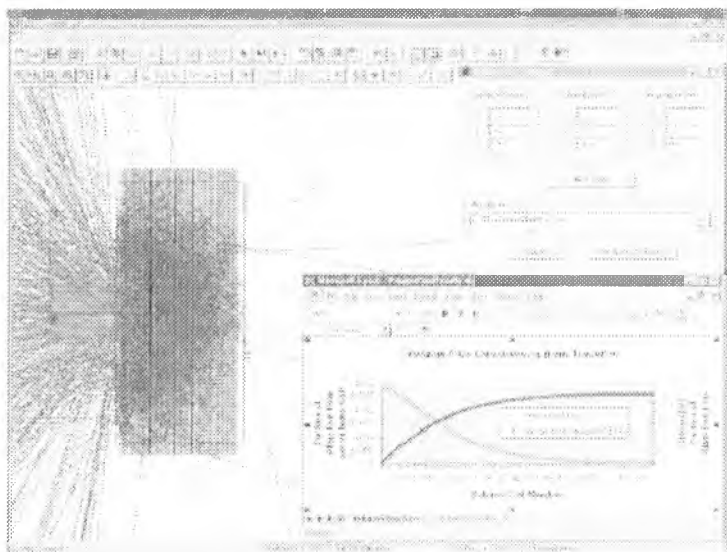
*Рис. 33 После импорта твердотельной и поверхностной геометрии из CAD систем соответствующим объектам могут быть назначены оптические характеристики и выполнен расчет*



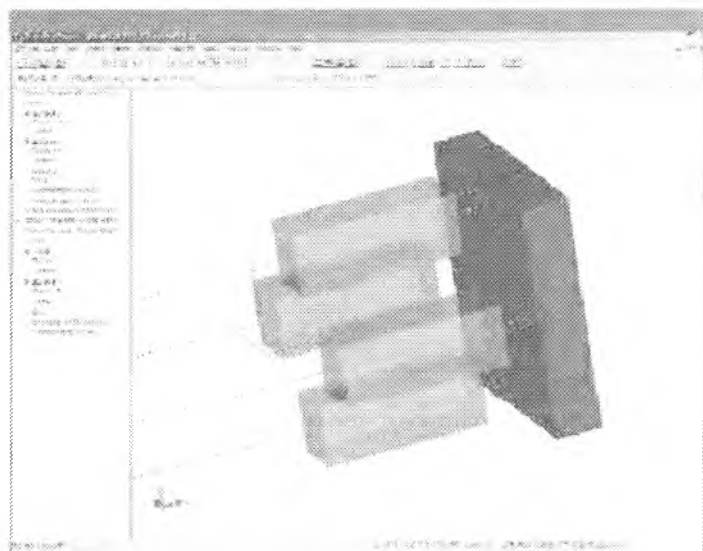
*Рис. 34 Результаты трассировки могут выводиться на фоне прозрачной модели с отображением лучей линиями различного цвета. Эти цвета могут соответствовать как величине потерянной энергии, так и длине световой волн*



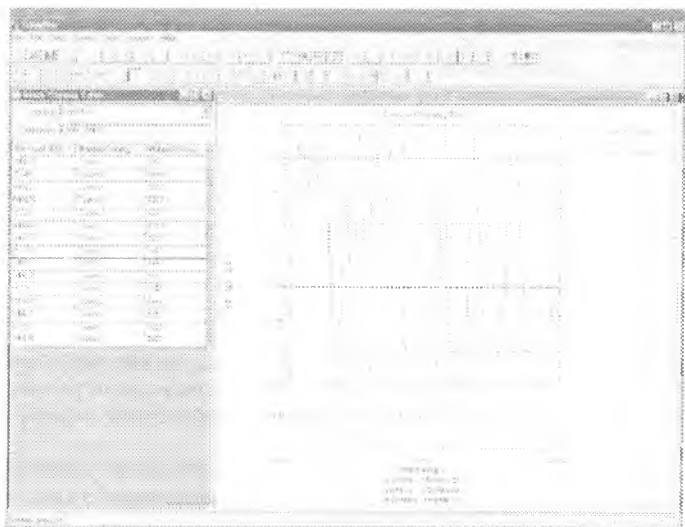
*Рис. 35 Пример расчета волокна, имеющего переменный коэффициент преломления. Такие характеристики содержатся в каталоге программы или могут быть вставлены на основе информации пользователя*



*Рис. 36* Возможности программы по моделированию эффектов объемного рассеяния демонстрируются на примере взаимодействия кожи со светом. Программа содержит разнообразные модели объемного рассеяния



*Рис. 37* Моделирование двоякопреломляющих кристаллов является новой возможностью программы



*Рис. 38 Специальные инструменты предназначены для визуализации параметров тонкопленочных объектов. На иллюстрации показаны оптические характеристики 15-слойного покрытия в зависимости от длины волны*

#### **Заключение к главе 4**

В данной главе дан краткий обзор существующих программных продуктов, предназначенных для моделирования оптических систем. Описано доступное студентам СГАУ (имеющееся в распоряжении факультета информатики СГАУ и, в частности, базовой кафедры высокопроизводительных вычислений ИСОИ РАН) программное обеспечение по компьютерной оптике и программное обеспечение «TracePro», разрабатываемое фирмой «Lambda Research Corporation» (находится в одном из центров мировой оптической индустрии - городе Tucson, США). Описанные продукты дают достаточное представление о современных тенденциях в создании оптического программного обеспечения.



## **Выводы**

1. В настоящее время на рынке имеется большой выбор программных продуктов, предназначенных для моделирования и проектирования оптических систем.

2. Стоимость больших программных комплексов, включающих разнообразные методы расчета и гигантские базы данных с информацией о выпускаемых во всем мире источниках света и оптических элементах, достигает 40 тысяч долларов США (ASAP). Несмотря на такую высокую стоимость время работы программного обеспечения (срок действия лицензии) составляет один год, а стоимость продления лицензии на следующий год (вместе с техническим обслуживанием) доходит до 10 тысяч долларов США.

3. На факультете информатики СГАУ и базовой кафедре высокопроизводительных вычислений ИСОИ РАН имеется ряд дополняющих друг друга программных комплексов по компьютерной оптике, позволяющих решать широкий круг задач расчета, оптимизации и моделирования работы дифракционных оптических элементов.

4. На факультете информатики СГАУ имеется сетевая учебная версия программного обеспечения «ТрасеПро» - мощного программного продукта, обладающего широким спектром возможностей для моделирования и проектирования разнообразных оптических систем (освещающих устройств, дисплеев, систем машинного зрения, светотехнических устройств для автомобилестроения, для медицинских приборов, лазерной техники, решения задач аэрокосмической промышленности и оборонной техники).

## Приложение 1. Обзор необходимых команд Linux.

Ниже приводятся некоторые наиболее употребляемые команды Linux. Большинство этих команд можно выполнить на управляющем узле с помощью MidnightCommander. Однако на вычислительных узлах MidnightCommander, как правило, отсутствует.

Чтобы получить более полную информацию по любой отдельной команде `command`, нужно ввести

```
man command
```

Выход из описания команды производится при нажатии клавиши «q».

### Работа с каталогами

`pwd` – показывает название текущей директории;

`cd dir` – устанавливает текущим каталогом каталог с именем `dir`, вызов команды `cd` без параметров возвращает в домашний каталог `/home/username ($HOME)`;

`mkdir subdir` – создает новый подкаталог с именем `subdir`;

`rmdir subdir` – удаляет пустой подкаталог с именем `subdir`;

`ls` – показывает список файлов и подкаталогов текущей директории,

`ls dir` – показывает список файлов и подкаталогов каталога `dir`;

`ls -A` - показывает все файлы, в том числе и скрытые;

`ls -l` - показывает атрибуты (владелец, разрешение на доступ, размер файла и время последней модификации);

`mv oldname newname` - изменяет имя подкаталога или перемещает его;

`cp -R dirname destination` - копирует подкаталог `dirname` в другое место `destination`.

### Работа с файлами

`file filename(s)` - определяет тип файла (например, ASCII , JPEG image data и др.);

`cat filename(s)` - показывает содержание файлов (используется только для текстовых файлов!);

**more filename (s)** - действует так же, как и **cat**, но позволяет листать страницы;

**less filename (s)** - улучшенный вариант команды **more**;

**head filename** - показывает первые десять строк файла **filename**;

**tail filename** - показывает последние десять строк файла **filename**;

**wc filename (s)** - показывает число строк, слов и байт для указанного файла;

**rm filename (s)** - уничтожает файлы или директории, для рекурсивного удаления следует использовать **rm** с ключом **-rf**.

**cp filename newname** - создает копии файлов с новыми именами;

**cp filename (s) dir** - копирует один или более файлов в другой каталог;

**mv oldname newname** - изменяет имя файла или каталога;

**mv filename (s) dir** - перемещает один или более файлов в другой каталог;

**find dir -name filename** - пытается локализовать файл (подкаталог) **filename** рекурсивно в подкаталоге **dir**.

### Другие полезные команды

**passwd** - изменяет пароль пользователя системы Linux; требует подтверждения старого;

**who** - показывает, кто в настоящее время работает в сети;

**finger** - дает более подробную информацию о пользователях сети;

**write** - позволяет послать сообщение пользователю, работающему в сети в данное время;

**top** - отображает информацию о процессах, использующих процессоры узла;

**ps -U user\_name** - показывает номера процессов(pid), инициированных пользователем **user\_name**;

**kill xxxxx** - досрочно завершает работы процесса с номером **xxxxx**;

**killall proc\_name** - досрочно завершает работу процесса **proc\_name**;

**date** - отображает дату и время;

**cal** - показывает календарь.

**exit** - выйти из терминала

**clear** - очистить окно терминала

**du dir** - показывает занятое место в директории **dir**

## Приложение 2. Примеры PBS скриптов

Подробно о возможностях PBS Torque можно узнать из руководства пользователя.

Ниже приведены несколько примеров использования Torque.

```
#PBS -o $DIR/stdout.log
```

Определяет имя файла, в который будет перенаправлен стандартный поток stdout

```
#PBS -e $DIR/stderr.log
```

Определяет имя файла, в который будет перенаправлен стандартный поток stderr

```
#PBS -l nodes=8:ppn=2:cpp=1
```

Определяет какое количество узлов и процессоров на них необходимо задействовать.

nodes - количество узлов

ppn - число процессоров на узле

cpp - число процессов на процессоре

```
#PBS -l walltime=20:00:00
```

Определяет максимальное время счета задания

```
#PBS -l mem=1000mb
```

Определяет количество необходимой оперативной памяти

```
cat $PBS_NODEFILE | grep -v master | sort | uniq -c | awk  
'{printf "%s:%s\n", $2, $1}' >
```

```
$PBS_O_WORKDIR/temp.tmp
```

Составляет список узлов в необходимом формате, на которых будет запущена задача и записывает их в файл temp.tmp

```
cd $PBS_O_WORKDIR
```

```
/usr/bin/mpirun -m temp.tmp -np 100 ./a.out
```

Запускает на узлах указанных в файле temp.tmp задачу 100 раз.

Пример скрипта:

```
#PBS -o $DIR/stdout.log
```

```
#PBS -e $DIR/stderr.log
```

```
#PBS -l nodes=50:ppn=2
```

```
#PBS -l walltime=20:00:00
```

```
#PBS -l mem=1000mb
```

```
cat $PBS_NODEFILE | grep -v master | sort | uniq -c | awk  
'{printf "%s:%s\n", $2, $1}' >
```

```
$PBS_O_WORKDIR/script1.temp.sh.mf
```

```
cd $PBS_O_WORKDIR
/usr/bin/mpirun -m script1.temp.sh.mf -np 100 ./a.out
```

Здесь будет запущена параллельная программа a.out на 50 узлах, с каждого узла будет использоваться 2 процессора. Файл вывода стандартного потока stdout — stdout.log, стандартного потока stderr — stderr.log.

\$DIR содержит путь к файлам stdout.log и stderr.log, например может принимать значение /home/user\_name. Под задачу отведено 20 часов. Необходимое количество памяти 1000 мегабайт.

Для запуска последовательной программы first можно использовать следующий скрипт:

```
#PBS -o $DIR/stdout.log
#PBS -e $DIR/stderr.log
#PBS -l walltime=10:00
#PBS -l mem=100mb
./first
```

При запуске программы через команду qsub заданию присваивается уникальный целочисленный идентификатор.

**qdel** – утилита для удаления задачи.

В случае, если задача уже запущена, процесс ее работы будет прерван. Синтаксис данной утилиты следующий:

**qdel [-W время задержки]идентификатор задачи**

Выполнение такой команды удалит задачи с заданными идентификаторами через указанное время. Если часть вычислительных узлов, на которых выполнялась задача, недоступны, то принудительно удалить ее с сервера можно путем добавления ключа -p.

### Приложение 3. Переменные окружения планировщика Torque

Далее приводится список переменных окружения Torque и примеры их значений.

```
PBS_JOBNAME=env
PBS_ENVIRONMENT=PBS_BATCH
PBS_O_WORKDIR=/home/test
PBS_TASKNUM=1
PBS_O_HOME=/home/test
PBS_MOMPOROT=15003
PBS_O_QUEUE=batch
PBS_O_LOGNAME=test
PBS_O_LANG=en_US.UTF-8
PBS_JOBCOOKIE=3088939E7FAA7F4414578D7A806955
PBS_NODENUM=0
PBS_O_SHELL=/bin/bash
PBS_JOBID=93.master.localdomain
PBS_O_HOST=master.localdomain
PBS_VNODENUM=0
PBS_QUEUE=batch
PBS_O_MAIL=/var/spool/mail/test
PBS_O_PATH=/home/test/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:
/usr/games:/opt/gnome/bin:/opt/kde3/bin:
/usr/lib/mit/bin:/usr/lib/mit/sbin
```

## Список литературы

1. Борн М., Вольф Э. Основы оптики. - М.: Наука, 1973. - 720 с.
2. Ландау Л.Д., Лифшиц Е.М. Теория поля, изд. 5-е. М.: Наука, Физматгиз, 1967.
3. Соيفер В.А. Введение в дифракционную микрооптику. - Самара: СГАУ, 1996. - 95с.
4. Методы компьютерной оптики / Под редакцией В.А. Соифера. - М.: Физматлит, 2003. - 688с.
5. [http://jdj.mit.edu/wiki/index.php/Meep\\_Reference](http://jdj.mit.edu/wiki/index.php/Meep_Reference)
6. A. Taflove, S.C. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, 3rd ed., Artech, Norwood, MA, 2005.
7. K.S. Kunz, R.J. Luebbers, The Finite-Difference Time-Domain Method for Electromagnetics, CRC Press, Boca Raton, 1993.
8. D.M. Sullivan, Electromagnetic Simulation Using the FDTD Method, Wiley-IEEE Press, New York, 2000.
9. [http://ab-initio.mit.edu/wiki/index.php/Guile\\_and\\_Scheme\\_links](http://ab-initio.mit.edu/wiki/index.php/Guile_and_Scheme_links)
10. [http://ab-initio.mit.edu/wiki/index.php/Libctl\\_manual](http://ab-initio.mit.edu/wiki/index.php/Libctl_manual)
11. Jenkins, F.A. Fundamentals of Optics, 4th edition / F.A. Jenkins, H.E. White - NY: McGraw-Hill Book Company, 1976. - Chapter 25.
12. Неймарк Ю.И. Математические модели в естествознании и технике. - Н.Новгород: ННГУ, 2004. - 401с.
13. Самарский А.А. Математическое моделирование и вычислительный эксперимент // Вестник АН СССР. - 1979. - № 5. - С.38-41.
14. Самарский А.А. Математическое моделирование и вычислительный эксперимент // Сборник лекций «Современные методы математического моделирования» по материалам Международной конференции «Математическое моделирование – 2001», Самара: СГАУ, 2001, с.4-12.
15. Математическое моделирование / Под редакцией Дж. Эндрюса и Р. Мак-Лоуна. - М.: Мир, 1979. - 277с.
16. Родионов С.А. Автоматизация проектирования оптических систем. - Л.: Машиностроение, 1982.
17. Алямовский А.А. Компьютерное моделирование в инженерной практике. - М.: ЗАО «Сикор», 2004. - 186 с.
18. Гудмен Дж. Введение в Фурье-оптику. - М.: Мир, 1970.
19. Применение методов Фурье-оптики / Под ред. Г. Старка: М.: Радио и связь, 1988.
20. Сороко Л.М. Основы голографии и когерентной оптики. - М.: Наука, 1971.



Учебное издание

*Казанский Николай Львович  
Серафимович Павел Григорьевич  
Хонина Светлана Николаевна*

**ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ  
В ДИФРАКЦИОННОЙ НАНООПТИКЕ**

*Учебное пособие*

Компьютерная верстка Я.Е. Тахтарова

Подписано в печать 15.12.2010 г. Формат 60×84 1/16.

Бумага офсетная. Печать офсетная.

Печ. л. 6,93. Тираж 100 экз. Заказ 112

Учреждение Российской академии наук Институт систем обработки изображений РАН,  
Самарский государственный аэрокосмический университет имени академика С.П. Королёва  
(национальный исследовательский университет)

---

Издательство «ВЕК#21»  
443099 Самара, Чапаевская 69 а.