МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

С. Н. Хонина, П. Г. Серафимович

Использование параллельных вычислений для решения задач нанофотоники

Электронное учебно-методическое пособие

САМАРА
2010

Авторы: ХОНИНА Светлана Николаевна,
СЕРАФИМОВИЧ Павел Григорьевич

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.

Рассмотрен один из общепринятых методов численного моделирования в нанофотонике - метод FDTD (Finite-Difference Time Domain); с помощью метода FDTD исследуются электромагнитные поля прямого и изогнутого волноводов.

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».

Оглавление

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 HP. 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.

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»).

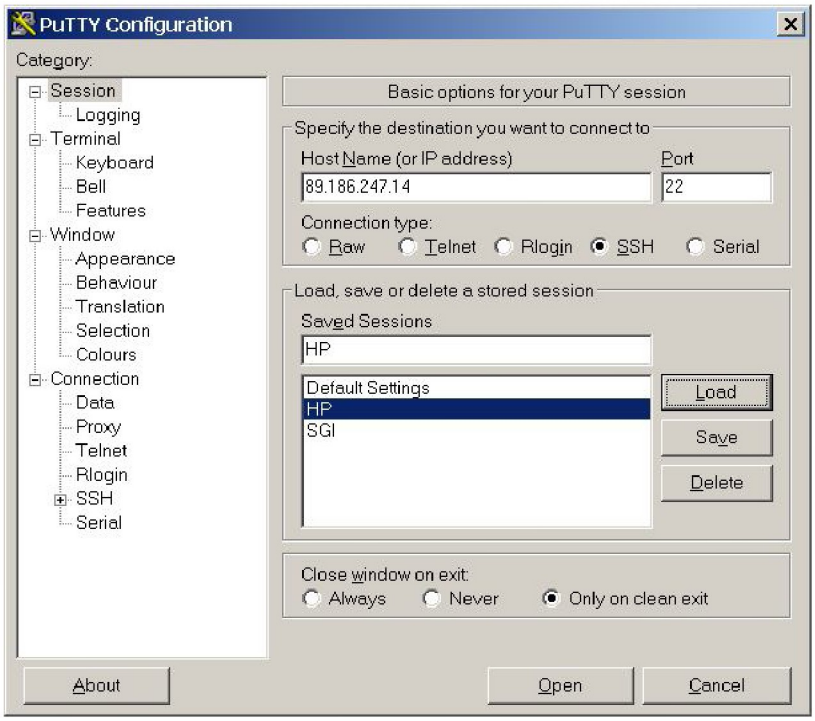

Рис. 1. Конфигурация программы PuTTy

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.

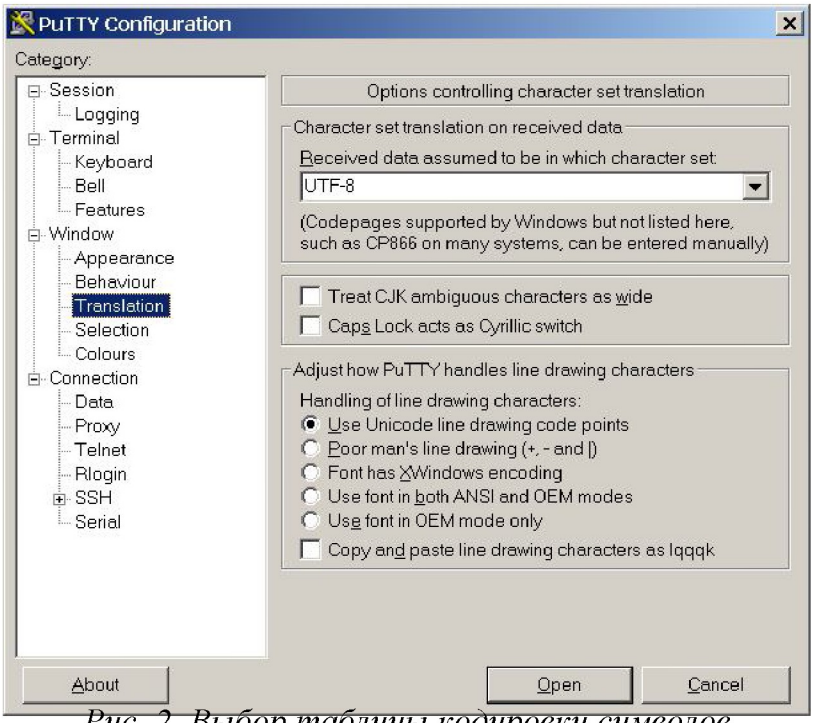

Рис. 2. Выбор таблицы кодировки символов

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.

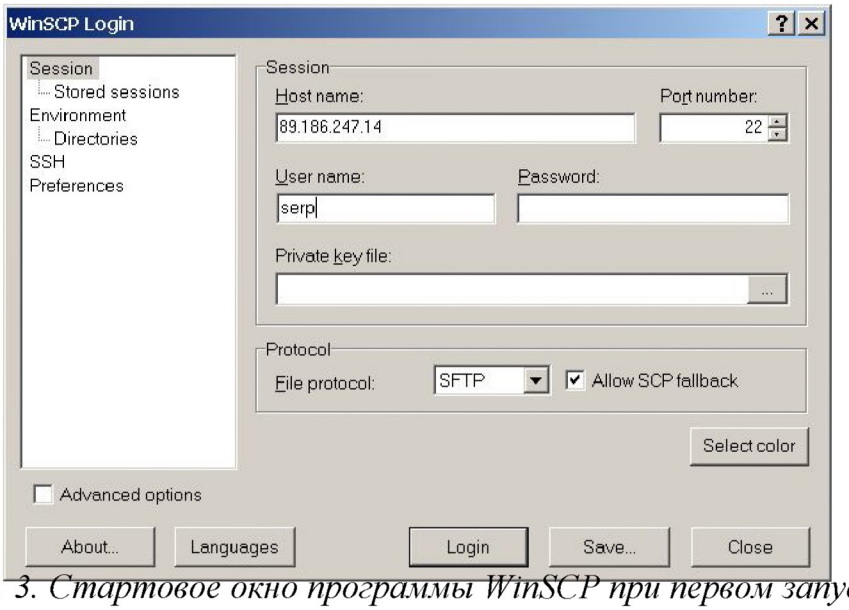

Рис. 3. Стартовое окно программы WinSCP при первом запуске

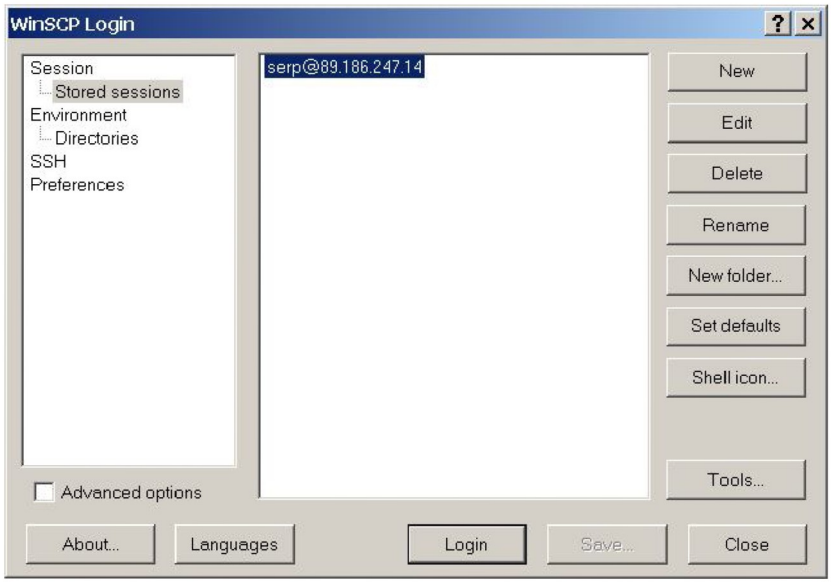

Рис. 4. Окно соединения

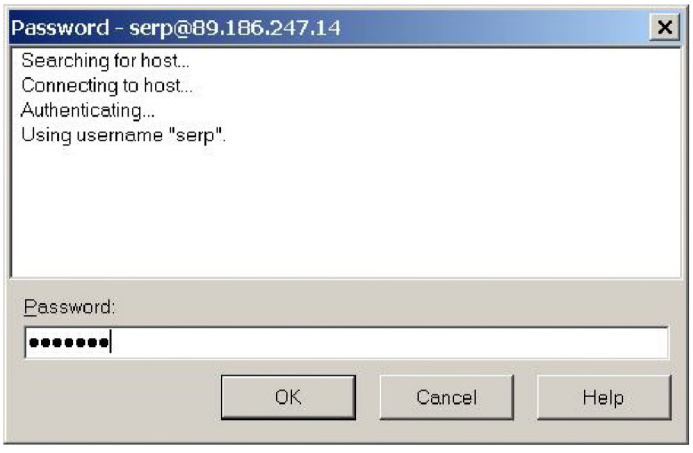

Рис. 5. Окно ввода пароля

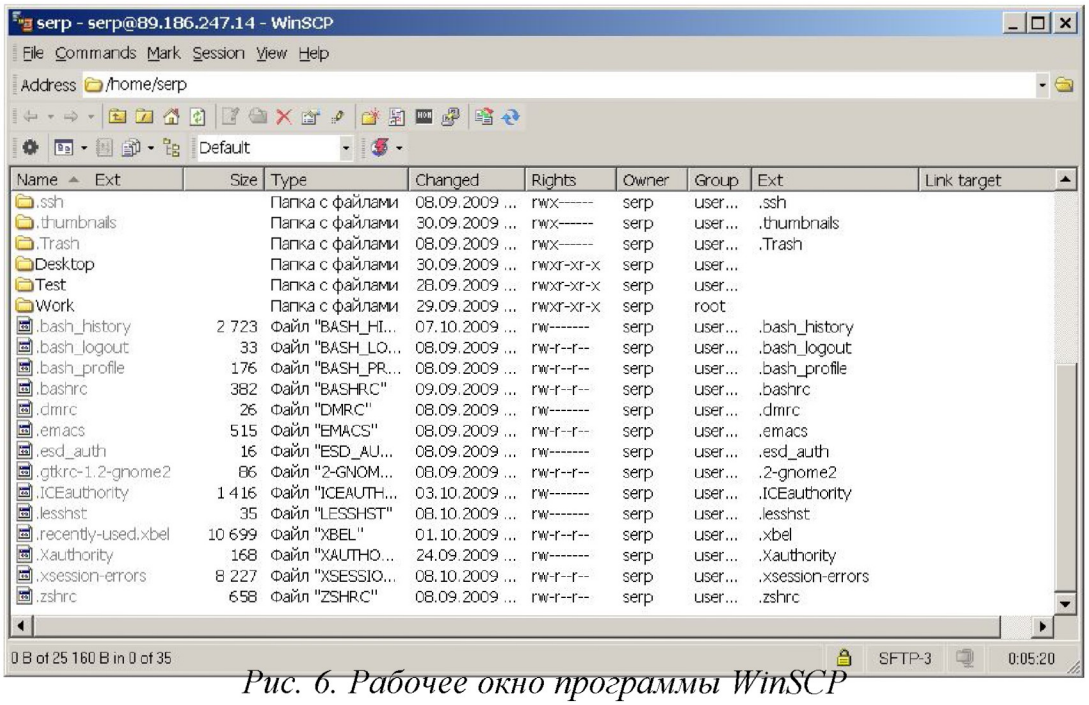

Рис. 6. Рабочее окно программы WinSCP

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**

1.2.5 Проверить установленные настройки

[tester@master ~]$ **switcher mpi --show**
user:default=mpich-ch_p4-gcc-1.2.7
user:exists=1
и/или
[tester@master ~]$ **which mpirun**
/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun
и/или
[tester@master ~]$ **cexec which mpirun**
********** *oscar_cluster* **********
--------- n1---------
/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun
--------- n2---------
/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun
--------- n3---------
/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun
--------- n4---------
/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun

1.3 Подготовка исполняемого файла MPI приложения

1.3.1 Редактирование текста программ пользователя

Процесс разработки программного обеспечения, независимо от размеров и предназначения программы содержит этап редактирования исходных текстов программы. Конечно, вы можете изменять свои программы на рабочем компьютере, а потом копировать их на кластер. Также вы можете редактировать файлы прямо на кластере. Для этого на кластере имеется несколько текстовых редакторов разной степени удобства: vim, ed, emacs, joe. Наиболее простым для освоения, на наш взгляд, является редактор, встроенный в оболочку MidnightCommander (рис. 7).

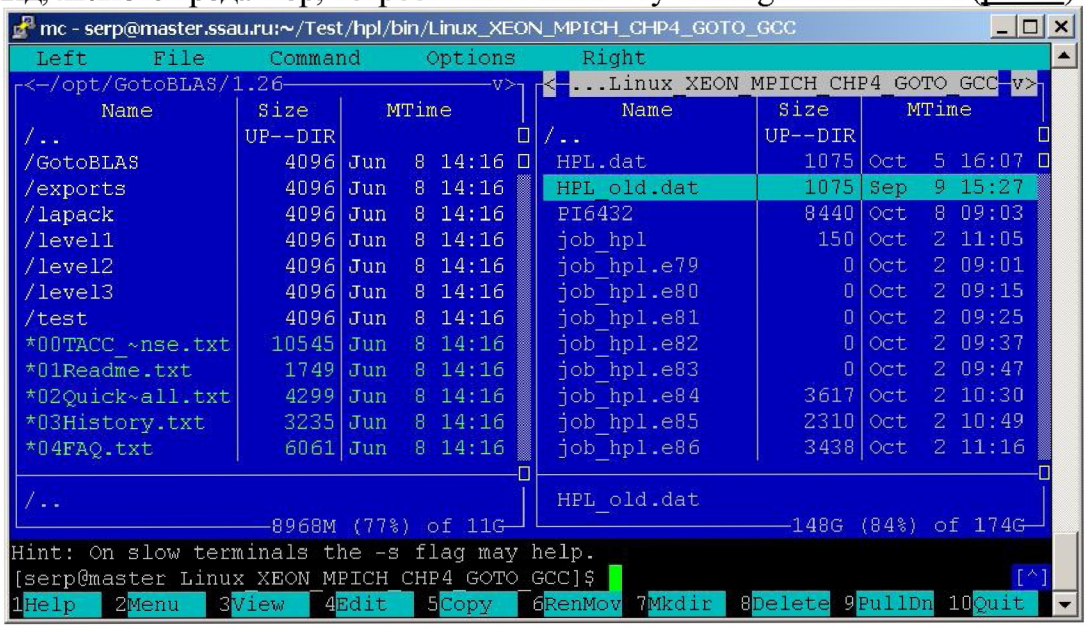

Рис. 7 Окно оболочки MidnightCommander

Для запуска этой программы используется команда mc.

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 MC, 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 ctl-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, которой 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/Meep/Example
/usr/mpi/intel/openmpi-1.2.8/bin/mpirun -np 16 -hostfile /home/student/cluster_ns /usr/local/bin/meep-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 инфомация 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 ctl-файла

Использование Meep тесно связано с управляющим файлом, который имеет расширение «ctl», т.е. называется, например, example.ctl. Ctl-файл определяет изучаемую геометрию, источники тока, вычисляемую мощность и все остальное, касающееся наших расчетов. Как было уже сказано, ctl-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 ctl-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 ctl-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 $\varepsilon = 12$ (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 $\infty \times 1 \times \infty$. $\varepsilon = 12$, волновод центрирован в точке с кординатами (0,0) (положение центра рабочей области). По умолчанию в любом месте, где нет никаких объектов, находится воздух ($\varepsilon = 1$), хотя это можно изменить, устанавливая значение переменной *default-material*. Получившаяся структура показана на рис. 8.

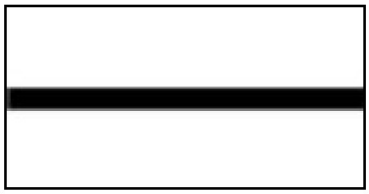

Рис. 8 Геометрия прямого волновода

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 (− iωt)* с постоянной частотой. Этот источник (по умолчанию) запускается в момент времени *t=0*. Отметим, что в Meep единица частоты соответствует значению 2πc, 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 $\varepsilon = 12$. 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 Hx, Hy и/или Hz). 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 точек.

(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 ε и компоненту электрического поля *Ez*. 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.ctl 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 бесплатные 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.

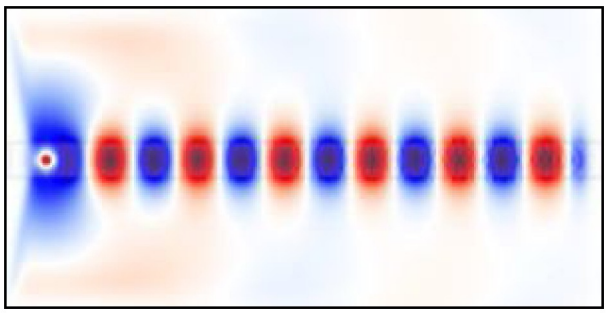

Рис. 9 Поле в прямом волноводе

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 плавнее.

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 Meep в интерактивном режиме, то необходимо сначала избавиться от старой структуры и полей, чтобы Meep заново рассчитал их:

```
(reset-meep)
```

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 12×1 находится в точке (-2, -3,5). Зеленый отрезок показывает положение источника при x=-7.

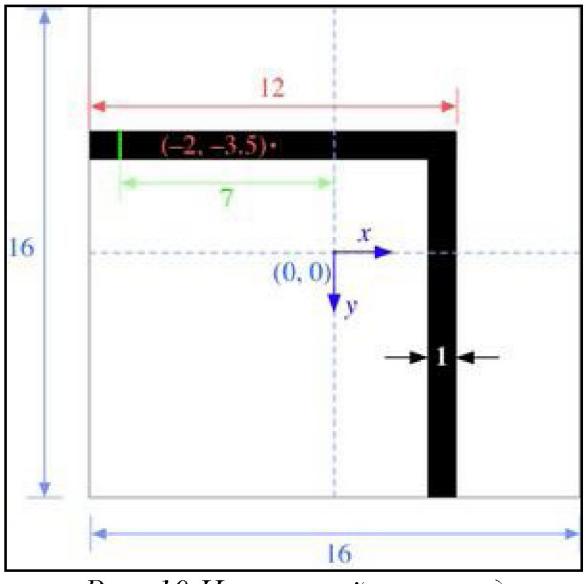

Рис. 10 Изогнутый волновод

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 Meep, 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 Meep 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 161×161×330; последнее измерение – это время. Получившийся файл достаточно большой, 69 MB; 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 со-

здадим 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.

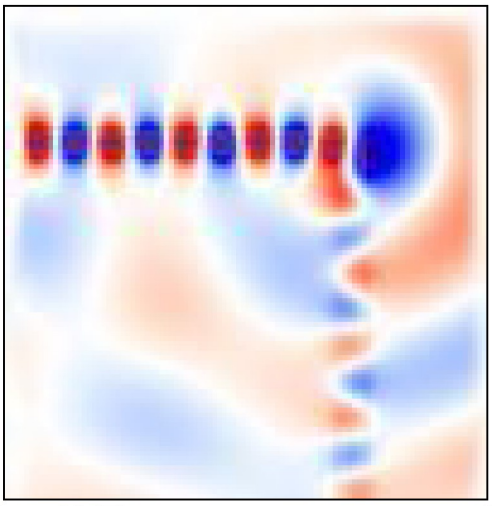

Рис. 11 Поле в изогнутом волноводе

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 зааданному моменту времени. Хранить такие изображения намного более выгодно, чем непосредственно массивы чисел. Чтобы воспользоваться этим, в Meep предусмотрена

возможность выводить данные не только в HDF5-файлы, но и в изображения формата 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 огромно 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 Meep – 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 внимание 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 × 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 ω. Более того, мы получим полный спектр пропускания, запустив модель лишь однажды. Для этого потребуется построить Фурье-образ отклика системы на короткий импульс. Но с целью нормирования значения прошедшего излучения (т. е. чтобы рассчитать прошедшее излучение как долю от падающего), нам потребуется 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 определенно 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 «прокладки» *(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 определятся 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 раздельных 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 разрешение 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 | Ez | ^ 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.ctl | tee bend0.out
unix% meep bend-flux.ctl | 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 *dlmread* и вывести графическое представление результатов (рис. 12).

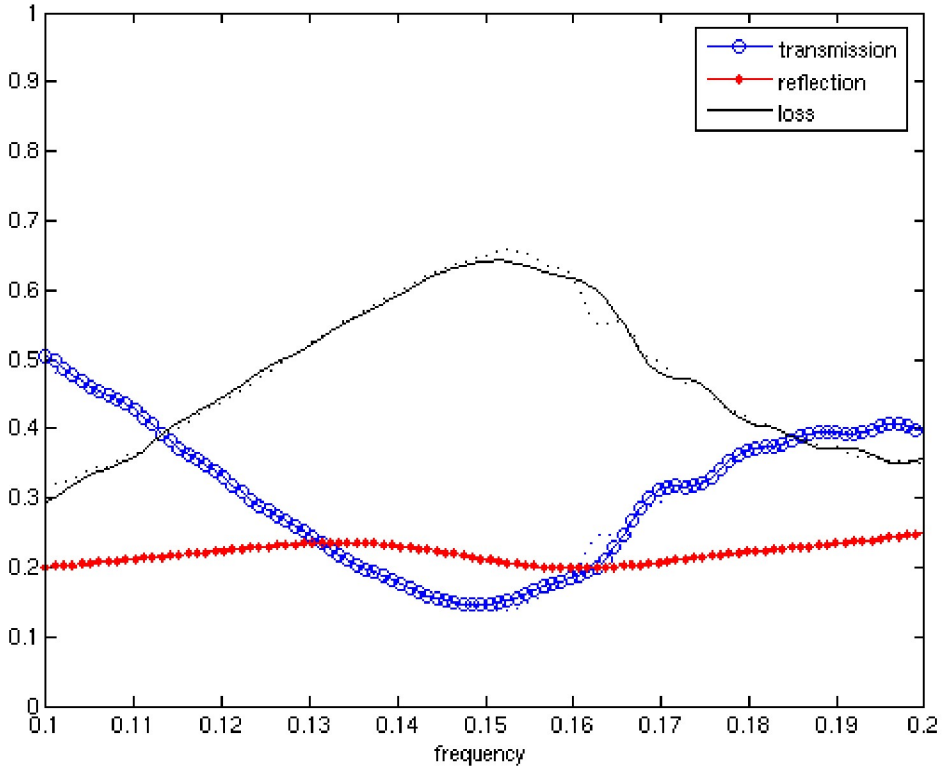

Рис. 12 Отражение и пропускание изогнутого волновода

Рассмотрим приведенный график. Transmission соответствует прошедшему потоку (второй столбец в файле *bend.dat*) , поделенному на падающий поток (второй столбец в *bend0.dat)*. Таким образом, мы рассматриваем долю прошедшего излучения. *Reflection* соответствует отраженному излучению (третий столбец из *bend.dat)* , поделенному 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– transmission–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.ctl*, 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) ; расстоние 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 (ε = 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 (TM-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 выполняем 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, -0.0030502290529417 5-0.00153321402956404i,
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 *an* – комплексные амплитуды, *ωn* – комплексные частоты. Все 6 столбцов связаны с этими величинами: первый столбец содержит действительную часть *ωn* , выраженную в привычных нам единицах *2πc,* второй столбец содержит мнимую часть *ωn* – отрицательную величину, соответствующую экспоненциальному затуханию. Для резонаторов скорость затухания чаще выражается как безразмерное «время жизни» Q , определяемое как

$$Q = \frac{\mathrm{Re}\,\omega}{-2\mathrm{Im}\,\omega}.$$

(Q равняется числу оптических периодов, за которые мощность излучения падает в *exp(-2π)* 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 |*an*| и комплексную величину амплитуд *an*. 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, удавается 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 ω на радиус резонатора. Теперь предположим, что требуется построить картину электромагнитных полей соответствующим этим модам. Это не составля-

ет 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 Ez 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/fcen. С этой целью добавим команду:

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

Теперь можно получить требуемые моды, запустив программу:

unix% meep fcen=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 dkbluered -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.

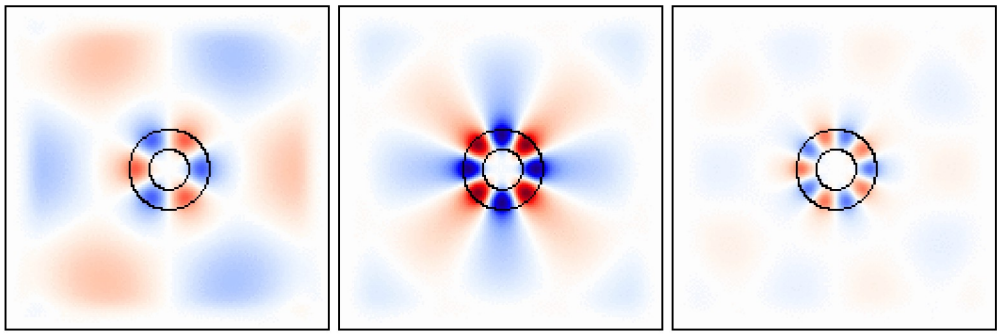

Рис. 13 Моды кольцевого резонатора

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 ω = 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 ω 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 Meep вводит в системы зеркальную симметрию, плоскость которой проходит через начало координат и перпендикулярно оси *y*. При этом Meep 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 Meep, 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.ctl.,* который содержится в пакете Meep.

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,φ,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 двухмерная. Переменная φ также безразмерная (no-size), что соответствует непрерывной вращательной симметрии (конечный размер φ 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(imφ)**, 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 r. 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 TM колебания определенной центральный частоты и ширины полосы:

```
(define-param fcen 0.15) ; центральныя 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 φ в виде exp(imφ). Наконец, как и ранее, мы запускаем моделирование до момента выключения источника и, затем, ждем еще 200 временных единиц: в течение которых мы изучаем поле Ez (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. Meep 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-efield-z))))
```

Теперь мы готовы начать моделирование. Напомним, что при 2-мерном вычислении у нас получились три волны: одна при ω = 0.11785 ;Q = 77 и m = 3; другая при ω = 0.14687 ; Q = 351 и m = 4; последняя при ω = 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 моделирования 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 ω = 0.14748 и Q = 383. Для 2-мерного моделирования, если удвоить разрешение до 20, то получим ω = 0.14733 и Q = 321. Похоже, что значения частот сходятся, при этом, цилиндрическое моделирование точнее Этого можно было ожидать, учитывая, что направление φ 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 «error», погрешность для двумерного моделирования 4e-7, в то время, как для цилиндрического 3e-11. Можно снизить ошибку, проведя моделирования 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 *dpml*). Для цилиндрического моделирования при разрешении 20 если увеличить dpml до 16, то получим Q=342, что куда лучше согласуется с 2-мерными расчетами (а если увеличить dpml до 32, то также получится Q=342; таким образом, эти значения сходятся).

Этот анализ иллюстрирует то, что при использовании метода FDTD небходимо 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 ω = 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 160×18, соответствующий *r*×*t* слою. Повторив это для всех трех волн, получим изображения полей (рис. 14).

Поскольку мы рассматриваем слои с φ = 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 наружу) становится меньше, как и предполагалось.

Ez при моде ω = 0.118 *m* = 3

Ez при моде ω = 0.148 *m* = 4

Ez при моде ω = 0.176 *m* = 5

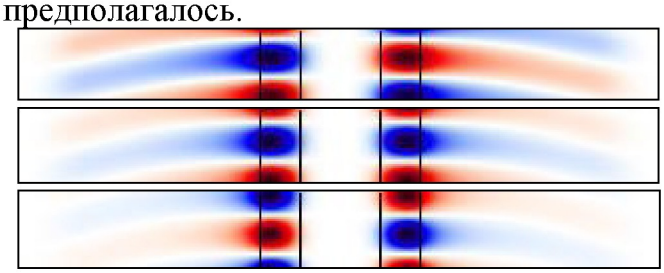

Рис. 14 Моды кольцевого резонатора в цилиндрических координатах

2.4 Дисперсия света в веществе

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

$$\varepsilon(\omega) = \left(\frac{ck}{\omega}\right)^2 .$$

В дальнейшем мы проведем сравнение значений ε(ω), 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 использованы 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

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

Важно обратить внимание на то, что в Meep единицей измерения частоты ω является 2πc/*a*, поэтому в действительности приведенное выражение представляет собой следующее:

$$\varepsilon(f) = 2.25 + \frac{1.1^2 \cdot 0.5}{1.1^2 - f^2 - if \cdot 10^{-5}/2\pi} + \frac{0.5^2 \cdot 2 \cdot 10^{-5}}{0.5^2 - f^2 - if \cdot 0.1/2\pi}$$

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

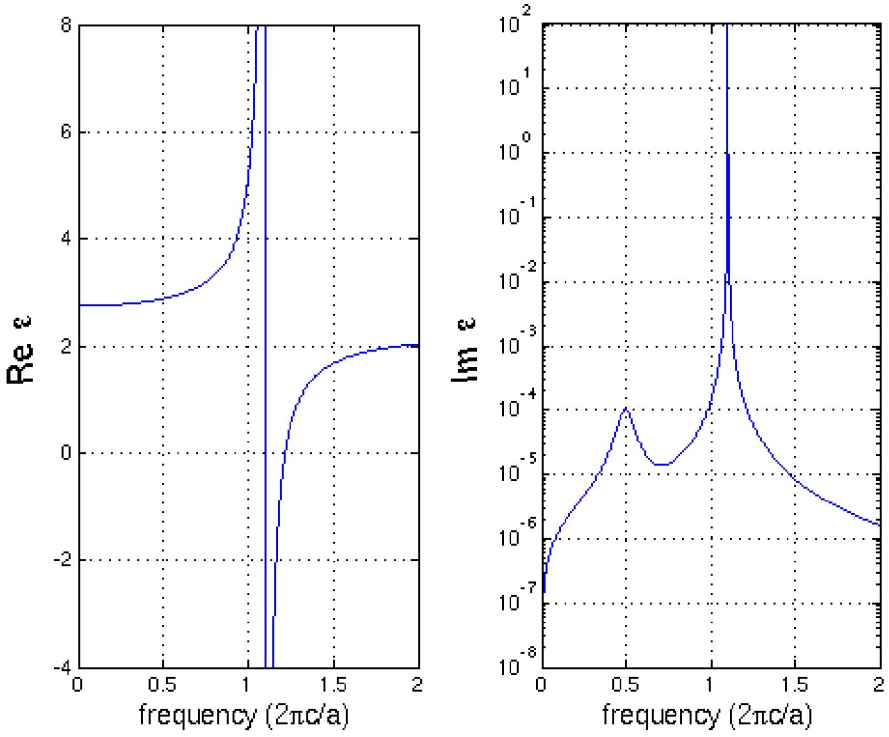

Рис. 15 Действительная и мнимая части рассматриваемой диэлектрической функции

Можно видеть, что резонанс при значении ω = 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 ε 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 ω = 0.5 , из-за малости параметра **sigma** приводит к лишь небольшим изменениям в действительной части ε. 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 ω (*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 ε (используется соотношение (ck / ω)^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)
               ", " (sqr (/ 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.ctl | tee material-dispersion.out
```

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

Здесь красные кружки соответствуют значениям, рассчитанным Meep, тогда как голубая линия – аналитическая зонная диаграмма по заданным значе-

ниям ε(ω). 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 Meep. Но почему нет полного совпадения, как на предыдущем графике? Причиной является не ошибка вычислений Meep или какая-либо численная погрешность. Дело в том, что мы пытаемся сравнить несравнимое.

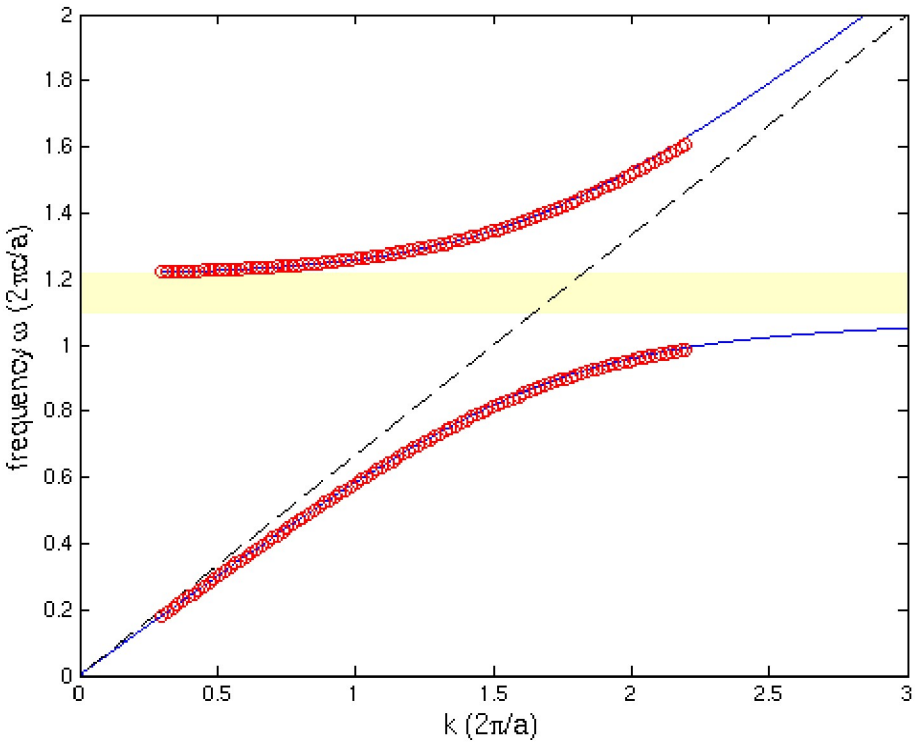

Рис. 16 Дисперсионное соотношение Re[ω(k)]

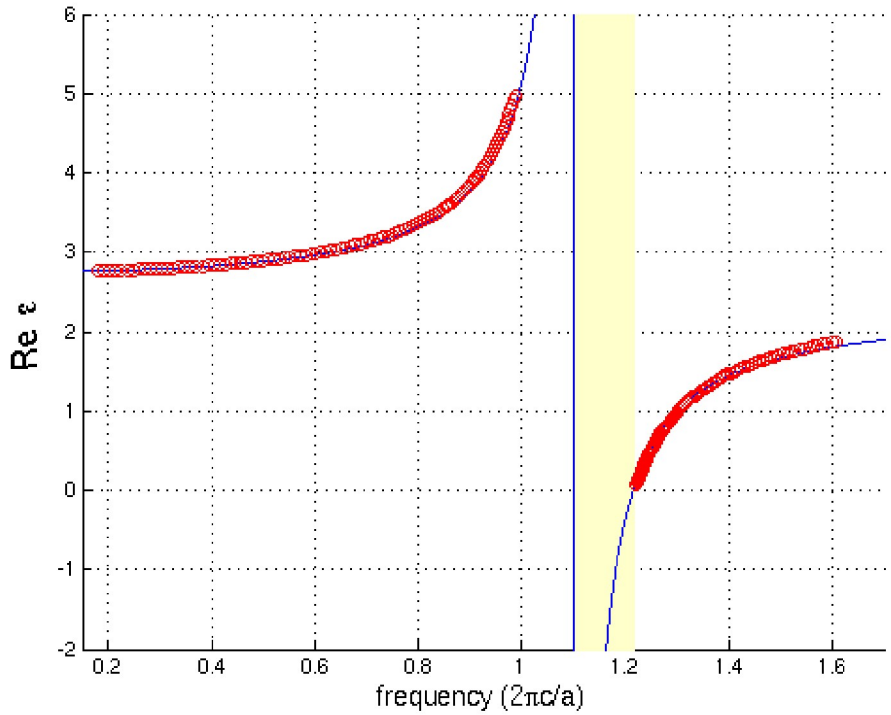

Рис. 17 Действительная часть диэлектрической функции ε(ω)

Голубая линия соответствует аналитическому расчету ε(ω), проведенному для *действительной* частоты ω (что соответствует решениям с *комплексным* значением волнового вектора *k*), тогда как в Meep ε вычислялось при *комплексном* ω и *действительном k*. Таким образом, чтобы провести верное сопоставление, требуется подставить *комплексное* значение ω, использованное Meep, в аналитическую формулу для ε(ω). Результат отражен на графике в виде зеленых линий, которые лежат практически поверх красных кружков.

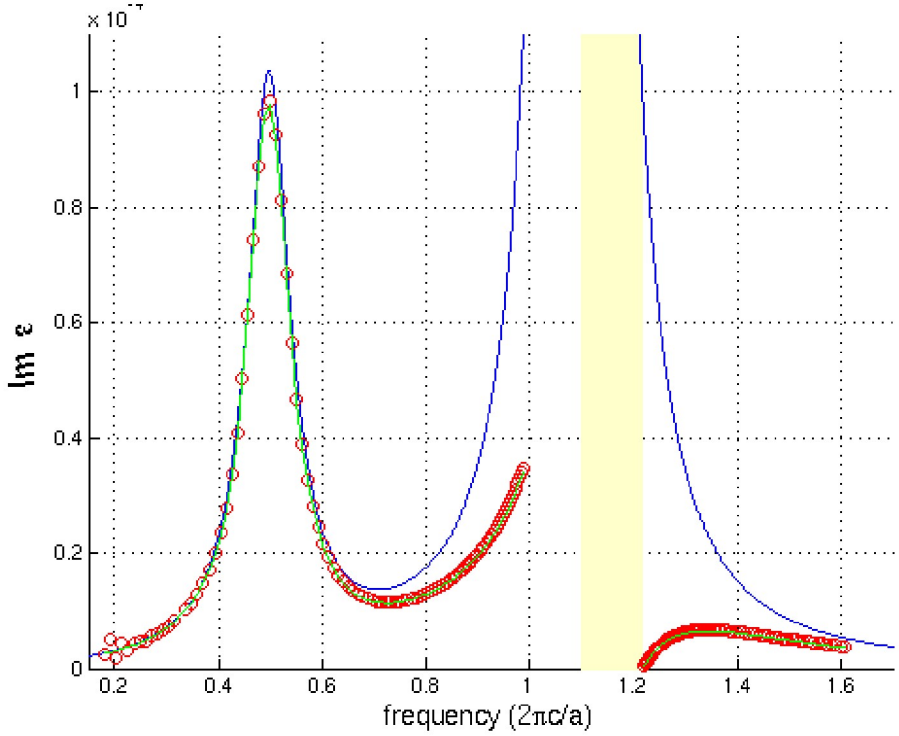

Рис. 18 Мнимая часть диэлектрической функции ε(ω)

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

3. Задания к лабораторным работам

1. Путем численных экспериментов, найти ширину волновода при которой прямой волновод перестает быть одномодовым.

2. Исследовать влияние толщины PML-слоя на результаты расчета поля в прямом волноводе.

3. Путем численных экспериментов, найти ширину волновода при которой изогнутый волновод перестает быть одномодовым.

4. Исследовать влияние толщины PML-слоя на результаты расчета поля в изогнутом волноводе.

5. Исследовать влияние длины прямого волновода на величину прошедшего поля через волновод.

6. Исследовать влияние длины прямого волновода на величину отражения поля в волноводе.

7. Исследовать влияние длины горизонтальной части изогнутого волновода на величину прошедшего поля через волновод.

8. Исследовать влияние длины вертикальной части изогнутого волновода на величину прошедшего поля через волновод.

9. Исследовать влияние длины горизонтальной части изогнутого волновода на величину отраженного поля в волноводе.

10. Исследовать влияние длины вертикальной части изогнутого волновода на величину отраженного поля в волноводе.

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_MOMPORT=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. Сойфер В.А. Введение в дифракционную микрооптику. - Самара: СГАУ, 1996. - 95с.
4. Методы компьютерной оптики / Под редакцией В.А. Сойфера. - М.: Физматлит, 2003. – 688с.
5. 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
10. http://ab-initio.mit.edu/wiki/index.php/Libctl_manual.