Д.В. Ситанов, С.А. Пивоваренок

FreeMat

для инженерных и научных расчетов

Иваново 2018

Министерство науки и высшего образования Российской Федерации
Ивановский государственный химико-технологический университет

Д.В. Ситанов, С.А. Пивоваренок

FreeMat для инженерных и научных расчетов

Учебное пособие

Иваново 2018

УДК 621.382

Ситанов, Д.В. FreeMat для инженерных и научных расчетов: учеб. пособие / Д.В. Ситанов, С.А. Пивоваренок; Иван. гос. хим.-технол. ун-т. – Иваново, 2018. – 87 с.

Учебное пособие содержит описание свободно распространяемой программы для математического моделирования и обработки данных, представляющую собой матричную систему для инженерных и научных расчетов (FreeMat). FreeMat использует средства объектно-ориентированного программирования и имеет удобный интерфейс доступа к своим программным модулям.

Предназначено для бакалавров, обучающихся по направлению подготовки «Электроника и наноэлектроника» (профиль подготовки «Микроэлектроника и твердотельная электроника») при изучении дисциплины «Наноэлектроника» и магистрантов, обучающихся по направлениям подготовки «Электроника и наноэлектроника» (магистерская программа «Микро- и нанотехнологии в производстве изделий твердотельной электроники»), а также «Химическая технология» (магистерская программа «Микро- и нанотехнологии в производстве изделий электронной техники») при изучении дисциплин «Проектирование электронной компонентной базы» и «Проектирование микро- и нанотехнологических процессов в электронике».

Печатается по решению редакционно-издательского совета Ивановского государственного химико-технологического университета.

Рецензенты:

кафедра энергетики теплотехнологий и газоснабжения Ивановского государственного энергетического университета имени В.И. Ленина;

кандидат физико-математических наук, доцент Н.Е. Егорова (ФГБОУ ВО Ивановская пожарно-спасательная академия ГПС МЧС России).

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

FreeMat – свободно распространяемая матричная система для инженерных и научных расчетов, а также обработки данных и математического моделирования.

В данном учебном пособии описываются основы работы в FreeMat и охватываются только те возможности пакета, которые необходимы при выполнении лабораторного практикума в рамках бакалавриата и моделирования при выполнении НИР магистрантами различных направлений подготовки. Основное внимание уделяется описанию работы из командной строки, вычислению арифметических выражений, использованию одномерных и двумерных массивов, а также основам написания программ в среде FreeMat. Подробно рассмотрены графические возможности FreeMat для визуализации данных и построения графиков функций.

Целевая аудитория данного учебного пособия, в первую очередь, это бакалавры и магистранты технологических направлений подготовки, преподаватели математики, физики, технологических дисциплин и близких им предметных областей, в которых часто применяются сложные математические расчеты и геометрические построения.

В пособии описаны следующие возможности FreeMat:

- арифметические действия над числами, векторами, матрицами;
- основы написания программ по заданным алгоритмам;
- построение 2D и 3D графиков.

К моменту написания данного учебного пособия актуальной версией FreeMat являлась версия v.4.2, которая и была взята за основу при обсуждении вышеуказанных вопросов.

Цели и принципы построения лабораторного практикума

Пособие содержит описание программного пакета FreeMat, необходимое для выполнения технологических и научных расчетов, а также описание 8 лабораторных работ, последовательное выполнение которых позволит сформировать должный уровень компетенций у обучающихся в сфере работы и моделирования наносхемотехнических устройств.

Каждой лабораторной задаче предшествует краткое теоретическое введение, отражающее суть изучаемого явления, а также основы метода моделирования. Описание работы включает в себя рекомендации по методикам расчетов и обработки полученных результатов.

Одной из важных задач лабораторного практикума является приобретение студентами навыков экспериментальной работы, грамотного представления результатов, обработки и анализа полученных данных. В связи с этим в практикуме даны рекомендации по построению графиков и правила обработки результатов расчетов.

Порядок прохождения практикума

Лабораторный практикум выполняется в соответствии с графиком и календарным планом, которые составляются на каждый учебный год. С содержанием и планом прохождения практикума, требованиями к выполнению работ и оформлению отчетов студенты знакомятся на первом вводном занятии. На этом же занятии проводится инструктаж по технике безопасности при работе в дисплейном классе, и выдаются задания на выполнение первой лабораторной работы.

Лабораторные работы выполняются индивидуально.

В конце занятия студент должен получить индивидуальное задание для подготовки к очередной лабораторной работе.

Подготовка к выполнению лабораторной работы включает в себя:

- изучение теоретического материала по краткому теоретическому введению, конспектам лекций и учебной литературе, указанной в конце практикума;
- определение всех расчетных величин и процедур, необходимых для дальнейших расчетов с целью достижения конечной цели работы.

Перед каждой работой студент проходит краткое собеседование с преподавателем для выяснения уровня готовности к выполнению задачи. Результаты собеседования учитываются при выставлении оценки за выполнение работы.

Общие требования к оформлению отчетов по лабораторным работам

1. Отчет оформляется на отдельных чистых листах бумаги, скрепленных между собой.
2. Отчет должен содержать титульный лист с указанием даты выполнения лабораторной работы, темы лабораторной работы, основного исполнителя и преподавателя, выдавшего задание, а также даты сдачи лабораторной работы в соответствие с графиком выполнения лабораторных работ.
3. Основная часть отчета должна содержать:

– задачи и цели лабораторного исследования объекта, в данном случае с использованием средств программирования и компьютерного моделирования в среде FreeMat;
– небольшое теоретическое введение, самостоятельно сформулированное и оформленное обучающимся, отражающее суть и цели лабораторной работы (копирование текста не допускается);
– отлаженные в среде FreeMat тесты программ, таблицы с результатами расчетов и промежуточными данными, графики, отражающие результаты расчетов, а также анализ полученных результатов с привлечением приведенного в отчете исходного материала;
– выводы по лабораторной работе;
– расчет времени на выполнение лабораторной работы, обработку экспериментальных результатов и сдачи лабораторного исследования преподавателю.

1. Физика матриц. Основные понятия

Матрицы – это особые математические объекты, которые могут быть представлены в виде таблицы, состоящей из строк и столбцов, с произвольным числом в каждой клетке:

1	25	-100
16	-32	0,001
13,6	-70	-0,1

Обычно их пишут в скобках и без клеток:

$$\begin{pmatrix} 1 & 25 & -100 \\ 16 & -32 & 0{,}001 \\ 13{,}6 & -70 & -0{,}1 \end{pmatrix}.$$

С матрицами можно производить различные операции (сложение, вычитание, умножение или деление), которые дают новые матрицы в соответствии с особыми математическими правилами.

В физике, в частности в квантовой механике, наиболее часто используются квадратные матрицы. К примеру, квадратная матрица из двух строк и двух столбцов выглядит так:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.$$

Элементы матрицы, помимо своего значения (какого-либо конкретного числа), имеют индексы: первый указывает строку, в которой находится число, второй – столбец. Таким образом, можно напрямую определить конкретное число матрицы, указав индексы, то есть строку и столбец. Например, имеем квадратную *C*:

$$C = \begin{pmatrix} 16_{11} & -72_{12} \\ -0{,}01_{21} & 100_{22} \end{pmatrix}.$$

В этой матрице *C* (массиве, говоря языком программистов) располагаются (хранятся) четыре числа: 16; -72; -0,01 и 100, а индексы лишь указывают их расположение в самой матрице (массиве). Например, $\boldsymbol{16_{11}}$ – означает, что число ***16*** находится в первой строке первого столбца и т.д.

Сложение и вычитание матриц интуитивно понятны: для этого нужно почленно сложить или вычесть элементы исходных матриц. Произведение матриц, к примеру, рассчитывается по особому закону:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}.$$

При умножении матриц порядок множителей, в общем случае, влияет на конечный результат. К примеру, произведения матриц:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \text{ и } B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ равны:}$$

$$A \cdot B = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \text{ и } B \cdot A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Эти матрицы различаются между собой. Разность этих произведений будет матрица:

$$A \cdot B - B \cdot A = \begin{pmatrix} 0 & 2 \\ -2 & 0 \end{pmatrix}.$$

Таким образом, одним из их основных свойств матричного исчисления является некоммутативность матричного произведения: $A \cdot B \neq B \cdot A$. Это означает, что хорошо известный принцип, согласно которому «порядок множителей не влияет на произведение», не выполняется. Чтобы привести более наглядный пример некоммутативности какой-либо операции, рассмотрим вращения в пространстве. Повороты математически могут быть представлены как произведение матриц. Пусть M и S – это две точки на сфере. Если мы осуществляем два последовательных оборота вокруг осей, которые проходят через них, результат будет зависеть от направления (см. рис. 1).

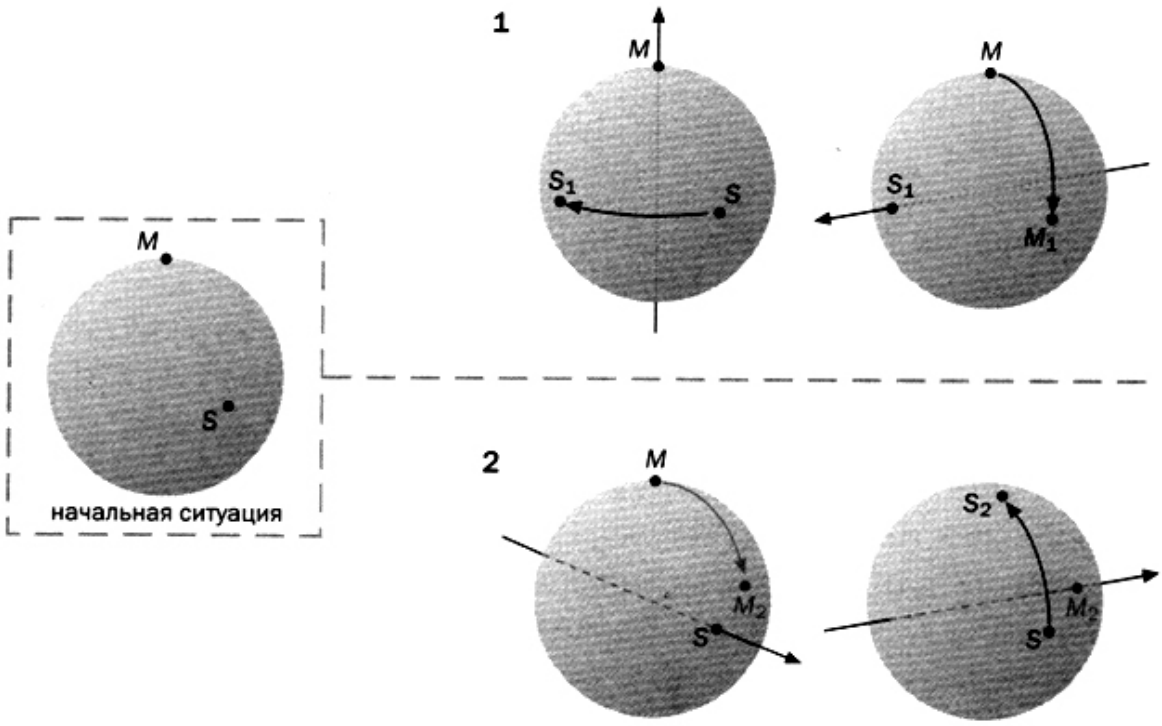


Рис. 1. Демонстрация принципа некоммутативности операций последовательного вращения сферы: в первом случае (показан сверху) вначале вращают S вокруг оси, проходящей через M; во втором случае (показан снизу) вначале вращают M вокруг оси, проходящей через S. При первом повороте конечное расположение M и S – это M_1 и S_1; при втором повороте, соответственно, M_2 и S_2. Как можно убедиться M_2 и S_2 для различных действий не совпадают. Второй случай переводит точку M_2 на другую сторону сферы

Применительно к квантовой механике можно записать следующее выражение:

$$P \cdot Q - Q \cdot P = -\frac{ih}{2\pi} \cdot I \,, \qquad (1)$$

где P и Q – являются матрицами, представляющими количество движений: импульс (P) и расположение (Q) квантовой частицы;

i – корень от -1;

h – постоянная Планка;

I – единичная матрица, которая играет такую же роль в алгебре матриц, что и число 1 в арифметике.

Уравнение (1) означает, что произведение $(P \cdot Q)$ дает матрицу, отличную от $(Q \cdot P)$. Из этого можно сделать вывод: каждое измерение материального объекта (например, электрона) меняет его. Таким образом, если вначале определяют положение, а затем импульс, результат отличается от того, который мы получим при изменении сначала импульса, а затем положения. Такой феномен называют принципом неопределенности (Вернер Гейзенберг, 1927 г.). В общем, если h можно рассматривать как исчезающе малую величину, мы имеем дело с явлением, которое можем наблюдать с помощью наших органов чувств, и можно предположить, что постоянная планка (h) как бы равна нулю.

Таким образом, если $h \rightarrow 0$, тогда: $P \cdot Q - Q \cdot P = 0$, откуда: $P \cdot Q = Q \cdot P$. Произведение вновь становится коммутативным, и мы оказываемся в обычной ситуации, описываемой классической физикой. Аналогично, расстояние между дискретными значениями стремится к нулю и доходит до него, что позволяет вернуться к классическому подходу.

Уравнение (1) играет такую же важную роль в матричной механике (идеологом которой считается В. Гейзенберг), как и уравнение Шредингера для волновой механики. На самом деле некоммутативность матриц означает, что мы имеем дело с квантовым состоянием. Классическая непрерывность выражается функциями. Квантовая дискретность отлично сочетается с матрицами.

Если представить уровни энергии атома водорода с помощью горизонтальных линий, то получим схему как на рис. 2.

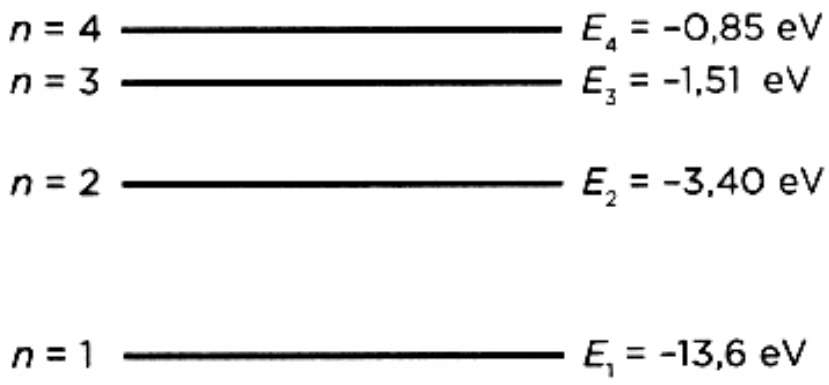


Рис. 2. Уровни энергии атома водорода, построенные с помощью формулы Бора

Затем мы записываем данные в клетки матрицы, указывая значения для каждого уровня энергии вдоль диагонали, а возможные переходы – вне диагонали. Таким образом, элемент E_{mn} матрицы соответствует скачку ($E_n - E_m$). Принимая во внимание, что n и m могут расти до бесконечности, матрица тоже увеличивается до бесконечности (рис. 3). Значения других наблюдаемых величин, таких как положение или импульс, также могут быть записаны в бесконечной матрице.

$$E = \begin{pmatrix} E_{11} & E_{12} & E_{13} & \ldots E_{1n} \\ E_{21} & E_{22} & E_{23} & \ldots E_{2n} \\ E_{31} & E_{32} & E_{33} & \ldots E_{3n} \\ \cdot & \cdot & \cdot & \cdots \cdot \\ \cdot & \cdot & \cdot & \cdots \cdot \\ \cdot & \cdot & \cdot & \cdots \cdot \\ E_{n1} & E_{n2} & E_{n3} & \ldots \mathrm{E}_{\mathrm{nm}} \end{pmatrix}$$

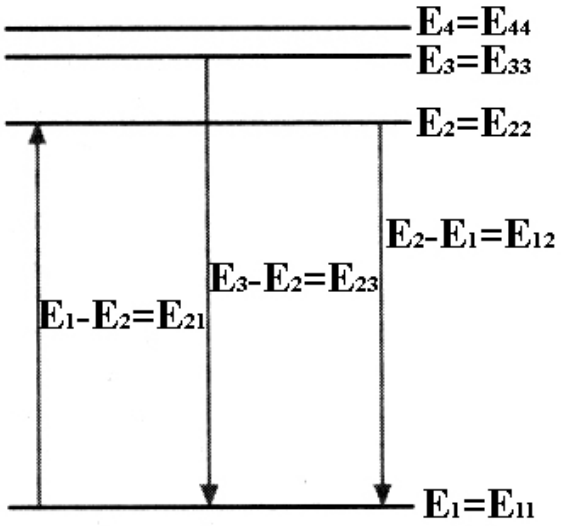


Рис. 3. Энергетическая диаграмма (справа) и энергетическая матрица (слева), в которую записаны энергии связи (главная диагональ) и соответствующие переходы

2. Основные навыки работы с пакетом FreeMat

После запуска программы открывается основное рабочее окно пакета FreeMat, показанное на рис. 4.

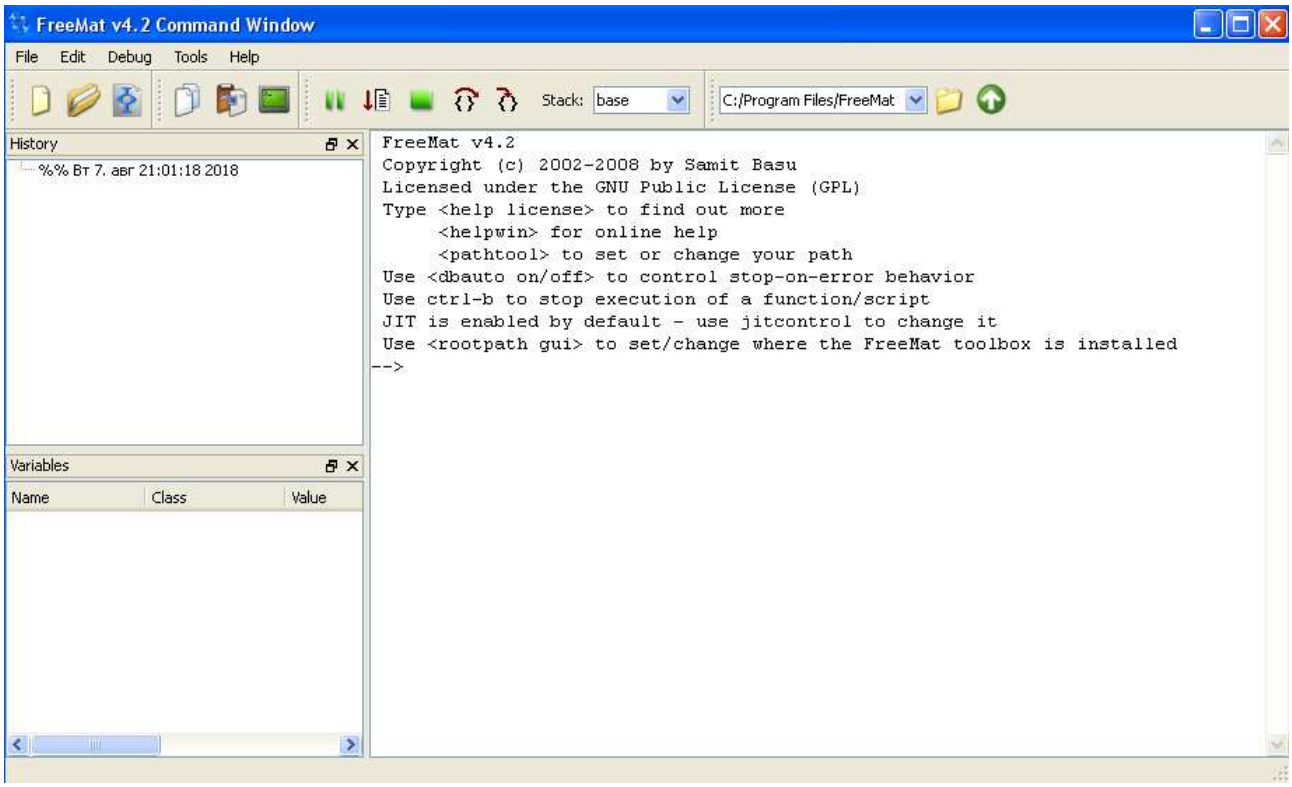
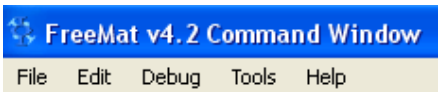


Рис. 4. Основное рабочее окно программы FreeMat

Интерфейс программы интуитивно понятен и содержит:

– основное меню:



– панель управления:

– интерфейсные окна:

а) **History** (История), в котором ведется протокол действий в программе;

б) **Variables** (Переменные), в котором отображаются объявленные в программе переменные;

в) **Command Window** (Окно команд), в котором следует записывать после мигающего курсора команды, функции, элементы программ. В этом же окне выводится результат выполнения команд.

При необходимости из элемента меню «Tools» могут быть вызваны дополнительные окна, наиболее важным из которых является окно редактора программ. Редактор программ может быть вызван также комбинацией клавиш: «Ctrl + E» (рис. 5):

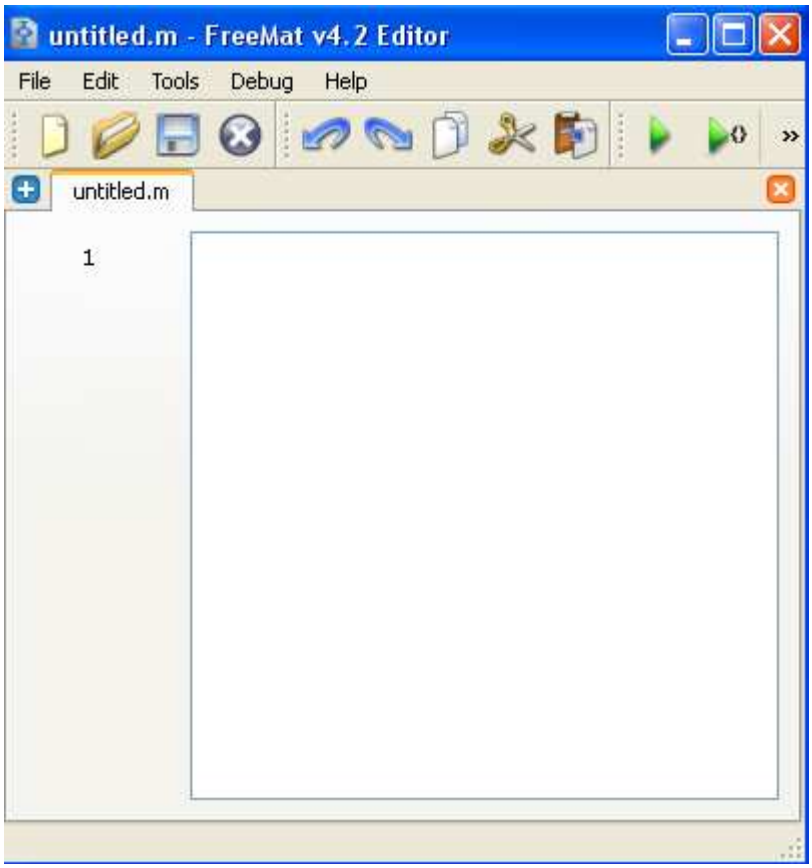


Рис. 5. Редактор программ в FreeMat

При появлении приглашения « – –> » в **Command Window** вы можете ввести переменную или выполнить команду. Система FreeMat чувствительна к выбору регистра, то есть FreeMat различает переменные «a» и «A». Порядок работы с программой может осуществляться в двух режимах – пошаговом и программном.

В первом случае каждой переменной последовательно присваивается некоторое значение при нажатии клавиши **Enter**. Для задания переменным определенных значений используется ***оператор присваивания***, вводимый знаком равенства «=», например:

Имя _ переменной = Значение или Выражение.

Пример работы с константами и переменными в среде FreeMat показан на рис. 6. Сначала программа присвоила переменной ***a*** значение **2**, затем переменной ***b*** значение **0.5** (заметим, что при вводе десятичных дробей используется точка), затем переменной ***c*** было присвоено значение, которое соответствует отношению переменной ***a*** к ***b***. Разница между константами и переменными в среде FreeMat достаточно прозрачна. Считается, что если в процессе расчетов значение какой-либо величины не меняется, например, заряд электрона или постоянная Планка, то эту величину следует интерпретировать

как константу, в противном случае мы будем иметь дело с переменными. Константы и переменные имеют свой тип в зависимости от класса точности (Class), по умолчанию они имеют класс двойной точности (double).

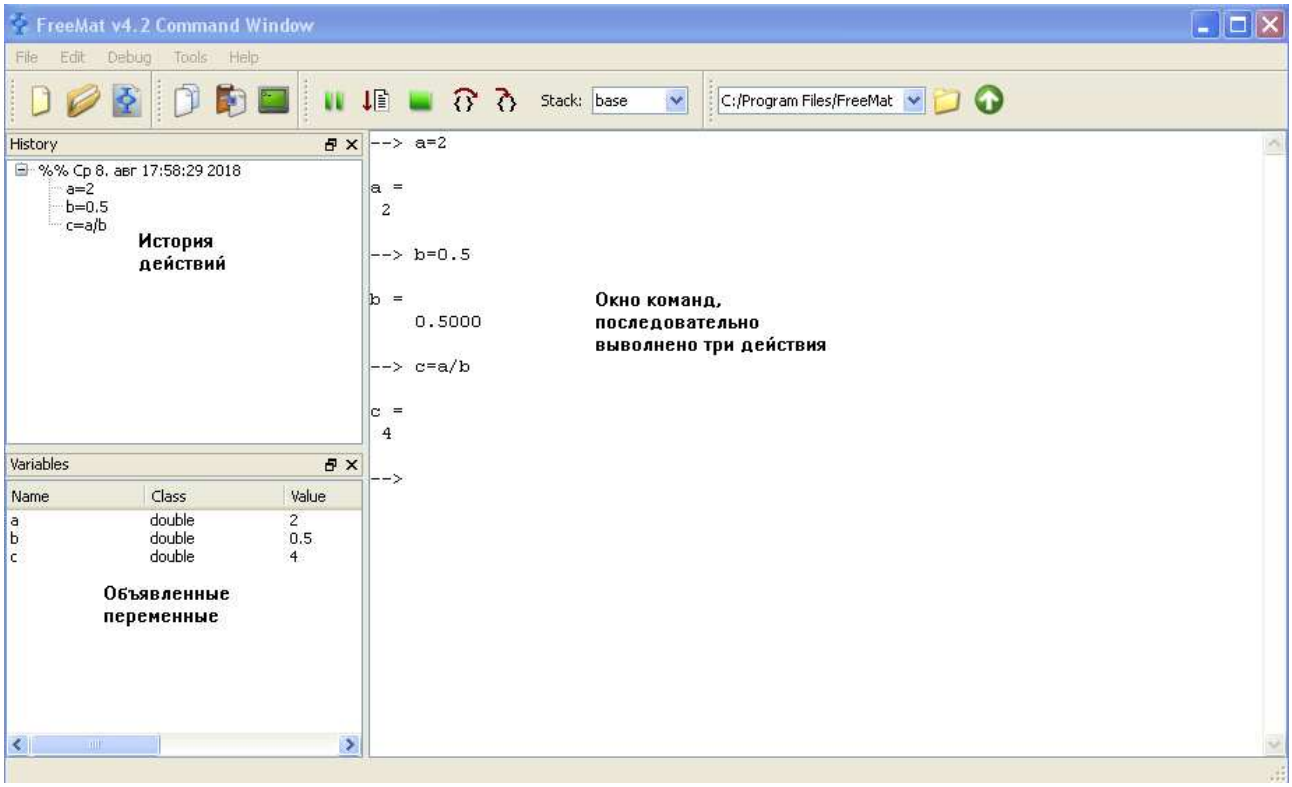


Рис. 6. Пример работы в среде FreeMat в пошаговом режиме

Во втором случае (программный режим) в командном окне набирается программный код (последовательность операторов), и лишь потом программа приступает к вычислениям. Для ввода нескольких операторов (функций) без немедленного выполнения их нужно разделить точкой с запятой (**;**). Если математическое выражение выходит за размер экрана монитора, то целесообразно перенести его часть на следующую строку. Для этого используется символ многоточие (**…**) – три и более точки. На рис. 7 показан второй вариант расчета переменной «***c***» с использованием операторов присваивания, объединенных в простейший программный код.

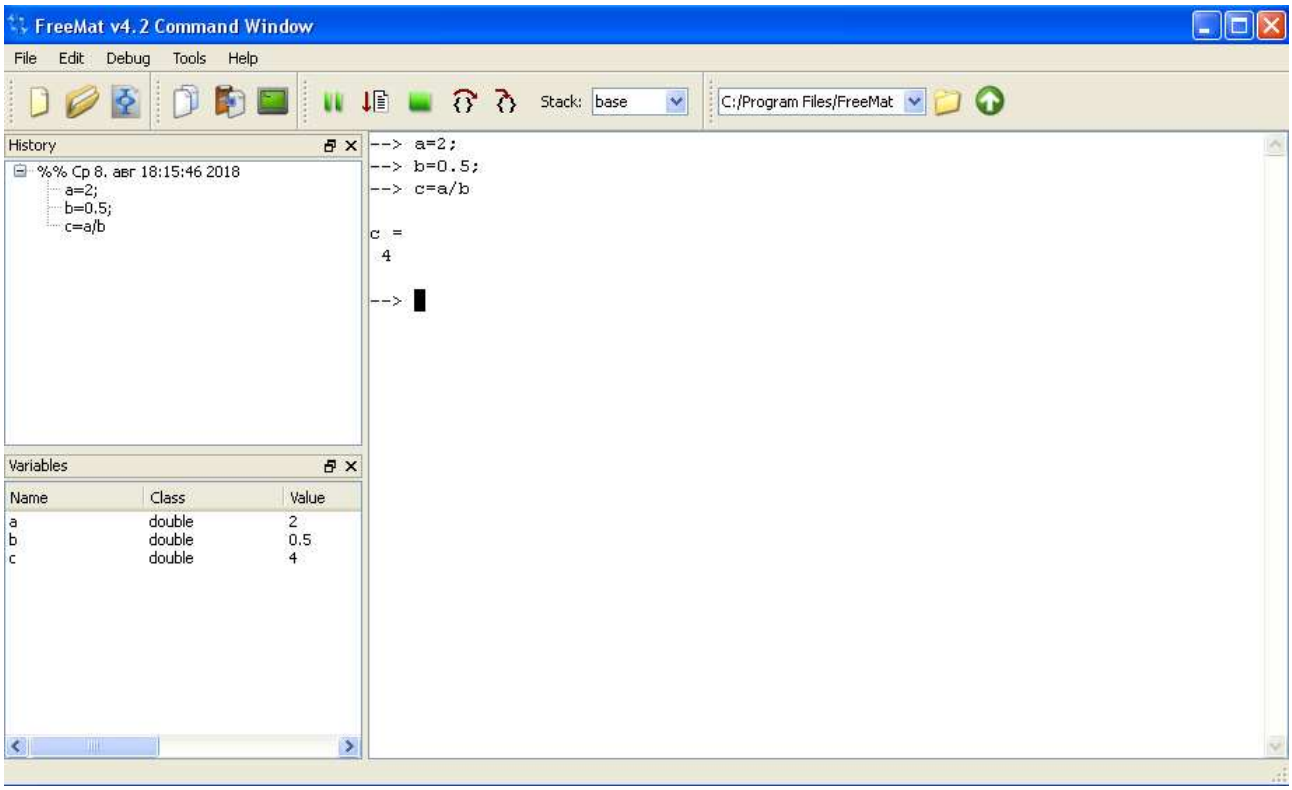


Рис. 7. Пример расчета переменной «*c*» с использованием программного режима

Полезные замечания

1. Если результат от использования операторов явно не присваивается какой-либо переменной, то он будет присвоен служебной (зарезервированной) переменной с именем «**ans**».
2. Чтобы стереть все переменные, используйте **clear all**.
3. Для проверки значений оставшихся переменных используйте **who**.
4. Для просмотра значения переменной наберите ее имя.
5. **Inf** – символизирует бесконечность (при попытке деления на ноль).
6. Стереть видимое содержимое командного окна можно при помощи **clc**.
7. **%** – означает комментарий, который игнорируются в командном окне.

Некоторые из перечисленных действий можно выполнить, используя панель управления или контекстное меню (вызывается нажатием правой кнопки мыши).

3. Арифметические операции с простыми переменными

Операторы – это неотъемлемая часть любых математических выражений. В FreeMat используются арифметические операторы двух типов:

– операторы, которые позволяют выполнять действия, соответствующие правилам арифметических расчетов (см. табл. 1);

– операторы, служащие для выполнения поэлементных операций над массивами, которые предваряются точкой (см. табл. 2).

Каждый оператор имеет свой приоритет на выполнение: наибольший приоритет имеет оператор возведения в степень, наименьший – операторы сложения и вычитания. Приоритет можно изменить, используя скобки в выражении.

Таблица 1

Операторы для арифметических расчетов

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
^	Возведение в степень

Таблица 2

Операторы поэлементных операций над массивами

Оператор	Описание
+	Унарный плюс
-	Унарный минус
.*	Поэлементное умножение массивов
./ или .\	Поэлементное деление массивов
.^	Поэлементное возведение массива в степень

Примечание к табл. 2:

– над массивами одинаковых размеров допускаются операции « + » и « - » (унарное сложение и вычитание происходит поэлементно – первый с первым, второй со вторым и так далее, в результате получают массив той же размерности);

– если используется « + » или « - » для массива и скаляра, то в результате получают массив поэлементно сложенный или за вычетом скалярной величины;

– для поэлементного перемножения массивов одинаковой размерности используется оператор « . »;*

– для поэлементного деления массивов одинаковой размерности используется:

а) правое поэлементное деление « ./ »;

б) левое поэлементное деление « .\ » .

Отличия состоят в том, что (a ./ B) → (a / B); (a .\ B) → (B/a).

– возможно поэлементное возведение массива в степень «.^», например, «mas .^ 2».

В практике программирования есть несколько устоявшихся подходов использования арифметических операций. Например, если необходимо

увеличить значение переменной **arg** на 1, то этого можно добиться следующим образом:

arg = arg + 1

При этом совершенно неважно, чему равна переменная **arg**, в любом случае ее значение будет увеличено на 1. Аналогичным образом можно выполнять увеличение или уменьшение значения переменной на любую величину.

Другим устоявшимся приемом программирования является обмен значений между двумя переменными. Например, заданы переменные **arg_a** и **arg_b**, и необходимо произвести обмен значениями между ними. Это достигается следующим образом:

```
temp = arg_a;
arg_a = arg_b;
arg_b = temp.
```

Здесь **temp** – это временная переменная, необходимая для сохранения значения переменной **arg_a**, т.к. оно затирается во второй строчке значением **arg_b**. Поэтому в третьей строке переменной **arg_b** присваивается сохраненное в **temp** значение переменной **arg_a**.

Если в результате выполнения вычислений появляется **комплексное число**, то FreeMat автоматически будет оперировать с такими числами в соответствии с арифметикой комплексных чисел. Например, при извлечении квадратного корня из -1, получим следующий результат:

c = (-1) ^ 0.5 → c = 0.0000 + 1.0000i,

здесь i – мнимая единица и представленная запись обозначает комплексное число с нулевой действительной частью и единичной мнимой.

Для того чтобы задать комплексную переменную, достаточно указать значения их действительной и мнимой частей как показано ниже:

```
c = 6 + 5i;      % комплексное число
```

и с заданными комплексными переменными также можно выполнять описанные выше арифметические операции.

При работе с комплексными числами существуют специальные функции:

real(x) – взятие действительной части комплексного числа x;
imag(x) – взятие мнимой части комплексного числа x;
abs(x) – вычисление абсолютного значения (модуля) комплексного числа x;
conj(x) – вычисление комплексно-сопряженного числа x;
angle(x) – вычисление аргумента комплексного числа x.

4. Основные функции FreeMat

Аргументами функций могут быть действительные или комплексные числа, а также массивы. Если в качестве аргумента функции задан массив, то в результате получится массив того же размера и типа, элементы которого будут равны функциям от соответствующих элементов исходного массива. В табл. 3 приведены наиболее часто используемые функции в программе FreeMat.

Таблица 3

Основные математические функции системы FreeMat

Функция	Описание
sin(x)	Синус (угол задан в радианах)
cos(x)	Косинус (угол задан в радианах)
tan(x)	Тангенс (угол задан в радианах)
cot(x)	Котангенс (угол задан в радианах)
asin(x)	Арксинус (угол задан в радианах)
acos(x)	Арккосинус (угол задан в радианах)
atan(x)	Арктангенс (угол задан в радианах)
exp(x)	Экспонента числа x (возведение в степень числа «*e*»)
log(x)	Натуральный логарифм
log10(x)	Десятичный логарифм
log2(x)	Логарифм по основанию 2
abs(x)	Модуль числа x
sqrt(x)	Вычисление квадратного корня
sum(x)	Вычисление суммы
pi	Число пи
round(x)	Округление до ближайшего целого
fix(x)	Отбрасывание дробной части числа
floor(x)	Округление до меньшего целого
ceil(x)	Округление до большего целого
rand	Генерация псевдослучайного числа с равномерным законом распределения
randn	Генерация псевдослучайного числа с нормальным законом распределения

Примечание:

имена переменных и объектов FreeMat не должны совпадать с именами функций. Проверить зарезервирована ли комбинация символов в качестве имени функции можно при помощи оператора **which** следующим образом:
which имя_функции, например:

which sin

ответ системы будет следующий: *Function sin is a built in function* (функция sin является встроенной функцией) или
which variable_name
в этом случае система ответит так: *Function variable_name is unknown!* (*Функция variable_name неизвестна!*).

5. Векторы и матрицы

Обычно в языках высокого уровня предусматривается возможность хранить значения в виде массивов, представляющих собой упорядоченную упаковку отдельных элементов. Обращение ко всему массиву осуществляется по его имени. Выборка конкретных элементов массива происходит с использованием индексации элементов массива. В FreeMat роль массивов выполняют векторы и матрицы.

В первую очередь необходимо различать: «**вектор-строку**» и «**вектор-столбец**», так как эти объекты по-разному взаимодействуют с матрицами (см. табл. 4).

Таблица 4

Способы объявления различных видов векторов

	Вектор-столбец	Вектор-строка
Способ объявления	v=[1;2;3] v=[1 2 3]'	v=[1,2,3] v=[1 2 3]
Результат	v=	v=
	1	1 2 3
	2	
	3	

Ниже показан пример задания вектора строки с именем «**a**», содержащим значения 1, 2, 3, 4:

```
a = [1 2 3 4];    % вектор-строка.
```

Для отображения того или иного элемента вектора используется следующая конструкция:

```
disp(a(1));       % отображение значения 1 -го элемента вектора;
disp(a(2));       % отображение значения 2-го элемента вектора;
disp(a(3));       % отображение значения 3-го элемента вектора;
disp(a(4));       % отображение значения 4-го элемента вектора,
```

то есть нужно указать имя вектора и в круглых скобках написать номер индекса элемента, с которым предполагается работать.

Для изменения значения, например 2-го элемента вектора на 10, достаточно записать:

a(2) = 10; % изменение значения 2-го элемента на 10.

Если же просто написать, к примеру, *a(3)*, то результатом будет присвоение переменной **ans** значения третьего элемента вектора **a,** то есть значения **3**. Аналогично можно присвоить любой переменной значение любого элемента вектора, например:

variable_n=a(3); % переменной variable_n присваивается значение 3.

Часто возникает необходимость определения общего числа элементов в векторе, то есть определения его размера. Это можно сделать, воспользовавшись функцией **length()** следующим образом:

N = length(a); % (N=4) – это число элементов массива a.

Если требуется задать вектор-столбец, то это можно сделать так:

a = [1; 2; 3; 4]; % вектор-столбец

или так:

b = [1 2 3 4]'; % вектор-столбец

при этом доступ к элементам векторов осуществляется так же, как и для векторов-строк.

Следует отметить, что векторы можно составлять не только из отдельных чисел или переменных, но и из векторов. Например, следующий фрагмент программы показывает, как можно создавать один вектор на основе другого:

a = [1 2 3 4]; % начальный вектор a = [1 2 3 4]
b = [a 5 6]; % второй вектор b = [1 2 3 4 5 6].

Здесь вектор **b** состоит из шести элементов и создан на основе вектора **a**. Используя этот прием, можно осуществлять увеличение размера векторов в процессе работы программы:

a = [a 5]; % увеличение вектора a на один элемент.

Недостатком описанного способа задания (инициализации) векторов является сложность определения векторов больших размеров, состоящих, например, из 100 или 1000 элементов. Чтобы решить данную задачу, в FreeMat существуют функции инициализации векторов нулями, единицами или случайными значениями:

```
a1 = zeros(1, 100);      % вектор-строка из 100 элементов с нулевыми
                         % значениями;
a2 = zeros(100, 1);      % вектор-столбец из 100 элементов с нулевыми
                         % значениями;
a3 = ones(1, 1000);      % вектор-строка из 1000 элементов с единичными
                         % значениями;
a4 = ones(1000, 1);      % вектор-столбец из 1000 элементов с
                         % единичными значениями;
a5 = rand(1, 1000);      % вектор-строка из 1000 элементов со
                         % случайными значениями;
a6 = rand(1000, 1);      % вектор-столбец из 1000 элементов со
                         % случайными значениями.
```

Равномерно распределенный значениями вектор, состоящий из определенного числа значений, например от -5 до 5 в 100 элементов, можно задать так:

a7 = linspace (-5,5,100).

Вектор с определенным шагом значений, например, от начального элемента, равного нулю, до последнего, равного «Пи», можно получить в случае шага, например, равного 0,01 при помощи следующего оператора:

a8 = 0:0.01:pi.

Матрицы в FreeMat задаются аналогично векторам с той лишь разницей, что указываются обе размерности. Приведем пример инициализации единичной матрицы размером *3x3*:

```
E = [1 0 0;0 1 0;0 0 1];      % единичная матрица 3x3
или
E = [1 0 0
0 1 0
0 0 1];                       % единичная матрица 3x3.
```

Аналогичным образом можно задавать любые другие матрицы, а также использовать приведенные выше функции **zeros()**, **ones()** и **rand()**, например:

```
A1 = zeros(10,10);   % нулевая матрица 10x10 элементов или
A2 = zeros(10);      % нулевая матрица 10x10 элементов;
A3 = ones(5);        % матрица 5x5, состоящая из единиц;
A4 = rand(100);      % матрица 100x100, из случайных чисел.
```

Для доступа к элементам матрицы применяется такой же синтаксис, как и для векторов, но с указанием строки и столбца, где находится требуемый элемент:

```
A = [1 2 3;4 5 6;7 8 9];   % матрица 3x3
disp(A(2,1));              % вывод на экран элемента, стоящего во
                           % второй строке первого столбца, т.е. 4.
```

disp(A(1,2)); *% вывод на экран элемента, стоящего в первой % строке второго столбца, т.е. 2.*

Также возможны операции выделения указанной части матрицы, например:

B1 = A(:,1); *% B1 = [1; 4; 7] – выделение первого столбца*
B2 = A(2,:); *% B2 = [4 5 6] – выделение второй строки*
B3 = A(1:2,2:3); *% B3 = [2 3;5 6] – выделение первых двух % строк из 2-го и 3-го столбцов матрицы A.*

Размерность любой матрицы или вектора в FreeMat можно определить с помощью функции **size()**, которая возвращает число строк и столбцов переменной, указанной в качестве аргумента:

a = 5; *% переменная a*
A = [1 2 3]; *% вектор-строка*
B = [1 2 3; 4 5 6]; *% матрица 2x3*

size(a); *% результат выглядит так: ans 1 1*
size(A); *% теперь так: ans 1 3*
size(B); *% и в этом случае так: ans 2 3.*

6. Операции над матрицами и векторами

В системе FreeMat достаточно просто выполняются математические операции над матрицами и векторами. Рассмотрим сначала простые операции сложения и умножения матриц и векторов.

Пусть даны два вектора:

a = [1 2 3 4 5]; *% вектор-строка*
b = [1; 1; 1; 1; 1]; *% вектор-столбец,*

тогда умножение этих двух векторов можно записать так:

*c = a*b;* *% c=1+2+3+4+5=15*
*d = b*a;* *% d – матрица 5x5 элементов.*

В соответствии с операциями над векторами, умножение вектор строки на вектор-столбец дает число, а умножение вектор столбца на вектор строку дает двумерную матрицу, что и является результатом вычислений в приведенном примере, то есть:

$$c = \sum_{i-1}^{5} a_i b_i = 1 + 2 + 3 + 4 + 5 = 15$$

$$d = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

Сложение и вычитание двух векторов записывается так:

```
a1 = [1 2 3 4 5];
a2 = [5 4 3 2 1];
c = a1 + a2;            %  c = [1+5 2+4 3+3 4+2 5+1]
c1 = a2 –a1;            %  c1= [5-1 4-2 3-3 2-4 1-5].
```

Следует обратить внимание, что операции сложения и вычитания можно выполнять между двумя векторами-столбцами или двумя векторами-строками. Иначе FreeMat выдаст сообщение об ошибке, т.к. разнотипные векторы складывать нельзя. Так обстоит дело со всеми недопустимыми арифметическими операциями: в случае невозможности их вычисления система FreeMat сообщит об ошибке, и выполнение программы будет завершено на соответствующей строке.

Аналогичным образом выполняются операции умножения и сложения между матрицами:

```
A = [1 2 3; 4 5 6; 7 8 9]
A =
 1 2 3
 4 5 6
 7 8 9
B = ones(3)
B =
 1 1 1
 1 1 1
 1 1 1
C = A+B;     % сложение двух матриц одинакового размера
C =
  2  3  4
  5  6  7
  8  9 10
D = A+5;     % сложение матрицы и числа
D =
```

```
 6  7  8
 9 10 11
12 13 14
E = A*B;      % умножение матрицы A на B
E =
 6  6  6
15 15 15
24 24 24
F = B*A;      % умножение матрицы B на A
F =
12 15 18
12 15 18
12 15 18
G = 5*A;      % умножение матрицы на число
G =
 5 10 15
20 25 30
35 40 45
```

Операции транспонирования матриц и векторов записываются следующим образом:

```
a = [1 1 1];      % вектор-строка
b = a';           % вектор-столбец, образованный транспонированием
                  % вектора-строки a.
```

То есть в результате получаем: **b = [1; 1; 1]**.

```
A = [1 2 3; 4 5 6; 7 8 9];        % матрица 3x3 элемента
B = a*A;                          % B = [12 15 18] – вектор-строка
C = A*b;                          % C = [6; 15; 24] – вектор-столбец
D = a*A*a';             % D = 45 – число, сумма элементов матрицы A
E = A';                 % E – транспонированная матрица A
     E =
    1 4 7
    2 5 8
    3 6 9
```

Далее рассмотрим операции вычисления обратной матрицы на примере квадратной матрицы 2x2:

```
F = [1 2; 3 4];                   % матрица 2x2 элемента
G= inv(F);                        % F – обратная матрица F
     G =
  -2.0000    1.0000
   1.5000   -0.5000
```

```
J = F ^ -1;                    % J – обратная матрица F
      J =
  -2.0000    1.0000
   1.5000   -0.5000
```

Из приведенных примеров видно, что операция транспонирования матриц и векторов обозначается символом ' (апостроф), который ставится после имени вектора или матрицы, а вычисление обратной матрицы можно выполнить путем вызова функции **inv()** или возводя матрицу **в степень -1**. Результат в обоих случаях будет одинаковым, а два способа вычисления предусмотрены для удобства использования при реализации различных алгоритмов.

Скалярное произведение двух векторов **a** и **b** возвращает функция **dot**, а векторное **– cross**:

```
s=dot(a,b);        % скалярное произведение векторов a и b
s =
 3
c=cross(a,b)       % векторное произведение
c =
0 0 0
```

Результатом скалярного произведения векторов является число, а векторного произведения – вектор. Векторное произведение определено только для векторов из трех элементов. При вычислении скалярного и векторного произведений функциями cross и dot необязательно следить за тем, чтобы оба вектора были либо столбцами, либо строками. Результат получается верный, например, при обращении:

```
c=cross(a,b')
```

только ***c*** становится вектор строкой.

FreeMat поддерживает поэлементные операции с векторами. Наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения « .* » (точка со звездочкой). Данная операция применяется к векторам одинаковой длины и приводит к вектору той же длины, что исходные, элементы которого равны произведениям соответствующих элементов исходных векторов. Например, для векторов **a** и **b**, введенных выше, поэлементное умножение дает следующий результат:

```
c=a.*b'
c =
 1 1 1
```

однако следующая операция:

```
c=a.*b
```

приведет к ошибке.

Приведем более наглядный пример:

```
aa=[2 4 6]; bb=[3 5 7];
cc=aa.*bb
cc =
  6 20 42
```

Аналогичным образом работает поэлементное деление « ./ » (точка с косой чертой). Кроме того, операция « .\ » (точка с обратной косой чертой) осуществляет обратное поэлементное деление, то есть выражения a./b и b.\a эквивалентны:

```
cc=aa./bb
cc =
   0.6667    0.8000    0.8571
cc=bb.\aa
cc =
   0.6667    0.8000    0.8571
```

Возведение элементов вектора **a** в степени, равные соответствующим элементам *b*, производится с использованием « .^ »:

```
cc=aa.^bb
cc =
    8  1024 279936
```

Для транспонирования векторов строк или векторов столбцов предназначено сочетание « .' » (точка с апострофом) и « ' » (апостроф):

```
cc=aa'
cc =
 2
 4
 6
cc=aa.'
cc =
 2
 4
 6
```

Операции ' и .' для вещественных векторов приводят к одинаковым результатам. Необязательно применять поэлементные операции при

умножении вектора на число и числа на вектор, делении вектора на число, сложении и вычитании вектора и числа. При выполнении, например, операции **a*2**, результат представляет собой вектор того же размера, что и **a**, с удвоенными элементами.

Если в процессе вычислений требуется поэлементно умножить, разделить или возвести в степень элементы вектора или матрицы, то для этого используются операторы:

.* – поэлементное умножение;
./ и .\ – поэлементные деления;
.^ – поэлементное возведение в степень.

Рассмотрим ещё несколько функций полезных при работе с векторами и матрицами.

Для поиска максимального значения элемента вектора используется функция **max()**, которая возвращает найденное максимальное значение элемента и его позицию (индекс):

```
a = [1 6 3 4];
[v, i] = max(a);          % v = 6, i = 2
```

или

```
v = max(a);               % v = 6
```

Приведенный пример показывает два разных способа вызова функции **max()**. В первом случае определяется и максимальное значение элемента, и его индекс в векторе, а во втором – только максимальное значение элемента.

В случае с матрицами данная функция определяет максимальные значения, стоящие в столбцах, как показано ниже в примере:

```
A = [4 3 5; 6 7 2; 3 1 8];
[V, I] = max(A);          % V=[6 7 8], I=[2 2 3]
V = max(A);               % V=[6 7 8].
```

Аналогичным образом работает функция **min()**, которая определяет минимальное значение элемента вектора или матрицы и его индекс.

Другой полезной функцией работы с матрицами и векторами является функция **sum()**, которая вычисляет сумму значений элементов вектора или столбцов матрицы:

```
a = [3 5 4 2 1];
s = sum(a);               % s = 3+5+4+2+1=15
```

или

```
A = [4 3 5; 6 7 2; 3 1 8];
S1 = sum(A);                % S1 = [13 11 15]
S2 = sum(sum(A));           % S2 = 39.
```

При вычислении суммы **S2** сначала вычисляется сумма значений элементов матрицы **A** по столбцам, а затем суммируются полученные значения вектора. В результате, переменная **S2** содержит сумму значений всех элементов матрицы **A**.

Для сортировки значений элементов вектора или матрицы по возрастанию или убыванию используется функция **sort()** следующим образом:

```
a = [3 5 4 2 1];
b1 = sort(a);               % b1 = [1 2 3 4 5]
b1 = sort(a,2,'ascend');    % b1 = [1 2 3 4 5]
b1 = sort(a,2,'descend);    % b1 = [5 4 3 2 1]
```

для матриц:

```
A = [4 3 5; 6 7 2; 3 1 8]
A =
 4 3 5
 6 7 2
 3 1 8
B1 = sort(A)
B1 =
 3 1 2
 4 3 5
 6 7 8
B1 = sort(A,1,' ascend')
B1 =
 3 1 2
 4 3 5
 6 7 8
B1 = sort(A,1,'descend')
B1 =
 6 7 8
 4 3 5
 3 1 2
```

Во многих практических задачах часто требуется найти определенный элемент в векторе или матрице. Это можно выполнить с помощью функции **find()**, которая в качестве аргумента принимает условие, в соответствии с которым и находятся требуемые элементы, например для вектора:

```
a = [3 5 4 2 1];
b1 = find(a==2);          % b1 = 4 – индекс элемента 2
b2 = find(a~=2);          % b2 = [1 2 3 5] – индексы элементов без 2
b3 = find(a > 3);         % b3 = [2 3] – индексы элементов больше 3.
```

Для матрицы результатом уже будет вектор-столбец, содержащий индексы элементов матрицы, удовлетворяющих условию:

```
A = [4 3 5; 6 7 2; 3 1 8]
A =
 4 3 5
 6 7 2
 3 1 8
b3 = find(A > 3)
b3 =
 1
 2
 5
 7
 9
```

В приведенном примере символ **==** означает проверку на равенство, а символ **~=** выполняет проверку на неравенство значений элементов вектора **a**.

Это операторы отношения, позволяющие строить логические выражения. В табл. 5 приведены наиболее часто используемые операторы отношения FreeMat.

Таблица 5

Логические выражения (выражения условия) FreeMat

Логическое выражение	Результат выполнения
a < b	Истинно, если переменная «*a*» меньше переменной «*b*» и ложно в противном случае
a > b	Истинно, если переменная «*a*» больше переменной «*b*» и ложно в противном случае
a == b	Истинно, если переменная «*a*» равна переменной «*b*» и ложно в противном случае
a <= b	Истинно, если переменная «*a*» меньше либо равна переменной «*b*» и ложно в противном случае
a >= b	Истинно, если переменная «*a*» больше либо равна переменной «*b*» и ложно в противном случае
a ~= b	Истинно, если переменная «*a*» не равна переменной «*b*» и ложно в противном случае

Кроме того, в пакет FreeMat включены логические операторы, такие как:

- **&** – «логическое И» или AND, дающее ИСТИНУ при выполнении условий, объединенных этим оператором;
- | – «логическое ИЛИ» или OR, дающее истину при выполнении одного из двух условий, объединенных этим оператором;
- **~** – «логическое отрицание» или NOT, дающее истину при невыполнении обоих условий, объединенных этим оператором.

Еще одной полезной функцией работы с векторами и матрицами является функция **mean()** для вычисления среднего арифметического значения, которая работает следующим образом:

```
a = [3 5 4 2 1];
m = mean(a);                % m = 3
A = [4 3 5; 6 7 2; 3 1 8];
M1 = mean(A);               % M1 = [4.3333 3.6667 5.0000]
M2 = mean(mean(A));         % M2 = 4.3333.
```

Векторы могут быть аргументами встроенных математических функций, таких, как **sin**, **cos** и т.д. В результате получается вектор с элементами, равными значению вызываемой функции от соответствующих элементов исходного вектора, например:

```
q=sin([0 pi/2 pi])
q =
0 1.0000 0.0000
```

Однако для вычисления более сложной функции от вектора значений, скажем, выражение **f=(v*sin(v)+v^2)/(v+1)** вызовет ошибку уже при попытке умножения **v** на **sin(v)**.

Пусть **v = [7 10 11 16]**. Тогда **v** является вектором-строкой определенной длины, то есть хранится в двумерном массиве размером, в данном случае, один на четыре. Точно также представлен и **sin(v)**, следовательно, умножение при помощи звездочки (по правилу матричного умножения) лишено смысла.

Аналогичная ситуация возникает и при возведении вектора **v** в квадрат, то есть, фактически, при вычислении **v*v**. Правильная запись выражения в FreeMat требует использования поэлементных операций:

```
f=(v.*sin(v)+v.^2)./(v+1)
```

7. Работа с двумерными графиками в FreeMat

Визуализация двумерных графиков обычно осуществляется с помощью функции **plot()**. Рассмотрим работу этой функции на конкретном примере. Пусть необходимо вывести график функции синуса в диапазоне от 0 до π.

Для решения этой задачи зададим множество точек по оси (0x) в виде массива:

x = 0:0.01:pi, для которых будут отображаться значения функции синуса. Затем вычислим множество значений функции синуса в этих точках: **y = sin(x)** и выведем результат на экран оператором **plot(x,y)**. В результате получим график, показанный на рис. 8.

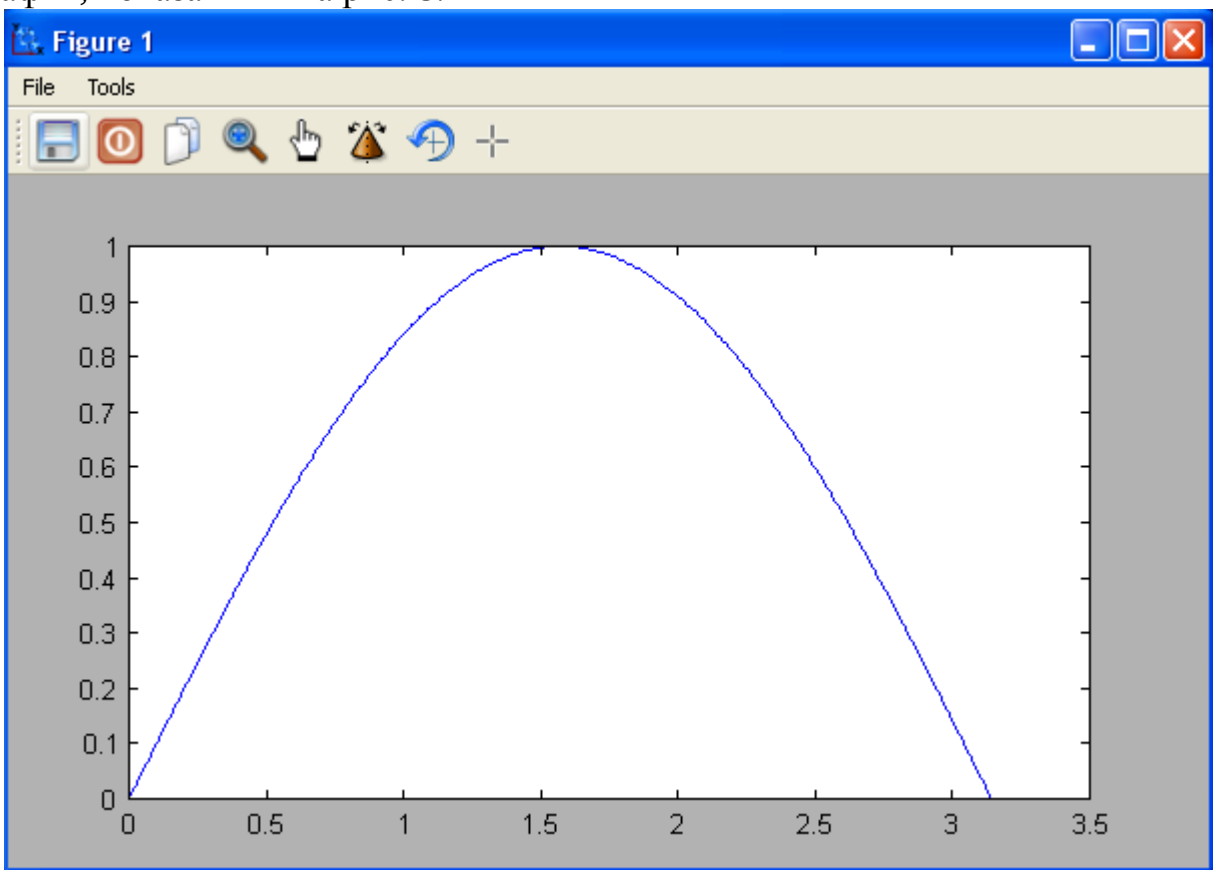


Рис. 8. Отображение функции синуса с помощью функции plot()

Функция **plot()** не только выводит точки на экран компьютера в виде графического объекта, но и осуществляет интерполяцию для придания непрерывного вида графика.

Функцию **plot()** можно записать и с одним аргументом: **plot(y)**. В этом случае функция **plot()** отображает множество точек по оси (0y), а по оси (0x) происходит автоматическая генерация множества точек с единичным шагом.

Для построения нескольких графиков в одних и тех же координатных осях функция **plot()** записывается следующим образом:

```
x = 0:0.01:pi;
y1 = sin(x);
y2 = cos(x);
plot(x,y1,x,y2)
```

Результат работы данного фрагмента программы представлен на рис. 9.

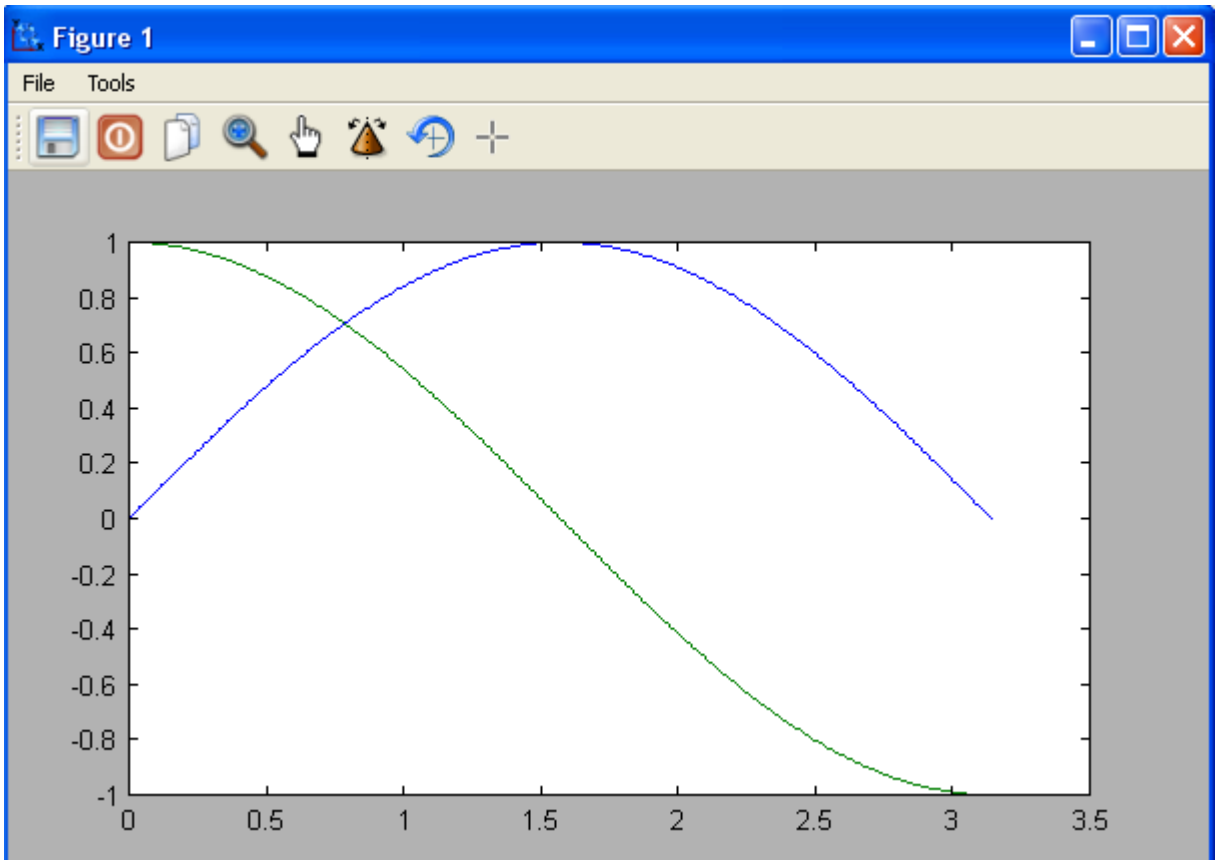


Рис. 9. Отображение двух графиков при помощи оператора plot()
в одних координатных осях

8. Оформление графиков и расширенные возможности оператора plot()

С помощью пакета FreeMat можно отображать графики с разным цветом и типом линий, показывать или скрывать сетку на графике, выполнять подпись осей и графика в целом. Для этого используют дополнительные параметры оператора **plot()**, которые записываются следующим образом:
plot(x,y,'цвет линии, тип линии, маркер точек').

Обратите внимание, что третий параметр (маркер) записывается в апострофах (' ') и имеет значения, приведенные в табл. 6-8. Маркеры записываются подряд друг за другом, например:
'ko' – на графике отображаются черными кружками точки графика;
'ko-' – выводится график черной непрерывной линией, и проставляются точки в виде кружков.

Таблица 6

Обозначение цвета линии на графике

Маркер	Цвет линии	Маркер	Цвет линии
c	Голубой	b	Синий
m	Фиолетовый	w	Белый
y	Желтый	k	Черный
r	Красный	g	Зеленый

Таблица 7

Обозначение типа линии графика

Маркер	Цвет линии
-	Непрерывная
--	Штриховая
:	Пунктирная
-.	Штрихпунктирная

Таблица 8

Обозначение типа точек графика

Маркер	Цвет линии
.	Точка
+	Плюс
*	Звездочка
o	Кружок
x	Крестик

Оформление графиков часто имеет немаловажное значение с целью составления отчетов по лабораторным работам. Для этого используются функции языка FreeMat, перечисленные в табл. 9.

Таблица 9

Функции оформления графиков

Название	Описание
grid [on, off]	Включает/выключает сетку на графике
title('заголовок графика')	Создает надпись заголовка графика
xlabel('подпись оси 0x')	Создает подпись оси 0x
ylabel('подпись оси 0y')	Создает подпись оси 0y
text(x,y, 'текст')	Создает текстовую надпись, привязанную к координатам (x,y)

Рассмотрим работу данных функций на примере следующей программы:

```
x = 0:0.1:2*pi; y = sin(x);
plot(x,y);
grid on;
title('График зависимости sin(x) ');
xlabel('Координатная ось x');
ylabel('Координатная ось y ');
text(3.05,0.16,'sin(x)')
```

Результат работы данной программы показан на рис. 10, из которого видно, каким образом работают функции создания подписей на графике, а также отображение сетки графика.

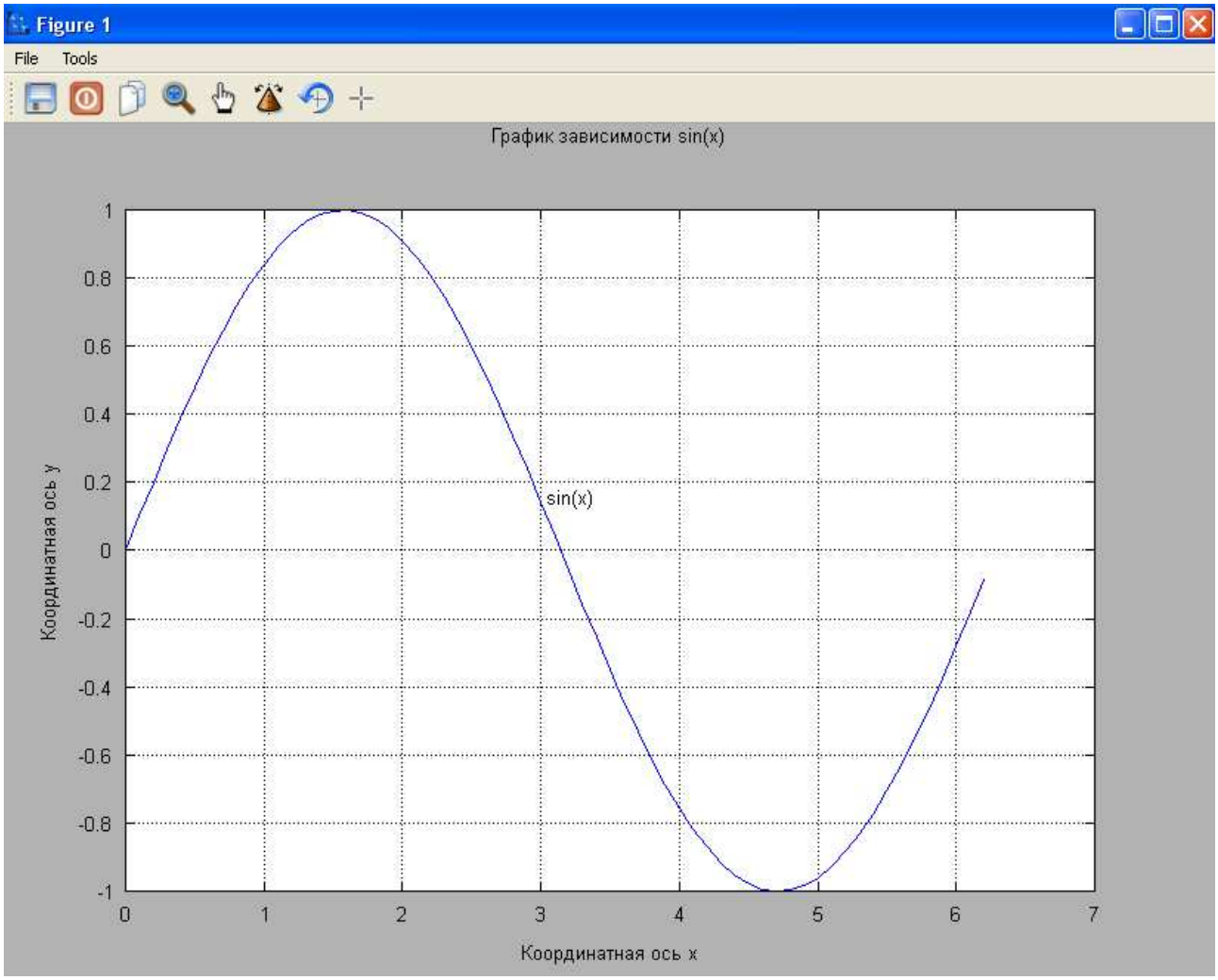


Рис. 10. Пример работы функций оформления графика

Если необходимо в одной программе вывести результаты расчетов в виде различных графиков, то можно воспользоваться оператором **figure**, который создает новое графическое окно и все последующие функции выводят

информацию именно в это графическое окно, до следующего использования оператора **figure**. Адресация графических функций и графического окна меняется с каждым разом использования оператора **figure**. Пример использования оператора **figure** совместно с графическими функциями показан ниже:

```
% Вывод графиков
plot (E,D,'k');    % Электронная плотность состояний
xlabel('Энергия E, эВ ');
ylabel('Электронная плотность состояний D(E)');
figure;
plot (V,N,'k');    % Концентрация электронов
xlabel('Напряжение стока Vd, B');
ylabel('Концентрация электронов');
figure;
plot(V,I,'k');       % ВАХ
xlabel('Напряжение стока Vd, B');
ylabel('Ток стока Id, A')
```

В этом примере опущен алгоритм расчета E, D, V и N, а показан только принцип использования графических операторов. Заметим, что если информация выводится в одно графическое окно, то использование оператора **figure** необязательно.

Неудобство работы приведенного выше фрагмента программы заключается в том, что повторный запуск программы отобразит на экране новые окна. В этом случае было бы лучше построить программу так, чтобы на экране всегда отображалось только три окна, и графики перестраивались именно в них. Этого можно достичь, если при вызове функции **figure** в качестве аргумента указывать номер графического окна, которое необходимо создать или сделать активным, если оно уже создано. Таким образом, программу можно видоизменить так:

```
% Вывод графиков
figure(1);                  % Создание графического окна с номером 1
plot (E,D,'k');             % Электронная плотность состояний
xlabel('Энергия E, эВ ');
ylabel('Электронная плотность состояний D(E)');
figure(2);                  % Создание графического окна с номером 2
plot (V,N,'k');             % Концентрация электронов
xlabel('Напряжение стока Vd, B');
ylabel('Концентрация электронов');
figure(3);                  % Создание графического окна с номером 3
```

```
plot(V,I,'k');                    % ВАХ
xlabel('Напряжение стока Vd, В');
ylabel('Ток стока Id, А')
```

В некоторых случаях большего удобства представления информации можно достичь, отображая два графика в одном графическом окне. Это достигается путем использования функции **subplot()**, имеющей следующий синтаксис:

subplot(<число строк>, <число столбцов>, <номер координатной оси>).

Рассмотрим пример отображения двух графиков друг под другом функций синуса и косинуса:

```
x1=0:0.01:2*pi; y1=sin(x1); x2=0:0.01:pi; y2=cos(x2);
figure(1);
subplot(2,1,1);      % делим окно на 2 строки и один столбец
                     % строим первую координатную ось
plot(x1,y1);         % отображение первого графика
subplot(2,1,2);      % в рамках того же деления графического окна
                     % строим вторую координатную ось
plot(x2,y2);         % отображение второго графика в новых осях.
```

Результат работы программы показан на рис. 11.

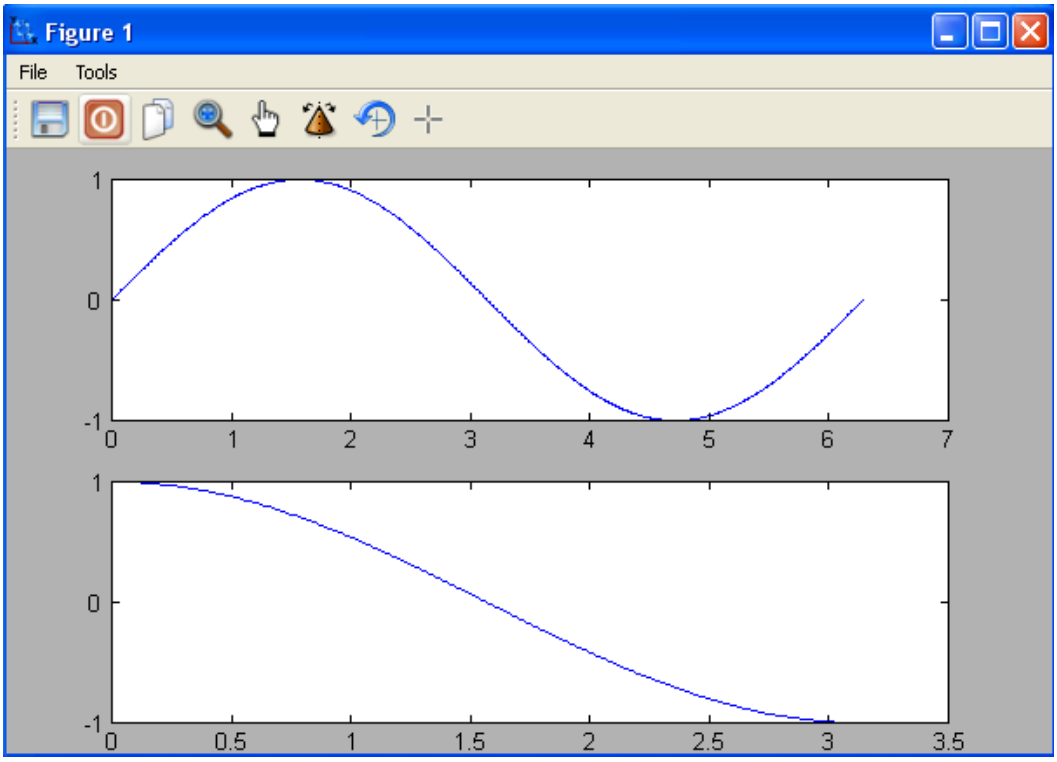


Рис. 11. Пример работы функции subplot()

Аналогичным образом можно выводить любое количество графиков в графическое окно, располагая их по аналогии с таблицей.

Любое графическое окно имеет свое собственное меню и панель управления для работы с графиками (см. табл. 10).

Таблица 10

Элементы панели управления графического окна программы FreeMat

Элемент панели управления	Описание элемента управления
	Сохранить участок в графическом окне в виде графического файла
	Закрыть графическое окно
	Создать копию окна
	Масштабирование по осям при помощи мыши
	Перемещение области построения с помощью мыши
	Позволяет вращать рисунок по оси
	Позволяет вращать рисунок по оси «x» и по оси «y»
	Дает привязку к координатам

9. Работа с трехмерными графиками в FreeMat

Рассмотрим общие принципы построения графиков функций двух переменных. Графики функций двух переменных представляют собой части поверхностей, как бы нависающие над областями определения функций. Пусть в точке с координатами **x1**, **y1** вычислено значение функции **z=f(x,y)** и оно равно **z1**. В некоторой другой точке (то есть при другом значении аргументов) **x2** и **y2** вычисляют значение функции **z2**. Продолжая этот процесс, получают массив (набор) точек **(x1,y1,z1)**, **(x2,y2,z2)**, ... **(xN,yN,zN)** в количестве **N** штук, расположенных в трехмерном пространстве. Специальные функции системы FreeMat проводят через эти точки гладкие поверхности и отображают их проекции на плоский дисплей компьютера.

Чаще всего точки аргументов расположены в области определения функции регулярно в виде прямоугольной сетки. Такая сетка точек порождает две матрицы одной и той же структуры: первая матрица содержит значения первых координат этих точек (**x – координат**), а вторая матрица содержит значения вторых координат (**y – координат**). Обозначим первую матрицу как **x**,

а вторую – **y**. Есть еще и третья матрица – матрица значений функции **z=f(x,y)** при этих аргументах. Эту матрицу обозначим буквой **z**.

Простейшей функцией построения графика функции двух переменных в системе FreeMat является функция **plot3 (x, y, z),** где **x**, **y** и **z** – матрицы одинаковых размеров.

В системе FreeMat имеется специальная функция для получения двумерных массивов **X** и **Y** по одномерным массивам **x**, **y**. Пусть по оси **x** задан диапазон значений в виде вектора **x = -2 : 0.1 : 2**, а по оси **y → y = -1 : 0.1 : 1**. Для получения матриц **X** и **Y**, представляющих первые и вторые координаты, получающейся прямоугольной сетки точек, используют специальную функцию системы FreeMat: **[X, Y] = meshgrid(x, y)**.

Эта функция формирует два двумерных массива на основе двух одномерных массивов (векторов). На прямоугольной сетке точек вычисляем значения функции, например, такой: **Z = exp(- X.^2 - Y.^2)** и далее, применяя описанную выше функцию **plot3**, получаем следующее изображение трехмерного графика этой функции (рис. 12).

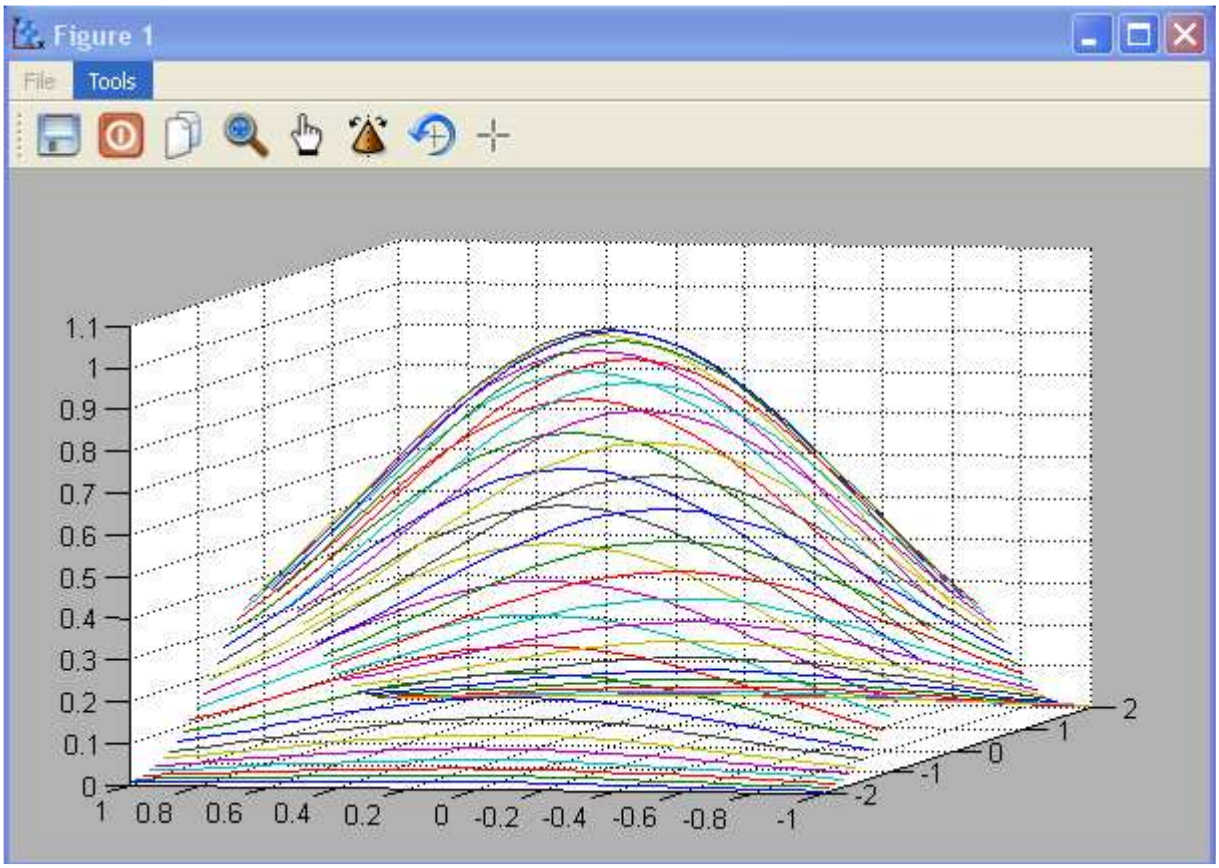


Рис. 12. Пример использования функции plot3() при построении функции двух переменных

Программный код при этом выглядит следующим образом:

```
x = -2 : 0.1 : 2;
y = -1 : 0.1 : 1;
[ X, Y] = meshgrid( x, y );
Z = exp( - X.^2 - Y.^2 );
plot3 (X, Y, Z)
```

Из этого рисунка видно, что функция **plot3()** строит график в виде набора линий в пространстве, каждая из которых является сечением трехмерной поверхности плоскостями, параллельными плоскости (**y0z**).

Для построения трехмерных линий, задаваемых параметрически, можно применить следующий программный код:

```
t = linspace (0,5*pi,200);
x = cos(t);
y = sin(t);
z = t;
plot3(x,y,z)
```

Результат показан на рис. 13.

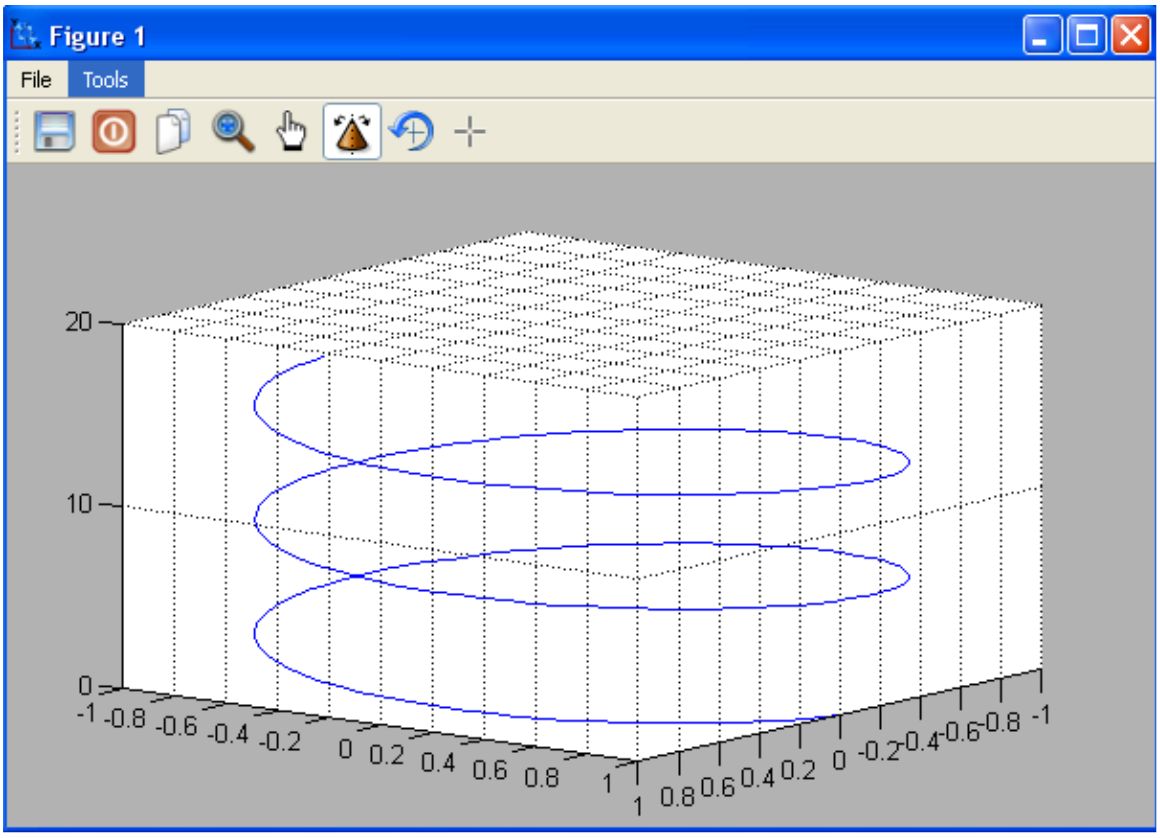


Рис. 13. Пример использования функции plot3() при построении трехмерных линий

В системе FreeMat предусмотрена функция визуализации непрерывной поверхности в трехмерных осях **surf()**. Выполним следующий программный код:

```
x = -2 : 0.1 : 2;
y = -1 : 0.1 : 1;
[ X, Y] = meshgrid( x, y );
Z = exp( - X.^2 - Y.^2 );
surf (X, Y, Z)
```

В результате получается график, представленный на рис. 14.

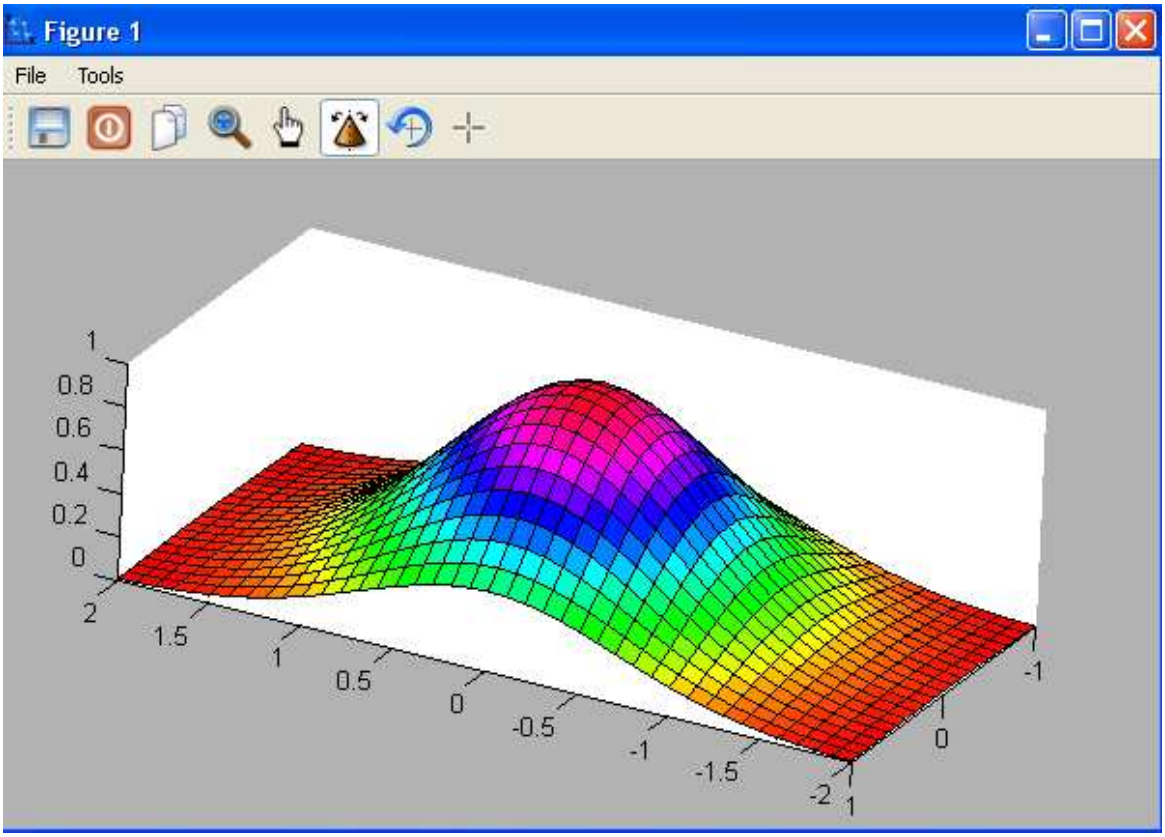


Рис. 14. Результат работы функции surf()

10. Основы программирования в среде FreeMat. Условные операторы и циклы в FreeMat

Ранее было сказано, что в среде FreeMat возможно объединять группы операторов в блоки, отделяя их между собой символом « **;** ». Кроме того, встроенный язык программирования высокого уровня (М-язык) включает в себя операторы с условием (условные операторы), позволяющие выполнять или пропускать целый блок операторов по результатам проверки некоторого

условия; а также операторы цикла, позволяющие создавать итерационные (повторяющиеся заданное количество раз) вычисления и выполнять сложные математические алгоритмы.

10.1. Условный оператор if

Оператор **if**, как правило, всегда используется совместно с оператором **end**, образуя единую конструкцию, внутри которой располагаются последовательности команд (блоки), причем конкретный блок команд выполняется после проверки условия на его выполнение. Данную конструкцию (**if … end**) часто называют операторами ветвления программного кода, так как действительно предоставляется возможность выполнения набора команд, которые будут исполняться только при условии, что значение параметра «выражение/условие» соответствует значению «истина» (т.е. верно), в противном случае (выражение/условие – неверно) программа пропускает этот набор команд.

В простом случае синтаксис оператора **if** имеет вид:

```
If <выражение/условие>      заголовок оператора и условие
   <операторы>;             если условие истинно, то выполняются
                            операторы
end

% эти строчки являются частью комментария в программе, а строки
жирным выше в реальной программе следует удалить.
```

Примечание:

1. *Символ « **%** » – означает оператор комментария. Он не сказывается на ходе выполнения программы и служит только для информативных целей.*
2. *Операторы* **if**, **end**, **else**, **switch**, **case**, **while**, **for** *(см. далее) являются зарезервированными «словами», ни одна из переменных и констант программы не должны иметь таких комбинаций символов и после их, в случае использования программного режима, символ « ; » не ставится.*

В качестве <выражения/условия> в конструкции **if … end** могут использоваться достаточно сложные логические выражения, объединенные логическими операторами, однако чаще всего используются простейшие логические выражения, приведенные в табл. 5.

Полная конструкция оператора **if** может содержать несколько блоков операторов, выполнение каждого из которых осуществляется в результате проверки условия на их выполнение:

```
if <выражение/условие>      заголовок оператора и условие
   <операторы 1> ;          если условие истинно, то выполняются
```

операторы 1
else
<операторы 2> ; **если условие ложно, то выполняются операторы 2**
end.

10.2. Условный оператор switch

Оператор **switch** также является оператором ветвления программного кода, с помощью которого обеспечивается многонаправленное ветвление программы. Среди некоторого набора альтернативных вариантов этот оператор позволяет сделать выбор дальнейшего хода выполнения программы.

Оператор **switch** действует следующим образом. Значение выражения будет последовательно сравниваться со всеми константами выбора из заданного списка (**case**), и при совпадении с одним из условий выбора будет выполняться последовательность операторов, пока не встретится оператор **break**. Ниже приведена общая конструкция оператора **switch**:

switch (выражение) {
case константа 1:
последовательность операторов
break;
case константа 2:
последовательность операторов
break;
case константа 3:
последовательность операторов
break;
...
…
default:
последовательность операторов, выполняемая, если совпадений нет
break;
}

Пакет FreeMat имеет два оператора цикла: **while** и **for**. С их помощью выполняется программирование рекуррентных алгоритмов, подсчета суммы ряда, перебора элементов массива и многое другое.

10.3. Оператор цикла while

В простом случае цикл создается с помощью оператора **while**, который имеет следующий синтаксис:

while *<выражение/условие>* **заголовок оператора цикла**
<операторы> **блок операторов, которые составляют тело цикла**
end

Здесь <выражение/условие> представляет собой условие (см. табл. 10), и цикл **while** будет работать до тех пор, пока это условие истинно (т.е. выполняется). Если условие изначально будет ложным, то операторы, входящие в цикл, вообще не будут выполняться.

10.4. Оператор цикла for

Иногда в программе необходимо повторять определенные действия. В том случае, если заранее известно число повторений, используется структура, которая называется циклом со счетчиком (или циклом **for**).

Синтаксис оператора цикла **for** имеет следующий вид:

for <счетчик> = <начальное значение>:<шаг>:<конечное значение>
<операторы цикла>
end

В этой конструкции *<счетчик>* – любая незадействованная в другой части программы переменная, которой автоматически будут присваиваться значения от *<начального значения>* до *<конечного значения>* с шагом *<шаг>*, например:

for i=1:1:100 … end. Заметим, что при выходе из цикла, в данном случае значение переменной « i » будет 101, то есть *<начальное значение>* плюс *<шаг>*.

11. Применение матриц в квантовой механике

11.1. Уравнение Шрёдингера

Волновое уравнение Шрёдингера – это дифференциальное уравнение в частных производных:

$$-\frac{\hbar^2}{2m}\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)+U(x,y,z)\psi=i\hbar\frac{\partial\psi}{\partial t}$$

или в предположении расчета энергетических уровней в поле произвольного потенциала $U(\vec{r})$:

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + U(\vec{r})\right)\psi = i\hbar\frac{\partial\psi}{\partial t},$$

где Ψ – функция времени и трех пространственных координат *(x, y, z)*, $i = \sqrt{-1}$, $\hbar$ – редуцированная постоянная Планка.

Ограничимся пока упрощенной версией уравнения – в одном измерении, и опустим зависимость от времени:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} + U(x)\psi(x) = E\psi(x).$$

Вспомним составляющие данного уравнения. Функция Ψ связана с информацией относительно расположения электрона. Функция *U(x)* представляет любое воздействие окружения на электрон. Когда она равна нулю, предполагают, что электрон является свободным, но как только электрон приближается к ядру и оказывается связанным с атомом, функция *U(x)* перестает быть равной нулю и оценить ее можно, предположив электрическую взаимосвязь электрона с протонами, то есть:

$$U(x) = -k_0\frac{Ze^2}{x},$$

где
Z – число протонов данного атома;
k_0 – коэффициент из закона Кулона;
x – расстояние электрона от ядра.

Введем это выражение в уравнение Шрёдингера:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} - k_0\frac{Ze^2}{x} = E\psi(x).$$

Характер внешнего воздействия на электрон обратно пропорционален расстоянию (также см. рис. 15):

$$U(x) = -(k_0 Ze^2)\frac{1}{x} = -Const\frac{1}{x}.$$

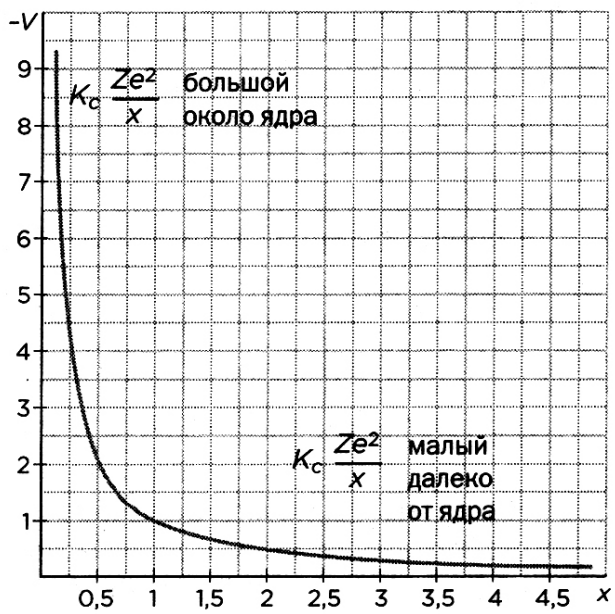


Рис. 15. Характер внешнего воздействия на электрон в зависимости от расстояния

График кривой показывает, что U оказывается принципиальным в уравнении, когда значение x мало (когда электрон находится около ядра). Соответственно, электрон будет двигаться свободно при больших расстояниях, и составляющая U будет вносить малый вклад.

Действие U, связывающее электроны с ядром, равносильно тому, как фиксируется струна музыкального инструмента, по аналогии с наложением условий на непрерывную функцию колебаний с целью получения стоячих волн (аналога квантования в макромире).

Представим себе колеблющуюся струну музыкального инструмента. Для определенности возьмем ноту «До» малой октавы и рассмотрим ее гармоники (рис. 16):

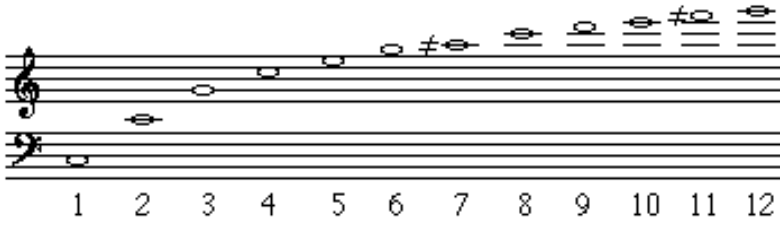


Рис. 16. Условное обозначение гармоник ноты «До» малой октавы на нотном стане: 1 – фундаментальная (первая гармоника), часто называемая также тоникой; 2 – вторая гармоника; 3 – третья гармоника и т.д.

Следующее уравнение описывает поведение струны после прикосновения и извлечения, в данном случае ноты «До» малой октавы:

$$\frac{\partial^2 a}{\partial t^2} = \frac{T}{\rho}\frac{\partial^2 a}{\partial x^2},$$

где ρ и T – постоянные (линейная плотность струны и сила, воздействующая на нее) a – пространственная и временная функция, соответствующая вертикальному расстоянию, отделяющему каждую точку струны от горизонтальной плоскости (см. рис. 17).

Уравнение струны допускает бесконечное множество решений. Некоторые из них приемлемы для математиков, но теряют физический смысл и потому отбрасываются; другие не удовлетворяют некоторым дополнительным условиям, к примеру тому, что концы струны никогда не колеблются, что струна остается неподвижной до того момента, пока ее не коснутся, или пока она не приобретет определенную форму. Эти требования сокращают диапазон приемлемых решений, но они также квантуют значение частоты v, с которой колеблется струна.

Так при фиксации струны волны останавливаются между двумя краями, и v становится дискретной величиной. Диапазон ее значений кратен фундаментальной частоте ν1 (тонике).

В общем случае можно записать:

$$\upsilon_n = \frac{n}{2L}\sqrt{\frac{T}{\rho}} \text{ или } \upsilon_n = n\upsilon_1$$

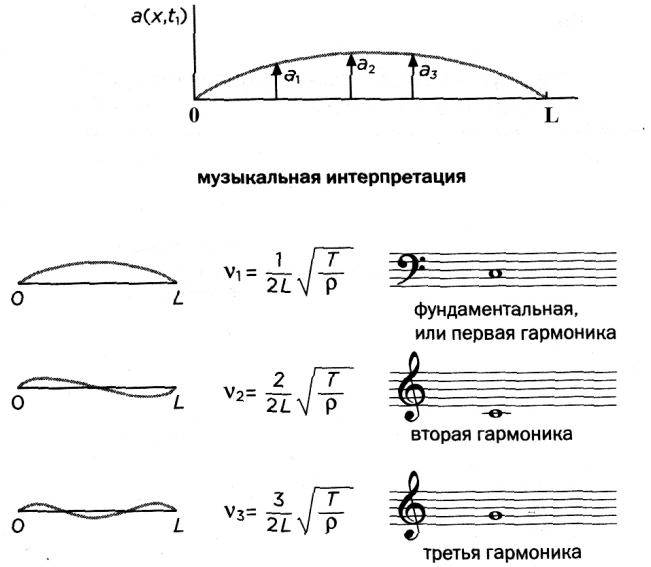


Рис. 17. Демонстрация принципа частотного квантования при возникновении стоячих волн музыкальной струны

Эти колебания называются стоячими волнами: в каждой их точке колебания происходят с той же частотой, что и у встречных волн.

Таким образом, струна оказывается разделенной на равные сегменты узлами стоячей волны, при этом оставшаяся часть струны колеблется. Узлы первой – фундаментальной частоты – находятся на концах струны, для второй гармоники добавляется один узел, в середине струны, для третьей – два, делящие струну на трети и так далее.

Иными словами, мы видим описание колебаний, сделанное с помощью непрерывной функции, со своими переменными, ограничениями, и соответственно, частотами и квантами.

Стоит отметить, что между квантованием энергии уравнения Бора для атомов и уравнением частоты гармоник нет существенного различия.

Вот, что предложил Бор для расчета энергии каждой орбиты в соответствии с целыми числами n, которые, впоследствии, будут названы главными квантовыми числами:

$$E_n = -\frac{2\pi^2 me^4 k^2}{n^2 h^2} = -\frac{Const}{n^2},$$

здесь m – масса электрона; e – его электрический заряд; k – коэффициент пропорциональности из закона Кулона; h – постоянная Планка.

Итак, действие U накладывает ограничения на пределы распространения квантовой электронной волны, при этом существуют дополнительные условия для Ψ – она должна быть дискретна, и ее значение должно стремиться к нулю при нахождении далеко от ядра.

В тот момент, когда условия будут выполнены, энергия системы будет квантована согласно формуле Бора. Функции решения «Ψ» будут вести себя так же, как стоячие волны, создавая в атоме стабильную ситуацию.

Так что же получается, материя представляет собой волны и, в общем-то, только волны. Вселенная состоит из колебаний, которые часто сосредотачиваются в определенных зонах пространства, создавая «иллюзию» частиц с макроскопической точки зрения.

Математики могут играть с волновыми конструктивными и деструктивными интерференциями, суммируя их и заставляя принимать почти все формы, какие только возможно, особенно форму сгустка или, говоря физическим языком, форму волнового пакета (рис. 18).

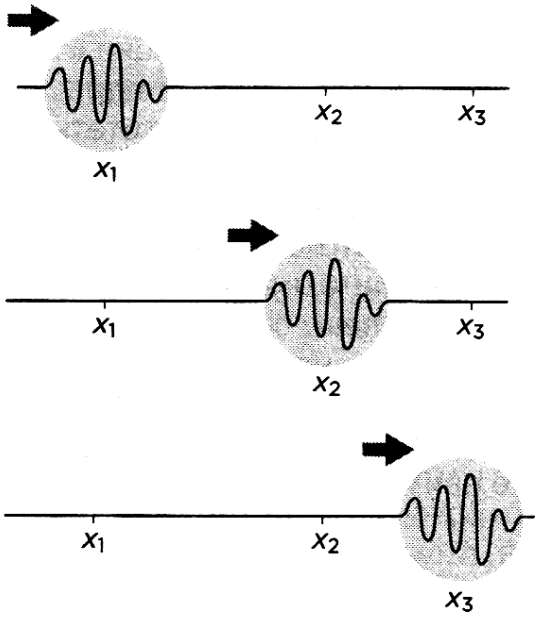


Рис. 18. Общая идея возникновения волновых пакетов

Проблема состоит в том, что практически все состояния очень нестабильны. Волны стремятся к тому, чтобы рассеяться при малейшем столкновении, а пакет рассыпается, и поведение частиц, когда они сцепляются с окружающей средой, сразу же меняется.

Однако чего-то не хватает в этой системе, и это новое будет добавлено Полем Дираком в 1928 году в виде релятивистского волнового уравнения, в котором будут учтены эффекты магнетизма.

Шрёдингер представлял электрон как электрически заряженное облако, обволакивающее атом, при этом сам электрон преобразовывался в пространственно-распределенную волну, движущуюся непрерывно, управляемую Ψ, и без всяких квантовых скачков, как предполагали Бор и Гейзенберг.

Так Эрвин Шрёдингер писал в журнале «Анналы физики» в 1926 году: «…не требует особых разъяснений то обстоятельство, что представление, по которому при квантовом переходе энергия преобразуется из одной колебательной формы в другую, значительно более удовлетворительно, чем представление о перескакивающем электроне».

Итак, сформировались две точки зрения на строение материи, с одной стороны, Вернера Гейзенберга – сторонника дискретности и корпускулярности, с другой стороны, Эрвина Шрёдингера – сторонника непрерывности и волнообразности.

11.2. Возникновение волновых пакетов

Рассмотрим уравнение Шрёдингера в трех координатах, то есть перейдем от одномерной модели:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2}+U(x)\psi(x)=E\psi(x) \qquad U(x)=-k\frac{Ze^2}{x}$$

к трехмерной:

$$-\frac{\hbar^2}{2m}\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)-k\frac{Ze^2}{\sqrt{x^2+y^2+z^2}}\psi(x,y,z)=E\psi(x,y,z)$$

.

Рассмотрим вид искомого решения.

Заметим, что $\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)$ – это вторая производная, которая должна отражать динамику изменения функциональной зависимости, в частности Ψ. Саму динамику можно визуализировать изменением касательных к функциональной зависимости. Анализируя волновое уравнение в трех координатах, можно заметить, что сумма динамики изменения касательных к Ψ в виде $\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)$ равна:

$$R_{\text{динам.}}=-\frac{2m}{\hbar^2}\cdot\left(E+k\frac{Ze^2}{\sqrt{x^2+y^2+z^2}}\right)\psi$$

.

Введем переобозначения для наглядности:

Пусть $a=-\frac{2mE}{\hbar^2}$ и $b=-\frac{2mkZe^2}{\hbar^2}$, тогда

$$R_{\text{динам.}}=\left(a+\frac{b}{\sqrt{x^2+y^2+z^2}}\right)\psi$$

.

Когда мы удаляемся от начала координат (то есть x, y и z большие величины), $\sqrt{x^2+y^2+z^2}$ приобретает намного большее значение, чем b, и второй коэффициент уменьшается с ростом пространственных координат, пока не исчезнет. Таким образом, получаем: $R_{\text{динам.}}=a\psi$.

Принимая во внимание то, что одним из условий для Ψ являлось ее стремление к нулю при удалении от ядра, то и произведение a на Ψ также будет стремиться к нулю. Тогда последнее уравнение показывает, что сумма динамики изменения трех касательных стремится к нулю с ростом расстояния: $R_{\text{динам.}} \to 0$.

С другой стороны, вполне можно предположить, что координаты могут изменяться по отдельности (неоднородная среда), тогда волны вполне могли бы проинтерферировать, чтобы образовать максимумы и минимумы при сложении.

Итак, вдалеке от протонов Ψ исчезает (касательные $\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)$ принимают горизонтальное положение) и наоборот, когда электрон находится рядом с ядром, где значения переменных x, y и z малы, сумма динамики изменения касательных выше.

К тому же, член $\frac{b}{\sqrt{x^2+y^2+z^2}}$ в выражении $R_{динам}$. стремительно растет и превышает постоянную a.

На кривой функции $\Psi(r)$ мы увидим взлеты и падения около начала координат (см. участок A на рис. 19). Затем функция теряет свою динамику и падает до нуля (см. участок C на рис. 19). Интерференционные флуктуации как раз и являются сутью появления волновых пакетов, и возникают они только в неоднородной среде, когда сама Ψ в трех пространственных координатах (а лучше в четырех: добавляется время – это так называемое четырехмерное пространство Минковского) – меняется по-разному.

Примечание: $r=\sqrt{x^2+y^2+z^2}$

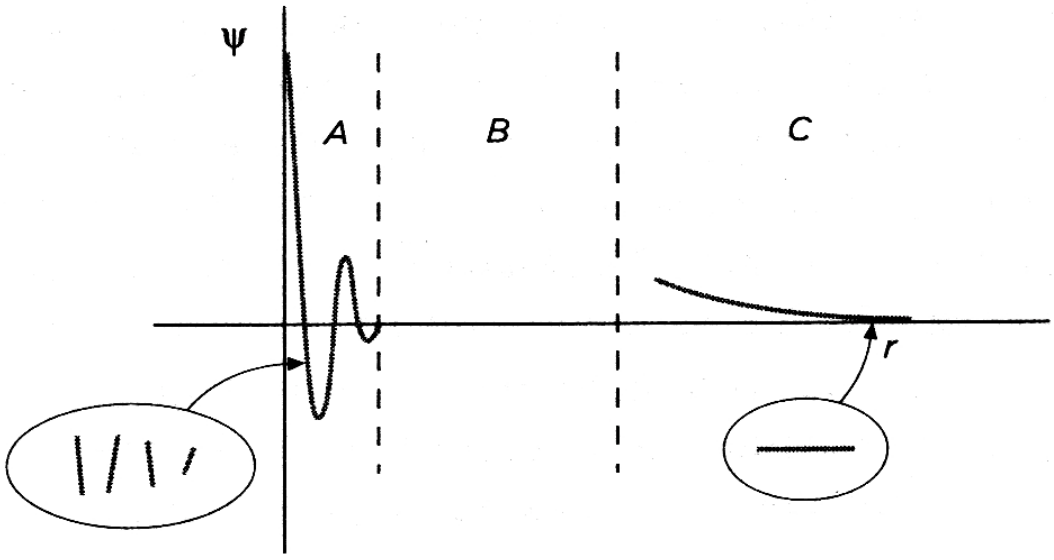


Рис. 19. Изменение динамики Ψ на различных пространственных диапазонах (в овалах показаны концептуальные расположения касательных к функции Ψ)

Примечание: Характер поведения Ψ на участке «А» может вызвать появление волновых пакетов, участок «В» остается вне досягаемости из-за ограничения рассмотрения сути вопроса.

Итак, когда атом поглощает или излучает свет, Ψ изменяется совсем как струна, тронутая гитаристом. Такую точку зрения поддерживал Шрёдингер.

Именно такая аналогия побудила Шрёдингера к выводу своего волнового уравнения. Его уравнение предполагает бесконечное число математических решений, но если ввести дополнительные условия, то один из его параметров (энергия) становится квантованным.

Итак, уравнение Шрёдингера позволяет рассчитать энергетические уровни в поле произвольного потенциала $U\left(\vec{r}\right)$. Запишем еще раз уравнение Шрёдингера:

$$i\hbar\frac{\partial\psi}{\partial t}=\left(-\frac{\hbar^2}{2m}\nabla^2+U(\vec{r})\right)\psi \quad .$$

Если $U\left(\vec{r}\right)=-Zq^2/4\pi\varepsilon_0 r$ для потенциала ядра с зарядом $+Zq$, то решения этого уравнения характеризуются тремя индексами n, l, m :

$$\psi(\vec{r},t)=\phi_{nlm}(\vec{r})\exp(-iE_n t/\hbar) \quad ,$$

где ф – фаза волны; энергия E_n зависит только от индекса n и дается выражением: $E_n=-(Z^2/n^2)E_0$.

Оказывается, что энергия электрона E играет примерно ту же роль, что и частота акустической волны. Как было сказано ранее, подобно волне в акустическом резонаторе, электронная волна в атомном «резонаторе» (параметры которого определяются потенциальной $U\left(\vec{r}\right)$ энергией) имеет свои резонансные частоты.

Если потенциал взаимодействия отсутствует (U=0), одномерное решение уравнения Шрёдингера $i\hbar\frac{\partial\psi}{\partial t}=-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2}$ можно записать в виде плоских волн, которым отвечает параболический закон дисперсии $E=\hbar^2k^2/2m$, то есть в виде:

$$\psi=\psi_0\exp(ikx)\exp(-iEt/\hbar) \quad \rightarrow \quad E=\hbar^2k^2/2m \quad .$$

Если мы введем ограничения для Ψ по аналогии с акустическим резонатором (как бы закрепим концы волны), то получим стоячие волны с дискретными значениями и квантованными значениями энергии:

$$\psi = \psi_0 \exp(ikx)\exp(-iEt/\hbar) \rightarrow k = n\pi/L \rightarrow E = \hbar^2\pi^2 n^2/2mL^2 .$$

Итак, как только мы ограничиваем волну областью резонатора, ее частота или энергия принимают дискретные значения.

11.3. Решение уравнения Шрёдингера. Метод конечных разностей

Уравнение Шрёдингера, как правило, решают с использованием численного метода. Большинство численных методов используют один общий прием – заменяют волновую функцию $\psi(\vec{r},t)$ на вектор-столбец $\psi(t)$, а дифференциальный оператор H_{op} (гамильтониан) на матрицу [H], вследствие чего уравнение Шрёдингера превращается из дифференциального уравнения в уравнение частных производных:

$$i\hbar\frac{\partial}{\partial t}\psi(\vec{r},t) = H_{op}\psi(\vec{r},t)$$

в матричное уравнение (или уравнение в конечных разностях:

$$i\hbar\frac{d}{dt}\{\Psi(t)\} = |H|\{\Psi(t)\} .$$

Наиболее простой способ такого преобразования состоит в записи уравнения на дискретной решетке (часто ее называют сеткой).

Рассмотрим для простоты случай только одного измерения и проведем дискретизацию переменной *x*, описывающей положение частицы на решетке как это показано на рис. 20: $x_n = na$.

Рис. 20. Демонстрация принципа представления непрерывной функции своими значениями в виде совокупности точек на дискретной решетке

Итак: можно представить волновую функцию как вектор-столбец, который содержит значения волновой функции только в точках решетки в момент времени *t*.

Опуская для простоты переменную t, можно записать:

$$\{\Psi_1, \Psi_2 .....\} = \{\psi(x_1)\psi(x_2) .....\}.$$

Это представление является точным лишь в пределе, когда $a \to 0$, то есть, до тех пор, пока величина *a* остается малой по сравнению с пространственным масштабом, на котором изменяется ψ.

Следующим шагом является получение матричного представления оператора Гамильтона (гамильтониана):

$$H_{op} = \frac{\hbar^2}{2m}\frac{d^2}{dx^2} + U(x).$$

В целом то, что мы попытаемся сделать, можно назвать переходом от дифференциального уравнения к разностному уравнению. Для этой цели используется стандартная процедура, называемая методом конечных разностей.

Из курса информатики известно, что для дифференциального уравнения вида: $\frac{d^2u}{dx^2} = f(x)$ (такой вид имеют также уравнения Пуассона и Лапласа, например), где функция определена на некотором интервале и *x* принадлежит конечному интервалу значений, решение этого уравнения может быть найдено в точках равномерной сетки по *x*, и если для аппроксимации производной в точке x_i использовались точки (x_{i-1}, x_i, x_{i+1}), то разностное уравнение будет иметь следующий вид:

$$f_i = \frac{-u_{i-1} + 2u_i - u_{i+1}}{a}, \text{ где "i"- номер узла сетки; "a" - шаг по сетке}.$$

Линейные функции заменяются последовательностью дискретных значений.

Для нашего случая все это будет выглядеть так:

$$\left(\frac{\partial^2\psi}{\partial x^2}\right) \Rightarrow \frac{1}{a^2}\left[\Psi(x_{n-1}) - \Psi(x_n) + \Psi(x_{n+1})\right]$$

$$U(x)\psi(x) \Rightarrow U(x_n)\Psi(x_n).$$

Вышесказанное позволяет нам перейти от общей записи уравнения Шрёдингера в матричном виде:

$$i\hbar\frac{d}{dt}\{\Psi(t)\}=\left|H\right|\{\Psi(t)\}$$

к разностной схеме:

$$i\hbar\frac{d\Psi_n}{dt}=\left|H_{op}\Psi\right|_{x=x_n}=(U_n+2t_0)\Psi_n-t_0\Psi_{n-1}-t_0\Psi_{n+1}=\sum_m\left[(U_n+2t_0)\delta_{n,m}-t_0\delta_{n,m+1}-t_0\delta_{n,m-1}\right]\Psi_m$$

Здесь $t_0=\frac{\hbar^2}{2ma^2}$; a – период сетки разбиения; $U_n=U(x_n)$; $\delta_{n,m}$ – символ Кронекера, равный единице при $n=m$ и равный нулю при n≠m.

Итак, $i\hbar\frac{d}{dt}\{\Psi(t)\}=\left|H\right|\{\Psi(t)\}$ – это запись уравнения Шрёдингера в матричном виде. Элементы матрицы, описывающей гамильтониан, определяются выражением:

$$H_{n,m}=\left[U_n+2t_0\right]\delta_{n,m}-t_0\delta_{n,m+1}-t_0\delta_{n,m-1}.$$

Такая запись означает, что матрица, осуществляющая представление оператора H_{op}, имеет вид:

	1	**2**	**3**	**…**	**N-2**	**N-1**	**N**
1	$\mathbf{2t_0+U_1}$	$\mathbf{-t_0}$	**0**	**…**	**0**	**0**	**0**
2	$\mathbf{-t_0}$	$\mathbf{2t_0+U_2}$	$\mathbf{-t_0}$	**…**	**0**	**0**	**0**
…	**…**	**…**	**…**	**…**	**…**	**…**	**…**
N-1	**0**	**0**	**0**		$\mathbf{-t_0}$	$\mathbf{2t_0+U_{N-1}}$	$\mathbf{-t_0}$
N	**0**	**0**	**0**	**…**	**0**	$\mathbf{-t_0}$	$\mathbf{2t_0+U_N}$

После представления уравнения Шрёдингера в матричной форме возникает вопрос о том, каким образом можно вычислить *{Ψ(t)}*, если задано некоторое начальное состояние *{Ψ(0)}*.

Сначала следует воспользоваться стандартной процедурой и найти собственные значения E_α и собственные вектора *{a}* матрицы *[H]*:

$$\left[H\right]\{\alpha\}=E_\alpha\{\alpha\}.$$

В принципе можно показать, что волновая функция

$$\{\Psi(t)\} = e^{-iE_\alpha t/\hbar}\{\alpha\}$$

удовлетворяет уравнению Шрёдингера в матричной форме. Поскольку уравнение Шрёдингера в матричной форме является линейным, любая суперпозиция его некоторых решений также будет решением:

$$\{\Psi(t)\} = \sum_\alpha C_\alpha e^{-iE_\alpha t/\hbar}\{\alpha\}$$

.

Для заданного начального состояния мы можем найти коэффициенты C_α. Поведение (динамику) системы можно описать в терминах собственных значений и соответствующих собственных векторов, связанных с этими уровнями матрицы *[H]*. Вот почему первым шагом при рассмотрении любой квантовой системы является запись матрицы *[H]* и поиск ее собственных значений и собственных векторов. Это несложно сделать, используя возможности FreeMat.

Контрольные вопросы по основам работы в FreeMat

1. С помощью каких команд можно задать вектор, матрицу?
2. Какими командами можно сложить два вектора (две матрицы) одинаковой размерности?
3. Какими командами можно вычислить произведение двух матриц или матрицы на вектор?
4. Какая матрица называется обратной, и какими способами она вычисляется в FreeMat?
5. Перечислите и объясните действие операторов, используемых при вычислениях с векторами и матрицами.
6. Опишите действие операций отношения.
7. Опишите действие логических операций.
8. Как построить несколько графиков в одной системе координат?
9. Как построить графики в разных подобластях одного графического окна?
10. Как изменить цвет и стиль линий на графиках?
11. Как сделать надписи на осях на полученном рисунке? Как сделать заголовок для графика?
12. Как построить график функции?
13. Как построить график поверхности?
14. Опишите основные приемы написания программ в среде FreeMat.
15. Как изменить последовательность выполнения операторов в среде FreeMat?

Далее в учебном пособии будет приведено описание нескольких расчетных лабораторных работ с использованием программы FreeMat.

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема. Использование функций и операторов программы FreeMat.

Цель работы

1. Изучить интерфейс и основные возможности программного пакета FreeMat.
2. Используя программу FreeMat, выполнить расчет переменной в соответствии с выданным вариантом задания.

Задание и ход выполнения работы

1. Задайте значения переменным, необходимые для расчета «y».
2. Рассчитайте значение переменной «y» в соответствии со своим вариантом задания (см. табл. 11).
3. В отчете по работе приведите текст программы, вычисляющей переменную «y», абсолютное значение переменной «y»и все операторы и функции, использованные для расчета.

Таблица 11

Варианты заданий к лабораторной работе № 1

Номер варианта	Параметры переменных	Вычислить в среде FreeMat
1	a=2,5; b=1,3; c=3	$y=\sqrt{\frac{a+b^3}{c}}$
2	a=1,75; b=3,8	$y=\frac{\lvert a^3+b^4\rvert}{tg(a-b)}$
3	a=1,1; b=3,6; c=2,4	$y=\frac{ab^3+c}{a^2-4ac+c^3}$
4	a=2,8; b=5,9; c=0,5	$y=\frac{\sqrt[3]{\ln a+ab^4}}{3c^2}$
5	a=3,2; b=1,2; c=1	$y=\frac{\sqrt{5}(a^3+e^{3b})}{\sin c}$
6	a=4,5; b=0,3 c=2,4	$y=\frac{ctg(a^2-\sqrt{a+b})}{\sqrt{c-b}}$

7	a=7,3;b=2,4;c=4,1	$y=\frac{(a^3-b)(a^2-c)}{\log(a-c)}$
8	a=3,1; b=1,8; c=2,2	$y=\frac{(abc)^2-\cos 2\pi}{3-2ab}$
9	a=8,1; b=5,5; c= –1,4	$y=\frac{\sin\frac{\pi}{2}+2a}{\left\|\sqrt{b}+bc\right\|}$
10	a=9; b= –2,1; c=3	$y=\frac{\left\|b\right\|+\cos\pi}{a^2+c}$
11	a=3,7; b= –2,3; c=1,2	$y=\frac{2+tg\frac{\pi}{4}}{\left\|b^3+\sqrt{a-c}\right\|}$
12	a=1,5; b=5,4 c=8	$y=\frac{e^{\pi}+3ab}{(a+c)^2\sqrt{c}}$
13	a=1,2; b=4; c=7,3	$y=\frac{\sqrt[4]{\left\|ctg3a-4ac\right\|}}{b^3+c}$
14	a=4,2; b=1,2; c=3,2	$y=\frac{\log 2a+\sqrt{(a-b)}}{3c+a}$
15	a=3,5; b=5,9; c=2,4	$y=\frac{\sqrt{3b}+(a^2+b)}{\ln(4a+c)}$

Контроль сформированных практических навыков

1. Студент должен уметь запускать программный пакет FreeMat и настраивать операционную среду работы с ним, демонстрируя сформированные компетенции преподавателю.
2. Студент должен знать интерфейсные возможности FreeMat и должен уметь демонстрировать преподавателю умение работать с различными окнами FreeMat.
3. Студент должен знать основные операторы, функции, управляющие конструкции FreeMat и уметь демонстрировать на практике их работу.

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема. Поэлементные операции с векторами в среде FreeMat.

Цель работы

1. Изучить основные возможности программного пакета FreeMat при работе с векторами, включая различные способы объявления и заполнения векторов.
2. Используя возможности программы FreeMat, выполнить расчет вектора «***c***» в соответствии с выданным вариантом задания, используя поэлементные операции, производимые с исходными векторами.

Задание и ход выполнения работы

1. Сформировать практические навыки работы с векторами в программе FreeMat.
2. Объявить и заполнить три произвольных различных вектора методом прямого набора, методом с заданным шагом и методом с определенным количеством элементов.
3. Используя возможности программного пакета FreeMat, задайте значения векторов в соответствии с выданным вариантом задания (см. табл. 12) и рассчитайте значение вектора «***c***».
4. В отчете по работе опишите способы задания и заполнения векторов в среде FreeMat, форматы операторов работы с векторами, а также содержание вектора «***c***», рассчитанного в соответствии с выданным вариантом задания.

Таблица 12

Варианты заданий к лабораторной работе № 2

Номер варианта	Параметры	Найти
1	a=[1,1;3,4;2,5] b=[6,3;1,5;2,7]	$c = \frac{a}{b} + a$
2	a=[2,3;7,1;4,5] b=[3,1;0,8;1,2]	$c = a * b - b$
3	a=[8,4;–1,2;5,1] b=[2,9;4,4;0,3]	$c = \frac{a - b}{a}$
4	a=[3,1;8,2;1,6] b=[8,8;2,2;6,2]	$c = a + \frac{b}{a}$
5	a=[5,3;2,7;7,1] b=[–1,2;3,8;4,3]	$c = a^3 - b$

6	a=[6,8;1,9;3,5] b=[7,2;5,4;2,2]	$c = \frac{a+b}{b^2}$
7	a=[–1,8;7,9;–1,4] b=[2,6;–3,7;0,1]	$c = \frac{a}{2} - 3b$
8	a=[–3,1;5,2;4,9] b=[6,5;2,4;2,9]	$c = \frac{a}{b-a}$
9	a=[1,7;4,3;6,1] b=[2,3;7,3;4,5]	$c = a^2 + \frac{b}{a}$
10	a=[2,5;3,9;0,5] b=[4,6;1,9;0,7]	$c = a * b - \frac{a}{b}$
11	a=[1,2;8,2;3,3] b=[5,9;6,3;–1,8]	$c = 2a + 3b^2$
12	a=[0,1;2,3;–5,2] b=[3,6;4,5;3,1]	$c = \frac{a^2}{2} + 5b$
13	a=[5,3;8,7;6,5] b=[4,1;3,5;3,7]	$c = b^2 + 3a$
14	a=[2,1;3,8;2,3] b=[6,4;0,5;2,7]	$c = 4a + \frac{a^2}{b}$
15	a=[–7,5;2,4;9,5] b=[2,3;2,8;4,7]	$c = \frac{a^2 + b}{2ab}$

Контроль сформированных практических навыков

1. Студент должен уметь запускать программный пакет FreeMat и настраивать операционную среду работы с ним, демонстрируя сформированные компетенции преподавателю.
2. Студент должен знать интерфейсные возможности FreeMat и уметь демонстрировать преподавателю навыки работы с различными окнами FreeMat.
3. Студент должен знать основные операторы, функции, управляющие конструкции FreeMat и уметь демонстрировать на практике их работу.
4. Студент должен уметь объявлять вектора, выбирая при этом оптимальный вариант между различными методами.
5. Студент должен уметь демонстрировать на практике использование операторов работы с векторами, определять количество элементов в векторе и выбирать любой его элемент.
6. Студент должен иметь четкое представление о специфике использования и уметь демонстрировать преподавателю работу арифметических

операторов, матричного исчисления и арифметических операторов поэлементных операций над векторами и матрицами.

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема. Работа с векторами в среде FreeMat.

Цель работы

1. Изучить основные возможности программного пакета FreeMat при работе с векторами, включая различные способы объявления и заполнения векторов.
2. Используя возможности программы FreeMat, выполнить расчеты в соответствии с заданием.

Задание и ход выполнения работы

Для заданных векторов **a** и **b**:

1. Вычислить их сумму, разность и скалярное произведение.
2. Для вектора **a** определить его максимальный и минимальный элементы.
3. Упорядочить вектор **b** по возрастанию и убыванию.
4. Переставить элементы вектора **a** в обратном порядке и записать результат в новый вектор.
5. Найти векторное произведение **a** и **b**.

Исходные данные для расчетов взять из табл. 13 в соответствии с выданным преподавателем вариантом.

Таблица 13

Варианты заданий к лабораторной работе № 3

Номер варианта	Вектор **a**	Вектор **b**
1	[0.5; 3; 6; -4.3; 1.2]	[3; 7; 7; 5.4; -2]
2	[-4.8; -1; -1; 0.7; 4]	[-1; -1.9; 7.1; -2; 6]
3	[1; -3.9; -2; 3; 2]	[2.7; -2.7; 4; 0.4 -6;]
4	[-2.4; 3.3; 0; 3; -7]	[6; 0.6; 4; -3; 7]
5	[8; -5.9; -6; 0; 6.8]	[0; 2; -1.5; 7.5; -4]
6	[5.3; 6; -7.1; 6; -4]	[7; -1.5; -9; -4.6; -2]
7	[1.2; -4; -0.8; -0.7; -2]	[-1; 2.2; 1; -4; -1.8]
8	[6.6; -5; -2.7; 8; 3.8]	[-1; 3.2; 4.2; -6; 1]

9	[-1.9; 0.4; 1; 4; -3.8]	[-8; -4; -1.4; 2.8; -2.2]
10	[9; 1.7; -3; -3.8; 7.3]	[0.6; -0.4; -6.9; -2; 1]
11	[-1.7; 1.7; 0; -3; -6.2]	[-2.3; -4; -0.2; -5; 5.5]
12	[1.7; 3.3; -6;-1.5; 2]	[1; 1; 2.7; -6.5; -6]
13	[-4; 4; -2.1; 7; 0]	[6; -0.2; 8.6; 3; -3.2]
14	[0; 2.3; -8.1; 0; 4]	[0; -1; 4.5; 9.9; 6]
15	[-6; 7.4; 5; 0; -6.9]	[0.5; -1; 3; 5.61; 4.09]

Контроль сформированных практических навыков

1. Студент должен уметь запускать программный пакет FreeMat и настраивать операционную среду работы с ним, демонстрируя сформированные компетенции преподавателю.
2. Студент должен знать интерфейсные возможности FreeMat и уметь демонстрировать преподавателю навыки работы с различными окнами FreeMat.
3. Студент должен знать основные операторы, функции, управляющие конструкции FreeMat и уметь демонстрировать на практике их работу.
4. Студент должен уметь объявлять вектора, выбирая при этом оптимальный вариант между различными методами.
5. Студент должен уметь демонстрировать на практике использование операторов работы с векторами, определять количество элементов в векторе и выбирать любой его элемент.
6. Студент должен иметь четкое представление о специфике использования и уметь демонстрировать преподавателю работу арифметических операторов, матричного исчисления и арифметических операторов поэлементных операций над векторами и матрицами.

ЛАБОРАТОРНАЯ РАБОТА № 4

Тема. Работа с матрицами в среде FreeMat.

Цель работы

1. Изучить основные возможности программного пакета FreeMat при работе с матрицами, включая различные способы их объявления и заполнения.
2. Используя возможности программы FreeMat, выполнить расчеты в соответствии с заданием.

Задание и ход выполнения работы

1) Введите матрицы:

1 вариант: $A=\begin{pmatrix} 8 & 4 & -6 \\ -2 & -4 & -6 \\ 6 & 4 & 4 \end{pmatrix}$ $B=\begin{pmatrix} 7 & 3 & 4 \\ 5 & -9 & -4 \\ 9 & -9 & -9 \end{pmatrix}$ $C=\begin{pmatrix} 10 & -7 & -7 \\ -3 & -11 & -1 \\ -1 & -9 & 4 \end{pmatrix}$

2 вариант: $A=\begin{pmatrix} -5 & 1 & -7 \\ 9 & -3 & 4 \\ -3 & 7 & 5 \end{pmatrix}$ $B=\begin{pmatrix} 7 & 1 & 3 \\ -6 & -6 & 9 \\ 3 & 5 & -6 \end{pmatrix}$ $C=\begin{pmatrix} -8 & 2 & -2 \\ -4 & -4 & 24 \\ 9 & -4 & 36 \end{pmatrix}$

3 вариант: $A=\begin{pmatrix} 6 & 8 & 6 \\ 10 & -10 & -2 \\ -2 & 6 & -10 \end{pmatrix}$ $B=\begin{pmatrix} -4 & 6 & -4 \\ 10 & 8 & 2 \\ 2 & -6 & 6 \end{pmatrix}$ $C=\begin{pmatrix} 4 & 4 & 4 \\ -8 & -2 & 6 \\ -2 & 2 & -8 \end{pmatrix}$

4 вариант: $A=\begin{pmatrix} -4 & -8 & -4 \\ 6 & -2 & -6 \\ 4 & 2 & -8 \end{pmatrix}$ $B=\begin{pmatrix} -2 & -2 & 2 \\ -8 & -6 & -8 \\ -4 & -10 & -10 \end{pmatrix}$ $C=\begin{pmatrix} -10 & 10 & -20 \\ 6 & 2 & -6 \\ 2 & 6 & 2 \end{pmatrix}$

5 вариант: $A=\begin{pmatrix} -9 & -9 & -5 \\ -4 & 7 & 5 \\ 9 & -5 & 1 \end{pmatrix}$ $B=\begin{pmatrix} -5 & 1 & -7 \\ 9 & -3 & -4 \\ -3 & 7 & 5 \end{pmatrix}$ $C=\begin{pmatrix} 3 & -11 & 5 \\ -8 & -5 & -3 \\ 3 & -1 & 5 \end{pmatrix}$

6 вариант $A=\begin{pmatrix} 5 & -7 & -11 \\ -6 & -9 & -3 \\ 3 & 5 & -5 \end{pmatrix}$ $B=\begin{pmatrix} 1 & 7 & -3 \\ 5 & -9 & -4 \\ 9 & -9 & -9 \end{pmatrix}$ $C=\begin{pmatrix} 7 & 1 & 3 \\ 3 & 5 & -6 \\ 1 & 9 & 5 \end{pmatrix}$

7 вариант $A=\begin{pmatrix} -3 & -11 & -13 \\ 7 & -9 & 6 \\ 5 & -3 & -1 \end{pmatrix}$ $B=\begin{pmatrix} -3 & 1 & -11 \\ -4 & -3 & -3 \\ 1 & -7 & -3 \end{pmatrix}$ $C=\begin{pmatrix} -7 & -7 & -5 \\ 5 & 7 & 0 \\ -3 & 7 & 1 \end{pmatrix}$

8 вариант $A=\begin{pmatrix} -5 & -3 & 5 \\ 4 & -7 & -3 \\ 1 & -11 & -1 \end{pmatrix}$ $B=\begin{pmatrix} 9 & -5 & 5 \\ -2 & 9 & -3 \\ 1 & 3 & 3 \end{pmatrix}$ $C=\begin{pmatrix} 3 & -3 & 1 \\ -9 & -5 & 8 \\ -1 & -3 & -3 \end{pmatrix}$

9 вариант $A=\begin{pmatrix} -3 & 9 & 7 \\ -6 & 5 & -9 \\ -7 & -9 & -7 \end{pmatrix}$ $B=\begin{pmatrix} -9 & -1 & -5 \\ 1 & -7 & -2 \\ 7 & 3 & -5 \end{pmatrix}$ $C=\begin{pmatrix} 3 & 3 & -1 \\ 2 & 7 & -5 \\ 7 & -7 & -5 \end{pmatrix}$

10 вариант $A=\begin{pmatrix}-1 & 5 & -1\\ -2 & 3 & -11\\ -3 & 5 & -11\end{pmatrix}$ $B=\begin{pmatrix}-1 & 9 & 7\\ -4 & 9 & -1\\ 5 & -3 & 5\end{pmatrix}$ $C=\begin{pmatrix}7 & 5 & -9\\ -8 & -5 & -1\\ 3 & -5 & 3\end{pmatrix}$

11 вариант $A=\begin{pmatrix}5 & -9 & -9\\ -6 & -9 & -5\\ -5 & -3 & -3\end{pmatrix}$ $B=\begin{pmatrix}9 & 3 & -9\\ 2 & 3 & -7\\ -7 & -7 & -7\end{pmatrix}$ $C=\begin{pmatrix}1 & -5 & 5\\ -6 & -9 & -5\\ 7 & 5 & -1\end{pmatrix}$

12 вариант $A=\begin{pmatrix}3 & -5 & 5\\ 7 & -5 & -9\\ -9 & -7 & -13\end{pmatrix}$ $B=\begin{pmatrix}-5 & -9 & 3\\ -12 & -3 & -11\\ 5 & -3 & -5\end{pmatrix}$ $C=\begin{pmatrix}-7 & 1 & 5\\ 4 & -5 & -3\\ 3 & -1 & -6\end{pmatrix}$

13 вариант $A=\begin{pmatrix}1 & 5 & 2\\ 3 & -1 & -6\\ -6 & -3 & -7\end{pmatrix}$ $B=\begin{pmatrix}-2 & -3 & -1\\ 9 & -11 & 2\\ -5 & -7 & 8\end{pmatrix}$ $C=\begin{pmatrix}-3 & -9 & -4\\ 7 & 1 & -8\\ 3 & -9 & -4\end{pmatrix}$

14 вариант $A=\begin{pmatrix}7 & 0 & -7\\ 3 & 1 & -4\\ -14 & -3 & 3\end{pmatrix}$ $B=\begin{pmatrix}-12 & 5 & -11\\ 4 & -5 & 3\\ 5 & -3 & 10\end{pmatrix}$ $C=\begin{pmatrix}-10 & -7 & -7\\ -8 & 2 & -6\\ -6 & -2 & 4\end{pmatrix}$

15 вариант $A=\begin{pmatrix}-9 & 4 & 1\\ -5 & 0 & 0\\ 2 & 4 & 8\end{pmatrix}$ $B=\begin{pmatrix}1 & 0 & 2\\ 3 & 0 & -1\\ 5 & 2 & 2\end{pmatrix}$ $C=\begin{pmatrix}0 & 0 & -1\\ -2 & 3 & 2\\ 9 & 3 & 4\end{pmatrix}$

2) Выполните следующие действия:

- A+B;
- C*A;
- B*5;
- вычислите транспонированную матрицу **C**;
- получите обратную матрицу **B**;
- возведите в квадрат матрицу **A**;
- вычислите сумму значений элементов столбцов матрицы **A**;
- найдите максимальное и минимальное значения матрицы **B**;
- отсортируйте значения элементов матрицы **C** по возрастанию и убыванию.

Контроль сформированных практических навыков

1. Студент должен уметь запускать программный пакет FreeMat и настраивать операционную среду работы с ним, демонстрируя сформированные компетенции преподавателю.
2. Студент должен знать интерфейсные возможности FreeMat и уметь демонстрировать преподавателю навыки работы с различными окнами FreeMat.
3. Студент должен знать основные операторы, функции, управляющие конструкции FreeMat и уметь демонстрировать на практике их работу.
4. Студент должен уметь объявлять матрицы, выбирая при этом оптимальный вариант между различными методами.
5. Студент должен уметь демонстрировать на практике использование операторов работы с матрицами, определять количество элементов в матрице и выбирать любой ее элемент.
6. Студент должен иметь четкое представление о специфике использования и уметь демонстрировать преподавателю работу арифметических операторов, матричного исчисления и арифметических операторов поэлементных операций над векторами и матрицами.

ЛАБОРАТОРНАЯ РАБОТА № 5

Тема. Изучение возможностей двумерной графики программы FreeMat.

Цель работы

1. Изучить основные графические возможности программы FreeMat при построении двумерных графиков.
2. Используя возможности программы FreeMat, построить графики в соответствии с выданным вариантом задания.

Задание и ход выполнения работы

Постройте два графика в одном графическом окне программы FreeMat в соответствии с выданным вариантом задания (см. табл. 14).

Таблица 14

Варианты заданий к лабораторной работе № 5

Номер варианта	Функции для построения		Область определения функции
1	$y = e^{-x^2}$	$z = arctg(x)^{1/2}$	$x \in [0;4\pi]$
2	$y = \sqrt{1+x^2}$	$z = \log_{0,5} x$	$x \in [0;1]$
3	$y = \sin(x)+1$	$z = \cos(x)$	$x \in [-\frac{\pi}{2};\frac{\pi}{2}]$
4	$y = (x+2)^2$	$z = -(x-2)^2$	$x \in [0;5]$
5	$y = \frac{2}{x}$	$z = 4x - x^2$	$x \in [0;\infty]$
6	$y = x + \sin x$	$z = -\cos x$	$x \in [0;\pi]$
7	$y = \sqrt{x}$	$z = x$	$x \in [-4;5]$
8	$y = 5x^2 + 4$	$z = 4{,}5x + 1$	$x \in [-10;29]$
9	$y = \log_3 x$	$z = \ln\left(\frac{x}{2}\right)$	$x \in [-3;9]$
10	$y = \sin x$	$z = \cos\left(\frac{4\pi}{3}\right) + 2$	$x \in [-\frac{4\pi}{3};0]$
11	$y = \frac{\sin x}{\cos x}$	$z = x$	$x \in [0;1]$
12	$y = x + 2$	$z = \sqrt{(x-2)^2}$	$x \in [0;3]$
13	$y = tg(x)$	$z = \frac{\cos x}{\sin x}$	$x \in [0;5\pi]$
14	$y = 1 - x$	$z = x^2 + 6x + 12$	$x \in [-\infty;60]$
15	$y = 3$	$z = ctg\left(\frac{x}{2}\right)$	$x \in [0;\infty]$

Контроль сформированных практических навыков

1. Студент должен уметь запускать программный пакет FreeMat и настраивать операционную среду работы с ним, демонстрируя сформированные компетенции преподавателю.
2. Студент должен знать интерфейсные возможности FreeMat и уметь демонстрировать преподавателю навыки работы с различными окнами FreeMat.
3. Студент должен знать основные операторы, функции, управляющие конструкции FreeMat и уметь демонстрировать на практике их работу.
4. Студент должен уметь задавать функциональные зависимости, используя различные возможности FreeMat, использовать на практике встроенные функции среды FreeMat и выводить функциональные зависимости в графические окна FreeMat.
5. Студент должен уметь демонстрировать на практике использование операторов работы с двумерной графикой среды FreeMat.
6. Студент должен иметь четкое представление о специфике использования и уметь демонстрировать преподавателю работу арифметических операторов, матричного исчисления и арифметических операторов поэлементных операций над векторами и матрицами.

ЛАБОРАТОРНАЯ РАБОТА № 6

Тема. Изучение возможностей трехмерной графики программы FreeMat.

Цель работы

1. Изучить основные графические возможности программы FreeMat при построении трехмерных графиков.
2. Используя возможности программы FreeMat, построить поверхность, задаваемую функцией двух переменных.

Задание и ход выполнения работы

Постройте поверхность в графическом окне FreeMat, задаваемую функцией двух переменных в соответствии с выданным вариантом задания (см. табл. 15).

Таблица 15

Варианты заданий к лабораторной работе № 6

Номер варианта	Функциональная зависимость	Область определения функции
1	$z = \ln(x^2 + y^2 - xy)$	$x, y \in [1;2]$
2	$z = \frac{x}{x^2 + y^2}$	$x, y \in [-1;1]$
3	$z = x^3 - 3xy^3$	$x, y \in [-5;5]$
4	$z = \frac{\sin(x^2 + y^2)}{x^2 + y^2}$	$x, y \in [-\pi;\pi]$
5	$z = 4x^2 + 9y^2 - 72y$	$x, y \in [-10;10]$
6	$-2x^2 + 3y^2 + 4z^2 = 0$	$x, y \in [-20;20]$
7	$2z = x^2 + y^2$	$x, y \in [-40;0]$
8	$z = \sqrt{xy}$	$x, y \in [-35;35]$
9	$z = \cos(xy)$	$x, y \in [-3;3]$
10	$z = \frac{x^2 - y}{1 + x^2 + y}$	$x, y \in [-\pi;\pi]$
11	$z = x^3 + y^3 - 5xy + \frac{1}{5}$	$x, y \in [-5;5]$
12	$z = (x^2 + y^2)^2 - x^2 + y^2$	$x, y \in [-1;1]$
13	$z = \cos(x^2) - \sin(x^2) - 1$	$x, y \in [0;10]$
14	$z = -\ln\left(\sqrt{(x+1)^2 + y^2}\right) - \ln\left(\sqrt{(x-1)^2 + y^2}\right)$	$x, y \in [-6;6]$
15	$z = x^3 - 3xy^2 + y^2$	$x, y \in [-10;10]$

Контроль сформированных практических навыков

1. Студент должен уметь запускать программный пакет FreeMat и настраивать операционную среду работы с ним, демонстрируя сформированные компетенции преподавателю.

2. Студент должен знать интерфейсные возможности FreeMat и уметь демонстрировать преподавателю навыки работы с различными окнами FreeMat.
3. Студент должен знать основные операторы, функции, управляющие конструкции FreeMat и уметь демонстрировать на практике их работу.
4. Студент должен уметь задавать функциональные зависимости нескольких переменных, используя различные возможности FreeMat, использовать на практике встроенные функции среды FreeMat и выводить функциональные зависимости в графические окна FreeMat.
5. Студент должен уметь демонстрировать на практике использование операторов работы с трехмерной графикой среды FreeMat.
6. Студент должен иметь четкое представление о специфике использования и уметь демонстрировать преподавателю работу арифметических операторов, матричного исчисления и арифметических операторов поэлементных операций над векторами и матрицами.

ЛАБОРАТОРНАЯ РАБОТА № 7

Тема. Использование операторов с условием и циклических операторов в FreeMat.

Цель работы

1. Изучить основные возможности программного пакета FreeMat при работе с условными операторами и операторами цикла.
2. Используя возможности программы FreeMat, составить программу и выполнить задание в соответствии с выданным вариантом.

Задание и ход выполнения работы

1. Сформировать практические навыки написания и выполнения программ в FreeMat.
2. Составить и выполнить программу в соответствии с выданным вариантом задания.
3. В отчете по работе приведите код программы, исходный и конечный варианты массивов в виде распечаток двух таблиц.

Для формирования исходного массива из 100 элементов используйте следующий программный код в FreeMat:

```
clear all
x=rand(1,100);
x=x-0.3;
x=x*100
```

Варианты заданий к лабораторной работе № 7

1. Составить массив из 100 элементов и найти максимальный положительный элемент.
2. Составить массив из 100 элементов и найти максимальный отрицательный элемент.
3. Составить массив из 100 элементов и найти максимальный по модулю элемент массива.
4. Составить массив из 100 элементов и поменять местами первый и последний элемент массива.
5. Составить массив из 100 элементов и найти номер элемента, имеющий наименьшее значение.
6. Составить массив из 100 элементов и найти номер элемента, имеющий наибольшее значение.
7. Составить массив из 100 элементов и удвоить наибольший элемент массива.
8. Составить массив из 100 элементов и уменьшить в два раза наименьший элемент массива.
9. Составить массив из 100 элементов, преобразовать его в массив целых чисел и найти все четные числа среди положительных элементов.
10. Составить массив из 100 элементов, преобразовать его в массив целых чисел и найти все четные числа среди отрицательных элементов.
11. Составить массив из 100 элементов и определить, сколько элементов массива больше 10.
12. Составить массив из 100 элементов, преобразовать его в последовательность из 10 векторов-строк и найти вектор-строку с наибольшей суммой элементов.
13. Составить массив из 100 элементов и заменить все элементы на 1.
14. Составить массив из 100 элементов и упорядочить его по возрастанию.
15. Составить массив из 100 элементов и найти среднее арифметическое элементов массива.

Контроль сформированных практических навыков

1. Студент должен уметь запускать программный пакет FreeMat и настраивать операционную среду работы с ним, демонстрируя сформированные компетенции преподавателю.
2. Студент должен знать интерфейсные возможности FreeMat и уметь демонстрировать преподавателю навыки работы с различными окнами FreeMat.
3. Студент должен знать основные операторы, функции, управляющие конструкции FreeMat и уметь демонстрировать на практике их работу.
4. Студент должен уметь:

4.1. Применять на практике условные операторы **if** и **switch** и записывать программные конструкции FreeMat на их основе.
4.2. Применять на практике операторы цикла **while**, **for** и уметь записывать программные конструкции на их основе.
4.3. Записывать и анализировать выражения – условия с использованием логических операторов.

ЛАБОРАТОРНАЯ РАБОТА № 8

Тема. Поведение квантовой частицы в потенциальной яме.

Цель работы:

1. Закрепить знания и практические навыки работы с FreeMat.
2. Изучить принцип матричного подхода при решении задач квантовой механики на примере рассмотрения поведения квантовой частицы в «потенциальном ящике».
3. Изучить характер поведения квантовой частицы в «потенциальном ящике».
4. На основе научно-лабораторного исследования выяснить возможные причины появления волновых пакетов.

Задание и ход выполнения работы

Рассмотрим задачу о поведении квантовой частицы в «потенциальной яме». Пусть потенциальная энергия является постоянной и конечной внутри ямы (практически равна нулю), а на границах при **x = 0** и **x = L** движение ограничено бесконечно высокими стенками – потенциальным барьером (см. рис. 21).

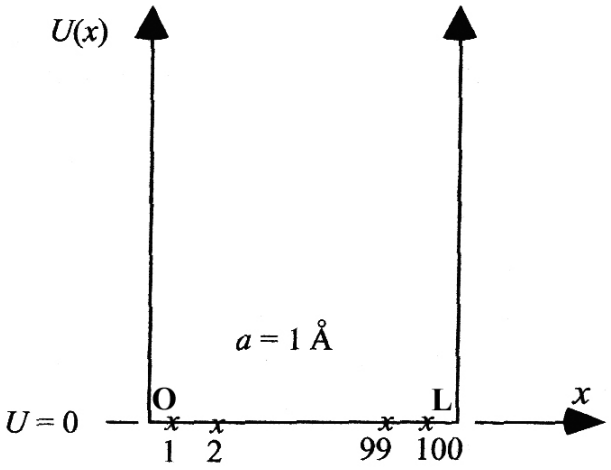


Рис. 21. Схематичное изображение потенциальной ямы для квантовой частицы: ширина ямы **L,** шаг по координате **x → a** условно составляет один ангстрем; количество точек разбиения (узлов решетки) условно составляет 100 шт.

Решим задачу численно. Если волновая функция задана на дискретной решетке со 100 точками, то матрица гамильтониана **[H]** также будет иметь размер **100x100**, где все **Un=0**:

	1	*2*	*3*	…	*98*	*99*	*100*
1	$2t_0$	$-t_0$	*0*	…	*0*	*0*	*0*
2	$-t_0$	$2t_0$	$-t_0$	…	*0*	*0*	*0*
…	…	…	…	…	…	…	…
99	*0*	*0*	*0*		$-t_0$	$2t_0$	$-t_0$
100	*0*	*0*	*0*	…	*0*	$-t_0$	$2t_0$

Для решения задачи о поведении квантовой частицы в потенциальной яме:

1. Используя возможности FreeMat, найдите 100 собственных значений и соответствующих собственных векторов для такой системы (все эти значения будут являться решениями волнового уравнения как линейные комбинации волновой функции – см. п. 14.3 данного учебного пособия).
2. Постройте зависимости:
   2.1. **E**, эВ (собственные значения, упорядоченные по возрастанию) = f(**α**) (номера собственного значения в узле сетки разбиения).
   2.2. Ψ=f(α).
   2.3 **γ** (вероятности) = f(**α**).

Рекомендации для составления программы для FreeMat:

Блок 1. Ввод констант (константы имеют рекомендательный характер):
– **hbar** → ħ = 1,055×10^{-34} Дж·с (редуцированная постоянная Планка);
– **m** – масса покоя кантовой частицы (электрона), равная 9,11×10^{-31} кг;
– **q** – элементарный заряд, равный 1,6×10^{-19} Кл.

Блок 2. Составление решётки:
– **Np** – константа, количество узлов решетки;
– **a** – константа, шаг решетки;
– **X** – заготовка решетки, вектор от **a** до **a*Np**, сделать его достаточно просто, задав массив и умножив его на скаляр:

X=a[1:1:Np]*;

Блок 3. Введение дополнительных переменных:

– **t_0** в эВ, как и любая энергетическая величина (так нам удобнее). Наполним, что t_0, выраженное в Дж:

$$t_0 = \frac{\hbar^2}{2ma^2}.$$

Для получения величины t_0 в эВ необходимо ее разделить на элементарный заряд.

– **L** – последнее значение вектора решетки;

– **H** – матрица гамильтониана.

Примечание:

Пояснения по методике расчета **H**:

$H_{n,m} = 2t_0\delta_{n,m} - t_0\delta_{n,m+1} - t_0\delta_{n,m-1}$ – выражение, определяющее элементы матрицы гамильтониана, с учетом символа Кронекера;

$H_{n,m} = 2t_0 - t_0 - t_0$ – то же, но символ Кронекера опущен, нули в незначимых диагоналях будут формироваться известным вам оператором «**ones**»:
ones(m,n) – формирует массив единиц размера (***m x n***).

Однако нам необходим следующий результат:

	1	***2***	***3***	***…***	***98***	***99***	***100***
1	$2t_0$	$-t_0$	***0***	***…***	***0***	***0***	***0***
2	$-t_0$	$2t_0$	$-t_0$	***…***	***0***	***0***	***0***
…	***…***	***…***	***…***	***…***	***…***	***…***	***…***
99	***0***	***0***	***0***		$-t_0$	$2t_0$	$-t_0$
100	***0***	***0***	***0***	***…***	***0***	$-t_0$	$2t_0$

«Средняя линия» – главная диагональ (см. выше) – количество значений в ней, в данном случае равно 100. Верхняя и нижняя линии относительно главной диагонали – это линии, прилегающие к главной диагонали, и, если в главной диагонали вектором является, например, «**v**», то для нижней диагонали вектором будет «**v-1**», а для верхней – «**v+1**».

Итак, **ones(m,n)** – формирует массив единиц размера **(m x n)**, и если мы формируем массив нулей, сдвигаясь относительно главной диагонали на

единицу, то этот оператор следует записывать так: **ones(1,100-1)**, если число точек на решетке равно 100.

Оператор **diag(v)** – формирует матрицу с вектором «**v**» на главной диагонали (какие значения будут записаны на главной диагонали – данный оператор не определяет).

Итак, diag(100) – означает лишь то, что будет создана матрица, у которой есть главная диагональ в 100 элементов.

Оператор **diag(v,k)** – тоже формирует матрицу, но другого порядка, с вектором на k-й диагонали относительно предыдущего случая.

Умножение элементов на скаляр позволит заполнить диагональные элементы матрицы (главной и k-й относительно главной).

Пример программы в FreeMat для главной диагонали матрицы, определяющей гамильтониан **H**, можно записать так:

```
H=(2*t0*diag(ones(1,Np)));  % осталось дописать для диагоналей +1 и -1,
                            % относительно главной
                            % (выражение записано без учета потенциала).
```

Полное выражение для гамильтониана с учетом введенных выше переменных будет выглядеть так:

```
H=(2*t0*diag(ones(1,Np)))-(t0*diag(ones(1,Np-1),1))-(t0*diag(ones(1,Np-1),-1));
```

При правильном заполнении матрицы гамильтониана вы должны получить результат примерно такого вида, как показано далее (для удобства показана только часть матрицы):

```
--> H

ans =

 Columns 1 to 6

    7.6360   -3.8180         0         0         0         0
   -3.8180    7.6360   -3.8180         0         0         0
         0   -3.8180    7.6360   -3.8180         0         0
         0         0   -3.8180    7.6360   -3.8180         0
         0         0         0   -3.8180    7.6360   -3.8180
         0         0         0         0   -3.8180    7.6360
         0         0         0         0         0   -3.8180
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

Блок 4. Основной расчет:

4.1. Находим собственные значения и собственные вектора матрицы **H**.

Суть здесь состоит в нахождении решений системы уравнений, которая может быть интерпретирована как алгебраический эквивалент системы обыкновенных дифференциальных уравнений в явной форме Коши. Все это было бы чрезвычайно трудоемко, если бы не специальные функции FreeMat.

Так, *[R,D] =eig(H)* – вычисляет диагональную матрицу **D** собственных значений и матрицу **R** собственных векторов. Эти векторы нормируются так, что норма каждого из них рана единице.

Результатом для собственных значений получается матрица **D** (размером **100x100** для нашего случая), главная диагональ которой заполнена собственными решениями:

```
--> D

ans =

 Columns 1 to 6

   0.0037         0         0         0         0         0
        0    0.0148         0         0         0         0
        0         0    0.0332         0         0         0
        0         0         0    0.0590         0         0
        0         0         0         0    0.0922         0
        0         0         0         0         0    0.1326
        0         0         0         0         0         0
        0         0         0         0         0         0
        0         0         0         0         0         0
```

Матрица собственных векторов **V** также (в данном случае) имеет размер **100x100** и заполнена значениями.

4.2. Заметим, что функция **diag()** – имеет и обратное действие, то есть извлечение из матрицы, которая указывается в качестве аргумента, главной диагонали в виде вектора-столбца. Сделайте это для удобства последующих расчетов:

D=diag(D); *% теперь **D** вектор-столбец.*

4.3. Далее отсортируем вектор **D** по возрастанию, но запомним следование элементов массива, для их ассоциации с собственными векторами. Для этого воспользуемся функцией:

[Enum,ind] =sort(D);

Результат извлечения и сортировка для некоторого примера показан далее:

```
--> D
ans =
   0.0037
   0.0148
   0.0332
   0.0590
   0.0922
   0.1326
   0.1803
   0.2352
   0.2973
   0.3664
```

Так же как и ранее (для удобства) приведена только часть вектора **D**.

Теперь **Enum** – это нумерованный и отсортированный по возрастанию массив (вектор) с энергией, все значения которого являются решениями уравнения Шрёдингера, а **ind** – индекс массива **D** до сортировки. Теперь, если нужно ассоциировать собственное решение с собственным вектором – есть массив с первоначальным расположением элементов.

4.4. Расчет **Ψ** (пси) на основе заданного начального состояния, так как оно самое точное (все остальные собственные значения вычисляются с погрешностью, ввиду использования метода конечных разностей).

Само **пси** не столь интересно, интереснее произведение **пси** на комплексно сопряженную с ней величину – это, как известно, вероятность нахождения квантовой частицы.

Пси – это и есть не что иное, как собственный вектор, но в отличие от него **пси** будет представлена действительной составляющей собственного вектора, которое выделяется функцией **abs()**.

Итак, пси – это одномерный массив, полученный для конкретного собственного значения – иными словами строка в массиве **V**.

Расчет **пси** на основе 1 собственного значения:

```
psi=abs(V(:,ind(1)));      % ind(1…2 …100) на основе какого собственного
                           % значения происходит расчет
                           % плотности вероятности:
PW=psi1.*conj(psi);
```

Разъяснение: **V(:,1**) – означает все значения столбцов для 1 строки, функция **conj(psi)** – возвращает комплексно сопряженные значения для элементов аргумента **psi**.

Блок 5. Вывод результатов в виде графиков:
(см. п.п. теории 2.1 – 2.3).

Примеры графиков из программы FreeMat, включаемые в отчет по лабораторной работе, приведены на рис. 22-33.

Результаты, полученные при расчете на основе первого собственного значения (**ind(1)**):

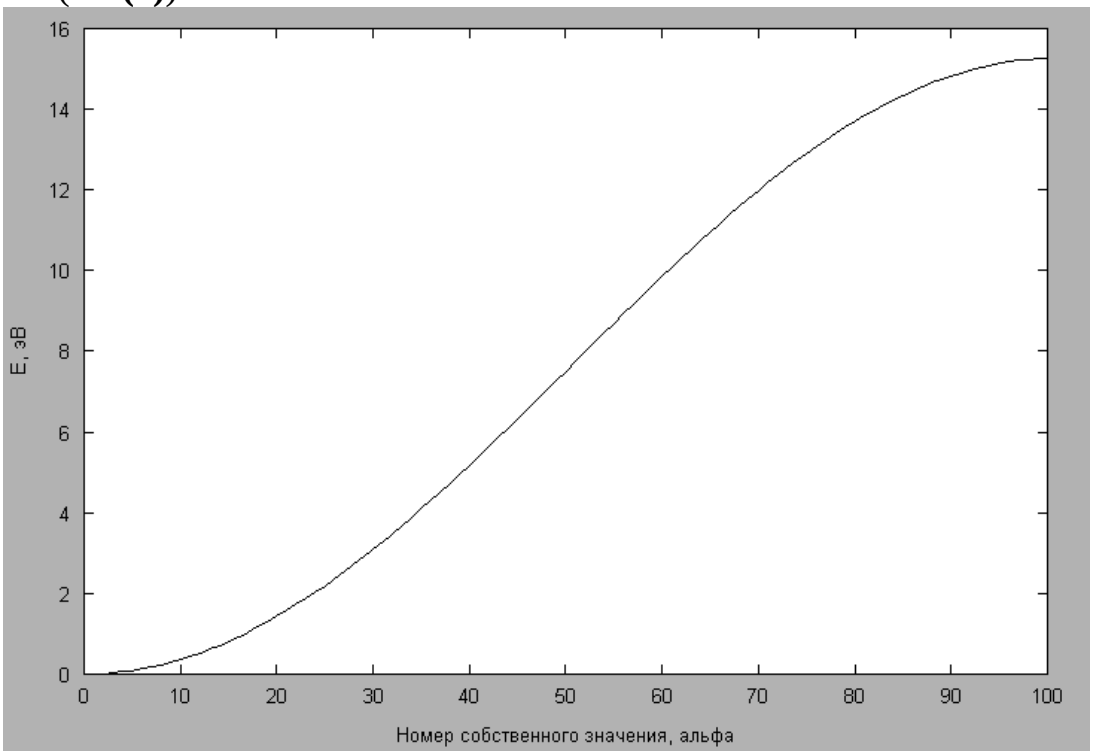


Рис. 22. Упорядоченные собственные значения от номера собственного значения, полученного на решетке (расчет произведен на основе первого собственного значения – **ind(1)**)

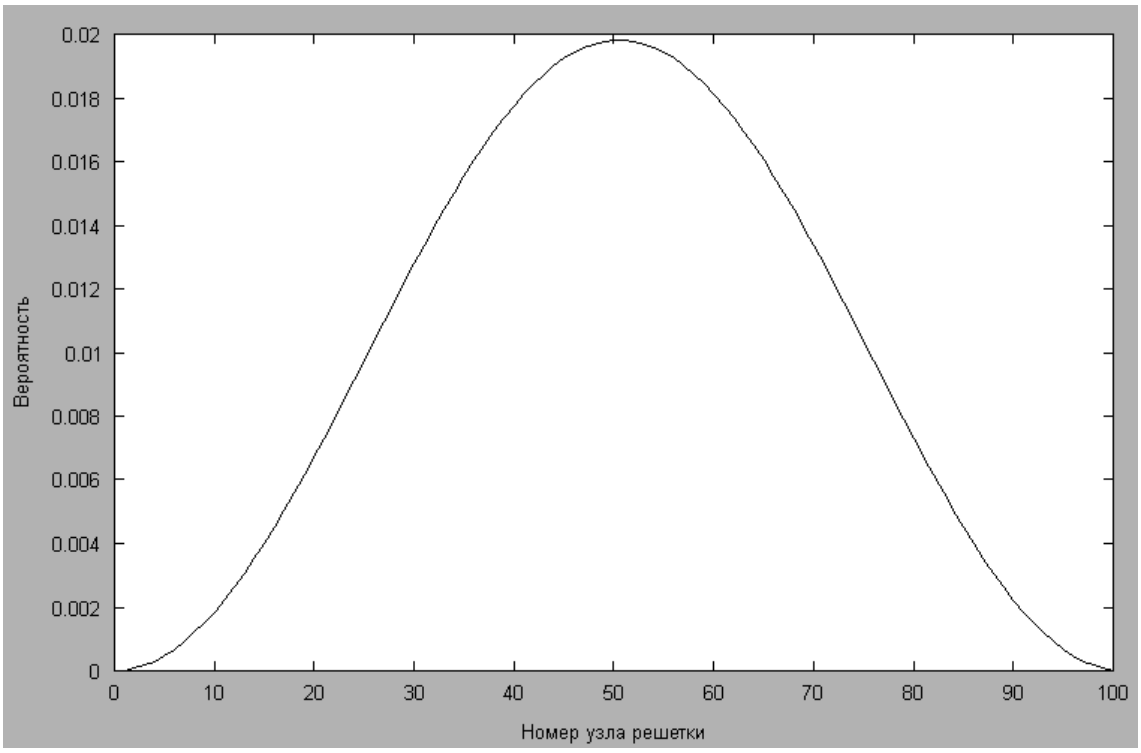


Рис. 23. Распределение плотности вероятности (квадрата волной функции) от номера узла решетки (расчет произведен на основе первого собственного значения – **ind(1)**)

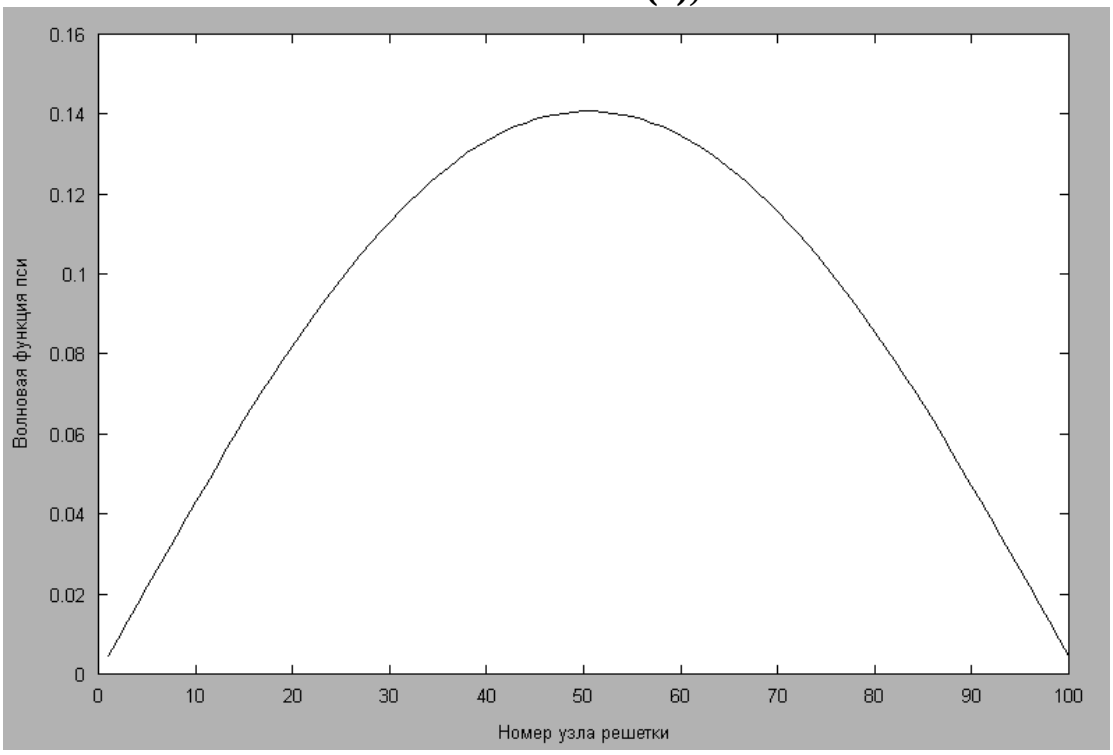


Рис. 24. Волновая функция пси от номера узла решетки (расчет произведен на основе первого собственного значения – **ind(1)**)

Результаты, полученные при расчете на основе пятого собственного значения (**ind(5)**):

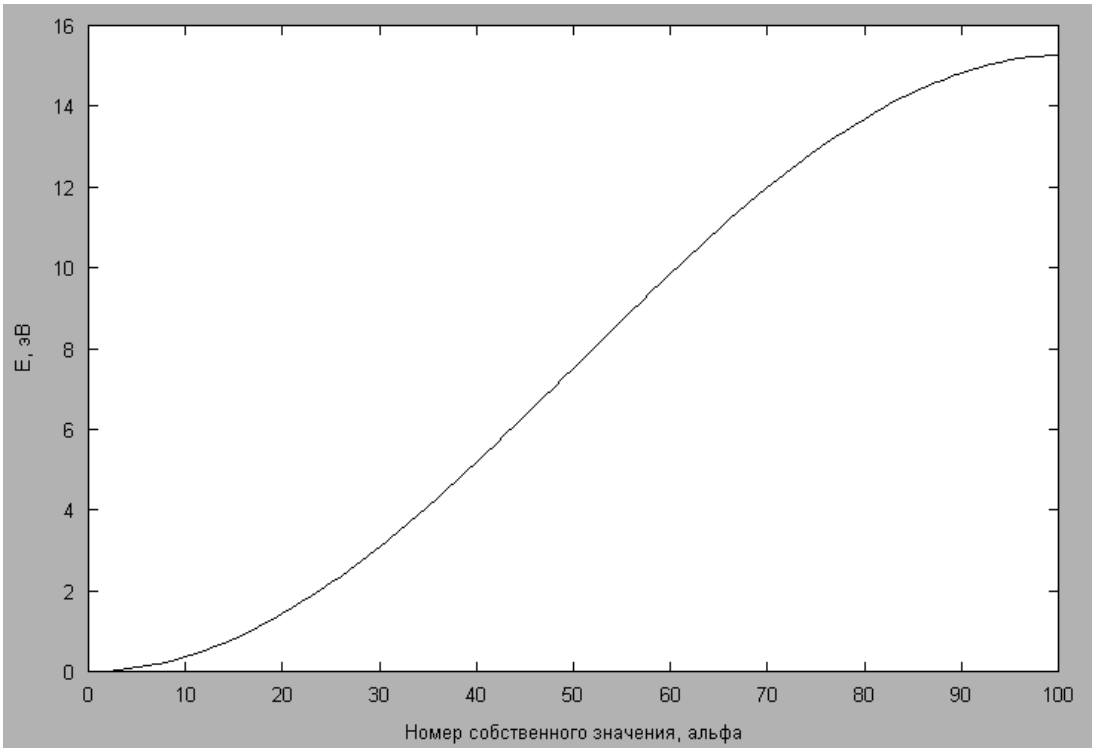


Рис. 25. Упорядоченные собственные значения от номера собственного значения, полученного на решетке (расчет произведен на основе пятого собственного значения – **ind(5)**)

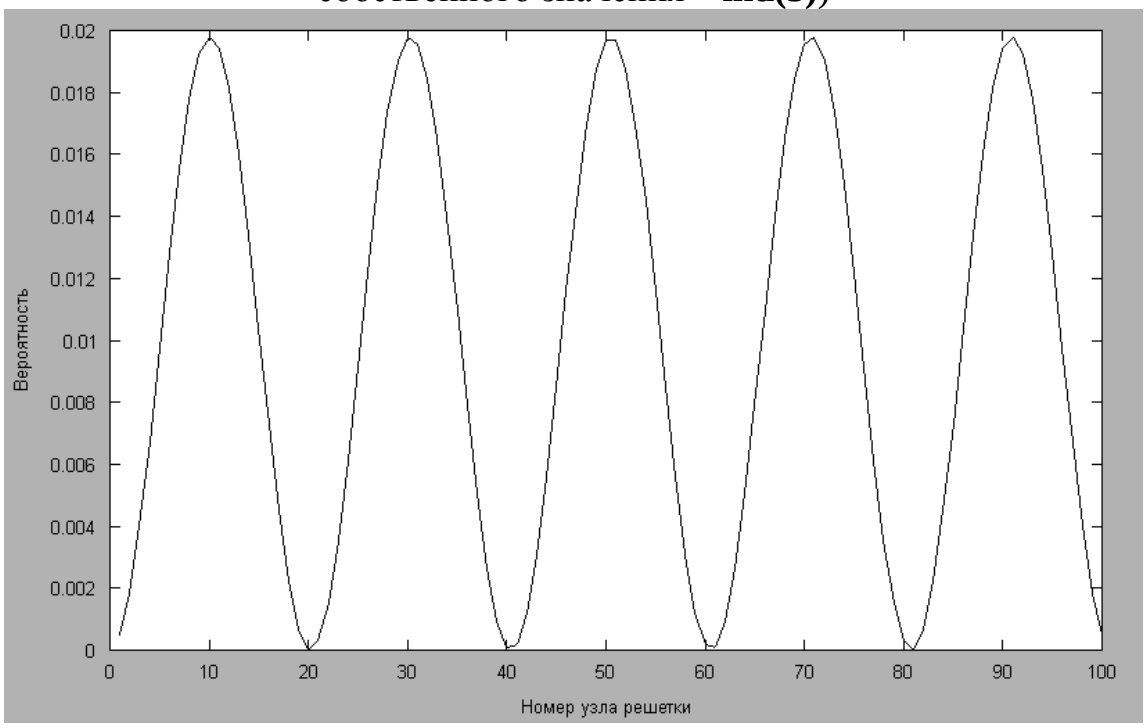


Рис. 26. Распределение плотности вероятности (квадрата волной функции) от номера узла решетки (расчет произведен на основе пятого собственного значения – **ind(5)**)

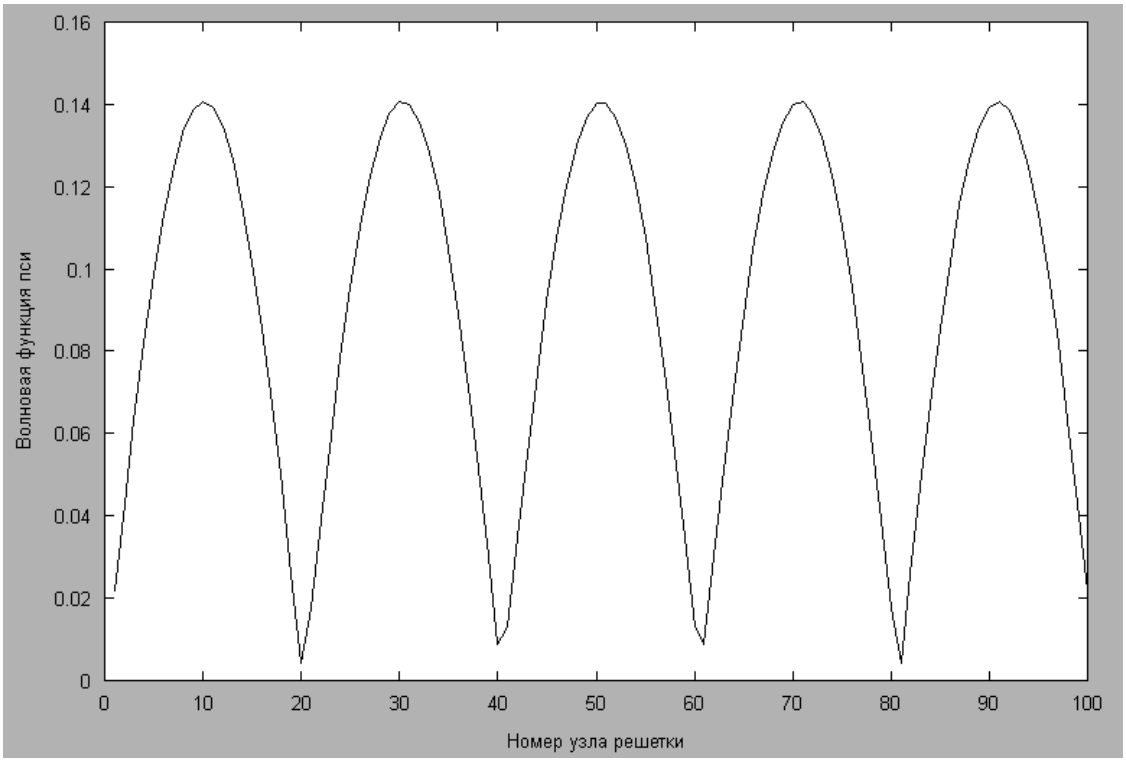


Рис. 27. Волновая функция пси от номера узла решетки (расчет произведен на основе пятого собственного значения – **ind(5)**)

Анализ графиков на рис. 22 и 25, а также всех остальных 98 аналогичных графиков, получаемых на основе других собственных значений, показывает их идентичность. По сути они и не могут различаться, так как являются всего лишь решениями одного и того же уравнения Шрёдингера. Остальные графики на рис. 23, 24, 26 и 27, а также на рис. 29-33 требуют объяснения. Дайте объяснения характеру их поведения на основе одного из выбранных ракурсов наблюдения и возможного появления волновых пакетов, за счет интерференции электрона самим с собой.

Результаты, полученные при расчете на основе двадцать пятого собственного значения (**ind(25)**):

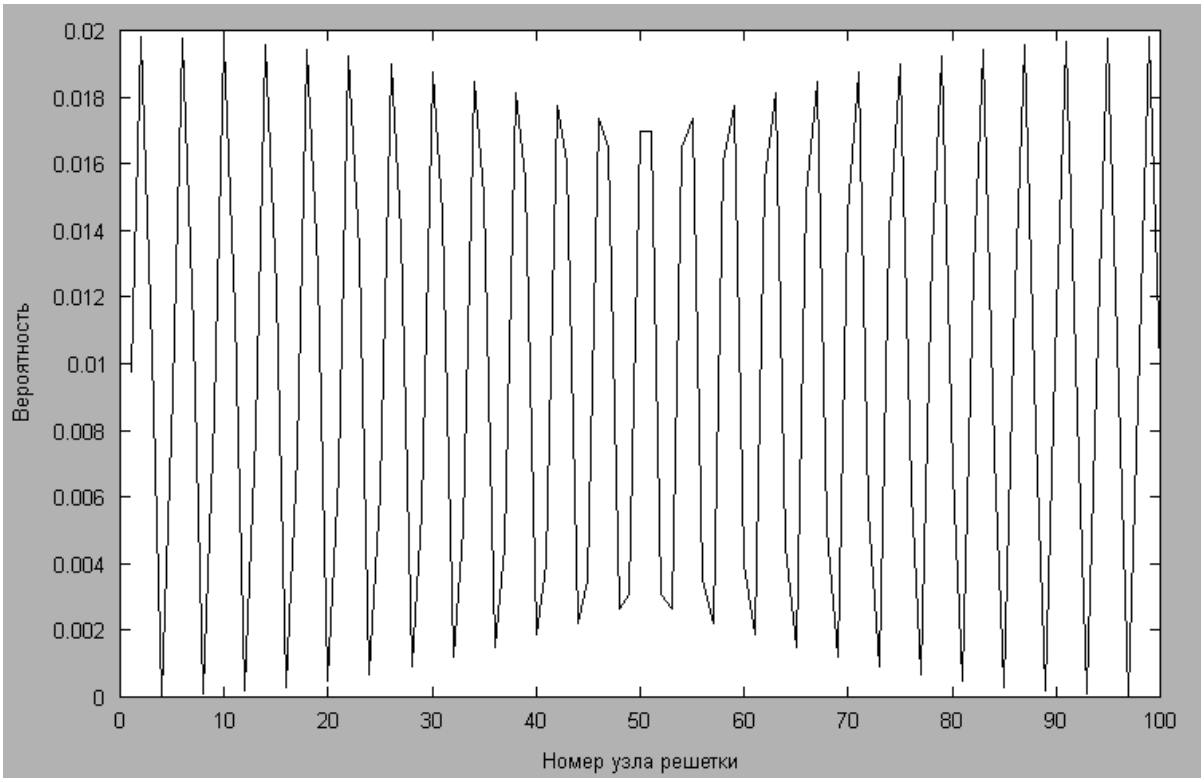


Рис. 28. Распределение плотности вероятности (квадрата волной функции) от номера узла решетки (расчет произведен на основе двадцать пятого собственного значения – **ind(25)**)

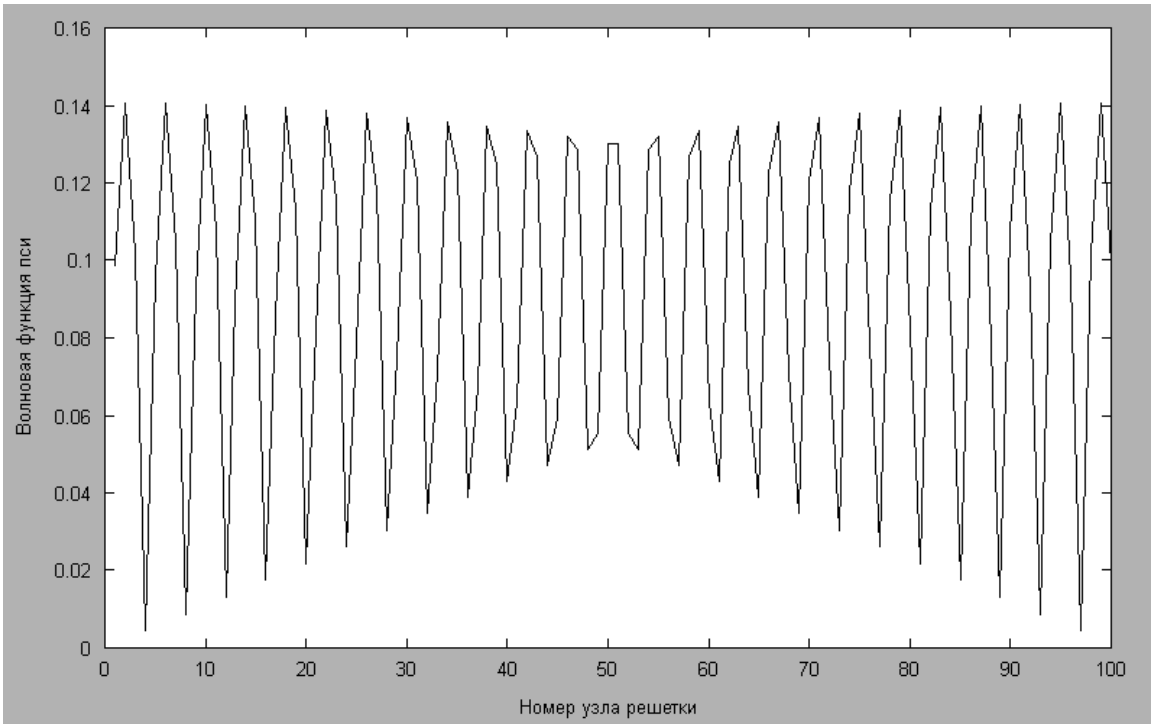


Рис. 29. Волновая функция пси от номера узла решетки (расчет произведен на основе двадцать пятого собственного значения – **ind(25)**)

Результаты, полученные при расчете на основе тридцатого собственного значения (**ind(30)**):

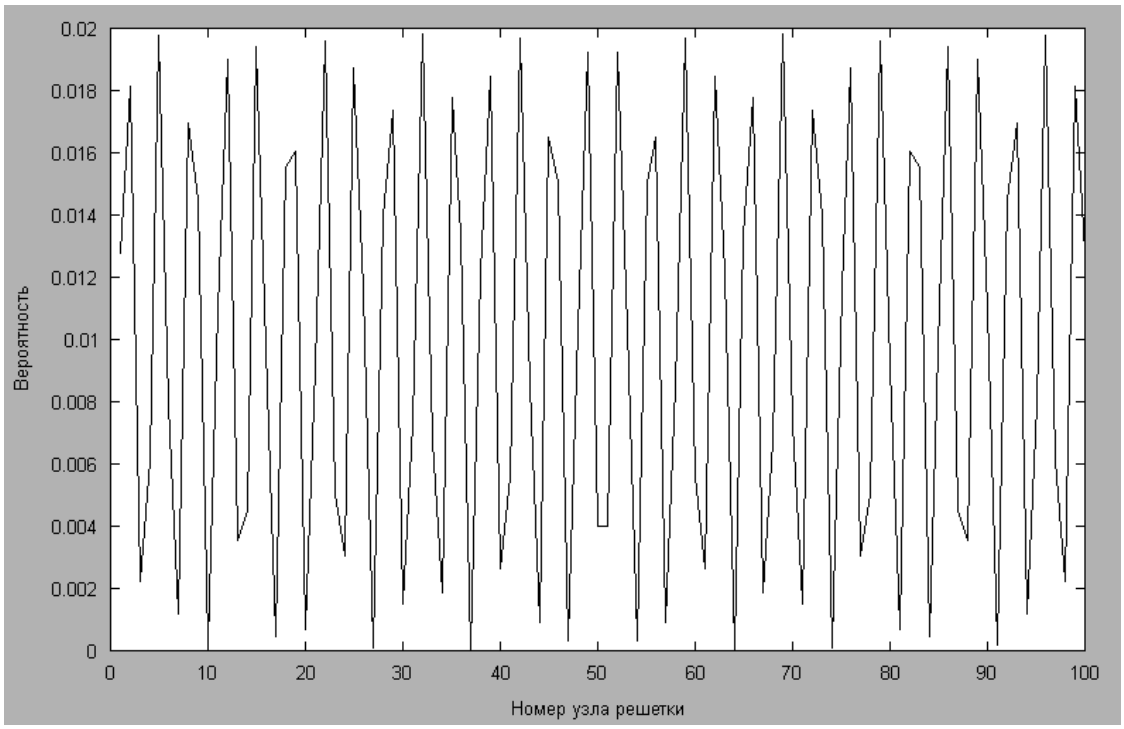


Рис. 30. Распределение плотности вероятности (квадрата волной функции) от номера узла решетки (расчет произведен на основе тридцатого собственного значения – **ind(30)**)

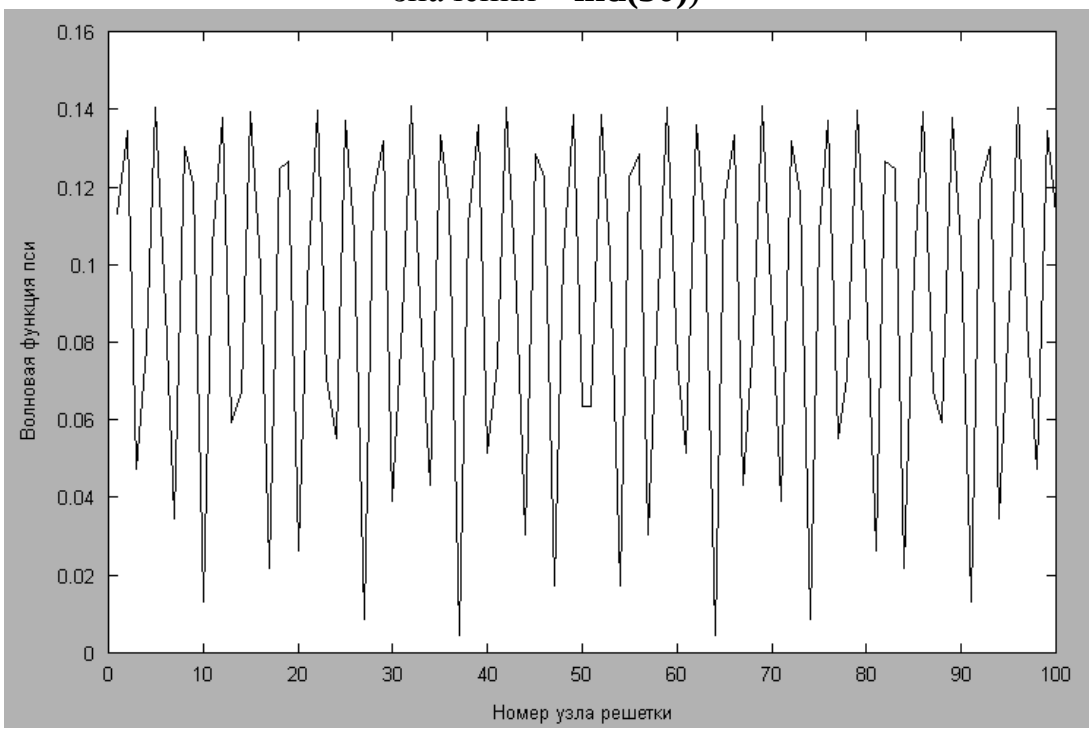


Рис. 31. Волновая функция пси от номера узла решетки (расчет произведен на основе тридцатого собственного значения – **ind(30)**)

Результаты, полученные при расчете на основе семьдесят пятого собственного значения (**ind(75)**):

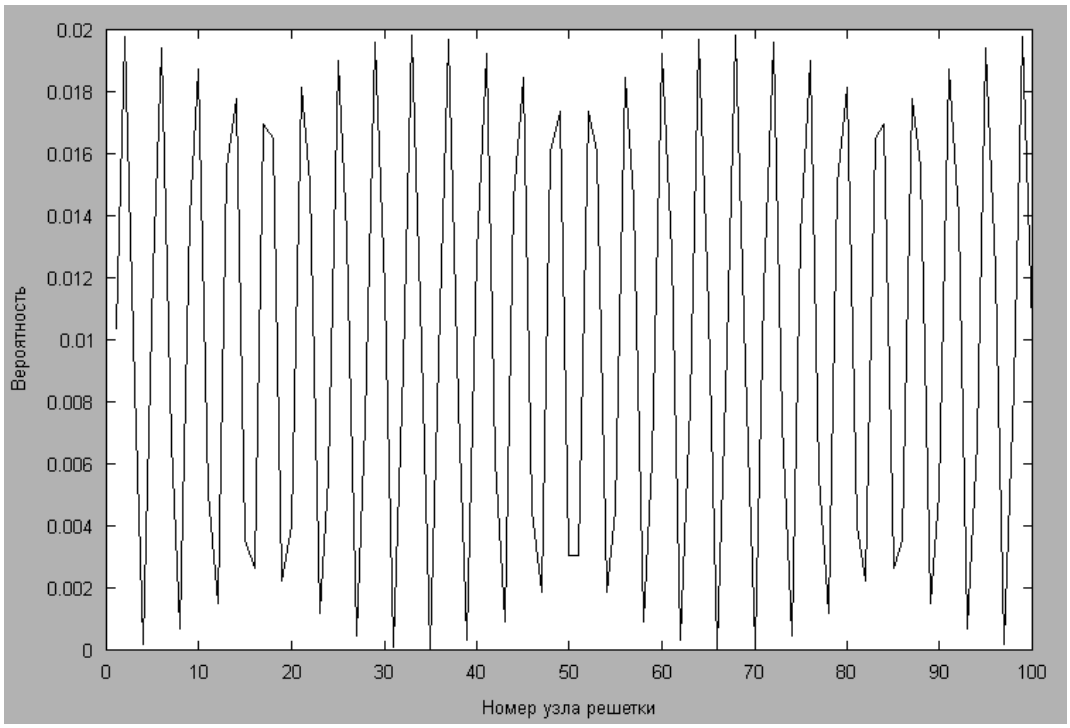


Рис. 32. Распределение плотности вероятности (квадрата волной функции) от номера узла решетки (расчет произведен на основе семьдесят пятого собственного значения – **ind(75)**)

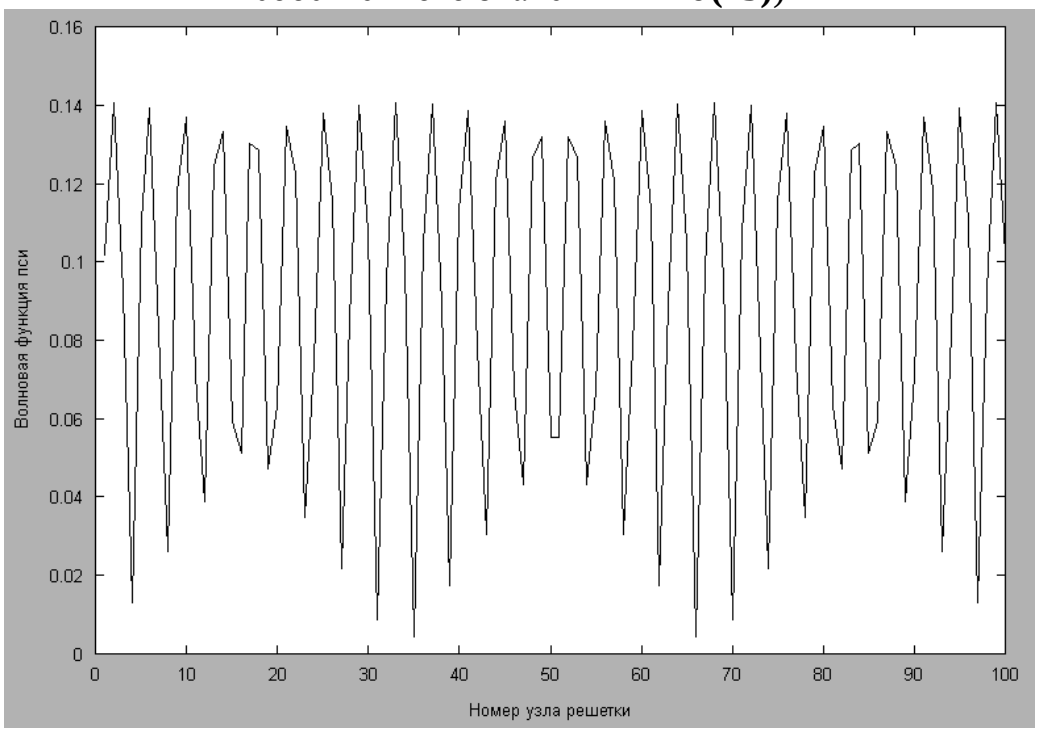


Рис. 33. Волновая функция пси от номера узла решетки (расчет произведен на основе семьдесят пятого собственного значения – **ind(75)**)

Варианты выполнения лабораторной работы № 8 приведены в табл. 16.

Таблица 16

Варианты заданий к лабораторной работе № 8

№ вар.	$N_{реш.}$, шт.	a, Å	α	№ вар.	$N_{реш.}$, шт.	a, Å	α	№ вар.	$N_{реш.}$, шт.	a, Å	α
1	100	1	1	6	100	1	26	11	100	1	51
	100	1	2		100	1	27		100	1	52
	100	1	3		100	1	28		100	1	53
	100	1	4		100	1	29		100	1	54
	100	1	5		100	1	30		100	1	55
2	100	1	6	7	100	1	31	12	100	1	56
	100	1	7		100	1	32		100	1	57
	100	1	8		100	1	33		100	1	58
	100	1	9		100	1	34		100	1	59
	100	1	10		100	1	35		100	1	60
3	100	1	11	8	100	1	36	13	100	1	61
	100	1	12		100	1	37		100	1	62
	100	1	13		100	1	38		100	1	63
	100	1	14		100	1	39		100	1	64
	100	1	15		100	1	40		100	1	65
4	100	1	16	9	100	1	41	14	100	1	66
	100	1	17		100	1	42		100	1	67
	100	1	18		100	1	43		100	1	68
	100	1	19		100	1	44		100	1	68

	100	1	20		100	1	45		100	1	70
5	100	1	21	10	100	1	46	15	100	1	71
	100	1	22		100	1	47		100	1	72
	100	1	23		100	1	48		100	1	73
	100	1	24		100	1	49		100	1	74
	100	1	25		100	1	50		100	1	75

Примечание:

– $N_{реш.}$ – количество точек на решетке, удовлетворяющих условию отсутствия влияния потенциала на характер движения квантовой частицы в потенциальном ящике;

– a – шаг решетки;

– α (альфа) – номер собственного значения, он же номер точки на решетке.

*– с целью более детальной проработки сути вопроса данной лабораторной работы по решению преподавателя диапазоны величин и значений $\boldsymbol{N_{реш.}}$ и **a** могут меняться.*

Методические рекомендации к лабораторной работе № 8

1. При выполнении данной лабораторной работы рекомендуется придерживаться командно-модульного принципа работы (каждая подгруппа или каждый обучающийся выполняет свой вариант работы, а общие выводы по работе принимаются на основе анализа всех полученных данных всеми подгруппами обучающихся).
2. Данная лабораторная работа носит научно-исследовательский характер, в связи с этим обучающимся рекомендуется изучить и проанализировать в отчете по данной лабораторной работе дополнительную литературу, подтверждающую выводы по данной работе.

Вопросы к лабораторной работе № 8 и контроль сформированных практических навыков

1. Записать и объяснить физический смысл уравнения Шрёдингера в дифференциальной форме и в операторном виде.
2. Объясните, как влияет на электрон характер внешнего воздействия.
3. Поясните принцип возникновения волновых пакетов.

4. Поясните термин волновой пакет.
5. В чем состоит суть принципа неопределенности Гейзенберга?
6. Как проявляется принцип неопределенности при рассмотрении задачи характера поведения квантовой частицы в «потенциальном ящике»?
7. Объясните физический смысл волной функции пси.
8. Приведите общий алгоритм расчетов, используемый в данной лабораторной работе.
9. Какие функции и операторы работы с матрицами (векторами) использованы при расчетах в данной лабораторной работе?
10. Какие операторы работы с графикой вы использовали при выполнении данной лабораторной работы?

Список рекомендуемой литературы

1. Наместников, С.М. Основы программирования в MatLab. Сборник лекций [Электронный ресурс] / С.М. Наместников. – Ульяновск, 2011. – 55 с. – URL: http://sernam.ru/lect_matlab.php (дата обращения 03.09.18).
2. FreeMat v4.0 – Online Documentation [Electronic resource]. – URL: http://freemat.sourceforge.net/help/index.html (accessed: 03.09.18).
3. FreeMat [Electronic resource]. – URL: http://freemat.sourceforge.net/ (accessed: 03.09.18).
4. Datta, S. Nanoelectronics and the meaning of resistance. The course of lectures [Electronic resource]. – URL: http://nanohub.org/resources/5279 (accessed: 03.09.2018).
5. Нанотехнологическое общество России (НОР) [Электронный ресурс]. – URL: http://ntsr.info/ (дата обращения: 03.09.2018).
6. Рыжонков, Д.И. Наноматериалы: учеб. пособие / Д.И. Рыжонков, В.В. Лёвина, Э.Л. Дзидзигури. – М.: БИНОМ, Лаборатория знаний, 2008. – 365 с.
7. Гусев, А.И. Наноматериалы, наноструктуры, нанотехнологии / А.И. Гусев. – М.: ФИЗМАТЛИТ, 2005. – 416 с.

Ситанов Дмитрий Вячеславович

Пивоваренок Сергей Александрович

FreeMat для инженерных и научных расчетов

Учебное пособие

Редактор В.Л. Родичева

Подписано в печать 02.10.2018. Формат 60×84 1/16. Бумага писчая.
Усл. печ. л. 5,12. Уч.-изд. л. 5,68. Тираж 50 экз. Заказ

ФГБОУ ВО «Ивановский государственный
химико-технологический университет»

Отпечатано на полиграфическом оборудовании
Редакционно-издательского центра ФГБОУ ВО «ИГХТУ»
153000, г. Иваново, пр. Шереметевский, 7