Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных средств

А. В. Станкевич

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Лабораторный практикум
для студентов специальности I-40 02 02
«Электронные вычислительные средства»
дневной формы обучения

Минск 2007

УДК 004.9 (075.8)
ББК 32.973.26-02 я 73
С 76

Рецензент
заведующий кафедрой радиоэлектронных средств БГУИР, профессор,
канд. техн. наук Н. С. Образцов

Станкевич, А. В.

С 76 Теоретические основы систем автоматизированного проектирования: лаб. практикум для студ. спец. I-40 02 02 «Электронные вычислительные средства» дневн. формы обуч. / А. В. Станкевич. – Минск : БГУИР, 2007. – 54 с. : ил.

ISBN 978-985-488-136-2

В практикуме приведены описания шести лабораторных работ по дисциплине «Теоретические основы систем автоматизированного проектирования» для студентов специальности I-40 02 02 «Электронные вычислительные средства» дневной формы обучения.

УДК 004.9 (075.8)
ББК 32.973.26-02 я 73

ISBN 978-985-488-136-2

Содержание

ЛАБОРАТОРНАЯ РАБОТА №1. ИЗУЧЕНИЕ ПРИНЦИПОВ РАБОТЫ СИСТЕМЫ MATHCAD

Цель: изучить пользовательский интерфейс и входной язык системы MathCAD, научиться выполнять вычисления, строить графики и таблицы.

1.1 Теоретические сведения

Общие понятия

MathCAD – математический пакет, предназначенный для выполнения инженерных, экономических и научных расчетов. Основное достоинство пакета – естественный математический язык, с помощью которого формулируются решаемые задачи, причем пакет не только позволяет провести расчеты, но и оформить документ с помощью графиков, рисунков, таблиц и математических формул.

Окно MathCAD содержит типовые элементы Windows-приложения: заголовок, главное меню, кнопочные панели и окно документа. Будем предполагать, что пользователь знаком с основными принципами работы с Windows-приложением (создание, открытие, сохранение файлов, приемы редактирования и т.п.). Далее рассмотрим только специфические команды системы, которые могут быть вызваны через меню или с помощью кнопок кнопочных панелей инструментов. Кнопочные панели визуализируются/скрываются с помощью команды ***View/Toolbars***.

При загрузке MathCAD автоматически создается новый документ. Для создания нового документа нужно выполнить команду ***File/New.***

Рабочая область окна MathCAD делится вертикальной штриховой линией на левую и правую части. Левая часть выводится на печать, поэтому все исходные данные, расчеты и результаты приводятся в левой части рабочего окна. В правой части обычно размещают комментарии и вспомогательные вычисления.

Все содержимое документа состоит из блоков (блоков формул, блоков текста или графиков). Для выделения блока необходимо по нему щелкнуть либо выбрать с помощью окна выбора. Для любых блоков работают операции копирования и вставки через буфер обмена (через пункт меню ***Edit***).

MathCAD относится к интерпретаторам. Исходное описание пользователя (документ MathCAD) просматривается слева направо, сверху вниз. Указанный порядок просмотра блоков означает, что, например, при построении графика функции вначале (сверху или слева) нужно расположить блоки, задающие саму функцию и пределы изменения аргумента, а уже затем блок графика.

Блоки перемещаются путем их перетаскивания. Для автоматического разделения перекрывающихся областей можно использовать команду ***Format/Separate Regions***.

Алфавит входного языка системы MathCAD содержит: строчные и прописные латинские буквы, строчные и прописные греческие буквы, арабские цифры от 0 до 9, системные переменные, операторы, имена функций, специальные символы.

Создание и редактирование формул

Формула может состоять из следующих элементов: числовые константы, переменные, функции, операторы.

Числовые константы задаются с помощью арабских цифр, десятичной точки и знака минус. К числовым константам могут относиться и предварительно определенные системные переменные. Значения этих системных переменных можно переопределить:

- число p (3,14159);
- основание натурального логарифма e (2,71823);
- бесконечность ¥ (10^{307});
- процент % (0,01);
- погрешность численных методов *TOL* (0,001);
- нижняя граница индексации массивов *ORIGIN* (0).

Переменные являются поименованными объектами, имеющими некоторое значение, которое может изменяться по ходу выполнения программы.

Имена констант, переменных и иных объектов называют идентификаторами. Идентификаторы в системе MathCAD могут содержать прописные и строчные латинские и греческие буквы (строчные и прописные буквы различаются), цифры, символ подчеркивания, символ штриха «`» (символ находится на одной клавише с тильдой «~»), символ процента. Идентификатор должен начинаться с буквы. Идентификаторы должны быть уникальными и не должны совпадать с именами встроенных или пользовательских функций. Для использования греческих букв целесообразно воспользоваться командой ***View/Toolbars/Greec.***

MathCAD имеет большое количество встроенных функций (например функция вычисления синуса *sin (x)* аргумента *x*). Обращение к функции осуществляется по имени. Функции возвращают некоторое значение – символьное, числовое, вектор или матрицу. Могут также определяться и пользовательские функции.

Рассмотрим клавиши клавиатуры (указаны в кавычках), предназначенные для ввода наиболее часто используемых операторов MathCAD:

- оператор присваивания := (клавиша «:»);
- арифметические операторы «+» , «-» , «*» , «/»;
- возведение в степень X «^» Y;
- факториал X «!» ;
- квадратный корень «\» X;
- абсолютное значение «|»X.

Другие операторы удобнее набирать с использованием шаблонов, которые будут рассмотрены позже.

При использовании комплексных чисел число *z* представляется в виде *ReZ+ j* ImZ* или *ReZ+ i* ImZ*, где *ReZ* – действительная часть комплексного числа *z*, *ImZ* – его мнимая часть, а символы *i* или *j* обозначают мнимую единицу. Перед использованием любых операций с комплексными числами

целесообразно вначале определить *i* или *j* как мнимую единицу (т.е. присвоить им значение квадратного корня из –1).

Перед использованием все переменные должны быть предварительно определены пользователем (им необходимо присвоить значение). Попытка использовать неопределенную переменную приведет к выводу сообщения об ошибке. При этом переменная окрашивается в красный цвет.

Если переменной присваивается начальное значение с помощью оператора :=, такое присваивание называется локальным. С помощью знака ≡, вводимого с помощью кнопочной панели ***View/Toolbars/Evaluation***, можно обеспечить глобальное присваивание, т.е. оно может производиться в любом месте документа. Например, если переменной присвоено таким образом значение в самом конце документа, то она будет иметь это же значение и в начале документа.

Для запуска редактора формул достаточно в любом свободном месте рабочего окна щелкнуть левой кнопкой мыши. Появится указатель в виде маленького красного крестика. Его можно перемещать клавишами перемещения курсора. Указатель определяет место, с которого можно начинать ввод блока формулы. В области формул указатель превращается в синий уголок, указывающий направление и место ввода. Для ввода значения константы *a* следует, например, набрать *a:=123*. Для определения переменной после знака присваивания следует указать требуемое математическое выражение.

Ввод формул облегчается благодаря использованию шаблонов при задании того или иного математического выражения. Для этого в MathCAD служат кнопочные панели с шаблонами различных математических символов. Наиболее часто при работе с формулами используются следующие кнопочные панели: **Calculator** (***View/Toolbars/Calculator*** – рисунок 1.1), **Matrix** (***View/Toolbars/Matrix*** – рисунок 1.2), **Calculus** (***View/Toolbars/Calculus*** – рисунок 1.3).

В составе сложных шаблонов часто встречаются шаблоны для ввода отдельных элементов сложного выражения. Они имеют вид черных прямоугольников. На рисунке 1.4 приведен шаблон суммы из кнопочной панели **Calculus.** Для ввода данных нужно щелкнуть левой кнопкой мыши по нужному месту шаблона и ввести данные с клавиатуры.

Если после определения переменной или константы ввести ее имя и поставить знак «=», то MathCAD выведет результат вычисления по выражению.

При задании сложных выражений вычисления могут быть достаточно долгими. Для их прерывания можно нажать клавишу «Esc».

Для редактирования формулы нужно щелкнуть левой кнопкой мыши по нужному месту этой формулы и после появления указателя в виде синего уголка осуществить ввод требуемых данных.

Функции пользователя вводятся с применением следующего синтаксиса:

ИмяФункции (СписокПараметров): = Выражение

Имя функции задается в соответствии с правилом задания идентификатора. В скобках указывается список параметров (аргументов) функции, разделяемых запятыми. Выражение – любое выражение, содержащее операторы и функции с

аргументами, указанными в списке параметров. Пример задания функции одной переменной: *fun (x):= - 10* (1 - exp (x))*.

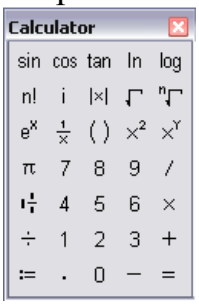

Рисунок 1.1 – Кнопочная панель **Calculator**

Рисунок 1.2 – Кнопочная панель **Matrix**

Рисунок 1.3 – Кнопочная панель **Calculus**

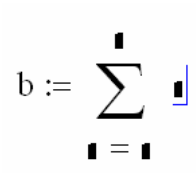

Рисунок 1.4 – Шаблон операции суммирования

Работа с массивами данных

В системе MathCAD используются массивы двух типов: одномерные (векторы) и двухмерные (матрицы).

Нижняя граница индекса элемента массива задается значением системной переменной *ORIGIN*, которая может принимать значение 0 или 1. Для ссылки на элемент массива нужно указать имя массива с нижним индексом. Например, для ссылки на второй элемент вектора *V* нужно записать V_1 (при *ORIGIN=0*).

Элементы матриц являются индексированными переменными, имена которых совпадают с именами матриц. Но в этом случае для каждой индексированной переменной указываются два нижних индекса: первый – для номера строки, второй – для номера столбца. Для элементов матрицы нижние индексы вводятся в круглых скобках с разделением их запятыми. При вводе формул для указания нижних индексов после имени переменной вводится знак открывающей квадратной скобки. Индексы могут иметь только целочисленные значения.

В отношении индексированных переменных действуют те же правила присваивания и вывода значений, что и для обычных переменных. Помимо операции присваивания можно создать матрицу или вектор с помощью команды ***Insert/Matrix...*** . В диалоговом окне команды нужно указать число строк **Rows** и число столбцов **Columns**. Нажав клавишу «Enter» или щелкнув по кнопке **Insert**, можно вывести шаблон матрицы или вектора. Шаблон содержит обрамляющие скобки и черные маленькие прямоугольники, обозначающие места ввода значений для элементов вектора или матрицы.

Создание текстовых блоков

В простейшем случае для открытия текстового редактора нужно ввести символ «"». В появившийся прямоугольник можно вводить текст. В текстовом блоке указатель имеет вид красной вертикальной черты и указывает место ввода. Текст редактируется общепринятыми способами работы с текстовыми редакторами.

Построение графиков

Для создания графиков в системе MathCAD имеется графический процессор.

Для построения графиков нужно выполнить команду ***Insert/Graph*** и из подменю выбрать требуемый вариант графика. В подменю ***Graph*** содержится список основных типов графиков:

- ***X-Y Plot*** – шаблон двухмерного графика в декартовой системе координат;
- ***Polar Plot*** – шаблон графика в полярной системе координат;
- ***Surface Plot*** – шаблон для построения трехмерного графика поверхности;
- ***Contour Plot*** – шаблон для карты линий уровня трехмерной поверхности;
- ***3D Scatter Plot*** – шаблон для трехмерного графика в виде точек;
- ***3D Bar Plot*** – шаблон для гистограммы в трехмерном пространстве;
- ***Vector Field Plot*** – шаблон для графика векторного поля.

Графики перемещаются путем перетаскивания за границу. Для изменения размеров нужно воспользоваться маркерами на границах блока графика.

В данной лабораторной работе рассмотрим только построение двухмерных и трехмерных графиков в декартовых координатах.

Для построения двухмерного графика нужно выполнить команду ***Insert/Graph/X-Y Plot***. Команда создает шаблон графика по текущему местоположению курсора. Перед применением этой команды необходимо определить функции, графики которых должны строиться, и диапазон изменения их аргумента.

Незаполненный шаблон графика представляет собою пустой прямоугольник с шаблонами данных в виде черных маленьких прямоугольников, расположенных около осей абсцисс и ординат. В средние шаблоны данных надо поместить имя переменной (около оси абсцисс) и задать формулы или имя для функции (около оси ординат). Если строятся графики нескольких функций на одном графике, то эти функции должны быть указаны через запятые.

Крайние шаблоны данных служат для указания предельных значений абсцисс и ординат, что позволяет задать масштабы графика. Если оставить эти шаблоны незаполненными, то масштабы по осям графика установятся автоматически.

Для построения графика в автоматическом режиме вычислений достаточно вывести курсор за пределы графика. В ручном режиме вычислений нужно нажать клавишу «F9».

Изменение внешнего вида графика производится путем его форматирования с помощью команды ***Format/Graph/X-Y Plot*** или двойным щелчком левой кнопкой мыши по графику. Диалоговое окно имеет несколько закладок. Кратко рассмотрим назначение закладок и основных элементов управления на них.

Закладка **X-Y Axis** позволяет изменять параметры осей:

- **Log Scale** – установка логарифмического масштаба;
- **Crid Lines** – установка линий масштабной сетки;

- **Numbered** – установка цифровых данных по осям,
- **Autoscale** – автомасштабирование графика по соответствующей оси;
- **Show Markers** – установка делений по осям;
- **Auto Grid** – автоматическая установка масштабных линий;
- **Number of Grids** – установка заданного числа масштабных линий.

Если опция **Grid Lines** отключена, то масштабная сетка графика не строится, хотя на осях размещаются короткие деления.

Возможна также установка стиля координатных осей (**Axes Style):**

- **Boxed** – оси в виде прямоугольника;
- **Crossed** – обычные оси в виде креста;
- **None** – отсутствие осей;
- **Equal Scales** – установка равенства масштабов по осям графика.

Закладка **Traces** позволяет форматировать кривые графика:

- **Legend Label** – указание типа линий у оси ординат;
- **Symbol** – выбор символа точки графика (none – без символов, x's – косой крест, +'x – прямой крест, box – квадрат, dmnd – ромб, o's – окружность);
- **Line** – установка типа линий (solid – непрерывная линия, dot – линия из точек, dash – пунктирная линия, dadot – штрихпунктирная линия);
- **Color** – цвет линий;
- **Type** – тип графиков (lines – линия, points – точки, error – вертикальные черточки с оценкой интервала погрешностей, bar – в виде столбцов гистограмм, step – ступенчатая линия, draw – построение линии от точки до точки, stem – в виде вертикальных черточек, solidbar – в виде сплошных столбцов);
- **Weight** – толщина линий;
- **Hide Argument** – скрытие обозначения аргументов по осям графика;
- **Hide Legend** – скрытие обозначения кривых графика.

Закладка **Labels** позволяет размещать дополнительные надписи у осей:

- **Title** – установка титульной надписи к рисунку;
- **X-Axis** – установка надписи по оси X;
- **Y-Axis** – установка надписи по оси Y.
- **Show Title** – разрешает отображать или не отображать титульную надпись.

Закладка **Defaults** позволяет установить параметры по умолчанию.

Для построения трехмерного графика выполните команду **Insert/Graph/Surface Plot.** Эта команда служит для построения трехмерной поверхности *z (x, y),* предварительно представленной матрицей M ординат *z*. При этом выводится шаблон графика, левый верхний угол которого помещается в место расположения курсора. Шаблон в свою очередь содержит единственный шаблон данных – темный прямоугольник у левого нижнего угла основного шаблона. В него надо занести имя матрицы со значениями ординат 3D-поверхности. Прежде чем строить график 3D-поверхности, ее нужно определить математически.

Поскольку график строится на основе матрицы, содержащей только координаты высот фигуры, то истинные масштабы по осям X и Y не известны и на рисунках не проставляются. При другом способе задания трехмерных поверхностей нужно формировать три матрицы *x, y* и *z* и указывать их в шаблоне в виде (*x, y, z*).

Для форматирования графика поверхности используется команда ***Format/Graph/3D Plot...***. В диалоговом окне имеется 9 закладок. Рассмотрим назначение элементов управления на некоторых наиболее часто используемых закладках.

Закладка **General** позволяет установить общие параметры поверхности:

- **View** – установка внешнего вида поверхности (**Rotation** – задание угла поворота от 0 до 180 градусов, **Tilt** здание угла наклона от 0 до 90 градусов);
- **Axes Style** задает тип отображения осей (**Perimeter** – по периметру, **Corner** – в углу, **None** – без вывода осей).

Закладка **Axes** позволяет форматировать координатные оси:

- **Draw Lines** – вывод масштабных линий;
- **Auto Grid** – автоматический выбор числа линий.

Вычисления в MathCAD

По умолчанию MathCAD работает в режиме автоматических вычислений. Это значит, что после вывода указателя из блока формул или графика блок автоматически пересчитывается, а график перестраивается. Однако иногда бывает удобнее работать в ручном режиме, например, при редактировании сложных выражений, вычисления по которым осуществляются долго. Команда ***Math/Automatic Calculation*** обеспечивает переключение между ручным и автоматическим режимами вычислений.

Вычисления в ручном режиме запускаются командой ***Math/Calculate*** (нажатием клавиши «F9»).

С помощью команды ***Math/Options…*** задается точность вычислений.

В диалоговом окне команды на вкладке **Built-in Variables** устанавливаются значения системных переменных. Действие системных переменных глобальное (они доступны в любом месте программы и их значения можно изменить также в любом месте программы).

При организации циклических вычислений или для перебора элементов векторов и матриц может понадобиться задавать значения переменной, меняющейся с определенным шагом от некоторого начального до конечного значения. Присваивание значения такой переменной задается следующей языковой конструкцией:

ИмяПеременной:= Start, Start + Step..End

Здесь ***Start*** – начальное значение переменной, ***Start + Step*** – значение второго элемента последовательности как сумма начального значения и шага, ***End*** – конечное значение. Отметим, что после ввода с клавиатуры значения ***Start + Step*** необходимо нажать на клавишу «;», что приведет к отображению на экране многоточия из двух точек (последовательный ввод с клавиатуры двух

точек не является эквивалентным действием). Пример задания значений переменной с шагом приведен на рисунке 1.5.

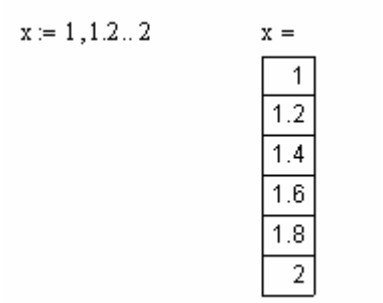

Рисунок 1.5 – Пример задания значения переменной с шагом

Значение ***Start + Step*** с предшествующей запятой может опускаться, тогда шаг будет равен единице. Если ***Start*** < ***End***, то шаг изменения переменной положительный, если ***Start*** > ***End***, то отрицательный.

Для установки вида полученных результатов используется команда ***Format/Result…*** . На вкладке **Number Format** задается числовой формат и число знаков после запятой (**Number of decimal places**). На вкладке **Display options** в поле со списком **Imaginary value** задается символ мнимой единицы для комплексных чисел *i* или *j*.

Установка системы единиц

Для установки требуемой системы единиц нужно выполнить команду ***Math/Options…,*** после чего на вкладке **Unit System** выбрать требуемую систему единиц. По умолчанию используется система СИ (SI).

Для установки требуемой размерности при определении констант и переменных необходимо установить указатель в конец выражения и выполнить команду **Insert/Unit…** . В диалоговом окне команды в списке **Dimension** нужно выбрать группу размерностей, к которой принадлежит единица, а в списке **Unit** выбрать требуемую единицу, после чего нажать кнопку **Insert** или **ОК**. К выражению после знака умножения будет добавлено обозначение единицы.

При работе с размерными величинами MathCAD выводит вычисленные числовые значения величин вместе с единицами их измерения.

Символические вычисления

Для символических вычислений используется символьный процессор, команды которого выполняются из меню ***Symbolics***. Перед выполнением команд необходимо выделить выражение или конкретную переменную выражения, над которыми эта операция должна быть выполнена.

Кратко рассмотрим наиболее часто используемые операции над выбранным выражением:

- ***Simplify*** – упростить выражение с выполнением таких операций, как сокращение подобных слагаемых, приведение к общему знаменателю, основные тождества для тригонометрических и обратных функций;
- ***Expand*** – раскрыть выражение (например, для выражения $(x + y)(x – y)$ результат операции будет $x^2 – y^2$);
- ***Factor*** – разложить выражение на множители (например, для выражения $x^2 – y^2$ результат операции будет $(x + y)(x – y)$).

С выделенными переменными наиболее часто используются следующие операции из пункта меню ***Symbolics/Variable***:

- ***Solve*** – решить уравнение или неравенство относительно выделенной переменной;

• ***Differentiate*** – дифференцировать выражение по выделенной переменной (остальные переменные рассматриваются как константы);

• ***Integrate*** – интегрировать выражение по этой переменной (возвращает символьное выражение для неопределенного интеграла);

• ***Expand to Series...*** – найти несколько членов разложения выражения в ряд Тейлора относительно выделенной переменной.

1.2 Порядок выполнения работы

1 Получить задание у преподавателя.

2 В соответствии с полученным заданием осуществить ввод математических выражений, построение таблиц значений, построение двух- и трехмерных графиков, проведение символических вычислений.

3 Продемонстрировать преподавателю форматирование результатов вычислений и графиков.

4 Продемонстрировать преподавателю использование размерностей при вычислениях.

5 Подготовить отчет по лабораторной работе.

1.3 Содержание отчета

1 Название работы и цель работы.

2 Исходные данные.

3 Результаты расчетов, таблицы и графики.

4 Анализ результатов и выводы.

1.4 Контрольные вопросы

1 Какой порядок интерпретации блоков на листе документа MatCAD?

2 Какие символы допускается использовать в именах идентификаторов?

3 Различаются ли в идентификаторах строчные и прописные буквы?

4 Как создать пользовательскую функцию?

5 Как определить матрицу или вектор?

6 Как просмотреть определенную часть графика в другом масштабе?

7 Как нанести на графике точки, соответствующие табличным данным?

8 Как осуществить символическое дифференцирование функции по заданной переменной?

9 Как представить на одном графике семейство кривых?

10 Как вывести таблицу значений?

Литература

1 Очков, В. MathCAD 12 для студентов и инженеров / В. Очков. – СПб. : БХВ-Петербург, 2005.

2 Половко, В. MathCAD для студента / В. Половко. – СПб. : БХВ-Петербург, 2005.

ЛАБОРАТОРНАЯ РАБОТА №2. ИЗУЧЕНИЕ МЕТОДОВ ИНТЕРПОЛЯЦИИ И АППРОКСИМАЦИИ ДАННЫХ

Цель: изучить методы интерполяции, используемые в САПР, изучить метод наименьших квадратов, использовать его для аппроксимации данных.

2.1 Теоретические сведения

Постановка задачи интерполяции и виды интерполяции

Задача интерполяции может быть сформулирована следующим образом. Пусть на отрезке $[a, b]$ заданы $n + 1$ опорные точки x_i (узлы интерполяции), причем $a \le x_0 < x_1 < \ldots < x_n \le b$, а также значения некоторой функции $y_i = f(x_i)$ в этих точках или некоторые данные y_i, соответствующие этим точкам. Требуется построить интерполяционную функцию $j(x)$, позволяющую вычислить значения функции $f(x)$ между узловыми точками, причем в узлах интерполяции интерполяционная функция должна принимать значения y_i, т.е.

$$j(x_0) = y_0,\ j(x_1) = y_1,\ \ldots,\ j(x_n) = y_n. \qquad (2.1)$$

Рассмотрим следующие виды интерполяции:

- глобальная интерполяция, при которой строится интерполяционный полином, проходящий через все точки (x_i, y_i) для всего отрезка интерполяции;
- локальная или кусочная интерполяция, при которой соседние точки соединяются прямолинейными или параболическими отрезками;
- сплайн-интерполяция, обеспечивающая гладкое сопряжение в узловых точках.

Глобальная интерполяция

В этом случае интерполяционная функция $j(x)$ ищется в виде полинома $P_n(x)$ степени не большей n, причем $P_n(x_i) = y_i$. Существует только один интерполяционный полином, который может быть представлен в различной форме. В форме Лагранжа интерполяционный полином ищется в следующем виде:

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)(x - x_1)\,\mathbf{K}\,(x - x_{i-1})(x - x_{i+1})\,\mathbf{K}\,(x - x_n)}{(x_i - x_0)(x_i - x_1)\,\mathbf{K}\,(x_i - x_{i-1})(x_i - x_{i+1})\,\mathbf{K}\,(x_i - x_n)}. \qquad (2.2)$$

В качестве примера рассмотрим построение полинома Лагранжа для $n = 2$. В этом случае получим уравнение параболы, проходящей через три точки x_0, x_1, x_2:

$$L_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2. \qquad (2.3)$$

Существуют другие формы составления интерполяционного полинома (например форма Ньютона) [1]. Однако при условии точных вычислений все они дадут одинаковые коэффициенты искомого полинома.

Интерполяция полиномом высокого порядка на отрезке с относительно большим числом узловых точек может давать значительные колебания на концах отрезка, что искажает поведение реальной функции.

Локальная интерполяция

Наиболее часто используемым видом локальной интерполяции является линейная интерполяция, что связано с простотой вычислений. В этом случае точки с координатами (x_i, y_i) соединяются прямолинейными отрезками. В общем случае для каждого из n интервалов интерполяции (x_{i-1}, x_i) в пределах отрезка $[a,b]$ уравнения прямой будут разные. При линейной интерполяции строится интерполяционный полином первой степени:

$$j(x)=a_i x + b_i, \quad x_{i-1}<x<x_i, \quad i=1,2,\ldots n.$$

Коэффициенты полинома находятся из уравнений

$$\begin{cases} a_i x_{i-1} + b_i = y_{i-1}, \\ a_i x_i + b_i = y_i. \end{cases}$$

Решив систему уравнений, можно получить выражения для определения коэффициентов полинома:

$$a_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \quad b_i = y_i - a_i x_i. \qquad (2.4)$$

При использовании линейной интерполяции сначала нужно определить интервал, в который попадает требуемое значение аргумента x, а затем подставить его в формулы для определения коэффициентов и найти приближенное значение функции в этой точке.

Для повышения точности интерполяции можно использовать квадратичную интерполяцию, при которой интерполяционный полином второй степени строится по трем соседним точкам и эти точки соединяются параболой.

Недостатком кусочной интерполяции является ломаный характер общей интерполирующей кривой, вследствие чего имеются разрывы производной в узловых точках.

Сплайн-интерполяция

Сплайном называется непрерывная функция, принимающая в узлах интерполяции значения $y_0, y_1, \ldots, y_n$, и описываемая на отдельных отрезках $[x_{i-1}, x_i]$ $(i=1,2..n)$ некоторыми полиномами $P_i(x)$ невысокого порядка (на практике чаще всего третьего). Сплайны хороши тем, что дают возможность получить приближенный аналитический вид функции, при этом степень полинома остается невысокой и в узловых точках сопряжение сегментов интерполирующей кривой – гладкое. Главным недостатком сплайнов является то, что на каждом отрезке $[x_{i-1}, x_i]$ функция приближается отдельным полиномом. С этой точки зрения сплайн-интерполяция относится к локальной интерполяции.

Наиболее часто на практике используется кубический сплайн. При построении кубического сплайна необходимо на каждом интервале интерполяции определять коэффициенты полинома

$$P_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i.$$

Коэффициенты кубического сплайна определяются из системы уравнений, в которую входят два уравнения для конечных точек отрезка $[x_{i-1}, x_i]$ и два

уравнения, требующие равенства в узловых точках первых и вторых производных. Для полиномов кубического сплайна коэффициенты могут быть получены по следующим выражениям (возможны другие варианты вычисления коэффициентов [1]):
для первого интервала

$$a_1 = \frac{\frac{y_1 - y_0}{x_1 - x_0} - 0{,}5 f''(x_0)(x_1 - x_0) - f'(x_0)}{(x_1 - x_0)^2}, \qquad b_1 = \frac{1}{2} f''(x_0) - 3a_1 x_0,$$

$$c_1 = f'(x_0) - 3a_1 x_0^2 - 2b_1 x_0, \qquad d_1 = y_0 - a_1 x_0^3 - b_1 x_0^2 - c_1 x_0, \qquad (2.5)$$

где значения $f'(x_0)$ и $f''(x_0)$ должны быть заданы. Если эти значения неизвестны, то на практике часто полагают их равными нулю;
для последующих i-х интервалов

$$a_i = \frac{\frac{y_i - y_{i-1}}{x_i - x_{i-1}} - g_i(x_i - x_{i-1}) - b_i}{(x_i - x_{i-1})^2}, \quad b_i = g_i - 3a_i x_{i-1}, \quad c_i = b_i - 3a_i x_{i-1}^2 - 2b_i x_{i-1},$$

$$d_i = y_{i-1} - a_i x_{i-1}^3 - b_i x_{i-1}^2 - c_i x_{i-1}, \qquad (2.6)$$

где $b_i = 3a_{i-1}x_{i-1}^2 + 2b_{i-1}x_{i-1} + c_{i-1}$, $g_i = 3a_{i-1}x_{i-1} + b_{i-1}$.

При практической реализации сплайн-интерполяции необходимо рассчитать векторы коэффициентов полиномов на всех интервалах, а затем определить интервал для интерполируемого значения и произвести расчет искомого значения.

Использование MathCAD для интерполяции

Рассмотрим некоторые функции MathCAD, используемые для интерполяции.

Функция ***linterp*** используется для линейной интерполяции. Функция соединяет точки данных отрезками прямых, создавая ломаную линию. Синтаксис функции следующий:

***linterp*(*vx*,*vy*,*x*)**

Здесь ***vx*,*vy*** – векторы данных одинаковой длины; ***x*** – точка, в которой требуется интерполировать значение ***y***. Вектор ***vx*** должен содержать вещественные значения, расположенные в порядке возрастания.

Для значений ***x***, расположенных перед первой точкой в векторе ***vx***, MathCAD продолжает ломаную прямой линией, проходящей через первые две точки данных. Для значений ***x***, расположенных за последней точкой ***vx***, MathCAD продолжает ломаную прямой линией, проходящей через последние две точки данных, однако эта функция не предназначена для экстраполяции данных.

Функции кубической сплайн-интерполяции

Для кубической сплайн-интерполяции имеются следующие функции: ***cspline*(*vx*,*vy*), *pspline*(*vx*,*vy*), *lspline*(*vx*,*vy*), *interp*(*vs*,*vx*,*vy*,*x*).**

Три первые функции возвращают вектор коэффициентов вторых производных ***vs***. Этот вектор используется в функции ***interp***. Аргументы ***vx,vy*** должны быть вещественными векторами одинаковой длины. Значения вектора ***vx*** должны быть расположены в порядке возрастания. Эти три функции отличаются только граничными условиями:

- функция ***lspline*** генерирует кривую сплайна, которая приближается к прямой линии в граничных точках;
- функция ***pspline*** генерирует кривую сплайна, которая приближается к параболе в граничных точках;
- функция ***cspline*** генерирует кривую сплайна, которая может быть кубическим полиномом в граничных точках.

Функция ***interp*** возвращает интерполируемое значение ***y***, соответствующее аргументу ***x***. Вектор вычисляется на основе векторов данных и одной из функций ***pspline, lspline*** или ***cspline***.

Чтобы построить кубический сплайн через набор точек, необходимо:

- создать векторы ***vx*** и ***vy***, содержащие координаты ***x*** и ***y***, через которые нужно провести кубический сплайн. Элементы должны быть расположены в порядке возрастания. В качестве имен векторов могут быть любые имена, соответствующие ограничениям MathCAD;
- с помощью функций ***cspline*(*vx,vy*), *pspline*(*vx,vy*), *lspline*(*vx,vy*)** вычислить вектор ***vs,*** который содержит вторые производные интерполяционной кривой в рассматриваемых точках;
- для нахождения интерполируемого значения в произвольной точке ***x*** вычислить функцию ***interp*(*vs*,*vx,vy,x*)**.

При вычислениях для реализации выбора по условию можно использовать функцию ***if***, которая имеет следующий синтаксис:

if (cond , x , y)

Функция возвращает значение ***x***, если ***cond*** отличен от 0 (истина), либо значение ***y***, если ***cond*** равен 0 (ложь).

Пример использования функции ***if*** при построении полинома Лагранжа приведен на рисунке 2.1.

Аппроксимация

Задача аппроксимации – представление произвольной сложной функции *f(x)* более простой и удобной для практического использования аппроксимирующей функцией *j (x)* таким образом, чтобы отклонение *j (x)* от *f(x)* на заданном отрезке *[a,b]* было минимальным по определенному критерию приближения. При этом в отличие от задачи интерполяции значения функции *j (x)* могут отличаться от значений функции *f(x)* в заданных точках.

Наиболее распространенным методом аппроксимации данных является метод наименьших квадратов (МНК).

$$x:\begin{pmatrix}0\\1\\2\\5\\6\\9\\10\\12\\14\end{pmatrix}\qquad y:\begin{pmatrix}0\\1\\3\\18\\12\\3\\5\\6\\1\end{pmatrix}$$

$$i := 0..8 \qquad j := 0..8$$

$$p := 0, 0.2..14$$

$$f(p): \sum_i y_i \prod_j \text{if}\left(i = j, 1, \frac{p - x_j}{x_i - x_j}\right)$$

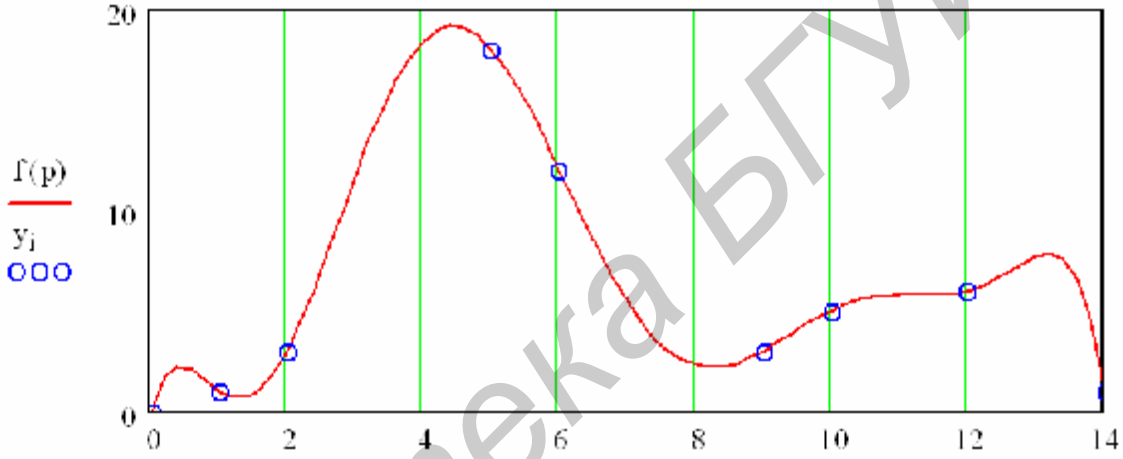

Рисунок 2.1 – Пример использования функции ***if*** при построении полинома Лагранжа

Пусть функция $y = f(x)$ задана таблицей своих значений: $y_i = f(x_i)$, $i = 0,1,\ldots n$. Критерием близости в методе наименьших квадратов является требование минимальности среднего квадратического отклонения (СКО):

$$s = \sqrt{\frac{1}{n+1}\sum_{i=0}^{n}(y_i - j(x_i))^2}, \qquad (2.7)$$

откуда следует требование минимальности суммы квадратов отклонений от аппроксимирующей функции до экспериментальных точек:

$$\Phi = \sum_{i=0}^{n}(y_i - j(x_i))^2 \to \min. \qquad (2.8)$$

Таким образом, в отличие от задачи интерполяции не требуется, чтобы аппроксимирующая функция проходила через все заданные точки, что особенно важно при аппроксимации экспериментальных данных, содержащих погрешности.

Важной особенностью метода является то, что аппроксимирующая функция может иметь различный вид, который определяется особенностями решаемой задачи (например физическими соображениями, если проводится

аппроксимация результатов физического эксперимента). Наиболее часто встречаются аппроксимация прямой линией (линейная регрессия), аппроксимация полиномом (полиномиальная регрессия), аппроксимация линейной комбинацией произвольных функций.

При аппроксимации функции прямой по МНК задан набор точек и значений функции в них, и предполагается, что эти точки должны лежать на одной прямой. При этом число точек может быть произвольным, намного больше двух, и поэтому обычно нельзя провести прямую линию через все точки одновременно.

Из всех прямых $j\ (x) = ax + b$ выбирается та, для которой сумма квадратов отклонений заданных значений функции от этой прямой минимальна, т.е. минимизируется функция

$$F(a,b) = \sum_{i=0}^{n} (y_i - ax_i - b)^2 \to \min.$$

Для поиска минимума приравняем к нулю производные $\partial F / \partial a$ и $\partial F / \partial b$:

$$\begin{cases} \dfrac{\partial F}{\partial a} = -2\sum\limits_{i=0}^{n} x_i (y_i - ax_i - b) = 0, \\ \dfrac{\partial F}{\partial b} = -2\sum\limits_{i=0}^{n} (y_i - ax_i - b) = 0. \end{cases}$$

Имеем систему из двух уравнений с двумя неизвестными. Для ее решения можно использовать численные методы решения систем линейных уравнений. Однако для такого простого случая можно аналитически получить значения для коэффициентов.

$$a = \frac{(n+1)\sum\limits_{i=0}^{n} x_i y_i - \sum\limits_{i=0}^{n} x_i \sum\limits_{i=0}^{n} y_i}{(n+1)\sum\limits_{i=0}^{n} x_i^2 - (\sum\limits_{i=0}^{n} x_i)^2}, \quad b = \frac{\sum\limits_{i=0}^{n} x_i^2 \sum\limits_{i=0}^{n} y_i - \sum\limits_{i=0}^{n} x_i \sum\limits_{i=0}^{n} x_i y_i}{(n+1)\sum\limits_{i=0}^{n} x_i^2 - (\sum\limits_{i=0}^{n} x_i)^2}. \qquad (2.9)$$

Аппроксимация полиномом с помощью МНК. Пусть функция $y = f(x)$ задана таблицей своих значений: $y_i = f(x_i),\ i = 0,1, \ldots n$. Требуется найти полином фиксированной степени m, для которого СКО

$$s = \sqrt{\frac{1}{n+1}\sum_{i=0}^{n} (P_m(x_i) - y_i)^2}$$ минимально.

Так как многочлен $P_m(x)=a_0+a_1x+a_2x^2+\ldots+a_mx^m$ определяется своими коэффициентами, то нужно подобрать набор коэффициентов $a_0,a_1,a_2\ldots a_m$, минимизирующий функцию

$$\Phi(a_0, a_1, \ldots a_m) = \sum_{i=0}^{n} (P_m(x_i) - y_i)^2 = \sum_{i=0}^{n} \left(\sum_{j=0}^{m} a_j x_i^j - y_i \right)^2.$$

Используя необходимое условие экстремума $\frac{\partial \Phi}{\partial a_k} = 0$, *k=0,1,...m,* получаем так называемую нормальную систему метода наименьших квадратов:

$$\sum_{j=0}^{m}\left(\sum_{i=0}^{n} x_i^{j+k}\right)a_j = \sum_{i=0}^{n} y_i x_i^k, \qquad k=0,1,...m. \tag{2.10}$$

Полученная система есть система линейных алгебраических уравнений относительно неизвестных $a_0,a_1,a_2 \ldots a_m$. Можно показать, что определитель этой системы отличен от нуля, т.е. решение существует и единственно. Однако при высоких степенях *m* система является плохо обусловленной. Поэтому МНК обычно применяют для нахождения многочленов, степень которых не выше 5.

Для полинома второй степени $P_2(x)=a_0+a_1x+a_2x^2$ нормальная система уравнений примет следующий вид:

$$\begin{cases} (n+1)a_0 + \left(\sum_{i=0}^{n} x_i\right)a_1 + \left(\sum_{i=0}^{n} x_i^2\right)a_2 = \sum_{i=0}^{n} y_i, \\ \left(\sum_{i=0}^{n} x_i\right)a_0 + \left(\sum_{i=0}^{n} x_i^2\right)a_1 + \left(\sum_{i=0}^{n} x_i^3\right)a_2 = \sum_{i=0}^{n} y_i x_i, \\ \left(\sum_{i=0}^{n} x_i^2\right)a_0 + \left(\sum_{i=0}^{n} x_i^3\right)a_1 + \left(\sum_{i=0}^{n} x_i^4\right)a_2 = \sum_{i=0}^{n} y_i x_i^2. \end{cases}$$

Использование MathCAD для аппроксимации

Для аппроксимации данных прямой линией можно использовать функции ***slope(vx,vy)*** и ***intercept(vx,vy).*** Функция ***slope*** определяет угловой коэффициент прямой, а функция ***intercept*** – точку пересечения графика с вертикальной осью. В качестве аргументов функций задаются векторы значений ***x*** и ***y***, при этом размеры векторов должны совпадать.

Для этих же целей в Mathcad 2000 и более поздних версиях можно использовать функцию ***line(vx, vy)***, которая возвращает вектор коэффициентов прямой $a+b\cdot x$.

Для аппроксимации полиномами можно использовать функции ***regress(vx,vy,k)*** и ***interp(vs,vx,vy,x)***.

Векторы ***vx,vy*** имеют то же назначение, что и в ранее рассмотренных функциях. Аргумент ***k*** является порядком (степенью) полинома. Функция генерирует вектор ***vs,*** содержащий в том числе и коэффициенты полинома. Функция ***regress*** является вспомогательной, она готовит данные, необходимые для работы функции ***interp***. Назначение аргументов функции ***interp*** было рассмотрено ранее (она возвращает значение полинома в точке *x*).

Полиномиальную аппроксимацию можно также провести без использования функций ***regress*** и ***interp***. В этом случае нужно определить коэффициенты нормальной системы (2.10) и решить полученную систему уравнений матричным методом. Последовательность действий при таком подходе следующая:

1 Вычислить элементы матрицы коэффициентов нормальной системы (матрица размерностью (*m+1)* x (*m+1*)) $X = \sum_{i=0}^{n} x_i^{j+k}$.

2 Вычислить вектор *m+1* коэффициентов свободных членов $B = \sum_{i=0}^{n} y_i x_i^k$.

3 Найти коэффициенты полинома, решив систему матричным методом $A := X^{-1} \times B$.

При необходимости аппроксимации произвольной функцией с *m* неизвестными параметрами можно воспользоваться функцией ***genfit(vx,vy,vg,F).*** Аргументы функции имеют следующее назначение: ***vx,vy*** – векторы, содержащие координаты заданных точек; ***F*** – вектор из *m*+1 элемента, который содержит значение функции, задающей искомую функциональную *m*-параметрическую зависимость и значения производных по всем параметрам; ***vg*** – вектор из *m* элементов начальных значений *m* параметров для инициализации функции.

Использование функции рассмотрим на примере построения аппроксимирующей функции дробно-рационального типа $f(x) = \frac{u_0 x^2}{u_1 + x}$ (рисунок 2.2).

2.2 Порядок выполнения работы

1 Получить задание у преподавателя.

2 С использованием формул (2.4) и функции ***linterp*** провести линей-ную интерполяцию. Построить график, содержащий исходные данные (кружки), результаты интерполяции.

3 Составить формулу интерполяционного полинома Лагранжа, ис-пользуя операторы суммирования и перемножения по дискретному аргументу, а также функцию ***if***. Построить график интерполяционного полинома и нанести на него исходные данные.

4 Провести сплайн-интерполяцию с помощью функций ***lspline***, ***pspline***, ***cspline*** и ***interp***. Построить график функции ***interp*** и и нанести на него исходные данные. Рассчитать коэффициенты кубического сплайна в соответствии с формулами (2.5–2.6). Построить график сплайна на одном из отрезков интерполяции.

5 Провести аппроксимацию данных прямой линией с использованием функций ***slope(vx,vy), intercept(vx,vy)*** и ***line(vx, vy)***. Построить графики исходных данных и аппроксимирующих функций. По формуле (2.7) рассчитать значение СКО.

6 Провести аппроксимацию данных полинимом 2-, 3- и 4-й степени с использованием функций ***regress(vx,vy,k)*** и ***interp(vs,vx,vy,x)***. Построить

графики исходных данных и аппроксимирующих функций. По формуле (2.7) рассчитать значение СКО.

$$F(z,u) := \begin{bmatrix} \frac{u_0 \cdot z^2}{u_1 + z} \\ \frac{z^2}{u_1 + z} \\ \frac{-\left(u_0 \cdot z^2\right)}{\left(u_1 + z\right)^2} \end{bmatrix} \qquad vx := \begin{pmatrix} 0 \\ 2 \\ 4 \\ 6 \\ 8 \end{pmatrix} \qquad vy := \begin{pmatrix} 0 \\ 1 \\ 8 \\ 10 \\ 70 \end{pmatrix}$$

$$gv := \begin{pmatrix} -2 \\ -10 \end{pmatrix}$$

$$vu := genfit(vx, vy, gv, F) \qquad vu = \begin{pmatrix} -0.908 \\ -8.831 \end{pmatrix} \qquad x := 0, 0.1 .. 8$$

$$f(x) := \frac{\left(vu_0 \cdot x^2\right)}{vu_1 + x}$$

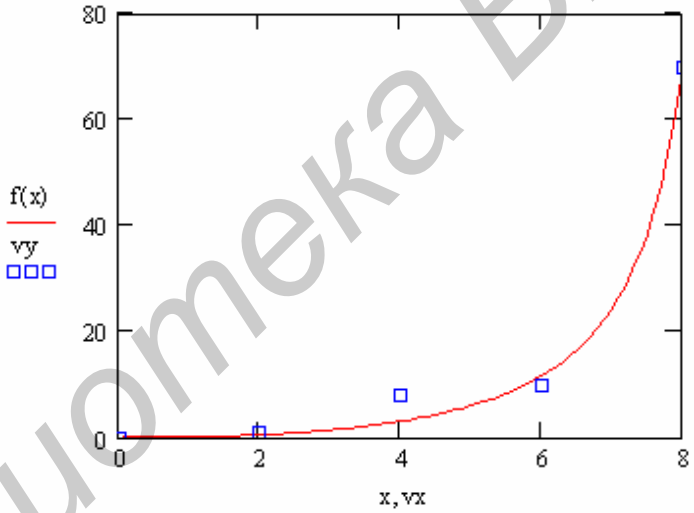

Рисунок 2.2 – Пример использования функции ***genfit***

7 Для полинома 3-й степени определить коэффициенты нормальной системы (2.10) и решить полученную систему уравнений матричным методом. Построить графики исходных данных и аппроксимирующей функции. По формуле (2.7) рассчитать значение СКО.

8 С помощью функции ***genfit(vx,vy,vg,F)*** аппроксимировать данные зависимостью $j(x) = e^{a+bx+cx^2}$. Построить графики исходных данных и аппроксимирующей функции. По формуле (2.7) рассчитать значение СКО.

2.3 Содержание отчета

1 Название работы и цель работы.

2 Исходные данные.

3 Результаты расчетов, таблицы и графики.

4 Анализ результатов и выводы о наиболее предпочтительном виде интерполяционной и аппроксимирующей функции.

2.4 Контрольные вопросы

1 Какой степени можно построить интерполяционный полином при глобальной интерполяции по 8 точкам?

2 Чем отличается сплайн-интерполяция от линейной интерполяции?

3 Могут ли быть одинаковыми коэффициенты кубического полинома на разных интервалах при сплайн-интерполяции?

4 Какие недостатки имеет глобальная интерполяция?

5 Чем отличается аппроксимация от интерполяции?

6 Какой критерий используется в МНК для построения аппроксимирующей кривой?

7 Какой степени можно построить аппроксимирующий полином при аппроксимации по 8 точкам?

8 Накладывает ли МНК ограничения на вид аппроксимирующей функции?

Литература

1 Автоматизированное проектирование радиоэлектронных средств : учеб. пособие для вузов / О. В. Алексеев [и др.]; под ред. О. В. Алексеева. – М. : Высш. шк., 2000.

2 Бахвалов, Н. С. Численные методы в задачах и упражнениях / Н. С. Бахвалов, А. В. Лапин, Е. В. Чижонков. – М. : Высш. шк., 2000.

ЛАБОРАТОРНАЯ РАБОТА №3. ИЗУЧЕНИЕ МЕТОДА КОНЕЧНЫХ РАЗНОСТЕЙ

Цель: изучить метод конечных разностей и использовать его для анализа процессов переноса теплоты теплопроводностью в ЭВС.

3.1 Теоретические сведения

Конечно-разностные аппроксимации

Точное аналитическое решение краевых задач математической физики, описывающих физические поля в конструкциях ЭВС (электромагнитные, тепловые и т.д.), удается получить лишь для немногих частных случаев. В САПР решение дифференциальных уравнений производится численными методами. Одним из таких численных методов является метод конечных разностей.

Основная идея метода заключается в замене частных производных их разностными аппроксимациями.

Предположим, что имеется некоторая функция двух переменных ***F(x,z)***. Пусть известны значения функции в некоторых точках ***(x,z), (x+Dx,z), (x–Dx,z).*** Если ввести обозначения $F(x+Dx,z) = F_{i+1}$, $F(x,z) = F_i$, $F(x–Dx,z) = F_{i-1}$, то можно записать следующие аппроксимации частных производных данной функции:

$$\frac{\partial F}{\partial x} \approx \frac{F(x+\Delta x, z) - F(x, z)}{\Delta x} = \frac{F_{i+1} - F_i}{\Delta x}$$ – правая разностная схема, (3.1)

$$\frac{\partial F}{\partial x} \approx \frac{F(x, z) - F(x-\Delta x, z)}{\Delta x} = \frac{F_i - F_{i-1}}{\Delta x}$$ – левая разностная схема, (3.2)

$$\frac{\partial F}{\partial x} \approx \frac{F(x+\Delta x, z) - F(x-\Delta x, z)}{2\Delta x} = \frac{F_{i+1} - F_{i-1}}{2\Delta x}$$ – центральная разностная схема. (3.3)

Приведенные выражения могут быть получены из разложения функции ***F(x,z)*** в ряд Тейлора. Следует отметить, что центральная разность имеет меньшую погрешность аппроксимации.

При необходимости можно получить аппроксимацию производных более высоких порядков, так, для второй производной можно записать следующую аппроксимацию:

$$\frac{\partial^2 F}{\partial x^2} \approx \frac{\frac{F(x+\Delta x) - F(x)}{\Delta x} - \frac{F(x) - F(x-\Delta x)}{\Delta x}}{\Delta x} = \frac{F(x+\Delta x) - 2F(x) + F(x-\Delta x)}{\Delta x^2} = \frac{F_{i+1} - 2F_i + F_{i-1}}{\Delta x^2}.$$ (3.4)

Метод конечных разностей предполагает выполнение следующих шагов:

1. В исследуемой области строится сетка путем дискретизации области изменения аргумента. В результате получается конечное множество точек, отстоящих друг от друга на величину шага *Dx*. Чаще всего используется постоянный шаг сетки *Dx = const*. Искомая функция *F* аппроксимируется совокупностью значений в узлах сетки F_i (сеточной функцией).

2. В исходных дифференциальных уравнениях операторы $\partial F/\partial x$, $\partial^2 F/\partial x^2$ заменяются конечной разностью по одной из разностных схем (3.1–3.4). Записывается система уравнений с конечными разностями для точек сетки. Каждая точка сетки представляется шаблоном, отражающим свойства среды и физического поля. На рисунке 3.1 представлены шаблоны для случаев одной и двух координат (переменных) в решаемой задаче.

Рисунок 3.1 – Шаблоны метода конечных разностей

На рисунке 3.1 номера точек сетки для одномерного случая являются значениями индекса *i,* для двухмерного случая – двойным индексом *(i,j),* соответствующим номеру дискретной точки сетки по первой и по второй координатам. Полученная система дополняется граничными и начальными условиями. Для производных в граничных условиях второго и третьего рода также используется аппроксимация конечной разностью. В результате будет получена замкнутая система в общем случае нелинейных алгебраических уравнений.

3. Полученная система алгебраических уравнений решается численно.

Рассмотрим применение метода конечных разностей для анализа процессов переноса теплоты теплопроводностью в ЭВС.

Краевая задача теплопроводности

Перенос теплоты теплопроводностью можно описать с помощью уравнения теплопроводности. Для изотропного тела и независимости коэффициента теплопроводности от координат уравнение теплопроводности имеет вид [1]

$$\frac{\partial^2 t}{\partial x^2}+\frac{\partial^2 t}{\partial y^2}+\frac{\partial^2 t}{\partial z^2}+\frac{q_v}{l}=\frac{1}{a}\frac{\partial t}{\partial t}\ , \qquad (3.5)$$

где t – температура; x,y,z – пространственные координаты; t – время; q_v – объемная плотность теплового потока внутренних источников теплоты Вт/м^3; λ – теплопроводность вещества Вт/(м·К); a – температуропроводность материала, характеризующая скорость распространения фронта температурной волны, м^2/с.

В стационарном режиме правая часть уравнения теплопроводности равна нулю.

Для нахождения температурного поля кроме дифференциального уравнения теплопроводности необходимо знать поле температур в начальный момент времени $t = 0$ (начальное условие), форму тела и закон теплообмена на границах тела (граничные условия), а также теплофизические свойства тела.

Начальное и граничные условия в совокупности называются краевыми условиями. Начальное условие определяется заданием закона распределения температуры в теле в начальный момент времени, т.е. $t(x,y,z,0)=f(x,y,z)$. Граничные условия представляются в различной форме в зависимости от характера теплообмена на границе тела. Различают следующие виды граничных условий [1]:

1. Граничное условие первого рода, при котором задается закон распределения температуры на поверхности тела в любой момент времени:

$$t_{\text{п}} = f(t). \tag{3.6}$$

2. Граничное условие второго рода, при котором задается закон изменения плотности теплового потока на границе $q_{\text{п}}(t)$ в любой момент времени:

$$q_{\text{П}}(t) = -l\left.\frac{\partial t}{\partial n}\right|_{\text{п}}, \tag{3.7}$$

где $-l\left.\frac{\partial t}{\partial n}\right|_{\text{п}}$ – плотность теплового потока, уходящего в глубь тела (закон Фурье); n – нормаль к поверхности тела.

3. Граничное условие третьего рода, когда на поверхности имеет место конвективный теплообмен тела с окружающей средой. На основании закона Ньютона–Рихмана плотность теплового потока на границе тело–среда равна

$$q_1(t) = a[t_{\text{Ж}}(t) - t_{\text{с}}(t)],$$

где α – коэффициент теплообмена, Вт/(м^2·К).

По закону Фурье к поверхности тела подходит поток, плотность которого равна

$$q_2(t) = -l\left(\left.\frac{\partial t}{\partial n}\right)_{\text{п}}\right..$$

Если на границе тело–среда отсутствуют стоки или источники энергии, то $q_1(t)=q_2(t)$ и граничное условие имеет вид

$$l_{\text{п}}\left(\left.\frac{\partial t}{\partial n}\right)_{\text{п}}\right. + a(t_{\text{П}} - t_{\text{с}}) = 0. \tag{3.8}$$

Для случая контакта двух твердых тел рассматривается граничное условие четвертого рода.

Рассмотрим несколько примеров применения метода конечных разностей к решению уравнения теплопроводности.

Решение одномерных стационарных задач

Рассмотрим стационарное распределение температуры в плоской стенке толщиной 3 мм. Такая модель хорошо описывает процесс теплопроводности, например, в стенке корпуса ЭВС в точках, где краевые эффекты оттока теплоты по краям стенки незначительны и задачу можно считать одномерной. Одна поверхность стенки находится при температуре 20 $^{\text{o}}$C, на другую поверхность падает тепловой поток удельной мощностью q = 10^4 Вт/м^2 . Теплопроводность стенки l = 1 Вт/(м· $^{\text{o}}$C).

Уравнение теплопроводности в этом случае будет иметь вид

$$\frac{\partial^2 t}{\partial x^2} = 0. \qquad (3.9)$$

Градиент температуры $\partial t/\partial x = q/l = 10$ °С /мм.

Отметим по координате x шаг сетки, равный 1 мм. По толщине стенки получим четыре точки (рисунок 3.2).

На основании уравнения теплопроводности, граничных условий и одномерного шаблона составим систему уравнений метода конечных разностей:

$$\begin{cases} t_0 = 20; \\ \dfrac{t_2 - 2t_1 + t_0}{\Delta x^2} = 0; \\ \dfrac{t_3 - 2t_2 + t_1}{\Delta x^2} = 0; \\ \dfrac{t_3 - t_2}{\Delta x} = 10. \end{cases}$$

Из данной системы уравнений можно найти значения температур $t_1 = 30$ °С, $t_2 = 40$ °С, $t_3 = 50$ °С.

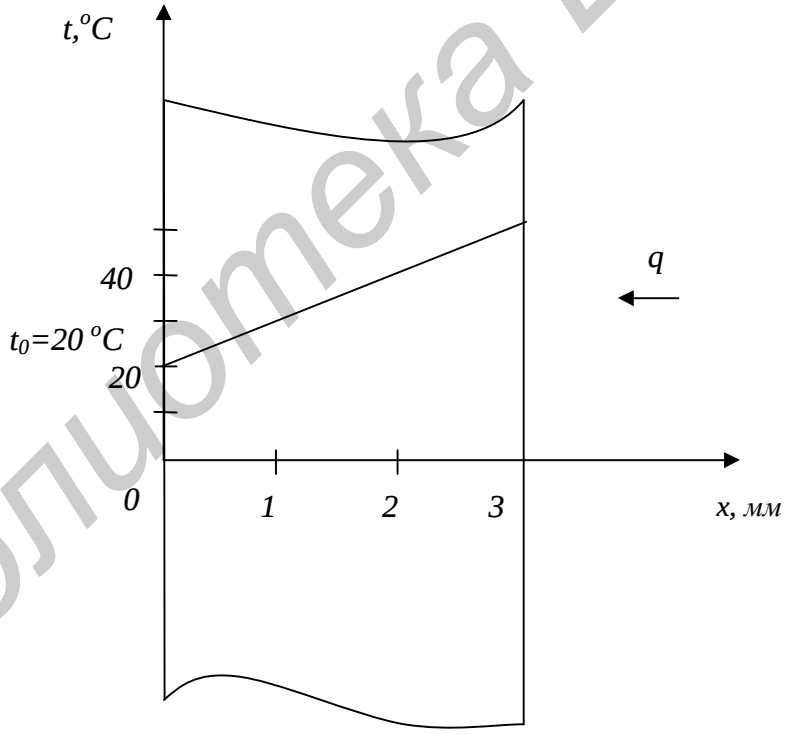

Рисунок 3.2 – Одномерная задача теплопроводности

Решение одномерных нестационарных задач

При исследовании временных зависимостей формирования физических полей в конструкциях ЭВС приходится решать нестационарные задачи.

Для решения нестационарных задач можно использовать явный и неявный методы. В явном методе последующие вычисления в следующих точках сетки базируются на результатах предыдущих вычислений. В неявном методе нужно

составить полную систему уравнений, которую затем численно решить. Явные методы по сравнению с неявными имеют большие ограничения по устойчивости (понятие устойчивости см. далее).

В одномерных задачах теплопроводности имеется две переменные – пространственная координата и время. Следовательно, сетку необходимо устанавливать по обеим переменным.

Рассмотрим варианты шаблонов для одномерной нестационарной задачи теплопроводности. В этом случае уравнение теплопроводности имеет вид

$$\frac{\partial t}{\partial t} = a\frac{\partial^2 t}{\partial x^2}. \qquad (3.10)$$

Можно использовать двухмерный шаблон из рисунка 3.1. В этом случае производная по времени должна аппроксимироваться центральной разностной схемой. При использовании правой и левой разностных схем применяется четырехточечный шаблон. Введем следующие обозначения: Dt – шаг по сетке времени; $t_{i,j}$ – значение температуры в точке i в момент времени j.

Для явного метода шаблон представлен на рисунке 3.3.

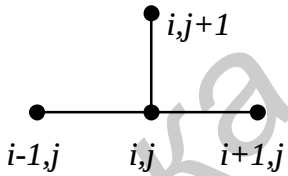

Рисунок 3.3 – Шаблон явного метода

В этом случае разностная аппроксимация дифференциального уравнения теплопроводности для i-й точки в момент времени j будет иметь следующий вид:

$$\frac{t_{i,j+1} - t_{i,j}}{\Delta t} = a\frac{t_{i+1,j} - 2t_{i,j} + t_{i-1,j}}{\Delta x^2}. \qquad (3.11)$$

Для явного метода значение температуры в следующий момент времени рассчитывается по значениям температуры в предыдущие моменты времени:

$$t_{i,j+1} = a\Delta t\,\frac{t_{i+1,j} - 2t_{i,j} + t_{i-1,j}}{\Delta x^2} + t_{i,j}.$$

Рассмотрим пример использования явного метода.

Имеем плоскую стенку толщиной 3 мм. В момент времени t = 0 одна поверхность стенки остается при начальной температуре t = 20 ^{0}C, другая начинает поддерживаться (термостатироваться) при температуре 80 ^{0}C. Начальное распределение температуры – равномерное с температурой $t(x,0)$ = = 20 ^{0}C. Температуропроводность примем равной 10^{-7} м^2/с. Шаг сетки по координате x выберем равным 1 мм, шаг сетки по времени выберем равным 1 с.

Результаты расчета представим в виде таблицы 3.1.

Таблица 3.1 – Результаты расчета для явного метода

j	*i*			
	0	1	2	3
0	20	20	20	80
1	20	20	26	80
2	20	20,6	30,8	80
3	20	21,56	34,64	80
4	20	22,7	33,7	80

Следует иметь в виду, что для данной задачи условие устойчивости вычислений имеет следующий вид: $\Delta t \le \frac{\Delta x^2}{2a}$.

В рассмотренном примере $Dx^2/2a = 5$. Проиллюстрируем неустойчивый вычислительный процесс, выбрав шаг сетки по времени $Dt = 10$ с. Результаты вычислений представлены в таблице 3.2.

Таблица 3.2 – Неустойчивый вычислительный процесс

j	*i*			
	0	1	2	3
0	20	20	20	80
1	20	20	80	80
2	20	80	20	80
3	20	-40	80	80
4	20	140	20	80

Для неявного метода используется шаблон, приведенный на рисунке 3.4.

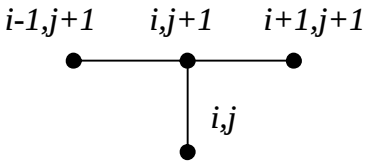

Рисунок 3.4 – Шаблон неявного метода

Разностная аппроксимация дифференциального уравнения теплопроводности для *i*-й точки в момент времени *j+1* для неявного метода будет иметь следующий вид:

$$\frac{t_{i,j+1} - t_{i,j}}{\Delta t} = a\frac{t_{i+1,j+1} - 2t_{i,j+1} + t_{i-1,j+1}}{\Delta x^2}. \qquad (3.12)$$

При такой аппроксимации необходимо составить систему уравнений для точек сетки, которую потом нужно будет решать численными методами.

В таблице 3.3 приведен результат расчета с использованием неявного метода для трех первых узлов сетки времени.

Таблица 3.3 – Результаты расчета для неявного метода

j	*i*			
	0	1	2	3
0	20	20	20	80
1	20	20,42	25,04	80
2	20	21,12	29,3	80

Существуют и другие разностные схемы, которые позволяют повысить точность решения за счет более сложной аппроксимации производных конечными разностями.

Как уже отмечалось, при исследовании нестационарных процессов в конструкциях ЭВС при определенных соотношениях шага сетки по пространственным и временным координатам вычислительный процесс может оказаться неустойчивым. Например, при очень малых шагах временной координаты по сравнению с шагом по пространственным координатам любая малая ошибка конечной разрядности или округления, сделанная на первом шаге интегрирования по времени, будет возрастать на последующих шагах. К концу вычисления эта ошибка может существенно перекрыть любые другие погрешности, связанные с кусочно-разностной аппроксимацией. Это явление называется вычислительной неустойчивостью процесса. Имеются строгие критерии оценки устойчивости решения. С практической точки зрения неустойчивость будет наблюдаться при некоторых отношениях *Dt /Dx*. Необходимо контролировать решение в зависимости от величины этого отношения.

Конечно-разностная аппроксимация сходится, если приближенное решение конечно-разностных уравнений стремится к точному решению задачи по мере последовательного уменьшения шага сетки при сохранении постоянным отношения пространственных шагов сетки по разным координатам. В вычислительной математике доказывается утверждение, что если конечно-разностная система устойчива, то она также сходится.

Погрешность аппроксимации производных конечной разностью может быть оценена с помощью ряда Тейлора. Например, для правой, левой разностных схем и аппроксимации второй производной можно получить следующие оценки погрешности [2]:

$$e_{пр} = -\left(\frac{\Delta x}{2!}\frac{\partial^2 V}{\partial x^2} + \frac{\Delta x^2}{3!}\frac{\partial^3 V}{\partial x^3} + \frac{\Delta x^3}{4!}\frac{\partial^4 V}{\partial x^4} + \right)\ldots,$$

$$e_{лев} = \left(\frac{\Delta x}{2!}\frac{\partial^2 V}{\partial x^2} - \frac{\Delta x^2}{3!}\frac{\partial^3 V}{\partial x^3} + \frac{\Delta x^3}{4!}\frac{\partial^4 V}{\partial x^4} + \right)\ldots,$$

$$e_2 = -\frac{\Delta x^4}{12}\frac{\partial^4 V}{\partial x^4} + \ldots .$$

Из оценок погрешности следует практическая рекомендация: если при анализе поля в конструкции ЭВС методом конечных разностей производные

высоких порядков окажутся достаточно большими (их также можно посчитать с помощью конечных разностей), то необходимо уменьшить шаг сетки и повторить вычисления.

Использование MathCAD для решения систем уравнений

После составления систем уравнений метода конечных разностей необходимо численно решать эти системы. MathCAD позволяет решать системы уравнений (в том числе и нелинейных) с помощью итерационных методов. Для этого используется блок решения уравнений ***Given*** и функция ***Find*** (рисунок 3.5). Число уравнений и переменных не должно превышать 50. Результатом решения системы будет численное значение искомого корня.

Алгоритм решения системы уравнений

Шаг 1. Задать начальное приближение для всех неизвестных, входящих в систему уравнений.

Шаг 2. После ключевого слова ***Given*** ввести систему уравнений. Порядок уравнений не принципиален. Для вывода символа равенства следует использовать клавиатурную комбинацию CTRL + клавиша «=» либо кнопочную панель ***Boolean***.

Шаг 3. Использовать функцию ***Find(var1, var2, ...)***, где ***var1, var2, ...*** - неизвестные системы уравнений. Функция возвращает вектор значений, представляющих собой решение системы.

$$x := 1 \qquad y := 1$$

Given

$$x^2 + y^2 = 6$$

$$x + y = 2$$

$$\text{Find}(x, y) = \begin{pmatrix} 2.414 \\ -0.414 \end{pmatrix}$$

Рисунок 3.5 – Пример решения системы уравнений

Блоки решения уравнений не могут быть вложены друг в друга, каждый блок может иметь только одно ключевое слово ***Given*** и одну функцию ***Find***.

В результате вычислений может быть выдано сообщение об ошибке: No solution was found (решение не найдено). Можно попробовать увеличить значение TOL (повысить точность решения), а также попробовать задать различные начальные приближения.

В случае если устраивает приближенное решение, то в блоке ***Given*** можно использовать функцию ***Minner***. Аргументы этой функции имеют тот же смысл, что и для функции ***Find***. В отличие от ***Find*** функция ***Minner*** возвращает последнее приближение к решению в случае невозможности его дальнейшего уточнения (функция ***Find*** в этом случае выдаст сообщение об ошибке).

3.2 Порядок выполнения работы

1 Получить задание у преподавателя.

2 Решить стационарную одномерную задачу теплопроводности мето-дом конечных разностей с заданными граничными условиями. Число точек сетки выбрать равным пяти–шести.

3 Решить нестационарную одномерную задачу теплопроводности яв-ным и неявным методом конечных разностей с заданными краевыми условиями. Сравнить полученные результаты. Построить графические зависимости изменения температуры от времени.

3.3 Содержание отчета

1 Название работы и цель работы.

2 Исходные данные.

3 Результаты расчетов, таблицы и графики.

4 Анализ результатов и выводы.

3.4 Контрольные вопросы

1 Что такое конечная разность?

2 Какие шаблоны используются при решении одномерных задач?

3 Явный и неявный метод решения нестационарных задач. Их различие.

4 Какая конечная разность для аппроксимации первой производной обладает большей точностью?

5 Как повысить точность решения краевой задачи теплопроводности?

6 Причины возникновения вычислительной неустойчивости решения?

7 Как аппроксимировать конечной разностью граничное условие третьего рода?

Литература

1 Лыков, А. В. Теория теплопроводности / А. В. Лыков. – М. : Высш. шк., 1967.

2 Деньдобренько, Б. Н. Автоматизация конструирования РЭА / Б. Н. Деньдобренько, А. С. Малика. – М. : Высш. шк., 1980.

ЛАБОРАТОРНАЯ РАБОТА №4. ИЗУЧЕНИЕ МЕТОДОВ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ ОПТИМАЛЬНОМ ПРОЕКТИРОВАНИИ

Цель: изучить методы решения задачи линейного программирования, используемые при проектировании ЭВС.

4.1 Теоретические сведения

Постановка задачи линейного программирования

Ряд задач проектирования ЭВС и разработки технологических процессов характеризуется линейными целевыми функциями и ограничениями. Такая задача оптимизации называется задачей линейного программирования (ЗЛП) и математически записывается следующим образом:

$$\min(\max)\ F = c_0 + \sum_{j=1}^{n} c_j x_j \quad \text{при ограничениях}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad (i = \overline{1,m}) \quad ; \qquad x_j \ge 0, \qquad (j = \overline{1,n}).$$

Переменные x_j в прикладных задачах обычно представляют собой некоторые физические величины, такие, как размеры, время и т.д. Поэтому чаще всего переменные x_j неотрицательные. Однако если переменные проектирования x_j имеют произвольный знак, то их всегда можно представить в виде разности двух неотрицательных переменных. Это приводит к увеличению размерности вектора переменных проектирования на единицу. Например, в ограничении $3x_1 + 4x_2 \le 20$ известно, что x_2 имеет произвольный знак. Тогда введем две новые переменные x_3 и x_4. Ограничение перепишется в следующем виде: $3x_1 + 4x_3 - 4x_4 \le 20$, $x_j \ge 0$.

Каноническая форма (канонический вид) записи ЗЛП:

$$\min(\max)\ F = c_0 + \sum_{j=1}^{n} c_j x_j, \quad \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad (i = \overline{1,m}); \quad x_j \ge 0, \quad (j = \overline{1,n}).$$

В качестве примера постановки задачи линейного программирования рассмотрим задачу о назначениях [1]. Пусть имеется n мест на плате для размещения n элементов, причем на каждом месте может быть размещена только одна микросхема. Эффективность размещения i-й микросхемы на j-м месте C_{ij}. На основании этих данных можно построить квадратную матрицу $\|C_{ij}\|_{n\times n}$. Требуется так распределить n микросхем на n мест на плате, чтобы сумма эффективностей их размещения была максимальной. Определим некоторую матрицу $X = \|x_{ij}\|_{n\times n}$, в которой

$$x_{ij} = \begin{cases} 1, & \textit{если } j\textit{-е место занимает } i\textit{-я микросхема}; \\ 0 & - \textit{ в противном случае.} \end{cases}$$

Математическая модель задачи оптимизации такова: $max \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$

при ограничениях

$\sum_{i=1}^{n} x_{ij} = 1, \quad j = \overline{1,n}$ (на каждом месте может размещаться только одна микросхема);

$\sum_{j=1}^{n} x_{ij} = 1, \quad i = \overline{1,n}$ (за каждой микросхемой может быть закреплено только одно рабочее место).

Геометрическая интерпретация линейных задач. Графический метод решения задач линейного программирования

Некоторые задачи линейного программирования можно решить графически, используя геометрические модели условий задачи.

Пусть задана система ограничений в каноническом виде:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \mathbf{KK} + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \mathbf{KK} + a_{2n}x_n = b_2, \\ \mathbf{KKKKKKKKKKKKKK}, \\ a_{m1}x_1 + a_{m2}x_2 + \mathbf{KK} + a_{mn}x_n = b_m, \end{cases} \qquad (4.1)$$

где m – число ограничений; n – общее число переменных (параметров).

Возможны три варианта соотношений между числом переменных n и числом ограничений m: $n < m$; $n = m$; $n > m$.

1. Если $n < m$, то в этом случае задача несовместна (не имеет решений). Например, $n = 1$; $m = 2$; $F = 3x_1$ при ограничениях $2x_1 = 4$, $x_1 = 5$.

2. Если $n = m$, то задача имеет одно единственное решение. Например, $n = 2$; $m = 2$; $F = 5x_1 + 3x_2$ при ограничениях $x_1 + x_2 = 3$; $3x_1 - x_2 = 1$; откуда $x_1 = 1$; $x_2 = 2$.

3. Если $n > m$, то система ограничений имеет бесчисленное множество решений. Задача оптимизации позволяет из бесчисленного множества решений выбрать такое, которое минимизирует или максимизирует целевую функцию.

В случае если часть ограничений представлена в виде неравенств, то, введя в неравенства дополнительные переменные, можно перейти к уравнениям. При ограничениях в виде неравенств со знаком ≤ можно к левой части неравенства прибавить переменную $x_i \geq 0$, неравенство со знаком ≥ можно превратить в равенство, вычтя из левой части переменную $x_i \geq 0$. Например, от системы ограничений

$$\begin{cases} 5x_1 + 6x_2 - 3x_3 = 18, \\ 4x_1 + x_2 + 2x_3 \leq 2, \\ 15x_1 - x_2 - x_3 \geq 7 \end{cases} \quad \text{перейдем к системе} \quad \begin{cases} 5x_1 + 6x_2 - 3x_3 = 18, \\ 4x_1 + x_2 + 2x_3 + x_4 = 2, \\ 15x_1 - x_2 - x_3 - x_5 = 7. \end{cases}$$

Очевидно, что в этом случае общее число переменных увеличится на l, где l – число дополнительных переменных или число ограничений в виде неравенств.

В общем случае задача оптимизации решается в K-мерном пространстве. При этом мерность пространства зависит как от числа переменных n, так и от вида ограничений. В случае если все m ограничений имеют вид неравенств, то $K = n$. Если все m ограничений имеют вид уравнений, то $K = n - m$. В случае если m ограничений имеют вид уравнений, а l – вид неравенств, то $K = n + l - m$. Если $K \le 2$, то задачу оптимизации можно графически изобразить на плоскости.

Пусть мерность пространства $K = 2$. Если в системе ограничений выбрать любые две переменные в качестве свободных или координатных (пусть x_1 и x_2), то через них можно выразить остальные переменные, называемые базисными, а также целевую функцию. В качестве свободных переменных лучше выбирать переменные, которые в системе ограничений встречаются большее число раз.

Так как в ЗЛП есть дополнительное условие $x_i \ge 0$, то графическое решение всегда находится в первом квадранте. Далее в системе координат свободных переменных строят область допустимых решений, используя выражения для базисных переменных. Например, ограничение $x_3 = a_{31}x_1 + a_{32}x_2 + b_3 \ge 0$ является полуплоскостью. Если $x_3 = 0$, то получается уравнение прямой в координатах x_1 , x_2. По одну сторону этой прямой располагается полуплоскость, где $x_3 > 0$, по вторую – полуплоскость, где $x_3 < 0$. Чтобы определить, с какой стороны от прямой лежит область допустимых решений (ОДР), необходимо взять любую точку, не лежащую на прямой, подставить координаты этой точки в выражение для соответствующей базисной переменной и проверить выполнение неравенства (например $x_3 > 0$). Если неравенство выполняется, значит, точка принадлежит ОДР, если нет, то полуплоскость штрихуется (точка не принадлежит ОДР). Аналогичным образом строят ОДР с учетом всех остальных ограничений и получают многоугольник области допустимых решений. Поскольку целевая функция также выражена через свободные переменные, то можно в этих же координатах построить линии уровня целевой функции, задавая целевой функции некоторые значения. По двум линиям уровня определяют направление изменения целевой функции к экстремальному значению и получают решение поставленной задачи. ОДР также можно построить для ограничений неравенств, не приводя задачу к каноническому виду [2].

Пример 1. Найти $\max F = 3x_1 + 5x_2 + 4x_3$ при ограничениях

$$\begin{cases} x_1 - x_4 = 5, \\ x_2 + 2x_4 - 3x_5 = 3, \\ x_3 + 2x_4 + 5x_5 = 5; \end{cases} \quad x_i \ge 0, \ (i = \overline{1,5}).$$

В качестве свободных переменных принимаем x_4, x_5. Выразим целевую функцию и базисные переменные через свободные переменные:

$$F = 3(5 + x_4) + 5(3 + 3x_5 - 2x_4) + 4(5 - 2x_4 - 5x_5) = 50 - 15x_4 - 5x_5;$$

$$\begin{cases} x_1 = 5 + x_4, \\ x_2 = 3 - 2x_4 + 3x_5, \\ x_3 = 5 - 2x_4 - 5x_5. \end{cases}$$

Область допустимых решений и линии отклика целевой функции приведены на рисунке 4.1.

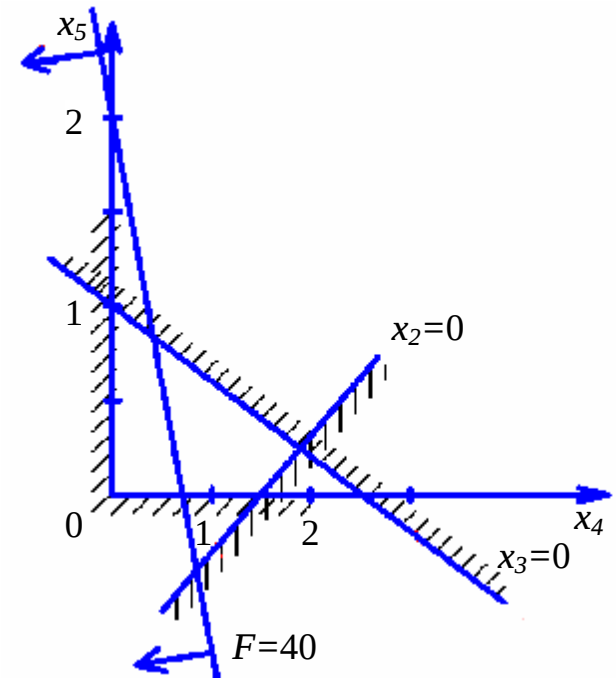

Рисунок 4.1 – Область допустимых решений и линии отклика целевой функции

Направление возрастания значения целевой функции указано стрелками.

Решение задачи x_1=5, x_2=3, x_3=5, x_4=0, x_5=0, F_{max}=50.

Как следует из системы неравенств, для каждой стороны многоугольника ОДР одна переменная равна 0. В каждой вершине ОДР две переменные равны 0. Оптимальным решением ЗЛП являются координаты одной из вершин многоугольника ОДР.

Минимум целевой функции $F_{\min} = \frac{165}{8}$ достигается при следующих значениях переменных: $x_1 = \frac{55}{8}$; $x_2 = 0$; $x_3 = 0$; $x_4 = \frac{15}{8}$; $x_5 = \frac{1}{4}$.

Симплекс-метод

Основным алгоритмом решения ЗЛП является симплекс-метод. Его можно применять в случае, если задача оптимизации записана в специальном каноническом виде. При этом все ограничения имеют вид равенств и каждая базисная переменная встречается с коэффициентом 1 только в одном уравнении, а в других коэффициент при базисной переменной равен 0. Аналитическое выражение для целевой функции не должно содержать базисных переменных, а выражаться только через свободные переменные. Если при этом $b_i \geq 0$, то говорят о допустимом каноническом виде.

Суть симплекс-метода заключается в направленном переборе вершин ОДР с целью определения координат такой вершины, в которой целевая функция достигает экстремума. Введем несколько определений.

Опорным решением системы ограничений называется такое решение, когда для системы ограничений в допустимом каноническом виде все свободные переменные равны нулю.

Вырожденное опорное решение – опорное решение, в котором одна или несколько базисных переменных равны нулю.

Допустимое опорное решение – опорное решение, в котором все базисные переменные ≥ 0.

Оптимальное решение ЗЛП будет соответствовать одному из допустимых опорных решений. В симплекс-методе, начиная с допустимого опорного решения, расположенного в вершине многогранника ОДР, переходят к соседней вершине, сохраняя допустимость решения.

Для удобства формулировки правил метода запишем допустимую каноническую форму ЗЛП в ином виде, называемом стандартной формой. Например:

$$F = c_0 - (-c_1 x_1 - c_2 x_2 - c_3 x_3);$$

$$\begin{cases} x_4 = b_1 - (a_{11} x_1 + a_{12} x_2 + a_{13} x_3), \\ x_5 = b_2 - (a_{21} x_1 + a_{22} x_2 + a_{23} x_3), \\ x_6 = b_3 - (a_{31} x_1 + a_{32} x_2 + a_{33} x_3), \\ x_7 = b_4 - (a_{41} x_1 + a_{42} x_2 + a_{43} x_3). \end{cases}$$

Для упрощения реализации симплекс-метода на ЭВМ его представляют в табличной форме. В этом случае каждое опорное решение соответствует новой симплекс-таблице. Каждая строка таблицы соответствует либо строке целевой функции, либо строке одного из ограничений. Пример таблицы для случая семи переменных для стандартной формы записи приведен ниже.

	Свободн. член	x_1	x_2	x_3
F	c_0	$-c_1$	$-c_2$	$-c_3$
x_4	b_1	a_{11}	a_{12}	a_{13}
x_5	b_2	a_{21}	a_{22}	a_{23}
x_6	b_3	a_{31}	a_{32}	a_{33}
x_7	b_4	a_{41}	a_{42}	a_{43}

В симплекс-таблице вводится понятие разрешающего элемента – элемента, лежащего на пересечении разрешающего столбца (соответствующего новой базисной переменной) и разрешающей строки (соответствующей новой свободной переменной) при процедуре замены базиса. Замена базиса необходима для перехода к следующему опорному решению.

В симплекс-методе требуется иметь исходное допустимое опорное решение. Иногда такое решение получить легко. Например, если существует ограничение вида ≤ с положительной правой частью, то дополнительные переменные, используемые при записи ограничений в виде равенств, образуют допустимое опорное решение, в котором все переменные проектирования равны нулю, а дополнительные переменные являются базисными. Однако во многих задачах существуют ограничения вида ≥ 0 и = 0, для которых начальное опорное решение может быть недопустимым. Поэтому применению симплекс-метода в общем случае должен предшествовать этап поиска допустимого решения. Рассмотрим один из вариантов алгоритма симплекс-метода с учетом этапа поиска допустимого решения.

Алгоритм симплекс-метода

1-й этап: нахождение допустимого опорного решения.

Шаг 1. Ограничения неравенства приводятся к виду ограничений равенств. Задача записывается в стандартной форме.

Шаг 2. Для каждого ограничения, имеющего отрицательный свободный член, просматриваются знаки коэффициентов при свободных переменных. Если отсутствует хотя бы один отрицательный коэффициент, то ограничения несовместны. Если во всех ограничениях свободные члены неотрицательны, то опорное решение является допустимым и необходимо переходить ко второму этапу (этапу нахождения оптимального решения).

Шаг 3. Для ограничения, имеющего отрицательный свободный член, выбирается свободная переменная с отрицательным коэффициентом. Эта переменная будет новой базисной переменной. Если отрицательных коэффициентов более одного, то в качестве новой базисной переменной можно выбрать любую, имеющую отрицательный коэффициент (выбор переменной в этом случае скажется на числе замен базиса, но не на конечном результате). Пусть это будет переменная x_l. Если ограничений с отрицательным свободным членом более одного, то можно для анализа коэффициентов выбрать любое.

Шаг 4. Для выбора новой свободной переменной рассматривается отношение b_j/a_{jl} для всех ограничений, причем b_j и a_{jl} должны иметь одинаковый знак. Находится минимальное отношение. Новой свободной переменной будет та базисная переменная, для ограничения которой отношение b_j/a_{jl} оказалось минимальным. В этом случае разрешающий элемент таблицы будет находиться на пересечении столбца, соответствующего переменной x_l и строки, соответствующей ограничению, для которого получено минимальное отношение.

Шаг 5. Делается замена базиса. Для пересчета значений элементов симплекс-таблицы после смены базиса введем в таблицу следующие обозначения:

	Своб. член	x_1	x_2	...	x_i
F	α_{11}	α_{12}			α_{1l}
x_{i+1}	α_{21}	α_{22}			α_{2l}
...	...	...	...	...	...
x_n	α_{k1}	α_{k2}			α_{kl}

С учетом введенных обозначений после смены базиса все элементы таблицы могут быть пересчитаны по следующим выражениям с учетом того, что α_{rp} – разрешающий элемент (r, q – строка, p, s – столбец).

$a_{qs}^{\nu+1} = \dfrac{1}{a_{rp}}$ при $q=r$; $s=p$, ν – номер итерации(смены базиса);

$a_{qs}^{\nu+1} = \dfrac{a_{qs}^{\nu}}{a_{rp}}$ при $q=r$; $s \neq p$, $s=\overline{1,l}$;

$a_{qs}^{\nu+1} = -\dfrac{a_{qs}^{\nu}}{a_{rp}}$ при $q \neq$ r; $s=p$; $q=\overline{1,k}$;

$a_{qs}^{\nu+1} = a_{qs}^{\nu} - \dfrac{a_{rs}^{\nu} a_{qp}^{\nu}}{a_{rp}}$ для остальных элементов.

Шаг 6. Возвращаемся на шаг 2.

2-й этап: нахождение оптимального решения.

Шаг 1. Проверяется признак оптимальности решения.

Признаки оптимального решения:

а) целевая функция будет иметь максимальное значение в том случае, когда в строке (выражении) целевой функции в стандартной форме записи все коэффициенты при свободных переменных положительны (свободный член не рассматривается);

б) целевая функция будет иметь минимальное значение в том случае, если в строке целевой функции в стандартной форме все коэффициенты при свободных переменных отрицательны;

в) если в выражении целевой функции в стандартной форме все коэффициенты при переменных одного знака и при этом есть хотя бы один нулевой коэффициент, то полученное решение является альтернативным, т.е. будет другой набор переменных, доставляющих целевой функции такое же значение.

Если признак оптимальности выполняется, то решение найдено. Значения свободных переменных равны нулю, значения базисных переменных равны свободным членам соответствующих ограничений, значение целевой функции равно свободному члену целевой функции. Если признак оптимальности не выполняется, то переходим к шагу 2.

Шаг 2. В соответствии с правилом выбора новой базисной переменной одну из свободных переменных переводим в базисные.

Правило выбора свободной переменной, переводимой в базис

При поиске максимума (минимума) функции, записанной в стандартной форме, в базис должна переводиться переменная, имеющая отрицательный (положительный) коэффициент в выражении целевой функции в стандартной форме. Если имеется более одного отрицательного (положительного) коэффициента c_j, то выбирают переменную, имеющую больший по абсолютной величине отрицательный (положительный) коэффициент c_j.

Шаг 3. В соответствии с правилом выбора переменной, переводимой из базисной в свободные, определяем новую свободную переменную.

Сформулируем правило выбора переменной, переводимой из базисных в свободные.

Для выбора новой свободной переменной x_i необходимо рассмотреть отношение b_j / a_{ij} для всех ограничений ($j = \overline{1,m}$), ($i = \overline{1,n}$), из этих отношений выбрать минимальное, а в качестве новой свободной переменной выбирать базисную переменную, для ограничения которой получено минимальное отношение b_j / a_{ij}. Отношение b_j / a_{ij} необходимо рассматривать только для положительных a_{ij}. Если минимум достигается более чем для одного индекса j, то из базиса исключается любая из переменных, соответствующих j-м ограничениям.

Если ни один из a_{ij} не является положительным, то получение нового допустимого решения невозможно, т.е. при поиске минимума целевая функция не ограничена снизу, а при поиске максимума – не ограничена сверху (признак неограниченности целевой функции).

Шаг 4. Осуществляется замена базиса аналогично шагу 5 первого этапа.

Шаг 5. Переходим к шагу 1.

Пример 2. Найти $\min F = 2x_1 - 2x_2 - x_3$ при ограничениях

$$\begin{cases} x_1 + x_2 - x_3 \ge -1, \\ 2x_1 + x_3 \ge 5, \\ x_1 + x_2 \le 2, \\ 2x_2 + x_3 \le 2, \\ x_j \ge 0; (j = \overline{1,3}). \end{cases}$$

Введем дополнительные переменные и перейдем в ограничениях к знакам «=».

$$\begin{cases} -x_1 - x_2 + x_3 + x_4 = 1; \\ -2x_1 - x_3 + x_5 = -5; \\ x_1 + x_2 + x_6 = 2; \\ 2x_2 + x_3 + x_7 = 2; \\ x_j \ge 0, \ (j = \overline{1,7}). \end{cases}$$

Запишем задачу в стандартной форме и представим ее в виде симплекс-таблицы:

$$\min F = 0 - (-2x_1 + 2x_2 + x_3);$$

$$\begin{cases} x_4 = 1 - (-x_1 - x_2 + x_3), \\ x_5 = -5 - (-2x_1 - x_3), \\ x_6 = 2 - (x_1 + x_2), \\ x_7 = 2 - (2x_2 + x_3). \end{cases}$$

	Св. чл.	x_1	x_2	x_3
F	0	-2	2	1
x_4	1	-1	-1	1
x_5	-5	-2	0	-1
x_6	2	**1**	1	0
x_7	2	0	2	1

Признак несовместности ограничений не выполняется. Допустимое решение не найдено, так как свободный член для ограничения x_5 равен –5. Для получения допустимого решения произведем смену базиса. В качестве разрешающего столбца выберем x_1; разрешающей строки – x_6 (так как 2/1 < <–5/–2). Разрешающий элемент подчеркнут. Строим новую симплекс-таблицу.

	Св. чл.	x_6	x_2	x_3
F	4	2	4	1
x_4	3	1	0	1
x_5	-1	2	2	**-1**
x_1	2	1	1	0
x_7	2	0	2	1

Допустимое решение не найдено. Произведем следующую замену базиса.

	Св. чл.	x_6	x_2	x_5
F	3	4	6	1
x_4	2	3	2	1
x_3	1	-2	-2	-1
x_1	2	1	1	0
x_7	1	2	**4**	1

Допустимое решение найдено. Осуществляем поиск оптимального решения (см. таблицы ниже).

	Св. чл.	x_6	x_7	x_5
F	3/2	1	-3/2	-1/2
x_4	3/2	2	-1/2	1/2
x_3	3/2	-1	1/2	-1/2
x_1	7/4	1/2	-1/4	-1/4
x_2	1/4	**1/2**	1/4	1/4

	Св. чл.	x_2	x_7	x_5
F	1	-2	-2	-1
x_4	1/2	-4	-3/2	-1/2
x_3	2	2	1	0
x_1	3/2	-1	-1/2	-1/2
x_6	1/2	2	1/2	1/2

Признак оптимальности удовлетворяется. Решение задачи:

$\min F$=1; x_1=3/2; x_2=0; x_3=2; x_4=1/2; x_5=0; x_6=1/2; x_7=0.

4.2 Порядок выполнения работы

1 Получить задание у преподавателя.
2 Решить задачу графическим и симплекс-методом.

4.3 Содержание отчета

1 Цель работы.
2 Исходное задание.
3 Решение задачи графическим и симплекс-методом.
4 Выводы по полученным решениям.

4.4 Контрольные вопросы

1 Какие ЗЛП можно решить графическим методом?
2 Чем характеризуется каноническая форма записи ЗЛП?
3 Как построить область допустимых решений графическим методом?
4 Сформулируйте признак оптимальности решения в симплекс-методе.
5 Что является признаком допустимости решения в симплекс-методе?
6 Как выбрать новые базисную и свободную переменные в симплекс-методе?
7 Что является признаком наличия альтернативных решений и какова геометрическая интерпретация такой ситуации?
8 Как в симплекс-методе определить ситуацию несовместности ограничений?
9 Как ограничения в виде неравенств привести к ограничениям в виде равенств?
10 Что является признаком неограниченности целевой функции в симплекс-методе?

Литература

1 Деньдобренько, Б. Н. Автоматизация конструирования РЭА / Б. Н. Деньдобренько, А. С. Малика. – М. : Высш. шк., 1980.
2 Зайченко, Ю. П. Исследование операций / Ю. П. Зайченко, С. А. Шумилова. – Киев. : Вища шк., 1990.

ЛАБОРАТОРНАЯ РАБОТА №5. ИЗУЧЕНИЕ ГРАДИЕНТНЫХ МЕТОДОВ РЕШЕНИЯ ЗАДАЧИ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Цель: изучить градиентные методы решения задачи нелинейного программирования.

5.1 Теоретические сведения

Постановка задачи нелинейного программирования

В общем виде задача нелинейного программирования математически может быть записана следующим образом: необходимо найти минимум целевой функции *F(X)* в области допустимых решений *G*:

$$\min_{X \in G} F(X),$$

где *G* определяется ограничениями $g_i(X)$ {£ ,=, ³} 0, $i = 1,2\ldots m;$ а вариант решения задачи характеризуется многомерным вектором $X = (x_1,x_2,x_3\ldots..x_n)$, компонентами x_j которого могут быть первичные параметры объекта проектирования. При этом целевая функция или хотя бы одно из ограничений должны быть нелинейны.

В отличие от линейного программирования в нелинейном программировании отсутствуют универсальные методы решения типа симплекс-метода. Это связано с тем, что допустимое множество решений, определяемое ограничениями, в общем случае не является выпуклым, а кроме того, даже в случае выпуклости множество его крайних точек может не быть конечным. Кроме того, в нелинейных функциях может иметь место седловая точка и нелинейные функции могут иметь локальные экстремумы. Поэтому методы нелинейного программирования разрабатываются под специальные классы задач [1].

В САПР применяют численные (регулярные) методы решения задач оптимизации, поскольку они приспособлены для решения на ЭВМ. Среди численных методов поиска экстремума наибольшее распространение получили прямые поисковые методы (которые используют только значения функции для получения следующей точки) и градиентные методы, которые используют дополнительно значения первых производных [2]. В данной лабораторной работе будут рассмотрены градиентные методы, которые по сравнению с прямыми поисковыми методами обладают более быстрой сходимостью.

Градиентные методы безусловной оптимизации

В задаче безусловной оптимизации отсутствуют ограничения.

Напомним, что градиентом многомерной функции называют вектор, который аналитически выражается геометрической суммой частных производных

$$\overrightarrow{gradF(X)} = \frac{\partial F}{\partial x_1}\overrightarrow{x}_1 + \frac{\partial F}{\partial x_2}\overrightarrow{x}_2 + \ldots\ldots + \frac{\partial F}{\partial x_n}\overrightarrow{x}_n.$$

Градиент скалярной функции *F(X)* в некоторой точке направлен в сторону наискорейшего возрастания функции и ортогонален линии уровня (поверхности постоянного значения *F(X)*, проходящей через точку X_k). Вектор, противоположный градиенту – антиградиент – направлен в сторону наискорейшего убывания функции *F(X)*. В точке экстремума *grad F(X)*=0.

В градиентных методах движение точки при поиске минимума целевой функции описывается итерационной формулой

$$(x_i)_{k+1} = (x_i)_k - l_k\left(\frac{\partial F}{\partial x_i}\right)_k,$$

где l_k – параметр шага на *k*-й итерации вдоль антиградиента. Для методов восхождения (поиска максимума) нужно двигаться по градиенту.

Различные варианты градиентных методов отличаются друг от друга способом выбора параметра шага, а также учета направления движения на предыдущем шаге [1]. Рассмотрим следующие варианты градиентных методов: с постоянным шагом, с переменным параметром шага (дроблением шага), метод наискорейшего спуска и метод сопряженных градиентов.

Метод с постоянным параметром шага. В этом методе параметр шага постоянен на каждой итерации. Возникает вопрос: как практически выбрать величину параметра шага? Достаточно малый параметр шага может привести к неприемлемо большому количеству итераций, необходимых для достижения точки минимума. С другой стороны, слишком большой параметр шага может привести к проскакиванию точки минимума и к колебательному вычислительному процессу около этой точки. Указанные обстоятельства являются недостатками метода. Поскольку невозможно заранее угадать приемлемое значение параметра шага l_k, то возникает необходимость использования градиентного метода с переменным параметром шага.

По мере приближения к оптимуму вектор градиента уменьшается по величине, стремясь к нулю, поэтому при l_k = const длина шага постепенно уменьшается. Вблизи оптимума длина вектора градиента стремится к нулю. Длина вектора или норма в *n*-мерном евклидовом пространстве определяется по формуле

$$\left\|gradF(X_k)\right\| = \sqrt{\sum_{i=1}^{n}\left(\frac{\partial F}{\partial x_i}\right)_k^2}\quad,\quad \text{где } n - \text{число переменных.}$$

Варианты остановки процесса поиска оптимума:

1 По разности значений целевой функции $\left|F(X_k) - F(X_{k+1})\right| < e_1$.

2 По величине нормы $\left\|gradF(X_k)\right\| < e_2$.

3 По величине изменения шага $\left|(x_i)_k - (x_i)_{k+1}\right| < e_3$.

С практической точки зрения удобней пользоваться 3-м критерием остановки (поскольку представляют интерес значения параметров проектирования), однако для определения близости точки экстремума нужно ориентироваться на 2-й критерий. Для остановки вычислительного процесса можно использовать несколько критериев.

Рассмотрим пример. Найти минимум целевой функции $F(X) = (x_1 - 2)^2 + (x_2 - 4)^2$. Точное решение задачи $X^* = (2,0;4,0)$. Выражения для частных производных

$$\frac{\partial F}{\partial x_1} = 2(x_1 - 2) = 2x_1 - 4, \qquad \frac{\partial F}{\partial x_2} = 2(x_2 - 4) = 2x_2 - 8 .$$

Выбираем шаг $l_k = 0{,}1$. Осуществим поиск из начальной точки $X_1 = [3;1]$. Решение представим в виде таблицы.

k	X_k	$F(X_k)$	$\partial F/\partial x_k$	$-\partial F/\partial x_k$	X_{k+1}	$F(X_{k+1})$
1	[3;1]	10,0	[2;-6]	[-2;6]	[2,8;1,6]	6,4
2	[2,8;1,6]	6,4	[1,6;-4,8]	[-1,6;4,8]	[2,64;2,08]	4,096
3	[2,64;2,08]	4,096	[1,28;-3,84]	[-1,28;3,84]	[2,51;2,46]	2,62
4	[2,51;2,46]	2,62	[1,02;-3,08]	[-1,02;3,08]	[2,41;2,77]	1,68
5	[2,41;2,77]	1,68	[0,82;-2,46]	[-0,82;2,46]	[2,33;3,02]	1,07
	…		…		…	

Градиентный метод с дроблением параметра шага. В этом случае в процессе оптимизации параметр шага λ_k уменьшается, если после очередного шага целевая функция возрастает (при поиске минимума). При этом часто длина шага дробится (делится) пополам, и шаг повторяется из предыдущей точки. Так обеспечивается более точный подход к точке экстремума.

Метод наискорейшего спуска. Методы с переменным шагом являются более экономичными с точки зрения количества итераций. В случае если оптимальная длина шага λ_k вдоль направления антиградиента является решением одномерной задачи минимизации, то такой метод называется методом наискорейшего спуска. В этом методе на каждой итерации решается задача одномерной минимизации:

$$F(X_{k+1}) = F(X_k - l_k S_k) = \min_{l_k > 0} F(l_k), \qquad S_k = \tilde{N}F(X);$$

$$\nabla F(X) = \frac{\partial F(X)}{\partial x_1}, \frac{\partial F(X)}{\partial x_2}, \ldots\ldots \frac{\partial F(X)}{\partial x_{1n}} .$$

В данном методе движение в направлении антиградиента продолжается до достижения минимума целевой функции (пока значение целевой функции убывает). На примере рассмотрим, как аналитически может быть записана на каждом шаге целевая функция в зависимости от неизвестного параметра шага λ.

Пример. $\min F(x_1,x_2) = 2x_1^2 + 4x_2^3 - 3$. Тогда $\tilde{N}F(X) = [\,4x_1;\ 12x_2^2]$. Пусть точка $X_k = [2,0;1,0]$, следовательно, $-\tilde{N}F(X) = [\,-8;\ -12]$, $F(X_k - lS_k) = 2(2 - 8l)^2 + 4(1 - 12l)^3 - 3$. Необходимо найти λ, доставляющее минимум данной функции.

Алгоритм метода наискорейшего спуска (для поиска минимума)

Начальный шаг. Пусть ε – константа остановки. Выбрать начальную точку X_1, положить $k = 1$ и перейти к основному шагу.

Основной шаг. Если $||gradF(X)|| < \varepsilon$, то закончить поиск, в противном случае определить $\tilde{N}F(X_k)$ и найти l_k – оптимальное решение задачи минимизации $F(X_k - l_k S_k)$ при $l_k \geq 0$. Положить $X_{k+1} = X_k - l_k S_k$, присвоить $k = k + 1$ и повторить основной шаг.

Для поиска минимума функции одной переменной в методе наискорейшего спуска можно использовать методы унимодальной оптимизации. Из большой группы методов рассмотрим метод дихотомии (бисекции) и золотого сечения. Суть методов унимодальной оптимизации заключается в сужении интервала неопределенности размещения экстремума.

Метод дихотомии (бисекции) Начальный шаг. Выбирают константу различимости ε и конечную длину интервала неопределенности l. Величина ε должна быть по возможности меньшей, однако позволяющей различать значения функции $F(l)$ и $F(m)$. Пусть $[a_1,b_1]$ – начальный интервал неопределенности. Положить $k = 1$ и перейти к основному этапу.

Основной этап состоит из конечного числа однотипных итераций.

k-я итерация.

Шаг 1. Если $b_k - a_k \leq l$, то вычисления заканчиваются. Решение $x^* = (a_k + b_k)/2$. В противном случае

$$l_k = \frac{a_k + b_k}{2} - e \quad , \qquad m_k = \frac{a_k + b_k}{2} + e \quad .$$

Шаг 2. Если $F(l_k) < F(m_k)$, положить $a_{k+1} = a_k$; $b_{k+1} = m_k$. В противном случае $a_{k+1} = l_k$ и $b_{k+1} = b_k$. Присвоить $k = k + 1$ и перейти к шагу 1.

Метод золотого сечения. Более эффективный метод, чем метод дихотомии. Позволяет получить заданную величину интервала неопределенности за меньшее число итераций и требует меньшего числа вычислений целевой функции. В этом методе новая точка деления интервала неопределенности вычисляется один раз. Новая точка ставится на расстоянии $\alpha = 0{,}618034$ от конца интервала.

Алгоритм метода золотого сечения

Начальный шаг. Выбрать допустимую конечную длину интервала неопределенности $l > 0$. Пусть $[a_1,b_1]$ – начальный интервал неопределенности. Положить $l_1 = a_1 + (1 - a)(b_1 - a_1)$ и $m_1 = a_1 + a(b_1 - a_1)$, где $a = 0{,}618$. Вычислить $F(l_1)$ и $F(m_1)$, положить $k = 1$ и перейти к основному этапу.

Шаг 1. Если $b_k - a_k \leq l$, то вычисления заканчиваются $x^* = (a_k + b_k)/2$. В противном случае если $F(l_k) > F(m_k)$, то перейти к шагу 2; если $F(l_k) \leq F(m_k)$, перейти к шагу 3.

Шаг 2. Положить $a_{k+1} = l_k$, $b_{k+1} = b_k$, $l_{k+1} = m_k$, $m_{k+1} = a_{k+1} + a(b_{k+1} - a_{k+1})$. Вычислить $F(m_{k+1})$, перейти к шагу 4.

Шаг 3. Положить $a_{k+1} = a_k$, $b_{k+1} = m_k$, $m_{k+1} = l_k$, $l_{k+1} = a_{k+1} + (1 - a)(b_{k+1} - a_{k+1})$. Вычислить $F(l_{k+1})$.

Шаг 4. Присвоить $k = k + 1$, перейти к шагу 1.

На первой итерации необходимы два вычисления функции, на всех последующих только одно.

Метод сопряженных градиентов (Флетчера–Ривса). В этом методе выбор направления движения на $k+1$ шаге учитывает изменение направления на k шаге. Вектор направления спуска является линейной комбинацией направления антиградиента и предыдущего направления поиска. В этом случае при минимизации овражных функций (с узкими длинными впадинами) поиск идет не перпендикулярно оврагу, а вдоль него, что позволяет быстрее прийти к минимуму. Координаты точки при поиске экстремума методом сопряженных градиентов рассчитываются по выражению $X_{k+1}=X_k - l V_{k+1}$, где V_{k+1} – вектор, рассчитываемый по следующему выражению:

$$V_{k+1} = gradF(X_k) - \frac{\left\| gradF(X_k) \right\|^2}{\left\| gradF(X_{k-1}) \right\|^2} V_k \ .$$

На первой итерации обычно полагается $V = 0$ и выполняется поиск по антиградиенту, как в методе наискорейшего спуска. Затем направление движения отклоняется от направления антиградиента тем больше, чем значительнее менялась длина вектора градиента на последней итерации. После n шагов для коррекции работы алгоритма делают обычный шаг по антиградиенту.

Алгоритм метода сопряженных градиентов

Шаг 1. Ввести начальную точку X_0, точность e, размерность n.

Шаг 2. Положить $k = 1$.

Шаг 3. Положить вектор $V_k = 0$.

Шаг 4. Вычислить $grad\ F(X_k)$.

Шаг 5. Вычислить вектор V_{k+1}.

Шаг 6. Выполнить одномерный поиск по вектору V_{k+1}.

Шаг 7. Если $k < n$, положить $k = k + 1$ и перейти к шагу 4, иначе – к шагу 8.

Шаг 8. Если длина вектора V меньше ε, окончить поиск, иначе – перейти к шагу 2.

Метод сопряженных направлений является одним из наиболее эффективных в решении задач минимизации. Метод в совокупности с одномерным поиском часто практически используется в САПР. Однако следует отметить, что он чувствителен к ошибкам, возникающим в процессе счета.

Недостатки градиентных методов

1 В задачах с большим числом переменных трудно или невозможно получить производные в виде аналитических функций.

2 При вычислении производных по разностным схемам возникающая при этом ошибка, особенно в окрестностях экстремума, ограничивает возможности такой аппроксимации.

Условная оптимизация градиентным методом

В случае если решение задачи достигается на границе области допустимых решений (ОДР), то градиентные методы требуют модификации с целью поиска вдоль границы ОДР. Рассмотрим задачу поиска минимума для ограничений вида $g_i(X) \leq 0$. Пока представляющая точка находится в ОДР, поиск производится по обычной схеме. Однако как только будет нарушено хотя бы одно ограничение, то процедуру поиска нужно менять. В этом случае одним из возможных способов движения является перемещение в направлении, противоположном направлению суммы градиентов тех функций $g_i(X)$, для которых в данной точке нарушаются ограничения. В этом случае траектория движения будет представлять собой зигзагообразную траекторию вдоль ограничения (рисунок 5.1).

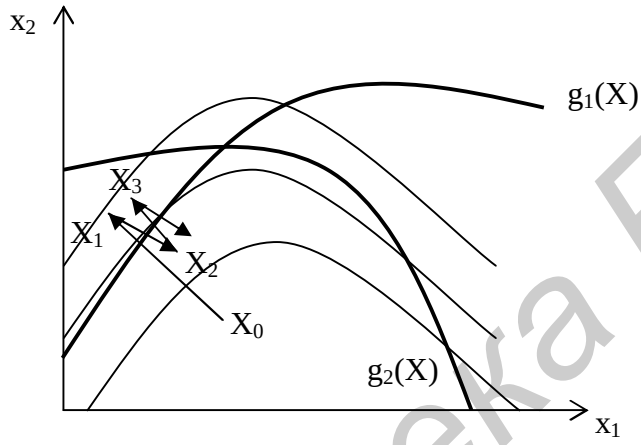

Рисунок 5.1 – Траектория движения при поиске экстремума

Для реализации такого способа алгоритмы, основанные на методе градиента с дроблением шага или наискорейшего спуска, должны быть изменены следующим образом:

1 После каждого рабочего шага в методе с дроблением шага (или малого рабочего шага в методе наискорейшего спуска) должно проверяться выполнение ограничений $g_i(X_{k+1}) \leq 0$ в новой точке X_{k+1}.

2 Если ограничения выполняются, то поиск происходит по обычной схеме.

3 Если хотя бы одно из ограничений нарушено, то дальнейший поиск ведется следующим образом:

а) для всех $g_i(X)$, которые стали положительными, составляется новый комплексный критерий $H = \sum_{i=1}^{m} g_i(X)$, где m – число нарушенных ограничений;

б) вычисляется $grad\ H$ так же, как и для $F(X)$;

в) делается шаг в направлении $-grad\ H$ до тех пор, пока все $g_i(X)$ не станут отрицательными (представляющая точка не войдет в ОДР).

4 Далее поиск происходит по обычной схеме до нового нарушения хотя бы одного из ограничений.

5.2 Порядок выполнения работы

1 Получить задание у преподавателя.

2 Написать программу поиска экстремума целевой функции заданным методом.

3 Для выданной задачи нелинейного программирования продемонстрировать работу программы при различных стартовых точках.

5.3 Содержание отчета

1 Цель работы.

2 Исходное задание.

3 Решение задачи заданным методом.

4 Выводы по полученным результатам.

5.4 Контрольные вопросы

1 Куда направлен градиент скалярной функции?

2 Каково условие дробления параметра шага в градиентном методе с дроблением шага?

3 Каковы критерии остановки процесса поиска оптимума в градиентном методе?

4 В чем отличие метода наискорейшего спуска от градиентного метода с дроблением шага?

5 Какие методы могут использоваться для поиска значения параметра шага в методе наискорейшего спуска?

6 Как изменится основное расчетное выражение градиентных методов для определения следующей точки при решении задачи поиска максимума?

7 Как учесть ограничения при оптимизации с помощью градиентных методов?

8 Почему в задаче нелинейного программирования поисковый процесс необходимо начинать из нескольких разных точек?

Литература

1 Автоматизированное проектирование радиоэлектронных средств : учеб. пособие для вузов / О. В. Алексеев [и др.]; под ред. О. В. Алексеева. – М. : Высш. шк., 2000.

2 Норенков, И. П. Основы автоматизированного проектирования / И. П. Норенков. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2000.

ЛАБОРАТОРНАЯ РАБОТА №6. ИЗУЧЕНИЕ АЛГОРИТМОВ РАЗМЕЩЕНИЯ ЭЛЕМЕНТОВ

Цель: изучить алгоритмы размещения конструктивных элементов на печатной плате.

6.1 Теоретические сведения

Постановка задачи размещения

Задача размещения состоит в отыскании для каждого размещаемого конструктивного элемента таких позиций на плате, при которых оптимизируется выбранный показатель качества.

Основная сложность в постановке задач размещения заключается в выборе целевой функции решения задачи оптимизации. Связано это с тем, что одной из главных целей размещения является создание наилучших условий для дальнейшей трассировки печатных соединений, что возможно только при совместном решении задач размещения элементов и трассировки соединений, что практически нереально вследствие огромных затрат машинного времени на решение такой задачи. Поэтому все применяемые в настоящее время алгоритмы размещения используют частные критерии оптимизации, которые лишь качественно способствуют решению основной задачи: получению оптимальной трассировки соединений. К таким критериям относятся: минимум суммарной взвешенной длины соединений; минимум числа соединений, длина которых больше заданной; минимум числа пересечений проводников; максимальное число соединений между элементами, находящимися в соседних позициях либо в позициях, указанных разработчиком [1]. Наибольшее практическое распространение в алгоритмах размещения получил первый критерий, что объясняется уменьшением длин соединений, улучшением электрических характеристик устройства, упрощением трассировки печатных проводников.

Исходной информацией при решении задачи размещения конструктивных элементов (микросхем, транзисторов, пассивных компонентов) являются:

- данные о форме и размерах коммутационного пространства печатной платы, которые зависят от требований к креплению печатного узла в аппаратуре и конструктивных особенностей этой аппаратуры;
- количество и геометрические размеры конструктивных элементов;
- принципиальная схема соединений конструктивных элементов между собой;
- ограничения на взаимное расположение отдельных элементов, учитывающих особенности разрабатываемой конструкции (по электромагнитной совместимости, тепловыделению, особенностям крепления).

При определении длины соединений нужно рассчитывать расстояние между элементами на печатной плате. Для измерения длин соединений необходимо связать с коммутационным пространством некоторую систему координат *XOY* (рисунок 6.1).

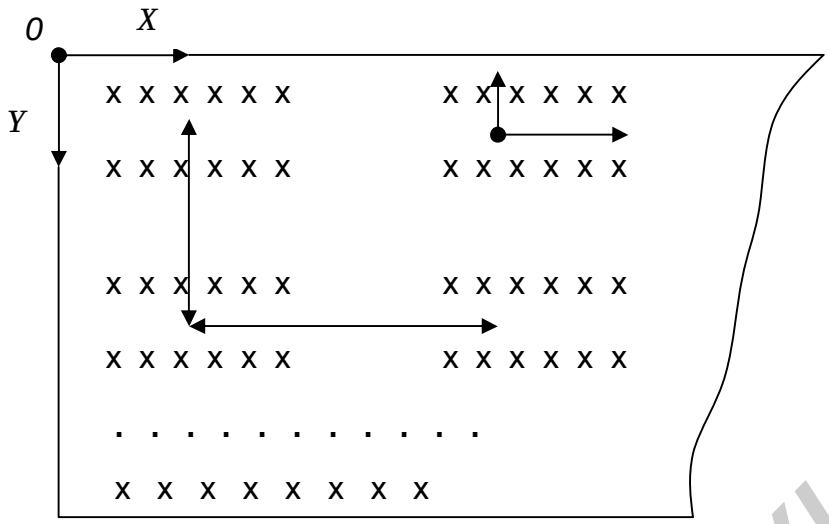

Рисунок 6.1 – Параметры коммутационного поля

Расстояние между соединяемыми элементами (выводами) можно определить одним из следующих способов:

$$d_{ij}^{(1)} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad (6.1)$$

$$d_{ij}^{(2)} = | x_i - x_j | + | y_i - y_j |, \quad (6.2)$$

$$d_{ij}^{(3)} = | x_i - x_j |^S + | y_i - y_j |^S . \quad (6.3)$$

Первый способ соответствует прокладке соединений по кратчайшему расстоянию между соединяемыми точками (эвклидова метрика). Второй способ предполагает проведение соединений по каналам или магистралям, параллельным координатным осям (ортогональная метрика). Третий применяется, когда помимо минимизации суммарной длины соединений требуется уменьшение их максимальной длины. Действительно, при использовании (6.3) длинные соединения будут давать наибольший вклад в суммарную длину и критерий минимума суммарной взвешенной длины соединений косвенным образом будет учитывать указанное требование. Параметр S в (6.3) может быть принят равным 2, 3, ... [1].

Следует иметь в виду, что расчет реальных длин монтажных соединений на этапе размещения элементов практически не осуществим, поскольку при этом потребовалось бы решение соответствующей задачи трассировки всех соединений.

В общем виде задача размещения конструктивных элементов на коммутационной плате (коммутационном пространстве) по критерию минимума суммарной взвешенной длины соединений формулируется следующим образом. Задано множество конструктивных элементов $R=\{r_1,r_2,\ldots,r_n\}$ и множество связей между этими элементами $V=\{v_1,v_2,\ldots,v_p\}$, а также множество установочных мест (позиций) на коммутационной плате $T=\{t_1,t_2,\ldots,t_k\}$. Найти такое отображение множества R на множестве T, которое обеспечивает экстремум целевой функции

$$\min F = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} d_{ij} , \qquad (6.4)$$

где c_{ij} – коэффициент взвешенной связности.

Наиболее часто используется следующее выражение для коэффициента взвешенной связности:

$$c_{ij} = \sum_{k=1}^{N} f_{ij}^{(k)} / s_k , \qquad (6.5)$$

где $f_{ij}^{(k)}$ – вес k-й цепи, связывающей элементы i и j; s_k – число контактов, объединенных цепью; N – число цепей схемы между i и j элементами. Весовой коэффициент $f_{ij}^{(k)}$ определяет важность k-й цепи с точки зрения минимизации ее длины. Данный коэффициент назначается исходя из практических соображений.

Алгоритмы размещения

Всю совокупность алгоритмов размещения можно разделить на две основные группы: непрерывно-дискретные и дискретные [1,2].

Для **непрерывно-дискретных методов** задача размещения решается в два этапа. На первом этапе решается задача нелинейного программирования для непрерывных значений координат по целевой функции (6.4), где d_{ij} рассчитывается чаще всего по (6.3). Ограничениями на координаты центров компонентов могут быть ограничения габаритов платы и компонентов и ограничения на зоны, запрещенные для размещения. Указанная задача решается известными методами нелинейного программирования (чаще градиентными), после чего проводится округление полученных координат до ближайших целочисленных с учетом сетки размещения. Недостаток – приближенное решение, нахождение локального экстремума.

Для **дискретных методов** используются возможные фиксированные позиции для размещения элементов, поэтому изменение координат центров компонентов дискретно. Среди дискретных методов можно выделить следующие группы алгоритмов: алгоритмы случайного поиска, алгоритмы решения задачи назначения, эвристические алгоритмы.

В алгоритмах случайного поиска происходит генерация по равномерному закону случайного номера позиции на коммутационном пространстве для каждого компонента. Для полученного случайного размещения подсчитывается значение целевой функции. После генерации заранее заданного количества размещений определяется лучшее по заданному критерию. Достоинство – возможность проведения оптимизации сразу по нескольким показателям качества. Недостаток – быстрый рост затрат машинного времени при необходимости повышения точности результата.

В алгоритмах решения задачи назначения решается задача целочисленного линейного программирования. Один из возможных методов решения – метод ветвей и границ, позволяющий найти точное решение задачи размещения, но требующий больших затрат машинного времени.

Эвристические алгоритмы позволяют значительно сократить затраты машинного времени при значительном (более 100) количестве размещаемых компонентов. К данной группе относятся последовательные, итерационные и смешанные алгоритмы.

Последовательный алгоритм размещения

Последовательные алгоритмы основаны на допущении, что для получения оптимального размещения необходимо в соседних позициях располагать элементы, максимально связанные друг с другом. Сущность этих алгоритмов состоит в последовательном закреплении заданного набора конструктивных элементов на коммутационной плате относительно ранее установленных.

В качестве исходных данных алгоритм использует принципиальную схему, по которой составляется матрица взвешенной связности, а также коммутационное поле платы, по которому составляется матрица расстояний для посадочных мест. Матрица расстояний обычно строится в ортогональной метрике.

Рассмотрим алгоритм метода.

Шаг 1. Строится матрица расстояний коммутационного поля платы в ортогональной метрике. Элементы матрицы рассчитываются по формуле (6.2). При однотипных компонентах расстояние рассчитывается в относительных единицах (расстояние между двумя соседними посадочными местами принимается равным 1 по соответствующей координате).

Шаг 2. По матрице расстояний строится вектор очередности позиций (посадочных мест), определяющий порядок назначения позиций на размещение. Очередь позиций выстраивается в соответствии с суммарным расстоянием позиции до всех остальных. На первое место ставится позиция с минимальным суммарным расстоянием и далее в порядке возрастания. При равных суммарных расстояниях раньше располагается позиция с меньшим номером.

Шаг 3. По схеме устройства строится матрица взвешенной связности. Строится вектор очередности размещения элементов схемы на плате. Вектор очередности строится в соответствии с суммарным весом элементов в порядке их убывания.

Шаг 4. Первый элемент из вектора размещения назначается на первую позицию из вектора позиций, второй – на вторую и т.д.

Алгоритмы, использующие последовательный процесс закрепления элементов в позициях, являются в настоящее время самыми быстродействующими. Их используют обычно для получения начального размещения элементов на плате. Более сложными являются итерационные алгоритмы, рассмотрение которых выходит за рамки лабораторной работы.

Пример. Произвести размещение элементов с помощью алгоритма последовательного размещения для приведенной схемы и коммутационного поля (рисунок 6.2). В качестве критерия размещения использовать минимум суммарной взвешенной длины.

Предположим, что элементы имеют однотипные корпуса. Примем значения весов всех цепей одинаковыми и равными 100.

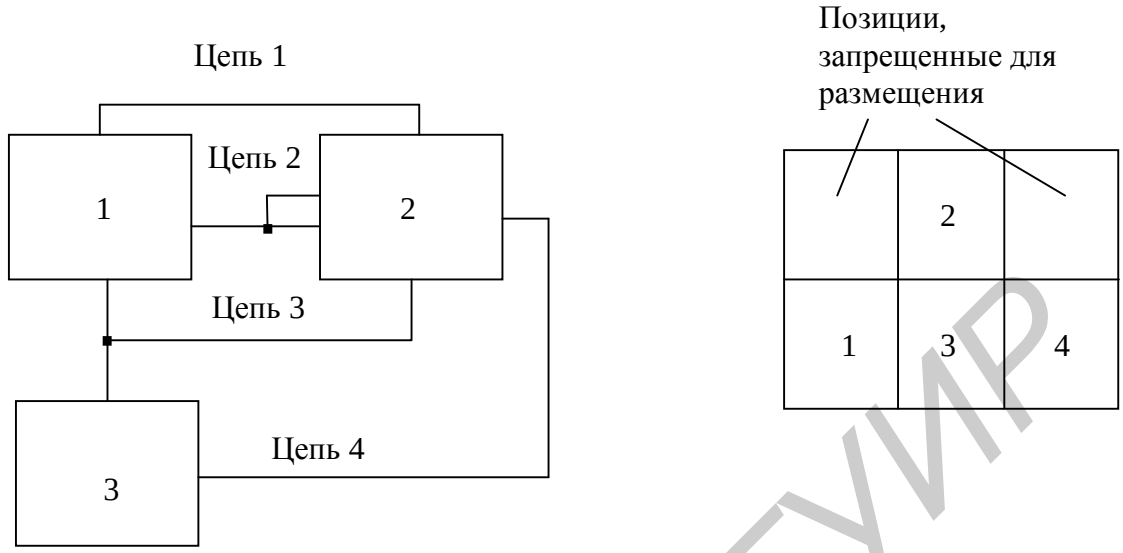

Рисунок 6.2 – Схема соединений элементов и расположение посадочных мест на печатной плате

Составим матрицу относительных расстояний в ортогональной метрике.

	1	2	3	4	Σd
1	0	2	1	2	5
2	2	0	1	2	5
3	1	1	0	1	3
4	2	2	1	0	5

Вектор очередности позиций для размещения будет следующим: 3; 1; 2; 4 (позиции 1, 2, 4 равноценны).

Строим матрицу взвешенной связности. Для этого рассчитаем элементы матрицы (коэффициенты взвешенной связности).

$$c_{12} = \left[\frac{100}{2}\right] + \left[\frac{100}{3}\right] + \left[\frac{100}{3}\right] = 116,$$

$$c_{13} = \left[\frac{100}{3}\right] = 33,$$

$$c_{23} = \left[\frac{100}{3}\right] + \left[\frac{100}{2}\right] = 83.$$

	1	2	3	Σc
1	0	116	33	149
2	116	0	83	199
3	33	83	0	116

Вектор назначения элементов на позиции: 2; 1; 3.

Решение: на первой позиции – первый элемент; на второй – третий элемент; на третей – второй элемент; четвертая позиция – пустая.

6.2 Порядок выполнения работы

1 Получить задание у преподавателя.

2 Печатная плата имеет посадочные места, приведенные на рисунке 6.3. Соединитель печатной платы должен быть размещен в позиции 1. Предполагается, что остальные элементы имеют одинаковые корпуса и могут быть размещены в любой из оставшихся позиций.

3 Решить задачу размещения с помощью алгоритма случайного поиска и последовательного алгоритма. Написать программы, реализующие указанные методы.

5	6	7
2	3	4
Запрет-ная зона	1	Запрет-ная зона

Рисунок 6.3 – Зоны печатной платы для размещения элементов

6.3 Содержание отчета

1 Цель работы.
2 Исходное задание.
3 Решение задачи заданными алгоритмами.
4 Выводы по полученным результатам.

6.4 Контрольные вопросы

1 В чем заключается задача размещения конструктивных элементов?
2 Какие критерии оптимальности используются при решении задачи размещения?
3 Из каких соображений назначаются весовые коэффициенты цепей?
4 Как рассчитываются расстояния между соседними элементами?
5 В чем заключается метод случайного поиска решения задачи размещения?
6 Какие достоинства и недостатки у последовательного алгоритма размещения?
7 Почему при решении задачи размещения в качестве критерия чаще всего используется минимум суммарной взвешенной длины соединений?

Литература

1 Деньдобренько, Б. Н. Автоматизация конструирования РЭА / Б. Н. Деньдобренько, А. С. Малика. – М. : Высш. шк., 1980.

2 Автоматизированное проектирование радиоэлектронных средств : учеб. пособие для вузов / О. В. Алексеев [и др.]; под ред. О. В. Алексеева. – М. : Высш. шк., 2000.

Св. план 2007, поз. 24

Учебное издание

Станкевич Андрей Владимирович

Теоретические основы систем автоматизированного проектирования

Лабораторный практикум
для студентов специальности I-40 02 02
«Электронные вычислительные средства»
дневной формы обучения

Редактор Т. П. Андрейченко
Корректор Е. Н. Батурчик

Подписано в печать 02.04.2007.	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ. л. 3,37.
Уч.- изд. л. 3,0.	Тираж 200 экз.	Заказ 32.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6