

**БИБЛИОТЕЧКА  
ПРОГРАММИСТА**

**Ю. М. БАЯКОВСКИЙ  
В. А. ГАЛАКТИОНОВ  
Т. Н. МИХАЙЛОВА**

# **Графтор** **Графическое расширение фортрана**



# БИБЛИОТЕЧКА ПРОГРАММИСТА

---

Ю. М. БАЯКОВСКИЙ, В. А. ГАЛАКТИОНОВ,  
Т. Н. МИХАЙЛОВА

## ГРАФОР. ГРАФИЧЕСКОЕ РАСШИРЕНИЕ ФОРТРАНА



МОСКВА «НАУКА»  
ГЛАВНАЯ РЕДАКЦИЯ  
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ  
1985

ББК 22.18  
Б 34  
УДК 519.6

Баяковский Ю. М., Галактионов В. А., Михайлова Т. Н. **Графор. Графическое расширение фортрана.**— М.: Наука. Главная редакция физико-математической литературы, 1985.— 288 с.

В книге излагаются функциональные возможности Графора — библиотеки графических подпрограмм и функций на фортране. Она служит интерфейсом между прикладной программой и конкретным графическим устройством. Описывается базисный (нижний) уровень пакета, программные средства, позволяющие изображать плоские объекты, а также программы, предназначенные для графического представления функций двух переменных. В приложении приведены сведения об особенностях установки Графора на ЕС ЭВМ, а также полный алфавитный список программ Графора.

Рецензент:

кандидат физико-математических наук Ю. И. Кетков

*Юрий Матвеевич Баяковский  
Владимир Александрович Галактионов  
Татьяна Николаевна Михайлова*

#### ГРАФОР. ГРАФИЧЕСКОЕ РАСШИРЕНИЕ ФОРТРАНА

Редактор Л. Г. Силкова  
Техн. редактор Е. В. Морозова  
Корректоры О. А. Сизал, Е. В. Сидоркина

ИБ № 12581

---

Сдано в набор 31.08.84. Подписано к печати 20.05.85.  
Т-12304. Формат 84×108<sup>1</sup>/<sub>32</sub>. Бумага тип. № 3. Гарнитура  
обыкновенная. Высокая печать. Усл. печ. л. 15,12. Усл. кр.-  
отт. 15,33. Уч.-изд. л. 17,43. Тираж 31 000 экз. Заказ № 380.  
Цена 1 р. 10 к.

---

Ордена Трудового Красного Знамени издательство «Наука»  
Главная редакция физико-математической литературы  
117071 Москва В-71, Ленинский.проспект, 15

---

4-я типография издательства «Наука»  
630077 г. Новосибирск-77, Станиславского, 25

Б  $\frac{1702070000-083}{053(02)-85}$  39-85

© Издательство «Наука»,  
Главная редакция  
физико-математической  
литературы, 1985

## ОГЛАВЛЕНИЕ

Предисловие . . . . .	5
<b>1. Базис. Структура и основные принципы . . . . .</b>	<b>7</b>
1.1. Основные понятия . . . . .	7
1.2. Особенности реализации Графора, обеспечивающие его мобильность . . . . .	10
1.3. Кадр, страница . . . . .	14
1.4. Прямая линия . . . . .	18
1.5. Тексты, числа, маркеры . . . . .	21
1.6. Графические элементы и геометрические фигуры . . . . .	26
<b>2. Линейные преобразования, след пера, экранирование, штриховка . . . . .</b>	<b>36</b>
2.1. Линейные преобразования . . . . .	37
2.2. След пера . . . . .	41
2.3. Экранирование . . . . .	47
2.4. Штриховка . . . . .	51
2.5. Примеры . . . . .	52
<b>3. Геометрические вычисления . . . . .</b>	<b>55</b>
3.1. Геометрические операции . . . . .	55
3.2. Геометрические построения . . . . .	73
<b>4. Построение графиков . . . . .</b>	<b>88</b>
4.1. Определение области. Математические координаты . . . . .	88
4.2. Графики в декартовой системе координат . . . . .	90
4.3. Построение осей координат в декартовой системе . . . . .	96
4.4. Графики в логарифмической и полулוגарифмической сетке . . . . .	100
4.5. Графики в полярной системе координат . . . . .	103
4.6. Гистограммы и календарная ось . . . . .	108
<b>5. Аппроксимация и сглаживание функций . . . . .</b>	<b>112</b>
5.1. Сплайн-аппроксимация . . . . .	112
5.2. Параметрическое задание функций . . . . .	117
5.3. Локальные сплайны . . . . .	119
5.4. Сглаживание методом наименьших квадратов . . . . .	121
5.5. Аппроксимация функции конечным рядом Фурье . . . . .	126
5.6. Линейный фильтр . . . . .	129
5.7. Аппроксимация функций ортогональными многочленами Чебышева . . . . .	132
5.8. Аппроксимация методом Безье . . . . .	134
5.9. Аппроксимация на основе B-сплайнов . . . . .	137



<b>6. Гладкое восполнение и интерполяция функций двух переменных</b>	142
6.1. Восполнение функций двух переменных, основанное на методе В. С. Рябенского	142
6.2. Восполнение функций двух переменных по методу Х. Акима	145
6.3. Примеры	146
<b>7. Построение линий уровня функций двух переменных</b>	151
7.1. Построение изолиний с использованием гладкого восполнения функций	152
7.2. Построение изолиний с использованием линейной интерполяции	158
7.3. Построение изолиний и линий пересечения поверхностей	174
<b>8. Построение плоских изображений трехмерных объектов</b>	188
8.1. Изображение функций двух переменных. Метод параллельных сечений	190
8.2. Изображение функций двух переменных. Метод ячеек	210
8.3. Триангуляция и изображение функции двух переменных, заданной в произвольно расположенных точках	222
8.4. Изображение трехмерных объектов с использованием эффекта ореола	236
<b>Приложение А. Дополнительные сведения о Графоре</b>	254
А1. Генерация комплекса Графор	255
А2. Графор и операционная система	257
А3. Каталогизированные процедуры	259
А4. Сообщения системы	261
А5. Внешние спецификации драйверов устройств для Графора в ОС ЕС ЭВМ	262
А6. Графор на ЭВМ БЭСМ-6 в ОС Диспак	265
<b>Приложение Б. Алфавитный указатель программ</b>	267
<b>Приложение В. Общие блоки и их длины</b>	283
<b>Список литературы</b>	284
<b>Предметный указатель</b>	286

## ПРЕДИСЛОВИЕ

Аббревиатура Графор известна многим программистам. Названный этим именем пакет графических программ на фортране развивается уже более десяти лет и описан в ряде препринтов и других изданий Института прикладной математики им. М. В. Келдыша АН СССР (см. [1, 2]). Изложению Графора, а также некоторым его модификациям посвящен еще ряд работ [3—10]. Первая публикация по Графору относится к 1972 году.

Несколько причин определили популярность, а следовательно, и живучесть Графора.

1. *Функциональное разнообразие.* В настоящее время в Графоре более 400 программ (см. приложение Б), которые позволяют строить графики, гистограммы, карты изолиний, проекции поверхностей, применять методы сплайн-интерполяции и сглаживания, производить аффинные преобразования и экранирование, выполнять геометрические вычисления и т. д.

2. *Связь с фортраном.* По сути своей Графор является графическим расширением фортрана, который по-прежнему остается самым популярным языком в области научных и инженерных приложений ЭВМ. Впрочем фортранная природа Графора не препятствует его использованию в программах, написанных на других языках (например, на алголе-ГДР, ПЛ/1 и, разумеется, на автокоде или языке ассемблера).

3. *Портативность (мобильность).* С прикладной программой Графор связан через стандартный фортранный интерфейс (вызов подпрограмм и функций). Все (или почти все) программы Графора реализованы на фортране. Зависимость от графических устройств четко определена и локализована так, что при подключении нового устройства не требуется сколько-нибудь значительных переделок. Имена программ и параметров строятся на основе латинского алфавита. Поэтому Графор используется как на отечественных (БЭСМ-6, ЕС ЭВМ, СМ-4, СМ-2, М-6000, М-222, БЭСМ-4, МИНСК-32 и др.), так и на зарубежных (CDC-6500, CYBER-172, ECLIPSE, NORD, PDP-11, IRIS-80) машинах. Допускается вывод информации и на графопостроители (ЕС-7051, ЕС-7052, ЕС-7053, ЕС-7054, АП-7251, АП-7252, ИТЕКАН, АТЛАС, CALCOMP, BENSON), и на дисплеи (ЕС-7064, СИГДА, ЭПГ СМ, VU-2000, ТЕКТРОНИХ).

4. *Документированность.* Графор доведен до уровня программного продукта. Имеется документация как по применению, так и по эксплуатации Графора.

В гл. 1—5 описан базисный (нижний) уровень пакета, а также представлены программы, позволяющие изобразить плоские (двумерные) объекты. Гл. 6—8 содержат описания программ, которые предназначены для графического представления пространственных (трехмерных) объектов — функций двух переменных. В Графоре

существует два способа представления объектов: проекции поверхностей и карты изолиний. Каждый из способов имеет несколько отличающихся друг от друга реализаций. В приложении приводятся сведения об особенностях установки Графора на ЕС ЭВМ, а также полный алфавитный список программ Графора.

Мы надеемся, что включенная в книгу информация в большинстве случаев будет достаточной, и лишь в редких случаях возникнет потребность в более подробном изложении и, следовательно, в обращении к препринтам или другим источникам, указанным в списке литературы. Мы не приводим здесь полных ссылок на препринты — при необходимости читатель может найти их в [2].

Нам трудно перечислить здесь всех, кто внес вклад в разработку и реализацию программ Графора, а также тех, кто своими советами, участием в обсуждениях, в освоении, в тестировании способствовал его развитию и популяризации. Всем, причастным к Графору, мы выражаем искреннюю признательность. Особо авторы благодарны А. Б. Ходулеву, который внимательно прочитал рукопись в ее первоначальном варианте и высказал ряд полезных замечаний, способствовавших существенному ее улучшению.

Авторы с благодарностью примут любые критические замечания и пожелания, касающиеся содержания и оформления этой книги.

*Ю. М. Баяковский,  
В. А. Галактионов,  
Т. Н. Михайлова*

## 1. БАЗИС. СТРУКТУРА И ОСНОВНЫЕ ПРИНЦИПЫ

Графтор представляет собой библиотеку подпрограмм и функций на фортране и служит интерфейсом между прикладной программой и конкретным графическим устройством. Такой интерфейс позволяет перейти от краткого задания сложного комплексного изображения к немногочисленным и весьма простым командам устройства.

В библиотеке Графтора более 400 программных компонент, каждая из которых относится либо к базисному уровню, либо к одной из функциональных групп. Базисный уровень определяет общую организацию библиотеки и ее мобильность по отношению к разнообразным графическим устройствам. В базисный уровень входит набор графических примитивов, к которым относятся отрезки прямых линий, дуги окружностей и эллипсов, многоугольники, числовые и текстовые цепочки, маркеры. Базисный уровень служит основой при разработке функциональных групп, каждая из которых несет очевидную предметную окраску.

### 1.1. Основные понятия

Решение любой задачи с помощью ЭВМ — это, по существу, моделирование некоторого явления или процесса. Прикладная программа, которая осуществляет такое моделирование, оперирует с объектами, представленными в памяти машины некоторыми структурами данных.

К наиболее простым и распространенным объектам можно отнести табулированную функцию одного переменного, представленную двумя одномерными массивами (значениями абсцисс и ординат). Функция двух переменных (поверхность), как правило, представляется матрицей и, возможно, двумя одномерными массивами, задающими сетку. В некоторых случаях, когда сетка равномерная (например, шаг по оси  $X$  постоянен), бывает достаточно и одного массива.

В системах машинного проектирования объекты имеют иную природу, и может потребоваться более сложное представление с привлечением таких структур, как списки и записи. В самом деле,

элементы электронной схемы (скажем, диоды, резисторы и т. п.) имеют несколько разнородных характеристик, а топология самой схемы может быть довольно сложной и определяться связями между элементами.

Далее нас будут интересовать те из объектов, которые требуется получить в виде графического образа на *видовой поверхности*,

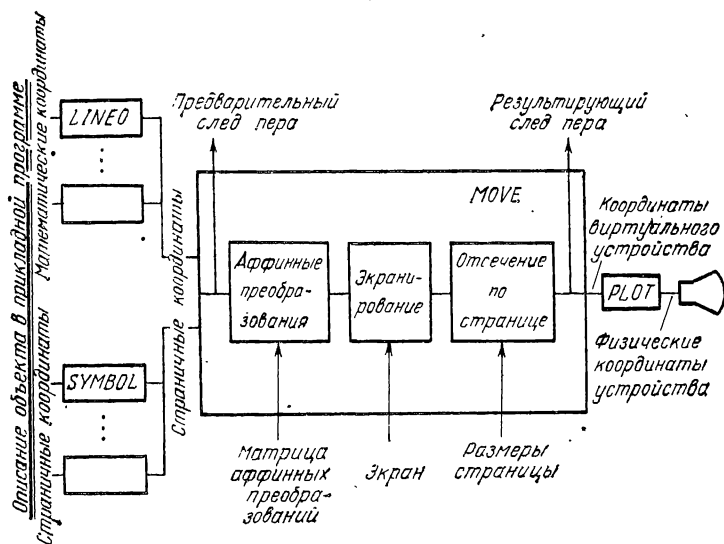


Рис. 1.1. Общая структура графической системы.

т. е. на бумаге, экране дисплея, киноплёнке или на каком-либо другом носителе. Если для описания объекта применяются так называемые *математические координаты*, т. е. единицы измерения, собственные модели (например, километры, миллибары, секунды и т. п.), то для получения графического образа необходимы *координаты устройства*, которые попадают в дисплейный файл или непосредственно используются драйвером для управления устройством.

Следовательно, графическая система должна обеспечить получение образа для заданного описания объекта с соответствующим преобразованием системы координат. Разумеется, один и тот же объект можно изобразить разными способами. Так, однозначной функции двух переменных может соответствовать плоская проекция поверхности, карта изолиний либо линий пересечения поверхности с заданными плоскостями и т. д. Выбор способа изображения — это, в сущности, выбор функции вывода в прикладной программе.

Кроме преобразования системы координат, при переходе от объекта к его образу могут потребоваться и некоторые другие преобразования. В частности, в математическом пространстве можно определить прямоугольное окно и отсечь все части объекта, не попадающие в его пределы. Далее, на видовой поверхности определяется прямоугольная область, занимающая, возможно, весь экран

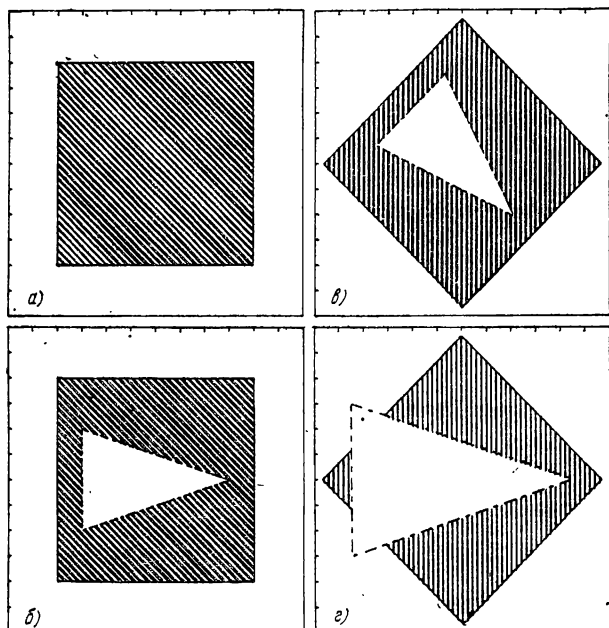


Рис. 1.2. Некоторые возможности формирования изображения:

а) На странице задается квадратный участок, который штрихуется. Граница участка очерчивается сплошной линией.

б) Задается треугольный экран. Граница экрана рисуется штрихпунктирной линией. Далее выполняются те же действия, что и в случае а).

в) Задается преобразование поворота на  $45^\circ$  вокруг центра страницы (формируется матрица аффинных преобразований) и затем выполняются те же действия, что и в случае б).

г) Устанавливается преобразование масштабирования и задается экран такой же, как в случае б). Рисуется граница экрана (в результате преобразования изменяется размер экрана, а также длины штрихов и пунктиров). Затем преобразование масштабирования отменяется. Устанавливается преобразование поворота на  $45^\circ$  и, наконец, выполняются те же действия, что и в случае а).

дисплея или, в случае графплоттера, всю страницу, и производится отображение окна на эту область. В процессе вывода к образу

объекта можно применить необходимые аффинные (линейные) преобразования. Те части образа объекта, которые закрыты экраном или оказались за пределами страницы, на видовой поверхности не изображаются. Предусматривается возможность получения *следа пера* (т. е. фиксирования его траектории при рисовании) как предварительного (до преобразований), так и истинного. Информация о следе пера может использоваться, например, для выполнения нелинейных преобразований графического образа.

В итоге общую структуру графической системы можно представлять себе такой, как она изображена на рис. 1.1. В дальнейшем мы ограничимся в основном средствами вывода и преобразования рисунка, имея в виду их конкретную реализацию в графическом пакете Графор. На рис. 1.2 показано как можно использовать при формировании изображения аффинные преобразования и экранирование.

## 1.2. Особенности реализации Графора, обеспечивающие его мобильность

В Графоре вся специфика конкретного графического устройства сосредоточена в программе PLOT и в BLOCK DATA (или в *драйвере и программе инициализации*). Это следует учитывать при адаптации комплекса для новых графических устройств.

Все начальные установки в общих блоках вынесены в отдельную подпрограмму. На БЭСМ-6 — это BLOCK DATA.

### BLOCK DATA

#### C. ОБЛАСТЬ СВЯЗИ БАЗИСНЫХ ПРОГРАММ

COMMON/GFTAB/IRDB(5), RDB(17)

DATA IRDB/3\*0, 2\*6/, RDB/1000, 0, 60., 26., 12\*0., 1000./

#### C. ВЫБОР ПЕРА

COMMON/GFCOL/NCOL, JPEN, JCOL, JCOLC, IDX, IDY,

\* IR(2, 3)

DATA NCOL/3/, JPEN/0/, JCOL/1/, JCOLC/0/,

\* IDX, IDY/2\*0/, IR/0, 0, 0, 140, 0, 280/

#### C. АФФИННЫЕ ПРЕОБРАЗОВАНИЯ

COMMON/GFATRM/A(6), B(6), N

COMMON/GFATR/R(6), IS

DATA A/1., 3\*0., 1., 0./, B/1., 3\*0., 1., 0./, N/0/

DATA R/1., 3\*0., 1., 0./, IS/0/

#### C. ЭКРАНИРОВАНИЕ

COMMON/GFBLAN/JJ, XX1, XX2, YY1, YY2, NCOR,

\* JCH, KCH, XXX, YYY, ICH

DATA JJ, KCH, XX1, XX2, YY1, YY2/0, 0, 4\*0./

#### C. СЛЕД ПЕРА

```

COMMON/GFNTCH/CHEK3(50), NSIGN
COMMON/GFGOBS/NJCH, NKCH, JG, NRIS, NACH(16)
DATA NSIGN, NKCH, JG, NRIS, NACH/0, 0, 0, 16*/
С. БУФЕР ХРАНЕНИЯ ГЕОМЕТРИЧЕСКИХ ЭЛЕМЕНТОВ
COMMON/GFGEL/GF(28)
DATA GF/8*0., 1., 18*0., 0.01745329/
С. ФИКСАЦИЯ ОШИБКИ ПРОГРАММОЙ GRAFER
COMMON/GFERR/NN, ITER(4)
DATA NN/0/
С. БЕРГ — ШТРИХИ В ИЗОЛИНИЯХ
COMMON/GFBET/KT, ISTEP, SIZE
DATA KT, ISTEP, SIZE/0, 0, 0./
END

```

В версии Графора для ЕС ЭВМ эти функции выполняет программа GRINIT. Связь в базисных программах осуществляется через общий блок GFTAB. Его состав, а также смысл содержащихся в этом блоке величин поясняются в табл. 1.

Функции, выполняемые программой PLOT, в значительной степени зависят от аппаратных возможностей графического устройства и операционной обстановки. Так, при использовании простых устройств, не имеющих аппаратной генерации линий, алгоритм линейной интерполяции включается в саму программу PLOT.

В некоторых своих вариантах комплекс Графор непосредственно управляет графическим устройством. Это неизбежно при работе на машинах, где используется однопрограммный режим и отсутствуют прерывания от устройств. В больших машинах, которые работают под управлением мультипрограммной операционной системы, выдача команд графическому устройству выполняется супервизором операционной системы (соответствующая его часть называется *графическим супервизором*). К программе PLOT, которая взаимодействует с внешней средой, предъявляются дополнительные требования.

Так, в случае использования комплекса Графор в рамках ОС Диспак на машине БЭСМ-6 с шаговым графплоттером связь с операционной системой осуществляется с помощью экстракода. При реализации Графора на ЕС ЭВМ вследствие различия в системе приказов графических устройств и каналных программ ввода/вывода нижний уровень распадается еще на два: уровень генерации графических приказов (*компилирующая программа*) и уровень собственно ввода/вывода (*хэндлер устройства*). Программа PLOT в этом случае будет выполнять функции посредника, обращаясь к соответствующим входам компилирующей программы (см. приложение А).

Привязка конкретного устройства к графическим программам верхнего уровня осуществляется с помощью программы



## Область связи

COMMON/GFTAB/IRDB(5), RDB(17)

Элемент общего блока GFTAB	Смысл	Программа, изменяющая значение
IRDB(1) IRDB(2) IRDB(3) IRDB(4)	Номер набора литер Признак страницы Свободно	SET PAGE
IRDB(5)	Количество байт для представления целого числа (для БЭСМ-6 — число 6) Количество байт для представления числа с плавающей точкой (для БЭСМ-6 — число 6)	
RDB(1)	Количество шагов виртуального устройства в выбранной единице измерения (CMS, MMS, INCHES), первоначально 1000	
RDB(2)	Наклон литеры	ITALIC
RDB(3)	Максимальные размеры страницы по оси X	
RDB(4)	Максимальные размеры страницы по оси Y	
RDB(5)	Длина стороны страницы по X	PAGE
RDB(6)	Длина стороны страницы по Y	PAGE
RDB(7), RDB(8)	XOR, YOR — координаты левого нижнего угла прямоугольной области (REGION) или центра полярной области (POLREG)	REGION, POLREG
RDB(9), RDB(10)	XLR, YLR — длины сторон прямоугольной области (REGION) или внутренний и внешний радиусы полярной области (POLREG)	REGION, POLREG
RDB(11), RDB(12), RDB(13), RDB(14)	XMIN, YMIN, XMAX, YMAX — пределы изменения функции и аргумента	LIMITS
RDB(15)	THO — начальный угол полярной области	POLREG
RDB(16)	THF — конечный угол полярной области	POLREG
RDB(17)	Число 1000 — характеристика виртуального устройства (количество шагов в сантиметре), т. е. шаг виртуального устройства выбран равным 0.01 мм	

GRINIT. Вызов этой программы необходимо выполнять до обращения к каким-либо программам Графора. Завершает вывод программа GRFIN. К ней пользователь должен обратиться после окончания работы главной программы, в противном случае часть данных (или все) может быть утеряна. Обе программы без параметров.

Программа GRINIT определяет тип выводного устройства. При этом в случае графического устройства производится загрузка соответствующей компилирующей программы и хэндлера устройства. Кроме того, устанавливается связь программы PLOT с компилирующей программой, а компилирующей программы с хэндлером устройства. После этого выполняются иницилирующие функции компилирующей программы и хэндлера устройства.

Если же выходным устройством является перфолента или магнитная лента, то программа GRINIT выдает запрос оператору с требованием указать тип графического устройства, для которого предназначается подготавливаемая лента. После ответа оператора, как и в предыдущем случае, производится загрузка соответствующей компилирующей программы и модуля графического вывода на ленту. Использование другой компилирующей программы обусловлено тем, что для некоторых устройств (например, ЕС-7054) форматы данных в режиме ONLINE отличаются от форматов в режиме OFFLINE.

Реализация нижнего уровня для других машин может отличаться от описанного выше.

Для обеспечения графических устройств, имеющих несколько пишущих элементов, в Графоре имеется установочная программа SETPEN(J), работающая с общим блоком GFCOL. Параметр этой программы: J — номер пера. Структура блока GFCOL зависит от параметров устройства и программы, реализующей работу со сменой пишущих элементов. Например, в случае использования ЕС-7052 этот блок содержит следующие величины:

- 1 — число пишущих элементов,
- 2 — состояние опущено/поднято,
- 3 — требуемый номер,
- 4 — текущий номер,
- 5 — смещение по оси X при смене пера,
- 6 — смещение по оси Y при смене пера,
- 7 — двумерный массив расстояний от первого элемента в шагах виртуального устройства по двум осям.

Если же устройство поддерживает несколько различных типов линий, то для их установки может использоваться программа SLINST(J). Параметр J — номер типа линии: J = 1 — сплошная линия, J = 2 — штрихпунктирная линия, J = 3 — штриховая линия.

В Графоре имеется еще несколько программ, зависящих от конкретной ЭВМ. Об этом нужно помнить при переносе комплекса на новые машины. Имена этих программ: SYMTAB, IDENT, IADR (функция), BCD, IBCD, RAND, MARKER.

### 1.3. Кадр, страница

Прежде всего необходимо отметить важное для дальнейшего изложения различие между дисплеем и графплоттером. Для каждого типа дисплея размер экрана и, следовательно, размер кадра постоянны и с ними жестко связана система координат устройства. Скажем, кадр на экране дисплея ЕС-7064 — это квадрат размером 250 мм × 250 мм, и по каждой координате адресуется 1024 точки. В этом случае удобно применять нормированные координаты устройства, изменяющиеся в пределах от 0 до 1. Тем самым достигается независимость от размера экрана. Правда, при таком подходе невозможно сохранить линейные размеры образов на дисплеях с разными размерами экранов. В графплоттерах, особенно в графплоттерах рулонного типа, размер кадра (применительно к графплоттерам мы будем называть его страницей) не фиксирован. Длина страницы может существенно (во много раз) отличаться от ее ширины. Кроме того, иногда (например, при подготовке фототаблонов) должна сохраняться не только форма, но и абсолютные размеры элементов рисунка.

В Графоре *страница* — это прямоугольное поле на бумаге или экране дисплея, в пределах которого должны размещаться области для графиков, тексты и другие графические элементы. При задании страницы определяются ее размер, связанная с ней система координат и фиксируется *единица измерения*. Существуют три программы CMS, MMS и INCHEs, устанавливающие в качестве единицы измерения сантиметр, миллиметр и дюйм соответственно. Эти программы не имеют параметров. Если ни одна из этих программ не вызывалась, то устанавливаются сантиметры. Единица измерения выбирается до определения страницы (размеры самой страницы указываются в этих единицах) и не может быть изменена внутри страницы. Таким образом, масштаб рисунка не зависит от шага конкретного графплоттера или размеров экрана дисплея.

Программа PAGE(XL, YL, NAME, N, J) позволяет определить страницу и имеет следующие параметры:

XL, YL — размеры страницы вдоль осей X и Y;

NAME — название страницы,

N — количество литер в названии;

J — признак очерчивания границы: J = 0 — граница не очерчивается, J = 1 — граница очерчивается,

Определение страницы имеет принципиальное значение. Если страница не определена, то любые обращения к графическим программам игнорируются. При определении страницы считается, что перо графплоттера находится в левом нижнем углу страницы, т. е. в точке с координатами  $(0, 0)$ , тогда правый верхний угол страницы имеет координаты  $(XL, YL)$ . Поэтому расположение первой

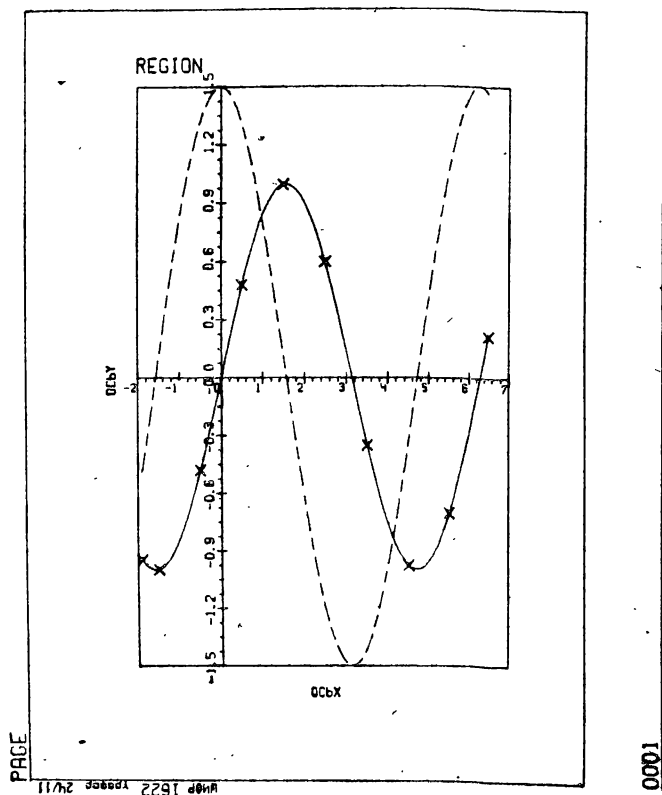


Рис. 1.3. Пример построения графика функции.

страницы зависит от действий человека, включившего устройство и подготовившего его к работе. Расположение каждой следующей страницы выбирается автоматически программой ENDPG. Существуют предельные, т. е. максимально допустимые, размеры страницы, которые зависят от выбора графического устройства. Если какая-либо из координат выходит за указанные пределы, она полагается равной предельному значению. Перо в процессе рисова-

ния не может выйти за границы, установленные программой PAGE (даже, если они не очерчены). Его можно вывести за пределы страницы только после того, как страница будет закрыта. Поэтому, если графический элемент или какая-то его часть лежит за пределами страницы, то либо весь элемент не рисуется, либо изображается только его часть, находящаяся в пределах страницы.

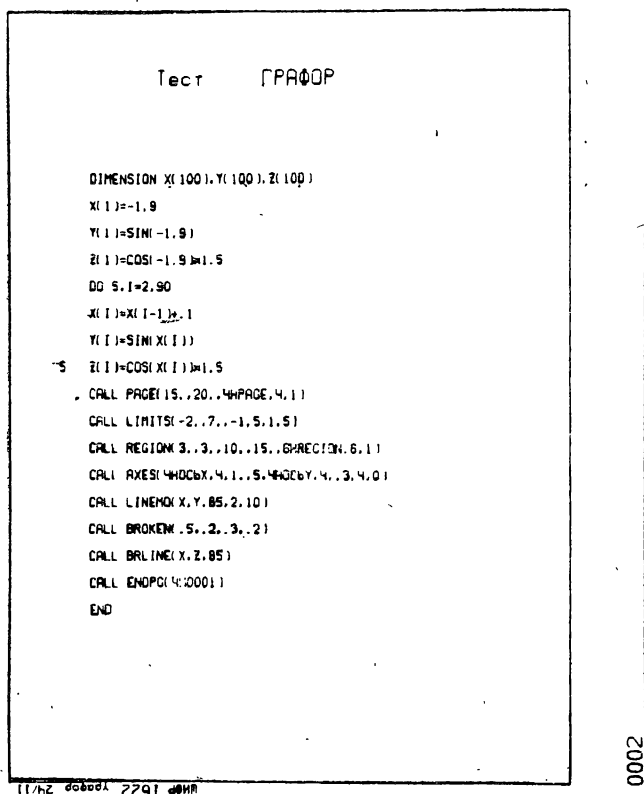


Рис. 1.4. Программа, с помощью которой выполнено построение графика на рис. 1.3.

Такая локализация пера внутри рабочего поля позволяет в случае ошибки сохранить в неприкосновенности ранее подготовленные страницы.

Название страницы пишется за ее пределами вдоль левой границы. Также за пределами страницы вдоль нижней границы пишутся имя (пифр) задачи и дата решения (рис. 1.3 и 1.4). Эти данные извлекаются из операционной системы с помощью про-

граммы IDENT(MD, ND), которая формирует полученные из операционной системы имя (шифр) задачи MD и дату ND.

При любом перемещении пера из точки в точку запоминаются его текущие координаты как результирующие, так и предварительные.

*Результирующие координаты* пера — это те, которые получаются после применения всех преобразований (линейных, экранирования, отсечения по странице).

*Предварительные координаты* — это координаты до выполнения преобразований.

Узнать значения этих координат можно с помощью подпрограмм WHERE и WHERP.

Программа WHERE(X, Y, F) позволяет узнать результирующие координаты пера на странице (параметры X, Y), а также коэффициент F перевода сантиметра в выбранные единицы измерения.

Программа WHERP(X, Y, F) аналогична предыдущей, но позволяет определить предварительные координаты текущего положения пера (параметры X, Y) и коэффициент F перевода сантиметра в выбранные единицы измерения.

После окончания вывода на текущую страницу ее необходимо закрыть, только после этого можно определить следующую страницу.

Программа ENDPG(NUMB) закрывает доступ к текущей странице и выполняет начальные установки, необходимые для определения следующей страницы. При этом в случае графопостроителя перо выводится за пределы страницы, проводится линия разреза между страницами параллельно оси Y на расстоянии 2 см от правой границы закрываемой страницы. Вдоль линии разреза пишется текст, заданный параметром NUMB, затем перо отводится от линии разреза вправо на расстояние 2 см в точку, которая станет левым нижним углом новой страницы. После того как страница закрыта, может быть изменена и единица измерения.

Обращение к программе ENDPG обязательно не только между текущими страницами данной задачи, но и после последней страницы, чтобы тем самым исключить наложение на эту страницу рисунков другой задачи.

В Графоре имеется также программа NEWFRM, которая предназначена для смены кадра при выводе изображения на дисплей или планшетный графплоттер. Обращение к этой программе приостанавливает работу основной программы. Продолжить ее можно лишь после вмешательства оператора. Например, на ЕС ЭВМ с дисплеем ЕС-7064 для этого необходимо нажать функциональную клавишу. При этом очищается память дисплея и возобновляется вывод.

## 1.4. Прямая линия

В графическом пакете каждому объекту соответствует подпрограмма (функция вывода), интерпретирующая описание объекта и генерирующая его образ как композицию других более простых объектов. К ним относятся: прямая линия, алфавитно-цифровые и специальные знаки, маркеры, правильные многоугольники, прямоугольники, дуги окружности и эллипсы и др. Но примитивным (неразложимым) является только один объект — *отрезок прямой линии*.

В Графоре роль генератора прямых линий выполняет программа MOVE. Эта программа позволяет перевести перо из точки, в которой оно находится в текущий момент, в любую другую точку внутри страницы. При этом перо может перемещаться, оставляя след на бумаге, если оно опущено, или не оставляя следа, если оно поднято. Заметим, что точкой отсчета считается точка с нулевыми координатами в левом нижнем углу страницы. Поэтому координаты всех других точек задаются относительно этого угла в тех единицах измерения, которые были выбраны перед определением страницы. Программа MOVE выполняет преобразование заданных координат в шаги виртуального графплоттера. Непосредственно же управляет движениями пера программа PLOT.

Назначение программы PLOT состоит в том, чтобы отобразить виртуальное устройство на то или иное конкретное устройство. В ней учитываются все специфические особенности используемого устройства. Поэтому при смене устройства программа PLOT подвергается изменениям. Это существенно, поскольку при наличии весьма разнообразных графических устройств графический пакет должен быть относительно независим от устройств и от способа их подключения. К программе PLOT не следует обращаться из прикладной программы.

Программа PLOT(IX, IY, J) позволяет присвоить указанные значения координат точке, в которой в данный момент находится перо, или перевести перо в указанную точку с вычерчиванием или без вычерчивания прямой линии. Параметры программы:

IX, IY — целочисленные значения координат X и Y, заданные в шагах (размер шага равен 0.01 мм);

J — управление пером:

J = 0 — присвоить значения координат точке, в которой находится перо,

J = -2 — перевести перо из текущей точки в точку с координатами (IX, IY) без вычерчивания линии,

J = 2 — перевести перо из текущей точки в точку с координатами (IX, IY) с вычерчиванием прямой линии.

Программа  $\text{MOVE}(X, Y, J)$  позволяет переместить перо из текущей точки в точку с заданными координатами без рисования, провести прямую от текущей точки к точке с заданными координатами или присвоить заданные значения точке, в которой находится перо. Параметры программы:

$X, Y$  — координаты точки, в которую перемещается перо;

$J$  — признак вычерчивания линии:  $J = 0$  или  $990$  — перемещение без рисования,  $J = 1$  или  $991$  — перемещение с рисованием,  $J = 992$  — присвоить текущей точке значения координат  $X, Y$ .

Заметим, что если  $J = 0$  или  $J = 1$  и значение какой-либо из координат находится за пределами страницы, то этой координате присваивается граничное значение. При  $J = 990, 991$  такой контроль не производится и разрешается рисование за пределами страницы.  $J = 990, 991, 992$  — допускаются только в программах Графора.

Таким образом, в результате срезки по границе вблизи нее могут возникнуть некоторые искажения рисунка. Такое отсечение, однако, носит чисто технический характер. Для точного отсечения следует воспользоваться программой экранирования.

Существует еще несколько вариантов программы  $\text{MOVE}$ , учитывающих режимы работы с экранированием ( $\text{MOVE1}, \text{MOVE3}$ ) и формированием следа пера ( $\text{MOVE2}, \text{MOVE3}$ ). Подробнее об этом сказано в §§ 2.2 и 2.3.

Итак, программа  $\text{MOVE}(X, Y, J)$  строит отрезок по координатам конечной его точки. В качестве начальной точки отрезка берется текущая точка, т. е. та, в которой оказалось перо перед обращением к соответствующей программе.

Как вы уже заметили, при описании программ термином *перо* обозначается не только перо графплоттера, но и пишущий элемент любого графического устройства, например электронный луч ЭЛТ. Если при перемещении пишущего элемента остается след на видовой поверхности, то мы будем говорить, что *перо опущено*, в противном случае — *перо поднято*.

В Графоре имеется еще несколько программ для перемещения пера по прямой в опущенном или поднятом состоянии.

Программа  $\text{MOVA}(DL, TH, J)$  строит отрезок по его длине  $DL$  и углу с осью  $x$ . Угол  $TH$  задается в градусах (рис. 1.5, а).

Программа  $\text{MOV}(DX, DY, J)$  строит отрезок по приращением координат  $DX$  и  $DY$  вдоль осей  $x$  и  $y$  соответственно (рис. 1.5, б).

Программа  $\text{MOV}(XM, YM, DL, J)$  строит отрезок по его длине  $DL$  и точке  $(XM, YM)$ , лежащей на отрезке или его продолжении. Если  $DL > 0$ , то перо из начальной точки движется в сторону точки  $(XM, YM)$ , в противном случае — в противоположную. Точка  $(XM, YM)$  может находиться вне страницы (рис. 1.5, в).



Для этих программ:  $J = 0$  — перо поднято,  $J = 1$  — перо опущено.

В рисунках и технических чертежах используются, как правило, различные типы линий: сплошные, штриховые, штрихпунктирные, линии различной толщины. Программы для построения таких линий также описаны в этом разделе.

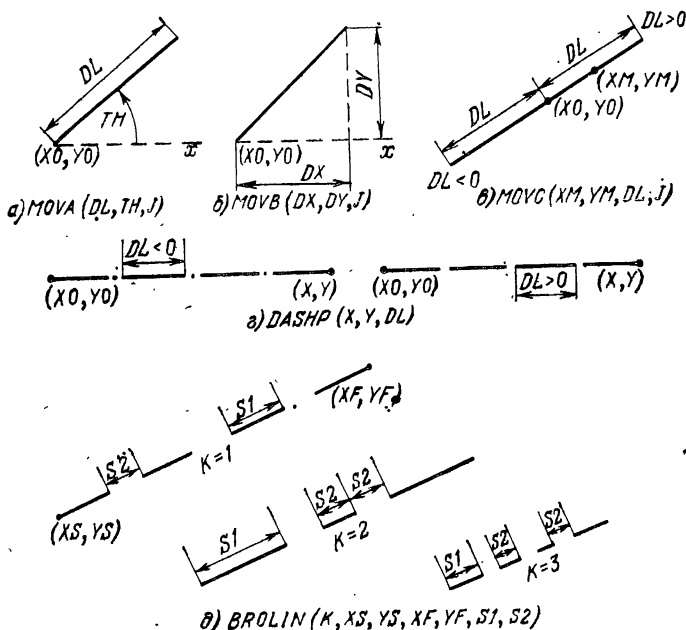


Рис. 1.5. Типы прямых линий.

Программа  $FATLIN(X, Y, D)$  позволяет провести из текущей точки в заданную точку прямую линию указанной толщины. Ее параметры:

$X, Y$  — координаты конечной точки,

$D$  — толщина линии в миллиметрах.

Поскольку толщина линии достигается лишь трехкратным проведением, при больших  $D$  линия может оказаться не «налитой».

Программа  $DASHP(X, Y, DL)$  позволяет провести из текущей точки в заданную точку штриховую или штрихпунктирную прямую линию (рис. 1.5, г). Параметры программы:

$X, Y$  — координаты точки, в которую проводится линия;

$|DL|$  — длина основного штриха линии:  $DL > 0$  — проводится штриховая линия,  $DL < 0$  — проводится штрихпунктирная линия.

В штриховой линии расстояние между штрихами (интервал) равно  $0.3 * |DL|$ . В штрихпунктирной линии длина промежутка между штрихами равна  $0.23 * |DL|$  и в середине этого промежутка изображается дополнительный штрих размером  $0.03 * |DL|$ .

Программа BROLIN(K, XS, YS, XF, YF, S1, S2) позволяет провести три типа штриховых линий (рис. 1.5, *б*). Параметры программы:

K — тип линии: K = 1 — штриховая линия, K = 2 — штрихпунктирная линия, K = 3 — штрихпунктирпунктирная линия;

XS, YS — координаты начальной точки;

XF, YF — координаты конечной точки;

S1 — длина штриха;

S2 — длина пунктира и интервала.

## 1.5. Тексты, числа, маркеры

**1.5.1. Вывод текстовой информации.** В некоторых устройствах имеются аппаратные генераторы символов. Но они обладают довольно ограниченными возможностями в отношении выбора шрифтов, размеров литер и углов наклона текстовых строк. Кроме того, во многих устройствах, особенно в графплоттерах, такие генераторы вообще отсутствуют.

При программной реализации генератора символов предполагается, что каждая литера кодируется как последовательность штрихов. Следовательно, необходимо иметь таблицу с описанием каждой литеры. При большом количестве литер (в Графоре, например, их 176) проблемы компактного описания литеры и экономной организации таблицы становятся весьма серьезными.

В Графоре используется известный и хорошо зарекомендовавший себя способ кодирования образа литеры, при котором каждая литера описывается последовательностью не более чем шестнадцати штрихов на сетке размером  $4 \times 9$  (рис. 1.6).

Ширина литеры составляет  $4/7$  высоты прописной буквы, расстояние между двумя соседними литерами равно половине ширины литеры.

Генератор символов состоит из двух программ. Программа SYMTAB, написанная на автокоде, содержит *таблицу кодов литер*. При вызове этой программы указываются слово и номер литеры в слове, а также номер набора (все множество литер разделено на 4 набора, текущим считается тот набор, который был задан при вызове программы SET). Результатом работы программы SYMTAB является массив пар координат для последовательности штрихов, воспроизводящих указанную литеру, и величина, равная фактическому количеству штрихов.

Программа SYMTAB(JTEXT, I, NPLOT, KK, ISET) позволяет выбрать из таблицы массив координат для построения литеры. Параметры программы:

JTEXT — слово текста;

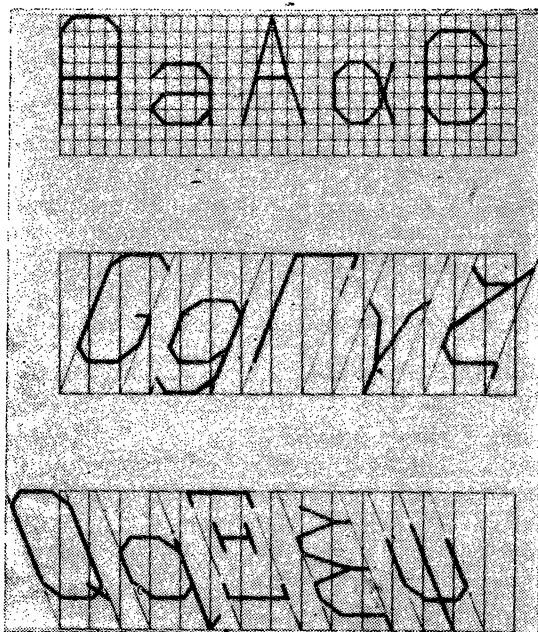
I — номер литеры в слове;

NPLOT — массив координат для построения литеры;

KK — количество пар координат, задающих литеру;

ISET — номер набора.

Основная же программа генератора символов SYMBOL написана на фортране. Эта программа пишет строку текста заданной



```
CALL ITALIC(1)  
CALL ITALIC(0)  
CALL ITALIC(-1)
```

Рис. 1.6. Примеры изображения литер.

длины под заданным углом к оси X. Строку можно начать либо в указанной точке, либо указать ее место относительно конца предыдущей строки.

При работе программы SYMBOL строка текста выводится на графплоттер последовательно, литера за литерой, причем, если ко-

ординаты очередной литеры оказываются за пределами страницы, то она не рисуется. Таким образом, выводится только та часть строки текста, которая размещается в пределах страницы.

Программа SYMBOL(X, Y, SIZE, JTEXT, N, THETA) позволяет написать заданный текст. Параметры программы следующие:

X, Y — координаты левого нижнего угла первой литеры текста или приращения к конечным координатам текста, предшествующего данному;

SIZE — высота прямоугольника, в котором вычерчивается литера;

JTEXT — заданный текст;

|N| — количество литер в строке:  $N > 0$  — (X, Y) считаются координатами на странице,  $N < 0$  — (X, Y) считаются приращениями;

THETA — угол наклона строки текста к оси X (в градусах).

Программа SET(J) позволяет выбрать один из четырех наборов литер (рис. 1.7). Имеются наборы, в которые входят:

- 1) прописные русские и латинские буквы, цифры и знаки,
- 2) строчные русские и латинские буквы, цифры и знаки,
- 3) прописные греческие буквы, цифры, знаки и спецсимволы,
- 4) строчные греческие буквы, цифры, знаки и спецсимволы.

Цифры и знаки во всех наборах одинаковы.

Обращаться к программе SET можно как до, так и после определения страницы. Информация о выбранном наборе будет сохраняться до следующего обращения к программе SET в пределах одной страницы. Если не было ни одного обращения к программе SET, то используется набор, содержащий прописные русские и латинские буквы. Этот набор восстанавливается и при закрытии каждой страницы.

Кроме обычного прямого шрифта, можно получить и *курсив*, установив соответствующий режим обращением к программе ITALIC(J). Ее параметр J задает признак курсива:  $J = 0$  — прямой шрифт,  $J = 1$  — правый наклон,  $J = -1$  — левый наклон. Тогда угол наклона каждого символа к строке текста будет составлять около  $70^\circ$  или  $110^\circ$  (наклон может быть соответственно вправо или влево).

Обращаться к программе ITALIC можно любое число раз как до, так и после определения страницы. Установленный режим наклона символов распространяется на все тексты до следующего обращения к программе ITALIC в пределах одной страницы. Если не было ни одного обращения к этой программе, действует стандартный режим — прямой шрифт. Этот же стандартный режим устанавливается при закрытии страницы. Примеры работы с программой ITALIC приведены на рис. 1.6.

SET(0) /+- , )\$\*.(=':БГДЖЗИЙАЛФЦЧШЩЬЪЮЯАВВСДЕФГHI JKL MNOPQRSTUVWXYZ0123456789

```
SET(1) /+- . !$%,('=:беджзшщлфцчщпбвгяабсдефghijklmnopqrstuvwxyz0123456789
```

SET(2) /+-. )\$%.! '=':~+ABΓΔΕΦΗΘΚΛΜΝΟΠΡΣΤΥΦΧΨΩ123456789

SET(3) /+- . )\$\*( =': √<>||[ ]↑↓↔-∂\_!@%αβγδεφγηθικλμνοπξρστψχζ0123456789

**Рис. 1.7. Наборы литер.**

**1.5.2. Вывод чисел.** В Графоре роль генератора чисел выполняет программа NUMBER. Она дает возможность написать на странице число, представленное в машине в нормализованном двоичном виде. Форма обращения к программе NUMBER во многих отношениях аналогична форме обращения к программе SYMBOL. Программа NUMBER переводит числа в текстовый вид и затем с помощью программы SYMBOL рисует их.

Число может состоять из знака числа, целой части, точки и дробной части. Если число положительно, то знак опускается. Количество дробных знаков задается в обращении к программе. Если оно равно 0, то точка не пишется.

Программа NUMBER(X, Y, HGF, FNUM, N, TH) дает возможность перевести число в текстовый вид и написать его. Параметры программы:

X, Y — координаты точки, в которой надо начать писать число, или приращения к конечным координатам текста, предшествующего данному;

|HGF| — высота прямоугольника, в котором вычерчивается цифра:  $HGF > 0$  — (X, Y) считаются координатами на странице,  $HGF < 0$  — (X, Y) считаются приращениями;

FNUM — заданное вещественное число;

N — количество требуемых дробных знаков.

Имеются, кроме того, две программы (BCD, IBCD) для перевода чисел из внутренней машинно-зависимой формы в текстовый вид, приемлемый для программы SYMBOL.

Программа BCD(FNUM, JT, N) предназначена для перевода вещественных чисел, а программа IBCD(NUM, JT) — для перевода целых чисел. Здесь FNUM, NUM — задаваемые числа, JT — переведенное число в текстовом виде, N — число дробных знаков.

Однако эти программы обладают несколько ограниченными возможностями. Они рассчитаны на то, что выводимое число занимает не более шести текстовых позиций (с учетом знака и точки, если они требуются). Поэтому, если не помещается дробная часть, то выдается столько знаков после точки, сколько можно поместить. Если не помещаются целая часть числа и точка, то вместо переведенного числа появляется текст `_____**`. Число, которое занимает меньше шести позиций, дополняется пробелами слева.

Существует несколько реализаций этих программ как на автокоде, так и на фортране. В последнем варианте используется служебная программа BCDX. В зависимости от используемой ЭВМ переведенное число может помещаться в одном машинном слове (БЭСМ-6) или занимать первые 6 байт в массиве из нескольких слов (ЕС ЭВМ).

Программы BCD и IBCD являются служебными для некоторых программ Графоре.

**1.5.3. Вывод маркеров.** Маркеры предназначены для выделения точек. Там, где находилось перо в момент обращения к программе MARKER, рисуется точка, затем вокруг этой точки рисуется маркер. По окончании рисования перо выводится в позицию, откуда начиналось рисование. Величина маркера такова, что он вписывается в квадрат размером  $3 \times 3$  мм. Задавая при обращении к программе номер маркера отрицательным, можно нарисовать маркер

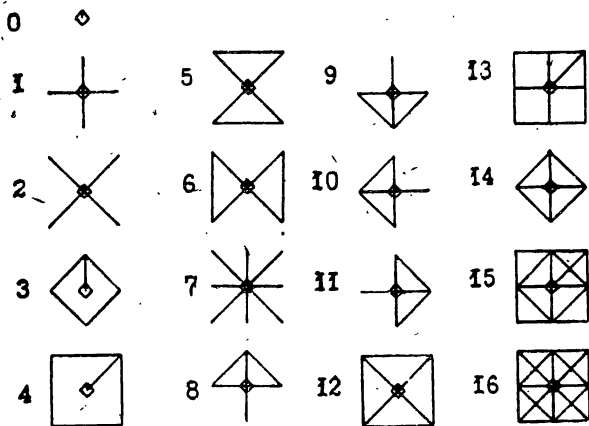


Рис. 1.8. Маркеры.

вдвое меньшего размера. В Графоре существует 17 различных маркеров. Они показаны на рис. 1.8 (в увеличенном размере).

Сложные маркеры строятся как суперпозиции простых. Точка рисуется как маленький ромбик (маркер № 3).

Программа MARKER(NMARK) предназначена для вычерчивания маркеров. Здесь NMARK — номер маркера,  $|NMARK| \leq 16$ . Если  $NMARK < 0$ , рисуется маркер вдвое меньшего размера.

## 1.6. Графические элементы и геометрические фигуры

В этом разделе описаны программы для построения различных геометрических элементов и фигур. Их линейные размеры задаются в предварительно выбранных единицах измерения.

**1.6.1. Многоугольники.** Программа TRIGL (X0, Y0, W, H, THETA, SLOPE) рисует треугольник с заданными основанием, высотой и углом между основанием и одной из сторон (рис. 1.9, а). Параметры программы:

$X_0, Y_0$  — координаты начальной точки;  
 $W$  — длина основания;  
 $H$  — высота треугольника;

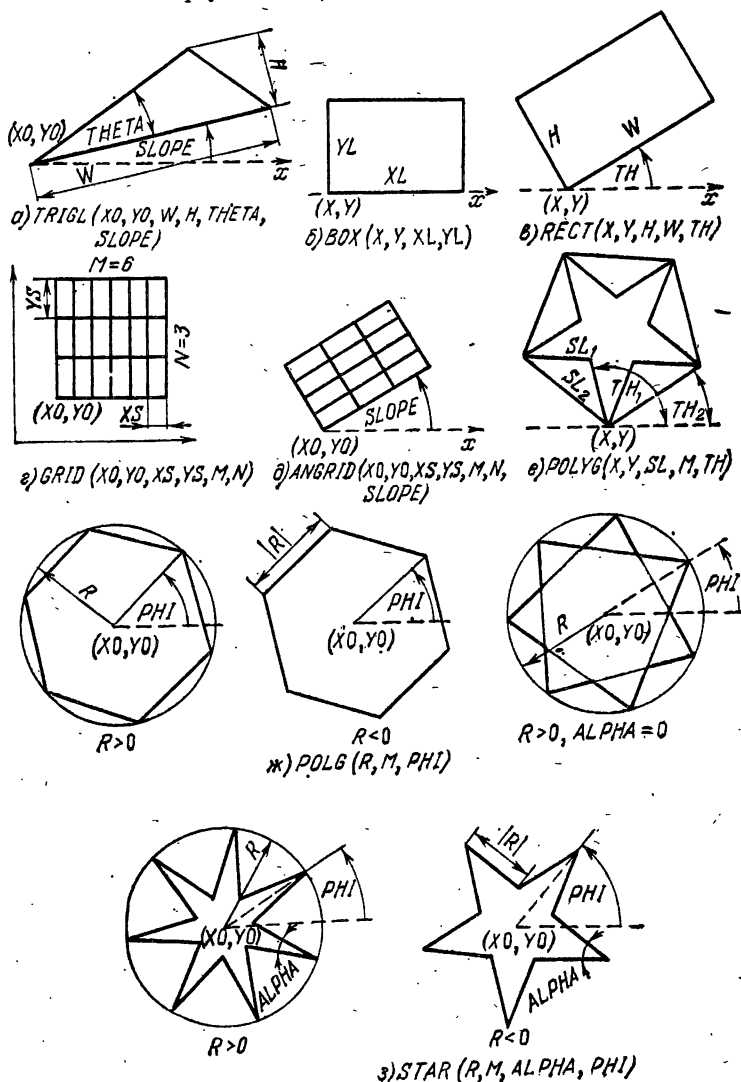


Рис. 1.9. Многоугольники.

$THETA$  — угол, образованный пересечением двух сторон (в градусах);

$SLOPE$  — угол наклона основания к оси  $x$  (в градусах).



Программа BOX( $X, Y, XL, YL$ ) позволяет начертить прямоугольник заданных размеров со сторонами, параллельными осям. Перо выводится в точку ( $X, Y$ ) и далее чертится прямоугольник в направлении по часовой стрелке (рис. 1.9, б). Параметры программы:

$X, Y$  — координаты левого нижнего угла прямоугольника;

$XL$  — размер стороны, параллельной оси  $x$ ;

$YL$  — размер стороны, параллельной оси  $y$ .

Программа RECT( $X, Y, H, W, TH$ ) позволяет начертить прямоугольник заданных размеров с поворотом на заданный угол. Прямоугольник чертится из точки ( $X, Y$ ) в направлении по часовой стрелке (рис. 1.9, в). Параметры программы:

$X, Y$  — координаты левого нижнего угла прямоугольника;

$H, W$  — длины сторон прямоугольника;

$TH$  — угол поворота вокруг точки ( $X, Y$ ) относительно оси  $x$  (в градусах).

Программа GRID( $X0, Y0, XS, YS, M, N$ ) строит равномерную прямоугольную сетку со сторонами, параллельными осям координат (рис. 1.9, г). Ее параметры:

$X0, Y0$  — координаты левой нижней вершины прямоугольника;

$XS, YS$  — шаги разбиения сетки по ширине и высоте;

$M$  — количество шагов по ширине сетки;

$N$  — количество шагов по высоте сетки.

Программа ANGRID( $X0, Y0, XS, YS, M, N, SLOPE$ ) позволяет нарисовать прямоугольную сетку с поворотом на заданный угол. Параметры программы аналогичны соответствующим параметрам программы GRID. SLOPE задает угол наклона основания сетки к оси  $x$  (рис. 1.9, г, д).

Программа POLYG( $X, Y, SL, M, TH$ ) позволяет начертить правильный многоугольник (рис. 1.9, е). Параметры программы:

$X, Y$  — координаты начальной точки;

$SL$  — размер стороны многоугольника;

$|M|$  — число сторон многоугольника:  $M > 0$  — рисуется выпуклый многоугольник,  $M < 0$  — рисуется звездчатый многоугольник;

$TH$  — угол наклона к оси  $x$  стороны, с которой начинается рисование (в градусах).

Программа POLG( $R, M, PHI$ ) изображает выпуклый правильный многоугольник с центром в текущей точке (рис. 1.9, ж). Ее параметры:

$|R|$  — радиус описанной окружности или длина стороны:  $R > 0$  — задан радиус описанной окружности,  $R < 0$  — задана сторона многоугольника;

$M$  — число сторон многоугольника;

$PHI$  — угол от оси  $x$  до ближайшего луча, проведенного в вершину (в градусах).

Программа STAR(R, M, ALPHA, PHI) рисует правильный звездчатый многоугольник с центром в текущей точке (рис. 1.9, з). Ее параметры:

$|R|$  — радиус описанной окружности или длина стороны:  $R > 0$  — задан радиус описанной окружности,  $R < 0$  — задана сторона многоугольника;

M — число лучей в многоугольнике;

ALPHA — угол раствора луча в градусах: ALPHA = 0 — правильная звезда, ALPHA  $\geq 180^\circ * (M - 2) / M$  — правильный выпуклый M-угольник;

PHI — угол между осью  $x$  и ближайшим лучом, измеряемый против часовой стрелки.

#### 1.6.2. Окружности и их дуги, спирали, сектор кругового кольца.

Прежде чем перейти к описанию программы построения дуг окружностей, отметим общую особенность ряда программ, описываемых в данной главе. Если соотношение фактических параметров при обращении в какой-либо из программ геометрически не позволяет выполнять указанную операцию (например, расстояние между начальной и конечной точками дуги больше диаметра), то такое обращение мы будем называть *некорректным*. При некорректном обращении на печатающее устройство выдается диагностический текст НЕ ТЕ ПАРАМЕТРЫ, после чего происходит выход из данной программы без выполнения каких-либо действий. При этом, если в программе некоторым переменным должны быть присвоены какие-то значения, то такого присваивания тоже не произойдет. Об этом следует помнить, если значения этих переменных используются в последующих программах. От некорректного обращения следует отличать обращение с параметрами, значения которых лежат вне допустимой области определения (например, отрицательные числа для страничных координат). В последних случаях никакой диагностической печати не выдается, а результаты работы программ не определены.

Программа ARC(X1, Y1, X2, Y2, X3, Y3, J) позволяет провести окружность или дугу окружности по трем заданным точкам. Параметры программы:

X1, Y1; X2, Y2; X3, Y3 — координаты трех точек;

J — признак вычерчивания дуги: J = 1 — проводится дуга от первой точки к третьей через вторую, J = 0 — проводится полная окружность, J = -1 — от первой точки к третьей проводится дуга, не проходящая через вторую точку.

Программа CIRC(R) позволяет начертить окружность заданного радиуса с центром в текущей точке. После вычерчивания окружности перо возвращается в исходную точку. Параметр программы:

R — радиус окружности.

Программа ARCIA(R, THS, THF) проводит из текущей точки дугу окружности с заданным наклоном начального и конечного радиусов (рис. 1.10, а). Параметры программы:

$|R|$  — величина радиуса:  $R > 0$  — перемещение пера из начальной точки в конечную против часовой стрелки,  $R < 0$  — перемещение пера по часовой стрелке;

THS, THF — углы наклона начального и конечного радиусов к оси  $x$  (в градусах).

Если  $|THS - THF| > 360^\circ$ , обращение к программе считается некорректным, хотя геометрически построение такой дуги возможно.

Программа ARCIB(R, XF, YF, J) соединяет текущую точку с заданной точкой дугой окружности заданного радиуса. Параметры программы:

$|R|$  — величина радиуса:  $R > 0$  — перемещение пера из начальной точки в конечную против часовой стрелки,  $R < 0$  — перемещение пера по часовой стрелке;

XF, YF — координаты конечной точки;

J — признак выбора дуги:  $J = 0$  — короткая дуга ( $\leq 180^\circ$ ),  $J = 1$  — длинная дуга ( $\geq 180^\circ$ ).

Если расстояние между концами дуги равно диаметру окружности, то оба варианта дуг, соответствующие разным значениям J, совпадают. Если начальная точка совпадает с конечной, то для любого знака параметра R вычерчивается полная окружность с углами наклона начального и конечного радиусов соответственно  $0^\circ$  и  $360^\circ$ . Если расстояние между заданными концами дуги превышает величину диаметра, то обращение к программе считается некорректным.

Программа ARCIC(XM, YM, XF, YF, J) проводит в заданную точку из текущей дугу окружности, проходящей через дополнительно указанную точку. Параметры программы:

XM, YM — координаты дополнительной точки дуги;

XF, YF — координаты конечной точки дуги;

J — признак включения дополнительной точки:  $J = 0$  — дуга проходит через дополнительную точку,  $J = 1$  — дуга не проходит через дополнительную точку.

Дополнительная точка может находиться и вне страницы в тех случаях, когда дуга через нее не проходит. Если начальная точка совпадает с конечной, то вычерчивается полная окружность с диаметром, равным расстоянию между начальной и дополнительной точками. Если все три точки лежат на одной прямой, то обращение к программе считается некорректным. Программа ARCIC использует служебную программу ARCC1.

Программа ARCID(XC, YC, PHI) из текущей точки для заданных координат центра окружности (XC, YC) проводит дугу

заданной угловой величины (PHI) (рис. 1.10, б). Значение угла задается в градусах. Положительным значениям угла соответствует направление против часовой стрелки. Если  $|PHI| > 360^\circ$ , то обращение к программе считается некорректным.

Программа FATARC(R, XF, YF, J, D) аналогична программе ARCIB, но проводит дугу утолщенной линией. Параметры

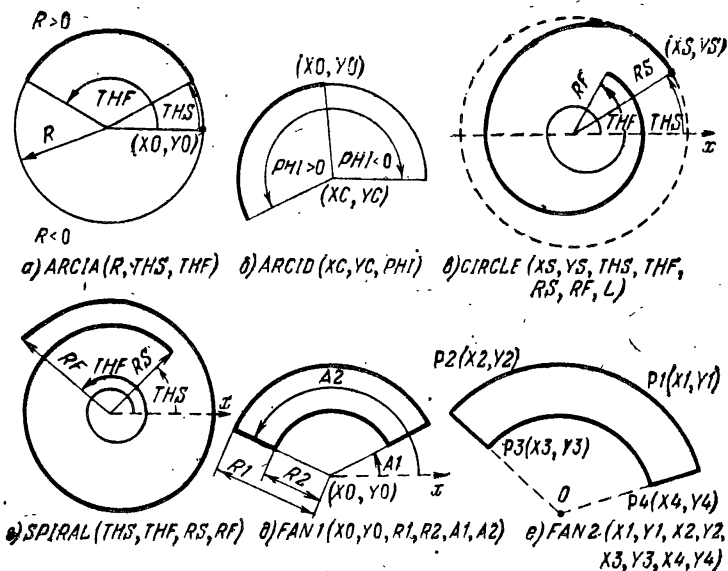


Рис. 1.10. Окружности, дуги окружностей и спирали.

R, XF, YF, J имеют тот же смысл, что и одноименные параметры программы ARCIB. Параметр D задает (как и в программе FATLIN) толщину линии в миллиметрах.

Программа CIRCLE(XS, YS, THS, THF, RS, RF, L) предназначена для вычерчивания окружностей, спиралей, дуг окружностей и дуг спиралей (рис. 1.10, е). Параметры программы:

XS, YS — координаты начальной точки;

THS, THF — углы наклона начального и конечного радиусов к оси  $x$  (в градусах);

RS — начальный радиус;

RF — конечный радиус;

L — признак непрерывности линии:  $L = 0$  — штриховая линия,  $L = 1$  — сплошная линия.

Если начальный и конечный радиусы равны, вычерчиваются окружности и дуги окружностей. При  $RF < RS$  получаем скручивающуюся спираль, а при  $RF > RS$  — раскручивающуюся.

Программа SPIRAL(THS, THF, RS, RF) позволяет начертить спираль с началом в текущей точке (рис. 1.10, *з*). Параметры программы:

THS, THF — углы наклона начального и конечного радиусов к оси  $x$  (в градусах);

RS, RF — начальный и конечный радиусы.

Если  $RF > RS$ , рисуется раскручивающаяся спираль, а если  $RF < RS$ , то скручивающаяся. Если  $RF = RS$ , то рисуется дуга окружности.

Программа FAN1(X0, Y0, R1, R2, A1, A2) позволяет начертить сектор кругового кольца («веера») (рис. 1.10, *д*). Параметры программы:

X0, Y0 — координаты центра окружностей;

R1 — радиус внешней дуги;

R2 — радиус внутренней дуги;

A1, A2 — углы наклона сторон сектора к оси  $x$ .

Программа FAN2(X1, Y1, X2, Y2, X3, Y3, X4, Y4) позволяет начертить сектор кругового кольца по заданным значениям координат четырех вершин (рис. 1.10, *е*). Эти вершины должны быть расположены так, чтобы  $P1P4 = P2P3$  и  $P4O = P3O$ .

Программы FAN1 и FAN2 используют служебную программу ARC1, которая проводит в указанном направлении дугу окружности по заданным координатам центра, начальной и конечной точек.

**1.6.3. Эллипсы, дуги эллипсов.** Для вычерчивания эллипсов и их дуг в Графоре имеется четыре программы.

Программа ELIPS(XS, YS, A, B, ALPHA, THS, THF) позволяет начертить эллипс или дугу эллипса (рис. 1.11, *а*). Параметры программы:

XS, YS — координаты начальной точки;

A, B — размеры полуосей;

ALPHA — угол наклона полуоси  $A$  эллипса к оси  $x$ ;

THS — угол между начальным диаметром и осью  $A$ ;

THF — угол между конечным диаметром и осью  $A$ .

При равных осях получается окружность или ее дуга. Углы задаются в градусах.

Программа ELPS(A, B, ALPHA) позволяет начертить эллипс с заданными размерами полуосей с центром в текущей точке и с заданным наклоном полуоси  $A$  к оси  $x$  (рис. 1.11, *б*). Параметры программы:

A, B — размеры полуосей;

ALPHA — угол наклона полуоси  $A$  эллипса к оси  $x$ .

Программа ARCELA(A, B, ALPHA, THS, THF) позволяет провести из текущей точки дугу эллипса (рис. 1.11, *в*). Параметры программы:

$|A|$  — размер большой полуоси:  $A > 0$  — перемещение пера против часовой стрелки,  $A < 0$  — перемещение пера по часовой стрелке;

$B$  — размер малой полуоси;

$ALPHA$  — угол наклона полуоси  $A$  эллипса к оси  $x$  (в градусах);

$THS$ ,  $THF$  — углы наклона начального и конечного радиусов к полуоси  $A$ .

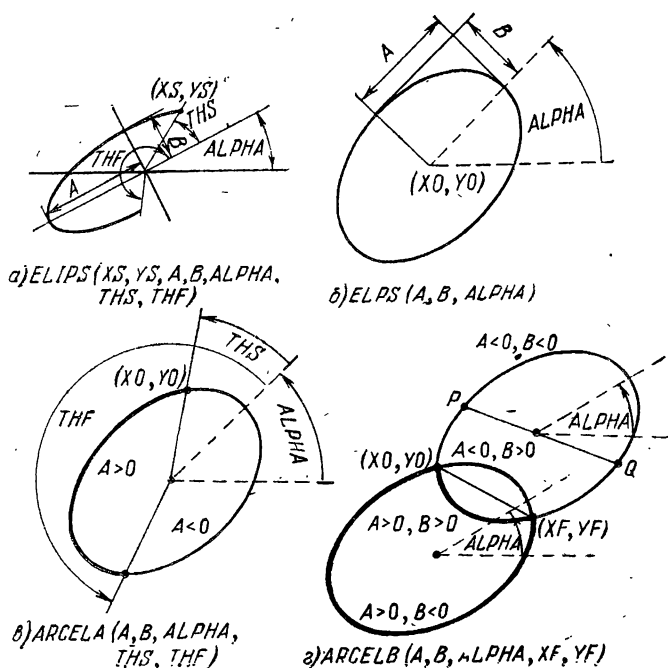


Рис. 1.11. Эллипсы и их дуги.

Обращение к программе считается некорректным, если  $|THS - THF| > 360^\circ$ .

Программа  $ARCELB(A, B, ALPHA, XF, YF)$  проводит из текущей точки в заданную точку дугу эллипса с указанными размерами полуосей и направлением рисования относительно центра (рис. 1.11, г). Ее параметры:

$|A|$ ,  $|B|$  — размеры полуосей:  $A > 0$  — перемещение пера против часовой стрелки,  $A < 0$  — перемещение пера по часовой стрелке,  $B > 0$  — выбирается малая дуга ( $\leq 180^\circ$ ),  $B < 0$  — выбирается большая дуга ( $\geq 180^\circ$ );

$ALPHA$  — угол наклона полуоси  $A$  эллипса к оси  $x$ ;

$X_F$ ,  $Y_F$  — координаты конечной точки дуги.

Если при обращении к программе ARCELB начальная точка совпадает с конечной, то независимо от знаков полуосей вычерчивается полный эллипс с углами наклонов (к полуоси  $A$ ) начального и конечного радиусов соответственно  $0^\circ$  и  $360^\circ$ . Если расстояние между конечной и начальной точками превышает величину диаметра эллипса  $PQ$  (рис. 1.11,  $z$ ), параллельного отрезку, соединяющему эти точки, то обращение к программе ARCELB считается некорректным.

#### 1.6.4. Размерные линии.

Программа NARROW( $XS$ ,  $YS$ ,  $X_F$ ,  $Y_F$ ,  $S$ , ICODE) чертит отрезок со стрелками на концах. Параметры программы:

$XS$ ,  $YS$  — координаты начальной точки отрезка;

$X_F$ ,  $Y_F$  — координаты конечной точки отрезка;

$S$  — длина стрелки;

ICODE — параметр, управляющий стрелкой: ICODE=1 — стрелка обращена к начальной точке, ICODE=2 — стрелка обращена к конечной точке, ICODE=3 — стрелки указывают на оба конца.

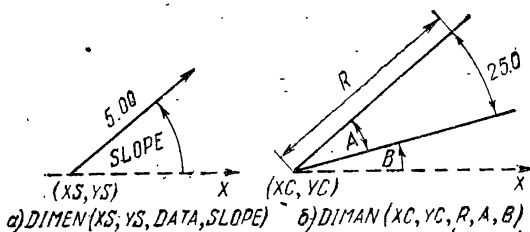


Рис. 1.12. Размерные линии.

Программа DIMEN( $XS$ ,  $YS$ , DATA, SLOPE) позволяет изобразить размерную линию заданной длины (рис. 1.12, а). Параметры программы:

$XS$ ,  $YS$  — координаты начальной точки;

DATA — длина отрезка (размерное число);

SLOPE — угол наклона отрезка к оси  $x$ .

Программа DARC( $XC$ ,  $YC$ ,  $XS$ ,  $YS$ ,  $A$ , ICODE) позволяет начертить дугу со стрелками на концах. Параметры программы:

$XC$ ,  $YC$  — координаты центра;

$XS$ ,  $YS$  — координаты начальной точки дуги;

$|A|$  — угловая величина дуги:  $A > 0$  — дуга вычерчивается против часовой стрелки,  $A < 0$  — дуга вычерчивается по часовой стрелке;

ICODE — параметр, управляющий стрелкой: ICODE=1 — стрелка обращена к начальной точке, ICODE=2 — стрелка обра-

щена к конечной точке,  $ICODE=3$  — стрелки указывают на оба конца.

Программа DIMAN ( $X_C$ ,  $Y_C$ ,  $R$ ,  $A$ ,  $B$ ) рисует размерную дугу заданного радиуса. Размерное число надписывается с точностью до одного десятичного знака (рис. 1.12, б). Параметры программы:

$X_C$ ,  $Y_C$  — координаты центра;

$R$  — радиус дуги;

$A$  — угловая величина дуги;

$B$  — угол между осью  $x$  и ближайшей к ней стороной угла.



## 2. ЛИНЕЙНЫЕ ПРЕОБРАЗОВАНИЯ, СЛЕД ПЕРА, ЭКРАНИРОВАНИЕ, ШТРИХОВКА

В главе 1 мы рассматривали построение графических объектов, которые чаще бывают элементами изображений, чем законченными изображениями. Средства системы Графтор, рассматриваемые в этой главе, достаточно разнородны. Их объединяет то, что с их помощью упрощается компоновка сложных многоэлементных изображений, хотя они могут оказаться полезными и в некоторых других случаях.

На практике мы постоянно сталкиваемся с изображениями, которые содержат много компонент (подрисунков), отличающихся друг от друга только местоположением, ориентацией, масштабом, т. е. обладающих некоторым геометрическим сходством. В таких случаях бывает выгодно иметь одно описание подрисунка, т. е. одну программу, рисующую эталонную компоненту, и получать из нее все остальные путем соответствующего преобразования плоскости рисунка. Чаще других для этой цели используется аппарат *линейных преобразований*, хорошо разработанный математически и легко реализуемый программно.

Однако в некоторых случаях желаемое преобразование изображения не является линейным. Поэтому в системе Графтор предусмотрены средства для запоминания информации о траектории пера в виде так называемого *следа пера*. След пера представляет собой последовательность координат точек, принадлежащих траектории пера. Занесенные в память координаты можно переработать и получить тем самым произвольное преобразование изображения. След пера может оказаться полезным еще и для того, чтобы составить границы участков, которые должны быть заэкранированы или заштрихованы, а также для того, чтобы сохранить в памяти картинку и рисовать ее в дальнейшем, не повторяя вычислений.

Еще один предмет, рассматриваемый в этой главе — *экранирование*, которое можно считать особым видом нелинейного преобразования. При задании одного или нескольких экранов — участков с кусочно-линейной границей — части генерируемого изображения, покрытые экраном, не рисуются.

Программы преобразования являются установочными, т. е. они лишь задают соответствующие преобразования, а выполняет их программа MOVE. Если задано линейное преобразование или указан режим регистрации следа пера, или, наконец, установлены экраны, то соответствующие действия будут применяться ко всем генерируемым в дальнейшем элементам изображения вплоть до отмены соответствующих установок.

Материал о *штриховке* отнесен к этой главе, поскольку границы заштрихованных участков задаются так же, как границы экранов.

## 2.1. Линейные преобразования

Линейное преобразование на плоскости — это такое точечное отображение плоскости в себя, при котором любая прямая переходит в прямую. Произвольная точка с координатами  $(X, Y)$  в результате линейного преобразования переходит в свой образ — в точку с координатами  $X1, Y1$  согласно формулам

$$X1 = A * X + B * Y + C, \quad Y1 = D * X + E * Y + F,$$

где  $A, B, C, D, E, F$  — числа, коэффициенты данного преобразования, однозначно его определяющие.

Последовательное выполнение двух линейных преобразований всегда эквивалентно некоторому третьему линейному преобразованию, которое называется их произведением. Это свойство позволяет говорить о *резльтирующем* преобразовании, эквивалентном некоторой последовательности преобразований.

Если перейти к *однородным координатам* точки (см., например, [15, 16]), то формулы линейного преобразования можно записать в матричном виде:

$$(X1, Y1, 1) = (X, Y, 1) \cdot \begin{bmatrix} A & D & 0 \\ B & E & 0 \\ C & F & 1 \end{bmatrix} = (X, Y, 1) \cdot M.$$

Тогда последовательное применение двух преобразований выглядит следующим образом:

$$(X2, Y2, 1) = (X1, Y1, 1) \cdot M2 = (X, Y, 1) \cdot M1 \cdot M2 = (X, Y, 1) \cdot M,$$

где  $M = M1 \cdot M2$  — матрица результирующего преобразования. В общем случае операция умножения матриц некоммутативна. А значит, и два последовательно выполняемых линейных преобразования также, вообще говоря, некоммутативны.

Если значение определителя матрицы  $M$  отлично от нуля, то преобразование называется *аффинным*. В отличие от общего линейного преобразования при аффинном преобразовании плоскость

не может вырождаться в линию или точку. Аффинное преобразование переводит параллельные прямые в параллельные и всегда имеет обратное преобразование. В подавляющем большинстве случаев на практике мы имеем дело именно с аффинными преобразованиями. Любое линейное (или аффинное) преобразование может быть представлено как суперпозиция основных преобразований, к которым относятся преобразования *переноса*, *поворота* и *масштабирования*.

**2.1.1. Программы установки линейных преобразований.** Можно считать, что текущее преобразование задается матрицей коэффициентов. Вначале, пока преобразования не задавались, текущее преобразование является тождественным, а матрица единичной. То есть, коэффициенты  $A = E = 1$ , а  $B = C = D = F = 0$ . Тогда любая операция установки преобразования сводится к умножению матрицы текущего преобразования на матрицу заданного преобразования. Таким образом, определяется результирующее преобразование. При выводе рисунка координаты всех точек умножаются на матрицу результирующего преобразования.

Для установки линейных преобразований в Графоре предусмотрены пять программ, с помощью которых можно задать любое из названных выше основных преобразований, а также определить произвольное линейное преобразование, указав его коэффициенты.

Программа TRANSL (DX, DY) задает параллельный перенос с вектором переноса (DX, DY).

Программа ROTATE (X, Y, PSI) задает поворот вокруг центра с координатами (X, Y) на угол PSI, измеряемый в градусах.

Программа FSCALE (X, Y, R) задает растяжение в  $|R|$  раз относительно центра (X, Y) и, кроме того, если  $R < 0$ , задает центральную симметрию.

Программа LSCALE (X1, Y1, X2, Y2, R) задает растяжение в  $|R|$  раз относительно оси, проходящей через точки (X1, Y1) и (X2, Y2), и, кроме того, если  $R < 0$ , задает осевую симметрию. Если точки (X1, Y1) и (X2, Y2) совпадают, то осью преобразования считается прямая, параллельная оси X.

**Примечание.** Очевидно, что при  $R = -1$  две последние программы будут задавать преобразования симметрии без растяжений, а при  $|R| < 1$  получим преобразование сжатия в  $1/|R|$  раз.

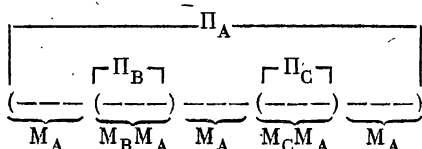
Программа ATRAN2 (A, B, C, D, E, F) задает произвольное линейное преобразование с коэффициентами A, B, C, D, E, F (см. приведенные выше формулы). Для умножения матриц используется служебная программа MTMPL.

Программа RESET отменяет накопленное результирующее преобразование, делая матрицу преобразования единичной. Эта

программа параметров не имеет. В программе RESET имеется внутренний вход ATRST, выполняющий начальные установки в общие блоки аффинных преобразований.

**2.1.2. Уровни вложенности линейных преобразований.** Во многих случаях удобно строить рисунок как совокупность подрисунков, состоящих в свою очередь из подподрисунков, и т. д. При этом подрисунок может оформляться как подпрограмма или фрагмент программы и описываться с использованием своих внутренних (локальных) преобразований. Поскольку подрисунок является частью некоторого объемлющего рисунка, на него должны также распространяться внешние (глобальные по отношению к нему) преобразования.

Рассмотрим простой пример. Рисунок А, кроме некоторых собственных графических элементов, включает подрисунки В и С. Тогда структура программы может быть такой:



Пусть в подпрограммах (или фрагментах)  $P_B$  и  $P_C$  устанавливаются локальные преобразования соответственно  $M_B$  и  $M_C$ , а программа  $P_A$  устанавливает свое (глобальное) преобразование  $M_A$ . Тогда элементы подрисунка В должны подвергаться преобразованию  $M_B M_A$ , элементы подрисунка С — преобразованию  $M_C M_A$ , а в программе  $P_A$  повсюду (за исключением фрагментов  $P_B$  и  $P_C$ ) должно действовать только преобразование  $M_A$ .

В Графоре начало каждого подрисунка отмечается вызовом подпрограммы BEGLEV, которая открывает новый уровень вложенности и сохраняет матрицу текущего преобразования глобальную по отношению к этому подрисунку. Подрисунок заканчивается вызовом подпрограммы ENDLEV, которая отменяет все локальные преобразования и восстанавливает матрицу преобразования предыдущего (более высокого) уровня. В Графоре допускается 16 уровней вложенности преобразований. При попытке открыть уровень сверх этого числа на печатающее устройство выдается диагностический текст СЛИШКОМ МНОГО СКОБОК, страница принудительно закрывается и вдоль линии разреза страниц пишется GFFALS — название программы Графора, выполняющей печать диагностик. Аналогичные действия производятся, если при обращении к программе ENDLEV оказывается, что не было соответствующего обращения к BEGLEV (нарушено соответствие скобок). Только диагностический текст будет другим: НЕВЕРНОЕ ЧИСЛО СКОБОК.

Заметим, что подпрограмма RESET действует только на текущем уровне и отменяет только локальные преобразования. После аннулирования программой RESET накопленного преобразования на текущем уровне может накапливаться новое, и этот процесс может повторяться любое число раз. Закрытие страницы (обращение к программе ENDPG) закрывает все еще не закрытые уровни

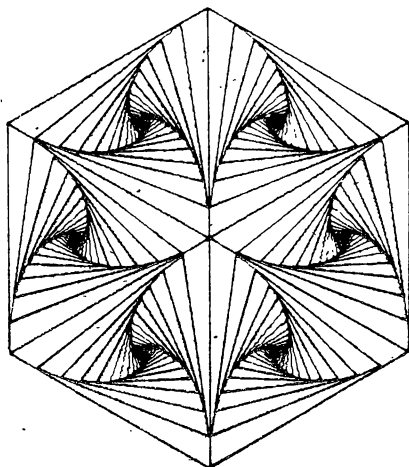


Рис. 2.1. Пример использования программ аффинных преобразований.

подрисунков и, стало быть, аннулирует все накопленные преобразования.

Примером использования вложенных преобразований может служить рис. 2.1. Он построен с помощью следующей программы:

```
//MISH JOB
// EXEC GRAFGCLG
//FORT.SYSIN DD *
  CALL GRINIT
  CALL BANTIK
  CALL GRFIN
END
SUBROUTINE BANTIK
  R=3.6
  XC=6.
  YC=6.
  YL=-R/2
  XL=YL/SQRT(3.)
  CALL PAGE(12.,12.,0,0,1)
  CALL TRANSL(XC, YC)
```

```

CALL BEGLEV
DO 1 I=1, 3
CALL BEGLEV
DO 2 J=1, 2
CALL BEGLEV
DO 3 K=1, 20
CALL MOVE(0., 0., 0)
CALL MOVE(0., -R, 1)
CALL MOVE(-SQRT(3.)/2*R, -R/2, 1)
CALL MOVE(0., 0., 1)
CALL PSCALE(XL, YL, 8)
CALL ROTATE(XL, YL, -8.)
3 CONTINUE
CALL ENDLEV
CALL LSCALE(0., 0., 0., 1., -1.)
2 CONTINUE
CALL ENDLEV
CALL ROTATE(0., 0., 120.)
1 CONTINUE
CALL ENDLEV
CALL ENDPG('2, 1')
RETURN
END
//GOSYSGRF DD UNIT=007, DCB=BLKSIZE=80
//

```

На самом нижнем уровне вложенности (в программе это цикл DO 3) строится фигура, образованная 20 треугольниками. В цикле каждый следующий треугольник уменьшается (коэффициент масштабирования 0.8) и поворачивается (на угол  $8^\circ$ ) по отношению к предыдущему.

На следующем уровне (цикл DO 2) строится зеркальное отражение фигуры относительно одной из сторон исходного треугольника.

И, наконец, на самом верхнем уровне (цикл DO 1) фигура и ее зеркальное отражение трижды повторяются с поворотом на  $120^\circ$ .

## 2.2. След пера

Формирование следа пера сводится к тому, что с некоторой дискретностью отбираются точки, принадлежащие траектории пера, и координаты этих точек запоминаются. Отобранные точки мы будем называть *зарубками*, а массивы в памяти, где запоминаются их координаты, — *банком зарубок*. Введение последнего

термина позволяет различать информацию (сам след) и место его хранения (банк).

Довольно легко перечислить параметры для управления процессом накопления следа, которые было бы желательно предоставить пользователю. Прежде всего, нужно уметь управлять выбором точек траектории так, чтобы при относительно небольшом числе точек траектория была задана достаточно хорошо. Далее очевидно, что интерес представляет не только *результатирующая траектория* пера, т. е. та, которая получится после применения всех преобразований (линейных, экранирований, отсечений по странице), но и *предварительная траектория*, т. е. та, по которой перо двигалось бы, если бы никаких преобразований не выполнялось.

Без рассмотрения предварительной траектории не обойтись, если мы хотим, например, построить подрисунок, используя след пера, а затем подвергнуть ее аффинному преобразованию. В случае экранирования элементы, попавшие на заэкранированные участки, вообще не будут присутствовать в результирующей траектории, тогда как предварительная будет их содержать.

Далее ясно, что нам нужно уметь отключать рисование во время формирования следа, поскольку возможно, что рисовать мы будем после преобразования занесенных в банк координат.

И, наконец, определенные удобства предоставляла бы возможность работать одновременно с несколькими банками зарубок, индивидуально управляя накоплением следов в каждом из них.

Средства работы со следом пера представлены в Графоре в основном двумя программами: NOTCH — открытие банка зарубок и включение режима формирования следа и RENTCH — закрытие банка и отмена режима формирования следа.

Подробнее об этих программах будет сказано ниже.

2.2.1. Два метода выбора зарубок на траектории. Траектория пера представляет собой ломаную линию, каждое из прямолинейных звеньев которой соответствует перемещению пера, осуществляемому за одно обращение к программе MOVE. При этом, естественно, одни звенья проходятся с опущенным, а другие — с поднятым пером.

В Графоре реализовано два метода отбора зарубок вдоль ломаной, причем в каждом из них задается шаг  $S$ . При решении вопроса о включении текущей точки траектории в число зарубок рассматривается длина  $L$  участка траектории от предыдущей зарубки до текущей точки по отношению к шагу  $S$ .

В качестве первой зарубки всегда берется начальная точка траектории, т. е. точка текущего положения пера в момент соответствующего вызова программы NOTCH. По первому методу в качестве зарубок берутся только вершины ломаной. Если для оче-

редной вершины выполняется соотношение  $L < S$ , то вершина не становится зарубкой и рассматривается следующая вершина. Если же  $L \geq S$ , то вершина становится зарубкой. Таким образом, когда все отрезки траектории больше шага  $S$ , в качестве зарубок берутся все вершины и только они. Если же ломаная содержит короткие отрезки, часть вершин не попадет в число зарубок. На рис. 2.2 иллюстрируется описанный метод отбора зарубок. При заданном

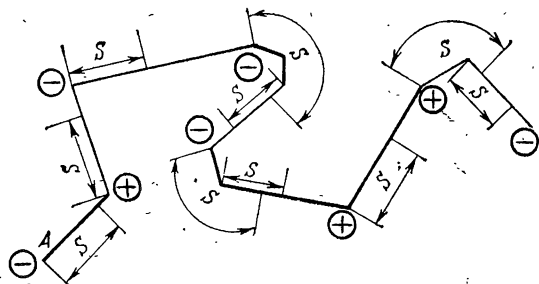


Рис. 2.2. Первый метод отбора зарубок. Обозначения: темная линия — перо опущено, светлая линия — перо поднято,  $\oplus$  — зарубка с положительным  $X$ ,  $\ominus$  — зарубка с отрицательным  $X$ ,  $A$  — начальная точка траекторий (в нее перо пришло поднятым);  $S$  — шаг.

на рисунке шаге  $S$  в число зарубок попадут вершины, отмеченные кружочками, при этом внутри кружочков будут стоять знак  $+$ , если координаты положительны (см. ниже), и знак  $-$ , если отрицательны.

Рассмотренный метод выбора зарубок удобен, например, когда требуется записать рисунок с точностью до шага  $S$ , а впоследствии воспроизвести его без преобразований. Им можно также воспользоваться, если след необходимо подвергнуть линейным преобразованиям (прямая при этом переходит в прямую). Для траекторий с относительно большой длиной звеньев получается достаточно экономная запись следа. Если же след формируется для выполнения впоследствии нелинейных преобразований рисунка, а средняя длина звеньев заметно превышает  $S$ , то такой метод отбора зарубок становится непригодным.

Второй метод ничем не отличается от первого для тех участков траектории, которые проходятся с поднятым пером. Когда же перо опущено, то зарубки расставляются на траектории равномерно с шагом  $S$ , невзирая на концы отрезков ломаной (рис. 2.3).

Информация о том, какие зарубки на траектории проходились с опущенным пером, а какие с поднятым, запоминается в виде знака координаты каждой зарубки. Мы знаем, что результирующие страничные координаты всегда положительны, и, следовательно, для результирующего следа пера проблем не возникает.



Что же касается предварительных страничных координат, то они, в принципе, хотя и очень редко, могут принимать отрицательные значения. Поэтому, когда формируется предварительный след пера, значения всех отрицательных координат  $X$  заменяются малым по абсолютной величине числом, не равным 0, и знак этого числа

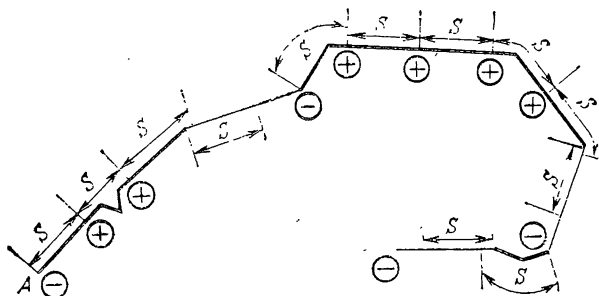


Рис. 2.3. Второй метод отбора зарубок. Обозначения см. на рис. 2.2.

используется для указания состояния пера. Фактически это соответствует отсечению по левому краю страницы в предварительных координатах.

Итак, координата  $X$  положительна, если перо пришло в зарубку опущенным, и отрицательна в противном случае. То есть согласно этому правилу, знак координаты  $X$  в первой зарубке определяется состоянием пера при вычерчивании звена, которое предшествует траектории.

**2.2.2. Банки зарубок.** Для того чтобы открыть банк зарубок, следует обратиться к программе NOTCH. Эта программа запоминает адреса переданных ей массивов  $X$ ,  $Y$ , их длину и адрес переменной  $N$ , которые будут использоваться при накоплении зарубок. Программа NOTCH устанавливает значение  $N$  равным 1.

Программа NOTCH ( $X$ ,  $Y$ ,  $NMAX$ ,  $N$ ,  $S$ ,  $J$ ,  $ITR$ ) открывает банк зарубок и включает заданный режим формирования следа пера. Параметры программы следующие:

$X$ ,  $Y$  — массивы для накопления координат зарубок;

$NMAX$  — размер каждого из этих массивов;

$N$  — переменная для хранения текущего индекса заполнения банка;

$S$  — величина шага следа пера ( $|S|$ ) в выбранных единицах измерения и метод его формирования:  $S > 0$  — первый метод,  $S < 0$  — второй метод;

$J$  — признак рисования:  $J = 0$  — рисование отключено,  $J = 1$  — рисование включено;

$ITR$  — тип траектории:  $ITR = 0$  — предварительная,  $ITR = 1$  — результирующая.

Важно помнить, что для формирования следа пера необходимо не только установить соответствующий режим, но и правильно выбрать вариант программы MOVE (MOVE3, если след пера используется в совокупности с экранированием, и MOVE2, если без экранирования). В функциональную часть своего пакета пользователь должен вставить подпрограмму:

```
SUBROUTINE MOVE (X, Y, J)  или SUBROUTINE MOVE (X, Y, J)
CALL MOVE2 (X, Y, J)      CALL MOVE3 (X, Y, J)
RETURN                    RETURN
END                        END
```

В программе MOVE2 (MOVE3) есть обращение к служебной программе NOTCH1, которая осуществляет накопление следа пера, т. е. формирует массивы координат точек с признаком перемещения пера в поднятом или опущенном состоянии.

В каждый момент времени могут существовать до 16 открытых банков зарубок. При этом все параметры, кроме параметра J, действуют лишь на свой банк зарубок и не зависят от значений аналогичных параметров в других банках. Параметр J управляет рисованием одной-единственной результирующей траектории, поэтому, для того чтобы отключить рисование, достаточно в одном банке задать J=0.

Закрыть банк можно программой RENTCH. Программа не имеет параметров и закрывает тот из существующих банков, который был открыт последним. Можно продолжить заполнение уже закрытого банка, если предварительно сохранить значение параметра N, а сразу после повторного открытия банка восстановить это значение.

При попытке открыть банк сверх 16 уже существующих на печатающее устройство выдается диагностический текст: ЧИСЛО БАНКОВ БОЛЬШЕ ДОПУСТИМОГО, а сама попытка игнорируется.

Если в каком-то банке емкость выделенных массивов для записи зарубок исчерпана, то банк закрывается и печатается диагностический текст: БАНК НОМЕР I ПЕРЕПОЛНЕН. (Считается, что перенумерованы только существующие в данный момент банки в том порядке, как они открывались.)

При закрытии страницы (при обращении к программе ENDPG) закрываются все существующие банки зарубок, так что режим формирования следа пера, установленный для одной страницы, на другую страницу не распространяется.

Для получения адресов переменных в программе NOTCH используется служебная функция IADR(A). Она написана на автокоде.

В качестве примера использования следа пера рассмотрим рис. 2.4. Изображение построено из однотипных элементов, полу-

ченых аффинными преобразованиями одного и того же эталонного элемента. Эталонный элемент представляет собой пучок касательных к дуге эллипса и строится с помощью приведенной ниже программы UNIT. Основная идея состоит в том, что при малом шаге накопления следа прямая, проходящая через соседние зарубки, достаточно хорошо приближает касательную.



Рис. 2.4. Пример, иллюстрирующий действие программ формирования следа пера.

Заметим, что след пера формируется в страничных координатах. Кроме того, при использовании следа пера для задания границ областей штриховки экранов следует учитывать, что координатам точек, в которые перо приходит в поднятом состоянии, приписывается знак —.

```
SUBROUTINE UNIT
DIMENSION X1(2),Y1(2),X(400),Y(400)
CALL MOVE(220.,176.,0)
CALL NOTCH(X,Y,400,N,3.,0,0)
```

```

C... N ~ ЧИСЛО ЗАРУВОВ, ТОЧЕК НА ЭЛЛИПСЕ,
C... УДАЛЕННЫХ ОДНА ОТ ДРУГОЙ НА 3ММ
CALL ELIPS(220., 176., 54., 99., 270., 140., 273.)
CALL RENTCH
XN=212./FLOAT(N-1)
N=N-2
DO 1 J=1,N
X(J)=ABS(X(J))
1 Y(J)=ABS(Y(J))
T=212.+3.*XN
DO 2 J=1,N
X1(1)=X(J)
Y1(1)=Y(J)
S=SQRT((X(J+1)-X(J))**2+(Y(J+1)-Y(J))**2)
T=T-XN
X1(2)=X(J)+T*(X(J+1)-X(J))/S
Y1(2)=Y(J)+T*(Y(J+1)-Y(J))/S
2 CALL LINEO(X1, Y1, 2)
RETURN
END
SUBROUTINE MOVE(X, Y, J)
CALL MOVE2(X, Y, J)
RETURN
END

```

Еще один пример применения следа пера, демонстрирующий возможность выполнения произвольных (в том числе нелинейных) преобразований рисунка, будет рассмотрен в § 2.5.

## 2.3. Экранирование

Экранирование можно считать особым видом преобразования изображения, заключающемся в «стирании» части рисунка, принадлежащей некоторому участку — экрану. Мы рассчитываем на то, что употребляемое тут слово «экран» читатель не спутает с экраном дисплея.

Экран представляет собой область с кусочно-линейной границей. Эта область может быть односвязной, несвязной или многосвязной, как показано на рис. 2.5; во всех случаях она сводится к односвязной с помощью фиктивных соединений и разрезов. Задать экран — значит, задать координаты вершин его границы, обходя всю границу от точки к соседней точке. Начальную точку и направление обхода можно выбирать произвольно. При этом последняя вершина может не совпадать с первой, поскольку всегда строится дополнительное ребро, соединяющее «последнюю» и

первую вершины. Некоторые из возможных порядков обхода показаны на рис. 2.5.

Включение режима экранирования и задания экрана выполняется программой BLANC(X, Y, N, IN) с параметрами:

X, Y — массивы координат вершин границы экрана;

N — количество вершин границы;

IN — признак «внутри — вне»:  $IN=1$  — экранируется область внутри границы,  $IN=0$  — экранируется область вне границы.

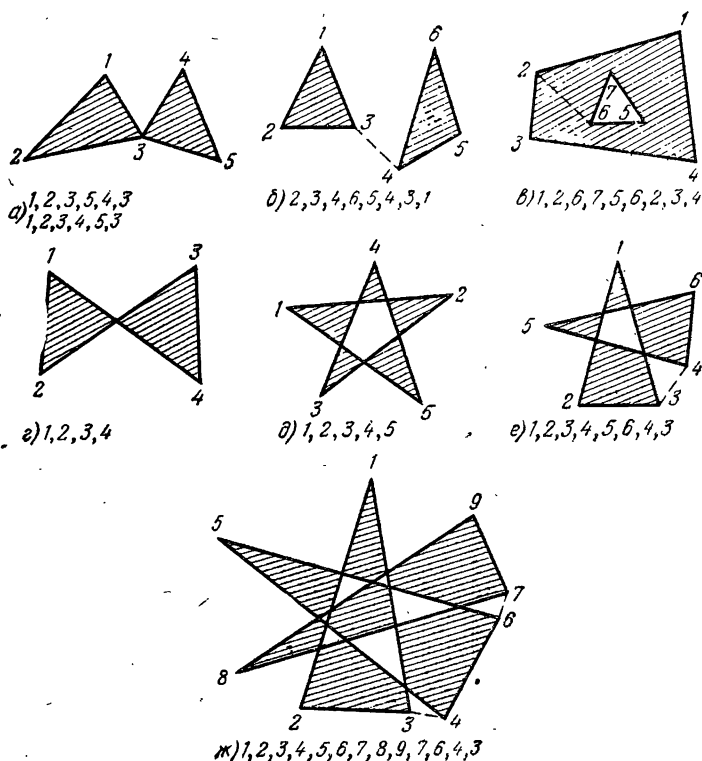


Рис. 2.5. Задание границ экранов. Штриховкой показан экранируемый участок при  $IN=1$ .

Кроме обращения к программе BLANC, необходимо выбрать соответствующий вариант программы MOVE (MOVE1 или MOVE3). Если в программе пользователя есть обращения к программам формирования следа пера и программам экранирования, необходимо использовать программу MOVE3. Если же программы экранирования применяются сами по себе, то должна быть выбрана про-

грамма MOVE1. В функциональную часть своего пакета пользователю следует вставить подпрограмму:

SUBROUTINE MOVE(X, Y, J) или	SUBROUTINE MOVE(X, Y, J)
CALL MOVE1(X, Y, J)	CALL MOVE3(X, Y, J)
RETURN	RETURN
END	END

В программе MOVE1 (MOVE3) есть обращение к служебной программе BLANCH, выполняющей экранирование.

Программа BLANCH выделяет и отсекает экранируемые части заданного отрезка. Программа без параметров. В этой программе имеется дополнительный вход BLAN.

Если к моменту обращения к программе BLANC на странице действует некоторое линейное преобразование, то этому преобразованию подвергается и граница экрана. Задание границы не предусматривает ее очерчивания. Если такая необходимость возникает, то об этом следует специально позаботиться, например, воспользовавшись программами LINEO или LINEC.

При работе с экранированием реальная траектория пера существенно отличается от соответствующей траектории при отсутствии экранов. Перо, попадая на границу экранируемого участка, останавливается и остается там до тех пор, пока траектория снова не выйдет за границу экрана. Тогда перо перемещается в поднятом состоянии в соответствующую точку границы и рисование продолжается. Это следует иметь в виду, например, если формируется след пера.

До сих пор речь шла об одном экране, устанавливаемом одним обращением к программе BLANC. Однако можно несколько раз обратиться к программе BLANC, и тогда на изображение будет действовать результирующий экран, который представляет собой объединение отдельных экранов. Любой из установленных экранов можно отменить, если он больше не нужен. При этом будет соответствующим образом скорректирован и результирующий экран.

Отмена экранов осуществляется программой REBLAN. Она не имеет параметров. Эта программа отменяет экран, установленный последним. При закрытии страницы по программе ENDPG отменяются все экраны.

Максимальное число одновременно установленных экранов не может превышать 16. Если пользователь обращается к программе BLANC, когда уже действуют 16 экранов, то отменяется самый старый из экранов и устанавливается новый. При этом печатается диагностический текст: ЧИСЛО ЭКРАНОВ БОЛЬШЕ ДОПУСТИМОГО.

На рис. 2.6 объектом изображения является ломаная линия, которая от раза к разу меняет координаты вершин и смещается на

фиксированную величину по вертикали и по некоторому закону по горизонтали. Каждый раз после изображения ломаной устанавливается новый экран, в качестве точек границы которого указываются вершины этой ломаной. Таким образом, в каждый данный

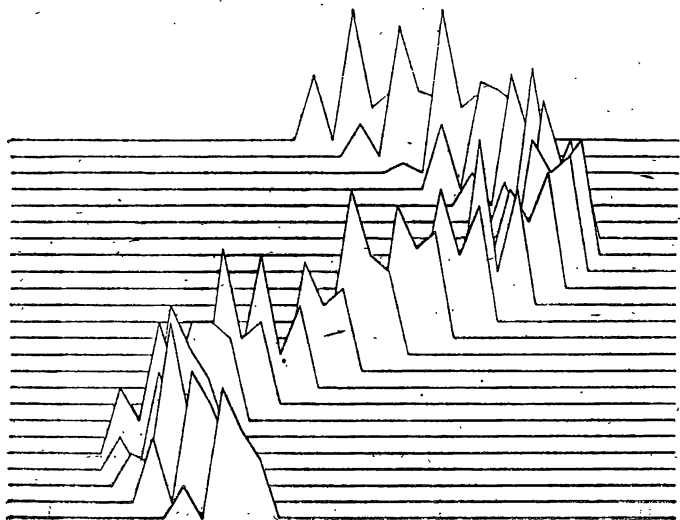


Рис. 2.6. Пример, иллюстрирующий объединение экранов.

момент на странице действует результирующий экран, представляющий собой объединение ранее установленных экранов, ограниченных уже начерченными ломаными. В данном примере ограничение числа установленных экранов не играет роли: соотношение высоты «гор» и шага по вертикали таково, что автоматически отменяемый экран каждый раз оказывается вне области рисования очередной ломаной.

```

DIMENSION Y(9), XX(288), YY(288)
DATA Y/2., 2., 3., 2., 6., 4., 4.5, 2., 2./
CALL PAGE(20., 26., 0, 0, 0)
CALL LSCALE(0., 26., 1., 26., 0.85)
DO 1 J=1,24
  XX(1+(J-1)*9)=0.
  YY(1+(J-1)*9)=0.5*(J-1)+2.
  XX(2+(J-1)*9)=10.-5.*SIN(J-1)*3.14/13.)
  YY(2+(J-1)*9)=0.5*(J-1)+2.
  XX(8+(J-1)*9)=10.-5.*SIN((J-1)*3.14/13.)+3.
  YY(8+(J-1)*9)=0.5*(J-1)+2.
  XX(9+(J-1)*9)=20.

```

```

YY(9+(J-1)*9)=0.5*(J-1)+2.
DO 2 I=3,7
XX(I+(J-1)*9)=10.-5.*SIN((J-1)*3.14/13.)+0.5*(I-2)
2 YY(I+(J-1)*9)=AMAX1(Y(I)+0.5*(J-1)
, +SIN((I-1)*(J-1)), 0.5*(J-1)+2.)
CALL LINEO(XX(1+(J-1)*9), YY(1+(J-1)*9), 9)
CALL BLANC(XX(1+(J-1)*9), YY(1+(J-1)*9), 9, 1)
1 CONTINUE
CALL ENDPG('2.6')
END.
SUBROUTINE MOVE(X, Y, J)
CALL MOVE1(X, Y, J)
RETURN
END

```

## 2.4. Штриховка

Штриховка заданных полей является неотъемлемой частью подготовки различного вида чертежей и рисунков. Для выполнения штриховки служат две программы: SDPG и SHADE. Первая из них позволяет заштриховать все поле страницы. Совместное ее использование с программами экранирования дает возможность заштриховать требуемые участки на странице. Вторая программа штрихует требуемые участки, но для нее необходимо задавать границы участков. Оба способа выполнения штриховки совершенно независимы — программы SDPG и SHADE не используют друг друга.

Программа SDPG(STEP, EPS, BETA) позволяет заштриховать все поле страницы. Параметры программы:

STEP — расстояние между линиями штриховки в выбранных единицах измерения;

EPS — сдвиг линий штриховки относительно основной штриховки по нормали к ней;

BETA — угол наклона линий штриховки к оси  $x$  (в градусах).

Основной штриховкой в данном случае считается та, для которой одна из линий проходит через точку  $(0, 0)$ , т. е. через левый нижний угол страницы.

Программа SHADE(X, Y, N, STEP, EPS, BETA) позволяет заштриховать участок страницы, ограниченный замкнутой кусочно-линейной ориентированной кривой. Параметры программы:

X, Y — массивы длины N абсцисс и ординат соответственно задающие вершины границы;

N — число вершин границы;

STEP — расстояние между линиями штриховки в выбранных единицах измерения;



EPS — сдвиг линий штриховки относительно основной штриховки по нормали к ней;

BETA — угол наклона линий штриховки к оси  $x$  (в градусах).

Эта программа позволяет штриховать также и самопересекающиеся многоугольники. При задании многоугольника (т. е. массивов  $X$  и  $Y$ ) последняя вершина может не совпадать с первой, поскольку всегда строится дополнительное ребро, которое их соединяет. Значение  $N$  (число вершин) не должно превышать 250. Никакая линия штриховки не должна пересекать более 40 ребер.

В тех случаях, когда обращением к программам LIMITS и REGION (см. гл. 4) заданы математические координаты и на странице определена область, программа SHADE будет проводить штриховку заданного многоугольника внутри этой области, не выходя за ее границы.

## 2.5. Примеры

Рассмотрим еще два примера, иллюстрирующих описанные в этой главе средства.

1. Изображение на рис. 2.7 построено с помощью приведенной ниже программы ASTRA. В качестве основной компоненты

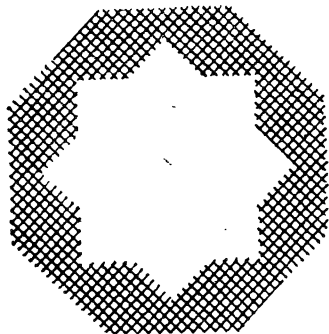


Рис. 2.7. Пример, иллюстрирующий применение программ аффинных преобразований, экранирования и штриховки.

взят квадрат, который повернут на  $45^\circ$ , затем растянут в 1.75 раза и снова повернут. Все повороты и растяжения делаются относительно центра квадратов — точки (5, 5). Все четыре квадрата используются как границы экранов для штриховки.

```
DIMENSION X(4), Y(4)
DATA X/3., 7., 7., 3./, Y/3., 3., 7., 7./
CALL PAGE(10., 10., 'ASTRA', 5, 0)
CALL BLANC(X, Y, 4, 1)
CALL ROTATE(5., 5., 45.)
CALL BLANC(X, Y, 4, 1)
CALL PSCALE(5., 5., 1.75)
```

```

CALL BLANC(X, Y, 4, 0)
CALL ROTATE(5., 5., 45.)
CALL BLANC(X, Y, 4, 0)
CALL RESET
CALL SDPG(0.1, 0., 45.)
CALL SDPG(0.1, 0., 135.)
CALL ENDPG('2.7')
END
SUBROUTINE MOVE(X, Y, J)
CALL MOVE2(X, Y, J)
RETURN
END

```

2. Обратимся к рис. 2.8. Справа внизу изображен «эталонный» крокодил с закрытой пастью, нарисованный программой CROCKY, которая здесь не приводится. Программа JAWS рисует крокодила

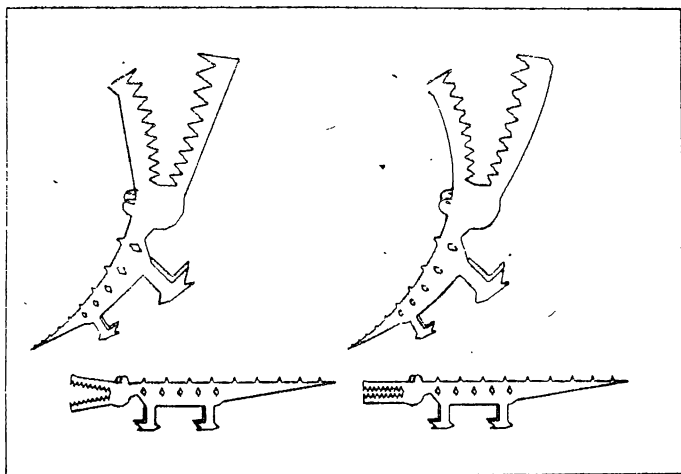


Рис. 2.8. Иллюстрация возможностей применения нелинейного преобразования к следу пера.

с раскрытой пастью слева внизу. В ней до обращения к программе CROCKY устанавливается формирование следа пера и при последующем анализе точек следа выделяются челюсти и поворачиваются на заданный угол.

```

SUBROUTINE JAWS(XJ, YJ, NJ, R, TH)
DIMENSION XJ(NJ), YJ(NJ)
CALL WHERE(DX, DY, F)
CALL NOTCH(XJ, YJ, NJ, 0.5, 0, 0)
CALL CROCKY(DX, DY, R)

```

```

CALL RENTCH
DO 8 J=1, NT
IF(ABS(XJ(J))-DX-26.*R.GT.0.) GOTO 4
IF(ABS(YJ(J))-DY-15.*R.GT.0.) GOTO 3
CALL ROTATE(DX+26.*R, DY+15.*R, TH)
GO TO 4
3 CALL ROTATE(DX+26.*R, DY+15.*R, -TH)
4 IF(XJ(J).GT.0.) GOTO 6
  CALL MOVE(-XJ(J), -YJ(J), 0)
  GO TO 7
6 CALL MOVE(XJ(J), YJ(J), 1)
7 CALL RESET
8 CONTINUE
  RETURN
  END

```

Оба крокодила наверху — результат применения нелинейного преобразования к крокодилу с раскрытой пастью. Это преобразование выполнялось над следом пера по формулам

$$x' = \frac{kx}{x^2 + y^2}, \quad y' = \frac{ky}{x^2 + y^2},$$

причем для левого верхнего крокодила зарубки отбирались по первому методу, а для правого — по второму. На длинных отрезках хорошо заметно отличие.

### 3. ГЕОМЕТРИЧЕСКИЕ ВЫЧИСЛЕНИЯ

В этой главе описаны программы, ориентированные на обработку геометрической информации. Геометрические вычисления широко применяются в машинном проектировании и при подготовке программ для станков с числовым программным управлением (ЧПУ). Графор имеет достаточно полный набор геометрических операций с простейшими геометрическими элементами.

Реализовано два комплекта программ, которые рассматриваются в двух параграфах, составляющих эту главу. В первом случае не предполагается каких-либо иных структур данных для хранения геометрических элементов, кроме массива — стандартной структуры фортрана. Во втором случае вводится понятие буфера, состоящего из двух позиций, и определяются операции с этим буфером.

#### 3.1. Геометрические операции

К простейшим геометрическим элементам (примитивам) мы относим точку, прямую и окружность. Точка задается координатами  $X$  и  $Y$ . Прямая задается коэффициентами  $A$ ,  $B$ ,  $C$  в уравнении  $Ax + By + C = 0$  (где  $A$ ,  $B$ ,  $C$  могут иметь любые значения, лишь бы коэффициенты  $A$ ,  $B$  не были нулями оба сразу). Окружность задается координатами центра  $(a, b)$  и радиусом  $R$  и удовлетворяет уравнению  $(x - a)^2 + (y - b)^2 = R^2$ .

Геометрические элементы описаны в программах как одномерные массивы: точка — массив из двух вещественных чисел, прямая и окружность — массив из трех вещественных чисел.

Существуют различные способы определения плоских геометрических элементов. Каждому из них соответствует подпрограмма. Описанные ниже программы, конечно, не охватывают всех возможных способов задания геометрических элементов. Реализованы наиболее часто встречающиеся в практике ситуации. Другие способы определения геометрических элементов могут быть реализованы с помощью комбинации нескольких имеющихся в наборе подпрограмм.

Имя подпрограммы однозначно указывает тип определяемого геометрического элемента (первая буква в имени программы) и

способ ее параметризации (остальные буквы в имени). Принята английская мнемоника. Например, имя подпрограммы PIRC расшифровывается так: определяются точки (P), полученные при пересечении (I) прямой линии (L) с окружностью (C). Имя STLLP означает, что определяются окружности (C), касающиеся (T) двух прямых линий (LL) и проходящие через заданную точку (P).

Как правило, начальные буквы в наименовании подпрограмм зависят от типов геометрических элементов: P — точка (point), L — прямая (line), C — окружность (circle), A — угловая величина (angle). Имеются отступления от этого правила. В пакете есть несколько программ, в которых определяется не вся окружность, а только ее центр. Первые три буквы в названии этих программ — CNC (CN — сокращение от centre, C — от circle).

Фактические параметры могут быть как входными, так и выходными. Они содержат информацию о форме и положении определяемого геометрического элемента относительно системы координат или относительно других геометрических элементов, определенных ранее. Алгоритмы решения задач не представляют большой сложности. При их реализации использовались методы аналитической геометрии.

Особенностями многих алгоритмов решения геометрических задач является необходимость отбирать нужное решение из нескольких вариантов, удовлетворяющих условию задачи. Например, из двух точек пересечения прямой с окружностью выбрать ту, которая имеет большую или меньшую ординату, либо выбрать точку, ближайшую или наиболее удаленную от некоторой прямой и т. д. В программах, где возможны несколько решений, в качестве выходного параметра выдается число возможных решений. А затем уже программы выбора находят из нескольких возможных вариантов, удовлетворяющих условию задачи, нужное решение.

В программах, реализующих конкретные геометрические операции, и в программах, выбирающих решения, используются общие блоки. Таких блоков два: GFGMP(4) и GFGMC(16). Геометрические элементы почти во всех программах записаны в блок в произвольном порядке. Только в программах CNCTCL и CNCTCC сначала записываются окружности, касающиеся снаружи окружности, стоящей первой в списке параметров программ, а затем окружности, касающиеся внутри.

Если пользователя не устраивают те критерии, по которым производится выбор решения, то он может написать свои программы выбора решения с использованием другого критерия. Для этого есть вся необходимая информация: известны количество решений, общий блок, куда записаны решения, представления геомет-

рических элементов (точка —  $X, Y$ , прямая —  $A, B, C$ , окружность —  $a, b, R$ ).

В случаях некорректного обращения происходит выход из данной программы. Если в программе должны быть присвоены какие-то значения некоторым переменным, то при некорректном обращении этого не происходит.

Для каждой из описанных далее программ на рис. 3.1 дается геометрическая иллюстрация. На рисунке для некоторых программ указаны значения признака выбора решения  $J$ .

### 3.1.1. Программы определения точки.

P1. Программа  $PXY(X, Y, P)$  определяет точку с координатами  $X$  и  $Y$ . Параметры программы:

$X, Y$  — координаты точки;

$P$  — полученная точка.

P2. Программа  $PRAP(P1, R, ALPHA, P2)$  задает точку в полярных координатах. Параметры программы:

$P1$  — полюс;

$R$  — полярный радиус;

$ALPHA$  — полярный угол (в градусах);

$P2$  — полученная точка.

P3. Программа  $PCNTRC(C, P)$  задает точку как центр окружности. Параметры программы:

$C$  — окружность;

$P$  — полученная точка.

P4. Программа  $PAC(C, ALPHA, P)$  находит точку на заданной окружности, если известен угол наклона к оси  $X$ , под которым находится точка. Параметры программы:

$C$  — окружность;

$ALPHA$  — угол (в градусах);

$P$  — полученная точка.

P5. Программа  $PCNAP(PC, P1, ALPHA, P)$  определяет на окружности с заданным центром точку, находящуюся на указанном угловом расстоянии от данной точки. Параметры программы:

$PC$  — центр окружности;

$P1$  — заданная точка на окружности;

$ALPHA$  — угол (в градусах);

$P$  — полученная точка.

Если  $ALPHA > 0$ , то точка лежит в направлении против часовой стрелки от заданной точки, если  $ALPHA < 0$  — по часовой стрелке.

P6. Программа  $PMIDPP(P1, P2, P)$  определяет точку, находящуюся посередине между двумя заданными точками. Параметры программы:

$P1, P2$  — заданные точки;

$P$  — полученная точка.

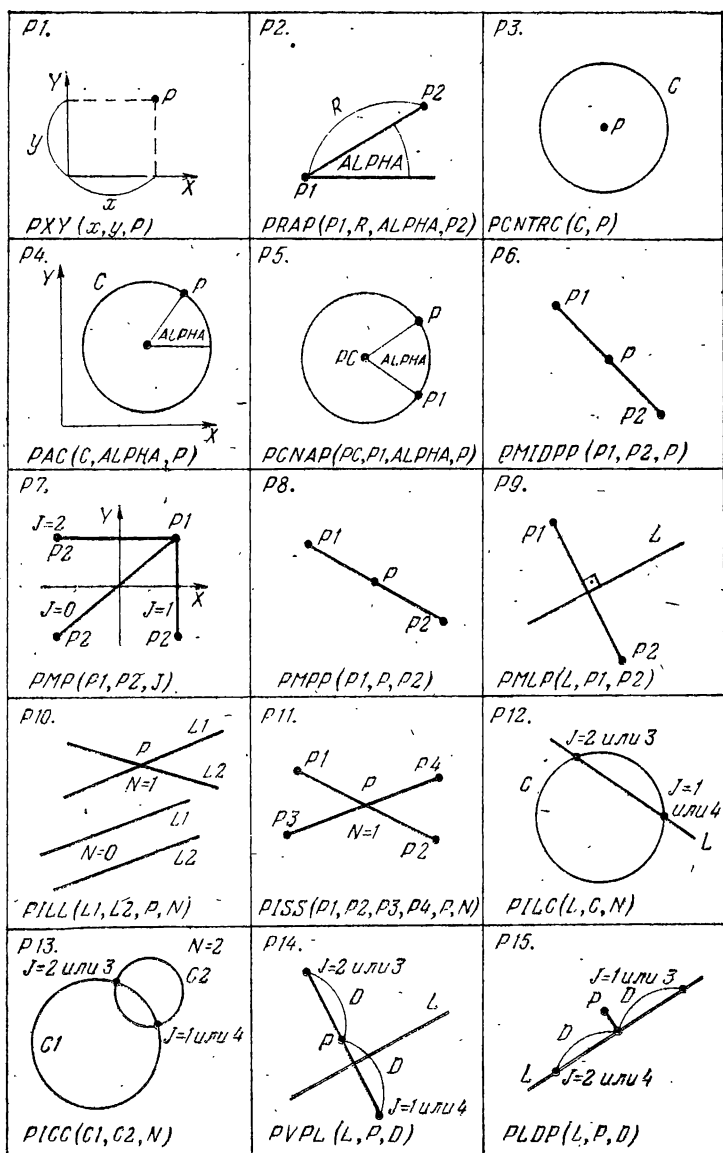


Рис. 3.1. Иллюстрации к программам геометрических вычислений.

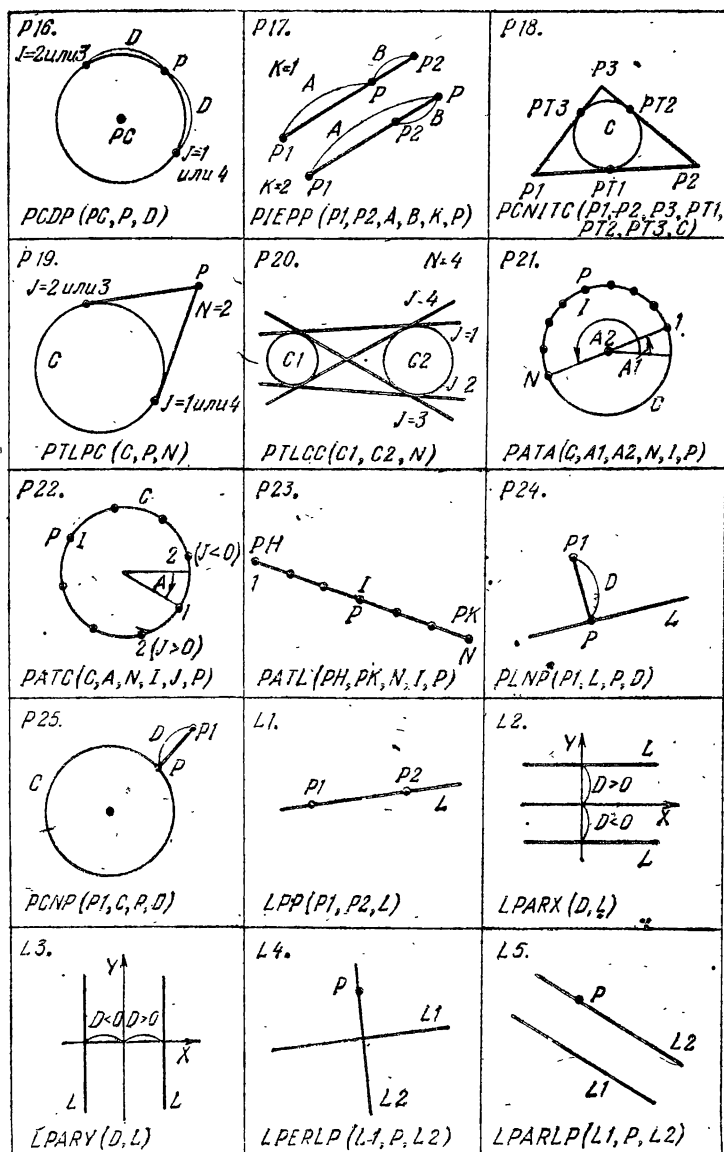


Рис. 3.1. Иллюстрации к программам геометрических вычислений.



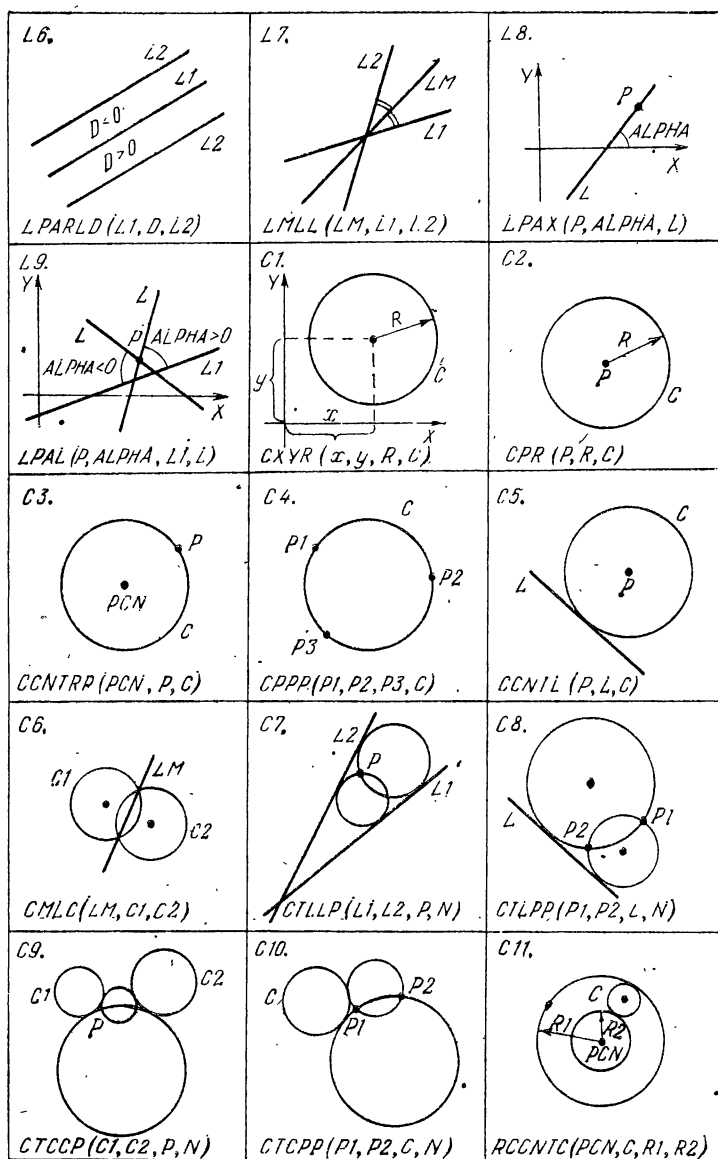


Рис. 3.1. Иллюстрации к программам геометрических вычислений

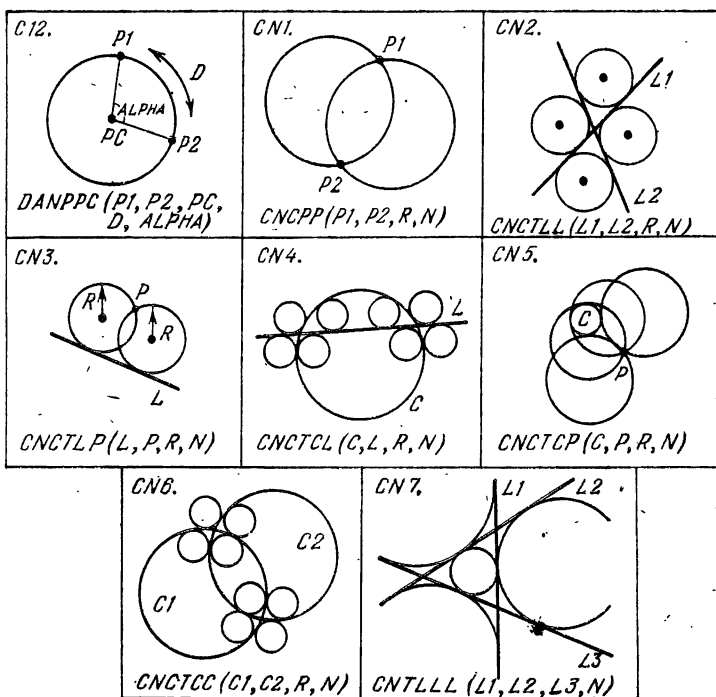


Рис. 3.1. Иллюстрации к программам геометрических вычислений.

P7. Программа RMP (P1, P2, J) находит точку, симметричную заданной относительно начала координат или осей. Параметры программы:

P1, P2 — заданная и полученная точки;

J — признак выбора симметричной точки: J = 0 — по отношению к началу координат, J = 1 — по отношению к оси X, J = 2 — по отношению к оси Y.

P8. Программа RMP (P1, P, P2) находит точку, симметричную заданной относительно другой известной точки. Параметры программы:

P1, P2 — заданная и полученная точки;

P — точка, относительно которой надо получить симметричную.

P9. Программа RMLP (L, P1, P2) определяет точку, симметричную заданной точке относительно известной прямой линии. Параметры программы:

L — заданная прямая;

P1, P2 — заданная и полученная точки.

P10. Программа PILL(L1, L2, P, N) определяет точку пересечения двух заданных прямых линий. Параметры программы:

L1, L2 — заданные прямые;

P — полученная точка;

N — параметр, характеризующий число точек:  $N = 0$  — нет точек пересечения (прямые параллельны),  $N = 1$  — одна точка пересечения.

P11. Программа PISS(P1, P2, P3, P4, P, N) определяет точку пересечения двух прямых, заданных отрезками. Параметры программы:

P1, P2 — концевые точки первого отрезка;

P3, P4 — концевые точки второго отрезка;

P — точка пересечения;

N — число точек (параметр N такой же, как в программе PILL).

P12. Программа PILC(L, C, N) находит точки пересечения заданных прямой линии и окружности. Параметры программы:

L, C — заданные прямая и окружность;

N — число точек пересечения:  $N = 0$  — нет общих точек,  $N = 1$  — прямая и окружность касаются,  $N = 2$  — прямая и окружность пересекаются.

Результат в общем блоке GFGMP. Решение выбирается с помощью программы SORTP или программы SPNP.

P13. Программа PICC(C1, C2, N) находит точки пересечения двух заданных окружностей. Параметры программы:

C1, C2 — заданные окружности;

N — число точек:  $N = 0$  — нет общих точек,  $N = 1$  — окружности касаются,  $N = 2$  — окружности пересекаются.

Результат в общем блоке GFGMP. Решение выбирается с помощью программы SORTP или программы SPNP.

P14. Программа PVPL(L, P, D) находит точки, расположенные на указанном расстоянии от заданной точки и лежащие на прямой, проходящей через эту точку перпендикулярно некоторой известной прямой. Параметры программы:

L, P — заданные прямая и точка;

D — расстояние.

Результат в общем блоке GFGMP. Решение выбирается программой SORTP или программой SPNP.

P15. Программа PLDP(L, P, D) находит точки на прямой линии, равноудаленные от заданной точки и расположенные на заданном расстоянии от основания перпендикуляра, опущенного из этой точки на прямую. Параметры программы:

L, P — заданные прямая и точка;

D — расстояние.

Результат в общем блоке GFGMP. Решение выбирается с помощью программы SORTP или программы SPNP.

P16. Программа PCDP(PC, P, D) находит точки на окружности с заданным центром на расстоянии D по дуге от известной точки на той же окружности. Параметры программы:

PC — центр окружности;

P — заданная точка;

D — длина дуги.

Результат в общем блоке GFGMP. Решение выбирается с помощью программы SORTP или программы SPNP.

Если заданная точка совпадает с центром окружности, то обращение к программе считается некорректным.

P17. Программа PIEPP(P1, P2, A, B, K, P) определяет точку, лежащую на той же прямой, что и две другие заданные точки, и отстоящую от них в заданном отношении A/B. Параметры программы:

P1, P2 — заданные точки;

A/B — заданное отношение;

K — признак расположения точки:  $K = 1$  — точка лежит между заданными точками,  $K = 2$  — точка лежит вне отрезка, образованного заданными точками;

P — искомая точка.

При  $A = B$ ,  $K = 2$  обращение к программе считается некорректным.

P18. Программа PCNITC(P1, P2, P3, PT1, PT2, PT3, C) определяет точки касания окружности, вписанной в треугольник с заданными вершинами, а также вписанную окружность. Параметры программы:

P1, P2, P3 — вершины треугольника;

PT1, PT2, PT3 — точки касания окружности со сторонами треугольника, проходящими соответственно через точки P1 и P2, P2 и P3, P1 и P3;

C — вписанная окружность.

Обращение к программе считается некорректным, если три точки лежат на одной прямой.

P19. Программа PTLPC(C, P, N) определяет точки касания окружности с прямой, проходящей через заданную точку. Параметры программы:

C, P — заданные окружность и точка;

N — число точек касания ( $0 \leq N \leq 2$ ).

Результат в общем блоке GFGMP. Решение выбирается с помощью программы SORTP или программы SPNP.

P20. Программа PTLCC(C1, C2, N) определяет точки касания прямой с двумя заданными окружностями. Параметры программы:

$C_1, C_2$  — заданные окружности;

$N$  — число возможных касательных прямых ( $N \leq 4$ ).

Результат в общем блоке CFGMC. Решение выбирается с помощью программы SORTPT. В блок CFGMC записываются сначала две пары точек для внешних касательных, затем две пары точек для внутренних касательных (для каждой пары точек первой в блок заносится точка касания с окружностью, стоящей первой в списке параметров, второй — со второй окружностью).

P21. Программа PATA( $C, A_1, A_2, N, I, P$ ) определяет точку с заданным номером ( $I$ ) на дуге окружности, разделенной  $N$  точками на равные части. Точки нумеруются от 1 до  $N$ . Параметры программы:

$C$  — заданная окружность;

$A_1, A_2$  — начало и конец дуги (в градусах);

$N$  — число точек на дуге;

$I$  — номер точки, которую нужно найти;

$P$  — найденная точка.

Обращение к программе считается некорректным, если  $N < I$ .

P22. Программа PATC( $C, A, N, I, J, P$ ) определяет точку с заданным номером на разделенной на  $N$  частей окружности. Параметры программы:

$C$  — окружность;

$A$  — угол к оси  $X$ , под которым находится на окружности первая точка (задан в градусах);

$N$  — число точек на окружности;

$I$  — номер точки, которую требуется найти;

$J$  — направление нумерации точек:  $J < 0$  — по часовой стрелке,  $J > 0$  — против часовой стрелки;

$P$  — полученная точка.

Обращение к программе считается некорректным, если  $N < I$ .

P23. Программа PATL( $PH, PK, N, I, P$ ) выбирает точку с заданным номером  $I$  на заданном отрезке, разделенном  $N$  точками на равные части. Точки нумеруются от 1 до  $N$ . Параметры программы:

$PH, PK$  — начальная и конечная точки отрезка;

$N$  — число точек на отрезке;

$I$  — номер точки, которую надо найти;

$P$  — выбранная точка.

Точки нумеруются от начальной к конечной.

Обращение к программе считается некорректным, если  $N < I$ .

P24. Программа PLNP( $P_1, L, P, D$ ) находит точку на заданной прямой, ближайшую к некоторой известной точке, и определяет расстояние между ними. Параметры программы:

$P_1$  — заданная точка;

$L$  — заданная прямая;

- Р — искомая точка;
- D — расстояние между точками.

Если заданная точка лежит на прямой, то искомая точка совпадает с заданной.

Р25. Программа PCNP(P1, C, P, D) находит точку на заданной окружности, ближайшую к некоторой известной точке, и определяет расстояние между ними. Параметры программы:

- P1 — заданная точка;
- C — заданная окружность;
- P — искомая точка;
- D — расстояние между точками.

Если заданная точка лежит на окружности, то искомая точка совпадает с заданной.

Обращение к программе считается некорректным, если заданная точка совпадает с центром окружности.

### 3.1.2. Программы определения прямой.

L1. Программа LPP(P1, P2, L) определяет прямую линию (коэффициенты A, B, C), проходящую через две заданные точки. Параметры программы:

- P1, P2 — заданные точки;
- L — полученная прямая.

L2. Программа LPARX(D, L) определяет прямую линию, параллельную оси абсцисс на заданном расстоянии. Параметры программы:

|D| — расстояние:  $D > 0$  — прямая над осью X,  $D < 0$  — прямая под осью X;

- L — найденная прямая.

L3. Программа LPARY(D, L) определяет прямую линию, параллельную оси ординат на заданном расстоянии. Параметры программы:

|D| — расстояние:  $D > 0$  — прямая справа от оси Y,  $D < 0$  — прямая слева от оси Y;

- L — найденная прямая.

L4. Программа LPERLP(L1, P, L2) определяет прямую линию, проходящую через заданную точку и перпендикулярную заданной прямой. Параметры программы:

- L1 — заданная прямая;
- P — заданная точка;
- L2 — полученная прямая.

L5. Программа LPARLP(L1, P, L2) определяет прямую линию, проходящую через заданную точку и параллельную заданной прямой. Параметры программы:

- L1 — заданная прямая;
- P — известная точка;
- L2 — найденная прямая.

L6. Программа LPARLD(L1, D, L2) определяет прямую линию, параллельную заданной прямой и расположенную на заданном расстоянии. Параметры программы:

L1 — заданная прямая;

|D| — заданное расстояние:  $D > 0$  — прямая проводится в сторону возрастания координаты X,  $D < 0$  — в сторону убывания координаты X;

L2 — полученная прямая.

В случае, когда заданная прямая линия горизонтальная, то при  $D > 0$  параллельная прямая проводится выше заданной прямой, при  $D < 0$  — ниже.

L7. Программа LMLL(LM, L1, L2) определяет прямую линию, симметричную заданной прямой относительно другой прямой. Параметры программы:

LM — прямая, являющаяся осью симметрии;

L1 — заданная прямая;

L2 — найденная симметричная прямая.

L8. Программа LPAX(P, ALPHA, L) находит прямую линию, проходящую через заданную точку под заданным углом к оси X. Параметры программы:

P — заданная точка;

ALPHA — угол (в градусах);

L — найденная прямая.

L9. Программа LPAL(P, ALPHA, L1, L) определяет прямую линию, проходящую через заданную точку и составляющую с заданной прямой известный угол. Параметры программы:

P — точка;

ALPHA — угол (в градусах);

L1 — заданная прямая;

L — найденная прямая.

Если  $ALPHA > 0$ , то искомая прямая получается поворотом на заданный угол относительно заданной прямой против часовой стрелки, если  $ALPHA < 0$  — по часовой стрелке.

### 3.1.3. Программы определения окружности.

C1. Программа CXUR(X, Y, R, C) определяет окружность по заданным координатам центра и радиусу. Параметры программы:

X, Y — координаты центра;

R — радиус;

C — найденная окружность.

C2. Программа CPR(P, R, C) определяет окружность по заданной точке (центру) и радиусу. Параметры программы:

P, R — заданные точка и радиус;

C — найденная окружность.

С3. Программа CCNTRP(PCN, P, C) определяет окружность по заданным центру и точке, лежащей на окружности. Параметры программы:

PCN — центр окружности;  
P — заданная точка;  
C — найденная окружность.

С4. Программа CPPP(P1, P2, P3, C) находит окружность, проходящую через три заданные точки, не лежащие на одной прямой. Параметры программы:

P1, P2, P3 — три заданные точки;  
C — найденная окружность.

Обращение к программе считается некорректным, если все три точки лежат на одной прямой.

С5. Программа CCNTL(P, L, C) определяет окружность по заданным центру и касательной к окружности. Ее параметры:

P — центр окружности;  
L — касательная линия;  
C — найденная окружность.

С6. Программа CMLC(LM, C1, C2) определяет окружность, симметричную заданной окружности относительно известной прямой. Параметры программы:

LM — прямая, являющаяся осью симметрии;  
C1 — заданная окружность;  
C2 — найденная окружность.

С7. Программа CTLLP(L1, L2, P, N) определяет окружности, касающиеся двух заданных прямых и проходящие через заданную точку. Параметры программы:

L1, L2 — заданные прямые;  
P — заданная точка;  
N — число окружностей ( $0 \leq N \leq 2$ ).

Результат в блоке GF GMC. Решение выбирается с помощью программы SORTC.

С8. Программа STLPP(P1, P2, L, N) находит окружности, проходящие через две заданные точки и касающиеся заданной прямой. Параметры программы:

P1, P2 — заданные точки;  
L — заданная прямая;  
N — число окружностей ( $0 \leq N \leq 2$ ).

Результат в блоке GF GMC. Решение выбирается с помощью программы SORTC.

С9. Программа STCCP(C1, C2, P, N) определяет окружности, касающиеся двух заданных окружностей и проходящие через известную точку. Параметры программы:

C1, C2 — заданные окружности;  
P — заданная точка;  
N — число окружностей ( $0 \leq N \leq 2$ ).



Результат в блоке GFGMC. Решение выбирается программой SORTC.

C10. Программа CTCPP(P1, P2, C, N) находит окружности, проходящие через две заданные точки и касающиеся заданной окружности. Параметры программы:

P1, P2 — заданные точки;

C — заданная окружность;

N — число окружностей ( $0 \leq N \leq 2$ ).

Результат в общем блоке GFGMC. Для выбора решения используется программа SORTC.

C11. Программа RCCNTC(PCN, C, R1, R2) определяет радиусы окружностей с заданным центром, касающихся другой известной окружности. Параметры программы:

PCN — центр окружности;

C — заданная окружность;

R1, R2 — радиусы (R1 — больший радиус, R2 — меньший).

C12. Программа DANPPC(P1, P2, PC, D, ALPHA) находит длину меньшей дуги между двумя заданными точками окружности и ее угловую величину. Параметры программы:

P1, P2 — заданные точки, лежащие на окружности;

PC — центр окружности;

D — длина дуги;

ALPHA — угол (в градусах).

### 3.1.4. Определение центров окружностей заданного радиуса.

CN1. Программа CNCPP(P1, P2, R, N) находит центры окружностей заданного радиуса, проходящих через две заданные точки. Ее параметры:

P1, P2 — заданные точки;

R — радиус;

N — число окружностей ( $0 \leq N \leq 2$ ).

Результат в общем блоке GFGMP. Решение выбирается с помощью программы SORTP или программы SPNP.

CN2. Программа CNCTLL(L1, L2, R, N) определяет центры окружностей заданного радиуса, касающихся двух заданных прямых. Параметры программы:

L1, L2 — заданные прямые;

R — радиус;

N — число окружностей ( $N = 4$ , если прямые пересекаются, и  $N = 0$ , если прямые параллельны).

Результат в общем блоке GFGMC. Решение выбирается с помощью программы SORTCN. Обращение к программе считается некорректным, если прямые параллельны (в том числе совпадают).

CN3. Программа CNCTLP(L, P, R, N) находит центры окружностей заданного радиуса, проходящих через заданную точку и касающихся заданной прямой. Параметры программы:

L, P — заданные прямая и точка;  
R — радиус;  
N — число окружностей ( $0 \leq N \leq 2$ ).

Результат в общем блоке GFGMP. Решение выбирается с помощью программы SORTP или программы SPNP.

CN4. Программа CNCTCL(C, L, R, N) определяет центры окружностей заданного радиуса, касающихся известных прямой и окружности. Параметры программы:

C, L — заданные окружность и прямая;  
R — радиус;  
N — число окружностей ( $0 \leq N \leq 8$ ).

Результат в общем блоке GFGMC. Решение выбирается с помощью программы SORTCN. (При  $N = 2$  можно воспользоваться программой SORTP или программой SPNP.)

CN5. Программа CNCTCP(C, P, R, N) определяет центры окружностей заданного радиуса, проходящих через заданную точку и касающихся заданной окружности. Параметры программы:

C, P — заданные окружность и точка;  
R — радиус;  
N — число окружностей ( $0 \leq N \leq 4$ ).

Результат в общем блоке GFGMC. Решение выбирается с помощью программы SORTCN. (При  $N = 2$  можно воспользоваться программой SORTP или программой SPNP.)

CN6. Программа CNCTCC(C1, C2, R, N) находит центры окружностей заданного радиуса, касающихся двух заданных окружностей. Параметры программы:

C1, C2 — заданные окружности;  
R — радиус;  
N — число окружностей ( $0 \leq N \leq 8$ ).

Результат в общем блоке GFGMC. Решение выбирается с помощью программы SORTCN. (При  $N = 2$  можно воспользоваться программой SORTP или программой SPNP.)

CN7. Программа CNTLLL(L1, L2, L3, N) определяет центры окружностей, касающихся трех заданных прямых. Параметры программы:

L1, L2, L3 — заданные прямые;  
N — число окружностей ( $N = 0, 2, 4$ ).

Если прямые параллельны или пересекаются в одной точке, то обращение к программе считается некорректным.

Результат в общем блоке GFGMC. Решение выбирается с помощью программы SORTL.

**3.1.5. Программы выбора решения.** В тех случаях, когда при выполнении геометрической операции получается несколько решений, результат операции заносится в один из общих блоков. Таких блоков два: GFGMP(4) и GFGMC(16). Программы SORTP и

SPNP ведут поиск решения в общем блоке GFGMP, программы SORTC, SORTCN, SORTL и SORTPT — в общем блоке GFGMC.

Программа SORTP(J, P) позволяет выбрать точку из набора, полученного в результате геометрических вычислений с помощью любой из программ PILC, PICC, PVPL, PLDP, PCDP, PTLPC, CNCPP, CNCTLP. Параметры программы:

J — признак выбора точки: J = 1 — с большей X-координатой (если X-координаты равны, то с большей Y-координатой), J = 2 — с меньшей X-координатой (если X-координаты равны, то с меньшей Y-координатой), J = 3 — с большей Y-координатой (если Y-координаты равны, то с большей X-координатой), J = 4 — с меньшей Y-координатой (если Y-координаты равны, то с меньшей X-координатой);

P — найденная точка.

Программа SORTPT(J, PT1, PT2) предназначена для программы PTLCC. Параметры программы:

J — признак, по которому выбирается решение: J = 1 — обе точки касания находятся слева от линии центров, J = 2 — обе точки касания справа от линии центров, J = 3 — первая точка касания слева, а вторая — справа от линии центров, J = 4 — первая точка касания справа, а вторая — слева от линии центров;

PT1, PT2 — выбранные точки касания.

Линия центров направлена от центра окружности, записанной первой в списке параметров программы PTLCC, к центру окружности, записанной второй. PT1 — точка касания прямой с окружностью, записанной первой в списке параметров программы PTLCC, PT2 — точка касания с окружностью, записанной второй.

Программа SORTC(J, C) позволяет выбрать окружность из набора, полученного в результате геометрических вычислений с помощью любой из программ CTLLP, STLPP, STCCP, STCPP. Выбор окружности определяется положением ее центра. Параметры программы:

J — признак выбора нужной окружности:

J = 1 — с большей X-координатой центра (если X-координаты равны, то с большей Y-координатой), J = 2 — с меньшей X-координатой центра (если X-координаты равны, то с меньшей Y-координатой), J = 3 — с большей Y-координатой центра (если Y-координаты равны, то с большей X-координатой), J = 4 — с меньшей Y-координатой центра (если Y-координаты равны, то с меньшей X-координатой);

C — выбранная окружность.

Программа SORTCN(M, J, PC) предназначена для программ CNCTCC, CNCTCL, CNCTLL, CNCTCP. Параметры программы:

M — признак выбора группы окружностей (нужен для программ CNCTCC и CNCTCL): M = 1 — окружности, касающиеся ок-

ружности, записанной первой в списке параметров в программах CNCTCC и CNCTCL, снаружи,  $M \neq 1$  — касающиеся внутри;

J — признак выбора нужного решения: J = 1 — выбор центра окружности с большей координатой X, J = 2 — со второй координатой X, J = 3 — с третьей координатой X, J = 4 — с наименьшей координатой X, J = 5 — с большей координатой Y, J = 6 — со второй координатой Y, J = 7 — с третьей координатой Y, J = 8 — с наименьшей Y-координатой;

PC — выбранный центр окружности.

Программа SORTCN использует служебную программу SORT3.

Замечание. Для всех программ, кроме CNCTCC и CNCTCL, параметр M всегда должен быть равен 1.

Программа SORTL(L, J, C) предназначена для программы CNTLLL. Параметры программы:

L — любая из прямых, определенных в списке параметров программы CNTLLL;

J — признак выбора решения (тот же, что в программе SORTCN);

C — выбранная окружность.

Программа SPNP(P, PN) выбирает из двух точек точку, ближайшую к некоторой известной точке. Параметры программы:

P — заданная точка;

PN — искомая точка.

Если расстояния от обеих точек до заданной точки равны, то точка выбирается произвольно.

**3.1.6. Примеры.** На рис. 3.2 показан пример использования программ геометрических вычислений. Для данного чертежа приводится фрагмент программы, представляющий ее вычислительную часть. Рисование осуществлялось с помощью программ Графора.

REAL P1(2), P2(2), P3(2), P4(2), P5(2)

REAL P6(2), P7(2), P8(2), P9(2)

REAL P10(2), P11(2), P12(2), P14(2)

REAL P15(2), P16(2), P17(2), P18(2)

REAL L1(3), L2(3), L3(3), C1(3)

REAL C2(3), C3(3), C4(3), C5(3), C6(3)

REAL C7(3), C8(3), C9(3), C10(3)

DATA C1/0., 0., 35./, C2/-30., 0., 28./

DATA C4/-58., 0., 15./

**C...ВЫЧИСЛИТЕЛЬНАЯ ЧАСТЬ**

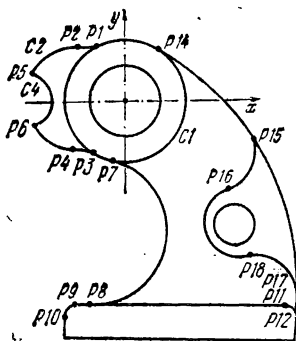


Рис. 3.2. Пример использования программ геометрических вычислений.

CALL CNCTCC(C1, C2, 15., N)  
 CALL SORTCN(1, 5, C3)  
 CALL CPR(C3, 15., C3)  
 CALL PICC(C1, C3, N)  
 CALL SORTP(2, P1)  
 CALL PICC(C2, C3, N)  
 CALL SORTP(3, P2)  
 CALL PMP(P1, P3, 1)  
 CALL PMP(P2, P4, 1)  
 CALL PICC(C4, C2, N)  
 CALL SORTP(3, P5)  
 CALL SORTP(4, P6)  
 CALL LPARX(-115., L1)  
 CALL CNCTCL(C1, L1, 41., N)  
 CALL SORTP(2, C5)  
 CALL CPR(C5, 41., C5)  
 CALL PICC(C1, C5, N)  
 CALL SORTP(1, P7)  
 CALL PILC(L1, C5, N)  
 CALL SORTP(1, P8)  
 CALL LPARY(-35., L2)  
 CALL CNCTLL(L1, L2, 10., N)  
 CALL SORTCN(1, 2, C6)  
 CALL CPR(C6, 10., C6)  
 CALL PILC(L1, C6, N)  
 CALL SORTP(1, P9)  
 CALL PILC(L2, C6, N)  
 CALL SORTP(1, P10)  
 CALL LPARY(31., L3)  
 CALL PMLP(L3, P9, P11)  
 CALL PMLP(L3, P10, P12)  
 CALL CNCTCP(C1, P12, 158.,  
 CALL SORTCN(1, 7, C7)  
 CALL CPR(C7, 158., C7)  
 CALL PICC(C7, C1, N)  
 CALL SORTP(1, P14)  
 CALL CXYR(60., -70., 20., C8)  
 CALL CNCTCC(C8, C7, 20., N)  
 CALL SORTP(3, C9)  
 CALL CPR(C9, 20., C9)  
 CALL SORTP(4, C10)  
 CALL CPR(C10, 20., C10)  
 CALL PICC(C7, C9, N)  
 CALL SORTP(1, P15)  
 CALL PICC(C8, C9, N)

CALL SORTP(1, P16)  
CALL PICC(C7, C10, N)  
CALL SORTP(1, P17)  
CALL PICC(C8, C10, N)  
CALL SORTP(1, P18)

## 3.2. Геометрические построения

**3.2.1. Запоминание геометрических элементов.** Для запоминания геометрических элементов предусмотрен специальный буфер, состоящий из двух позиций, в каждой из которых может быть сохранен один элемент (отрезок прямой, дуга окружности или эллипса). Если программа работает с одним из двух занесенных в буфер элементов, то он всегда берется из текущей позиции буфера. В тех случаях, когда используются оба элемента и когда важен их порядок, будем называть первым элемент в текущей позиции, а вторым — в другой позиции.

Существуют программы записи элементов в буфер и программы считывания элементов. Кроме того, есть подпрограмма STRMOD и функция MODGF. С помощью первой можно управлять адресацией в буфере, т. е. управлять выбором позиций в буфере, к которым производится обращение при записи или считывании элементов. С помощью второй можно опрашивать состояние адресной системы буфера, а также узнавать тип занесенных в буфер элементов.

Наконец, существуют две программы (WRSTR и RDSTR), которые позволяют пересылать информацию между буфером и масками в программе пользователя как в одну, так и в другую сторону. Таким образом, можно сохранить элемент, записанный в буфере, а впоследствии вновь занести его в буфер, не вычисляя.

**3.2.2. Программы записи и считывания элементов.** Программы записи и считывания элементов разбиты на три группы по типу элемента: отрезок, окружность, эллипс. Каждая программа соответствует одной из рассматривавшихся выше программ рисования (см. табл. 2). Все программы записи имеют имена, начинающиеся с букв WR, а программы считывания — с букв RD. За ними следуют трехбуквенные сокращения имен соответствующих программ рисования.

Программы записи имеют те же параметры, что и соответствующие программы рисования, за исключением программ записи отрезка, для которых отсутствует параметр J (перо опущено/поднято). Это естественно, так как он не относится к геометрическим характеристикам элемента. В программах считывания к параметрам соответствующих программ рисования добавляются координаты начальной точки (X0, Y0), которые для программы записи и

	Программы рисования	Программы записи	Программы считывания
Группа прямых	MOVE(X, Y, J) MOVE(DL, TH, J) MOVE(DX, DY, J) MOVE(X, Y, DL, J)	WRMVE(X, Y) WRMVA(DL, TH) WRMVB(DX, DY) WRMVC(X, Y, DL)	RDMVE(X0, Y0, X, Y) RDMVA(X0, Y0, DL, TH) RDMVB(X0, Y0, DX, DY) RDMVC(X0, Y0, X, Y, DL)
Группа окружностей	CIRC(R) ARCIA(R, THO, THF) ARCIB(R, XF, YF, J) ARCIC(XM, YM, XF, YF, J) ARCID(XC, YC, BETA)	WRCRC(R) WRACA(R, THO, THF) WRACB(R, XF, YF, J) WRACC(XM, YM, XF, YF, J) WRACD(XC, YC, BETA)	RDCRC(XC, YC, R) RDACA(X0, Y0, R, THO, THF) RDACB(X0, Y0, R, XF, YF, J) RDACC(X0, Y0, XM, YM, XF, YF, J) RDACD(X0, Y0, XC, YC, BETA)
Группа эллипсов	ELPS(A, B, ALPHA) ARCELA(A, B, ALPHA, THO, THF) ARCELB(A, B, ALPHA, XF, YF)	WRELPA(A, B, ALPHA) WRAELA(A, B, ALPHA, THO, THF) WRAELB(A, B, ALPHA, XF, YF)	RDELPA(XC, YC, A, B, ALPHA) RDAELA(X0, Y0, A, B, ALPHA, THO, THF) RDAELB(X0, Y0, ALPHA, XF, YF)

рисования соответствуют текущему положению пера и явно не задаются.

При выполнении программ записи информация об элементе записывается в буфер. При выполнении программ считывания переменным — фактическим параметрам — присваиваются характеристики считываемого геометрического элемента. При этом в пределах каждой группы элементов выбор программы считывания не зависит от того, какой программой был записан элемент. Таким образом, по одним характеристикам элемента можно получить другие.

Если элемент был записан как дуга окружности, можно считать его как полную окружность, которой эта дуга принадлежит (программой RDCRC), не получая при этом части записанной информации. Можно, наоборот, считывать как дугу окружности элемент, записанный как полная окружность. При этом начальная точка считываемой дуги будет совпадать с конечной, а радиус, проведенный в эту точку, будет иметь нулевой наклон по отношению к оси  $X$ .

Аналогичная картина имеет место и для эллипсов. При считывании полного эллипса, как дуги, начальная точка будет совпадать с конечной, а радиус, проведенный в конечную точку, будет совпадать с полуосью  $A$ . При считывании отрезка прямой программой RDMVC в качестве дополнительной точки выдается середина записанного отрезка и  $DL > 0$  (см. рис. 1.5,  $\epsilon$ ). Аналогично, при считывании дуги окружности программой RDACC в качестве дополнительной точки выдается середина записанной дуги и  $J = 0$ . Последние правила связаны с тем, что дополнительная точка не имеет непосредственного отношения к элементу и поэтому не запоминается в буфере.

Заметим, что в буфере вместе с характеристиками элемента запоминается его тип (отрезок, окружность, эллипс). Поэтому, если делается попытка считать элемент, записанный программой другой группы, то обращение считается неправильным и на печатающее устройство выдается диагностический текст **НЕ ТОГ ГЕОМЕТРИЧЕСКИЙ ЭЛЕМЕНТ**, затем происходит выход из программы без присвоения значений переменным, указанным в параметрах этой программы.

В заключение подчеркнем, что ни программы записи, ни программы считывания не рисуют изображения и не перемещают пера.

**3.2.3. Адресация позиций в буфере и обмен информацией между буфером и массивами пользователя.** Выше говорилось, что в буфере есть всего две позиции для запоминания элементов. Хотя эти позиции имеют номера 1 и 2, обращение к ним в командах считывания и записи осуществляется некоторым «безадресным» способом. Для команд записи существует два режима: работа с чередо-



ванием и без чередования позиций. При работе с чередованием каждые две последовательные команды записи запоминают элементы в разных позициях. Без чередования все элементы записываются в одну и ту же позицию. Любая команда считывания всегда выбирает элемент из текущей позиции, т. е. из той, в которую происходила последняя запись.

Программа STRMOD(J) устанавливает требуемый режим адресации в буфере и номер текущей позиции. Параметры программы:

J — признак адресации.

С сохранением режима адресации:

J = 0 — меняется номер текущей позиции,

J = 4 — данные первого и второго элементов буфера меняются местами, номер текущей позиции не меняется.

С установкой режима адресации:

$1 \leq J \leq 3$  — устанавливается режим с чередованием позиций,

$-3 \leq J \leq -1$  — устанавливается режим без чередования позиций,

$|J| = 1$  — текущей становится позиция 1,

$|J| = 2$  — текущей становится позиция 2,

$|J| = 3$  — номер текущей позиции не меняется.

Первоначально устанавливается режим, соответствующий J = 1.

Функция MODGF(J) позволяет опросить режим адресации, номер текущей позиции, а также узнать, к какому типу принадлежит геометрический элемент, записанный в текущей позиции. Значение единственного входного параметра J определяет вопрос, а значение самой функции — ответ на заданный вопрос:

J = 0 — опрашивается режим адресации и номер текущей позиции:

MODGF(0) > 0 — режим с чередованием позиций,

MODGF(0) < 0 — режим без чередования позиций,

$|MODGF(0)| = 1$  — позиция 1 — текущая,

$|MODGF(0)| = 2$  — позиция 2 — текущая;

J = 1 — опрашивается тип геометрического элемента в текущей позиции:

MODGF(1) = 0 — позиция пуста,

MODGF(1) = 1 — записан отрезок прямой линии,

MODGF(1) = 2 — записана окружность или ее дуга,

MODGF(1) = 3 — записан эллипс или его дуга.

Программа WRSTR(STORE) осуществляет перепись в буфера (обеих позиций, содержащих 28 вещественных чисел) в выделенный пользователем массив. Имя массива задается параметром STORE.

Программа RDSTR(STORE) выполняет перепись в буфер первых 28 чисел массива, задаваемого параметром STORE.

Эти программы дают возможность временно сохранять элементы, а затем пересылать их в буфер, чтобы вновь оперировать с ними.

Информацию, содержащуюся в буфере, можно использовать при изображении стрелок, размерных линий, а также для выполнения геометрических операций.

Программа ARROW(J) исходит из предположения, что элемент (отрезок прямой или дуга окружности), хранящийся в текущей позиции буфера, нарисован, и «пририсовывает» стрелки на

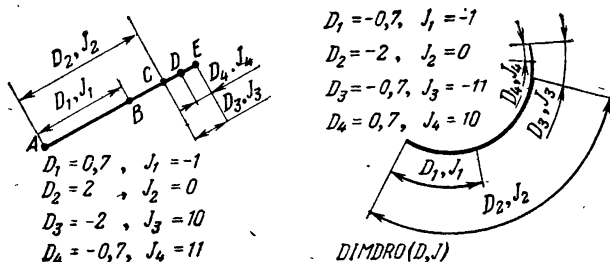


Рис. 3.3. Иллюстрация к программе DIMDRO(D, J).

его концах. Входной параметр J имеет следующие значения:  $J = -1$  — стрелка изображается при начальной точке,  $J = 1$  — стрелка изображается при конечной точке,  $J = 0$  — стрелки изображаются при обоих концах.

Стрелка всегда направлена изнутри во вне, а острое совпадает с концевой точкой.

Программа DIMDRO(D, J) изображает размерные линии для основной линии (отрезка прямой или дуги окружности), записанной в буфере. Программа имеет следующие параметры (рис. 3.3):

D — смещение главной размерной линии по отношению к основной линии (в выбранных единицах измерения):

$D > 0$  — главная размерная линия изображается выше (когда нормаль горизонтальна, то левее) основной линии,

$D < 0$  — главная размерная линия изображается ниже (когда нормаль горизонтальна, то правее) основной линии,

$D = 0$  — главная размерная линия совпадает с основной;

J — тип размерных линий:

$|J| < 10$  — расположение стрелок внутреннее,

$|J| \geq 10$  — расположение стрелок наружное,

$J < 0$  — боковая размерная линия проводится только справа (когда нормаль горизонтальна, то сверху) от нормали,

$J > 0$  и  $J \neq 10$  — боковая размерная линия проводится только слева (когда нормаль горизонтальна, то снизу) от нормали,

$J=0$  или  $J=10$  — боковые размерные линии проводятся с обеих сторон от нормали.

**3.2.4. Операции с геометрическими элементами.** Простейшее вычисление состоит в определении координат текущего положения пера и значения выбранной единицы измерения на странице. Дело в том, что перо может попасть в некоторую точку не только при явном задании ее координат. Например, после проведения отрезка заданной длины под заданным углом (MOVA) координаты пера неизвестны. Точно так же нужно считать их неизвестными после того, как нарисован подрисунок по подпрограмме, во внутренности детали которого нет необходимости вникать. В то же время координаты пера могут оказаться нужными для того, чтобы, например, «привязать» к определенному месту поясняющий текст или какие-либо другие элементы изображения.

Возможность опрашивать выбранные единицы измерения оказывается полезной при программировании подрисунков, размеры которых не должны зависеть от единицы измерения. Например, удобно, чтобы надписи, относящиеся к разметке координатных осей при построении графиков, имели постоянный размер. Во всех этих случаях можно воспользоваться программами WHERE или WHEREP (см. § 1.3).

Функция DIST(J) позволяет определить минимальное расстояние от текущей точки до ближайшей точки, принадлежащей геометрическому элементу, записанному в текущей позиции буфера. При вычислении расстояния имеется возможность «расширения» геометрического элемента: отрезка прямой до полной прямой, дуги окружности до полной окружности, а дуги эллипса до полного эллипса. Параметр J — это признак такого расширения. Если  $J=0$ , то геометрический элемент берется с расширением, а если  $J=1$ , — без расширения. Значение функции DIST и дает искомое расстояние.

Программа DDIST(X, Y) позволяет узнать координаты той точки, расстояние до которой было определено последней выполненной функцией DIST. Между обращениями к функции DIST и программе DDIST могут выполняться любые другие действия, кроме обращения к программе CROSS. В последнем случае, так же как и в случае, когда не было предварительного обращения к функции DIST, программа DDIST выдаст неверные результаты.

Программа SECANT(SL, ALPHA, X, Y) вычисляет координаты конца отрезка заданной длины, начинающегося в текущей точке и проходящего под заданным углом к прямой, которая хранится в текущей позиции буфера (рис. 3.4, а). Начальные точки этих прямых обозначены на рисунке соответственно через  $X0$ ,  $Y0$  и  $X01$ ,  $Y01$ . Параметры программы:

SL — длина отрезка (если  $SL=0$ , т. е. не задана длина, концом отрезка считается точка его пересечения с прямой);

ALPHA — угол между отрезком и прямой (в градусах);

X, Y — координаты конца отрезка.

Обращение к программе считается некорректным, если  $ALPHA=0^\circ$  или  $ALPHA=180^\circ$ , а  $SL=0$ , и неправильным, если очередной элемент для считывания не является прямой.

Программа CROSS(X, Y, K) проверяет, пересекаются ли два хранящихся в буфере геометрических элемента и, если пересекаются, то вычисляет координаты точек пересечения. Каждое обращение к программе дает координаты только одной точки. Если геометрические элементы пересекаются в нескольких точках, то для получения всех их необходимо несколько раз обратиться к программе CROSS. Точки пересечения ищутся для «расширенных» геометрических элементов (см. функцию DIST). Программа имеет следующие параметры:

-X, Y — координаты очередной точки пересечения;

K — характеристика очередной точки:

$K < 0$  — выданная точка не последняя,

$K > 0$  — выданная точка последняя;

$K = 0$  — элементы не пересекаются, координаты не определены,

$|K|=1$  — пересекаются сами элементы,

$|K|=2$  — первый элемент пересекает «продолжение» второго,

$|K|=3$  — второй элемент пересекает «продолжение» первого,

$|K|=4$  — пересекаются «продолжения» элементов.

Если после выдачи последней точки не было записей в буфер и снова происходит обращение к программе CROSS, то значения X, Y, K не определены, на печатающем устройстве выдается диагностический текст ВСЕ ТОЧКИ ВЫДАНЫ. Программа CROSS вычисляет координаты точек пересечения только для прямых и окружностей. Если в какой-либо позиции буфера окажется эллипс, обращение к программе считается неправильным.

Программа LITAN(XT, YT, J) находит точку касания (XT, YT) прямой, проходящей через текущую точку и касающейся окружности в буфере (рис. 3.4, б). Параметр J задает вариант касания:  $J=1$  — касание справа,  $J=-1$  — касание слева. Направления «справа» и «слева» определяются по отношению к лучу, проведенному из текущей точки в центр окружности. Если текущая точка оказывается внутри окружности, обращение к программе считается некорректным. Если в буфере не окружность, обращение к программе считается неправильным.

Программа CIRTAL(R, XT, YT, J) находит точку (XT, YT), в которой окружность радиуса R, проходящая через текущую точку, касается прямой, записанной в буфере. При  $J=1$  точка касания ищется справа от луча, проведенного из текущей точки

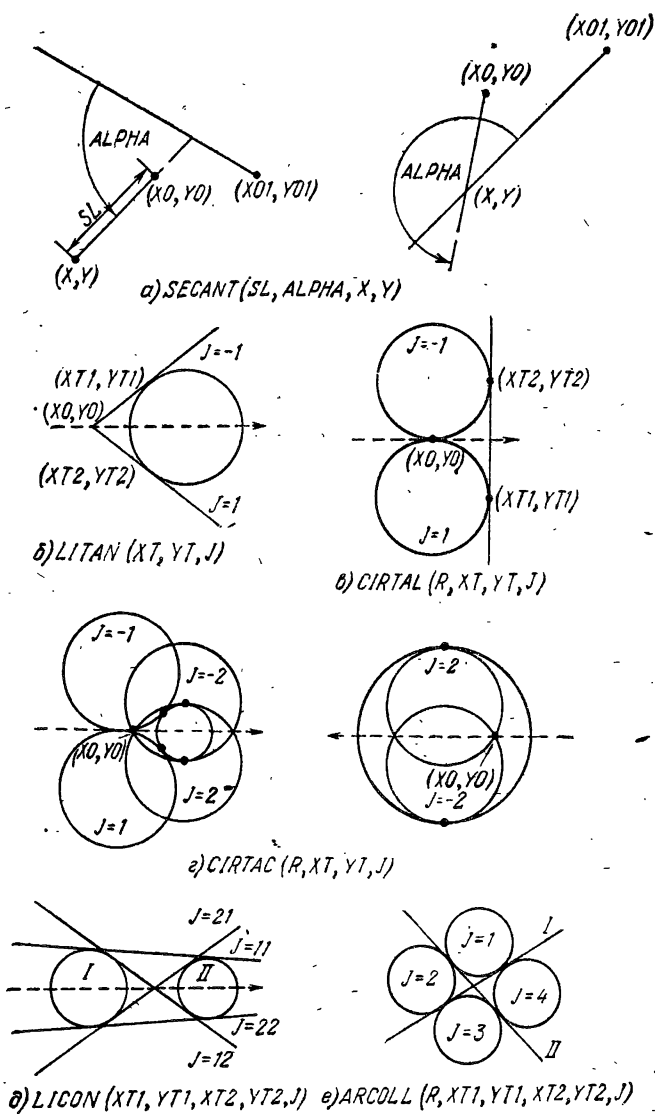


Рис. 3.4. Иллюстрации к программам геометрических построений.

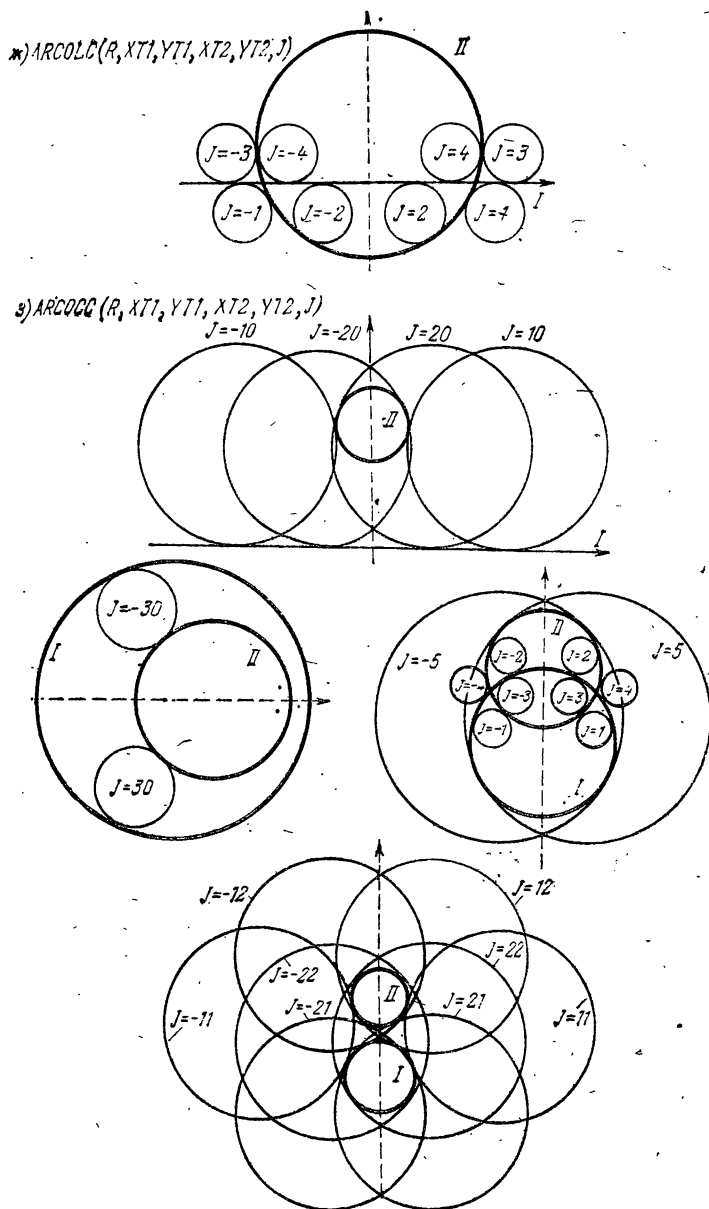


Рис. 3.4. Иллюстрации к программам геометрических построений.

перпендикулярно прямой, при  $J = -1$  — слева (рис. 3.4, е). Если расстояние от текущей точки до прямой превышает диаметр окружности, обращение к программе считается некорректным. Если в буфере записана не прямая, то обращение к программе считается неправильным.

Программа CIRTAC(R, XT, YT, J) находит точку (XT, YT), в которой окружность заданного радиуса R касается окружности, хранящейся в текущей позиции буфера (рис. 3.4, е). В дальнейшем будем называть эту окружность основной. Параметр J определяет вариант касания окружностей следующим образом:

$|J| = 1$  — касание наружное,

$|J| = 2$  — касание внутреннее,

$J > 0$  — центр касательной окружности находится справа от луча, идущего из точки (X0, Y0) в центр основной окружности,

$J < 0$  — центр касательной окружности находится слева от того же луча.

Обращение к программе считается некорректным, если:

а) диаметр касающейся окружности меньше расстояния от текущей точки до ближайшей точки, лежащей на основной окружности;

б) при внутреннем касании диаметр касательной окружности меньше (текущая точка вне окружности) или больше (текущая точка внутри окружности) расстояния от текущей точки до самой удаленной точки основной окружности;

в) касание задано наружное, а текущая точка лежит внутри основной окружности;

г) текущая точка совпадает с центром основной окружности.

Если при обращении к программе CIRTAC окажется, что в текущей позиции буфера записана не окружность, то обращение к программе считается неправильным.

Программа LICON(XT1, YT1, XT2, YT2, J) находит точки касания общей касательной прямой для двух окружностей, записанных в буфере. Программа имеет следующие параметры (рис. 3.4, д):

XT1, YT1; XT2, YT2 — координаты точек касания на первой и второй окружностях;

J — вариант касания:

J=11 — обе точки касания находятся слева от линии центров,

J=12 — точка касания первой окружности находится слева от линии центров, а второй — справа,

J=21 — точка касания первой окружности находится справа от линии центров, а второй — слева,

J=22 — обе точки касания находятся справа от линии центров.

Линия центров в этом случае направлена от центра первой окружности к центру второй окружности.

Обращение к программе считается некорректным, если одна окружность находится внутри другой или окружности пересекаются, а точки касания требуется найти по разные стороны от линии центров. Если же в буфере хотя бы один элемент не является окружностью, то обращение к программе считается неправильным.

Программа ARCOLL(R, XT1, YT1, XT2, YT2, J) находит точки касания общей касательной окружности для двух пересекающихся прямых, записанных в буфере. Параметры программы (рис. 3.4, е):

R — радиус касательной окружности;

XT1, YT1; XT2, YT2 — координаты точек касания на первой и второй прямых;

J — номер квадранта, образованного прямыми, в котором должна находиться касательная окружность.

Квадранты нумеруются, начиная с единицы, против часовой стрелки от положительного направления первой прямой. Направление второй прямой несущественно. Если прямые параллельны, то обращение к программе считается некорректным. Если хотя бы в одной позиции буфера записана не прямая, то обращение к программе считается неправильным.

Программа ARCOLC(R, XT1, YT1, XT2, YT2, J) находит точки касания общей касательной окружности для прямой и окружности, записанных в буфере. Параметры программы (рис. 3.4, ж):

R — радиус общей касательной окружности; •

XT1, YT1; XT2, YT2 — координаты точек касания на первом и втором элементах в буфере;

J — вариант касания:

$J > 0$  — центр касательной окружности находится справа,

$J < 0$  — центр касательной окружности находится слева,

$|J| < 10$  — прямая и окружность пересекаются, образуя два сегмента:  $|J|=1$  — наружное касание со стороны меньшего сегмента,  $|J|=2$  — внутреннее касание со стороны меньшего сегмента,  $|J|=3$  — наружное касание со стороны большего сегмента,  $|J|=4$  — внутреннее касание со стороны большего сегмента.

$|J| \geq 10$  — прямая и окружность не пересекаются:  $|J|=10$  — наружное касание,  $|J|=20$  — внутреннее касание.

Расположение «слева» или «справа» для центра касательной окружности определяется относительно перпендикуляра к записанной в буфере прямой, проходящего через центр записанной в буфере окружности. Перпендикуляр направлен от прямой к центру окружности, если прямая находится в текущей позиции буфера, и наоборот, от центра окружности к прямой, если в текущей позиции находится окружность.

Когда прямая проходит через центр окружности, большим условно считается сегмент, находящийся слева от самой прямой с учетом ее направления от начала к концу.



Обращение к программе считается некорректным, если:

а)  $|J| < 10$ , а занесенные в буфер окружности и прямая не пересекаются, либо наоборот,  $|J| \geq 10$ , а они пересекаются;

б)  $|J| < 10$  и четно (внутреннее касание), а окружность радиуса  $R$  не может поместиться внутри заданного сегмента;

в)  $|J| \geq 10$ , и диаметр касающейся окружности меньше при наружном касании минимального, а при внутреннем касании максимального расстояния от прямой до точки, принадлежащей окружности из буфера.

Если в буфер занесены не прямая и окружность, то обращение к программе считается неправильным.

Программа ARCOCC( $R, XT1, YT1, XT2, YT2, J$ ) находит точки касания общей касательной окружности для двух окружностей из буфера. Программа имеет следующие параметры (рис. 3.4, а):

$R$  — радиус общей касательной окружности;

$XT1, YT1; XT2, YT2$  — координаты точек касания соответственно на первой и второй окружностях в буфере;

$J$  — вариант касания:

$J > 0$  — центр касательной окружности справа от линии центров,

$J < 0$  — центр касательной окружности слева от линии центров (пусть  $I$  обозначает первую, а  $II$  — вторую окружность в буфере);

$|J| < 10$  — окружности, записанные в буфере, пересекаются:  $|J| = 1$  — касание с  $I$  внутреннее, а с  $II$  — наружное,  $|J| = 2$  — касание с  $I$  наружное, а с  $II$  — внутреннее (касательная окружность внутренняя по отношению к  $II$ ),  $|J| = 3$  — касание с  $I$  и  $II$  внутреннее (касательная окружность внутренняя по отношению к  $I$  и  $II$ ),  $|J| = 4$  — касание с  $I$  и  $II$  наружное,  $|J| = 5$  — касание с  $I$  и  $II$  внутреннее (касательная окружность внешняя по отношению к  $I$  и  $II$ ),

$|J| > 10$  — окружности, записанные в буфере, не пересекаются:  $|J| = 11$  — касание с  $I$  и  $II$  наружное,  $|J| = 12$  — касание с  $I$  наружное, а с  $II$  — внутреннее (касательная окружность внешняя по отношению к  $II$ ),  $|J| = 21$  — касание с  $II$  наружное, а с  $I$  — внутреннее (касательная окружность внешняя по отношению к  $I$ ),  $|J| = 22$  — касание с  $I$  и  $II$  внутреннее (касательная окружность внешняя по отношению к  $I$  и  $II$ ),  $|J| = 30$  — с меньшей окружностью касание наружное, а с большей — внутреннее.

Линия центров здесь считается направленной от первого элемента ко второму.

Обращение к программе ARCOCC считается некорректным, если:

а)  $|J| < 10$ , однако занесенные в буфер окружности не пересекаются, либо наоборот,  $|J| > 10$ , а окружности пересекаются;

б) диаметр внутренней окружности недостаточно мал либо диаметр наружной окружности недостаточно велик в многочисленных случаях внутреннего касания;

в) окружности не пересекаются и удалены друг от друга более чем на диаметр касающейся окружности;

г) окружности находятся одна внутри другой, а  $|J| \neq 30$ .

Если в буфер занесена не пара окружностей, обращение считается неправильным.

При реализации многих программ, описанных в этом параграфе, требовалось определить угол наклона отрезка к оси или же выяснить, где находится значение некоторой величины по отношению к окрестности некоторой точки на числовой оси. Для этих целей предназначены подпрограммы-функции **ANGLER** и **IVEST**.

Ф у н к ц и я **ANGLER(DELTA X, DELTA Y)** для отрезка прямой, заданного приращениями координат **DELTA X** и **DELTA Y**, вычисляет угол его наклона к оси **X**. Значение функции дает величину угла в градусах.

Ф у н к ц и я **IVEST(A, B, EPS)** определяет, где находится значение заданной переменной по отношению к окрестности некоторой точки на числовой оси. Параметры программы:

**A** — проверяемая величина;

**B** — центр окрестности на числовой оси;

**EPS** — радиус окрестности.

Значение самой функции отвечает на заданный вопрос следующим образом:

**IVEST** = -1 — значение лежит левее окрестности ( $A < (B - EPS)$ ),

**IVEST** = 0 — значение принадлежит окрестности ( $(B - EPS) \leq A \leq (B + EPS)$ ),

**IVEST** = 1 — значение лежит правее окрестности ( $(B + EPS) < A$ ).

**3.2.5. Примеры.** В качестве примера использования описанных выше программных средств рассмотрим чертеж ступицы, изображенный на рис. 3.5, а также два фрагмента программы построения этого чертежа:

а) Фрагмент программы, формирующий внешний контур чертежа ступицы с **N** зубьями:

```
CALL WHERE(XC, YC, F)
```

```
CALL WRCRC(D(5)/2)
```

```
CALL MOV B(D(5)/2, 0, 0)
```

```
CALL WRCRC(R)
```

```
CALL CROSS(X1, Y1, J)
```

```
CALL CROSS(X2, Y2, J)
```

```
TH=ANGLER(X1-XC, Y1-YC)*0.01745329252
```

```
CALL MOVE(X2, Y2, 0)
```



```

Y1=YC+D(5)/2*SIN(TH1)
Y2=YC+D(5)/2*SIN(TH2)
CALL FATARC(-R, X1, Y1, 0, 0.5)
CALL FATARC(D(5)/2, X2, Y2, 0, 0.5)
TH1=TH1+DTH
TH2=TH2+DTH

```

1 CONTINUE

б) Фрагмент программы, формирующий основные размерные  
линии:

```

CALL MOVE(XC-B/2, Y1, 0)
CALL WRMVE(XC+B/2, Y1)
CALL DIMDRO(-3.5, 0)
DO 2 J=1,4
CALL MOVE(XC-D(J)/2, YC, 0)
CALL WRMVE(XC+D(J)/2, YC)
CALL DIMDRO(-6.25-0.75*J, 0)

```

2 CONTINUE

## 4. ПОСТРОЕНИЕ ГРАФИКОВ

При решении довольно широкого класса задач графическое устройство используется для изображения различных функциональных зависимостей в виде графиков. В простейшем случае это может быть построение кривой (точнее ломаной), заданной таблицей значений, и проведение осей. При этом координаты точек могут иметь физический смысл (плотность, время, вес и т. д.) и представляться в том масштабе, который удобен для вычислений. В этом случае будем говорить, что функциональная зависимость задана в некотором *математическом пространстве*. Пределы изменения функции и ее аргумента являются характеристиками этого пространства.

Предметом обсуждения данной главы и будут вопросы определения математического пространства и его характеристик, построения кривых вида  $y=f(x)$  в декартовой и полярной системах координат, проведение координатных осей. Причем в декартовой системе координат могут использоваться равномерные, логарифмические и полулогарифмические шкалы. В эту главу также включены описания программ для построения гистограмм.

### 4.1. Определение области. Математические координаты

Как отмечалось выше, страница определяет *физическое пространство*; его размер, связанную с ним систему координат и линейные единицы измерения. При задании математического пространства определяется функция его отображения на физическое пространство, называемое в дальнейшем *областью отображения* или просто *областью*. Область — это рабочее поле, выделяемое в пределах страницы с некоторой другой, удобной для пользователя системой координат. В частности, при задании страницы ее поле автоматически определяется как область, совпадающая со страницей.

Задание функции преобразования требует обращения к двум программам REGION (область) и LIMITS (пределы). При вызове программы REGION (для полярной системы координат — POLREG)

указывается место области на странице и ее размеры. Другая программа, LIMITS, фиксирует *пределы изменения* аргумента и функции. Такое разделение естественно и логично, поскольку можно определить новую область, сохраняя прежние пределы, и наоборот, т. е. эти две компоненты, определяющие математическое пространство, по существу независимы.

Обращение к программе REGION задает в пределах страницы прямоугольную область с границами, параллельными сторонам страницы. При этом определяется место текущего графика и дается разрешение рисовать в этой области. Область может быть очерчена. Вдоль внешней стороны верхней границы области выписывается название области — текст, заданный программистом (см. рис. 1.3).

Программа REGION(X, Y, XL, YL, NAME, N, J) позволяет определить на заданной странице прямоугольную область для построения графиков, изолиний, проекций поверхностей и др. Параметры программы:

X, Y — координаты левого нижнего угла области относительно начала координат страницы (левого нижнего угла страницы);

XL, YL — размеры области вдоль осей X и Y;

NAME — название области;

N — количество литер в названии;

J — признак очерчивания границ области: J=0 — границы не очерчиваются, J=1 — границы очерчиваются.

Место расположения и размеры области должны быть такими, чтобы она целиком помещалась внутри страницы. Если длина хотя бы одной стороны области меньше 1 см, область не будет заведена. Если же координаты и размеры сторон заданы так, что область не помещается на странице, то она обрезается по правой и верхней границам страницы. При каждом обращении к программе REGION открывается новая область в пределах текущей страницы, а доступ в предыдущую прекращается. Проверок на пересечение областей не производится. Координаты и размеры области задаются в выбранных единицах измерения.

Все, что строится в области, масштабируется в зависимости от размеров области и пределов изменения функции, заданных программистом при обращении к программе LIMITS. Обычно левой и нижней границам области соответствуют минимальные значения координат X и Y, а правой и верхней — максимальные. Тем самым определяется правая система координат. Поменяв в обращении пределы местами, можно ориентировать оси произвольным образом. Установленные пределы сохраняются до очередного обращения к программе LIMITS. Величины, связанные с определением математического пространства, хранятся в общем блоке GFTAB (см. табл. 1).

Программа **LIMITS**(**XMIN**, **XMAX**, **YMIN**, **YMAX**) предназначена для задания пределов изменения функции и ее аргумента. Ее параметры:

**XMIN** — математическое значение координаты  $X$ , соответствующее левой границе области;

**XMAX** — математическое значение координаты  $X$ , соответствующее правой границе области;

**YMIN** — математическое значение координаты  $Y$ , соответствующее нижней границе области;

**YMAX** — математическое значение координаты  $Y$ , соответствующее верхней границе области.

Если программе априори неизвестны пределы изменения функции и/или аргумента, то для поиска в массиве максимальной и минимальной величины можно воспользоваться программой **MINMAX**.

Программа **MINMAX**(**A**, **N**, **AMN**, **AMX**) позволяет определить минимальное и максимальное значения в массиве чисел. Ее параметры: **A** — массив чисел длины **N**, **AMN** — минимальное значение, **AMX** — максимальное.

При рисовании в области может понадобиться вывести перо графплоттера в определенную «физическую» точку, координаты которой известны программисту только как математические величины. Это может потребоваться, например, чтобы надписать значение какой-либо точки кривой либо пометить определенную точку маркером и т. п. Для вычисления страничных координат точки по заданным математическим координатам имеется программа **TMF**.

Программа **TMF**(**XM**, **YM**, **XF**, **YF**) позволяет перевести математические координаты точки в области в координаты на странице. Параметры программы следующие:

**XM**, **YM** — математические значения координат;

**XF**, **YF** — значения координат в заданных единицах измерения.

Существует также и обратное преобразование координат.

Программа **TFM**(**XF**, **YF**, **XM**, **YM**) переводит страничные координаты в математические (знак страничной координаты не учитывается). Ее параметры аналогичны одноименным в программе **TMF**.

## 4.2. Графики в декартовой системе координат

Функция для вывода в общем случае задается как две последовательности координат точек и график функции рисуется в виде ломаной. Хотя после того как реализованы подпрограммы перехода к страничным координатам алгоритм вывода кривой оказывает-

ся тривиальным, некоторые моменты, тем не менее, заслуживают внимания.

Во-первых, после вычисления страничных координат точка может оказаться за пределами области. В этом случае (если не допускается выход за пределы области) точка проецируется на ближайшую границу области. Это экономный способ, но он приводит к искажению ближайшего к границе отрезка кривой. Заметим, что при автоматическом выборе масштаба (с использованием программы MINMAX) описанная ситуация не возникает и, кроме того, обеспечивается наилучшее заполнение поля рисунка.

Во-вторых, в программах для построения кривых выбирается ближайший к местонахождению пера конец кривой и с него начинается проведение этой кривой. При построении нескольких кривых на одном графике такой выбор может сократить (иногда вдвое) путь пера ценой лишь незначительного увеличения времени работы программы.

В-третьих, в комплексе Графтор имеется около десятка программ для проведения кривых. Вообще говоря, их можно было бы объединить в одну-две подпрограммы. Но это привело бы к значительному усложнению и удлинению списка параметров. Кроме того, в память загружались бы вспомогательные подпрограммы, которые необходимы в частных случаях, но оказываются «мертвым грузом» в большинстве ситуаций.

И, наконец, в-четвертых, кривые, если на графике их несколько, можно пометить маркерами из имеющегося набора. При этом, если номер маркера задан отрицательным, то размер маркера будет вдвое меньше стандартного (не 3 мм, а 1,5 мм).

Программа LINEO(X,Y,N) позволяет начертить в заданной области ломаную, связывающую N точек. В массивах X и Y передаются соответственно значения абсцисс и ординат точек, определяющих функцию, а N указывает количество элементов в этих массивах.

Программа LINEMO(X, Y, N, NM, JS) позволяет начертить в заданной области ломаную, связывающую N точек, и пометить заданные точки маркерами или пометить заданные точки маркерами, не вычерчивая кривой. Параметры программы:

X, Y — массивы абсцисс и ординат точек;

N — количество точек;

|NM| — номер маркера (если  $NM < 0$ , изображается маркер уменьшенных размеров);

|JS| — шаг маркировки (если  $JS < 0$ , линия не проводится, если  $JS = 0$  или  $JS = 1$  — маркируются все точки).

Вообще говоря, с любой точкой графика можно связать произвольные буквенные и числовые пометки. Для их изображения необходимо преобразовать математические координаты точки в



страничные (программа TMF) и затем воспользоваться программой SYMBOL или NUMBER.

В Графоре предусмотрена также возможность проведения замкнутых кривых. Для этого служат программы LINEC и LINEMC, обращение к которым аналогично обращению к программам LINEO и LINEMO. Отличие состоит лишь в том, что проводится отрезок прямой, соединяющий последнюю точку с первой.

В описанных выше программах предполагается, что функция задана в виде последовательности пар координат точек, вычисленных заранее. Однако если функция задана на равномерной сетке, то нет необходимости хранить все значения аргумента. И тогда можно воспользоваться программой INCLIN.

Программа INCLIN(XBEG, DXEX, JX, Y, N, NM, JS) позволяет вычертить график функции, заданной на отрезке с постоянным шагом. Ее параметрами являются:

XBEG — начальное значение аргумента;

DXEX — величина постоянного шага по аргументу или конечное значение аргумента (в зависимости от значения признака JX);

JX — признак параметра DXEX: JX=0 — задан шаг, JX=1 — задано конечное значение;

Y — вектор значений функции (длины N);

N — количество точек;

|NM| — номер маркера (если NM<0, изображается маркер уменьшенных размеров);

|JS| — шаг маркировки (если JS>0, вычерчивается линия с маркерами, если JS=0, вычерчивается линия без маркеров, если JS<0, изображаются только маркеры).

Следует заметить, что, задавая количество точек кривой N со знаком минус, мы разрешаем выход кривой за пределы области, но не далее границ страницы. Это замечание имеет отношение ко всем ранее описанным программам проведения линий.

В одной и той же области могут быть изображены несколько функциональных зависимостей. Один из способов различения кривых, соответствующих этим зависимостям, был описан ранее (маркировка). Другая возможность связана с рисованием прерывистых (штриховых и штрихпунктирных) линий. Эта возможность реализуется с помощью программы BRLINE. Предварительно до обращения к ней задается режим проведения линии и размеры штрихов и просветов между штрихами (программы FULL и BROKEN).

Программа FULL задает режим проведения непрерывной линии. Эта программа не имеет параметров.

Программа BROKEN(A1, A2, A3, A4) задает режим проведения прерывистой линии и характер линии. Ее параметры следующие:

- A1 — длина первого штриха;
- A2 — длина просвета после первого штриха;
- A3 — длина второго штриха;
- A4 — длина второго просвета.

Все значения задаются в выбранных единицах измерения. Конфигурация, заданная двумя штрихами и двумя просветами, повторяется необходимое число раз.

Программа `BRLINE(X, Y, N)` позволяет начертить непрерывную, штриховую или штрихпунктирную линию по заданной последовательности точек. Вид линии зависит от установленного режима. В массивах `X` и `Y` передаются соответственно значения абсцисс и ординат точек, определяющих функцию, а `N` указывает количество элементов в этих массивах. Задавая параметр `N` отрицательным, можно начертить «негативную» линию, когда штрихи заменяются просветами, а просветы штрихами.

Еще одна программа, `LINNUM`, предоставляет возможность вычерчивания прерывистых линий в зависимости от установок программами `FULL` или `BROKEN`. Кроме того, она позволяет вписать заданное число в разрыв линии. Число задается при обращении к программе `SIZNUM`.

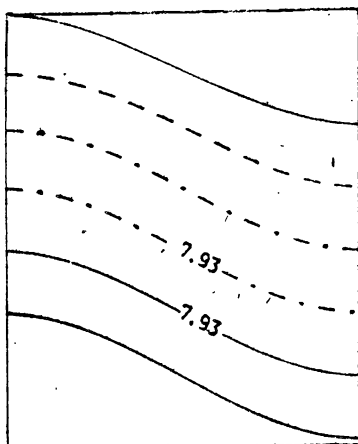
Программа `SIZNUM(SIZE, FNUM, N, M)` устанавливает режим вписывания числа в разрыв линии. Параметры программы: `SIZE` — высота литеры в выбранных единицах измерения; `FNUM` — число с плавающей точкой; `N` — количество цифр в числе; `M` — количество требуемых дробных знаков.

Отменить режим вписывания числа можно, обратившись к программе `RENUM`. Эта программа не имеет параметров.

Программа `LINNUM(X, Y, N)` позволяет по данному массиву точек начертить линию  $\Delta$  в том месте, где линия имеет наименьшую кривизну, вписать заданное число. Если длина линии меньше восьми длин текста, то это вписывание не произойдет. Параметры программы `X, Y`, как и в программе `BRLINE`, передают значения абсцисс и ординат точек, а `N` указывает количество элементов в этих массивах. Для программы `LINNUM` программа `BRLINE` является служебной (рис. 4.1).

На рис. 1.3 показан пример построения графиков с использованием описанных программ, а на рис. 1.4 приведена программа, с помощью которой выполнено это построение.

На рис. 4.2 три простые кривые изображены в двух системах координат, отличающихся только направлением оси ординат. Смена направления выполняется очень просто — заданием новых пределов (`LIMITS`). Приводится также пример нанесения надписи, связанной с точкой математического пространства. Страничные координаты определяются при обращении к программе `TMF`.



```
CALL LINNUM(X,Y1,N)
CALL BROKEN(0.5,0.5,0.5,0.5)
CALL LINNUM(X,Y2,N)
CALL BROKEN(0.5,0.5,0.1,0.5)
CALL LINNUM(X,Y3,N)
CALL SIZNUM(0.5,7.93,4,2)
CALL LINNUM(X,Y4,N)
CALL FULL
CALL LINNUM(X,Y5,N)
CALL RENUM
CALL LINNUM(X,Y6,N)
```

Рис. 4.1. Пример использования программ вычерчивания и разметки линий.

```

DIMENSION YL(50), YP(50), YC(50)
M=38
X=0.
YL(1)=0.
YP(1)=0.
CALL PAGE(20., 26., 0, 0, 0)
CALL LIMITS(-.5, 4.3, -4., 68.)
CALL REGION(1., 13.5, 14., 12., 0, 0, 0)
J=1
DO 100 I=2, M
  X=X+.1
  YL(I)=9.*X
  YP(I)=X**3
100 YC(I)=32./X
102 CALL INCLIN(0., .1, 0, YP, M, 5, 5)
  CALL INCLIN(0., .1, 0, YL, M, 0, 0)
  CALL TMF(3., 27., XF, YF)
  CALL SYMBOL(XF+.5, YF, .3, '3.00, 27.00', 10, 0.)
  CALL INCLIN(.3, .1, 0, YC(4), M-3, 2, -2)
  CALL AXES(0, 0, 0., 5, 0, 0, 10., 5, 0)
  GO TO (106, 107) J
106 CALL LIMITS(-.5, 4.3, 68., -4.)
  CALL REGION(1., 0.5, 14., 12., 0, 0, 0)
  J=2
```

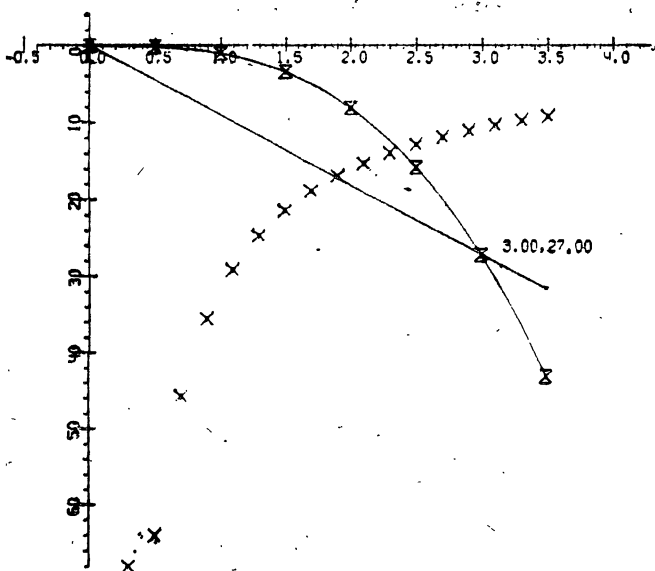
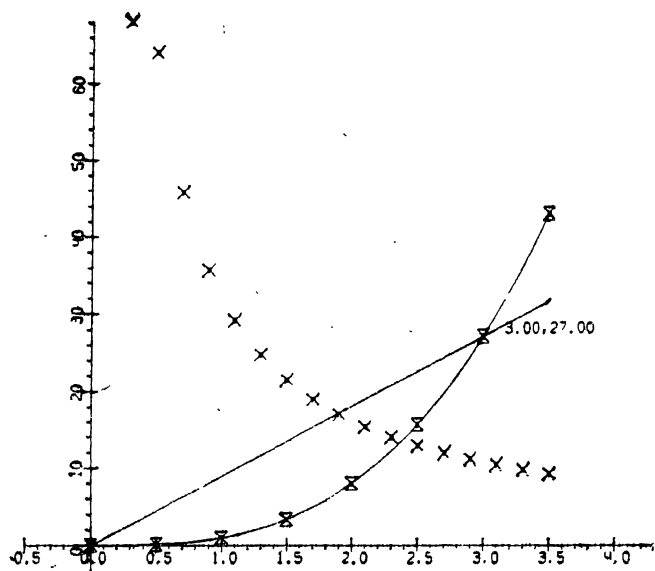


Рис. 4.2. Изображение графиков функций.

GO TO 102  
107 CALL ENDPG('4.2')  
END

#### **4.3. Построение осей координат в декартовой системе**

При оформлении графика важное значение имеют оси с пометками (или шкалы). При конструировании алгоритмов и программ для построения осей приходится учитывать разнообразные требования в различных сочетаниях и вариантах. Так, в Графоре не удалось ограничиться одной программой AXES, которая автоматически выбирает расположение осей в области, проводя их через нулевое или ближайшее к нулевому основное деление. Были добавлены программы, предоставляющие программисту возможность управлять положением осей и ориентацией числовых пометок. Кроме того, для построения логарифмических шкал в прямоугольной системе координат и для построения осей в полярной системе координат были разработаны отдельные программы.

Обратим внимание на принципы построения осей, принятые в Графоре. Оси могут быть названы и их названия подписываются за пределами области, вдоль ее границ (AXES) или рядом с осью (XAXIS, YAXIS). Ось размечается основными и дополнительными делениями. С основными делениями соотносятся их численные (математические) значения. Количество дополнительных делений внутри основного задается при обращении к соответствующей программе.

Размер основного деления задается программистом или выбирается автоматически. Автоматический выбор происходит в том случае, если указан нулевой размер или если размер задан неправильно и делается попытка разместить на оси более 20 основных делений. При автоматическом выборе на оси размещается от 5 до 15 основных делений.

Численные значения, которые соответствуют основным делениям, фиксируются на графике. Их дробная часть состоит не более чем из трех цифр, а общее количество значащих цифр не должно превышать пяти. Это ограничение, с одной стороны, определяется подпрограммой перевода двоичных чисел в десятичные (BCD), с другой стороны, оно разумно, поскольку иначе возникают затруднения при чтении графика. (Следует заметить, что при ручном построении графиков, как правило, ограничения еще жестче.) Однако эти ограничения чисто графические, и они не предъявляют каких-либо дополнительных требований к величинам, с которыми оперирует программа (т. е. к математическому пространству). При необходимости проводится масштабирование и указы-

вается масштабный коэффициент  $*10^{\pm m}$  вслед за названием оси (см., например, рис. 4.5). Кроме того, через основные деления на осях может быть проведена координатная сетка.

Программа AXES(NAMEX, NX, UX, KX, NAMEY, NY, UY, KY, M) позволяет провести и разметить оси абсцисс и ординат. Параметры программы:

NAMEX — название оси абсцисс;

NX — число литер в названии оси абсцисс;

UX — шаг основного деления на оси абсцисс (задается в математических единицах) (если UX = 0.0, производится автоматический выбор шага);

KX — число дополнительных делений внутри основного на оси абсцисс ( $0 \leq KX \leq 12$ );

NAMEY — название оси ординат;

NY — число литер в названии оси ординат;

UY — шаг основного деления на оси ординат (задается в математических единицах) (если UY = 0.0, производится автоматический выбор шага);

KY — число дополнительных делений внутри основного на оси ординат ( $0 \leq KY \leq 12$ );

M — признак нанесения координатной сетки: M = 00 — сетка не наносится, M = 10 — сетка по оси абсцисс, M = 01 — сетка по оси ординат, M = 11 — сетка по обоим осям.

Итак, программы построения осей позволяют автоматически:

- а) выбрать положение осей;

- б) определить размер литер в числовых и буквенных надписях;

- в) выбрать размер крупного деления;

- г) вычислить масштабный коэффициент МК, который в виде  $*10^{\pm m}$  записывается вслед за названием оси;

- д) определить количество значащих десятичных разрядов (но не более трех) в дробной части чисел, используемых в качестве пометок.

Три последние функции выделены в отдельную служебную программу ASTEP, которая используется не только в программе AXES, но также и в других программах построения осей: XAXIS, YAXIS, XLGAX, YLGAX, RAXES, THAXES, которые будут описаны ниже.

Программа ASTEP(AN, AX, BS, MK, KD) осуществляет коррекцию или автоматический выбор шага. Вычисляются масштабный коэффициент и количество дробных знаков шага. В параметрах AN и AX задаются минимальное и максимальное значения для данной оси. Параметр BS содержит заданное значение шага. В случае корректировки шага программа ASTEP соответствующим образом изменяет это значение. Значениями пара-

метров МК и КД являются соответственно вычисленные значения масштабного коэффициента и числа дробных знаков.

С помощью программ XAXIS, YAXIS можно в области графика провести и разметить произвольное количество осей координат. Положение оси абсцисс (в программе XAXIS) определяется значением математической координаты  $Y$  в пределах области. Аналогично, положение оси ординат (в программе YAXIS) определяется значением математической координаты  $X$  в пределах области.

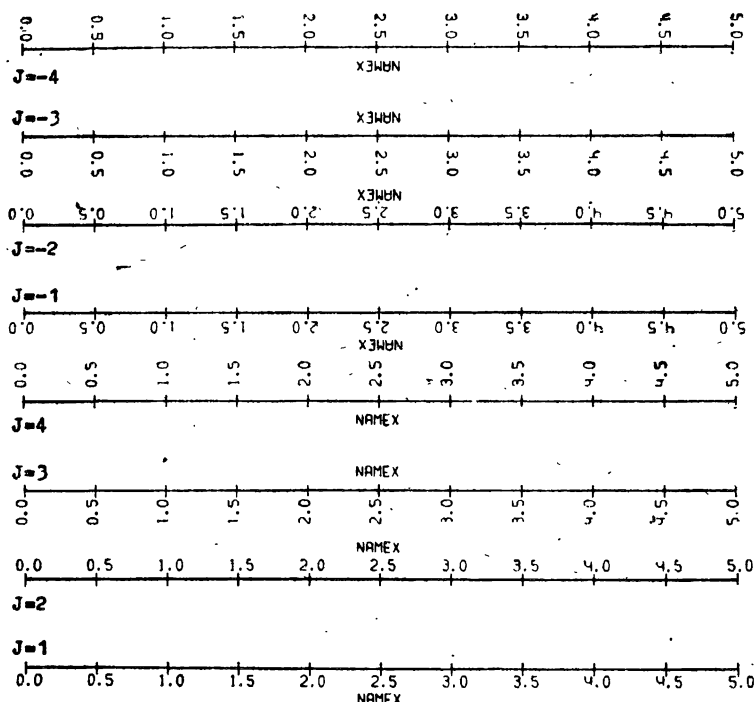


Рис. 4.3. Восемь возможных вариантов расположения пометок и названия оси  $X$ .

Так же как и в программе AXES, оси маркируются с заданным шагом, и на графике фиксируются математические значения, которые соответствуют основным делениям. Через основные деления на осях может быть проведена координатная сетка. Размерные надписи, относящиеся к осям, могут располагаться по одну либо по другую сторону от соответствующей оси как вдоль осей, так и перпендикулярно им. Название оси пишется вдоль оси непосредственно под разметкой (если размерные надписи распо-

гаются вдоль оси) или по другую сторону от оси (если размерные надписи располагаются перпендикулярно оси).

Программа **ХАХИС**(Y0, NAMEX, NX, UX, KX, M, J) позволяет провести и разметить ось абсцисс, т. е. прямую  $Y = Y_0$ . Параметры программы:

**Y0** — математическая координата Y, определяющая положение оси абсцисс;

**NAMEX** — название оси абсцисс;

**NX** — число литер в названии;

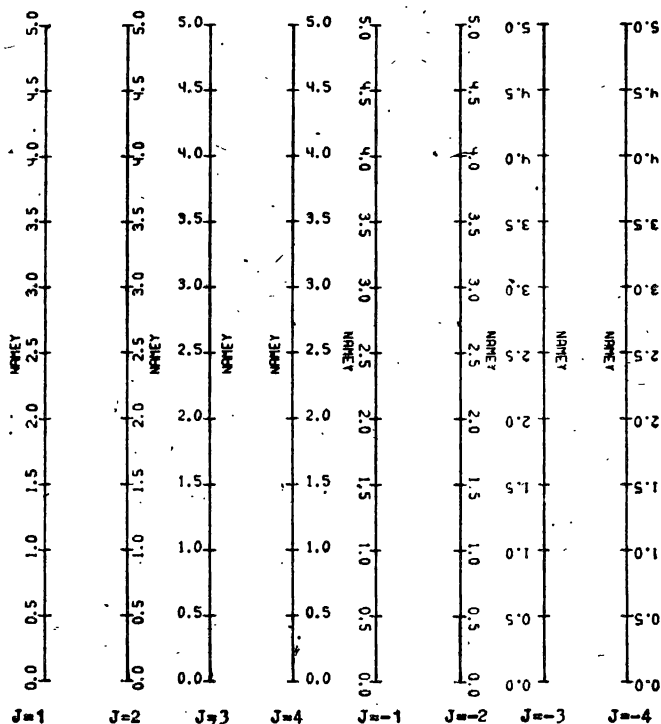


Рис. 4.4. Восемь возможных вариантов расположения пометок и названия оси Y.

**UX** — шаг основного деления на оси (задается в математических единицах) (если  $UX = 0.0$ , производится автоматический выбор шага);

**KX** — число дополнительных делений внутри основного (  $0 \leq KX \leq 12$  );

**M** — признак панесения координатной сетки:  $M = 0$  — без нанесения сетки,  $M = 1$  — с нанесением сетки;



**J** — признак расположения именующих и размерных надписей относительно оси абсцисс; если  $J = 0$ , то размерные надписи отсутствуют, игнорируется также название оси.

Восемь возможных вариантов расположения пометок и названия оси  $X$  (при  $0 < |J| \leq 4$ ) показаны на рис. 4.3.

Программа YAXIS( $X0, NAMEY, NY, UY, KY, M, J$ ) позволяет провести и разметить ось ординат, т. е. прямую  $X = X0$ . Программа имеет следующие параметры:

**X0** — математическая координата  $X$ , определяющая положение оси ординат;

**NAMEY** — название оси ординат;

**NY** — число литер в названии;

**UY** — шаг основного деления на оси (задается в математических единицах) (если  $UY = 0.0$ , производится автоматический выбор шага);

**KY** — число дополнительных делений внутри основного ( $0 \leq KY \leq 12$ );

**M** — признак нанесения координатной сетки:  $M = 0$  — без нанесения сетки,  $M = 1$  — с нанесением сетки;

**J** — признак расположения именующих и размерных надписей относительно оси ординат; если  $J = 0$ , то размерные надписи отсутствуют, игнорируется также название оси.

Восемь возможных вариантов расположения пометок и названия оси  $Y$  (при  $0 < |J| \leq 4$ ) показаны на рис. 4.4.

#### 4.4. Графики в логарифмической и полулогарифмической сетках

При построении графика изменяющейся в широких пределах зависимости логарифмическая шкала имеет преимущество перед равномерной. График функции в этом случае, так же как и при равномерных шкалах, вычерчивается внутри прямоугольной области, задаваемой обращением к программе REGION. Пределы изменения функции и ее аргумента определяются программой LIMITS. Естественно, что предельные значения координаты, относящейся к логарифмической шкале, должны быть положительными. С помощью программы LGLINE можно начертить кривую по заданным массивам точек  $X$  и  $Y$ .

Программа LGLINE( $X, Y, N, LG, NM, JS, L$ ) позволяет по последовательности точек начертить в заданной области кривую в прямоугольной системе координат с логарифмической шкалой. Параметры программы:

**X, Y** — массивы координат точек (длины  $N$ );

**|N|** — количество точек (если  $N < 0$ , допускается выход за пределы области);

LG — признак логарифмической шкалы:  $LG = 1$  — по оси абсцисс,  $LG = -1$  — по оси ординат,  $LG = 0$  — по обоим осям;

$|NM|$  — номер маркера (если  $NM < 0$ , изображается маркер уменьшенных размеров);

$|JS|$  — шаг маркировки:  $JS > 0$  — проводится линия с маркерами,  $JS = 0$  — проводится линия без маркеров,  $JS < 0$  — отмечается маркером каждая  $|JS|$ -я точка, причем линия не проводится;

L — признак замкнутости:  $L = 0$  — незамкнутая линия,  $L = 1$  — замкнутая линия.

Для перевода математических координат в страничные имеется программа TMLGF.

Программа TMLGF(XM, YM, LG, N, XF, YF) позволяет перевести математические координаты точки в страничные с учетом логарифмической сетки. Параметры программы:

XM, YM — математические значения координат X и Y;

LG — признак логарифмической шкалы:  $LG = 1$  — по оси абсцисс,  $LG = -1$  — по оси ординат,  $LG = 0$  — по обоим осям;

N — параметр, определяющий, может ли точка находиться за пределами области:  $N = -1$  — может,  $N = 0$  — точка, оказывающаяся за пределами области, проецируется на границу области;

XF, YF — значения координат X и Y в заданных единицах измерения.

Для проведения осей координат с логарифмической шкалой предназначены программы XLGAX и YLGAX. Оси метятся основными и дополнительными делениями. С основными делениями соотносятся их численные значения. Через основные деления могут быть проведены координатные линии.

Программа XLGAX(Y0, NAMEX, NX, JS, KX, M) позволяет провести и разметить оси абсцисс с логарифмической шкалой. Параметры программы:

Y0 — математическое значение координаты Y в точке, через которую проводится ось абсцисс;

NAMEX — название оси;

NX — количество литер в названии;

JS — определяет шаг  $10^{JS}$ , с которым наносятся основные деления;

$|KX|$  — число дополнительных делений внутри основного ( $|KX| \leq 10$ ):  $KX > 0$  — ось ординат с равномерной шкалой,  $KX \leq 0$  — ось ординат с логарифмической шкалой;

M — признак нанесения координатной сетки по оси абсцисс:  $M = 0$  — без нанесения сетки,  $M = 1$  — с нанесением сетки.

Программа YLGAX(X0, NAMEY, NY, JS, KY, M) позволяет провести и разметить ось ординат с логарифмической шкалой. Параметры программы:

X0 — математическое значение координаты X в точке, через которую проводится ось ординат;

NAMEY — название оси;

NY — количество литер в названии;

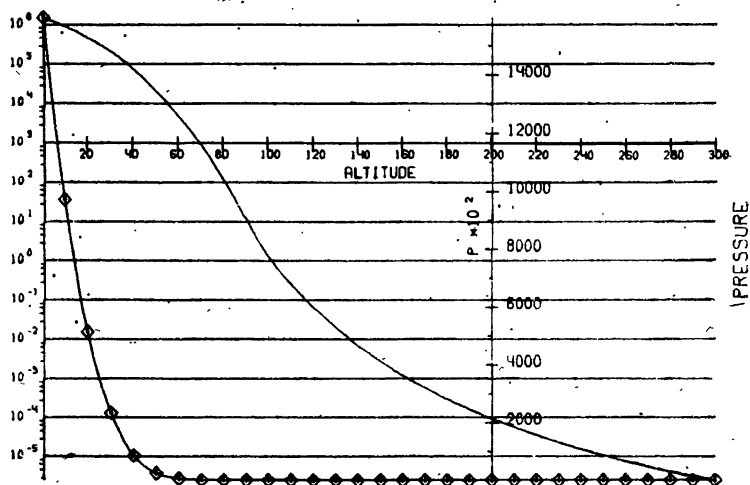
JS — определяет шаг  $10^{JS}$ , с которым наносятся основные деления;

|KY| — число дополнительных делений внутри основного ( $|KY| \leq 10$ ):  $KY > 0$  — ось абсцисс с равномерной шкалой,  $KY \leq 0$  — ось абсцисс с логарифмической шкалой;

M — признак нанесения координатной сетки по оси ординат:  $M = 0$  — без нанесения сетки,  $M = 1$  — с нанесением сетки.

Если используется полулогарифмическая шкала, т. е. одна из осей имеет логарифмическую шкалу, а другая — равномерную, то

ВЫСОТА В КМ. ДАВЛЕНИЕ В НЬЮТОН/К В.М



THIS IS A 1962 US STANDARD VENUS ATMOSPHERE.

Рис. 4.5. График функции с обычной и полулогарифмической шкалами.

ось с равномерной шкалой проводится с помощью подпрограмм XAXIS или YAXIS. Чтобы определить координату точки на оси с логарифмической шкалой, через которую проходит ось с равномерной шкалой, следует воспользоваться подпрограммой-функцией COORD.

Функция COORD(CO,J) позволяет получить математическое значение координаты на оси с равномерной шкалой, соответствующее заданному значению координаты на оси с логарифмической шкалой. Здесь:

CO — математическое значение координаты на логарифмической оси;

J — признак координаты: J = 0 — координата на оси абсцисс, J = 1 — на оси ординат.

Значение, полученное с помощью функции COORD(CO,J), используется при обращении к подпрограммам XAXIS или YAXIS.

На рис. 4.5 приведены два графика, характеризующие изменение давления в атмосфере Венеры в зависимости от высоты, в декартовой и прямоугольной системах координат с логарифмической шкалой по оси ординат. Рисунок получен с помощью следующей программы:

```
DIMENSION P(400), Z(400), CS(400), RHO(400), TM(400)
```

```
DO 2 I = 1, 300
```

```
  Z(I) = 1
```

```
  CALL VENUSA(Z(I), P(I), CS(I), RHO(I), TM(I))
```

```
2 CONTINUE
```

```
  CALL PAGE(20., 22., 0, 0, 0)
```

```
  CALL REGION(1., 3., 18., 12.,
```

```
  * 'ВЫСОТА' В КМ. ДАВЛЕНИЕ В НЬЮТОН/КВ.М', 35, 0)
```

```
  CALL MINMAX(P, 300, AMN, AMX)
```

```
  CALL LIMITS(1., 300., AMN, AMX)
```

```
  CALL YLGAX(Z(1), 'PRESSURE', 8, 0, 5, 1)
```

```
  CN = COORD(1000., 1)
```

```
  CALL XAXIS(CN, 'ALTITUDE', 8, 0., 1, 0, 1)
```

```
  CALL LGLINE(Z, P, 300, -1, 1, 0, 0)
```

```
  CALL LINEMO(Z, P, 300, 3, 10)
```

```
  CALL YAXIS(200., 'P', 1, 0., 2, 0, 4)
```

```
  CALL ENDPG('4.5')
```

```
END
```

#### 4.5. Графики в полярной системе координат

Наряду с декартовыми координатами широко используется полярная система координат. В настоящем параграфе читатель может познакомиться с программами Графора, которые позволяют рисовать графики функций, заданных в этой системе координат.

Напомним некоторые общие положения. Если на плоскости зафиксирована точка  $O$  (полюс) и задана направленная прямая  $OX$  (полярная ось), то с каждой точкой  $P$  плоскости, на которой задана полярная система координат, можно связать пару чисел  $\rho$ ,  $\theta$  (полярные координаты). Полярный радиус  $\rho$  есть длина отрезка  $OP$ , а полярный угол  $\theta$  — величина угла между полярной осью и отрезком  $OP$ , причем положительным направлением отсчета углов считается направление, противоположное вращению ча-

совой стрелки. Полярные координаты определяются однозначно с точностью до изменения  $\theta$  на  $360^\circ \times n$ , где  $n$  — любое целое число.

Если на этой же плоскости выбрана система прямоугольных координат с началом в точке  $O$  так, что положительное направление оси  $X$  совпадает с полярной осью, то, зная полярные координаты точки, можно легко найти значения декартовых координат.

Для этой цели применяются общеизвестные формулы преобразования:

$$x = \rho \cos \theta, \quad y = \rho \sin \theta.$$

При построении графиков в полярной системе координат сохраняются правила и соглашения, принятые для декартовой прямоугольной системы координат. Итак, в пределах страницы можно определить полярную область, в которой будут рисоваться графики. В общем случае полярная область имеет форму кольцевого

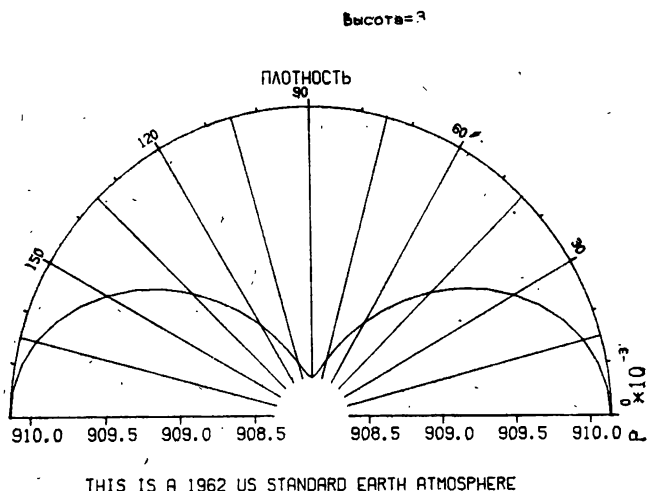


Рис. 4.6. Пример графика в полярной системе координат.

сектора. Задается полярная область обращением к программе POLREG, являющейся аналогом программы REGION в прямоугольной системе координат. Определение полярной области обязательно.

На рис. 4.6 приведен пример графика в полярной системе координат. Он описывает зависимость плотности атмосферы Земли от широты на некоторой заданной высоте.

Программа POLREG (X, Y, RIN, RFN, THO, THF, NAME, N, H, J) позволяет определить на заданной странице полярную область для построения графиков. Параметры программы:

X, Y — координаты центра области в выбранных единицах измерения;

RIN, RFN — внутренний и внешний радиусы в выбранных единицах измерения,  $RFN - RIN \geq 0.5$  см;

THO, THF — минимальный и максимальный углы, определяющие радиальные граничные значения области (углы задаются в градусах),  $360^\circ \geq THF - THO \geq 5^\circ$ ;

NAME — название области;

|N| — количество литер в названии (если  $N > 0$ , название области выписывается вдоль оси Y, если  $N < 0$  — вдоль оси X);

H — высота литеры (если  $H \leq 0$ , высота принимается равной 0.5 см);

J — признак очерчивания границ области:  $J = 0$  — границы не очерчиваются,  $J = 1$  — очерчиваются.

Для задания математического пространства используется программа LIMITS. Однако параметры этой программы имеют теперь другой смысл.

Программа LIMITS(RIN, RFN, THO, THF) позволяет зафиксировать пределы изменения функции в области рисования. Параметры программы:

RIN — математическое значение радиуса, соответствующее внутренней круговой границе области;

RFN — математическое значение радиуса, соответствующее внешней круговой границе области;

THO — математическое значение угла, соответствующее начальному радиальному граничному значению области;

THF — математическое значение угла, соответствующее конечному радиальному граничному значению области;

Как и при работе в прямоугольной системе координат, для определения пределов изменения функции и ее аргумента можно воспользоваться программой MINMAX.

Построение графиков в полярной области осуществляется с помощью программы POLINE. Кроме того, имеется возможность построить полярные оси координат и, если необходимо, провести координатные линии (координатную сетку).

Программа POLINE(R, TH, N, KS, NM, JS, L) позволяет в заданной полярной области начертить кривую. Параметры программы:

R — массив радиусов;

TH — массив углов (в градусах или радианах в зависимости от KS);

N — количество точек (при  $N < 0$  кривая может выходить за пределы области);

|KS| — шаг по массиву данных (если  $KS > 0$ , углы задаются в градусах, если  $KS < 0$  — в радианах);

$|NM|$  — номер маркера (если  $NM < 0$ , изображается маркер уменьшенных размеров);

$|JS|$  — шаг маркировки (если  $JS > 0$ , линия с маркерами, если  $JS = 0$  — без маркеров, если  $JS < 0$  — только маркеры);

$L$  — признак замкнутости:  $L = 0$  — незамкнутая кривая,  $L = 1$  — замкнутая кривая.

Провести и разметить в полярной области радиальные и угловые оси координат можно с помощью программ RAXES и THAXES. Оси маркируются с заданным шагом и математическое значение, соответствующее основному делению, фиксируется на графике. Величина шага задается в математических единицах или выбирается автоматически. Разрешается не более десяти основных делений на радиальной оси и до 72 основных делений на угловой оси. Допускается не более двенадцати вспомогательных делений внутри основного.

Численное значение, приписываемое основным делениям, может состоять из шести целых и трех дробных знаков. При необходимости проводится масштабирование и указывается масштабный коэффициент вида  $\cdot 10^{\pm n}$  вслед за названием оси. Для перевода математических координат точки в полярной области в координаты на странице имеется программа TPF.

Программа RAXES (NAME, N, II, UR, KR, MR, T) позволяет провести и разметить радиальную ось. Параметры программы:

NAME — название оси;

$|N|$  — количество литер в названии:  $N > 0$  — название оси пишется справа от области вдоль оси Y,  $N < 0$  — название оси пишется внизу;

$|H|$  — высота литер в названии (если  $H = 0$ , высота полагается равной 0,4 см, если  $H < 0$ , ось не проводится);

UR — размер основного деления на радиальной оси (если  $UR = 0$ , то производится автоматический выбор шага);

$|KR|$  — количество вспомогательных делений внутри основного (если  $KR < 0$ , математические значения не подписываются);

MR — признак проведения радиальных координатных линий;

MR = 0 — координатные линии не проводятся,

MR = 1 — через основные деления проводятся дуги окружностей,

MR = 2 — через основные и вспомогательные деления проводятся дуги окружностей (через вспомогательные — штриховой линией);

T — математическое значение угла (в градусах), под которым проводится ось.

Программа THAXES (NAME, N, H, UT, KT, MT, R) позволяет провести и разметить угловую ось. Параметры программы:

NAME — название оси;

$|N|$  — количество литер в названии (если  $N > 0$ , название описывается справа от области вдоль оси  $Y$ , если  $N < 0$  — внизу);

$|H|$  — высота литер в названии (если  $H = 0$ , высота полагается равной 0.4 см, если  $H < 0$ , ось не проводится);

$UT$  — размер основного деления по угловой оси (в градусах) (если  $UT = 0$ , производится автоматический выбор шага);

$|KT|$  — количество вспомогательных делений внутри основного (если  $KT < 0$ , математические значения не подписываются);

$MT$  — признак проведения угловых координатных линий;

$MT = 0$  — координатные линии не проводятся,

$MT = 1$  — координатные линии проводятся через основные деления,

$MT > 1$  — внутри основного деления проводится  $MT - 1$  угловых координатных линий;

$R$  — математическое значение радиуса, с которым проводится ось.

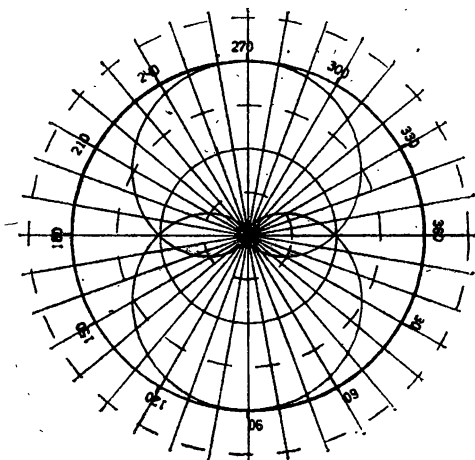


Рис. 4.7. Две кардиоды в полярной системе координат.

Программа  $TPF(RM, TM, KN, RF, TF, X, Y)$  позволяет перевести математические координаты точки в полярной области в координаты на странице. Параметры программы:

$RM$  — математическое значение радиуса;

$TM$  — математическое значение угла;

$KN$  — признак угла:  $KN = \pm 1$  — угол задан в градусах,  $KN = \pm 2$  — угол задан в радианах,  $KN < 0$  — точка может находиться за пределами области,  $KN > 0$  — точка, попадающая за пределы области, проецируется на границу области;

$RF$  — расстояние точки от центра области в выбранных единицах измерения;



TF — угол между полярным радиусом и осью X (в градусах);

X, Y — координаты точки на странице.

Здесь выходные параметры: RF, TF, X, Y.

В программах RAXES и THAXES используется служебная программа SIGNA.

На рис. 4.7 показаны две кардиоиды, симметричные относительно оси X. Ниже приводится программа, с помощью которой выполнен этот рисунок.

```
DIMENSION R(180), T(180)
DO 1 I=1, 180
  T(I)=2.*I-2.
1 R(I)=1.+SIN(T(I)*.01745329)
CALL PAGE(15., 15., 0, 0, 0)
CALL POLREG(7.5, 7.5, 0., 6., 0., 360., 0, 0, 0., 0)
CALL LIMITS(0., 2.6, 0., 360.)
CALL POLINE(R, T, 180, -1, 1, 0, 1)
CALL LIMITS(0., 2.6, 360., 0.)
CALL POLINE(R, T, 180, -1, 1, 0, 1)
CALL THAXES('HEARTS', -6, .35, 30., 1, 3, 2.)
CALL RAXES(0, 0, 0., 1., -2, 2, 0.)
CALL ENDPG('4.7')
END
```

## 4.6. Гистограммы и календарная ось

Читатель, несомненно, встречался с таким способом представления информации, как *гистограмма* или столбчатая диаграмма. Это один из видов графического изображения статистических распределений по количественному признаку. Гистограмма представляет собой совокупность смежных прямоугольников (столбиков), построенных на одной прямой линии. Высота каждого прямоугольника пропорциональна частоте нахождения данной величины в изучаемой совокупности.

В Графоре предусмотрено два слегка отличающихся способа построения гистограмм (программы HISTGM и BARS). Программа BARS позволяет одновременно строить две связанные зависимости (например, выпуск автомобилей, в том числе легковых). При этом одна из зависимостей, а именно та, которая представляет собой часть другой, выделяется штриховкой. Того же можно добиться и с помощью программы HISTGM, но за два обращения к ней. В программе HISTGM гистограмма строится относительно заданного базисного уровня (например, за 100% принят уровень 1940 г.).

В-связи с гистограммами довольно часто используется так называемая *календарная ось*. Программа AXISC позволяет построить

такую ось, на которой пометками являются сокращенные (трехбуквенные) названия месяцев. Календарная ось может быть проведена как по нижней, так и по верхней границам области.

В Графоре принято определять место графика на странице и пределы изменения аргумента и функции в обращениях к программам REGION и LIMITS. Эти соглашения полностью применимы и к программам, которые описаны в настоящем параграфе. Некоторым исключением является программа AXISC, при вызове которой указывается информация, заменяющая пределы.

Программа AXISC(NAME, NC, MMIN, MS, NM) позволяет провести и разметить календарную ось. Параметры программы:

NAME — название оси (обычно указывается год);

NC — количество литер в названии;

MMIN — номер месяца для первого деления оси;

MS — шаг по месяцам;

|NM| — количество делений на календарной оси (если  $NM > 0$ , ось проводится по нижней границе области, если  $NM < 0$  — по верхней, если  $NM = 0$ , ось не рисуется).

Размер каждой буквы в названиях месяцев зависит от фактической ширины деления и выбирается в пределах от 0.1 см до 0.4 см. Высота литеры в названии оси равна 0.5 см.

Программа HISTGM(XS, DX, Y0, YV, N, NP1) позволяет построить гистограмму относительно заданного базисного уровня. Ее параметры:

XS — абсцисса первого прямоугольника;

DX — ширина одного столбика гистограммы;

Y0 — базисный уровень, относительно которого строится гистограмма;

YV — вектор значений функции длины N;

|N| — количество столбиков в гистограмме (если  $N > 0$ , штриховка отсутствует, если  $N < 0$ , штриховка слева направо);

NP1 — количество линий штриховки на 1 см.

Высота каждого прямоугольника равна  $YV(I) - Y0$ , где  $I = 1, 2, \dots, N$ . На высоте Y0 проводится базисная линия.

Программа BARS(Y, YP, N, W, INAT, NP1) позволяет построить гистограмму для двух зависимостей. Параметры программы:

Y — вектор значений функции длины N;

YP — вектор значений функции (часть от Y) длины N;

N — количество столбиков в гистограмме;

W — ширина одного столбика гистограммы;

INAT — признак штриховки для столбиков, определяемых YP: INAT = 1 — штриховка отсутствует, INAT = 2 — штриховка слева направо, INAT = 3 — штриховка справа налево, INAT = 4 — штриховка в обоих направлениях;

NP1 — количество линий штриховки на 1 см.

Программа BAR(X, Y, H, W, SH, INAT, NP1) позволяет построить один столбик гистограммы и заштриховать его часть. Параметры программы:

X, Y — математические координаты левого нижнего угла прямоугольника;

H — высота прямоугольника;

W — ширина прямоугольника;

SH — высота части прямоугольника, подлежащей штриховке;

INAT — признак штриховки: INAT = 1 — штриховка отсутствует, INAT = 2 — штриховка слева направо, INAT = 3 — штриховка справа налево, INAT = 4 — штриховка в обоих направлениях;

NP1 — количество линий штриховки на 1 см.

На рис. 4.8 иллюстрируются возможности программ AXISC и HISTGM. Гистограммы показывают так называемую наработку на

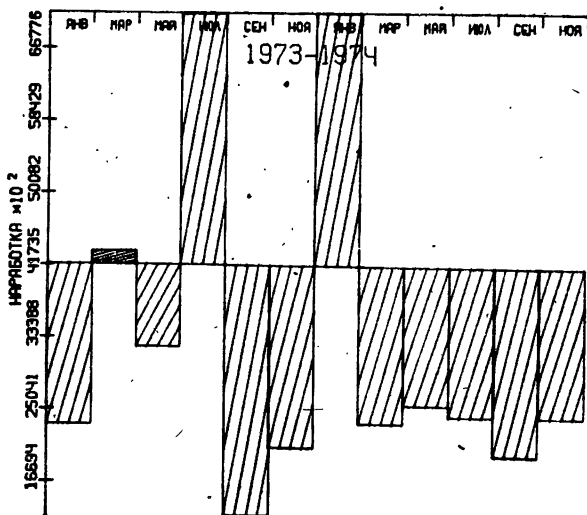


Рис. 4.8. Пример, иллюстрирующий работу программы HISTGM.

магнитофонах БЭСМ-6. (Использовалась статистическая информация, собираемая ОС Диспак.) Рис. 4.8 получен как результат работы следующей программы:

```
DIMENSION YV(12)
```

```
C... МАССИВ YV ВВЕДЕН С КАРТ
```

```
CALL MINMAX(YV, 12, YMN, YMX)
```

```
Y0 = (YMN + YMX) / 2
```

```
SM = (YMN + YMX) / 10
```

```

CALL PAGE(15., 15., 'PAGE', 4., 0)
CALL REGION(1., 0., 12., 11., 0, 0, 0)
CALL LIMITS(0., 12., YMN, YMX)
CALL YAXIS(0., 'НАРАБОТКА', 9, SM, 0, 0, 1)
CALL AXISC('1973-1974', 9, 1, 2, -12)
CALL HISTGM(0., 1., Y0, YV, -12, 4)
CALL ENDPG('4.8')
END

```

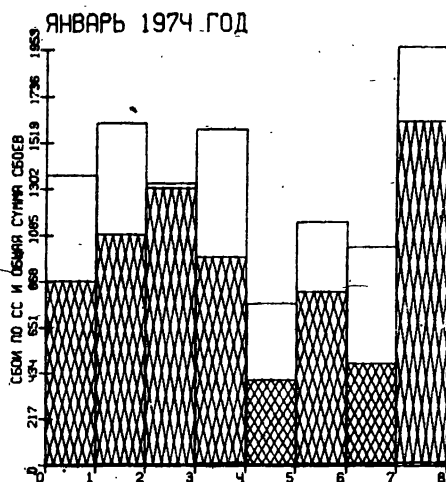


Рис. 4.9. Изображение гистограмм, построенных с помощью программы BARS.

На рис. 4.9 исследуется характер сбоев на магнитофонах БЭСМ-6. Гистограммы строятся программой BARS.

## 5. АППРОКСИМАЦИЯ И СГЛАЖИВАНИЕ ФУНКЦИЙ

Не всегда можно удовлетвориться прямым выводом информации на графическое устройство. Во многих случаях предварительная ее обработка (восполнение, приближение, сглаживание) позволяет выделить наиболее существенное, улучшить не только математическое, но и эстетическое представление информации. Важную роль методы аппроксимации играют и при анализе функций.

Современные устройства графического ввода и вывода в совокупности с быстродействующими ЭВМ позволяют математику не только видеть результаты, но и оперативно вмешиваться в процесс вычислений, выбирая другие узлы, меняя класс приближающих функций, определяя критерий согласия, контролируя точность приближения. На практике проблема приближения функций возникает во многих случаях, в частности при вводе в машину функциональных зависимостей, полученных в результате эксперимента, при выборе представления таких зависимостей в памяти ЭВМ, при выводе информации на графические устройства.

В системах автоматизации проектирования и производства методы аппроксимации применяются прежде всего в целях конструирования кривых и поверхностей. При построении кривой в этих случаях утрачивает (или почти утрачивает) смысл такой математический критерий, как точность аппроксимации, и главную роль начинают играть такие критерии, как внешний вид и гладкость кривой, отсутствие осцилляций и т. п.

В этой главе мы рассмотрим реализацию нескольких включенных в Графор методов аппроксимации и восполнения функций. Эти методы хорошо известны в математике, и поэтому их описание приводится лишь постольку, поскольку оно необходимо для понимания смысла и структуры программ.

### 5.1. Сплайн-аппроксимация

Математические *сплайны* берут свое начало от тонких гибких стержней, которыми пользовались чертежники для проведения плавных кривых через заданные точки. Стержень закреплялся в точках  $(x_i, y_i)$  и принимал форму кривой  $y(x)$  с минимальной «энергией натяжения», пропорциональной  $\int y''^2 dx$ .

Если перейти к математическому описанию сплайна, то *сплайн-функцией* степени  $k$  с точками соединения  $x_0 < x_1 < \dots < x_n$  будет функция  $y(x)$ , которая на отрезке  $[x_0, x_n]$  имеет непрерывные производные до  $k - 1$  включительно и на каждом из отрезков  $[x_{i-1}, x_i]$  равна многочлену степени  $k$ . Далее нас будут интересовать лишь кубические сплайны ( $k = 3$ ). Такой сплайн обеспечивает совпадение в узлах с исходной функцией и непрерывность первой и второй производных в точках соединения.

Прежде всего определяются первые производные во всех точках соединения. Решается трехдиагональная система  $n - 1$  уравнений с доминирующей главной диагональю. Она может быть решена, если задать два крайних условия. Программы, включенные в Графор, позволяют задавать четыре типа крайних условий:

а) известны значения первых производных на концах сетки  $(y'_0, y'_n)$ ;

б) известны значения вторых производных на концах сетки  $(y''_0, y''_n)$ ;

в) при построении периодического сплайна значения функции в первой и последней точках совпадают, а также  $y'_0 = y'_n$  и  $y''_0 = y''_n$ ;

г) если крайние условия не заданы, то считается, что  $y'_0 = y'_n = 0$ .

После того как построена трехдиагональная матрица (или дополненная трехдиагональная матрица в случае в)) и вектор свободных членов, система уравнений может быть решена. Для решения такой системы существует эффективный алгоритм, который в отечественной литературе называется методом прогонки. В результате решения системы уравнений получается вектор первых производных  $y'_i$  в точках соединения.

Теперь значение  $y(x)$  для  $x_{i-1} \leq x \leq x_i$  определяется из многочлена

$$y(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \quad (5.1)$$

коэффициенты которого легко найти, поскольку известны значения функции и первые производные в точках соединения.

Таким образом, метод позволяет выделить ряд самостоятельных функций:

— формирование трехдиагональной матрицы и вектора свободных членов в соответствии с заданными крайними условиями — подпрограмма TDMP,

— решение системы уравнений, заданных матрицей и вектором свободных членов (метод прогонки) — подпрограмма TRIDIG,

— вычисление коэффициентов кубических многочленов для всех отрезков — подпрограмма SPLINE,

— вычисление значений сплайна на заданной равномерной сетке — подпрограмма SPLINT.

Отметим, что программа SPLINE при своей работе использует подпрограммы TDMP и TRIDIG. Эти программы универсальны и их применение не ограничено сплайн-аппроксимацией. Их универсальность имеет и более глубокий смысл. В частности, коэффициенты для кубических парабол можно вычислить, задав вектор производных полностью. Это означает, что программист по своему усмотрению может модифицировать вектор производных. При интерактивной работе, таким образом, можно подбирать подходящую форму кривой. Математическое обоснование сплайнов и их анализ изложены в [11].

Программа SPLINE(X, Y, U, N, A, B, C, D, CODE, IER) вычисляет коэффициенты кубического сплайна. Попутно вычисляются (если не заданы) значения первых производных в точках соединения. Программа имеет следующие параметры:

X, Y, U — векторы значений аргумента, функции и первых производных (длины N);

N — количество точек;

A, B, C — векторы коэффициентов кубического многочлена при переменных  $x^3$ ,  $x^2$  и  $x$  (длины N);

D — вектор свободных членов кубического многочлена (длины N);

CODE — признак задания краевых условий:

CODE = -2 — на концах сетки заданы вторые производные; перед вызовом программы  $y_1''$  и  $y_n''$  должны быть занесены соответственно в D(1) и D(N),

CODE = -1 — на концах сетки заданы первые производные; перед вызовом программы  $y_1'$  и  $y_n'$  должны быть занесены соответственно в D(1) и D(N),

CODE = 0 — краевые условия не заданы, что равносильно  $y_1'' = y_n'' = 0$ ,

CODE = 1 — задан периодический сплайн, т. е.  $y_1' = y_n'$  и  $y_1'' = y_n''$ ,

CODE = 2 — вектор значений производных U задан полностью;

IER — код ошибки; этот код для программы SPLINE является транзитным и передается в том виде, в каком он получен в подпрограмме TRIDIG.

Коэффициенты A(I), B(I), C(I) и D(I) соответствуют отрезку от X(I) до X(I + 1),  $I = 1, \dots, N - 1$ .

Программа SPLINT(X, N, A, B, C, D, Y, M) позволяет вычислить значения кубического сплайна на равномерной сетке с заданным шагом. Параметры программы:

X — вектор значений аргумента (длины N);

N — количество точек;

A, B, C — векторы коэффициентов кубического сплайна при переменных  $x^3$ ,  $x^2$ , и  $x$  (длины N);

D — вектор свободных членов кубического сплайна (длины N);  
 Y — вектор значений кубического сплайна (длины M);  
 M — количество узлов равномерной сетки на отрезке  $[X(1), X(N)]$ .

На рис. 5.1 построен сплайн для функций  $\sin x$  и  $\cos x$ , значения которых определены на сетке с шагом  $\pi/6$ , но для функций

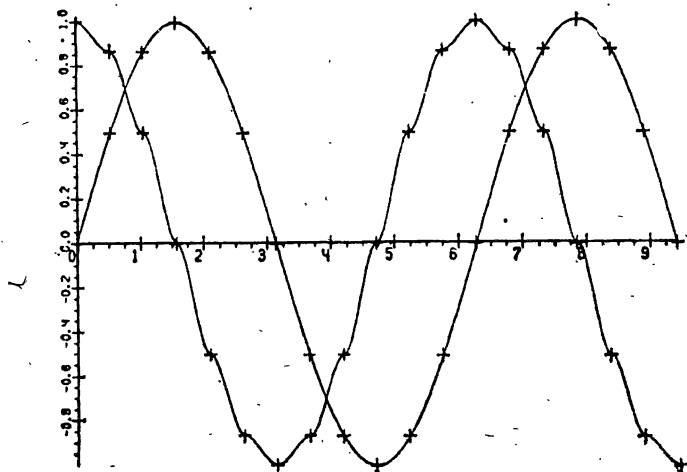


Рис. 5.1. Пример построения сплайнов для функций  $\sin x$  и  $\cos x$ .

$\cos x$  значения первых производных в узлах заданы равными нулю. Пример показывает, как, управляя производными, можно изменить форму кривой. Рисунок получен с помощью следующей программы:

```

DIMENSION X(20), Y(20), U(20)
DIMENSION A(20), B(20), C(20), D(20), YM(200)
DX=3.141593/6
X(1)=0.
DO 1 I=1, 19
  Y(I)=SIN(X(I))
1 X(I+1)=X(I)+DX
CALL PAGE(17., 26., 0, 0, 0)
CALL LIMITS(0., X(19), -1., 1.)
CALL REGION(1.5, 14., 14., 10., 0, 0, 0)
CALL LINEMO(X, Y, 19, 1, -1)
CALL SPLINE(X, Y, U, 19, A, B, C, D, 0, 1ER)
CALL SPLINT(X, 19, A, B, C, D, YM, 200)
DX=(X(19)-X(1))/19.
CALL INCLIN(X(1), DX, 0, YM, 200, 0, 0)
DO 2 I=1, 19
  
```



2 Y(I)=0.

CALL SPLINE(X, U, Y, 19, A, B, C, D, 2, IER)

CALL SPLINT(X, 19, A, B, C, D, YM, 200)

CALL INCLIN(X(1), X(19), 1, YM, 200, 0, 0)

CALL AXES(0, 0, 0., 4, 0, 0, 0., 4, 0)

CALL ENDPG('5.1')

END

Программа TDMP(X, Y, N, A, B, C, D, KODE) позволяет вычислить элементы трехдиагональной матрицы или дополненной трехдиагональной матрицы (в случае периодического сплайна) и вектор свободных членов. Программа имеет следующие параметры:

X, Y — векторы значений аргумента и функции (длины N);

N — количество точек;

A, B, C — поддиагональный, диагональный и наддиагональный векторы матрицы (длины N);

D — вектор свободных членов (длины N);

KODE — признак задания краевых условий:

KODE = -2 — на концах сетки заданы вторые производные; перед вызовом программы  $y_1''$  и  $y_n''$  должны быть занесены соответственно в D(1) и D(N),

KODE = -1 — на концах сетки заданы первые производные, перед вызовом программы  $y_1'$  и  $y_n'$  должны быть занесены соответственно в D(1) и D(N),

KODE = 0 — краевые условия не заданы, что равносильно  $y_1'' = y_n'' = 0$ ,

KODE = 1 — задан периодический сплайн, т. е.  $y_1' = y_n'$  и  $y_1'' = y_n''$ .

Программа TRIDIG(U, N, A, B, C, D, KODE, IER) позволяет найти решение системы уравнений, заданной трехдиагональной матрицей или дополненной трехдиагональной матрицей, методом прогонки. В частности, для кубического сплайна решением системы является вектор первых производных в точках соединения. Параметры программы:

U — вектор результатов;

N — количество уравнений;

A, B, C — поддиагональный, диагональный и наддиагональный векторы матрицы;

D — вектор свободных членов;

KODE — характеристика матрицы: KODE = 0 — трехдиагональная матрица, KODE = 1 — дополненная трехдиагональная матрица;

IER — код ошибки: IER = 0 — ошибки нет, IER = 1 — если B(1) = 0, IER = 2 — если B(J) + C(J) \* A(J-1) = 0.

З а м е ч а н и е. Способ задания матриц.

а) Трехдиагональная матрица (KODE = 0):

$$\begin{bmatrix} r_{11} & r_{12} & 0 & \dots & 0 & 0 & 0 \\ r_{21} & r_{22} & r_{23} & \dots & 0 & 0 & 0 \\ 0 & r_{32} & r_{33} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & r_{n-2,n-2} & r_{n-2,n-1} & 0 \\ 0 & 0 & 0 & \dots & r_{n-1,n-2} & r_{n-1,n-1} & r_{n-1,n} \\ 0 & 0 & 0 & \dots & 0 & r_{n,n-1} & r_{n,n} \end{bmatrix}$$

задается следующим образом:

$A(1) = 0$ ,  $A(2) = r_{21}$ , ...,  $A(N) = r_{n,n-1}$  — поддиагональный вектор,

$B(1) = r_{11}$ ,  $B(2) = r_{22}$ , ...,  $B(N) = r_{n,n}$  — диагональный вектор,

$C(1) = r_{12}$ ,  $C(2) = r_{23}$ , ...,  $C(N) = 0$  — наддиагональный вектор;

б) дополненная трехдиагональная матрица ( $KODE = 1$ );

$$\begin{bmatrix} r_{11} & r_{12} & 0 & \dots & 0 & 0 & r_{1,n} \\ r_{21} & r_{22} & r_{23} & \dots & 0 & 0 & 0 \\ 0 & r_{32} & r_{33} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & r_{n-2,n-2} & r_{n-2,n-1} & 0 \\ 0 & 0 & 0 & \dots & r_{n-1,n-2} & r_{n-1,n-1} & r_{n-1,n} \\ 0 & r_{n,2} & 0 & \dots & 0 & r_{n,n-1} & r_{n,n} \end{bmatrix}$$

задается следующим образом:

$A(1) = r_{1,n}$ ,  $A(2) = r_{21}$ , ...,  $A(N) = r_{n,n-1}$  — поддиагональный вектор,

$B(1) = r_{11}$ ,  $B(2) = r_{22}$ , ...,  $B(N) = r_{n,n}$  — диагональный вектор,

$C(1) = r_{12}$ ,  $C(2) = r_{23}$ , ...,  $C(N) = r_{n,2}$  — наддиагональный вектор.

## 5.2. Параметрическое задание функций

Для вычисления сплайн-функции, заданной на сетке  $x_1 < x_2 < \dots < x_n$ , требуется упорядоченная монотонно возрастающая последовательность  $x_i$ . Это означает, что сплайн-функция  $y = y(x)$  может быть построена только для однозначной функции.

Однако, используя параметрическое задание функции, можно построить неоднозначную, в том числе замкнутую, кривую (периодический сплайн). Для этого выбирается независимая переменная  $t$ , удовлетворяющая указанным выше требованиям, и вычисляются сплайн-приближения  $x(t)$  и  $y(t)$ , а затем строится кривая, проходящая через точки  $(x_i, y_i)$ , соответствующие выбранным  $t_i$ .

Предлагается два способа задания параметра  $t$ . В простейшем случае  $t$  задается как последовательность целых чисел:  $t_j = j$ . Другая возможность состоит в том, что  $t_j$  соответствует суммарной длине хорд, которая является аппроксимацией длины дуги между первой и  $j$ -й точками. В этом случае

$$t_1 = 1, \quad t_2 = 2 \quad \text{и} \quad t_j = t_{j-1} + \sqrt{\frac{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}{(x_2 - x_1)^2 + (y_2 - y_1)^2}}. \quad (5.2)$$

Вычислив приближенно длины отрезков получившейся кривой  $x = x(t)$ ,  $y = y(t)$ , можно построить более точные сплайны. Однако это не приводит к заметному изменению самой кривой [11].



Рис. 5.2. Сплайны для таблично заданных неоднозначных функций: а) исходное задание (около 100 точек); б) гладкое восполнение (около 3000 точек); в) изменены вычисленные производные в точках соединения.

В комплекс Графтор включена программа TCALC, которая позволяет вычислить вектор независимой переменной  $t$ , используя для этого соотношение (5.2).

Следует заметить, что параметрическое представление можно использовать не только при построении сплайнов. С тем же успехом им можно воспользоваться и при других способах аппроксимации, описанных в других параграфах этой главы.

Программа TCALC( $X$ ,  $Y$ ,  $T$ ,  $N$ ) позволяет вычислить вектор  $T$  для параметрического задания функции  $x = x(t)$ ,  $y = y(t)$ . Значение  $t_j$  определяется как приведенное расстояние от точки  $(x_1, y_1)$  до точки  $(x_j, y_j)$ , причем  $t_1 = 1$ ,  $t_2 = 2$ . Параметры программы следующие:

- $X$  — вектор значений аргумента;
- $Y$  — вектор значений функции;
- $T$  — вектор значений параметра;
- $N$  — число точек.

На рис. 5.2 показан пример построения сплайнов с использованием параметрического представления кривых. Исходное задание рисунка содержит около 100 точек. На рис. 5.2, а показано преобра-

жение, построенное путем соединения этих точек отрезками прямых. Рис. 5.2, б построен с применением сплайн-интерполяции. На рис. 5.2, в показано преобразование изображения: изменены численные производные в точках соединения. Ниже приведен фрагмент программы, который применялся при построении рисунка. Здесь X, Y — массивы, содержащие узлы участков ломаных, а XN, YN — массивы, в которые заносятся значения гладких кривых, соответствующие этим ломаным.

```

      DIMENSION X(30), Y(30), A(30), B(30), C(30),
* D(30), T(30)
      DIMENSION UX(30), UY(30), XN(150), YN(150)
      . . . . .
      GOTO (10, 20, 30) L
10 CALL LINEO(X, Y, M)
   GOTO 5
20 CALL TCALC(X, Y, T, M)
   CALL SPLINE(T, X, UX, M, A, B, C, D, 0, IER)
   CALL SPLINT(T, M, A, B, C, D, XN, N)
   CALL SPLINE(T, Y, UY, M, A, B, C, D, 0, IER)
   CALL SPLINT(T, M, A, B, C, D, YN, N)
   CALL LINEO(XN, YN, N)
   GOTO 5
30 CALL TCALC(X, Y, T, M)
   CALL SPLINE(T, X, UX, M, A, B, C, D, 0, IER)
   DO 8 I=1, M
8  UX(I)=FX*UX(I)
   CALL SPLINE(T, X, UX, M, A, B, C, D, 2, IER)
   CALL SPLINT(T, M, A, B, C, D, XN, N)
   CALL SPLINE(T, Y, UY, M, A, B, C, D, 0, IER)
   DO 9 I=1, M
9  UY(I) = FY*UY(I)
   CALL SPLINE(T, Y, UY, M, A, B, C, D, 2, IER)
   CALL SPLINT(T, M, A, B, C, D, YN, N)
   CALL LINEO(XN, YN, N)
   GOTO 5

```

### 5.3. Локальные сплайны

В § 5.1 проведение локальной кривой основывалось на аналогии длинного гибкого стержня, т. е. сплайн строился с учетом положения всех  $n$  точек, и поэтому требовалось решение  $n$  уравнений. Было отмечено, что, например, влияние краевых условий быстро затухает и сказывается лишь на ближайших к концам интервалах. На практике чертежники используют этот факт, применяя

лекала — фигурные линейки для проведения плавных кривых. Так, когда строится участок кривой, то учитывается положение лишь четырех-пяти точек, ближайших к этому участку. Построенный таким образом сплайн будем называть *локальным сплайном*.

Непрерывность второй производной имеет значение при проектировании контура механического инструмента (например, кулачка), но не существенна для графического представления кривой. Поэтому можно заметно упростить алгоритм вычисления первых производных в точках соединения. При построении локального сплайна будем определять производные в точках соединения по 3, 4 или 5 смежным точкам, пользуясь известными интерполяционными формулами Ньютона и Стирлинга для равномерной сетки. Вычисление по этим формулам выполняет подпрограмма-функция DERIV5.

Функция DERIV5 (DX, Y, N, I) позволяет вычислить приближенное значение производной в  $i$ -м узле для функции, заданной на сетке с постоянным шагом. Параметры функции задают:

DX — размер постоянного шага сетки;

Y — вектор значений исходной функции;

N — число точек;

I — номер точки, в которой вычисляется производная.

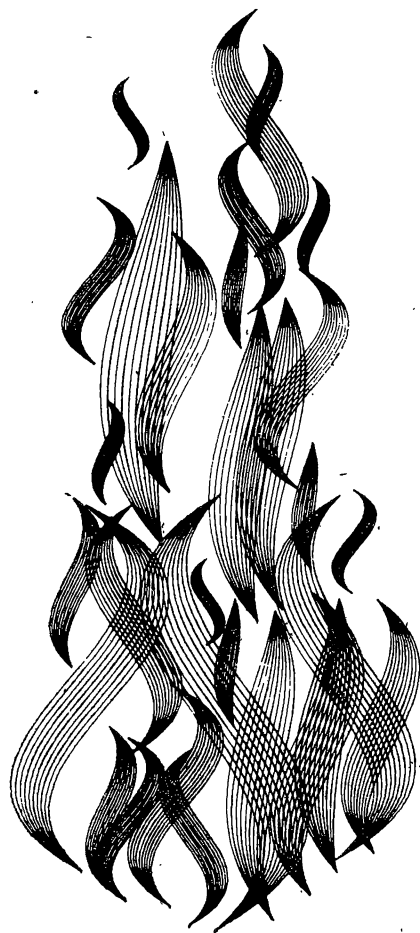


Рис. 5.3. Кубические параболы.

Таким образом, для вычисления первых производных в точках соединения не требуется решать систему уравнений, как это было в случае полного сплайна. Более того, не требуется знать

производные во всех точках одновременно. Как только вычислены производные на концах участка, подпрограмма CUBPOL определяет коэффициенты кубического многочлена для этого участка, и они немедленно используются для рисования части кривой.

Программа CUBPOL( $X_1, X_2, Y_1, Y_2, DY_1, DY_2, A$ ) позволяет вычислить коэффициенты кубического многочлена  $y(x) = a_1 + a_2x + a_3x^2 + a_4x^3$ . Программа имеет следующие параметры:

$X_1, X_2$  — значения аргумента на левом и правом концах отрезка;

$Y_1, Y_2$  — значения функции на левом и правом концах отрезка;

$DY_1, DY_2$  — производные в узлах  $X_1, X_2$ ;

$A$  — вектор коэффициентов кубического многочлена, причем  $A(1) = a_1, A(2) = a_2, A(3) = a_3, A(4) = a_4$ .

Внешне, с точки зрения программиста, рисование лекальной кривой ничуть не сложнее, чем рисование ломаной. Достаточно обратиться к программе SINCL, которая с помощью подпрограммы DERIV5 вычисляет производные, затем, обращаясь к CUBPOL, вычисляет коэффициенты кубического многочлена, и, наконец, строит участок кривой так, чтобы длина составляющих его отрезков прямых не превышала 1,5 — 2 мм.

При построении локального сплайна экономней используется как время, так и пространство в памяти машины. Тем не менее кривая, которая при этом получается, практически не отличается от кривой, построенной как полный сплайн.

Программа SINCL( $XBEG, DX, Y, N$ ) позволяет провести плавную кривую для функции, заданной на равномерной сетке. В узлах сетки обеспечивается совпадение с исходной функцией и непрерывность первой производной. Программа имеет следующие параметры:

$XBEG$  — начальное значение аргумента;

$DX$  — размер постоянного шага сетки;

$Y$  — вектор значений заданной функции;

$N$  — число точек.

На рис. 5.3 каждый «язычок» пламени образован десятью кубическими параболами. Для вычисления коэффициентов многочлена  $y = a_ix^3 + b_ix^2 + c_ix + d_i$  задавались координаты двух точек и первые производные в этих точках.

#### 5.4. Сглаживание методом наименьших квадратов

Метод наименьших квадратов используется в том случае, когда требуется функцию, заданную в  $m$  точках  $x_j, y_j$  ( $j = 1, 2, \dots, m$ ), приблизить многочленом степени  $n < m$ :

$$y(x) = \sum_{i=0}^n a_i x^i. \quad (5.3)$$

Поскольку вычисленная кривая  $y(x)$ , вообще говоря, не проходит через заданные точки и дает сглаженное множество значений  $y(x)$ , необходимо определить меру близости исходной и приближающей функций. Из метода наименьших квадратов следует, что необходимо выбрать многочлен, который обеспечивает минимум суммы

$$\sum_{j=1}^m \rho_j \left( y_j - \sum_{i=0}^n a_i x_j^i \right)^2. \quad (5.4)$$

(В этом критерии имеется коэффициент  $0 \leq \rho_j \leq 1$ , который определяет значимость точки или вес, с которым она влияет на выбор многочлена. Если  $\rho_j = 0$ , то отклонение в  $j$ -й точке не учитывается при выборе многочлена.) Если рассматривать выражение (5.4) как функцию переменных  $a_k$ , то для нахождения минимума надо продифференцировать это выражение по каждой из неизвестных  $a_k$ . В результате получим

$$2 \sum_{j=1}^m \rho_j \left( y_j - \sum_{i=0}^n a_i x_j^i \right) x_j^k = 0, \quad k=0, 1, \dots, n,$$

или после преобразования

$$\sum_{j=1}^m \rho_j y_j x_j^k = \sum_{i=0}^n a_i \left[ \sum_{j=1}^m \rho_j x_j^{i+k} \right], \quad k=0, 1, \dots, n.$$

То есть для определения коэффициентов необходимо решить систему  $n+1$  линейных уравнений, которые называют нормальными уравнениями, с  $n+1$  неизвестными  $a_i$ . В матричной форме систему можно записать как  $B \times \bar{a} = \bar{c}$ , где  $B$  — симметрическая матрица размерности  $n+1$ . Заметим, что матрица  $B$  не зависит от значений исходной функции, и поэтому при обработке нескольких функций, заданных на одной сетке, достаточно построить и обработать матрицу  $B$  только один раз.

Программа LESQ(X, Y, RO, M, A, N) позволяет вычислить коэффициенты аппроксимирующего многочлена  $y(x) = a_1 + a_2 x + \dots + a_n x^{n-1}$  заданной степени методом наименьших квадратов с учетом весовых коэффициентов. Программа строит исходную матрицу  $A$ , а затем обращает ее на своем поле, используя подпрограмму SMINV. Программа имеет следующие параметры:

$X, Y$  — векторы значений аргумента и аппроксимирующей функции (длины  $|M|$ );

$RO$  — вектор значений весовых коэффициентов (длины  $|M|$ );

$M$  — число точек (если  $M < 0$ , то считается, что все точки имеют одинаковый вес, равный 1, и число точек принимается равным  $|M|$ ; в этом случае в качестве формального параметра  $RO$

может быть задан любой вектор, например  $X$ , причем он не влияет на результаты вычислений и сохраняется неизменным);

$A$  — вектор коэффициентов аппроксимирующего многочлена длины  $|N|$  (коэффициенты заносятся в порядке возрастания индекса, т. е.  $A(1) = a_1, A(2) = a_2, \dots, A(N) = a_n$ );

$N$  — степень аппроксимирующего многочлена плюс 1 (если  $N < 0$ , программа использует вычисленную предыдущим обращением к ней матрицу  $B^{-1}$  порядка  $|N|$ ).

Программа  $SMINV(B, V, N)$  позволяет найти матрицу, обратную симметрической, методом квадратных корней. Программа использует наддиагональный треугольник исходной матрицы, т. е. те элементы  $b_{ij}$ , для которых  $i \leq j$ . Обратная матрица помещается на место исходной и является полной матрицей, для которой  $b_{ij} = b_{ji}$ . Параметры программы:

$B$  — исходная симметрическая матрица порядка  $N$ ;

$V$  — рабочий вектор длины  $N$ ;

$N$  — порядок матрицы.

Итак, мы рассмотрели программы, которые позволяют получить многочлен, аппроксимирующий функцию, заданную таблично. По найденным коэффициентам можно вычислять значения аппроксимирующего многочлена с помощью программы  $PVAL$ . В том случае, когда нас интересует лишь графическое представление аппроксимирующей кривой, можно воспользоваться программой  $LSFIT$ .

Программа  $PVAL(RES, ARG, A, N)$  позволяет вычислить значения многочлена

$$y(x) = a_1 + a_2x + \dots + a_nx^{n-1}$$

$(n-1)$ -й степени для заданного аргумента. Параметры программы следующие:

$RES$  — значение многочлена от заданного аргумента;

$ARG$  — значение аргумента;

$A$  — вектор коэффициентов многочлена (длины  $N$ );

$N$  — степень многочлена плюс 1.

Программа  $LSFIT(X, Y, RO, M, N, MPTS)$  позволяет начертить кривую, аппроксимирующую функцию методом наименьших квадратов с учетом весовых коэффициентов. Для вычисления коэффициентов аппроксимирующего многочлена используется подпрограмма  $LESQ$ . Программа  $LSFIT$  имеет следующие параметры:

$X, Y$  — векторы значений аргумента и аппроксимирующей функции (длины  $|M|$ );

$RO$  — вектор значений весовых коэффициентов (длины  $|M|$ );

$M$  — число точек (если  $M < 0$ , то считается, что все точки имеют одинаковый вес, равный 1, и число точек принимается равным  $|M|$ ; в этом случае в качестве формального параметра  $RO$  может



быть задан любой вектор, причем он не влияет на результаты вычислений и сохраняется неизменным);

$N$  — степень аппроксимирующего многочлена плюс 1 (этот параметр передается в программу LESQ, и если  $N < 0$ , то подпрограмма LESQ использует вычисленную предыдущим обращением к ней матрицу  $B^{-1}$  порядка  $|N|$ );

$|MPTS|$  — количество узлов равномерной сетки на отрезке  $[X(1), X(M)]$ : если  $MPTS > 0$ , аппроксимирующая кривая проводится непрерывной линией, если  $MPTS < 0$  — штриховой.

На рис. 5.4 показана аппроксимация синусоидальной кривой многочленом третьей степени. Исходная кривая помечена маркером номер 3. Сплошной линией показана кривая, принадлежащая

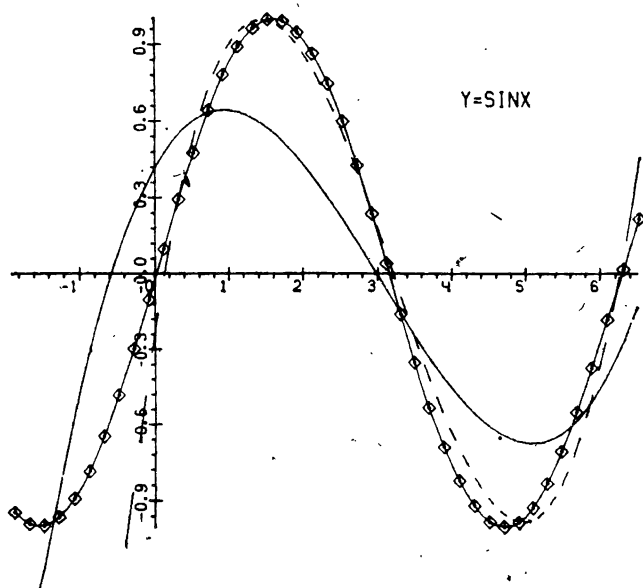


Рис. 5.4. Аппроксимация синусоидальной кривой многочленом третьей степени.

многочлену, который получен для точек, взятых с равными весами. Штриховая кривая принадлежит многочлену, полученному при условии, что точки на отрезке  $[-1.9, 0.]$  имеют вес, равный 0, точки на отрезке  $[0.1, 3.1]$  — вес, равный 1, а все остальные точки взяты с весом 0.2. Очевидно, улучшение приближения на выделенном отрезке  $[0.1, 3.1]$  получено в ущерб приближению для всей кривой. Ниже приведена программа, выполняющая этот рисунок.

```
DIMENSION X(100), Y(100), RO(100)
X(1)=-1.9
```

```

DO 1 I=1, 85
Y(I)=SIN(X(I))
1 X(I+1)=X(I)+.1
CALL PAGE(17., 26., 0, 0, 0)
CALL MINMAX(Y, 85, YN, YX)
CALL LIMITS(-1.9, X(85), YN, YX)
CALL REGION(1.5, 1.5, 14., 11., 0, 0, 0)
CALL LINEMO(X, Y, 85, 3, 1)
DO 2 I=1, 85
2 RO(I)=1.
CALL LSFIT(X, Y, RO, 85, 4, 100)
DO 3 I=1, 20
3 RO(I)=0.
DO 4 I=51, 85
4 RO(I)=.2
CALL LSFIT(X, Y, RO, 85, 4, -100)
CALL SYMBOL(11.5, 10.5, .4, 'Y=SINX', 6, 0)
CALL AXES(0, 0, .1, 5, 0, 0, .3, 5, 0)
CALL ENDPG('5.4')
END

```

Описанный метод не рекомендуется использовать, если степень многочлена выше пяти, так как матрица становится плохо обусловленной. Это обстоятельство снижает ценность прямого решения системы нормальных уравнений. Можно избежать затруднений, воспользовавшись для представления функции *ортгогональными многочленами*, т. е. вычислять не многочлен (5.3), а некоторый его эквивалент. Тогда наилучшей аппроксимацией

$$y(x) = \sum_{i=0}^n s_i p_i(x)$$

будет такая, при которой обеспечивается минимум суммы

$$\sum_{j=1}^m \rho_j \left[ y_j - \sum_{i=0}^n s_i p_i(x_j) \right]^2.$$

Для образования ортогональных многочленов используется трехчленное рекуррентное соотношение (см. [12]).

Программа APPOLY(X, Y, RO, M, A, N, C) позволяет вычислить коэффициенты аппроксимирующего многочлена (5.3) заданной степени методом наименьших квадратов с использованием ортогональных многочленов. Параметры программы:

X, Y — векторы значений аргумента и аппроксимируемой функции (длины |M|);

RO — вектор значений весовых коэффициентов (длины |M|);

M — количество точек (если M < 0, то считается, что все точ-

ки имеют одинаковый вес, равный 1, и количество точек принимается равным  $|M|$ ; в этом случае в качестве формального параметра  $RO$  может быть задан любой вектор, причем он не влияет на результаты вычислений и сохраняется неизменным);

$A$  — вектор коэффициентов аппроксимирующего многочлена (5.3) длины  $|N|$  (коэффициенты заносятся в порядке возрастания индекса, т. е.  $A(1) = a_1, \dots, A(N) = a_n$ );

$N$  — степень аппроксимирующего многочлена плюс 1;

$C$  — рабочий вектор длины  $2 * N$ .

По вычисленным коэффициентам можно получить значения аппроксимирующего многочлена, воспользовавшись подпрограммой  $PVAL$ , и затем начертить аппроксимирующую кривую с помощью одной из имеющихся подпрограмм проведения линий ( $LINEO$ ,  $LINEMO$ ,  $INCLIN$  и т. п.).

## 5.5. Аппроксимация функций конечным рядом Фурье

Если функция  $f(x)$  задана на отрезке  $[0, 2\pi]$  в  $2N+1$  дискретных точках, т. е. с шагом  $2\pi/(2N+1)$ , то она может быть представлена как

$$f(x) = \frac{A_0}{2} + \sum_{k=1}^M (A_k \cos kx + B_k \sin kx), \quad (5.5)$$

причем  $0 \leq M \leq N$  и  $0 \leq x \leq 2\pi$ .

Поскольку система функций, участвующих в разложении (5.5), ортогональна на заданном дискретном множестве точек, коэффициенты разложения представляются формулами

$$A_k = \frac{2}{2N+1} \sum_{l=0}^{2N} f\left(\frac{2\pi l}{2N+1}\right) \cos \frac{2\pi k}{2N+1} l, \quad k = 0, 1, \dots, N,$$

$$B_k = \frac{2}{2N+1} \sum_{l=0}^{2N} f\left(\frac{2\pi l}{2N+1}\right) \sin \frac{2\pi k}{2N+1} l, \quad k = 1, \dots, N.$$

Если в разложении (5.5)  $M = N$ , то  $2N+1$  значений  $f(x)$  ( $x = 0, 1, \dots, 2N$ ) определяют  $2N+1$  коэффициентов  $A_k, B_k$ . Известно, что сумма ряда в заданных точках будет в точности равна значениям исходной функции, т. е. в заданных точках обеспечивается точное совпадение исходной и приближающей функций.

Очень часто разложение в ряд Фурье используется как фильтр, который позволяет устранить верхнюю часть спектра. При разложении в ряд Фурье в спектре будут представлены лишь те гармони-

ки, которые содержат не более  $M$  периодов в интервале задания исходной функции.

Реализация метода состоит из трех подпрограмм. Подпрограмма FORIT вычисляет указанное количество коэффициентов  $M$  для функции, определенной на дискретном множестве точек, а подпрограмма FORIF — заданное число коэффициентов  $M$  ряда Фурье, аппроксимирующего исходную функцию, значения которой определяются с помощью подпрограммы-функции на отрезке  $[0, 2\pi]$  в  $2N + 1$  точках с шагом  $2\pi/(2N + 1)$ . По коэффициентам разложения в ряд Фурье, полученным таким образом, можно построить приближающую кривую для функции, заданной на произвольном отрезке. Для этой цели предназначена подпрограмма FORFIT. Кривая может быть вычерчена как непрерывной, так и штриховой линией. (Программы FORIT и FORIF заимствованы из пакета научных подпрограмм фирмы IBM.)

Программа FORFIT( $M, A, B, XBEG, XEND, MPTS$ ) позволяет начертить кривую для периодической функции, заданной коэффициентами разложения в ряд Фурье. Коэффициенты могут быть получены с помощью подпрограмм FORIT или FORIF. Параметры программы:

$M$  — количество гармоник, участвующих в разложении;

$A$  — вектор коэффициентов при косинусах в разложении функции длины  $M + 1$  (коэффициенты располагаются в порядке возрастания индекса);

$B$  — вектор коэффициентов при синусах в разложении функции длины  $M + 1$  (коэффициенты располагаются в порядке возрастания индекса, причем  $B(1) = 0$ );

$XBEG, XEND$  — значения аргументов, соответствующих левому и правому концам отрезка, на котором определена функция;  $|MPTS|$  — число интервалов, покрывающих аппроксимирующую кривую (ширина интервала  $= (XEND - XBEG)/|MPTS|$ ,  $MPTS \neq 0$ ; если  $MPTS > 0$ , аппроксимирующая кривая проводится непрерывной линией, если  $MPTS < 0$  — штриховой.

Программа FORIT( $FNT, N, M, A, B, IER$ ) позволяет вычислить коэффициенты ряда Фурье, аппроксимирующего исходную табулированную функцию, заданную на отрезке  $[0, 2\pi]$  в  $2N + 1$  равноотстоящих точках. Параметры программы:

$FNT$  — вектор, задающий  $2N + 1$  значений табулированной функции;

$N$  — определяет отрезок так, что на отрезке  $[0, 2\pi]$  значения заданы в  $2N + 1$  равноотстоящих точках с шагом  $2\pi/(2N + 1)$ ;

$M$  — количество гармоник, участвующих в разложении;

$A$  — вектор коэффициентов при косинусах в разложении функции длины  $M + 1$  (коэффициенты располагаются в порядке возрастания индекса);

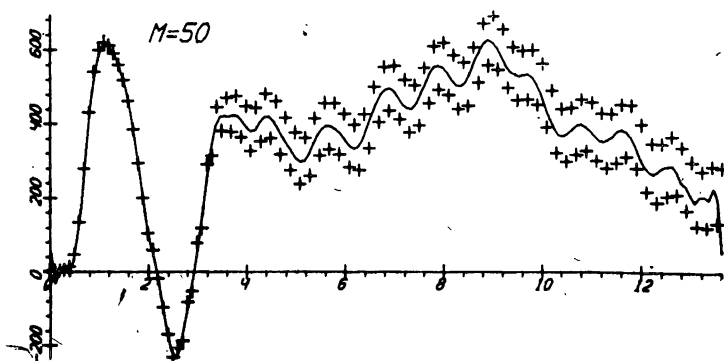
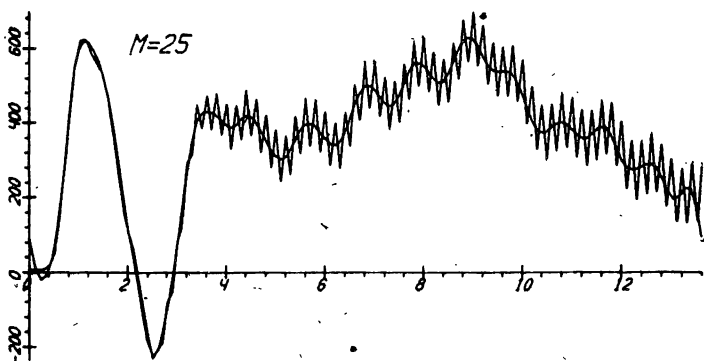
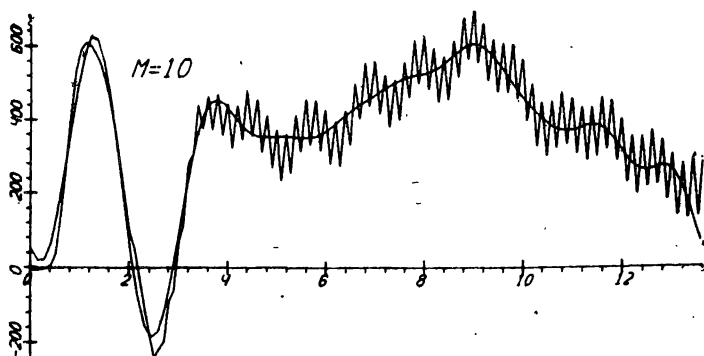


Рис. 5.5. Пример, иллюстрирующий использование разложения в ряд Фурье как фильтра.

$B$  — вектор коэффициентов при синусах в разложении функции длины  $M + 1$  (коэффициенты располагаются в порядке возрастания индекса, причем  $B(1) = 0$ );

$IER$  — код ошибки:  $IER = 0$  — ошибки нет,  $IER = 1$ , если  $N < M$ ,  $IER = 2$ , если  $M < 0$ .

Программа FORIF(FUN, N, M, A, B, IER) позволяет вычислять коэффициенты ряда Фурье, аппроксимирующего исходную функцию, значения которой вычисляются с помощью подпрограммы-функции на отрезке  $[0, 2\pi]$  в  $2N + 1$  точках с шагом  $2\pi/(2N + 1)$ . Параметры программы:

FUN — имя подпрограммы-функции, которая используется для вычисления значения функции в заданных точках.

Остальные параметры имеют тот же смысл, что и аналогичные параметры подпрограммы FORIT.

Использование разложения функции в ряд Фурье как фильтра иллюстрируется на рис. 5.5, где показаны исходная функция, помеченная маркером  $+$  на нижнем рисунке, и приближение, полученное при разложении в ряд Фурье. На рисунках указано число  $M$  — количество гармоник в разложении.

Ниже приводится программа, с помощью которой получен график, изображенный в нижней части рис. 5.5 ( $M = 50$ ).

```
DIMENSION X(140), Y(140), A(100), B(100)
READ 1, Y
1 FORMAT(7F10.3)
X(1) = 0.
DO 2 I = 1, 137
2 X(I+1) = X(I) + .1
CALL PAGE(17., 25.5, 0, 0, 0)
CALL MINMAX(Y, 137, YN, YX)
CALL LIMITS(0., X(137), YN, YX)
CALL REGION(1., .5, 15., 7.5, 0, 0, 0)
CALL LINEMO(X, Y, 137, 1, -1)
CALL FORIT(Y, 68, 50, A, B, IER)
CALL FORFIT(50, A, B, 0., X(137), 150)
CALL ITALIC(1)
CALL SYMBOL(3.2, 7.3, .5, 'M=50', 4, 0)
CALL AXES(0, 0, 0., 5, 0, 0, 200., 5, 0)
CALL ENDPG('5.5')
END
```

## 5.6. Линейный фильтр

В некоторых случаях сглаживание функции, связанное с устранением высокочастотных составляющих, можно выполнить более простым способом по сравнению с аппроксимацией рядами

Фурье. Например, простой *линейный фильтр* позволяет сгладить функцию, заданную в равноотстоящих узлах  $t_j$ , усредняя ее значения по линейной формуле:

$$\tilde{f}_j = \left( \sum_{i=j-m}^{j+m} f_i \right) / (2m+1), \quad j = 1, 2, \dots, n. \quad (5.6)$$

Заметим, что суммируется нечетное число точек. Если суммируется четное число точек, то используется формула

$$\tilde{f}_j = \left( \sum_{i=j-m+1}^{j+m} f_i \right) / (2m), \quad j = 1, 2, \dots, n. \quad (5.7)$$

Вообще говоря, формулы (5.6), (5.7) верны лишь в том случае, если  $m < j \leq n - m$ , т. е. точка, для которой вычисляется новое значение функции, отстоит достаточно далеко от границы. Во всех других случаях соответствующим образом изменяются нижний и верхний пределы суммирования и знаменатель. Такой фильтр уменьшает амплитуды гармоник в верхней половине спектра. Более подробный анализ линейных фильтров можно найти в монографии Хемминга [12].

Описываемый метод сглаживания реализован в программе LINFIL. Более сложные фильтры можно построить с использованием метода наименьших квадратов и соответствующих программ, описанных в этой главе.

Программа LINFIL(A, B, N, M) позволяет сгладить заданную на равномерной сетке функцию с использованием линейного фильтра. Параметры программы следующие:

A — вектор значений функции (длины |N|);

B — вектор усредненных значений функции в узлах (длины |N|);

N — число узлов сетки;

M — параметр, определяющий число узлов, участвующих в суммировании (индекс  $m$  из формул (5.6) и (5.7)).

При  $N > 0$  суммирование ведется по формуле (5.6), при  $N < 0$  используется формула (5.7). Число узлов берется равным |N|.

Если в обращении к программе в качестве формальных параметров A и B указан один и тот же вектор, то в сумму войдут вновь вычисленные значения функции в узлах, расположенных слева от текущего узла. Этот способ может быть полезен, если используется формула (5.7).

На рис. 5.6 показан пример сглаживания функции с помощью линейного фильтра (ср. с рис. 5.5).

```
DIMENSION Y(140), Z(140)
```

```
READ 5, Y
```

```
5 FORMAT(7F10.3)
```

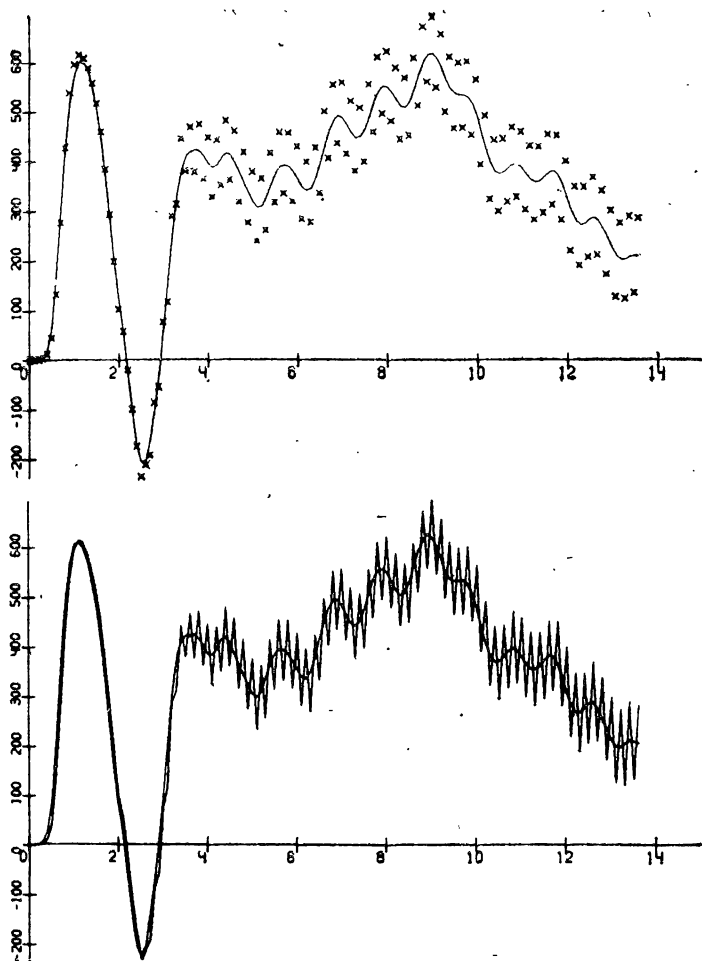


Рис. 5.6. Пример использования линейного фильтра.

```
CALL PAGE(16., 25., 0, 0, 0)
CALL LINFIL(Y, Z, -137, 1)
CALL MINMAX(Y, 137, ZMN, ZMX)
CALL REGION(1., .5, 15., 10., 0, 0, 0)
CALL LIMITS(0., 15., ZMN, ZMX)
CALL INCLIN(0., .1, 0, Y, 137, 0, 0)
CALL INCLIN(0., .1, 0, Z, 137, 0, 0)
CALL AXES(0, 0, 0, 0, 0, 0, 0, 0, 0)
CALL REGION(1., 11., 15., 10., 0, 0, 0)
```



```

CALL INCLIN(0., .1, 0, Y, 137, -2, -1)
CALL LINFIL(Y, Y, 137, 1)
CALL INCLIN(0., .1, 0, Y, 137, 0, 0)
CALL AXES(0, 0, 0., 0, 0, 0, 0., 0, 0)
CALL ENDPG('5.6')
END

```

## 5.7. Аппроксимация функций ортогональными многочленами Чебышева

Для приближения непериодических функций используется ортогональная система функций — многочлены Чебышева, которые, в сущности, являются функциями Фурье  $\cos n\theta$ , замаскированными простым преобразованием переменной  $\theta = \arccos x$ . Таким образом,  $T_n(x) = \cos(n \arccos x)$ . Многочлены Чебышева связаны рекуррентным соотношением

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$

причем  $T_0 = 1$ ,  $T_1 = x$ .

Приближающая функция ищется в виде суммы многочленов Чебышева, т. е.

$$f(x) \approx \sum_{i=0}^M c_i T_i(x). \quad (5.8)$$

Используя узловые точки многочленов, т. е. систему точек, на которой многочлены Чебышева ортогональны, получаем следующую формулу для вычисления коэффициентов  $c_i$ :

$$c_i = \left( \sum_{j=1}^N f(x_j) T_i(x_j) \right) / \left( \sum_{j=1}^N T_i^2(x_j) \right). \quad (5.9)$$

Узловые точки многочленов распределены неравномерно (они сгущаются к концам отрезка  $[-1, 1]$ ), а именно

$$x_j = \cos \left( \frac{2j-1}{2N} \pi \right), \quad j = 1, 2, \dots, N. \quad (5.10)$$

Программа CHENSP вычисляет коэффициенты  $c_i$  для дискретной функции, заданной в равноотстоящих точках, используя при этом соответствующее преобразование переменной (5.10). После того как коэффициенты известны, программа позволяет вычислить значения функции на заданной равномерной сетке, опять-таки с преобразованием переменной.

Следует заметить, что коэффициенты  $c_i$  (см. (5.9)) не зависят от  $M$ , т. е. от количества многочленов, входящих в сумму (5.8). Этот факт используется в примере, который приводится вслед за описанием программы.

Программа CHENSP(YM, N, YH, K, C, M) позволяет вычислить коэффициенты приближающей функции  $C$  (если  $N > 0$ ) и ее значения  $YH$  (если  $K > 0$ ) на множестве равноотстоящих точек, используя ортогональную систему функций — многочлены Чебышева. Если  $N = 0$ , то значения  $YH$  вычисляются в предполо-

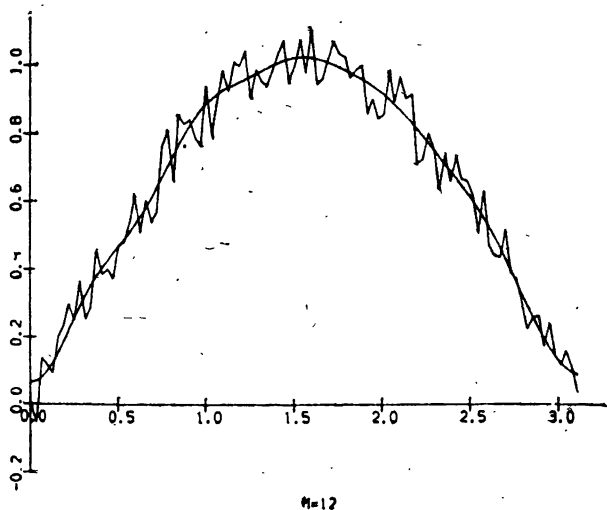
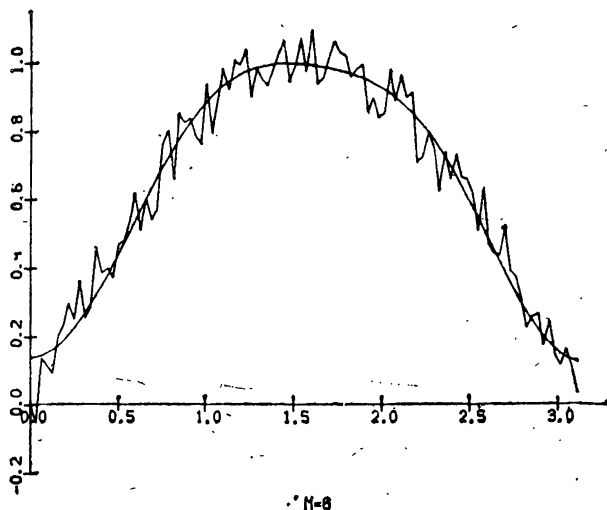


Рис. 5.7. Пример, иллюстрирующий аппроксимацию функции многочленами Чебышева.

жепни, что коэффициенты  $C$  вычислены ранее (см. пример). Программа имеет следующие параметры:

$YM$  — вектор значений дискретной функции длины  $|N|$ ;

$YN$  — вектор значений приближающей функции длины  $|K|$ ;

$C$  — вектор коэффициентов приближающей функции длины  $|M|$ .

На рис. 5.7 изображена часть синусоиды, на которую наложен шум ( $Y = \sin(X) + \text{RAND}(-0.1, 0.1, 1937)$ ), и приближение этой функции многочленами Чебышева (два случая:  $M = 12$  и  $M = 6$ ).

```
DIMENSION YM(100), YN(100), C(12)
```

```
KR=1937
```

```
XM=0.
```

```
YM(1)=RAND(-1.01, 0.01, KR)
```

```
DO 1 I=2, 100
```

```
XM=XM+0.0314159
```

```
1 YM(I)=SIN(XM)+RAND(-0.1, 0.1, KR)
```

```
CALL CHENSP(YM, 100, YN, 100, C, 12)
```

```
CALL PAGE(17., 26., 0, 0, 0)
```

```
CALL REGION(1., 1., 13., 10., 0, 0, 0)
```

```
CALL LIMITS(0., 3.3, -0.2, 1.15)
```

```
CALL AXES('M=12', 4, 0., 0, 0, 0, 0., 0, 0)
```

```
CALL INCLIN(0., 0.0314159, 0, YM, 100, 0, 0)
```

```
CALL INCLIN(0., 0.0314159, 0, YN, 100, 0, 0)
```

```
CALL CHENSP(0., 0, YN, 100, C, 6)
```

```
CALL REGION(1., 13., 13., 10., 0, 0, 0)
```

```
CALL AXES('M=6', 3, 0., 0, 0, 0, 0., 0, 0)
```

```
CALL INCLIN(0., 0.0314159, 0, YM, 100, 0, 0)
```

```
CALL INCLIN(0., 0.0314159, 0, YN, 100, 0, 0)
```

```
CALL ENDPG('5.7')
```

```
END
```

Функция  $\text{RAND}(A, B, KR)$  позволяет генерировать псевдослучайные числа с равномерным распределением в интервале  $(A, B)$ , где  $A < B$ . Функция имеет следующие параметры:

$A, B$  — нижняя и верхняя границы интервала;

$KR$  — целая переменная, принимающая значения в диапазоне от 1 до 67 108 863.

Перед первым обращением к функции  $\text{RAND}$  необходимо занести в  $KR$  нечетное число в указанных пределах. Значение  $KR$  не следует изменять между обращениями к функции.

## 5.8. Аппроксимация методом Безье

В системах автоматизации проектирования и производства для конструирования кривых и поверхностей применяется аппроксимация методом Безье. Сущность метода заключается в следующем.

Пусть задана совокупность из  $n + 1$  точек  $P_0, \dots, P_n$ , которую будем называть *ломаной Безье*. *Кривая Безье*, соответствующая этой ломаной, описывается в виде функции параметра  $t$  следующим полиномом:

$$\bar{p}(t) = \sum_{i=0}^n \bar{P}_i J_{ni}(t), \quad (5.11)$$

где  $\bar{p}(t)$  — радиус-вектор точек на кривой, а  $J_{ni}(t)$  — аппроксимирующие многочлены Бернштейна, равные

$$J_{ni}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}. \quad (5.12)$$

Здесь  $0 \leq t \leq 1$  и, кроме того, предполагается, что  $t^i = 1$  при  $i = 0$  и  $t = 0$ .

#### КРИВАЯ БЕЗЬЕ

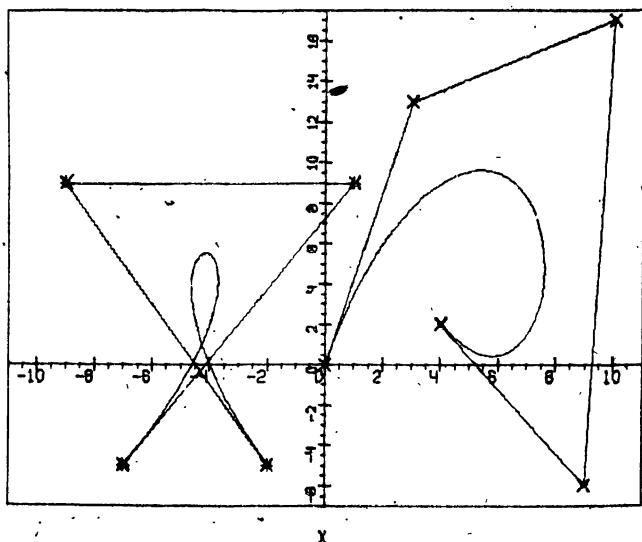


Рис. 5.8. Пример аппроксимации методом Безье.

Ломаная Безье однозначно определяет форму кривой Безье. Изменяя положения вершин ломаной, можно управлять формой соответствующей кривой Безье. При этом следует иметь в виду следующее:

1) самой кривой в общем случае будут принадлежать только первая и последняя вершины ломаной Безье, остальные вершины будут лишь оказывать влияние на вид и гладкость кривой (рис. 5.8);

2) наклоны касательных векторов в крайних точках кривой Безье и ломаной Безье совпадают (рис. 5.8), поэтому при сопря-

жении двух кривых Безье, заданных ломаными  $P_0, \dots, P_n$  и  $Q_0, \dots, Q_m$ , одинаковый наклон кривых в точке соединения получается в том случае, если точки  $P_{n-1}, P_n$  (которая совпадает с  $Q_0$ ) и  $Q_1$  лежат на одной прямой;

3) как видно из выражений (5.11) и (5.12), степень аппроксимирующего полинома равна  $n$  (т. е. числу звеньев в ломаной Безье), поэтому для увеличения порядка кривой Безье достаточно лишь задать дополнительные вершины в соответствующей ломаной Безье;

4) кривая Безье всегда целиком лежит внутри выпуклой оболочки ломаной Безье.

Метод аппроксимации Безье реализован в программах BEZ2 и BEZ3, с помощью которых можно построить соответственно двумерные и трехмерные кривые Безье.

Программа BEZ2(XP, YP, NPLG, XC, YC, NPTS) позволяет вычислить значения координат точек плоской кривой, аппроксимирующей заданную совокупность точек методом Безье. Программа имеет следующие параметры:

XP, YP — векторы значений соответственно X- и Y-координат заданной совокупности точек (ломаной Безье) длины NPLG;

NPLG — число точек в ломаной Безье;

XC, YC — векторы значений соответственно X- и Y-координат аппроксимирующей кривой Безье длины NPTS;

NPTS — число точек в аппроксимирующей кривой Безье.

Программа BEZ3(XP, YP, ZP, NPLG, XC, YC, ZC, NPTS) позволяет вычислить значения координат точек пространственной кривой, аппроксимирующей заданную совокупность точек методом Безье. Параметры у программы следующие:

XP, YP, ZP — векторы значений соответственно X-, Y- и Z-координат заданной совокупности точек (ломаной Безье) длины NPLG;

XC, YC, ZC — векторы значений соответственно X-, Y- и Z-координат аппроксимирующей кривой Безье (длины NPTS).

Параметры NPLG и NPTS имеют то же значение, что и в программе BEZ2.

На рис. 5.8 показаны ломаные Безье и соответствующие им кривые Безье. Построение выполнялось с помощью следующего фрагмента программы.

```
PARAMETER(NL=5, NS=41, NL1=4, NS1=61)
REAL XC(NS), YC(NS), XP(NP), YP(NP)
REAL XC1(NS1), YC1(NS1), XP1(NL1), YP1(NL1)
DATA XP/0., 3., 10., 9., 4./, YP/0., 13., 17., -6., 2./
DATA XP1/-2., -9., 1., -10./, YP1/-5., 7., 7., -5./
CALL BEZ2(XP, YP, NL, XC, YC, NS)
```

```

CALL BEZ2(XP1, YP1, NL1, XC1, YC1, NS1)
CALL PAGE(18., 14., 0, 0, 0)
CALL REGION(1., 1., 16., 12., 'КРИВАЯ БЕЗЬЕ', 12, 1)
CALL LIMITS(-11., 11., -7., 18.)
CALL AXES('X', 1, 0.0, 4, 'Y', 1, 0.0, 4, 00)
CALL LINEMO(XP, YP, NL, 2, 1)
CALL LINEO(XC, YC, NS)
CALL LINEMO(XP1, YP1, NL1, -7, 1)
CALL LINEO(XC1, YC1, NS1)
CALL ENDPG('5.8')
END

```

Для изображения пространственных кривых Безье можно воспользоваться программами, описанными в п. 8.1.3. При этом могут оказаться полезными также средства повышения наглядности изображения (см. п. 8.4.4, рис. 8.33).

## 5.9. Аппроксимация на основе *B*-сплайнов

Основу метода Безье составляет аппроксимация многочленами Бернштейна. Однако можно выбрать и другие наборы базисных функций, например *B-сплайны*, по отношению к которым базис Бернштейна является частным случаем. В сравнении с многочленами Бернштейна *B-сплайны* обладают следующими дополнительными преимуществами [16—18].

1. Аппроксимация *B-сплайнами* обеспечивает более точное приближение ломаной, чем аппроксимация многочленами Бернштейна.

2. При аппроксимации методом Безье на значения координат каждой точки кривой оказывают влияние все вершины ломаной Безье. Это затрудняет корректирование отдельных участков кривой. В то же время аппроксимация *B-сплайнами* обладает желательной локальностью. Этим способом можно производить локальные изменения кривой без полного ее пересчета.

3. Единственным путем увеличения (уменьшения) порядка кривой Безье является увеличение (уменьшение) числа вершин в соответствующей ломаной Безье. В методе *B-сплайнов* эти два параметра независимы и, следовательно, могут быть выбраны произвольно.

Кривая, построенная на основе *B-сплайн-базиса*, описывается следующим образом [16, 18]:

$$\bar{p}(t) = \sum_{i=0}^n \bar{P}_i N_{ik}(t), \quad (5.13)$$

где  $\bar{p}(t)$  — радиус-вектор точек на кривой,  $\bar{P}_i$  — вершины аппроксимируемой ломаной (всего вершин  $n + 1$ ), а  $N_{ik}(t)$  — весовая функ-

ция  $t$ -й нормализованной  $B$ -сплайн базисной кривой порядка  $k$  (т. е. степени  $k - 1$ ), задаваемая рекуррентными соотношениями:

$$N_{i,1}(t) = \begin{cases} 1, & \text{если } x_i \leq t \leq x_{i+1}, \\ 0 & \text{в остальных случаях} \end{cases}$$

и

$$N_{i,k}(t) = \frac{(t - x_i) N_{i,k-1}(t)}{x_{i+k-1} - x_i} - \frac{(x_{i+k} - t) N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}. \quad (5.14)$$

Здесь  $x_i$  — элементы *узлового вектора*, а  $t$  — параметр, изменяющийся в диапазоне от 0 до  $t_{\max} = n - k + 2$ .

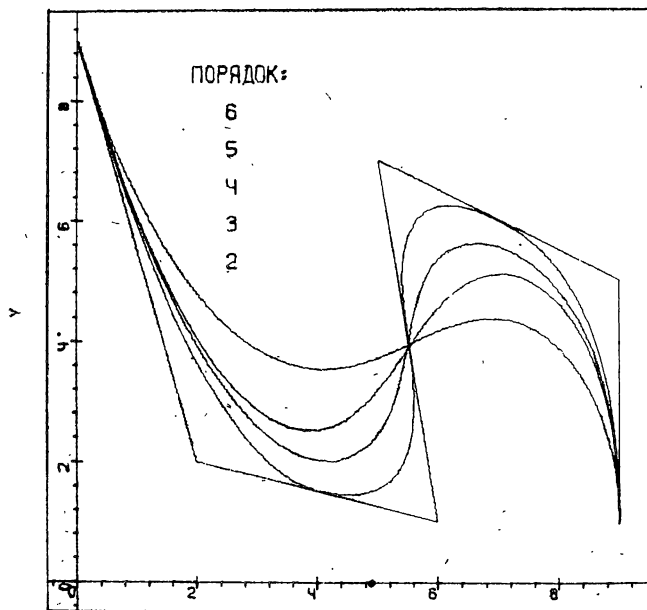


Рис. 5.9. Изображение нескольких  $B$ -сплайн-кривых различного порядка.

Узловой вектор, длина которого  $n + k + 1$ , вводится для учета собственной кривизны  $B$ -сплайн-кривых и представляет собой неубывающую последовательность целых чисел — *параметрических узлов*. Узловой вектор определяется числом точек в аппроксимируемой ломаной, порядком кривой, а также наличием сложных (кратных) узлов.

Из (5.13) и (5.14) следует, что  $B$ -сплайн-кривая является полиномом степени  $k - 1$  на каждом интервале  $(x_i, x_{i+1})$  и что все ее производные до  $(k - 2)$ -го порядка включительно непрерывны

вдоль всей кривой. То есть эта кривая представляет собой сплайн-функцию порядка  $k$  (степени  $k-1$ ).

На рис. 5.9 изображены несколько  $B$ -сплайн-кривых различного порядка, аппроксимирующих ломаную с шестью вершинами. При этом кривая пятого порядка является кривой Безье для заданной ломаной, кривая четвертого порядка — кубическим сплайном, а кривая второго порядка — последовательностью связанных отрезков, совпадающих с исходной ломаной. Представляет интерес также кривая третьего порядка — она касается внутренних отрезков

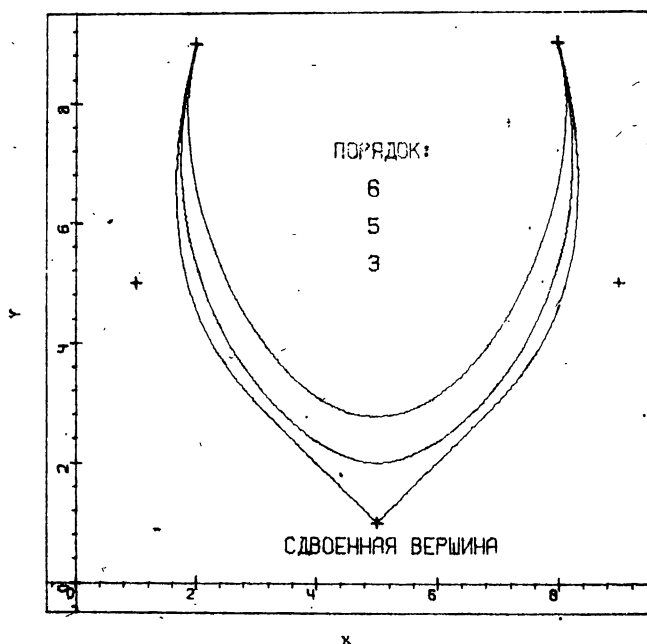


Рис. 5.10.  $B$ -сплайн-кривые со двойной вершиной. В этой вершине в кривой третьего порядка образуется излом.

ломаной в их средних точках. Заметим, что с ростом порядка  $B$ -сплайн-кривые как бы спрямляются, их форма все в большей степени становится отличной от аппроксимируемой ломаной.

В формируемые кривые можно вносить изломы путем включения в соответствующую ломаную кратных вершин. Так, для образования излома в кривой третьего порядка необходима двойная вершина (рис. 5.10). А чтобы создать излом в кривой четвертого порядка, потребуется вершина с кратностью три и т. д.

Большое количество примеров, иллюстрирующих различные способы построения  $B$ -сплайн-кривых, читатель может найти в [16].



Аппроксимация *B*-сплайнами реализована в программах BSPLN2 и BSPLN3. Первая из них предназначена для формирования плоских, а вторая — пространственных *B*-сплайн-кривых.

Программа BSPLN2(K, XP, YP, NPLG, XC, YC, NPTS, WFUN, NWF, VKNOTS, NKNOTS) позволяет вычислить значения координат точек плоской *B*-сплайн-кривой, аппроксимирующей заданную совокупность точек. Программа имеет следующие параметры:

K — порядок аппроксимирующей кривой (ее степень равна  $K - 1$ ),  $2 \leq K \leq NPLG$ ;

XP, YP — векторы значений соответственно *X*- и *Y*-координат заданной совокупности точек (аппроксимируемой ломаной) длины NPLG;

NPLG — число точек в аппроксимируемой ломаной;

XC, YC — векторы значений соответственно *X*- и *Y*-координат аппроксимирующей *B*-сплайн-кривой (длины NPTS);

NPTS — число точек в аппроксимирующей *B*-сплайн-кривой;

WFUN — двумерный рабочий массив, предназначенный для размещения весовых функций *B*-сплайн-базиса, размером (NWF, K);

NWF — максимальное значение, принимаемое первым индексом рабочего массива WFUN,  $NWF = NPLG + K - 1$ ;

VKNOTS — одномерный рабочий массив, предназначенный для размещения элементов узлового вектора, длины NKNOTS;

NKNOTS — длина узлового вектора,  $NKNOTS = NPLG + K$ .

Программа BSPLN3(K, XP, YP, ZP, NPLG, XC, YC, ZC, NPTS, WFUN, NWF, VKNOTS, NKNOTS) позволяет вычислить значения координат точек пространственной *B*-сплайн-кривой, аппроксимирующей заданную совокупность точек. Параметры у программы следующие:

XP, YP, ZP — векторы значений соответственно *X*-, *Y*- и *Z*-координат заданной совокупности точек (аппроксимируемой ломаной) длины NPLG;

XC, YC, ZC — векторы значений соответственно *X*-, *Y*- и *Z*-координат аппроксимирующей *B*-сплайн-кривой длины NPTS.

Остальные параметры имеют то же значение, что и в программе BSPLN2.

В программах BSPLN2 и BSPLN3 используются служебные программы KNOTV2(K, XP, YP, NPLG, VKNOTS, NKNOTS) и KNOTV3(K, XP, YP, ZP, NPLG, VKNOTS, NKNOTS), с помощью которых строятся узловые векторы соответственно для двумерного и трехмерного случаев. Параметры этих программ имеют то же значение, что и в программах BSPLN2 и BSPLN3.

Ниже приведена программа, с помощью которой выполнялось построение рис. 5.10,

```

REAL XP(6), YP(6)
REAL XC1(41), YC1(41), XC2(41), YC2(41),
*XC3(41), YC3(41)
REAL WF1(8, 3), WF2(10, 5), WF3(12, 7), VK(13)
DATA XP/2., 1., 2*5., 9., 8./
DATA YP/9., 5., 2*1., 5., 9./, NP, NS/6, 41/
CALL BSPLN2(3, XP, YP, NP, XC1, YC1, NS, WF1,
*8, VK, 9)
CALL BSPLN2(5, XP, YP, NP, XC2, YC2, NS, WF2,
*10, VK, 11)
CALL BSPLN2(7, XP, YP, NP, XC3, YC3, NS, WF3,
*12, VK, 13)
CALL PAGE(18., 18., 0, 0, 0)
CALL REGION(1., 1., 16., 16., 0, 0, 1)
CALL AXES('X', 1, 0.0, 5, 'Y', 1, 0.0, 5, 00)
CALL LINEMO(XP, YP, NP, 1, -1)
CALL LINEO(XC1, YC1, NS)
CALL LINEO(XC2, YC2, NS)
CALL LINEO(XC3, YC3, NS)
CALL ENDPG('5.10')
END

```

## 6. ГЛАДКОЕ ВОСПОЛНЕНИЕ И ИНТЕРПОЛЯЦИЯ ФУНКЦИЙ ДВУХ ПЕРЕМЕННЫХ

Значения функции, заданной на сетке, известны лишь в узлах сетки. Для определения значений функции в точках, не являющихся узлами сетки, применяются различные методы интерполяции, которые позволяют восполнить функцию. В главе рассматриваются два способа восполнения и интерполяции функций двух переменных, основанные на методах В. С. Рябенского [13] и Х. Акима [14]. Функция, заданная на прямоугольной сетке, восполняется либо в совокупности точек, расположенных на плоскости произвольным образом, либо в узлах новой, более частой сетки, полученной из исходной делением интервалов между каждой парой соседних точек по осям  $X$  и  $Y$  на равные отрезки. Аппроксимация производится кусочно-многочленными функциями гладкости 0, 1, 2, 3 (метод В. С. Рябенского) или бикубическими многочленами (метод Х. Акима).

К сожалению, не удалось провести сравнение этих двух методов, не совсем даже ясны критерии такого сравнения. По-видимому, только опыт позволит судить о том, какой из двух методов более предпочтителен в той или иной задаче.

### 6.1. Восполнение функций двух переменных, основанное на методе В. С. Рябенского

Пусть некоторая функция двух переменных определена на прямоугольной неравномерной сетке, заданной декартовым произведением двух одномерных сеток  $\{x_k\}$  и  $\{y_l\}$ , где

$$\{x_k\} = \{x_0, x_1, \dots, x_n | x_{i-1} \leq x_i < x_{i+1}\},$$

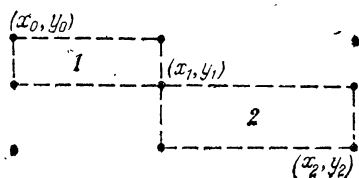
$$\{y_l\} = \{y_0, y_1, \dots, y_m | y_{j-1} \leq y_j < y_{j+1}\}.$$

Задача гладкого восполнения этой функции состоит в доопределении сеточной функции во всем прямоугольнике, обрамляющем сетку, до некоторой кусочно-многочленной функции заданной гладкости. В этом параграфе представлена программная реализация метода восполнения сеточной функции двух переменных для степеней гладкости  $p$ , равных 0, 1, 2, 3 (т. е. восполнение производит-

ся полипомами 1-й, 2-й, 3-й или 4-й степеней). Этот метод является частным случаем более общего метода, разработанного В. С. Рябенкиным [13].

Восполнение реализовано в программах SMTBVF, SMTSIM, SMTONE, которые позволяют аппроксимировать функцию с указанной степенью гладкости. Программы SMTSIM и SMTONE воспроизводят функцию с гладкостью единица. Они отличаются друг от друга формой задания исходных данных и получаемых результатов. В то же время их отличие от SMTBVF состоит в том, что в этих программах произведена *симметризация* области аппроксимации. Поясним, что это означает.

Для воспроизведения сеточной функции двух переменных на прямоугольнике  $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$ , где  $0 \leq i < n$ ,  $0 \leq j < m$ , необходимы значения функции в  $(p+2)^2$  точках (см. [1, и, 13]). Пусть гладкость равна 1, тогда для воспроизведения будет использоваться область сетки, состоящая из 9 точек, как показано на рисунке.



Функцию по этим точкам можно вычислять в прямоугольнике 1 (параметр  $S$ , который определяет точки сетки, используемые при локальном воспроизведении, равен 0) и в прямоугольнике 2 ( $S = 1$ ) [1, и]. Таким образом, видно, что для нечетной степени гладкости область аппроксимации всегда несимметрична, как бы мы ни выбирали параметр  $S$ . Следовательно, возникает зависимость *запаздывающая* или *опережающая* (см. пример 1 в § 6.3). Симметризация же позволяет ликвидировать этот недостаток. Функция вычисляется два раза в каждой точке, при  $S=0$  и  $S=1$ , и берется полусумма двух результатов (здесь учитываются уже не 9 точек, а 14). Потери времени при этом не слишком большие, а качество аппроксимации улучшается.

Программа SMTBVF(LP, LX, LY, X, Y, Z, N, U, V, W) воспроизводит с гладкостью 0, 1, 2 или 3 функцию двух переменных, заданную на прямоугольной неравномерной сетке, в точках, которые могут произвольным образом располагаться внутри прямоугольника, обрамляющего сетку. Программа имеет следующие параметры:

LP — целое число от нуля до трех, задающее гладкость воспроизведения;

$LX, LY$  — число точек сетки по осям  $X$  и  $Y$  ( $LX \geq LP + 2, LY \geq LP + 2$ );

$X, Y$  — массивы координат точек сетки по осям  $X$  и  $Y$ , расположенных в возрастающем порядке;

$Z$  — двумерный массив значений функции в узлах сетки (размером  $(LX, LY)$ );

$N$  — число точек, в которых выполняется функция ( $N \geq 1$ );

$U, V$  — массивы  $X$ - и  $Y$ -координат точек, в которых выполняется функция;

$W$  — массив значений функции в точках  $(U(I), V(I))$ ,  $I = 1, 2, \dots, N$  (длины  $N$ ).

В программе параметр  $W$  является выходным.

Программа  $SMTSIM(LX, LY, X, Y, Z, N, U, V, W)$  выполняет со степенью гладкости единица сеточную функцию двух переменных, заданную на прямоугольной сетке. Совокупность точек, в которых определяется функция, может располагаться внутри прямоугольника, обрамляющего сетку, произвольным образом. В программе используется симметризация. Параметры программы имеют то же значение, что и одноименные параметры предыдущей программы.

Программа  $SMTONE(LX, LY, X, Y, Z, MX, MY, NU, NV, U, V, W)$  выполняет с гладкостью единица функцию двух переменных, заданную на прямоугольной неравномерной сетке. Выполнение производится в точках новой сетки, полученной делением интервалов между каждой парой соседних точек исходной сетки по осям  $X$  и  $Y$  на равные отрезки. Параметры программы следующие:

$LX, LY$  — число точек исходной сетки по осям  $X$  и  $Y$  ( $LX \geq 3, LY \geq 3$ );

$X, Y$  — массивы координат точек исходной сетки по осям  $X$  и  $Y$ , расположенных в возрастающем порядке;

$Z$  — двумерный массив значений функции в узлах исходной сетки (размером  $(LX, LY)$ );

$MX, MY$  — число подынтервалов между каждой парой соседних точек исходной сетки по осям  $X$  и  $Y$ ;

$NU$  — число точек новой сетки по оси  $X$ , равное  $(LX-1) * MX+1$ ;

$NV$  — число точек новой сетки по оси  $Y$ , равное  $(LY-1) * MY+1$ ;

$U, V$  — массивы  $X$ - и  $Y$ -координат новой сетки;

$W$  — двумерный массив значений функции в узлах новой сетки (размером  $(NU, NV)$ ).

В программе параметры  $U, V$  и  $W$  являются выходными.

В программе  $SMTBVF$  используется служебная программа OPER.

Программа OPER(LRP1, K, X, RZ) вычисляет коэффициенты (массив RZ длины LRP1) разностных отношений определенного вида (см. [1, и]), заданных на K-м элементе сетки, описываемой массивом X.

## 6.2. Восполнение функций двух переменных по методу Х. Акима

Пусть на плоскости  $XY$  определена прямоугольная область  $[a, b] \times [c, d]$ , а на ней двумерная прямоугольная сетка, заданная декартовым произведением двух одномерных сеток  $\{x_k\}$ ,  $k = 0, \dots, n$  и  $\{y_l\}$ ,  $l = 0, \dots, m$ . Восполнение сеточной функции  $Z(x_k, y_l)$  внутри прямоугольников  $[x_k, x_{k+1}] \times [y_l, y_{l+1}]$ , где  $0 \leq k < n$ ,  $0 \leq l < m$ , производится с помощью бикубических многочленов вида

$$p_3(x, y) = \sum_{i,j=0}^3 a_{ij} x^i y^j,$$

причем результат такого восполнения будет гладким (т. е. непрерывны сама функция  $p_3(x, y)$  и ее первые частные производные) не только внутри прямоугольников сетки, но и в узлах сетки, а также вдоль сторон прямоугольников сетки, т. е. во всей области аппроксимации. Более подробные сведения о методе можно найти в [14].

Этот метод восполнения реализован в программах ITPLBV и SFCFIT, которые отличаются друг от друга заданием исходных данных и видом получаемых результатов. SFCFIT вычисляет значения функции на сетке более мелкой, чем исходная, а ITPLBV находит значения функции в заданной совокупности точек, принадлежащих области.

Программа ITPLBV(LX, LY, X, Y, Z, N, U, V, W) восполняет функцию двух переменных  $Z(X, Y)$ , заданную на прямоугольной неравномерной сетке, в указанной совокупности точек, принадлежащих области. Программа имеет следующие параметры:

LX, LY — число точек сетки по осям  $X$  и  $Y$  ( $LX \geq 2$ ,  $LY \geq 2$ );

X, Y — массивы координат точек сетки по осям  $X$  и  $Y$ , расположенных в возрастающем порядке;

Z — двумерный массив значений функции  $Z(X, Y)$  в узлах сетки (размером (LX, LY));

N — число точек, в которых восполняется функция  $Z(X, Y)$ ,  $N \geq 1$ ;

U, V — массивы  $X$ - и  $Y$ -координат точек, в которых восполняется функция  $Z$ ;

W — массив значений функции в точках  $(U(I), V(I))$ ,  $I = 1, 2, \dots, N$ , (длины N).

В программе параметр  $W$  является выходным.

Программа  $SFCFIT(LX, LY, X, Y, Z, MX, MY, NU, NV, U, V, W)$  выполняет функцию двух переменных  $Z(X, Y)$ , заданную на прямоугольной неравномерной сетке, в узлах новой сетки, полученной из исходной делением интервалов между каждой парой соседних точек исходной сетки по осям  $X$  и  $Y$  на равные отрезки. Программа имеет следующие параметры:

$LX, LY$  — число точек исходной сетки по осям  $X$  и  $Y$ ;

$X, Y$  — массивы координат точек исходной сетки по осям  $X$  и  $Y$ , расположенных в возрастающем или убывающем порядке;

$Z$  — двумерный массив значений функции  $Z(X, Y)$  в узлах сетки (размером  $(LX, LY)$ );

$MX, MY$  — число подынтервалов между каждой парой точек исходной сетки по осям  $X$  и  $Y$  ( $MX \geq 2, MY \geq 2$ );

$NU$  — число точек новой сетки по оси  $X$ , равное  $(LX-1) * MX+1$ ;

$NV$  — число точек новой сетки по оси  $Y$ , равное  $(LY-1) * MY+1$ ;

$U, V$  — массивы  $X$ - и  $Y$ -координат новой сетки;

$W$  — двумерный массив значений функции в узлах новой сетки (размером  $(NU, NV)$ ).

В программе параметры  $U, V, W$  являются выходными.

### 6.3. Примеры

1. На рис. 6.1 изображена поверхность, которая восполнялась программами: а)  $SMTSIM$  (с симметризацией); б)  $SMTBVF$  (без симметризации). Заметен эффект «запаздывания». В программе рисования восполнение производилось следующей последовательностью инструкций:

```
DIMENSION X(11), Y(9), Z(11, 9), U(2091),  
* V(2091), W(2091)
```

```
CALL SMTSIM(11, 9, X, Y, Z, 2091, U, V, W)
```

```
CALL SMTBVF(1, 11, 9, X, Y, Z, 2091, U, V, W)
```

2. На рис. 6.2 приведены проекции поверхности, описываемой функцией, восполнение которой производилось с разными степенями гладкости: 1 (рис. 6.2, а) и 3 (рис. 6.2, б).

3. На рис. 6.3 изображены проекции поверхности, заданной на сетке  $11 \times 9$  и восполненной на сетке  $51 \times 41$  программами:  $ITPLBV$  (рис. 6.3, а) и  $SMTBVF$  с гладкостью 1 (рис. 6.3, б). На участках

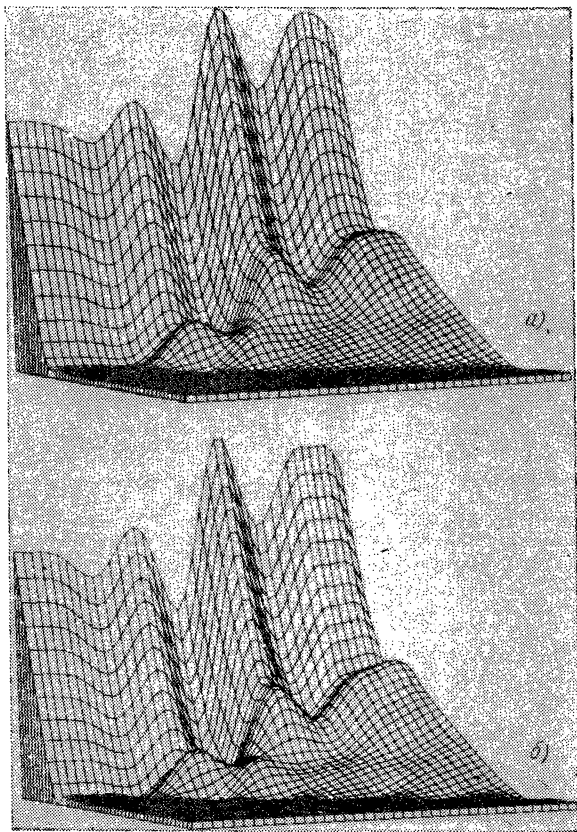


Рис. 6.1. Изображение поверхности, которая восполнялась с гладкостью 1: а) с симметризацией области аппроксимации; б) без симметризации.



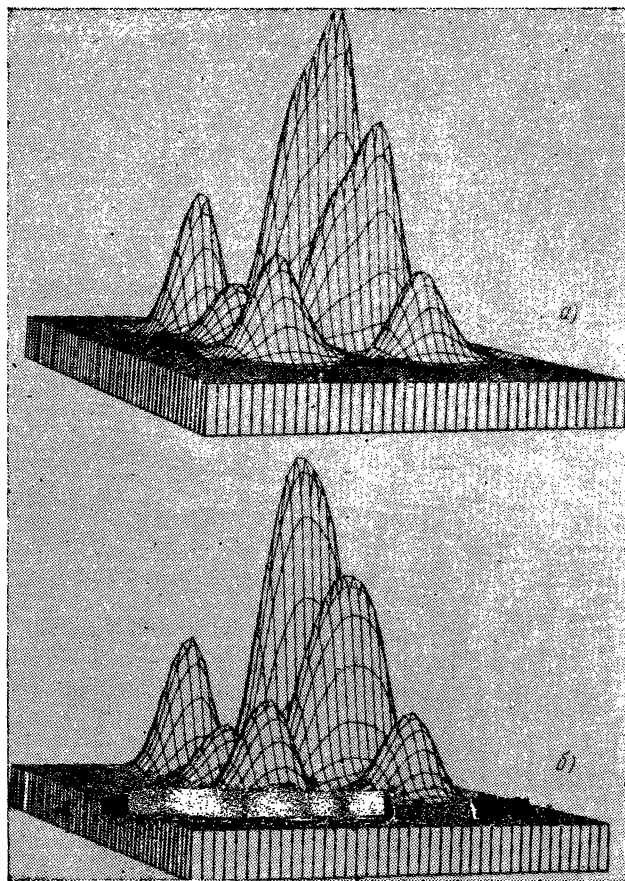


Рис. 6.2. Проекция поверхности, восполненной с разными степенями гладкости: а) с гладкостью 1, б) с гладкостью 3.

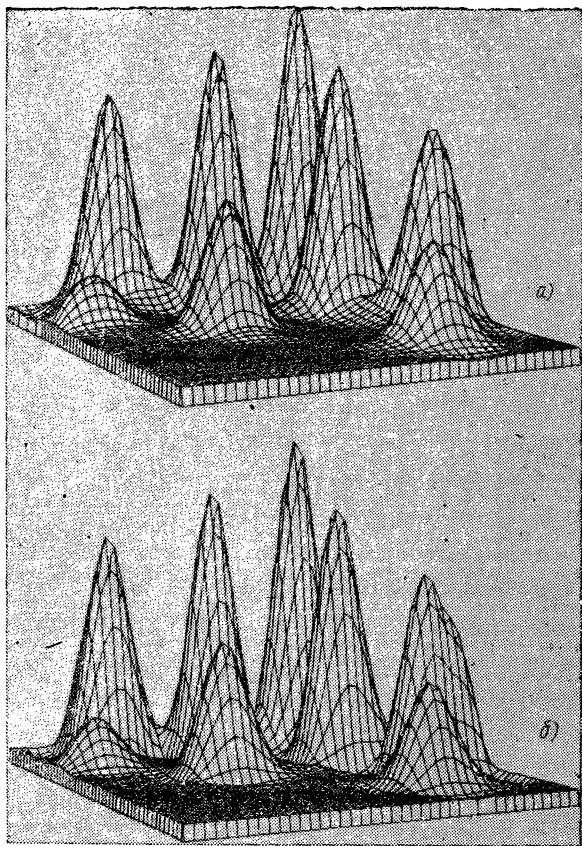


Рис. 6.3. Изображения поверхности, восполненной: а) по методу Х. Акима; б) по методу В. С. Рябенского с гладкостью 1,

резких перепадов значений функции заметно отличие выполненных значений.

4. Программы восполнения широко применяются при построении изолиний и проекций поверхностей. Примером может служить

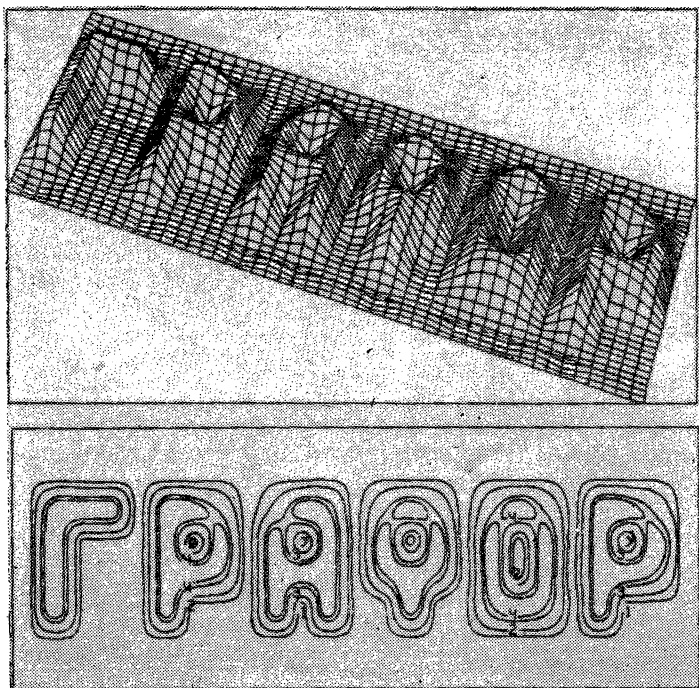


Рис. 6.4. Поверхность и соответствующая ей карта изолиний. При их построении использовались программы восполнения.

рис. 6.4, на котором изображена некоторая «искусственная» поверхность, а также соответствующие ей линии уровня. В последующих главах можно найти дополнительные примеры применения программ восполнения функций двух переменных.

## 7. ПОСТРОЕНИЕ ЛИНИЙ УРОВНЯ ФУНКЦИЙ ДВУХ ПЕРЕМЕННЫХ

Карты изолиний для целей исследования функций двух переменных широко применялись еще в «домашинный» период. В сущности задача сводится к изображению линий  $f(x, y) = c$ , где  $f(x, y)$  — исследуемая функция, а  $c$  — константа. При построении изолиний функция чаще всего задается не аналитически, а некоторым множеством значений. При сеточном задании эти значения известны в узлах сетки.

Мы будем иметь дело с функциями, которые заданы в узлах прямоугольной сетки. При построении изолиний функция доопределяется либо всюду (см. гл. 6), либо только на ребрах прямоугольных ячеек, из которых складывается сетка. В последнем случае используется линейная интерполяция.

Процесс построения изолинии разбивается на три этапа: поиск изолинии (определение начальной точки на изолинии), отслеживание изолинии (нахождение на изолинии последовательности точек) и, наконец, оформление изолинии (вписывание значений в разрывы изолиний, нумерация, расстановкаберг-штрихов).

В настоящей главе описаны три варианта программ. В первом варианте (§ 7.1) используется сплайн-интерполяция (см. гл. 6), и его целесообразно применять там, где сетка значений редкая. Разумеется, расход машинного времени в этом варианте будет заметно большим, чем в двух других (§§ 7.2 и 7.3). Два других варианта не имеют принципиальных отличий. В обоих случаях используется линейная интерполяция на ребрах ячейки. Однако третий вариант экономнее в отношении памяти и имеет более широкие оформительские средства.

Описываемые варианты программ позволяют строить карты изолиний не только в декартовой системе координат, но и в произвольной системе координат, однозначно связанной с декартовой. Однако делают они это по-разному. В первом и втором вариантах линии уровня сначала строятся на плоскости в произвольной системе координат точно так же, как в случае декартовой системы, а затем производится отображение этой плоскости вместе с изолиниями в декартову систему координат. В отличие от этого, в

третьем варианте программ недекларованное пространство сначала преобразуется в декартово, а в нем уже линии уровня строятся обычным образом.

## 7.1. Построение изолиний с использованием гладкого восполнения функций

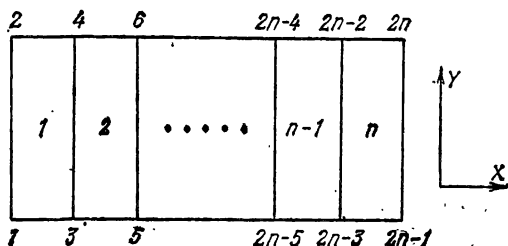
В этом параграфе рассматривается метод построения изолиний для функций, заданных таблично на прямоугольной неравномерной сетке в произвольной системе координат, однозначно связанной с-декартовой (см. также [1, и]). Предполагается, что сеточная функция является достаточно гладкой, например, получена как результат восполнения одним из способов, описанных в гл. 6.

**7.1.1. Метод поиска и отслеживания изолиний.** Пусть задана функция  $Z(X, Y)$ , вычисляемая каким-либо способом в любой точке прямоугольной области, искомые изолинии будут решением уравнения

$$Z(X, Y) = C. \quad (7.1)$$

Если функция в заданной области не имеет локальных экстремумов, то решением будут кривые, пересекающие границу области. И если найти корни уравнения (7.1) на границе области, то, отслеживая изолинии, начинающиеся в найденных точках, мы получим всю карту изолиний. Эта идея и лежит в основе описываемого метода.

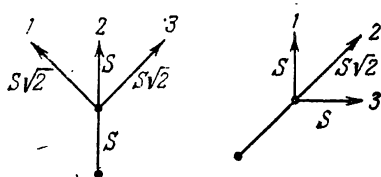
Пусть теперь функция, определенная на прямоугольной области, имеет локальные экстремумы. Чтобы провести линии уровня, заданные уравнениями (7.1), мы сводим задачу к предыдущей, т. е. разбиваем область на прямоугольные окна, в которых функция уже не имеет локальных экстремумов, или, говоря точнее, локальные экстремумы в заданных окнах несущественны для картины рельефа функции во всей области. Проведем разбиение так, как показано на рисунке, учитывая, что количеством окон можно управлять.



Поиск корней уравнения (7.1) на границе окон осуществляется делением сторон окон на части. Каждая часть проверяется на

наличие корня, и если корень есть, то он вычисляется методом деления пополам с достаточной точностью. Таким образом, ищутся корни на границах каждого окна. После того как корни найдены, проводятся изолинии с начальными точками в этих корнях, а чтобы не было повторного отслеживания изолинии, запоминаются их точки выхода на границу окон. При выборе корня для следующей изолинии проверяется его близость к точкам выхода. Если оказалось, что корень близок к точке выхода, то он отбрасывается и рассматривается следующий.

Опишем теперь алгоритм отслеживания изолиний. Пусть нам известен корень уравнения (7.1). Начальное направление линии выбирается ортогональным к стороне окна, на которой найден корень. Пусть шаг ведения линии равен  $S$ . Следующая точка линии выбирается исходя из условия минимума отклонения функции от заданного значения  $C$  в точках 1, 2, 3:



Такой метод, вообще говоря, непригоден для функций, линии уровня которых имеют изломы, но, уменьшая шаг  $S$ , можно добиться любого приближения к идеальной изолинии. В нашем же случае, когда сеточная функция восполняется гладкой, от этого метода следует ожидать хороших результатов.

**7.1.2. Описание программ.** Описанный выше метод построения карт изолиний реализован в двух программах CONDEK и DRACON. Программа CONDEK строит линии уровня функции двух переменных, заданной на декартовой прямоугольной сетке с неравномерным шагом и восполняемой гладким образом программой ITPLBV (см. п. 6.2). Программа CONDEK может нарисовать изолинии  $N$  уровней, причем уровни могут задаваться в произвольном порядке. Имеется также аппарат маркировки изолиний, т. е. в разрывы изолиний вписывается номер уровня в порядке их расположения во входном массиве. При этом контролируется, чтобы номера разных уровней не накладывались друг на друга.

Программа DRACON также рисует изолинии функций двух переменных, но не только на декартовой сетке, а, например, на полярной, логарифмической или какой-либо другой. Поэтому в эту программу введены два параметра, которыми должны быть имена подпрограмм-функций, описанных в операторе EXTERNAL. Эти подпрограммы-функции должны преобразовывать входные координаты в декартовы. Например, в случае полярной сетки две подпро-

граммы-функции преобразования полярных координат в декартовы будут иметь следующий вид:

FUNCTION XPOL(RO, FI)	FUNCTION YPOL(RO, FI)
XPOL=RO * COS(FI)	YPOL=RO * SIN(FI)
RETURN	RETURN
END	END

Параметры  $K$  в программе DRACON и  $KO$  в программе CONDEK имеют разный смысл. Оба параметра управляют разбиением области на окна, но параметр  $K$  задает не количество окон в области, а количество полос исходной сетки, которые будут рассматриваться в качестве одного окна области рисования. Это различие вызвано тем, что при использовании большинства преобразований сетка деформируется нелинейно, и, следовательно, выбор шага ведения изолинии в исходных координатах будет отнимать довольно много времени. Если же за окна принять полосы сетки, то выбор шага сильно упростится, что приведет к сокращению времени вычисления. Маркировка изолиний в программе DRACON производится аналогично тому, как это делается в CONDEK.

При использовании программы DRACON возникает необходимость в определении пределов изменения координат сетки, преобразованной функциями FUNX и FUNY, а также в очерчивании границ области рисунка, в которой изображаются изолинии. Для этой цели предназначены соответственно программы DRALIM и DRFRAM.

Программа CONDEK(LX, LY, X, Y, Z, STEP, KO, N, C) предназначена для вычерчивания изолиний функции, заданной на прямоугольной сетке в декартовой системе координат и выполняемой программой ITPLBV. Программа имеет следующие параметры:

LX, LY — число точек сетки по осям  $X$  и  $Y$ ;

$X$ ,  $Y$  — массивы координат точек сетки по осям  $X$  и  $Y$ , расположенные в возрастающем порядке;

$Z$  — двумерный массив размером (LX, LY), задающий значения функции  $Z(X, Y)$  в узлах сетки;

STEP — шаг отслеживания изолинии (в выбранных единицах измерения);

KO — целое положительное число, задающее количество окон, на которое делится область;

$N$  — количество уровней;

$C$  — одномерный массив уровней.

Программа DRACON(LX, LY, X, Y, Z, STEP, K, N, C, FUNX, FUNY) вычерчивает изолинии функции  $Z(X, Y)$ , заданной на произвольной сетке и выполняемой программой ITPLBV. Преобразование сетки в декартову производится с помощью подпрограмм-

функций FUNX и FUNY. Программа имеет следующие параметры:

K — число полос сетки, рассматриваемых в качестве одного окна ( $1 \leq K \leq LX-1$ );

FUNX, FUNY — подпрограммы-функции преобразования сетки (X, Y) в декартову (U, V) (соответствие сеток устанавливается следующим образом:  $U = \text{FUNX}(X, Y)$ ,  $V = \text{FUNY}(X, Y)$ ; имена этих подпрограмм-функций должны быть описаны в операторе EXTERNAL).

Остальные параметры аналогичны параметрам программы CONDEK.

З а м е ч а н и я и р е к о м е н д а ц и и.

1. При вызове программы CONDEK нет смысла задавать большое количество окон (больше LX), так как это потребует дополнительных затрат процессорного времени. По той же причине не рекомендуется задавать параметр STEP меньшим 0,1 мм.

2. Преобразование сетки в декартову не должно сильно отличаться от изометрического, линейного или аффинного, иначе могут возникнуть побочные эффекты, приводящие к искажению картинки.

3. Для того чтобы воспользоваться программой гладкого восполнения SMTSIM вместо программы ITPLBV, достаточно вложить в функциональную часть пакета следующую подпрограмму:

```
SUBROUTINE ITPLBV(LX, LY, X, Y, Z, N, U, V, W)
CALL SMTSIM(LX, LY, X, Y, Z, N, U, V, W)
RETURN
END
```

Если же потребуется восполнить функцию с разными степенями гладкости — 0, 1, 2, 3 — и без симметризации, то можно использовать следующую подпрограмму:

```
SUBROUTINE ITPLBV(LX, LY, X, Y, Z, N, U, V, W)
CALL SMTBVF(L, LX, LY, X, Y, Z, N, U, V, W)
RETURN
END
```

где вместо L должно быть одно из значений — 0, 1, 2 или 3. Однако при этом не следует забывать о затратах процессорного времени на восполнение.

Программа DRALIM(LX, LY, X, Y, FUNX, FUNY, R) предназначена для определения пределов UMIN, UMAX, VMIN, VMAX изменения координат сетки, преобразованной функциями FUNX и FUNY, и имеет следующие параметры:

LX, LY — число точек сетки по осям X и Y;

X, Y — массивы координат точек сетки по осям X и Y;



FUNX, FUNY — подпрограммы-функции преобразования сетки (X, Y) в декартову (U, V) (соответствие сеток устанавливается следующим образом:  $U = \text{FUNX}(X, Y)$ ,  $V = \text{FUNY}(X, Y)$ );

R — параметр, вычисляемый программой;  $R = (\text{VMAX} - \text{VMIN}) / (\text{UMAX} - \text{UMIN})$ .

Значения пределов помещаются в общий блок DRLM, описываемый следующим образом: COMMON/DRLM/UMIN, UMAX, VMIN, VMAX.

Программа DRFRAM(LX, LY, X, Y, FUNX, FUNY) служит для очерчивания границ области, в которой рисуются изолинии с помощью программы DRACON. Ее параметры имеют тот же смысл, что и в программе DRALIM.

Служебные программы.

Программа INSDEK(LX, LY, X, Y, Z) используется программой CONDEK для вычерчивания изолиний в окне (без преобразования координат).

Программа INSIDE(LX, LY, X, Y, Z, FUNX, FUNY) используется программой DRACON для вычерчивания изолиний в окне (с преобразованием координат).

Программа FNROOT(LX, LY, X, Y, Z, NROOTS, ROOTS) находит корни уравнений:  $Z(X_0, Y) = C$ ,  $Z(X, Y_0) = C$ .

### 7.1.3. Примеры.

1. На рис. 7.1 приведены изолинии функции, заданной на прямоугольной сетке в декартовой системе координат. Далее приводится программа, с помощью которой выполнялось построение.

```
DIMENSION Z(6, 6), X(6), Y(6), C(10)
```

```
DATA Z/0., 5., 5*0., 10., 5., 0., 0., 10., 15., 0., 10.,
```

```
* 0., 5., 0., 15., 0., 30., 0., 10., 0., 0., 10., 0.,
```

```
* 0., 30., 7*0./
```

```
DATA C/-0.1, 0.1, 3., 5., 10., 15., 20., 25., 29./
```

```
DATA X/1., 2., 3., 4., 5., 6./, Y/1., 2., 3., 4., 5., 6./
```

```
CALL PAGE(17., 25., 0, 0, 1)
```

```
CALL REGION(0., 0., 17., 25., 0, 0, 0)
```

```
CALL LIMITS(1., 6., 1., 6.)
```

```
CALL CONDEK(6, 6, X, Y, Z, 0.0125, 11, 9, C)
```

```
CALL ENDPG('7.1')
```

```
END
```

2. На рис. 7.2 изображены линии уровня функции, заданной на сетке в полярной и декартовой системах координат. Ниже приводится программа рисования. Двумерный массив, описывающий поверхность, формируется с помощью программы READZ.

```
DIMENSION X(11), Y(9), Z(11, 9), C(10)
```

```
DATA C/0.1, 1., 5., 10., 20., 30., 40., 50., 60./
```

```
EXTERNAL XPOL, YPOL
```

```
T=ATAN(1./60.)
```

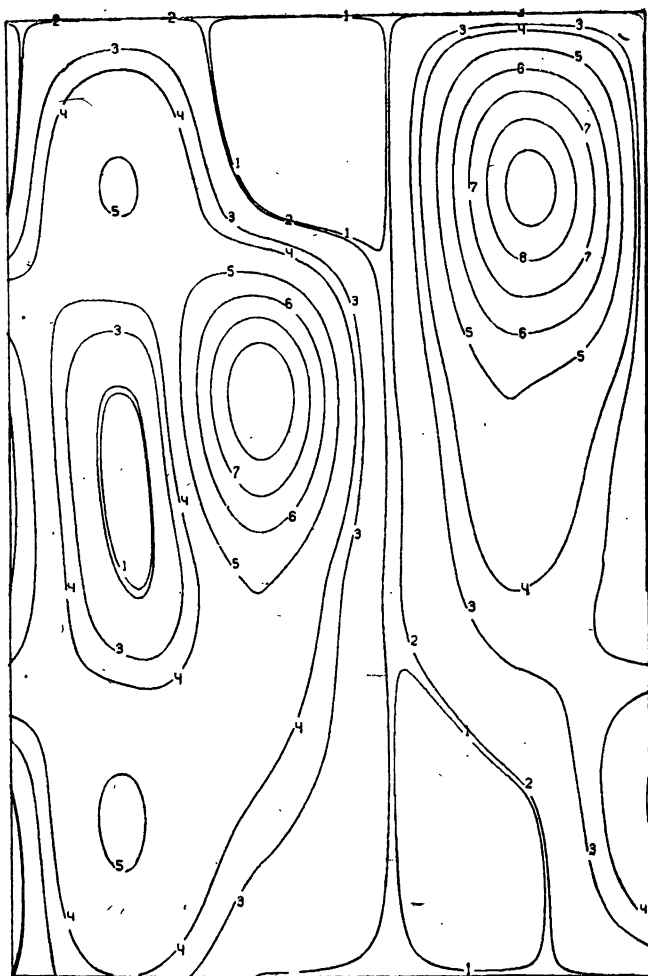


Рис. 7.1. Линии уровня функции двух переменных, заданной на прямоугольной сетке в декартовой системе координат.

```

DO 1 I=1, 11
  S=5.*I
  IF(I.LE.9) Y(I)=T*(S-5.)
1 X(I)=S
  CALL READZ(Z)
  CALL PAGE(25., 17., 0, 0, 0)
  CALL DRALIM(11, 9, X, Y, XPOL, YPOL, RR)
  CALL REGION(0., 0., 25., RR*25., 0, 0, 0)

```

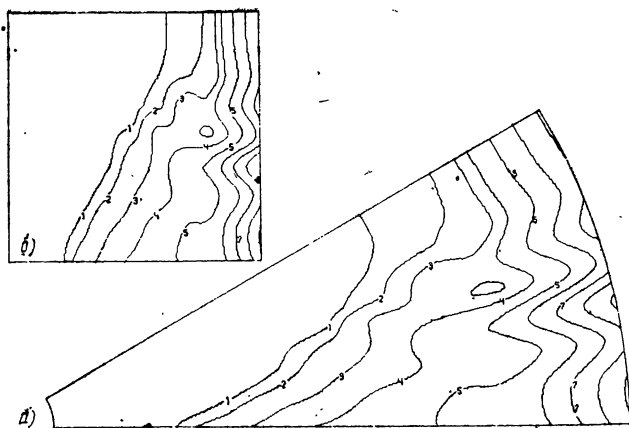


Рис. 7.2. Линии уровня функции, заданной на сетке: а) в полярной системе координат, б) в декартовой системе координат.

```
CALL LIMITS(0., 55., 0., RR*55.)
CALL DRFRAM(11, 9, X, Y, XPOL, YPOL)
CALL DRACON(11, 9, X, Y, Z, 0.05, 1, 9, C, XPOL,
* YPOL)
CALL REGION(0.5, 6.5, 10., 10., 0, 0, 4)
CALL LIMITS(X(1), X(11), Y(1), Y(9))
CALL CONDEK(11, 9, X, Y, Z, 0.05, 10, 9, C)
CALL ENDPG('7.2')
END
```

## 7.2. Построение изолиний с использованием линейной интерполяции

В этом параграфе описываются программы, позволяющие строить линии уровня для функции двух переменных. Исходная функция задается прямоугольным массивом чисел на сложной области в произвольной системе координат, однозначно отображаемой на декартову. Изолинии могут быть помечены порядковым номером и значением уровня.

**7.2.1. Алгоритм построения изолиний.** Данный алгоритм, реализуемый программой ISOLIN, позволяет строить линии уровня (изолинии) для функции двух переменных  $Z = Z(X, Y)$ , заданной в узлах прямоугольной неравномерной сетки  $\{X_i, Y_j\}$ ,  $i = 1, \dots, NX$ ,  $j = 1, \dots, NY$ . Двумерная сетка разбивает область определения функции на прямоугольные ячейки  $(X_{i-1} \leq X \leq X_i$ ,

$Y_{j-1} \leq Y \leq Y_j$ ). Если доопределить функцию  $Z(X_i, Y_j)$  на ребрах ячейки, используя линейную интерполяцию, тогда линиями уровня будут ломаные, проходящие через точки пересечения отрезков функции, заданных на ребрах ячеек, с плоскостями  $Z = \text{CONT}_k$ ,  $k = 1, \dots, \text{NCN}$ .

Различают *незамкнутые* изолинии, начинающиеся и заканчивающиеся на границе области, и *замкнутые* линии, лежащие целиком внутри области определения функции. Задача построения изолиний решается следующим образом. Для обнаружения начальных точек незамкнутых изолиний осуществляется обход по границе. Как только начальная точка обнаружена, изолиния отслеживается до конца, т. е. до выхода на границу. После того как построены все незамкнутые изолинии, производится последовательный просмотр всех горизонтальных ребер ячеек для выявления точки, принадлежащей замкнутой изолинии. После обнаружения такой точки, изолиния отслеживается до конца, т. е. до возврата в эту начальную точку. Как только построены все линии, соответствующие уровню  $\text{CONT}_k$ , производится переход к следующему,  $\text{CONT}_{k+1}$ , уровню и процедура повторяется.

Чтобы исключить повторное проведение изолиний, в процессе отслеживания регистрируется факт ее прохождения через данное ребро. При этом считается, что через каждое ребро нельзя провести более одной линии заданного уровня. Информация о пройденных ребрах сохраняется и используется для определения конца замкнутых изолиний.

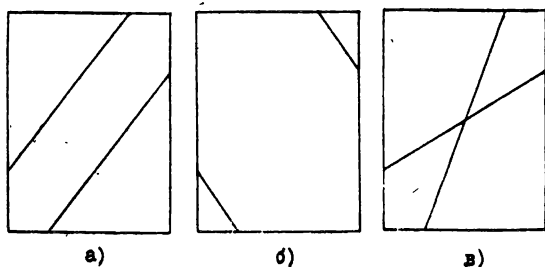
При обнаружении начальной точки рассматривается поведение линии внутри ячейки. Если линия вошла в ячейку, она должна выйти из нее через одно из оставшихся трех ребер. Проверка соотношения

$$(Z(a) - \text{CONT}_k) \times (Z(b) - \text{CONT}_k) \leq 0,$$

где  $a, b$  — граничные точки ребра, определяем через какое ребро линия вышла. Далее рассматривается ячейка, соседняя с этим ребром, и операция повторяется. Координаты точки пересечения линии уровня с ребром находятся с помощью линейной интерполяции. Эти координаты запоминаются в некотором массиве. Линия вычерчивается, когда достигнута граница области (либо в случае замкнутых линий ранее пройденное ребро) или когда массив заполнен.

При поиске линий уровня на сетке могут встретиться вырожденные ситуации: а) возможно проведение линии уровня через все ребра ячейки и б) линия проходит точно через вершину ячейки,

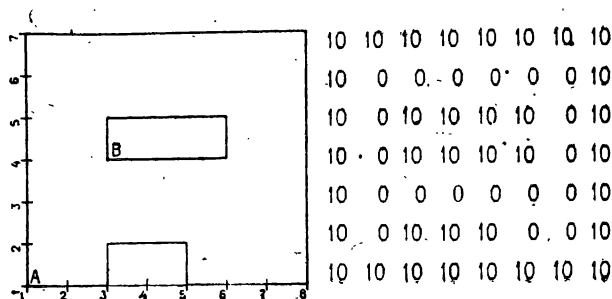
В первом случае возможны три способа соединения точек пересечения линий уровня с ребрами:



Поскольку значения функции известны только в четырех вершинах ячейки, нельзя сказать, какая из конфигураций правильна. Описываемые далее программы в такой ситуации выбирают вариант а). Вырождение второго вида приводит либо к повторению линии дважды, либо к ее обрыву. Чтобы избежать этого, к вырожденному узлу добавляется малое значение  $10^{-5}$ . В результате вырождение устраняется, вызывая незаметную для глаза ошибку в изолинии. Заметим, что для построения визуально более качественного изображения линий уровня следует задать более частую сетку. Можно, например, воспользоваться методами гладкого восполнения функций, изложенными в гл. 6.

Информация о форме области задается ее *описателем* — двумерным массивом ITAGB. Заметим, что эта область может иметь довольно сложную конфигурацию (см. пример 2 в п. 7.2.4). Определяется она следующим образом:  $ITAGB(I, J) = 0$ , если точка с индексами  $(I, J)$  принадлежит области, и  $ITAGB(I, J) = 10$ , если точка находится на границе или не принадлежит области. Граница области задается порядковыми номерами ее граничных точек на сетке, хранящихся в массивах  $IXBD(K)$ ,  $IYBD(K)$ ,  $K = 1, \dots, NB$ .

На рисунке приведен пример сложной области и ее описателя ITAGB.



Задание массивов IXBD и IYBD зависит от порядка обхода граничных точек. Например, если начать обход из точки А и сделать фиктивный разрез через точки А и В, то

К	1	2	3	4	5	...	29	30	31	32	...	38
IXBD(K)	1	2	3	3	4	...	1	3	3	4	...	3
IYBD(K)	1	1	1	2	2	...	1	4	5	5	...	4

В этих массивах первая точка является и последней точкой. Кроме того, следует иметь в виду, что заданная область при обходе должна всегда оставаться слева.

Таким образом, область, на которой рисуются линии уровня, описывается дважды: массивом ITAGB и массивами граничных точек IXBD и IYBD. Если заданы массивы граничных точек, то массив ITAGB можно подготовить с помощью программы BITA.

При построении карт изолиний чаще всего будет использоваться простая прямоугольная область. Для задания такой области служит программа QUADRA.

Линии уровня можно строить и в произвольной системе координат. С этой целью в программе вычерчивания изолиний и в некоторых других предусмотрен параметр ICM. Если  $ICM = 0$ , то предполагается декартова система координат, а при  $ICM = 1$  — полярная. Если же необходимо работать в какой-либо другой системе координат, то необходимо установить  $ICM = 1$  и в функциональную часть программы добавить подпрограммы-функции XPOL и YPOL со своими преобразованиями координат (см. ниже пример 3 в п. 7.2.4).

Для определения пределов изменения функции, заданной в недекартовой системе координат, предназначена программа EXTREM.

Для формирования массива уровней CONT предусмотрены следующие возможности. В случае, если неизвестны пределы изменения функции, для которой строятся изолинии, с помощью программы LEVFUN можно определить абсолютные максимум и минимум функции и построить массив уровней с равным шагом между ними. Другая программа, LEVMAP, задает массив уровней относительно нулевого уровня по положительному и отрицательному шагам. Кроме того, иногда полезно наряду с основными изолиниями построить также и промежуточные, используя другой тип линии. В этом случае для подготовки массива уровней удобно воспользоваться программой ADDLEV.

При вычерчивании изолиний бывает необходимо различать их. Для этого можно использовать программы, предоставляющие воз-

возможность рисовать различные типы линий или помечать их порядковым номером или значением уровня изолинии.

Тип линии определяется установочными программами FULL (сплошная линия) и BROKEN (штриховая и штрихпунктирная линии), вызов которых должен предшествовать обращению к программе ISOLIN. Описание этих программ приведено в § 4.2.

Две другие программы SIZLIN и NUMLIN позволяют задать соответственно режимы вписывания в разрыв изолинии значения или порядкового номера уровня. Отменить режим вписывания позволяет программа RENUM.

При работе в декартовой системе координат наряду с выводом изолиний бывает необходимо выводить другие линии. В этом случае следует воспользоваться программой SUPLIN, которая учит заданное преобразование координат. Кроме того, линия может быть помечена маркером, рисоваться сплошной или прерывистой и т. д.

#### 7.2.2. Описание программ.

Программа ISOLIN(NX, NY, X, Y, Z, ITAGB, NB, IXBD, IYBD, NCN, CONT, ICM) позволяет начертить изолинии непрерывной функции двух переменных. Программа имеет следующие параметры:

NX, NY — число точек сетки по осям X и Y;

X, Y — массивы координат точек по осям X и Y;

Z — массив значений функции (размером (NX, NY));

ITAGB — описатель области (массив размером (NX, NY));

|NB| — число граничных точек (если  $NB > 0$ , область не очерчивается, если  $NB < 0$  — очерчивается);

IXBD — массив порядковых номеров граничных точек вдоль оси X (длины NB);

IYBD — массив порядковых номеров граничных точек вдоль оси Y (длины NB);

|NCN| — количество линий равного уровня (если  $NCN < 0$ , то при большой густоте линий применяется экранирование, в результате чего часть линий не проводится);

CONT — массив значений уровней;

ICM — признак системы координат:  $ICM = 0$  — декартова система координат,  $ICM = 1$  — полярная система координат.

В этой программе все параметры входные.

Программа QUADRA(ITAGB, IXBD, IYBD, NX, NY, NB) позволяет по заданной сетке сформировать массивы описателей области и ее границы, определяя тем самым прямоугольную область, на которой будут вычерчиваться линии уровня. Параметры программы следующие:

ITAGB — описатель области (размером (NX, NY));

IXBD.— массив порядковых номеров граничных точек вдоль оси X (длины NB);

IYBD — массив порядковых номеров граничных точек вдоль оси Y (длины NB);

NX, NY — число точек сетки по осям X и Y;

NB — число граничных точек,  $NB=2*NX+2*NY-3$ .

В этой программе выходные параметры ITAGB, IXBD, IYBD.

Программа BITA(ITAGB, IXBD, IYBD, NX, NY, NB) формирует описатель области ITAGB по массивам граничных точек. Параметры этой программы аналогичны параметрам программы QUADRA. Выходным здесь является параметр ITAGB.

Программа LEVFUN(Z, NX, NY, CONT, NCN) предназначена для определения абсолютных экстремумов функции и построения массива уровней с равным шагом между экстремумами. Параметры программы следующие:

Z — массив значений функции (размером (NX, NY));

NX, NY — число точек сетки по осям X и Y;

CONT — массив значений уровней;

NCN — число уровней.

З а м е ч а н и е. CONT(1) равно максимальному значению Z, а CONT(NCN) — минимальному.

Программа LEVMAP(CONT, NCN, F0, SP, SN, NP) позволяет построить массив уровней относительно нулевого уровня по указанным шагам для положительных и отрицательных уровней. Параметры программы следующие:

CONT — массив значений уровней;

NCN — число уровней;

F0 — значение нулевого уровня;

SP — шаг задания положительных уровней;

SN — шаг задания отрицательных уровней;

NP — число положительных уровней.

Число отрицательных уровней равно  $NCN-NP-1$ .

Программа ADDLEV(CONT, NCN, IS, CAD, NAD) позволяет построить массив дополнительных уровней. Параметры программы следующие:

CONT — значения основных уровней;

NCN — число основных уровней;

IS — число дополнительных уровней между двумя основными;

CAD — массив дополнительных уровней (длины NAD);

NAD — полное число дополнительных уровней, равное  $IS* (NCN-1)$ .

Программа EXTREM(X, Y, NX, NY, XMN, XMX, YMN, YMX, ICM) предназначена для определения пределов изменения аргу-



ментов функции на области ее задания в недекартовой системе координат. Параметры программы:

$X, Y$  — массивы координат точек по осям  $X$  и  $Y$ ;

$NX, NY$  — число точек сетки по осям  $X$  и  $Y$ ;

$XMN, XMX$  — значения координаты  $X$ , соответствующие левой и правой границам;

$YMN, YMX$  — значения координаты  $Y$ , соответствующие нижней и верхней границам;

$ICM$  — признак системы координат:  $ICM=0$  — декартова система координат,  $ICM=1$  — полярная система координат.

Функция  $XPOL(R, T)$  позволяет по координатам точки в полярной системе координат вычислить  $X$ -координату в декартовой системе координат. Ее параметры:  $R$  — радиус точки в полярной системе координат,  $T$  — угол между осью абсцисс и радиусом.

Функция  $YPOL(R, T)$  позволяет по координатам точки в полярной системе координат вычислить  $Y$ -координату в декартовой системе координат. Ее параметры:  $R$  — радиус точки в полярной системе координат,  $T$  — угол между осью ординат и радиусом.

Программа  $NUMLIN(SIZE, N)$  задает режим вписывания в разрыв изолинии порядковых номеров уровней. Ее параметры указывают высоту цифры в выбранных единицах измерения ( $SIZE$ ) и количество цифр в номере уровня ( $N$ ).

Программа  $SIZLIN(SIZE, N, M)$  задает режим вписывания в разрыв изолинии значения уровня. Ее параметры указывают высоту цифры в выбранных единицах измерения ( $SIZE$ ), количество цифр в числе ( $N$ ) и количество дробных знаков ( $M$ ).

В программе  $ISOLIN$  используются служебные программы  $LOOK, CELL, LINT, ZAPIT$ .

Программа  $LOOK(NX, NY, X, Y, ITAGB, XX, YY, KT, ICM)$  отслеживает линию, следит за ее окончанием или за окончанием массивов  $XX$  и  $YY$  длины  $KT$ , в которых запоминаются точки пересечения линий уровня с ребрами.

Программа  $CELL(Z, NX, NY)$  позволяет найти ребро, через которое изолиния выходит из ячейки.

Программа  $LINT(X, Y, XX, YY, NX, NY, KT, ICM)$  позволяет произвести линейную интерполяцию на ребре.

Программа  $ZAPIT(ITAGB, NX, NY)$  фиксирует прохождения линии уровня через ребро.

**7.2.3. Вспомогательные программы.** В области можно нарисовать сетку, на которой задана функция  $Z$ . Это делается с помощью программы  $CMGRID$ . Линии сетки можно пометить как порядковым номером линии, так и соответствующим ей значением  $X$  или  $Y$ . Разметка линий производится по конечным значениям  $X$  и  $Y$ .

С помощью программы LOCEXT можно найти локальные экстремумы функций двух переменных в заданной области и нанести их на контурную карту. Поиск экстремумов производится перебором всех точек в области. Точка считается экстремальной, если функция принимает в ней значение большее (меньшее), чем значения функции во всех ближайших окружающих ее точках. Размер вписываемых цифр можно задать программой SIZLIN. По умолчанию этот размер устанавливается равным 0,3 см, и число выдается с одной цифрой после точки.

Иногда на карте изолиний бывает полезно отметить отдельные значения функции  $Z$ , так называемые *опорные точки*. Для этой цели предназначена программа KEYPO.

Программа CMGRID( $X, Y, NX, NY, MX, MY, SZ, KP, M, N, ICM$ ) позволяет построить сетку по значениям массивов координат  $X$  и  $Y$  и разметить ее. Разметка сетки производится по ее краям. Массивы  $X$  и  $Y$  могут быть заданы в произвольной системе координат, однозначно отображаемой на декартову. Параметры программы следующие:

$X, Y$  — массивы координат точек по осям  $X$  и  $Y$ ;

$|NX|$  — число точек сетки по оси  $X$  (если  $NX < 0$ , сетка по  $X$  не рисуется);

$|NY|$  — число точек сетки по оси  $Y$  (если  $NY < 0$ , сетка по  $Y$  не рисуется);

$MX$  — шаг сетки по  $X$  (чертится каждая  $MX * i + 1$  линия,  $i = 0, 1, 2, \dots$ );

$MY$  — шаг сетки по  $Y$  (чертится каждая  $MY * i + 1$  линия,  $i = 0, 1, 2, \dots$ );

$SZ$  — высота цифр (в выбранных единицах измерения);

$KP$  — количество требуемых дробных знаков в числе;

$M$  — шаг разметки по оси  $X$ ;

$M < 0$  — сетка метится значениями  $X$  с шагом  $|M|$ , т. е. метится каждый  $|M|$ -й узел сетки,

$M = 0$  — сетка не метится,

$M > 0$  — сетка метится порядковыми номерами линий с шагом  $|M|$ , т. е. метится каждый  $|M|$ -й узел сетки;

$N$  — шаг разметки по оси  $Y$ ;

$N < 0$  — сетка метится значениями  $Y$  с шагом  $|N|$ , т. е. метится каждый  $|N|$ -й узел сетки,

$N = 0$  — сетка не метится,

$N > 0$  — сетка метится порядковыми номерами линий с шагом  $|N|$ , т. е. метится каждый  $|N|$ -й узел сетки;

$ICM$  — признак системы координат:  $ICM = 0$  — декартова система координат,  $ICM = 1$  — полярная система координат.

Программа LOCEXT( $Z, X, Y, NX, NY, ICM$ ) позволяет найти в области определения функции  $Z$  локальные экстремумы этой

функции и написать их значения, пометив точки экстремумов маркером. Параметры программы следующие:

Z — массив значений функции (размером (NX, NY));

X, Y — массивы координат точек по осям X и Y;

NX, NY — число точек сетки по X и Y;

ICM — признак системы координат: ICM=0 — декартова система координат, ICM=1 — полярная система координат.

Если обнаружен максимум, то изображается маркер номер 3 (ромбик), если минимум — номер 14 (плюс в ромбике). Чтобы задать размер цифр в числе и их количество, необходимо обратиться к программе SIZLIN. По умолчанию высота цифр устанавливается равной 0,3 см, и число выдается с одной цифрой после точки.

Программа KEYPO(Z, X, Y, NX, NY, ISX, ISY, NM, SZ, KP, ICM) предназначена для того, чтобы отметить на графике значения функции двух переменных Z(X, Y) в избранных точках. Параметры программы следующие:

Z — массив значений функции (размером (NX, NY));

X, Y — массив координат точек по осям X и Y;

NX, NY — число точек сетки по X и Y;

ISX — шаг выдачи точек по X (отмечается каждая ISX-я точка);

ISY — шаг выдачи точек по Y (отмечается каждая ISY-я точка);

NM — номер маркера, которым метятся точки;

SZ — высота цифр в пометках;

KP — количество требуемых дробных знаков в числе;

ICM — признак системы координат: ICM=0 — декартова система координат, ICM=1 — полярная система координат.

Программа SUPLIN(X, Y, N, NM, JS, L, ICM) позволяет начертить линию по координатам точек в недекартовой системе координат. Параметры программы следующие:

X, Y — массивы абсцисс и ординат точек;

N — число точек;

NM — номер маркера (если NM<0, изображается маркер уменьшенных размеров);

|JS| — шаг маркировки: JS>0 — линия с маркерами, JS=0 — линия без маркеров, JS<0 — линия не проводится, вычерчиваются только маркеры;

L — признак линии: L=1 — сплошная замкнутая линия с маркерами, L=2 — сплошная незамкнутая линия с маркерами, L=1, 2 — сплошная или штрихпунктирная линия (в зависимости от установок FULL или BROKEN);

ICM — признак системы координат: ICM=0 — декартова система координат, ICM=1 — полярная система координат.

4	x <sup>7</sup>	x <sup>17</sup>	x <sup>9</sup>	x <sup>17</sup>	x <sup>13</sup>	x <sup>10</sup>	x <sup>12</sup>
4	x <sup>11</sup>	x <sup>15</sup>	x <sup>17</sup>	x <sup>29</sup>	x <sup>17</sup>	x <sup>13</sup>	x <sup>13</sup>
10	x <sup>19</sup>	x <sup>18</sup>	x <sup>14</sup>	x <sup>21</sup>	x <sup>21</sup>	x <sup>14</sup>	x <sup>12</sup>
15	x <sup>21</sup>	x <sup>22</sup>	x <sup>17</sup>	x <sup>24</sup>	x <sup>19</sup>	x <sup>15</sup>	x <sup>11</sup>
6	x <sup>23</sup>	x <sup>25</sup>	x <sup>27</sup>	x <sup>18</sup>	x <sup>13</sup>	x <sup>14</sup>	x <sup>13</sup>
23	x <sup>16</sup>	x <sup>26</sup>	x <sup>25</sup>	x <sup>25</sup>	x <sup>20</sup>	x <sup>16</sup>	x <sup>14</sup>
22	x <sup>20</sup>	x <sup>26</sup>	x <sup>26</sup>	x <sup>25</sup>	x <sup>17</sup>	x <sup>24</sup>	x <sup>21</sup>
24	x <sup>27</sup>	x <sup>25</sup>	x <sup>23</sup>	x <sup>26</sup>	x <sup>16</sup>	x <sup>14</sup>	x <sup>19</sup>
29	x <sup>25</sup>	x <sup>26</sup>	x <sup>25</sup>	x <sup>25</sup>	x <sup>22</sup>	x <sup>17</sup>	x <sup>18</sup>
25	x <sup>26</sup>	x <sup>26</sup>	x <sup>26</sup>	x <sup>25</sup>	x <sup>22</sup>	x <sup>19</sup>	x <sup>20</sup>
27	x <sup>25</sup>	x <sup>26</sup>	x <sup>26</sup>	x <sup>21</sup>	x <sup>21</sup>	x <sup>17</sup>	x <sup>9</sup>
10	x <sup>26</sup>	x <sup>24</sup>	x <sup>25</sup>	x <sup>16</sup>	x <sup>11</sup>	x <sup>6</sup>	x <sup>5</sup>
26	x <sup>25</sup>	x <sup>26</sup>	x <sup>21</sup>	x <sup>18</sup>	x <sup>10</sup>	x <sup>9</sup>	x <sup>4</sup>
25	x <sup>18</sup>	x <sup>11</sup>	x <sup>17</sup>	x <sup>11</sup>	x <sup>6</sup>	x <sup>4</sup>	x <sup>4</sup>
18	x <sup>15</sup>	x <sup>8</sup>	x <sup>6</sup>	x <sup>5</sup>	x <sup>5</sup>	x <sup>5</sup>	x <sup>5</sup>
5	x <sup>4</sup>	x <sup>5</sup>	x <sup>4</sup>	x <sup>16</sup>	x <sup>7</sup>	x <sup>10</sup>	x <sup>5</sup>
8	x <sup>4</sup>	x <sup>5</sup>	x <sup>6</sup>	x <sup>6</sup>	x <sup>4</sup>	x <sup>5</sup>	x <sup>5</sup>

Рис. 7.3. Исходные точки, задающие некоторую поверхность.

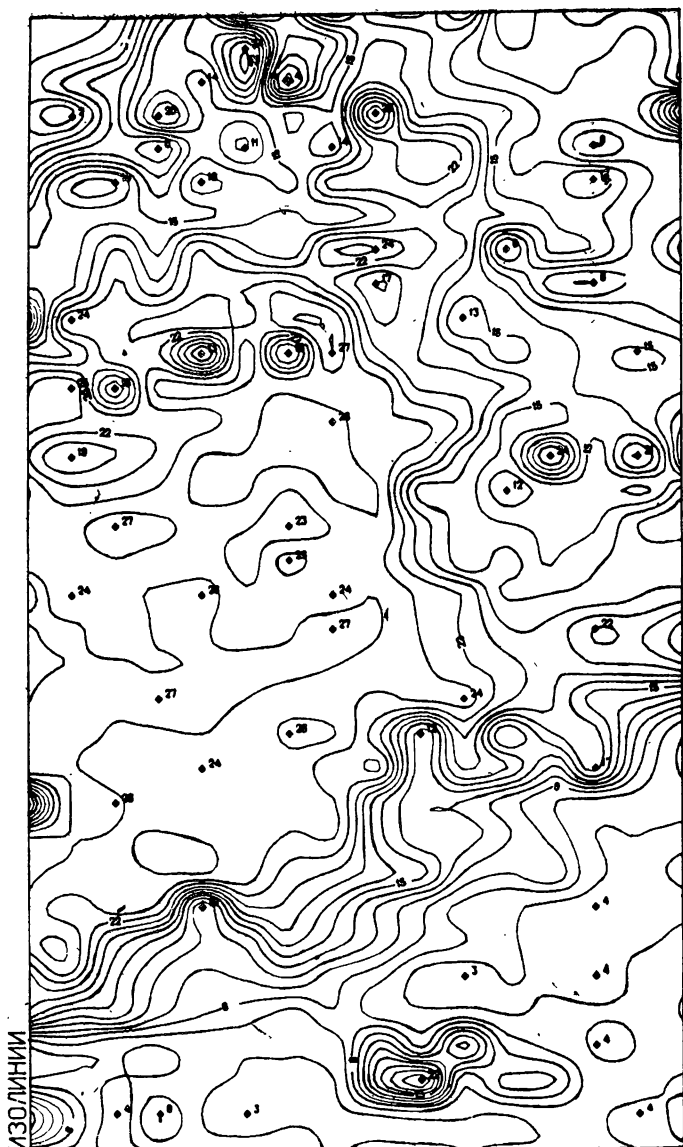


Рис. 7.4. Линии уровня поверхности, исходные точки которой показаны на рис. 7.3.

**7.2.4. Примеры.** 1. На рис. 7.3 и 7.4 показаны соответственно исходные точки и линии уровня некоторой поверхности. Изображение ее центральной проекции приведено ниже (см. рис. 8.11).

Первые два рисунка построены следующей программой:

```

DIMENSION Z(16, 34), X(16), Y(34), C(4), CAD(12)
DIMENSION PHI(76, 166), XP(76), YP(166)
DIMENSION IT(76, 166), IX(481), IY(481)
DATA NX, NY, LX, LY, MX, MY, NG, NC/16, 34, 76, 166,
' 5, 5, 481, 4/
DATA IS, NAD/4, 12/, C/0., 10., 20., 30./
999 FORMAT(3X, 8F6.2)
READ(t, 999) Z
DO 1 I=1, 16
1 X(I)=I-1
DO 2 I=1, 34
2 Y(I)=I-1
CALL ADDLEV(C, NC, IS, CAD, NAD)
CALL SFCFIT(NX, NY, X, Y, Z, MX, MY, LX, LY, XP,
' YP, PHI)
CALL QUADRA(IT, IX, IY, LX, LY, NG)
CALL PAGE(16., 26., 'ИСХОДНЫЕ ТОЧКИ
' ПОВЕРХНОСТИ', 26, 1)
CALL REGION(.2, .2, 15., 25., 0, 0, 0)
CALL LIMITS(0., 15., 0., 33.)
CALL KEYPO(Z, X, Y, NX, NY, 1, 1, -1., 35, 0, 0)
CALL ENDPG('7.3')
CALL PAGE(15., 25., 'ИЗОЛИНИИ', 8, 1)
CALL LIMITS(0., 15., 0., 33.)
CALL SIZLIN(.3, 2, 0)
CALL ISOLIN(LX, LY, XP, YP, PHI, IT, NG, IX, IY,
' NC, C, 0)
CALL LOCEXT(Z, X, Y, NX, NY, 0)
CALL RENUM
CALL ISOLIN(LX, LY, XP, YP, PHI, IT, NG, IX, IY,
' NAD, CAD, 0)
CALL ENDPG('7.4')
END

```

2. На рис. 7.5 изображены линии уровня поверхности, заданной на области со сложной границей. Граница задается массивами граничных точек. Описатель области подготавливается BITA.

```

DIMENSION Z(7, 8), X(7), Y(8), ZS(25), YS(29), IX(105),
' IY(105)
DIMENSION IT(25, 29), C(5), CAD(16)
DATA C/125., 150., 175., 200., 225./
DATA X, Y/1., 2., 3., 4., 5., 6., 7., 1., 2., 3., 4., 5., 6., 7., 8./

```

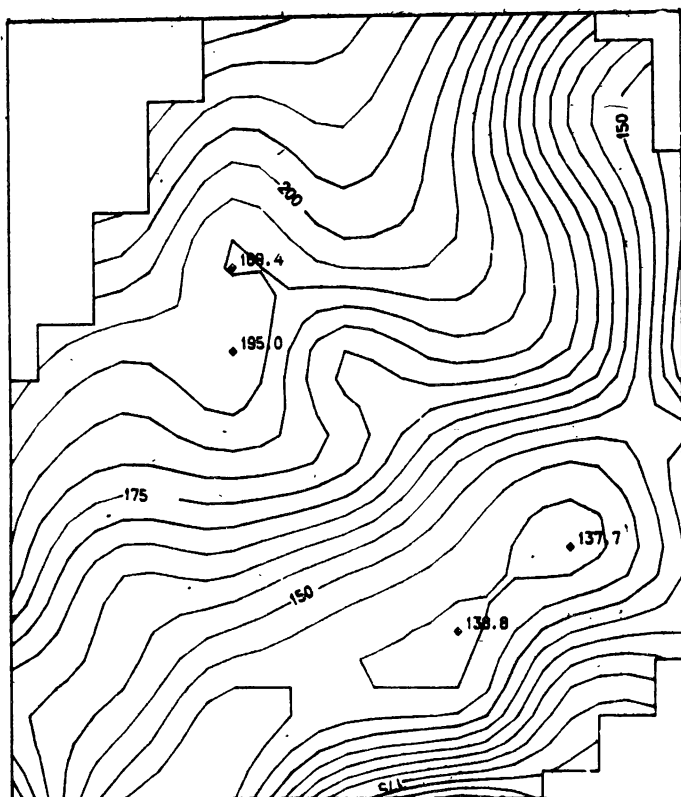


Рис. 7.5. Линии уровня поверхности, заданной на области со сложной границей.

```

99  FORMAT(7F5.2)
98  FORMAT(10I3)
    READ(1, 99) Z
    READ(1, 98) IX, IY
    CALL SFCFIT(7, 8, X, Y, Z, 4, 4, 25, 29, XS, YS, ZS)
    CALL BITA(IT, IX, IY, 25, 29, 105)
    CALL ADDLEV(C, 5, 4, CAD, 16)
    CALL PAGE(15., 17., 0, 0, 1)
    CALL LIMITS(1., 7., 1., 8.)
    CALL LOCEXT(ZS, XS, YS, 25, 29, 0)
    CALL ISOLIN(25, 29, XS, YS, ZS, IT, 105, IX, IY, 16., CAD, 0)
    CALL SIZLIN(.3, 3, 0)
    CALL ISOLIN(25, 29, XS, YS, ZS, IT, -105, IX, IY, 5, C, 0)
    CALL ENDPG('7.5')
    END

```

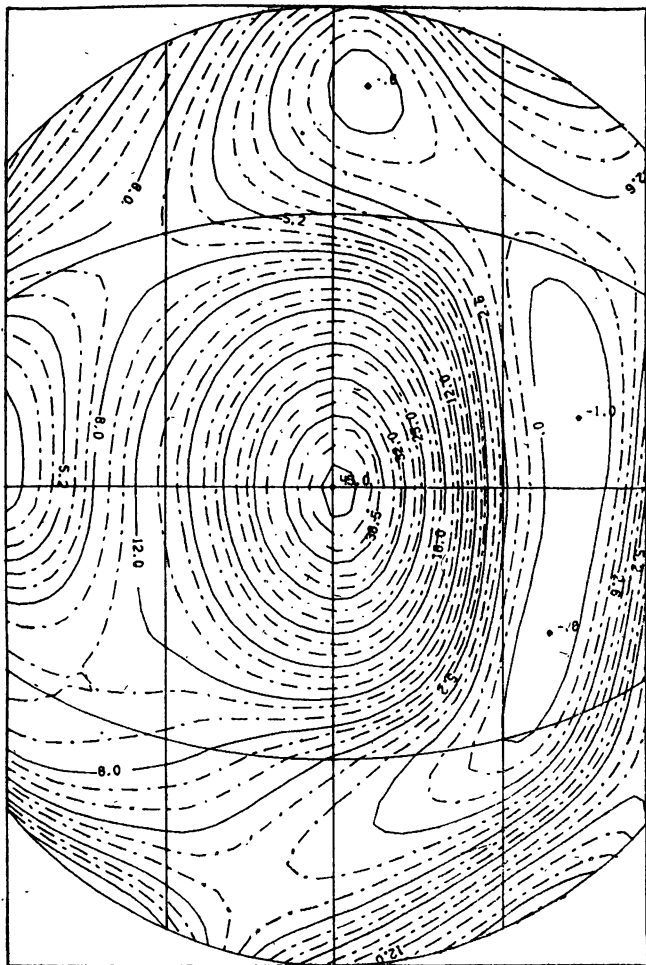
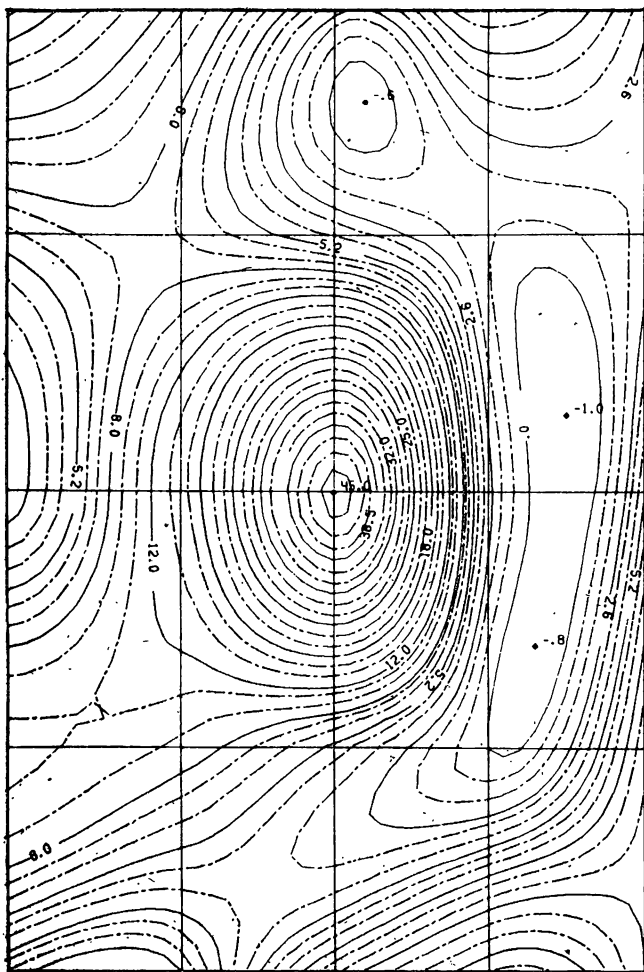


Рис. 7.6. Карта изолиний поверхности, заданной в сферической системе координат.





3. Если поверхность задана в некоторой произвольной системе координат, которая однозначно отображается на декартову, то можно построить карту изолиний такой поверхности. Для этого необходимо описать преобразование координат в подпрограммах-функциях XPOL и YPOL и вставить их в функциональную часть пакета. Границы области в этом случае устанавливаются программой EXTREM.

На рис. 7.6 и 7.7 показаны линии уровня одной и той же поверхности в предположении, что в первом случае исходные точки поверхности заданы в сферической системе координат, а во втором — в декартовой.

```

DIMENSION Z(5, 5), XZ(5), YZ(5), C(10), CAD(20)
DIMENSION PHI(41, 41), ITAGB(41, 41), X(41), Y(41)
DIMENSION IXBD(164), IYBD(164)
DATA Z/1., 0., 10., 20., 12., 11., 9., 5., 0., 10., 2.,
' 15., 45., 3., 5., 6., 7., 4., 1., 2., 21., 11., 3., 7., 3./
DATA XZ/0.736, 1.18, 1.57, 1.964, 2.36/
DATA YZ/-.736, -.393, 0., .393, .736/
DATA C/0., 2.6, 5.2, 8., 12., 18., 25., 32., 38., 44./
DATA NAD, NCN, L1, L2, N1, N2, MX, MY, N5, IS/18, 10,
' 5, 5, 41, 41, 10, 10, 161, 2/
CALL SFCFIT(L1, L2, XZ, YZ, Z, MX, MY, N1, N2, X,
' Y, PHI)
CALL QUADRA(ITAGB, IXBD, IYBD, N1, N2, N5)
CALL ADDLEV(C, NCN, IS, CAD, NAD)
CALL PAGE(17., 25., 0, 0, 1)
CALL EXTREM(X, Y, N1, N2, XMN, XMX, YMN, YMX, 1)
CALL LIMITS(XMN, XMX, YMN, YMX)
CALL CMGRID(X, Y, N1, N2, MX, MY, 0, 0, 0, 0, 1)
CALL LOCEXT(PHI, X, Y, N1, N2, 1)
CALL SIZLIN(.3, 4, 1)
CALL ISOLIN(N1, N2, X, Y, PHI, ITAGB, N5, IXBD,
' IYBD, NCN, C, 1)
CALL RENUM
CALL BROKEN(.5, .2, .1, .2)
CALL ISOLIN(N1, N2, X, Y, PHI, ITAGB, N5, IXBD,
' IYBD, NAD, CAD, 1)
CALL ENDPG('7.6')
CALL PAGE(17., 26., 0, 0, 1)
CALL EXTREM(X, Y, N1, N2, XMN, XMX, YMN, YMX, 0)
CALL LIMITS(XMN, XMX, YMN, YMX)
CALL LOCEXT(PHI, X, Y, N1, N2, 0)
CALL CMGRID(X, Y, N1, N2, MX, MY, 0, 0, 0, 0, 0)
CALL FULL

```

```

CALL SIZLIN(.3, .4, .1)
CALL ISOLIN(N1, N2, X, Y, PHI, ITAGB, N5, IXBD,
IYBD, NCN, C, 0)
CALL RENUM
CALL BROKEN (.5, .1, .1, .1)
CALL ISOLIN(N1, N2, X, Y, PHI, ITAGB, N5, IXBD,
IYBD, NAD, CAD, 0)
CALL ENDPG('7.7')
END
FUNCTION XPOL(X, Y)
XPOL = -COS(X)
RETURN
END
FUNCTION YPOL(X, Y)
YPOL = SIN(X) * SIN(Y)
RETURN
END

```

### 7.3. Построение изолиний и линий пересечения поверхностей

В этом параграфе представлены программы для построения изолиний и пространственных кривых, образованных пересечением поверхностей. Изолинии могут вычерчиваться линиями различного типа и снабжаться бергштрихами. Можно также вписывать в разрывы линий номера или значения уровней. Метод позволяет работать с поверхностями, заданными в произвольной системе координат, которая однозначно преобразуется в декартову.

**7.3.1. Поиск и проведение изолиний.** Задача нахождения и построения изолиний рассматривается в следующей постановке.

А. Изолинии являются решением уравнения  $F(X, Y) = \text{CONST}$  на области задания  $Z = F(X, Y)$ .

Б. Проекции линий пересечения двух функций  $Z_1 = F(X, Y)$ ,  $Z_2 = \Phi(X, Y)$  на области их задания являются решениями уравнения

$$\Phi(X, Y) = F(X, Y). \quad (7.1)$$

Если  $F(X, Y) - \Phi(X, Y) = P(X, Y)$ , то эта задача сводится к случаю А в виде

$$P(X, Y) = 0. \quad (7.2)$$

В. Пространственные линии пересечения двух поверхностей  $L$  также являются решением уравнения (7.2). Найдя множество значений  $(XI, YI)$ , удовлетворяющих (7.2),  $L$  можно выразить через  $F(X, Y)$  или  $\Phi(X, Y)$  следующим образом:

$$L(XI, YI) = F(XI, YI) = \Phi(XI, YI).$$

Таким образом, все три задачи сводятся к решению первой задачи.

Алгоритм, который использовался при решении поставленной задачи, достаточно традиционен (см. п. 7.2). Он позволяет строить изолинии для функций двух переменных  $Z = F(X, Y)$ , заданных в узлах прямоугольной неравномерной сетки. Если доопределить функцию на ребрах ячеек, используя линейную интерполяцию, тогда изолиниями будут ломаные, проходящие через точки пересечения отрезков функции, заданных на ребрах ячеек, с плоскостью  $Z = \text{CONST}$ .

В семействе изолиний различают незамкнутые изолинии, начинающиеся и заканчивающиеся на границе области задания, и замкнутые, лежащие целиком внутри области задания функции.

При построении незамкнутых линий производится обход по границе, чтобы найти начальные точки изолиний. Как только такая точка обнаружена, изолиния отслеживается до конца, т. е. до выхода ее на границу области задания функции.

Когда все незамкнутые линии данного уровня построены, производится последовательный просмотр всех горизонтальных ребер ячеек с целью выявления точек, принадлежащих замкнутой изолинии. Когда таковые найдены, изолинии отслеживаются до конца, т. е. до возврата в эти начальные точки. Эта процедура повторяется для всех заданных уровней. Чтобы исключить повторное проведение изолинии, в процессе отслеживания регистрируется факт ее прохождения через ребро (через каждое ребро нельзя провести более одной линии данного уровня).

Отметим некоторые особенности реализации алгоритма. Рассматриваемая поверхность предварительно «поднимается» в положительное полупространство на величину  $|ZMIN| + 1.0$ , где  $ZMIN$  — минимальное отрицательное значение функции в области задания либо нуль. Уровни также изменяются на эту величину. Поскольку все значения поверхности в результате становятся положительными, прохождение изолинии через ребро отмечается при освоении знака минус значениям функции в узлах, соединяемых этим ребром. Тем самым удастся избежать использования вспомогательного массива для хранения информации о прохождении изолинии через данное ребро. После построения всех линий уровня поверхность возвращается к исходному виду.

При отслеживании изолинии  $k$ -го уровня первое ребро, через которое проходит изолиния, определяется из следующих условий:

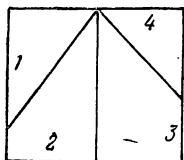
$$Z(I + 1, J) > ZIZ(K) \geq Z(I, J)$$

или

$$|Z(I + 1, J)| > ZIZ(K) \geq Z(I, J) \geq 0,$$

где  $ZIZ(K)$  — величина  $k$ -го уровня.

При выполнении любого из этих условий запоминаются соответствующие индексы  $I, J$ , и программа пытается отследить изолинию, перебирая последовательно в направлении против часовой стрелки ребра, через которые проходит изолиния. Координаты пересечения изолинии с ребром запоминаются в буферных массивах  $XI, YI$ . На рисунке



показан элементарный просмотр одной ячейки и возможные варианты прохождения изолинии. Цифры на ребрах указывают порядок просмотра.

Отслеживание изолинии производится до тех пор, пока не удастся выйти на ее начало (или на границу области, если линия начинается на границе). Когда буферные массивы координат заполнятся или изолиния будет просмотрена до конца, вся изолиния либо часть ее вычерчивается. Затем производится переход к обработке следующей изолинии или к продолжению текущей, если она выдана не полностью.

Программная реализация алгоритма позволяет строить карты изолиний как во всей области задания функции, так и в ее подобластях. Для работы программ требуются два буферных массива под координаты изолинии. Размеры этих массивов определяются программистом в зависимости от имеющейся свободной памяти.

Описываемые программные средства позволяют начертить изолинию с бергштрихами, провести ее как сплошную, штриховую или штрихпунктирную линию, вписать в разрыв линии номер или значение уровня. Для задания режимов рисования можно воспользоваться программами FULL, BROKEN, NUMLIN, SIZLIN (см. § 4.2, ил. 7.2.1, 7.2.2) и SETBE. При этом по умолчанию вычерчиваются сплошные линии без надписей и бергштрихов.

Заметим, что алгоритм построения линий уровня устроен таким образом, что изолинии, образованные пересечением секущей плоскости с выпуклыми участками поверхности, отслеживаются в направлении против часовой стрелки, а с вогнутыми — по часовой стрелке. Отсюда вытекает и способ задания бергштрихов как векторного произведения единичного базисного вектора  $k$  на отрезок-вектор  $\{(XI(L) - XI(L - 1)), (YI(L) - YI(L - 1))\}$ .

Данные программы построения линий уровня могут быть использованы не только в случае, когда исходная поверхность задана в декартовой системе координат. Допускается также обработка по-

верхностей, определенных в произвольной системе координат, которая однозначно отображается на декартову (например, в цилиндрической, сферической и др.). Для этого с помощью программы SETRA устанавливается режим преобразования, в результате каждая изолиния перед выводом будет преобразована указанным образом (с помощью служебной программы TRANSF).

Как обычно, при рисовании в области необходимо задавать пределы изменения обоих аргументов в этой области (LIMITS). Определить пределы в случае недекартовой системы координат удобно с помощью программы BOUND.

### 7.3.2. Описание программ построения карт изолиний.

Программа IZOLIN(Z, M, N, X, Y, ZIZ, L, XI, YI, NL) позволяет начертить изолинии функции двух переменных во всей области ее задания. Параметры программы следующие:

Z — массив значений функции (размером (M, N));

M, N — число точек сетки по осям X и Y;

X, Y — массивы координат сетки по осям X и Y;

ZIZ — массив значений уровней;

L — размер массива уровней;

XI, YI — буферные массивы X- и Y-координат изолинии;

NL — размер буферных массивов.

Программа IZFLIN(Z, M, N, X, Y, ZIZ, L, NX, NY, KX, KY, NF, XI, YI, NL) позволяет начертить изолинии функции двух переменных в указанных подобластях области ее задания. Параметры NX, NY, KX, KY (массивы индексов сетки по соответствующим осям) определяют нижние левые и верхние правые углы прямоугольных подобластей, NF — размер массивов (число подобластей). Остальные параметры такие же, как и в программе IZOLIN.

Программа SETBE(LSTEP, SIZEB) устанавливает режим вычерчивания изолинии с бергштрихами. Ее параметры:

LSTEP — шаг установки бергштриха (метится каждая LSTEP-я точка, начиная с первой);

|SIZEB| — размер бергштриха: SIZEB > 0 — размер бергштриха в выбранных единицах измерения, SIZEB < 0 — в рабочих единицах (математическое значение).

Программа SETRA(L) устанавливает признак системы координат: L = 0 — декартова система координат, L > 0 — недекартова система координат.

Программа BOUND(Z, M, N, X, Y, NX, NY, KX, KY, NF, XMI, XMA, YMI, YMA) позволяет определить пределы изменения аргументов функции на области ее задания в произвольной системе координат. Параметры программы следующие:

Z — массив значений функции (размером (M, N));

M, N — число точек сетки по осям X и Y;

X, Y — массивы координат сетки по осям X и Y;

NX, NY, KX, KY — массивы индексов сетки по осям X, Y; их элементы определяют нижние левые и верхние правые углы прямоугольных подобластей (длины NF);

NF — число подобластей;

XMI, XMA — минимальное и максимальное значения по оси X;

YMI, YMA — минимальное и максимальное значения по оси Y.

Для выполнения преобразований произвольной системы координат в декартову необходимы функции ATDX и ATDY.

Функция ATDX(Z, X, Y) вычисляет координату X в декартовой системе координат по координатам точки в исходной системе координат.

Функция ATDY(Z, X, Y) вычисляет координату Y в декартовой системе координат по координатам точки в исходной системе координат.

Базовой системой координат выбрана декартова. Для того чтобы можно было работать с другими системами координат, необходимо заменить функции ATDX и ATDY на другие, выполняющие желаемые преобразования.

```
FUNCTION ATDX(Z, X, Y)
  ATDX=Φ(Z, X, Y)
  RETURN
END
```

```
FUNCTION ATDY(Z, X, Y)
  ATDY=Φ(Z, X, Y)
  RETURN
END
```

Так, для цилиндрической системы координат вместо  $\Phi(Z, X, Y)$  следует воспользоваться программами ATDX1 и ATDY1, для сферической — это функции ATDX2 и ATDY2. Они имеются в библиотеке Графора. В случае использования других систем координат пользователю предоставляется возможность написать соответствующие программы-функции самостоятельно. Например, функции ATDX1 и ATDX2 имеют вид:

```
FUNCTION ATDX1(Z, X, Y)
  ATDX1=X*COS(Y)
  RETURN
END
```

```
FUNCTION ATDX2(Z, X, Y)
  ATDX2=Z*SIN(Y)*COS(X)
  RETURN
END
```

Служебные программы. Программы IZOLIN и IZFLIN используют при работе подпрограммы IZLIN, RAISE, LOWER, RECUR, LETIZO. Непосредственный вывод вычисленной изолинии осуществляет программа LETIZO. В качестве параметров ей передаются массивы координат изолинии и их размеры.

Программа IZLIN(Z, M, N, X, Y, ZMI, ZIZ, NUM, NXT, NYT, KXT, KYT, XI, YI, NL) позволяет начертить изолинии функции двух переменных в выделенной подобласти задания функции, если в этой подобласти функция положительна. Параметры программы:

ZMI — величина «поднятия» поверхности;  
ZIZ — значение текущего уровня с номером NUM;  
NXT, NYT — индексы сетки по осям X и Y, определяющие нижний угол прямоугольной подобласти;

KXT, KYT — индексы сетки по осям X и Y, определяющие верхний правый угол прямоугольной подобласти.

Остальные параметры те же, что и в программе IZOLIN.

Программы RAISE, LOWER, RECUR являются программами обработки поверхности.

Программа RAISE(Z, M, N, ZM1) определяет величину ZMI, на которую надо «поднять» поверхность, и «поднимает» ее в положительное полупространство.

Программа LOWER(Z, M, N, ZMI) позволяет «опустить» ( $ZMI > 0$ ) или «поднять» ( $ZMI < 0$ ) поверхность на заданную величину ZMI.

Программа RECUR(Z, M, N, NX, NY, KX, KY) заменяет все отрицательные значения функции  $Z(I, J)$  на  $|Z(I, J)|$  в заданной подобласти. Параметры этой программы задают массив значений функции, размерность массива, индексы сетки по осям, определяющие нижний левый и правый верхний углы прямоугольной подобласти.

Функция XILY(Z1, Z2, ARG1, ARG2, ZIZOL, I, J, IUS, JUS) производит линейную интерполяцию на ребре с крайними точками (ARG1, Z1), (ARG2, Z2) и регистрирует факт прохождения изолинии через ребро. Здесь: ZIZOL — величина уровня (с учетом «поднятия» поверхности), I, J — индексы внешнего (по отношению к изолинии) узла, IUS, JUS — приращения индексов сетки (по осям X и Y соответственно), определяющих внутренний по отношению к изолинии узел. (Внутренним по отношению к изолинии считается узел, в котором значение функции больше величины рассматриваемого уровня. Аналогично, узел, значение функции в котором меньше величины уровня, считается внешним.)

Для вывода изолинии по вычисленным значениям координат используется программа LETIZO.

Программа LETIZO(XI, YI, IL, ZIZ, NIZ, KIND) вычерчивает линию по заданным массивам координат XI, YI длины IL и вписывает в разрыв значение ZIZ или номер NIZ уровня. Параметр KIND определяет тип линии (0 — незамкнутая, 1 — замкнутая).

```
SUBROUTINE LETIZO(XI, YI, ILM, ZIZ, NIZ, KIND)
  DIMENSION XI(ILM), YI(ILM)
  COMMON/GFBET/KT, ISTEP, SIZE
  COMMON/GFIZOL/NC, C
  NC=NIZ
  C=ZIZ
```



```

IF(KIND.EQ.0) GO TO 1
XI(ILM)=XI(1)
YI(ILM)=YI(1)
1 CONTINUE
IF(KT.NE.0) CALL TRANSF(ZIZ, XI, YI, ILM)
IF(ISTEP.GT.0) CALL MARKBE(XI, YI, ILM)
CALL LINNUM(XI, YI, ILM)
RETURN
END

```

При работе этой программы используются программы LINNUM (см. п. 4.2), MARKBE и TRANSF.

Программа MARKBE(XI, YI, IL) устанавливает бергштрихи на изолинии. Ее параметры: XI, YI — массивы координат изолинии, IL — размер массивов.

Программа TRANSF(ZIZ, XI, YI, ILM) переводит массивы изолиний из произвольной системы координат в декартову в соответствии с заданными функциями преобразования. Ее параметры:

ZIZ — величина строящегося уровня;  
 XI, YI — массивы X- и Y-координат изолинии;  
 ILM — размер массивов изолиний.

**7.3.3. Вспомогательные программы.** Приведем описание нескольких программ, которые могут быть полезны при оформлении рисунка.

Программа EXUDE(Z, M, N, X, Y, NX, NY, KX, KY, NF, MRKA, MRKI, KD, H, TH) позволяет найти, пометить и надписать локальные экстремумы функции. Экстремальным считается тот узел, значение функции в котором больше (меньше) ее значений в смежных узлах. Параметры программы следующие:

Z — массив значений функции (размером (M, N));  
 M, N — число точек сетки по осям X и Y;  
 X, Y — массивы значений сетки по осям X и Y;  
 NX, NY — массивы индексов сетки по осям X и Y, определяющих нижние левые углы прямоугольников, в которых необходимо искать экстремумы функции;  
 KX, KY — массивы индексов сетки по осям X и Y, определяющих верхние правые углы прямоугольников, в которых необходимо искать экстремумы функции;  
 NF — количество прямоугольников;  
 MRKA — номер маркера для максимума;  
 MRKI — номер маркера для минимума;  
 KD — количество дробных знаков;  
 H — высота цифры в заданных единицах измерения;  
 TH — угол наклона текста (в градусах).

Программа TOKEN(XS, YS, MRK, FZ, KD, H, TH) помещает точку заданным маркером и надписывает заданное значение. Параметры программы:

XS, YS — декартовы математические координаты точки;

MRK — номер маркера;

FZ — надписываемое число;

KD — количество дробных знаков;

H — высота цифры в заданных единицах измерения;

TH — угол наклона текста (в градусах).

Программа EXMIMA(Z, M, N, ZMI, ZMA) позволяет разделить максимальное и минимальное значения в двумерном массиве чисел. Ее параметры: Z — двумерный массив размером (M, N), ZMI и ZMA — минимальное и максимальное значения.

**7.3.4. Построение пространственных кривых.** Пусть имеются две поверхности  $A$  и  $B$ , заданные на одной и той же прямоугольной неравномерной сетке. Если  $P = A - B$ , то  $P = 0$  в точках, где  $A$  и  $B$  совпадают. Построив изолинии нулевого уровня от  $P$ , получим координаты  $(XI, YI)$  проекции пространственных линий пересечения  $A$  и  $B$  на области задания. По этим координатам вычисляются значения  $Z$  пространственной линии  $L$ :

$$L(XI, YI) = A(XI, YI) = B(XI, YI).$$

Для реализации этой возможности имеется вариант программы-функции XILY — XILY1 и программы LETIZO — LETSPL. Пользователь должен обеспечить необходимую замену, т. е. вложить в пакет:

```
FUNCTION XILY(Z1, Z2, ARG1, ARG2, ZIZOL, I, J,
```

```
'IUS, JUS)
```

```
XILY1(Z1, Z2, ARG1, ARG2, ZIZOL, I, J,
```

```
'IUS, JUS)
```

```
RETURN
```

```
END
```

```
SUBROUTINE LETIZO(XI, YI, IL, ZIZ, NIZ, KIND)
```

```
DIMENSION XI(IL), YI(IL)
```

```
CALL LETSPL(XI, YI, IL, ZIZ, NIZ, KIND)
```

```
RETURN
```

```
END
```

Кроме того, необходимо в главной программе задать два общих блока:

```
COMMON/GFNAM1/M, B
```

```
COMMON/GFNAM2/KEY, PL
```

где  $B$  — массив размером  $(M, N)$ , описывающий одну из поверхностей,  $M$  — количество столбцов массива,  $KEY$  — текущий индекс заполнения массивов изолиний пространственной кривой (перед

обращением к программе IZOLIN должен быть установлен равным 1), PL — одномерный массив под пространственную линию (того же размера, что и буферный массив под изолинию). Пример приведен в следующем пункте.

Следует заметить, что при рисовании пространственная линия не замыкается.

**7.3.5. Примеры.** В этом разделе приведены примеры, показывающие различные возможности программ IZOLIN и IZFLIN.

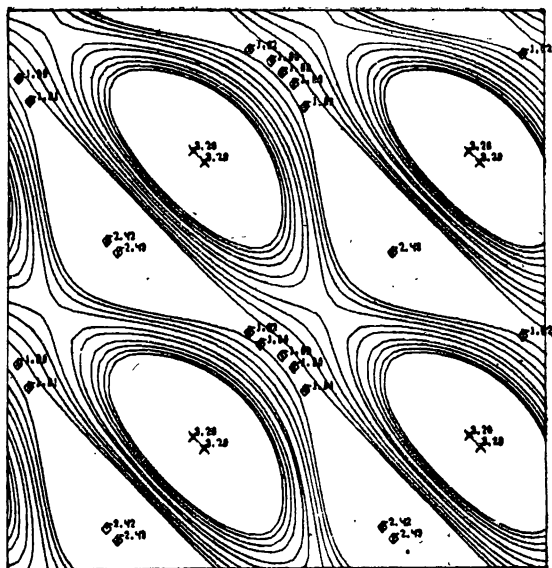
1. Рис. 7.8 и 7.9 иллюстрируют работу программы IZFLIN. На первом рисунке изолинии построены во всей области задания функции, на втором — в двух прямоугольных подобластях. Пределы изменения функции определены на всей области ее задания.

```

DIMENSION Z(50, 50), RO(50), FI(50), XI(200),
* YI(200), ZIZ(10)
DIMENSION NX(10), KX(10), NY(10), KY(10), X(50), Y(50)
EQUIVALENCE (X(1), RO(1)), (Y(1), FI(1))
DATA ZIZ/0.1, 0.4, 0.6, 0.8, 0.95, -.1, -.4, -.6, -.8, -.95/
DATA NLINE/200/
DATA NF/2/, NX, NY, KX, KY/1, 25, 8*0, 1, 25, 8*0, 25,
* 50, 8*0, 25, 50, 8*0/, M/50/, N/50/, L/10/
H=12.56/50
DO 1 I=1, 50
  X(I)=H*(I-1)
  Y(I)=X(I)
1 CONTINUE
DO 5 I=1, 50
DO 5 J=1, 50
  Z(I, J)=SIN(X(I)+Y(J))+COS(X(I)+Y(J))-
  * (SIN(X(I))+COS(Y(J)))
5 CONTINUE
CALL PAGE(17., 17., 'ПОВ., ЗАДАН. В ДЕК. КООРД',
* 24, 0)
CALL REGION(1., 1., 15., 15., 0, 0, 1)
CALL BOUND(Z, M, N, RO, FI, NX, NY, KX, KY, NF,
* XMI, XMA, YMI, YMA)
CALL LIMITS(XMI, XMA, YMI, YMA)
CALL IZFLIN(Z, M, N, RO, FI, ZIZ, L, NX, NY, KX,
* KY, NF, XI, YI, NLINE)
CALL EXUDE(Z, M, N, RO, FI, NX, NY, KX, KY, NF, 2,
* 3, 2, 0.23, 0.)
CALL ENDPG('7.9')

```

2. На рис. 7.10 приведен пример построения изолиний с бергштрихами.



7.8. Карта изолиний, построенная на всей области задания функции.

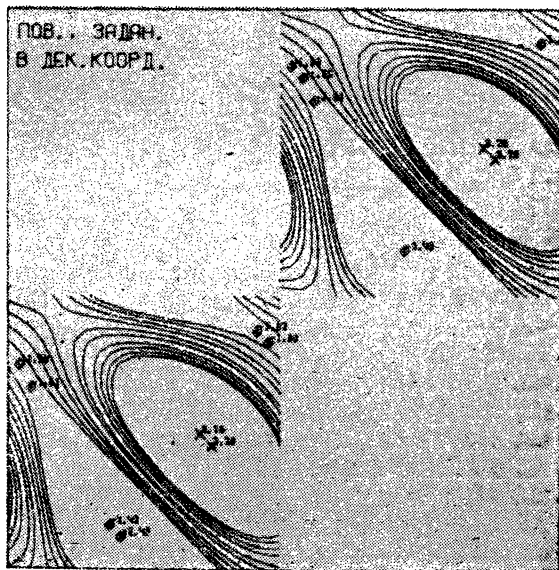


Рис. 7.9. Карта изолиний, построенная в двух прямоугольных под-областях.

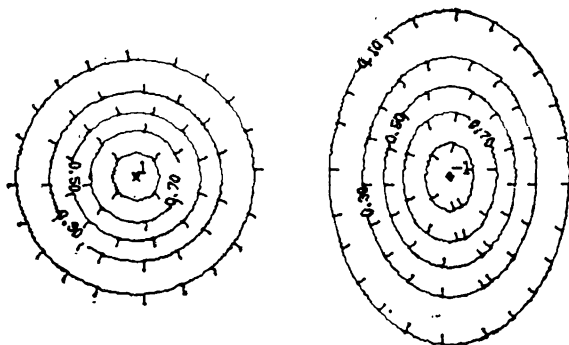


Рис. 7.10. Пример построения изолиний с бергштрихами.

```

        DIMENSION Z(50, 50), X(50), Y(50), XI(200),
*      YI(200), ZIZ(10)
        DATA M, N, L, NLINE/50, 50, 10, 200/
        X(1)=0.
        Y(1)=0.
        H=0.20
        DO 10 I=1, 50
            X(I)=H-FLOAT(I)
            Y(I)=X(I)
10      CONTINUE
        DO 20 I=1, 50
            DO 30 J=1, 50
                Z(I, J) = EXP(-(X(I)-3.)**2+(Y(J)-5.)**2))-
*      EXP(-((X(I)-7.)**2+0.5*(Y(J)-5.)**2))
30      CONTINUE
20      CONTINUE
        DO 40 I=1, 10
            ZIZ(I)=-1.1+0.200*FLOAT(I)
40      CONTINUE
        CALL PAGE(17., 17., 'BERG', 4, 0)
        CALL REGION(1., 1., 15., 15., 'SIZEB=0.15, LSTEP=3', 19, 0)
        CALL MINMAX(X, M, XMI, XMA)
        CALL MINMAX(Y, N, YMI, YMA)
        CALL LIMITS(XMI, XMA, YMI, YMA)
        CALL SIZLIN(0.23, 4, 2)
        CALL SETBE(3, 0.15)
        CALL IZOLIN(Z, M, N, X, Y, ZIZ, L, XI, YI, NLINE)
        CALL EXUDE(Z, M, N, X, Y, 1, 1, M, N, 1, -2, -3, 0,
*      0.3, 0.)
        CALL ENDPG('7.10')
    
```

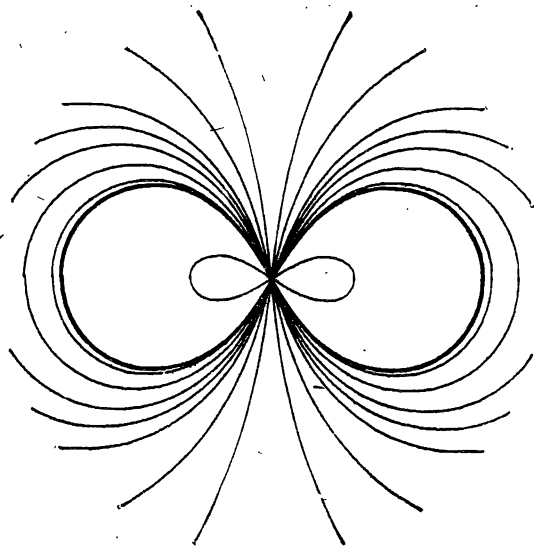


Рис. 7.11. Линии уровня в цилиндрических координатах.

3. Линии уровня в цилиндрических координатах показаны на рис. 7.11.

```

DIMENSION Z(50, 50), RO(50), FI(50), XI(200),
* YI(200), ZIZ(10)
DIMENSION NX(10), KX(10), NY(10), KY(10)
DATA ZIZ/0.1, 0.4, 0.8, 1., 1.2, 1.5, 1.8, 1.9, 1.93, 5./
DATA NF/1/, NX, NY, KX, KY/1, 9*0, 1, 9*0, 50, 9*0, 50, 9*0/
* M, N, L, NLINE/50, 50, 10, 200/
HRO=4.2/49.
HFI=3.14/49.
DO 1 I=1, 50
  RO(I)=HRO*(I-1)-2.1
  HFI=HFI*(I-1)
1 CONTINUE
DO 5 I=1, 50
  DO 5 J=1, 50
    Z(I, J)=7.*(COS(FI(J)))*2/(RO(I)**2+1.)
5 CONTINUE
CALL PAGE(17., 17., 'ПОВ., ЗАДАН. В ЦИЛ. КООРД.', 24, 0)
CALL REGION(1., 1., 15., 15., 0, 0, 1)
CALL SETRA(1)
CALL BOUND(Z, M, N, RO, FI, NX, NY, KX, KY, NF, XMI,
* XMA, YMI, YMA)

```

```

CALL LIMITS(XMI, XMA, YMI, YMA)
CALL IZOLIN(Z, M, N, RO, FI, ZIZ, L, XI, YI, NLINE)
CALL ENDPG('7.11')
END
FUNCTION ATDX(Z, X, Y)
ATDX=ATDX1(Z, X, Y)
RETURN
END
FUNCTION ATDY(Z, X, Y)
ATDY=ATDY1(Z, X, Y)
RETURN
END

```

4. На рис. 7.12 приведен пример построения пространственной кривой, образованной пересечением поверхностей (полусфера пересекается наклонной плоскостью).

```

SUBROUTINE PRLINE
COMMON/GFNAM1/M, B/GFNAM2/KEY, PL
DIMENSION A(50, 50), B(50, 50), X(50), Y(50)
DIMENSION XI(200), YI(200), PL(200)
DATA N, H/50, 0.084/
KEY=1
M=50
X(1)=-2.1

```

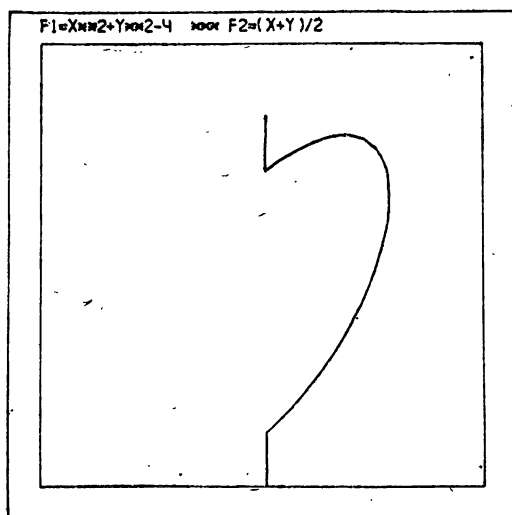


Рис. 7.12. Изображение пространственной кривой, образованной пересечением поверхностей,

```

Y(1) = -2.1
DO 10 I = 2, N
X(I) = X(I-1) + H
10 Y(I) = X(I)
DO 20 I = 1, N
DO 20 J = 1, N
DDD = X(I) ** 2 + Y(J) ** 2 - 4.
A(I, J) = 0.
GO TO 5
IF(DDD.LT.0) A(I, J) = SQRT(-DDD)
20 B(I, J) = (X(I) + Y(J)) / 2.
CALL PAGE(16., 16., 0, 0, 0)
CALL REGION(1., 1., 14., 14.,
* 'F1=X**2+Y**2-4***F2=(X+Y)/2', 30, 1)
DO 30 I = 1, N
DO 30 J = 1, N
30 A(I, J) = A(I, J) - B(I, J)
CALL INIT
CALL ISOMET
CALL TDLIM(X, Y, A, N, N, 1, N, 1, N, S)
ZIZ = 0.
CALL IZOLIN(A, N, N, X, Y, ZIZ, 1, XI, YI, 200)
CALL ENDPG('7.12')
RETURN
END
SUBROUTINE LETIZO(XI, YI, ILM, ZIZ, NIZ, KIND)
DIMENSION XI(ILM), YI(ILM)
CALL LETSPL(XI, YI, ILM, ZIZ, NIZ, KIND)
RETURN
END
FUNCTION XILII(Z1, Z2, ARG1, ARG2, ZIZOL, I, J,
* IUS, JUS)
XILII = XILII1(Z1, Z2, ARG1, ARG2, ZIZOL, I, J,
* IUS, JUS)
RETURN
END

```

Примечание. Изображение пространственной кривой в этом примере производится с использованием средств, описанных в п. 8.1.



## 8. ПОСТРОЕНИЕ ПЛОСКИХ ИЗОБРАЖЕНИЙ ТРЕХМЕРНЫХ ОБЪЕКТОВ

В этой главе описаны программные средства, позволяющие строить плоские проекции пространственных объектов. Для достижения большей реалистичности и наглядности изображения те части образа объекта, которые невидимы наблюдателю, стираются.

В Графоре реализовано несколько различных методов *удаления невидимых линий*, однако большинство из них предназначено для изображения графических объектов, представляющих собой однозначные функции двух переменных, хотя они могут использоваться и в некоторых более общих случаях.

С помощью программ, изложенных в §§ 8.1 и 8.2 (см. также [1, г, 1, ж]), можно строить проекции поверхностей, описываемых однозначными непрерывными функциями двух переменных, заданными на прямоугольной области. В основе первого метода — будем называть его *методом слоев* — лежит способ параллельных сечений. Секущие плоскости покрывают поверхность семейством кривых, которые затем проецируются на картинную плоскость. В зависимости от расположения секущих плоскостей в пространстве поверхность может быть покрыта сечениями, параллельными плоскостям  $XZ$  или  $YZ$ , или же она покрывается одновременно двумя этими семействами сечений, образующими криволинейную сетку (совокупность криволинейных четырехугольников). С помощью этого метода можно построить произвольные аксонометрические проекции, а также центральные проекции на плоскость, перпендикулярную *лучу зрения* (отрезку, соединяющему центр проекции с началом координат).

В другом алгоритме, построенном на основе *метода ячеек*, элементарными единицами являются участки поверхности, соответствующие лишь одной ячейке криволинейной сетки. Поэтому изображения, полученные с использованием этого алгоритма, обладают несколько более высоким качеством, чем проекции, построенные с помощью метода слоев. В алгоритме, реализующем метод ячеек, кроме того, применяется более простой способ проецирования по сравнению с традиционными подходами (см. [1, г, 1, к, 1, л]), основанными на аппарате однородных координат. Од-

нако с помощью второго метода можно получить лишь центральные проекции поверхностей.

В § 8.3 описаны программные средства, позволяющие получать практически произвольные (лишь с небольшими ограничениями) центральные проекции поверхностей (см. также [1, к]). Они позволяют строить изображения функций, которые заданы в произвольно расположенных точках. Более того, область определения может быть неодносвязной. Поверхности аппроксимируются с помощью треугольной сетки, формируемой по заданной границе области и узлам внутри нее с помощью *триангуляции*.

Алгоритмы удаления невидимых линий, применяемые в трех упомянутых случаях, во многом схожи. В основе их лежит упорядочение соответствующих элементарных единиц поверхности, будь то сечения параллельными плоскостями или треугольные, или четырехугольные элементы — «кусочки». При этом выбирается такой порядок построения элементарных единиц поверхности (назовем их элементами), при котором каждый элемент поверхности может быть закрыт только ранее построенными участками поверхности.

Такой упорядоченный перебор элементов поверхности позволяет ввести *экран*, определяемый как некоторый многоугольник на плоскости рисунка. При проецировании очередного элемента изображаются только те его участки, которые выходят за границы экрана, и экран соответствующим образом модифицируется. Практически экран отображает на плоскости рисунка проекцию участка поверхности, который соответствует построенным к данному моменту элементам поверхности. Для экономного задания экрана на картинной плоскости вводится *физическая сетка*, и экран определяется с точностью до этой сетки.

В § 8.4 описан еще один способ построения проекций трехмерных объектов (см. также [1, л]). Для создания эффекта глубины в нем используется *метод ореола*, когда дальние, невидимые отрезки стираются не полностью, но в них вносятся разрывы, которые вызваны ореолами вокруг более близких к наблюдателю отрезков.

Основное отличие метода ореола от других способов удаления невидимых линий состоит в том, что он позволяет работать с неструктурированной графической информацией, т. е. предназначен для изображения объектов, представленных в виде неупорядоченной совокупности отрезков прямых.

С точки зрения быстродействия первый и второй методы обладают преимуществом по сравнению с третьим и четвертым, поскольку время их работы линейно зависит от числа изображаемых отрезков, тогда как у третьего и четвертого наблюдается зависимость более высокого порядка.

## 8.1. Изображение функций двух переменных. Метод параллельных сечений

С помощью программ, описанных в этом параграфе, можно строить проекции поверхностей, заданных однозначной непрерывной функцией двух переменных, с удалением невидимых линий. Функция должна быть определена на прямоугольной сетке (не обязательно равномерной). Для аффинных преобразований пространственных объектов и для их проецирования на картинную плоскость используется аппарат однородных координат. С помощью

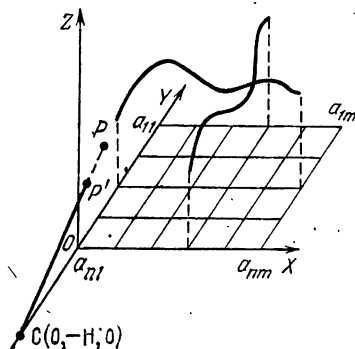


Рис. 8.1. Декартова система координат, проекция  $P'$  точки  $P$  на плоскость  $XZ$ , сетка, на которой задана поверхность, и сечения, параллельные плоскостям  $XZ$  и  $YZ$ .

этого метода можно получать как аксонометрические, так и центральные проекции на плоскость, перпендикулярную «линии визирования», проведенной из центра проекции в начало координат. Для графического представления функции используется способ параллельных сечений. Упорядоченные сечения лежат в основе алгоритма удаления невидимых линий. Имеются также средства, позволяющие строить изображение пространственных кривых.

### 8.1.1. Преобразование координат в трехмерном пространстве.

В основе программ аффинных преобразований пространственных объектов, а также их проеци-

рования на картинную плоскость лежит аппарат однородных координат (см., например, [15, 16]). При этом все необходимые для построения проекции и установления нужного ракурса преобразования координат описываются матрицами размером  $4 \times 4$  и представляются в виде суперпозиции некоторых основных преобразований: переноса точки в пространстве на фиксированный вектор, поворота вокруг указанной оси на заданный угол, масштабирования вдоль какой-либо оси, сдвига, перспективы и проецирования на одну из главных координатных плоскостей.

Основные преобразования координат. Рассмотрим некоторую декартову систему координат (рис. 8.1). Любая точка пространства представляется в ней вектор-матрицей вида  $(x \ y \ z)$ . Мы будем пользоваться однородными координатами точки в пространстве  $(x \ y \ z \ 1)$ .

В качестве *картинной* плоскости выберем плоскость  $XZ$ , описываемую уравнением  $Y = 0$ . *Проекция* точки объекта на эту пло-

скость получается в результате умножения  $(x \ y \ z \ 1)$   $A$ , где

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

задает проецирование на плоскость  $XZ$ .

Поворот вокруг заданной оси ( $X$ ,  $Y$  и  $Z$  соответственно) на указанный угол  $\alpha$  описывается следующими матрицами:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & b & -a & 0 \\ 0 & -a & b & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_y(\alpha) = \begin{bmatrix} b & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & b & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_z(\alpha) = \begin{bmatrix} b & a & 0 & 0 \\ -a & b & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

где  $a = \sin \alpha$ ,  $b = \cos \alpha$ . Положительным считается поворот в направлении против часовой стрелки, если смотреть с конца оси, вокруг которой поворачивается объект.

Матрицы преобразований переноса точки на фиксированный вектор и масштабирования вдоль осей имеют следующий вид:

$$TR = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}, \quad S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & s \end{bmatrix}.$$

Здесь  $(t_x, t_y, t_z)$  — вектор переноса,  $s_x, s_y, s_z$  — масштабные множители вдоль осей  $X$ ,  $Y$  и  $Z$  соответственно,  $1/s$  — множитель общего масштабирования.

Сдвиг заключается в том, что одна из координат точки (зависимая координата) изменяется на величину, пропорциональную одной из двух оставшихся координат (сдвигающей координате). Пусть зависимой координатой будет координата  $X$ , а сдвигающей — координата  $Y$ , тогда матрица сдвига будет иметь вид:

$$SH = \begin{bmatrix} 1 & 0 & 0 & 0 \\ F & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

где  $F$  — коэффициент сдвига. Проекцию точек объекта на плоскость  $XZ$  из центра проекции  $C$  можно получить с помощью центрального проецирования. Его матрица:

$$P(H) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/H \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Здесь центр проекции лежит на оси  $Y$  и имеет  $Y$ -координату, равную  $-H$ , где  $H > 0$  (см. рис. 8.1).

С помощью основных преобразований координат можно получить практически произвольные плоские геометрические проекции.

Рассмотрим сначала случай параллельного проецирования. В зависимости от того, какой угол образует направление проецирования с картинной плоскостью, параллельные проекции делятся на *прямоугольные* (например, аксонометрические проекции) и *косугольные*. В случае прямоугольных проекций направление проецирования перпендикулярно картинной плоскости. В случае косугольных проекций направление проецирования образует с картинной плоскостью угол, отличный от прямого. Более подробные сведения об этих типах проекций можно найти, например, в [16].

Более общие аксонометрические проекции можно получить с помощью двух последовательных поворотов объекта (сначала вокруг оси  $Z$  на некоторый угол  $Az$ , а потом вокруг оси  $X$  на угол  $Ax$ ) и затем ортогонального проецирования на плоскость  $XZ$ . Для двух наиболее распространенных типов аксонометрических проекций — изометрии и диметрии — углы поворота имеют следующие значения:  $Az = -45^\circ$ ,  $Ax = 35^\circ$  и  $Az = -20^\circ$ ,  $Ax = 20^\circ$ .

На рис. 8.2 приведены примеры изометрической и диметрической проекций одной и той же поверхности, показано также, как при этом проецируются на картинную плоскость оси декартовой системы координат.

Для построения косугольных проекций удобно воспользоваться преобразованием сдвига. Одну из косугольных проекций можно построить, например, следующей последовательностью преобразований:

1) сдвиг, в котором зависимой осью является ось  $X$ , сдвигающей осью — ось  $Y$ ; коэффициент сдвига  $F=1$  в случае, если задана «положительная» проекция (рис. 8.3, б), и  $F=-1$ , если — «отрицательная» (рис. 8.3, а);

2) сдвиг, в котором зависимой является ось  $Z$ , сдвигающей — ось  $Y$  и коэффициент сдвига  $F=1$ ;

3) проецирование на плоскость  $XZ$ .

В случае, если ни одна из упомянутых стандартных параллельных проекций (изометрия, диметрия и косугольная проекция) по каким-либо причинам не устраивает, можно построить требуемую проекцию с помощью переноса, поворота, масштабирования и сдвига.

Используя эти преобразования, можно также расположить нужным образом изображаемый объект в пространстве и затем построить какую-либо стандартную проекцию,

С помощью основных преобразований координат легко также формируется преобразование, которое позволит получать центральную проекцию объекта из произвольного центра проекции на плоскость, проходящую через начало координат перпендикулярно

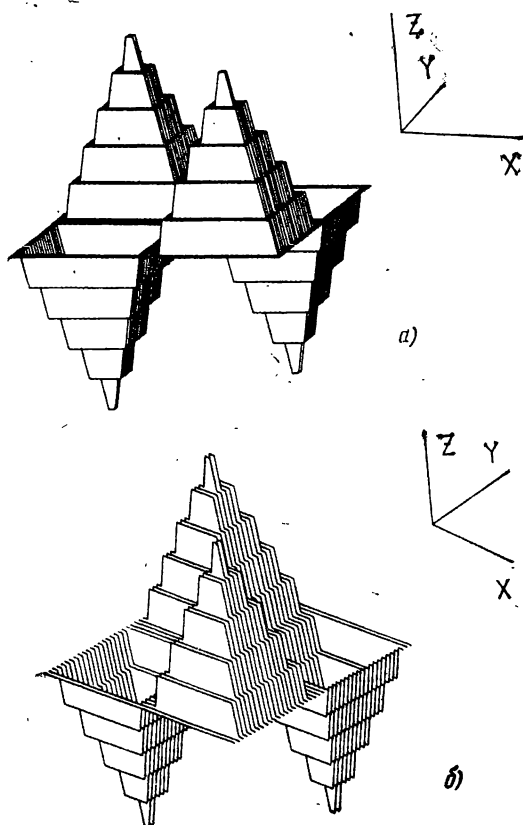


Рис. 8.2. Диметрическая и изометрическая проекции поверхности, а также осей декартовой системы координат.

лучу зрения. Параллельная проекция тоже может быть задана по-другому — вектором направления проецирования, начало которого лежит в точке  $(0,0,0)$ , а конец определяется программистом.

Таким образом, для построения произвольной проекции графического объекта достаточно сформировать матрицу преобразования, являющегося суперпозицией перечисленных выше основных преобразований координат. Умножая матрицу координат произвольной точки справа на матрицу результирующего преобразова-

ния, получим координаты проекции точки на картинную плоскость в соответствии с выбранным способом проецирования.

Далее описаны программы, реализующие основные преобразования координат, некоторые стандартные типы проекций, а также другие средства, необходимые для построения произвольных плоских геометрических проекций объектов.

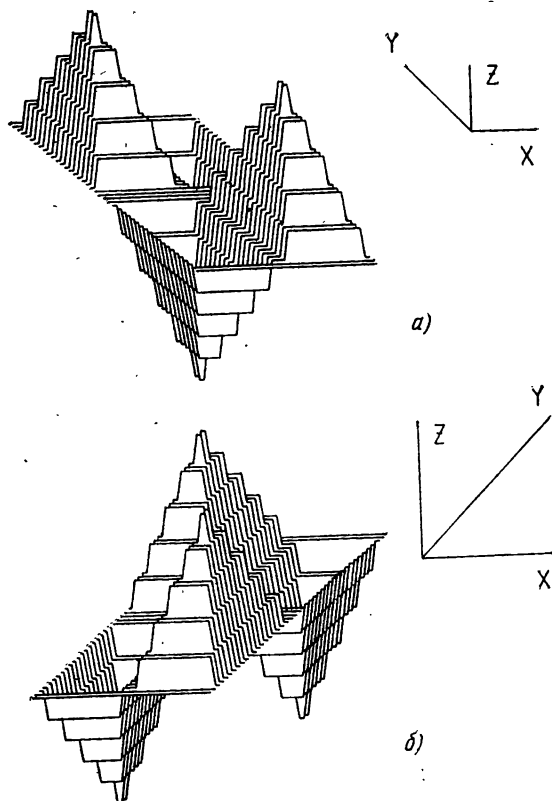


Рис. 8.3. «Отрицательная» (а) и «положительная» (б) косоугольные проекции поверхности, а также осей декартовой системы координат.

Программы преобразований. Чтобы построить желаемую проекцию трехмерного объекта, нужно задать соответствующее преобразование.

Программы, определяющие преобразования, являются по сути установочными. Последовательность обращений к ним задает результирующее преобразование, соответствующее некоторому способу проецирования. Программы рисования будут использовать

подготовленную матрицу преобразования для изображения объектов в выбранной проекции.

Каждая из программ, устанавливающих свое преобразование, формирует матрицу размером  $4 \times 4$  и умножает ее слева на матрицу текущего преобразования. В результате, преобразования будут выполняться в том порядке, в котором они задавались. Начальные установки выполняет программа INIT, которая формирует единичную матрицу. Обращение к ней отменяет уже накопленное преобразование. Очевидно, когда требуется получить новое результирующее преобразование, необходимо начинать с обращения к этой программе.

Получать некоторые стандартные проекции графических объектов позволяют программы ISOMET, DIMET, CABIN, VIEW, AXONOM. Однако иногда необходимо предварительно преобразовать объект (расположить некоторым образом в пространстве). Для этой цели можно воспользоваться программами, задающими поворот, масштабирование, перенос, сдвиг. Это программы: TDROT, TDSCAL, TDTRAN, SHEAR.

Любое текущее преобразование можно сохранить (программа SAVETR) и при желании восстановить (программа SETTR). Вообще с помощью программы SETTR можно установить в качестве текущего произвольное преобразование, расширив тем самым круг основных преобразований координат.

Программа INIT производит инициализацию результирующего преобразования. Программа без параметров.

Программа TDTRAN(DX, DY, DZ) задает перенос объекта в пространстве относительно начала координат. Параметры программы DX, DY, DZ определяют вектор переноса.

Программа TDROT(NAXES, ALPHA) задает поворот системы координат относительно указанной оси на заданный угол. Ее параметры:

NAXES — номер оси, относительно которой выполняется поворот: NAXES=1 — ось X, NAXES=2 — ось Y, NAXES=3 — ось Z (кроме того, если NAXES<0, угол поворота считается заданным в радианах, NAXES>0 — в градусах);

ALPHA — угол поворота:

ALPHA>0 — поворот выполняется против часовой стрелки относительно оси, вокруг которой выполняется поворот,

ALPHA<0 — поворот выполняется по часовой стрелке.

Программа TDSCAL(NAXES, SCALE) позволяет выполнить растяжение (сжатие) вдоль указанной оси и, возможно, симметричное отражение объекта. Параметры программы следующие:

NAXES — номер оси, вдоль которой выполняется растяжение (сжатие): NAXES=1 — ось X, NAXES=2 — ось Y, NAXES=3 — ось Z; NAXES=4 — растяжение (сжатие) по всем осям;



SCALE — коэффициент растяжения (сжатия):

$SCALE \geq 1$  — растяжение в SCALE раз,

$0 < SCALE < 1$  — сжатие в  $1/SCALE$  раз,

$SCALE < 0$  — симметричное отражение относительно соответствующей координатной плоскости или начала координат и растяжение в  $|SCALE|$  раз или сжатие в  $1/|SCALE|$  раз.

Программа SHEAR(I, J, F) определяет сдвиг. Параметры программы:

I — номер сдвигающей координаты: I=1 — координата X, I=2 — координата Y, I=3 — координата Z;

J — номер зависимой координаты;

F — коэффициент сдвига.

При I=J данное преобразование вырождается в преобразование масштабирования вдоль I-й оси с коэффициентом растяжения, равным F+1.

Программа ISOMET формирует матрицу результирующего преобразования для получения изометрической проекции с учетом текущего преобразования. Программа без параметров.

Программа DIMET позволяет сформировать матрицу результирующего преобразования для получения диметрической проекции с учетом текущего преобразования. Программа без параметров.

Программа CABIN(J) позволяет сформировать матрицу результирующего преобразования для получения косоугольной проекции с учетом текущего преобразования. Параметр программы J определяет вид косоугольной проекции. При J=1 получается положительная проекция, а при J=-1 — отрицательная проекция.

Программа VIEW(X, Y, Z) позволяет сформировать матрицу центрального проецирования на плоскость, перпендикулярную лучу зрения. Параметры программы:

X, Y, Z — координаты центра проекции (точки зрения).

Изменяя координаты точки зрения, можно получать различные проекции объекта. Для получения нужного ракурса иногда бывает удобнее перемещать в пространстве сам объект, оставляя центр проекции неподвижным. Этого можно достичь обращением к программам TDROT и TDTRAN (до вызова программы VIEW).

При обращении к программе VIEW надо следить, чтобы центр проекции не оказался внутри изображаемого объекта, иначе результаты работы программы рисования THREEED будут непредсказуемы.

Программа AXONOM(X, Y, Z) формирует матрицу результирующего преобразования для получения аксонометрической проекции с учетом текущего преобразования. Направление проециро-

вания определяется вектором, соединяющим точку  $(X, Y, Z)$  с началом координат.

Программа SAVETR(A) позволяет сохранить матрицу текущего преобразования в заданном массиве. Параметр программы: A — одномерный массив длины 16.

Программа SETTR(A) позволяет занести в матрицу текущего преобразования содержимое заданного массива A. Предполагается, что в массиве A последовательно записаны столбцы матрицы размером  $4 \times 4$ .

Вспомогательные и служебные программы.

Программа HCUNIT(A) формирует единичную матрицу A размером  $4 \times 4$ .

Программа HCMULT(A, B) перемножает две квадратные матрицы четвертого порядка  $A \times B$ . Результат помещается на место матрицы A.

Программа HCPSP(H) реализует преобразование центрального проецирования. Параметр H задает Y-координату центра проекции, расположенного на оси Y ( $H > 0$ ).

Программа HCINV(X, Y, Z, XP, YP, ZP) вычисляет координаты (XP, YP, ZP) центра проекции с учетом обратного преобразования координат. Предварительно вычисляется матрица обратного преобразования.

Программа HCR0T1(X, Y, Z) позволяет найти результирующее преобразование, переводящее двумя последовательными поворотами точку  $A(X, Y, Z)$  в точку в координатами  $(0, -\sqrt{X^2 + Y^2 + Z^2}, 0)$ .

**8.1.2. Построение проекций поверхностей. Удаление невидимых линий.** Класс трехмерных объектов, проекции которых могут быть изображены с помощью предлагаемых ниже программ, ограничивается поверхностями, заданными однозначной и непрерывной функцией двух переменных  $z = f(x, y)$ . Поверхность определяется следующим образом. На плоскости XY вводится прямоугольная сетка в общем случае с неравномерным шагом по осям X и Y. В узлах этой сетки задаются значения функции. Порядок расположения элементов сетки показан на рис. 8.1. Он отличается от расположения, принятого в методе ячеек, который описан в § 8.2 (см. рис. 8.9).

Для построения проекций определенной таким образом поверхности используется метод сечений. Поверхность пересекается рядом параллельных плоскостей. Кривые, образованные пересечением поверхности с секущими плоскостями, затем проецируются на картинную плоскость в соответствии с выбранным способом проецирования. Разработанные программы позволяют изобразить поверхность одним из следующих способов: либо сечения, покрывающие поверхность, располагаются параллельно плоскости XZ

исходной системы координат, либо секущие плоскости параллельны плоскости  $YZ$ , либо, наконец, поверхность пересекается одновременно двумя этими семействами плоскостей. В последнем случае поверхность оказывается покрытой сеткой, состоящей из криволинейных четырехугольников (криволинейная сетка). На рис. 8.4 показаны примеры таких сечений.

Задача построения изображения поверхности сводится, таким образом, к проецированию на картинную плоскость точек отдельных сечений, а именно тех точек плоскости  $XU$ , в которых заданы значения функции. Полученные образы точек затем соединяются отрезками, образуя ломаную линию — проекцию данного сечения на картинную плоскость. Это следует учитывать при задании поверхности. Например, если она не очень гладкая, то сетку, на которой задана функция, необходимо сделать более частой.

На рис. 8.4 показаны изометрические проекции некоторой поверхности. На нижнем рисунке при изображении использовались сечения, параллельные плоскости  $YZ$ , на верхнем — параллельные плоскости  $XZ$ , а на среднем рисунке поверхность покрыта криволинейной сеткой.

Программа **THREED**, выполняющая графические построения, позволяет выбирать тот или иной из трех упомянутых способов изображения проекций. Не следует, однако, думать, что криволинейная сетка всегда является лучшим из способов. Во-первых, он требует примерно вдвое больше времени по сравнению с двумя другими (как процессорного, так и времени работы графопостроителя). Во-вторых, опыт показывает, что если угол между направлением проецирования и секущими плоскостями не превышает  $5-10^\circ$ , то для зрительного восприятия рисунки получаются не совсем удачными. В этом случае криволинейная сетка не обеспечивает должного качества изображения. Лучше выбрать один из двух оставшихся способов, а именно тот, при котором этот угол близок к  $90^\circ$ .

Для достижения большей наглядности изображения чаще всего требуется, чтобы невидимые участки поверхностей не рисовались. Основная идея алгоритма удаления невидимых линий, разработанного применительно к поверхностям, изображаемым способом параллельных сечений, достаточно хорошо известна и заключается в следующем.

Выбирается такой порядок построения сечений, при котором каждое сечение может быть закрыто только участками поверхности, определяемыми ранее построенными сечениями. В программе **THREED**, выполняющей графические построения, этот порядок обхода устанавливается автоматически в соответствии с выбранным направлением проецирования или заданным центром проекции. Причем в случае перспективной проекции в зависимости от рас-

положения точки зрения относительно прямоугольной области определения функции поверхность при необходимости может быть автоматически разбита на 2 или 4 участка. Для каждого из них

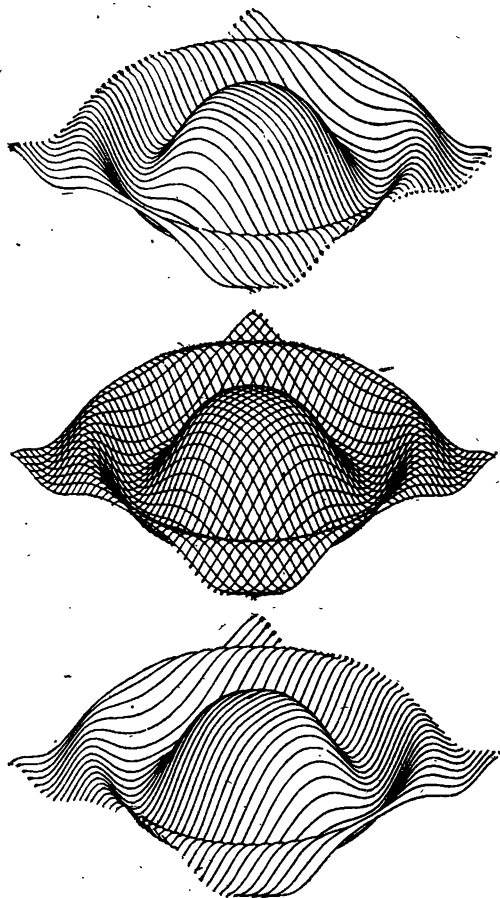


Рис. 8.4. Изометрические проекции поверхности с удаленными невидимыми линиями, построенные сечениями, параллельными плоскости  $XZ$  (верхний рисунок), сечениями, параллельными плоскости  $YZ$  (нижний рисунок), и способом «криволинейная сетка» (средний рисунок).

определяется свой порядок обхода сечений и каждый участок строится затем отдельно.

Если такой порядок обхода сечений фиксирован, то видимость каждой точки сечения может быть определена следующим образом.

Разобьем на картинной плоскости отрезок оси  $X$ , равный ширине рисунка, точками  $X_i$ ,  $i = 1, \dots, N$ , на  $N$  равных частей. Будем при этом считать, что на картинной плоскости задана физическая сетка. Рассмотрим на плоскости рисунка многоугольник, вершины которого расположены на прямых  $X_i$ ,  $i = 1, \dots, N$ . Этот многоугольник может быть невыпуклым, иногда вырожденным. Будем говорить в этом случае, что на плоскости рисунка существует экран. Этот экран представляет собой контур проекции участка поверхности, определяемого построенными к данному моменту сечениями. Выбранный порядок построения сечений позволяет использовать экран для определения видимости точек строящегося сечения. Если проекция точки попадает внутрь экрана, то такая точка считается невидимой. Рисоваться, таким образом, будут те участки сечений, которые расположены вне экрана. После построения проекции очередного сечения границы экрана изменяются в соответствии с построенным сечением.

Видимый участок сечения накапливается в виде массива абсцисс и ординат видимых точек этого участка. Служебная программа HCLINE использует для рисования видимого участка сечения программу LINEO. Заменяв программу HCLINE, можно, например, рисовать видимые и невидимые участки сечений линиями разного цвета или интенсивности, рисовать невидимые линии пунктиром и т. д.

Определение видимости отрезков сечений производится с точностью до границ экрана, которые лишь приближенно отслеживают реальный контур спроецированного к данному моменту участка поверхности. Поэтому на рисунках возможно появление незначительных дефектов. Делая физическую сетку более частой, можно уменьшить возможность появления таких искажений.

Границы экрана хранятся в массиве  $AMXMN$  размером  $2N$ . Первые  $N$  элементов отводятся под хранение *ломаной максимального рельефа* ( $MXR$ ), последние  $N$  элементов — для хранения *ломаной минимального рельефа* ( $MNR$ ). Поскольку для всех точек физической сетки, где экран уже существует,  $MXR(I) \geq MNR(I)$ , то перед рисованием в массив  $MXR$  заносятся нули, а в массив  $MNR$  — единицы. Это означает, что в области рисунка экрана еще нет. Если возникает необходимость на одном и том же рисунке изобразить две или более проекций поверхностей с учетом того, что они могут закрывать друг друга, то рисование последующих проекций надо производить, сохраняя то значение экрана, которое осталось после рисования предыдущих поверхностей. С этой целью в программе THREEED предусмотрена возможность выбирать один из двух режимов: 1) инициализировать экран, т. е. занести в  $MXR$  нули, а в  $MNR$  единицы, 2) сохранить накопленный

перед данным обращением к программе THREEED экран. Используя вторую возможность, можно, например, рисовать некоторые неоднозначные поверхности.

Определение пределов изменения функции. При построении проекций поверхностей сохраняется обычный принятый в Графоре порядок работы. Сначала заводится страница, в ней определяется область (REGION) и устанавливаются пределы изменения функции. В трехмерном случае для установки пределов используется программа TDLIM. Эта программа вначале находит минимальные и максимальные значения координат точек поверхности, спроецированных на картинную плоскость:  $XMIN$ ,  $XMAX$ ,  $ZMIN$ ,  $ZMAX$ . Затем выбирается одна из следующих возможностей: либо устанавливаются пределы изменения, равные найденным, либо в качестве пределов изменения берутся значения, общие среди найденных и сформированных при предыдущих обращениях к программе TDLIM. В последнем случае можно рисовать в одной области несколько поверхностей, сохраняя на рисунке их взаимное расположение в пространстве. В частности, так можно получать проекции и некоторых неоднозначных поверхностей (см. пример 3 в п. 8.1.4).

Заметим, что пределы, определяемые программой TDLIM, задают в пространстве математических координат некоторый прямоугольник. Если прямоугольная область, определяемая программой REGION, не является подобной этой математической области, то изображение в плоскости рисунка произвольно подвергается еще одному сжатию или растяжению по одной из осей. Для того чтобы получить рисунок, который сохранял бы пропорции, определяемые данным способом проецирования, область графика надо задавать подобной математической области. С этой целью в программе TDLIM в качестве выходного параметра выдается коэффициент, равный отношению длины математической области к ее ширине. Учитывая этот коэффициент при задании размеров области (т. е. при вызове программы REGION), можно добиться сохранения желаемых пропорций.

Еще один способ сохранить правильные пропорции состоит в том, чтобы не использовать программу TDLIM, а устанавливать пределы изменения спроецированных на картинную плоскость координат точек поверхности с помощью программы LIMITS.

Описание программ.

Программа TDLIM(X, Y, Z, NY, NX, ISTA, IFN, JST, JFN, S) предназначена для задания пределов изменения функции на картинной плоскости. Пределы ищутся во всей области определения функции либо в некоторой ее прямоугольной подобласти. Обращение к программе может быть до и после задания REGION, но обязательно после того, как обращениями к программой преобра-

зования координат сформировано результирующее преобразование. Эти пределы сохраняются до очередного обращения к программам, устанавливающим пределы. Параметры программы следующие:

$X, Y$  — массивы точек сетки по осям  $X$  и  $Y$ , расположенных в порядке возрастания;

$Z$  — двумерный массив значений функции в узлах сетки (размером  $(NY, NX)$ );

$NX, NY$  — число точек сетки по осям  $X$  и  $Y$ ;

$ISTA, IFN$  — индексы сетки по оси  $Y$ , определяющие нижнюю и верхнюю границы подобласти:

$ISTA > 0$  — в области рисования устанавливаются найденные для данной функции пределы,

$ISTA < 0$  — в области рисования устанавливаются пределы, общие среди найденных и тех, которые были получены при предыдущих обращениях к программе TDLIM;

$JST, JFN$  — индексы сетки по оси  $X$ , определяющие левую и правую границы подобласти;

$S$  — коэффициент, определяющий форму математической области значений функции на картинной плоскости; он равен отношению  $(XMAX - XMIN)/(ZMAX - ZMIN)$ , где  $XMIN, XMAX, ZMIN, ZMAX$  — пределы, устанавливаемые в области рисования.

Программа THREED( $X, Y, Z, NY, NX, ISTA, IFN, JST, JFN, LNTP, NGRD, AMXMN, AR$ ) предназначена для того, чтобы в соответствии с установленными пределами и выбранным способом проецирования построить в области рисования проекцию всей поверхности либо ее некоторого участка одним из имеющихся способов. Рисование может быть выполнено с удалением невидимых линий или без него. В первом случае при рисовании сечениями, параллельными плоскостям  $XZ$  или  $YZ$ , старое значение экрана, оставшееся от предыдущих обращений к программе THREED, может быть сохранено либо перед началом рисования уничтожено. При изображении криволинейной сетки старое значение экрана перед рисованием уничтожается. Программа имеет следующие параметры:

$X, Y$  — массивы точек сетки по осям  $X$  и  $Y$ , расположенных в порядке возрастания;

$Z$  — двумерный массив значений изображаемой функции в узлах сетки (размером  $(NY, NX)$ );

$NX, NY$  — число точек сетки по осям  $X$  и  $Y$ ;

$ISTA, IFN$  — индексы сетки по оси  $Y$ , определяющие нижнюю и верхнюю границы подобласти;

$ISTA > 0$  — рисование с удалением невидимых линий,

$ISTA < 0$  — рисование без удаления невидимых линий;

$JST, JFN$  — индексы сетки по оси  $X$ , определяющие левую и правую границы подобласти;

$JSTA > 0$  — перед рисованием старое значение экрана уничтожается,

$JSTA < 0$  — рисование выполняется с учетом имеющегося экрана;

LNTP — параметр, определяющий способ построения проекции:

LNTP = -1 — проекция строится сечениями матрицы значений, параллельными плоскости YZ (таких сечений  $JFN - JSTA + 1$ ),

LNTP = 1 — проекция строится сечениями матрицы значений, параллельными плоскости XZ (таких сечений  $IFN - ISTA + 1$ ),

LNTP = 0 — проекция строится способом криволинейной сетки;

NGRD — число делений физической сетки (т. е. размер экрана);

AMXMN — рабочий массив для хранения границ экрана (длины  $2 * NGRD$ );

AR — рабочий массив, используемый для выделения видимого участка сечения (длины  $2 * MAX(NY, NX)$ ).

З а м е ч а н и е. Если  $ISTA < 0$ , т. е. удаление невидимых линий не производится, то значения параметров NGRD и AMXMN, а также знак JSTA безразличны. В этом случае вместо AMXMN можно использовать любую переменную или массив, описанный в программе, которая обращается к THREEED.

Вспомогательные и служебные программы.

Программа HCNCRD(X, Y, Z) позволяет по заданным координатам точки в декартовой системе определить координаты этой точки после применения к ней результирующего преобразования. Новые координаты помещаются в общий блок, описываемый как COMMON/GFCRD/XP, YP, ZP.

Программа HCSURF(X, Y, Z, NY, NX, ISTA, IFN, JSTA, JFN, LNTP, MOUX, MOUY, AMXMN, AR) служит для построения в области рисования проекции заданного участка поверхности одним из двух способов: сечениями, параллельными плоскости XZ, (LNTP = 1) или сечениями, параллельными плоскости YZ, (LNTP = -1), — с указанным направлением обхода по строкам и столбцам матрицы значений Z. Если MOUX = -1, то элементы строки выбираются справа налево, если же MOUX = 1, то — слева направо. При MOUY = 1 элементы столбца выбираются в таком порядке, как они расположены в матрице Z, а при MOUY = -1 — в обратном порядке. Смысл остальных параметров тот же, что и в программе THREEED.

Ф у н к ц и я HCIND(X, IST, IFN, EL) позволяет для заданного числа EL найти среди элементов массива X, расположенных в порядке возрастания, элемент с таким индексом  $IST \leq I < IFN$ , что для него выполняется соотношение  $X(I) \leq EL \leq X(I+1)$ .

Программа TDSECT(ZIJ, X, Y, IX, IY, M, N, AMXMN, AR) предназначена для построения проекции одного сечения с указан-



лым направлением перебора точек сечения. Построение может быть выполнено как с удалением невидимых линий, так и без него.

Программа HCLINE(X, Y, NP) рисует линию по заданным массивам X и Y длины |NP|. Если  $NP < 0$  (участок невидим), линия не рисуется.

Функция ZINT(X1, Z1, X2, Z2, X) находит значение функции в точке X методом линейной интерполяции по заданным точкам (X1, Z1) и (X2, Z2).

**8.1.3. Построение проекций пространственных кривых.** В этом пункте приводится описание программ, позволяющих строить проекции пространственных кривых. Кривые задаются тремя одномерными массивами X, Y, Z размерности N, где N — количество точек, определяющих кривую. Так же как и при построении проекций поверхностей, заводится страница, в ней задается область для рисования. Для установления пределов служит программа TDLIML. Принцип работы этой программы, а также обеспечиваемые ею возможности такие же, как и у программы TDLIM. Непосредственное построение выполняет программа TDLINE.

Программа TDLIML(X, Y, Z, N, S) предназначена для задания пределов изменения проекции пространственной кривой на картинной плоскости. Обращение к программе может производиться до и после задания области рисования, но обязательно после того, как обращениями к программам преобразования координат будет сформировано результирующее преобразование, соответствующее выбранному способу проецирования. Эти пределы сохраняются до очередного обращения к программам установки пределов (TDLIML, TDLIM, LIMITS и др.). Параметры программы:

X, Y, Z — массивы длины |N|, определяющие X-, Y- и Z-координаты точек пространственной кривой;

|N| — количество точек, задающих кривую;

$N > 0$  — в области рисования устанавливаются найденные для данной кривой пределы,

$N < 0$  — в области рисования устанавливаются пределы, общие среди тех, которые были найдены для данной кривой, и полученных при предыдущих обращениях к программе TDLIML;

S — коэффициент, определяющий форму математической области, в которую попадает проекция кривой на картинной плоскости, и равный отношению длины этой области к ее высоте.

Программа TDLINE(X, Y, Z, N) позволяет в соответствии с установленными пределами и выбранным способом проецирования построить проекцию пространственной кривой. Параметры программы следующие:

X, Y, Z — массивы длины N, определяющие соответственно X-, Y- и Z-координаты точек пространственной кривой;

N — количество точек, задающих кривую.

**8.1.4. Примеры.** Ряд примеров иллюстрирует использование описанных выше программ.

1. На рис. 8.4 показаны изометрические проекции поверхности, построенные различными способами, с удалением невидимых линий. Массивы, описывающие поверхность, заполнялись подпрограммой SURF1. Рисунок строился с помощью следующей программы:

```
DIMENSION X(41), Y(41), Z(41, 41), A(200), AR(82)
CALL SURE1(X, Y, Z)
CALL INIT
CALL ISOMET
CALL PAGE(17., 26., 0, 0, 0)
CALL TDLIM(X, Y, Z, 41, 41, 1, 41, 1, 41, S)
SY=2.
LNTP=-1
DO 3 I=1, 3
CALL REGION(2., SY, 13., 7.5, 0, 0, 0)
CALL THREEED(X, Y, Z, 41, 41, 1, 41, 1, 41, LNTP,
, 100, A, AR)
SY=SY+8.
3 LNTP=LNTP+1
CALL ENDPG('8.4').
END
```

2. На рис. 8.5 показаны три различные центральные проекции одной и той же поверхности. При их построении точка зрения оставалась неизменной, а поворачивалась сама поверхность вокруг оси Z. С этой целью каждый раз перед обращением к программе VIEW матрица преобразования запоминалась, и это состояние матрицы восстанавливалось перед очередным поворотом поверхности. Массивы, задающие поверхность, формировались подпрограммой SURF2.

```
DIMENSION X(61), Y(61), Z(61, 61), A(400),
, AR(122), A1(16)
CALL SURF2(X, Y, Z)
CALL INIT
CALL PAGE(17., 26., 0, 0, 0)
SY=1.5
DO 1 I=1, 3
CALL REGION(1., SY, 15., 7., 0, 0, 0)
CALL SAVETR(A1)
CALL VIEW(-42., -42., 45.)
CALL TDLIM(X, Y, Z, 61, 61, 1, 61, 1, 61, S)
```

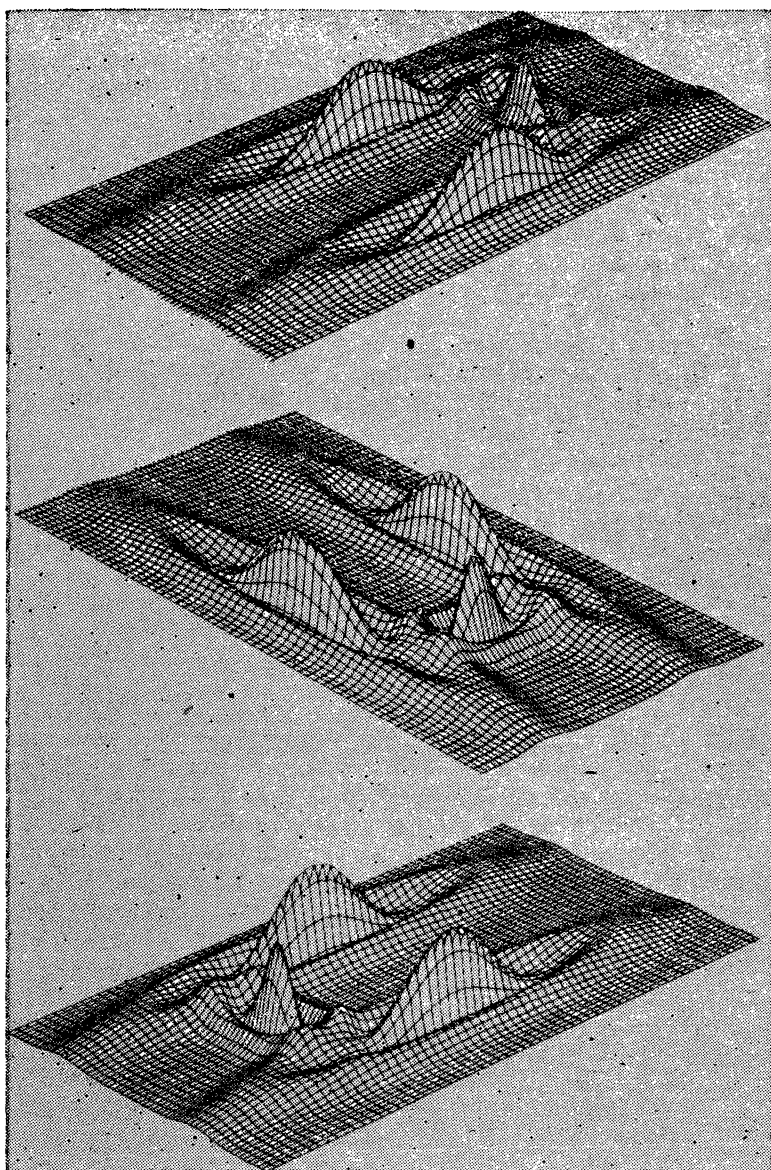


Рис. 8.5. Три различные центральные проеции одной и той же поверхности.

```

CALL THREEED(X, Y, Z, 61, 61, 1, 61, 1, 61, 0,
, 200, A, AR)
CALL SETTR(A1)
CALL TDROT(3, 90.)
1 SY=SY+8.
CALL ENDPG('8.5')
END

```

3. На рис. 8.6 изображена неоднозначная поверхность. При построении она разбивается на два участка, каждый из которых

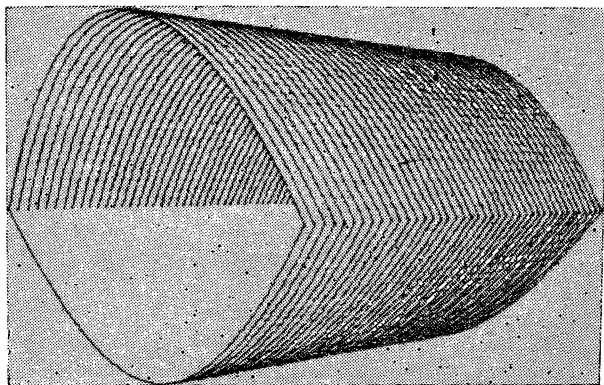


Рис. 8.6. Пример проекции неоднозначной поверхности.

является однозначной поверхностью в области задания. Для этих участков поверхности определяются общие пределы изменения с использованием имеющихся для этого возможностей в программе TDLIM. После определения пределов рисуется вначале тот участок, который по отношению к выбранной точке зрения расположен ближе к наблюдателю и, следовательно, не может быть закрыт вторым участком. Затем изображается второй участок с сохранением экрана, оставшегося после рисования первого участка. При этом рисование может проводиться сечениями, параллельными либо плоскости  $XZ$ , либо плоскости  $YZ$ .

```

DIMENSION X(41), Y(61), Z(61, 41), A(500),
' AR(122), T(16), T1(16)
DATA VX, VY, VZ/-10., -10., 2./, NY, NX/61, 41/
DO 1 I=1, NX
1 X(I)=-3.+6.*(I-1)/(NX-1)
DO 2 J=1, NY
2 Y(J)=-3.+6.*(J-1)/(NY-1)
DO 3 J=1, NY
DO 3 I=1, NX

```

```

3 Z(J, I)=9.-Y(J)**2
  CALL PAGE(17., 9., 0, 0, 0)
  CALL REGION(2., .5, 13., 8., 0, 0, 0)
  CALL INIT
  CALL VIEW(VX, VY, VZ)
  CALL SAVETR(T)
  CALL TDLIM(X, Y, Z, NY, NX, 1, NY, 1, NX, S)
  CALL INIT
  CALL TDSCAL(3, -1.)
  CALL VIEW(VX, VY, VZ)
  CALL SAVETR(T1)
  CALL TDLIM(X, Y, Z, NY, NX, -1, NY, 1, NX, S)
  CALL SETTR(T)
  CALL THREED(X, Y, Z, NY, NX, 1, NY, 1, NX, -1, 250,
, A, AR)
  CALL SETTR(T1)
  CALL THREED(X, Y, Z, NY, NX, 1, NY, -1, NX, -1, 250,
, A, AR)
  CALL ENDPG('8.6')
END

```

4. На рис. 8.7 изображена центральная проекция пространственной кривой. Задающие ее массивы формируются программой LINE3D.

```

  DIMENSION X(3417), Y(3417), Z(3417)
  CALL LINE3D(X, Y, Z)
  CALL PAGE(17., 17., 0, 0, 0)
  CALL REGION(1., 1., 15., 15., 0, 0, 0)
  CALL INIT
  CALL VIEW(15., -8., 3.)
  CALL TDLIML(X, Y, Z, 3417, FI)
  CALL TDLINE(X, Y, Z, 3417)
  CALL ENDPG('8.7')
END

```

5. На рис. 8.2 и 8.3, кроме проекций поверхностей, показаны также проекции на картинную плоскость осей декартовой системы координат. Их построение выполнялось с помощью подпрограммы TDAXES.

```

SUBROUTINE TDAXES(X, Y, REG, L)
COMMON/GFCRD/XP, YP, ZP
DIMENSION X1(3), Z1(3)
CALL HCNCRD(0., 0., 0.)
X0=XP
Z0=ZP

```

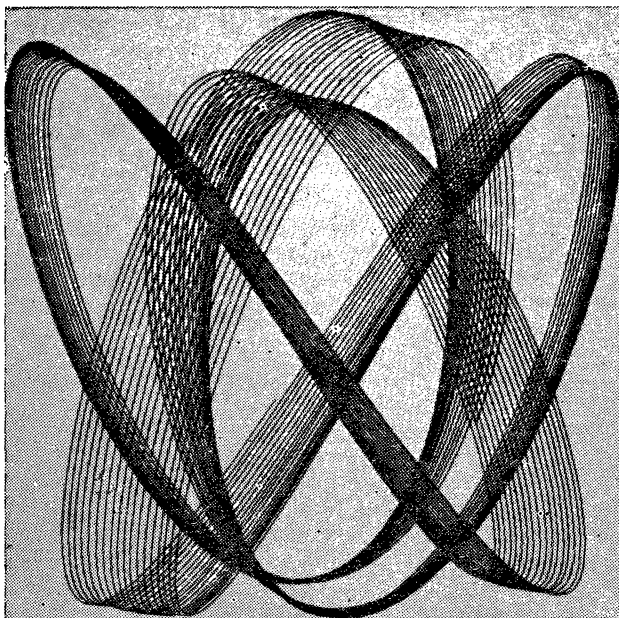


Рис. 8.7. Изображение центральной проекции пространственной кривой.

```

XMN=XP
XMX=XP
ZMN=ZP
ZMX=ZP
DO 7 J=1,3
IF (J.EQ.1) CALL HCNCRD(1., 0., 0.)
IF (J.EQ.2) CALL HCNCRD(0., 1., 0.)
IF (J.EQ.3) CALL HCNCRD(0., 0., 1.)
X1(J)=XP
Z1(J)=ZP
IF ((XP-XMN).LT.0.) XMN=XP
IF ((XP-XMX).GT.0.) XMX=XP
IF ((ZP-ZMN).LT.0.) ZMN=ZP
IF ((ZP-ZMX).GT.0.) ZMX=ZP
7 CONTINUE.
PX=XMX-XMN
PZ=ZMX-ZMN
DX=REG
DZ=REG
IF ((PX-PZ).LT.0.) DX=DZ*PX/PZ
IF ((PX-PZ).GT.0.) DZ=DX*PZ/PX

```

```

CALL REGION(X, Y, DX, DZ, ' ', 1, L)
CALL SET(0)
CALL LIMITS(XMN, XMX, ZMN, ZMX)
DO 8 J=1,3
CALL TMF(X0, Z0, XP, ZP)
CALL MOVE(XP, ZP, 0)
CALL TMF(X1(J), Z1(J), XP, ZP)
CALL MOVE(XP, ZP, 1)
IF (J.EQ.1) CALL SYMBOL(XP-.5,ZP-1.2,.7,'X',1,0.)
IF (J.EQ.2) CALL SYMBOL(XP-.5,ZP+.2,.7,'Y',1,0.)
IF (J.EQ.3) CALL SYMBOL(XP+.5,ZP-.8,.7,'Z',1,0.)
8 CONTINUE
RETURN
END

```

## 8.2. Изображение функций двух переменных. Метод ячеек

В этом параграфе описаны программы, позволяющие строить центральные проекции поверхностей, представленных однозначными непрерывными функциями двух переменных, заданными на прямоугольной сетке (в общем случае неравномерной), с удалением невидимых линий. Применяемый способ стирания невидимых линий основывается на упорядочении изображаемых элементов поверхности, которые в данном случае представляют собой ячейки криволинейной сетки.

**8.2.1. Алгоритм удаления невидимых линий.** Пусть на прямоугольной области  $R$  задана поверхность  $S$ , описываемая однозначной непрерывной функцией двух переменных. Без ограничения общности можем считать, что центр области  $R$  совпадает с началом координат, а ее стороны параллельны осям  $X$  и  $Y$ . В качестве картинной плоскости будем рассматривать плоскость  $P$ , проходящую через начало координат и перпендикулярную прямой, которая соединяет центр проекции  $V$  с началом координат (рис. 8.8). Определим на картинной плоскости прямоугольную систему координат  $X'Y'$  такую, что ось  $Y'$  является центральной проекцией оси  $Z$  на плоскость  $P$ .

Введем в области  $P$  прямоугольную сетку размером  $LLX \times LLY$  (не обязательно равномерную). Значения функции задаются в узлах сетки, а каждый элемент поверхности определяется четырьмя точками, в промежутках между которыми функция линейно интерполируется. Образом такой линейной фигуры на картинной плоскости будет четырехугольник, причем изображаться будут только видимые ребра (или их части).

Применяемый в этом случае алгоритм удаления невидимых линий основывается на упорядочении изображаемых элементов поверхности. При этом выбирается такой порядок их построения, при котором элемент, проецируемый в данный момент на картинную плоскость, не может закрывать другие, спроецированные ранее элементы поверхности. В результате тот или иной отрезок ребра невидим из точки  $V$  в том и только том случае, если его

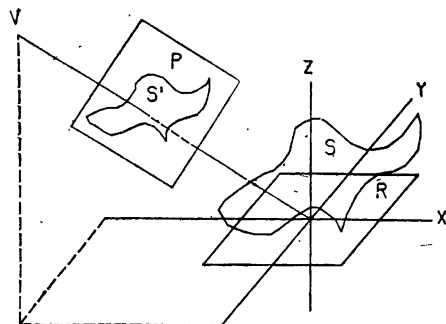


Рис. 8.8. Центральная проекция  $S'$  поверхности  $S$  на плоскость  $R$  (плоскость  $R$  для наглядности смещена относительно прямой, соединяющей центр проекции  $V$  с началом координат).

проекция попадает внутрь области, образованной проекциями предыдущих элементов. Образ поверхности на картинной плоскости достраивается по мере того, как берутся новые элементы.

Можно предложить несколько различных способов упорядочения элементов поверхности (точнее их ребер), при которых спроецированная часть поверхности не может быть закрыта последующими элементами. На рис. 8.9 показан один из возможных способов упорядочения (для юго-западного положения проекции  $V_0$  точки  $V$  на плоскость  $XY$  относительно области  $R$ ).

Такое упорядочение элементов поверхности позволяет определить экран, т. е. заключить невидимую область на картинной плоскости между двумя кусочно-линейными непрерывными функциями (ср. с п. 8.1.2). Одна из этих функций совпадает с верхней границей построенной к этому времени на картинной плоскости фигуры (ломаная максимального рельефа). Другая функция используется для определения нижней границы проекции (ломаная минимального рельефа). Изображаться при этом будут только те части элементов поверхности, проекции которых выходят за пределы экрана.

Для экономного задания экрана разобьем отрезок на оси  $X'$ , равный ширине изображения, на  $NV$  равных частей и будем задавать значения функций  $RMAX$  и  $RMIN$  в узлах этой равномерной



сетки. Таким образом, на картинной плоскости вводится физическая сетка. Заметим, что при этом шаг разбиения равен величине  $(X_{MAX} - X_{MIN})/NN$ , где  $X_{MAX}$ ,  $X_{MIN}$  — максимальное и минимальное значения  $X'$ -координат всех спроецированных на плоскость  $P$  точек поверхности.

В связи с тем, что границы экрана не всегда точно отображают форму видимой части проекции, в области рисунка обычно существуют участки, в которых могут появляться некоторые искажения: изменение угла наклона видимой части ребра, недоведение отдельных отрезков до границ экрана или, наоборот, вторжение некоторых отрезков в невидимую область. Эти дефекты, как правило, незначительны и мало влияют на качество изображения. Возможность появления дефектов такого рода можно уменьшить, сделав физическую сетку более частой.

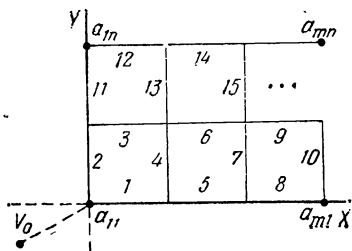


Рис. 8.9. Способ упорядочения ребер элементов поверхности (для юго-западного положения центра проекции).

Перед тем как начать проецирование, проводится инициализация экрана: в массив  $R_{MAX}$ , содержащий значения ломаной максимального рельефа, заносятся нули, а в массив  $R_{MIN}$ , описывающий ломаную минимального рельефа, — единицы. В результате можно легко находить те участки области рисунка, на которых экран еще не определен (в этих точках значения  $R_{MAX}$  меньше значений  $R_{MIN}$ ). После изображения каждого ребра элемента поверхности  $R_{MAX}$  или  $R_{MIN}$  (возможно и оба) будут модифицироваться, отражая тем самым изменение формы «нарастающего» образа поверхности на картинной плоскости.

**8.2.2. Выбор ракурса. Определение пределов изменения функции.** Способы упорядочения ребер элементов поверхности, аналогичные изображенному на рис. 8.9, можно предложить и для других, не только юго-западных угловых положений точки  $V_0$ . Если же  $V_0$  лежит непосредственно на западе, севере, востоке или юге от  $R$  или попадает внутрь  $R$ , то область  $R$  разбивается на две или четыре части (рис. 8.10), и алгоритм применяется к каждому из полученных прямоугольников, относительно которых  $V_0$  уже имеет угловое положение.

Перед изображением каждой из таких частей происходит инициализация экрана. Причина этого заключается в следующем.

Установлено (см. [1, г, 1, ж]), что при некоторых достаточно близких к изображаемому объекту положениях центра проекции рисунок часто получается неудачным для зрительного восприятия,

так как наблюдается нарушение монотонности изменения  $X'$ -ординаты на картинной плоскости. В результате некоторые на самом деле видимые отрезки не изображаются или изображаются только частично, что приводит к появлению непрорисованных белых пятен на изображении. В таких случаях рекомендуется сменить ракурс, например, отодвинув центр проекции от объекта

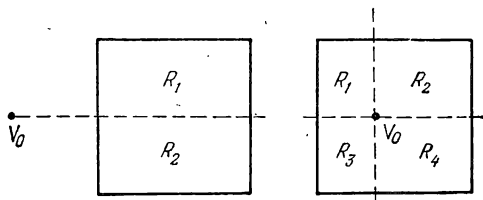


Рис. 8.10. Разбиение области  $R$  в случае неуглового положения  $V_0$ .

Если же проецируемая поверхность состоит из нескольких частей и перед изображением каждой из них (за исключением первой) экран не инициализируется, то вероятность появления белых пятен значительно возрастает.

Хотя инициализация экрана перед проецированием и очень желательна, однако при необходимости можно сохранить прежнее содержимое экрана, полученное при рисовании предыдущих частей поверхности, т. е. можно выбирать один из следующих режимов:

- а) инициализация экрана перед изображением каждой из частей поверхности;
- б) рисование с учетом экрана, оставшегося от изображения предыдущих частей поверхности.

Вторым режимом следует пользоваться только в тех случаях, когда возникает необходимость в одной и той же области нарисовать проекции двух или более поверхностей с учетом того, что они могут закрывать друг друга. В этом случае изображение каждой из последующих поверхностей надо производить с учетом значения экрана, которое осталось после проецирования предыдущих поверхностей. В связи с этим в программе PRSP, которая выполняет построение изображения, предусмотрены две возможности:

- а) инициализация экрана перед изображением объекта;
- б) сохранение старого значения экрана, оставшегося от предыдущих обращений к программе PRSP.

Используя вторую возможность, можно, в частности, получать проекции некоторых неоднозначных поверхностей (см. п. 8.2.4).

Центром проекции, вообще говоря, может быть любая точка пространства, не лежащая на самой поверхности. При этом, одна-

ко, следует иметь в виду, что всегда существует опасность выбрать такое взаимное расположение центра проекции и поверхности, при котором образы некоторых точек поверхности вообще могут не существовать, т. е. соответствующие проецирующие прямые будут параллельны картинной плоскости.

В связи с этим в алгоритм центрального проецирования был введен некоторый вариант отсеечения — ограничивающий *конус зрения* с углом между образующими  $160^\circ$ . Все точки поверхности, лежащие за его пределами, игнорируются. Таким образом, исключается возможность проецирования участков поверхности, не имеющих образов на картинной плоскости. Следует, однако, подчеркнуть, что наличие точек поверхности, не попавших в конус зрения, свидетельствует прежде всего о необходимости изменить положение центра проекции, например, отдалив его от поверхности. Поскольку угол между образующими конуса превышает угол ясного зрения ( $60^\circ$ ), то даже полное попадание в конус проецируемого участка поверхности еще не гарантирует получения хорошего изображения. Все эти обстоятельства необходимо принимать во внимание при выборе ракурса.

Для задания пределов изменения  $X'$ - и  $Y'$ -координат на картинной плоскости используется программа MNMX. Она выполняет те же функции, что и программа TDLIM, описанная в п. 8:1.2.

### 8.2.3. Описание программ.

Программа MNMX(VX, VY, VZ, X, Y, A, LLX, LLY, LXINI, LXE, LYIN, LYE, S) служит для задания пределов изменения изображаемой функции на картинной плоскости. Пределы устанавливаются либо для всей области задания функции, либо для некоторой ее прямоугольной подобласти. Обращение к программе может быть как до, так и после вызова программы REGION. Установленные пределы сохраняются до очередного обращения к программе MNMX. Программа имеет следующие параметры:

VX, VY, VZ — координаты центра проекции в системе координат XYZ;

X, Y — массивы координат точек сетки по осям X и Y, расположенные в порядке возрастания;

A — значения функции в узлах сетки (двумерный массив размером (LLX, LLY));

LLX, LLY — размеры сетки по осям X и Y;

LXINI, LXE — индексы сетки по оси X, определяющие левую и правую границы подобласти;

$LXINI > 0$  — в области рисования, определенной обращением к программе REGION, устанавливаются найденные для данной функции пределы,

$LXINI < 0$  — в области рисования устанавливаются пределы, общие для пределов изменения данной функции и преде-

лов, полученных при предыдущих обращениях к программе MNMX;

LYIN, LYE — индексы сетки по оси  $Y$ , определяющие нижнюю и верхнюю границы подобласти;

$S$  — коэффициент, определяющий форму математической области значений функции на картинной плоскости; он равен отношению  $(XMAX - XMIN)/(YMAX - YMIN)$ , где  $XMAX$ ,  $XMIN$ ,  $YMAX$ ,  $YMIN$  — пределы изменения функции, устанавливаемые в области рисования.

**З а м е ч а н и е.** При обращении к программе MNMX надо следить, чтобы центр проекции не попал на поверхность изображаемого объекта.

Программа PRSP(VX, VY, VZ, X, Y, A, LLX, LLY, LXINIT, LXEND, LYINIT, LYEND, XF1, YF1, RMAX, RMIN, NN) позволяет построить в области рисования центральную проекцию всей заданной поверхности либо ее некоторого участка с удалением невидимых линий. Программа имеет следующие параметры:

VX, VY, VZ — координаты центра проекции в системе координат XYZ;

X, Y — массивы координат точек сетки по осям  $X$  и  $Y$ , расположенных в порядке возрастания;

A — значения функции в узлах сетки (двумерный массив размером (LLX, LLY));

LLX, LLY — размер сетки по осям  $X$  и  $Y$ ;

LXINIT, LXEND — индексы сетки по оси  $X$ , определяющие левую и правую границы проецируемого участка поверхности:

LXINIT > 0 — перед рисованием экран иницируется, т. е. его старое значение, оставшееся от предыдущих обращений к программе PRSP, уничтожается,

LXINIT < 0 — рисование осуществляется с учетом экрана, оставшегося от предыдущих обращений к программе PRSP;

LYINIT, LYEND — индексы сетки по оси  $Y$ , определяющие нижнюю и верхнюю границы проецируемого участка поверхности:

LYINIT > 0 — перед изображением проецируемого участка поверхности или каждой из его частей, полученных при разбиении в случае неуглового положения центра проекции, происходит инициализация экрана,

LYINIT < 0 — рисование выполняется с учетом экрана, оставшегося от изображения предыдущих участков поверхности или их частей, полученных при разбиении (это дает возможность получить полное значение экрана от всего проецируемого участка поверхности независимо от положения центра проекции);

XF1, YF1 — рабочие массивы длины соответственно  $2 \times LLX$  и  $LLX$ , предназначенные для запоминания  $X$ - и  $Y$ -компонент координат точек сетки, расположенных вдоль оси  $X$ ;

RMAX, RMIN — массивы, используемые для хранения верхней и нижней границ экрана (они должны иметь длину не менее NN);  
NN — число делений физической сетки на области рисунка (т. е. размер экрана).

#### З а м е ч а н и я.

1) Если при изображении нескольких проекций поверхностей с учетом того, что они могут закрывать друг друга, центр проекции находится в угловом положении относительно первой поверхности, то при обращении к программе PRSP для первой поверхности знак LYINIT безразличен.

2) При обращении к программе PRSP надо следить, чтобы центр проекции не попал на поверхность изображаемого объекта.

Программа PRJT(VX, VY, VZ, X, Y, A, LLX, LLY, LXIN, LXE, LYIN, LYE, XF1, YF1) позволяет построить в области рисования центральную проекцию всей заданной поверхности либо ее некоторого участка без удаления невидимых линий. Параметры программы следующие:

LXIN, LXE — индексы сетки по оси X, определяющие левую и правую границы проецируемого участка;

LYIN, LYE — индексы сетки по оси Y, определяющие нижнюю и верхнюю границы проецируемого участка.

Остальные параметры имеют то же значение, что и в предыдущей программе.

**З а м е ч а н и е.** При обращении к программе рисования PRJT надо следить, чтобы центр проекции не попал на поверхность изображаемого объекта.

#### Вспомогательные и служебные программы.

Программа MAP(X, Y, Z) по заданным координатам точки в декартовой системе позволяет определить координаты проекции этой точки на картинную плоскость. Вычисленные координаты XP, YP помещаются в общий блок GFVP, описываемый как COMMON/GFVP/V1, V2, V3, XP, YP, INFIN. При этом параметр INFIN является признаком, определяющим, попадает ли данная точка поверхности в конус зрения (INFIN=0 — попадает, INFIN=1 — лежит за его пределами).

Программа CORNL(LXI1, LX1, LYJ1, LY1, K1, M1, K2, M2, X, Y, A, LXX, LYY, XF1, YF1, RMAX, RMIN) позволяет построить центральную проекцию части поверхности (задаваемой первыми четырьмя параметрами), относительно которой центр проекции имеет угловое положение. Построение производится с удалением невидимых линий. Параметры K1, K2 определяют координаты ближайшего к центру проекции угла проецируемой части поверхности, а M1, M2 задают порядок перебора элементов соответственно строк и столбцов рассматриваемой части поверхности (от начального элемента

к конечному или от конечного к начальному). Остальные параметры имеют те же значения, что и в программе PRSP.

Программа PATCH(XF1,YF1,RMAX,RMIN) служит для построения на картинной плоскости проекции одного элемента изображаемой части поверхности с удалением невидимых линий.

Программа SEARCH(IX1,Y1,IX2,Y2,XF1,YF1,RMAX,RMIN) предназначена для обработки частично видимых ребер элементов поверхности (первые два параметра задают видимую конечную точку ребра, а вторые два — невидимую). С этой целью на физической сетке ищется первая видимая точка ребра, после чего изображается отрезок, соединяющий видимую конечную точку с найденной точкой. (Параметры IX1, IX2 определены на физической сетке, введенной в области рисования, поэтому их значения — целые числа.)

Программа INTRSC(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X0,Y0) позволяет найти точку пересечения (X0, Y0) двух отрезков, заданных координатами конечных точек.

Программа BUFL(IX1,Y1,IX2,Y2,RMAX,RMIN) запоминает информацию о видимой части заднего ребра каждого элемента поверхности и после полного просмотра элемента модифицирует экран.

Программа LININT(JX1,Y1,JX2,Y2,RMAX,RMIN) позволяет линейно интерполировать функции RMAX или RMIN (возможно обе) между двумя заданными точками (JX1,Y1) и (JX2,Y2). При этом новые значения заносятся в RMAX (RMIN) только в том случае, если они больше (меньше) старых.

**8.2.4. Примеры.** Следующие примеры показывают различные возможности программ центрального проецирования.

1. На рис. 8.11 изображена поверхность, описывающая распределение концентрации полезного компонента на участке сложного структурного месторождения руды. Распределение получено в результате статистической обработки данных. Ниже приведена программа, с помощью которой строился этот рисунок. Массивы, описывающие поверхность, заполнялись подпрограммой MOUNT.

```
DIMENSION PHI(49, 103), XP(49), YP(103), XF(98),  
'YF(49), RX(2000), RN(2000)  
DATA A, B, C/1032., -1000., 1037./  
CALL MOUNT(XP, YP, PHI)  
CALL PAGE(15., 12., 0, 0, 0)  
CALL MNMX(A, B, C, XP, YP, PHI, 49, 103, 1, 49, 1,  
'103, S)  
CALL PRSP(A, B, C, XP, YP, PHI, 49, 103, 1, 49, 1,  
'103, XF, YF, RX, RN, 2000)  
CALL ENDPG('8.11')  
END
```

2. На рис. 8.12 показаны центральные проекции целой поверхности и некоторого ее участка. На среднем рисунке берутся пределы изменения, установленные для всей поверхности, а на верхнем — пределы, вычисленные только для рассматриваемого участка

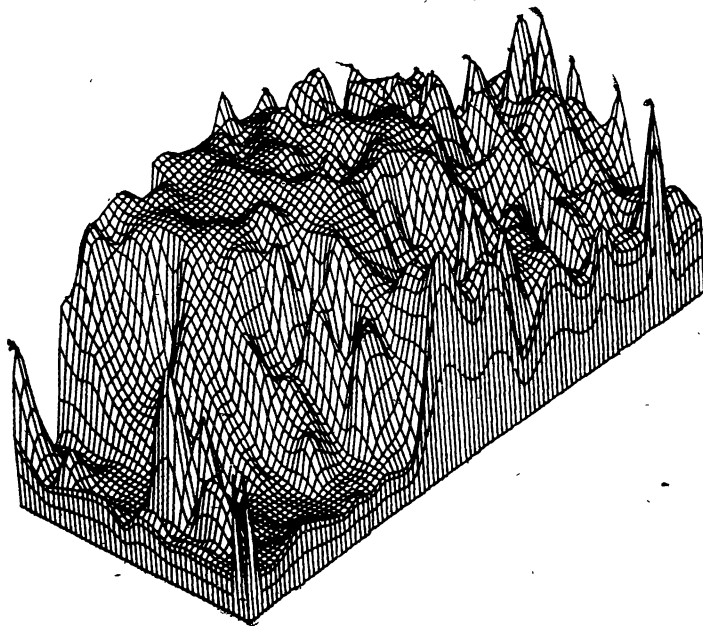


Рис. 8.11. Пример центральной проекции поверхности с удаленными невидимыми линиями.

поверхности. Изображения строились с помощью следующего фрагмента программы:

```
CALL PAGE(17., 26., 0, 0, 0)
CALL REGION(1., 2., 15., 7.5, 0, 0, 1)
CALL MNMX(35., 90., 10., X, Y, A, 37, 37, 1, 37, 1, 37, S)
CALL PRSP(35., 90., 100., X, Y, A, 37, 37, 1, 37, 1, 37,
' XF1, YF1, RMAX, RMIN, 300)
CALL REGION(1., 10., 15., 7.5, 0, 0, 1)
CALL PRSP(35., 90., 100., X, Y, A, 37, 37, 1, 18, 1, 27,
' XF1, YF1, RMAX, RMIN, 300)
CALL REGION(1., 18., 15., 7.5, 0, 0, 1)
CALL MNMX(35., 90., 100., X, Y, A, 37, 37, 1, 18, 1, 27, S)
CALL PRSP(35., 90., 100., X, Y, A, 37, 37, 1, 18, 1, 27,
' XF1, YF1, RMAX, RMIN, 300)
CALL ENDPG('8.12')
```

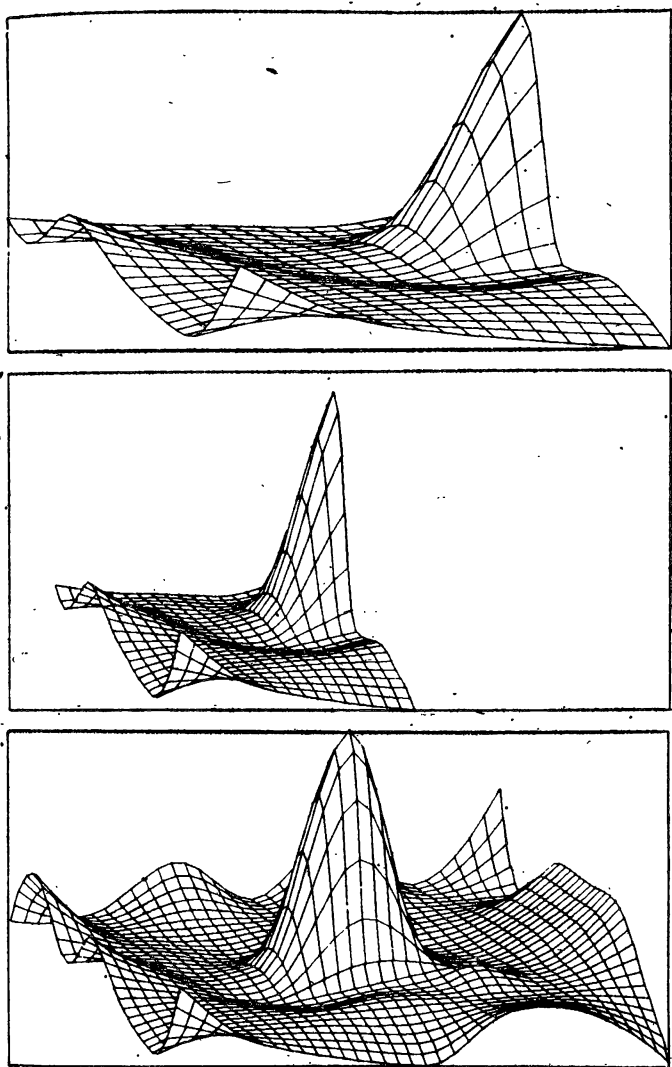


Рис. 8.12. Центральные проекции целой поверхности и некоторого ее участка.

3. При желании можно получить изображение поверхности, как бы стоящей на «постаменте» определенной высоты (рис. 8.13). Для этого крайним строкам и столбцам массива следует присвоить некоторое постоянное значение (в этом примере оно равно  $-0,3$ ).



```

DIMENSION X(42), Y(42), A(42, 42), XF1(84), YF1(42)
DIMENSION RMAX(500), RMIN(500)
DATA XA, YA, XB, YB/0.525, 1.025, 2*1.525/
DATA X(1), Y(1), X(2), Y(2)/-0.01, -0.01, 0., 0./
DO 85 I=3, 42

```

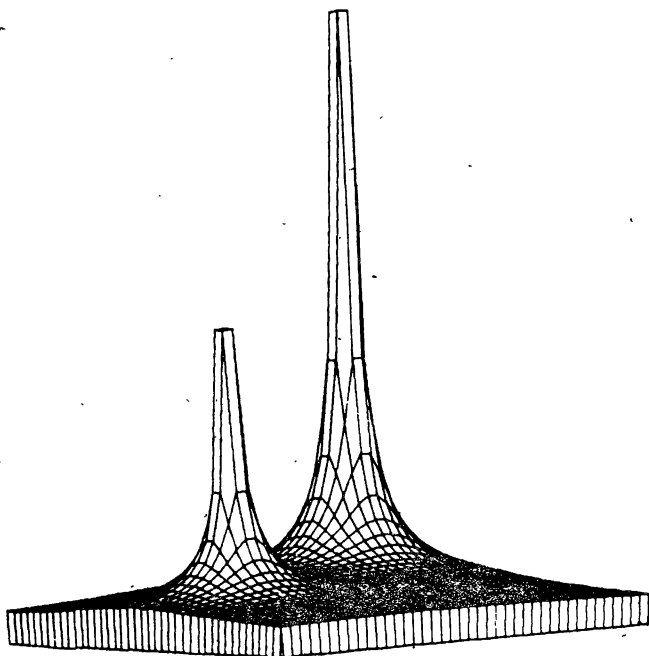


Рис. 8.13. Изображение алгебраической поверхности на «постаменте».

```

      X(I)=X(I-1)+.05
85  Y(I)=X(I)
      DO 86 K=1, 42
        A(1, K)=-0.3
86  A(K, 1)=-0.3
      DO 87 J=2, 42
        DO 81 I=1, 42
87  A(I, J)=.2*(1./SQRT((X(I)-XA)**2+(Y(J)-
* YA**2))+2./SQRT(X(1)-XB)**2+(Y(J)-YB)**2))
      CALL PAGE(17., 15., 0, 0, 0)
      CALL REGION(2.5, 2., 12., 12., 0, 0, 0)
      CALL MNMX(-15., -20., 10., X, Y, A, 42, 42, 1, 42, 1, 42, S)

```

```
CALL PRSP(-15., -20., 10., X, Y, A, 42, 42, 1, 42, 1, 42,
'XF1, YF1, RMAX, RMIN, 500)
CALL ENDPG('8.13')
END
```

4. На рис. 8.14 показана проекция неоднозначной поверхности. Заданная поверхность разбивается на два участка, каждый

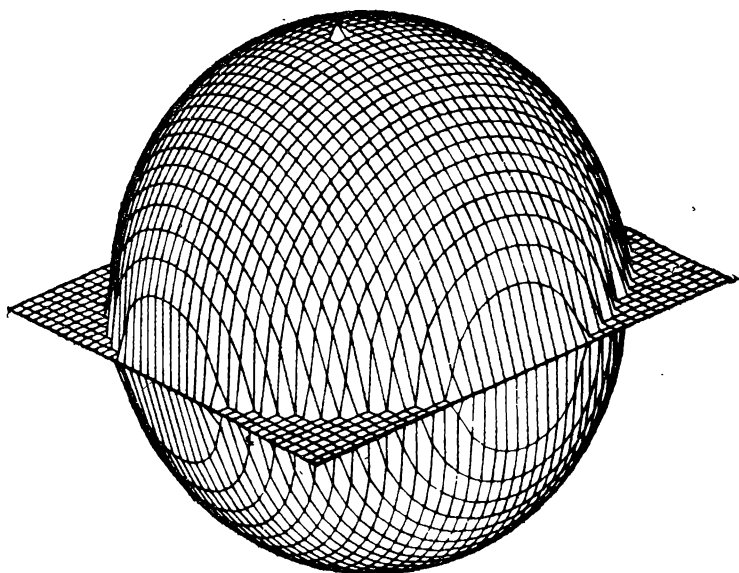


Рис. 8.14. Пример проекции неоднозначной поверхности.

из которых является однозначной функцией в своей области определения. С помощью программы MNMX ищутся пределы изменения для обоих участков поверхности. Далее, сначала изображается тот участок поверхности, который расположен ближе к выбранному центру проекции и, следовательно, не может быть закрыт другим участком. Только после этого проецируется второй участок поверхности, причем с сохранением экрана, оставшегося после рисования первого участка.

```
DIMENSION X(41), Y(41), A(41, 41), B(41, 41)
DIMENSION XF1(82), YF1(41), RMAX(500),
' RMIN(500)
X(1)=-2.
Y(1)=-2.
DO 1 I=2,41
X(I)=X(I-1)+0.1
```

```

1 Y(I)=X(I)
  DO 5 I=1, 41
    DO 5 J=1, 41
      D=X(I)*X(I)+Y(J)*Y(J)-4.
      IF(D) 4, 3, 3
3 A(I, J)=0.
  GOTO 5
4 A(I, J)=SQRT(-D)
5 B(I, J)=-A(I, J)
  CALL PAGE(17., 13., 0, 0, 0)
  CALL REGION(2., 2., 13., 10., 0, 0, 0)
  CALL MNMX(-50., -75., 40., X, Y, A, 41, 41, 1, 41, 1, 41, S)
  CALL MNMX(-50., -75., 40., X, Y, B, 41, 41, -1, 41, 1, 41, S)
  CALL PRSP(-50., -75., 40., X, Y, A, 41, 41, 1, 41, -1, 41,
    ' XF1, YF1, RMAX, RMIN, 500)
  CALL PRSP(-50., -75., 40., X, Y, B, 41, 41, -1, 41, -1, 41,
    ' XF1, YF1, RMAX, RMIN, 500)
  CALL ENDPG('8.14')
END

```

### 8.3. Триангуляция и изображение функции двух переменных, заданной в произвольно расположенных точках

В этом параграфе будет рассмотрен случай, когда функция задана в узлах, которые не образуют регулярную прямоугольную сетку. Более того, область определения может быть неоднозначной. Тогда целесообразно, применив триангуляцию, по заданной границе области и узлам внутри нее сформировать треугольную сетку. После этого можно построить проекцию поверхности, которая описывается исследуемой функцией. В последнем случае мы неизбежно сталкиваемся с проблемой удаления невидимых линий. Для ее решения применен существенно более общий метод, чем в §§ 8.1 и 8.2.

**8.3.1. Триангуляция.** Пусть на плоскости  $XU$  определена некоторая, возможно, несвязная область. Ее границы заданы одной или несколькими непересекающимися замкнутыми ломаными и внутри области задано некоторое количество точек. Задача триангуляции состоит в том, чтобы построить в этой области треугольную сетку, узлами которой служат все вершины ломаной и все заданные точки. Пример области и одно из возможных решений для нее приведены на рис. 8.15.

Известно, что при вычислениях обеспечивается более высокая точность, если каждая точка в сетке соединяется с одинаковым числом соседних точек. В идеальном случае сетка должна состоять из равносторонних треугольников. На практике при произволь-

ном расположении точек следует избегать тупоугольных вытянутых треугольников и по возможности добиваться, чтобы треугольники, образующие сетку, были остроугольными.

Рассмотрим алгоритм триангуляции, реализованный в одной из описанных ниже программ. Пусть в области, предназначенной

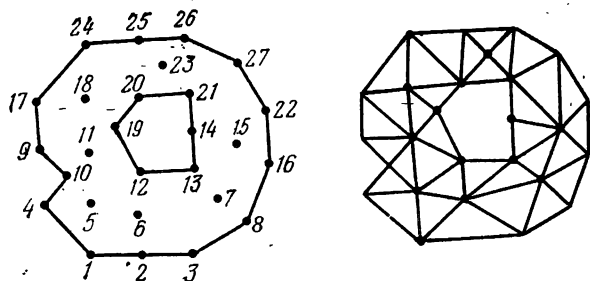


Рис. 8.15. Построение треугольной сетки в заданной области.

для триангуляции, все узловые точки перенумерованы, например так, как на рис. 8.15. Чтобы произвести разбиение области на треугольники, необходимо научиться решать задачу нахождения третьей вершины треугольника, основанием которого будет граничный отрезок. Тогда последовательным отщеплением треугольников и переопределением области можно будет решить и задачу триангуляции.

Какая же из узловых точек области считается наиболее подходящей вершиной треугольника. Поскольку граница области предполагается ориентированной, то известно, по какую сторону отрезка должна находиться искомая точка (рис. 8.16). Среди всех точек, лежащих по указанную сторону отрезка  $[A, B]$ , выбирается точка  $K$ , для которой расстояние до середины отрезка минимально.

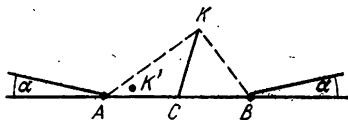


Рис. 8.16. К нахождению вершины треугольника, в основании которого — граничный отрезок.

При этом предпочтение отдается точкам, не попадающим в секторы  $\alpha$ . В данном случае угол выбран таким, что  $\sin \alpha = 0.2$ . Однако при таком выборе точки  $K$  в треугольник  $ABK$  могут попасть узловые точки, для которых расстояние до середины отрезка  $AB$  меньше его половины ( $CK' < 1/2 AB$ ). Если таковые имеются, они отмечаются. Точка  $K$  заменяется точкой  $K'$ , рассматривается новый треугольник и остальные отмеченные точки. Процесс повторяется до тех пор, пока внутри полученного треугольника не останется узловых точек.

Выбор третьей вершины треугольника проводится среди так называемых доступных вершин. Первоначально в список доступных вершин включаются все узловые точки. При последовательном отщеплении треугольников из этого списка исключаются точки, которые оказались вне текущей границы области.

Заметим, что последовательность граничных отрезков выбирается таким образом, что отщепленные треугольники по мере их получения образуют цепочку, скручивающуюся против часовой стрелки к центру области. Такой порядок отщепления позволяет надеяться, что

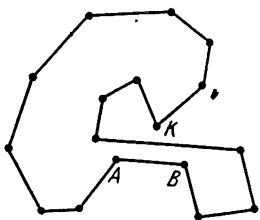


Рис. 8.17. Пример области, для которой разбиение на треугольники будет выполнено неправильно.

если вначале граница области была не слишком изломанной, то она останется такой в процессе модификации. Это важно, поскольку алгоритм не безупречен. На рис. 8.17 приведен пример, когда разбиение будет выполнено неправильно. Для отрезка  $AB$  в качестве третьей вершины будет выбрана точка  $K$ , но полученный треугольник  $ABK$  не лежит целиком в области. Однако можно предположить, что на практике будут рассматриваться области с более или менее равномерным распределением точек и «хорошей» границей.

Рассмотрим вопросы, связанные с заданием и хранением информации об узловых точках области, ее границе и полученной треугольной сетке. Программа триангуляции TRINGL предполагает, что значения координат узловых точек или вершин находятся в массивах  $X$  и  $Y$ . Все узлы считаются перенумерованными от 1 до  $N_0$ , где  $N_0$  — общее количество узлов. Будем считать, что граничный отрезок  $[i, j]$  выходит из точки  $i$  и входит в точку  $j$ , если при движении из  $i$  в  $j$  область остается слева.

Для областей, ограниченных замкнутыми непересекающимися ломаными, из каждой граничной вершины будет выходить только один отрезок. Но даже в случае таких областей процесс отщепления треугольников почти всегда приводит к ситуации, в которой оставшаяся часть области будет ограничена пересекающимися замкнутыми ломаными, при этом из граничных вершин будут выходить два отрезка. В то же время три и более отрезков практически не встречаются при выбранном порядке отщепления треугольников. Учитывая это и исходя из того, что изменения при отщеплении треугольников должны быть минимальными, был выбран способ представления информации о границе.

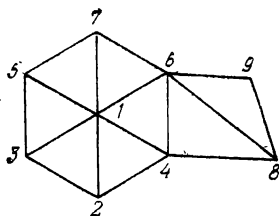
В двумерный массив  $IBOUND$  размером  $(2, N_0)$  заносится следующая информация о каждой узловой точке области:

а)  $IBOUND(1, i) = 0$ ,  $IBOUND(2, i) = 0$ , если точка  $i$  не граничная;

б)  $IBOUND(1, i) = j_1$ ,  $IBOUND(2, i) = 0$  или  $IBOUND(1, i) = 0$ ,  $IBOUND(2, i) = j_1$ , если из точки  $i$  выходит один граничный отрезок  $(i, j_1)$ ;

в)  $IBOUND(1, i) = j_1$ ,  $IBOUND(2, i) = j_2$  или  $IBOUND(1, i) = j_2$ ,  $IBOUND(2, i) = j_1$ , если из точки  $i$  выходят два граничных отрезка  $(i, j_1)$  и  $(i, j_2)$ .

Например, для области, изображенной ниже, массив может быть задан так:



$IBOUND(1, 1) = 0$	$IBOUND(2, 5) = 0$
$IBOUND(2, 1) = 0$	$IBOUND(1, 6) = 7$
$IBOUND(1, 2) = 4$	$IBOUND(2, 6) = 0$
$IBOUND(2, 2) = 0$	$IBOUND(1, 7) = 0$
$IBOUND(1, 3) = 0$	$IBOUND(2, 7) = 5$
$IBOUND(2, 3) = 2$	$IBOUND(1, 8) = 9$
$IBOUND(1, 4) = 0$	$IBOUND(2, 8) = 0$
$IBOUND(2, 4) = 8$	$IBOUND(1, 9) = 6$
$IBOUND(1, 5) = 3$	$IBOUND(2, 9) = 0$

Программа TRINGL(X, Y, N0, IBOUND, IDOM, NODES, NET, NT1) позволяет разбить область на треугольники. Ее параметры являются:

X, Y — массивы абсцисс и ординат узловых точек;

N0 — число узловых точек;

IBOUND — массив описателей узловых точек (размером (2, N0));

IDOM — признак вычерчивания треугольников: IDOM = 1 — вычерчиваются, IDOM = 0 — не вычерчиваются;

NODES — рабочий массив длины N0 (для списка доступных вершин);

NET — массив описателей отщепленных треугольников (длины  $6 * N0$ );

NT1 — число отщепленных треугольников.

Следует заметить, что в результате работы программы изменяется содержимое массива IBOUND. Кроме того, если при работе программы TRINGL отщепление треугольников происходит таким образом, что в некоторый момент из какой-либо граничной вершины выходят три или более отрезков, то выдается соответствующее диагностическое сообщение. Это означает, что либо граница области с самого начала была сложной, либо узловые вершины распределены по области «плохо». Результатом работы программы TRINGL является массив описателей треугольников, которые могут быть преобразованы в информацию о сетке с помощью программы TRGRID.

Программа TRGRID(X, Y, N0, NET, IBOUND, NODES, NETO, NT1) позволяет преобразовать список описателей отщепленных треугольников в информацию о сетке. Ее параметры:

X, Y — массивы абсцисс и ординат узловых точек;

N0 — число узловых точек;

NET — массив описателей отщепленных треугольников (длины  $6 \cdot N0$ );

IBOUND — рабочий массив (размером  $(2, N0)$ );

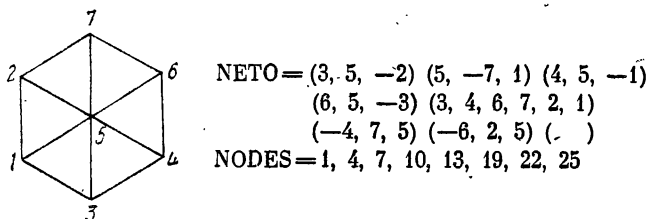
NODES — массив указателей (длины  $N0+1$ );

NETO — массив описателей сетки (длины  $6 \cdot N0$ );

NT1 — число отщепленных треугольников.

З а м е ч а н и е. Для выпуклых областей массив NETO достаточно задавать длиной  $6 \cdot N0 - 2 \cdot K$ , где K — число граничных точек.

Теперь опишем способ задания информации о сетке. Для каждой точки  $i$  области заводится список ее соседей в том порядке, в каком они встречаются при обходе против часовой стрелки, причем, если граничный отрезок  $[j, i]$  входит в точку  $i$ , то число  $j$  заносится в список со знаком минус. Списки для каждой вершины объединяются в один общий список, хранящийся в массиве NETO. А в некоторый массив NODES заносятся указатели на начало подсписка для каждой из вершин (индексы в массиве NETO). Например, для области, изображенной ниже, информация о сетке записывается следующим образом:



В тех случаях, когда в процессе работы сетка деформируется (меняются координаты узловых точек), возникает необходимость реорганизации сетки. Такую реорганизацию позволяет выполнить программа TRIG.

Программа TRIG(X, Y, N0, NODES, NETO, IBOUND, NET, NT1) позволяет разбить область на треугольники с учетом ее предыдущего состояния. Отличие этой программы от программы TRINGL в том, что в список доступных вершин для каждого граничного отрезка заносится его соседи по старой сетке до второго ранга. Параметры у программы следующие:

X, Y — абсциссы и ординаты узловых точек (массивы длины N0);

N0 — число узловых точек;

NODES — массив указателей (длины  $N0+1$ );  
 NETO — массив описателей старой сетки (длины  $6*N0$ );  
 IBOUND — рабочий массив (размером  $(2, N0)$ );  
 NET — массив описателей отщепленных треугольников (длины  $6*N0$ );  
 NT1 — число отщепленных треугольников.

**З а м е ч а н и е.** Для выпуклых областей массив NETO достаточно задавать длиной  $6*N0-2*K$ , где  $K$  — число граничных узловых точек.

**8.3.2. Построение центральных проекций.** Рассмотрим алгоритм построения центральной проекции поверхности, заданной функцией  $z = f(x, y)$ , область определения которой в плоскости  $XU$  ограничена непересекающимися ломаными линиями. В области определения строится треугольная сетка, в узлах которой заданы значения функции  $f(x, y)$ . Эта функция приближается функцией  $z = \tilde{f}(x, y)$  таким образом, что над каждым треугольником функция  $f(x, y)$  заменяется линейной по трем заданным значениям. Функция  $z = \tilde{f}(x, y)$  представляет собой поверхность, сложенную из треугольных элементов, их проекции и будут рассматриваться. В узловых точках значения функции  $f(x, y)$  и  $\tilde{f}(x, y)$  будут совпадать.

На взаимное расположение центра проекции, поверхности и картинной плоскости накладываются следующие ограничения:

а) точка  $(XV, YV)$  — проекция точки зрения на плоскость  $XU$  — должна лежать вне области определения функции;

б) проецируемая поверхность должна целиком находиться «перед лицом» наблюдателя, т. е. она должна полностью лежать по одну сторону от плоскости, параллельной картинной плоскости и проходящей через центр проекции.

Применяемый алгоритм удаления невидимых линий основывается на методе экранирования. Смысл этого метода заключается в том, что отмечается часть плоскости, занятая изображением, и при проведении новой линии рисуется только тот ее отрезок, который не попадает в эту уже занятую часть плоскости.

Следовательно, треугольные элементы поверхности должны быть упорядочены так, чтобы при рисовании последующие не закрывали предыдущих. Упорядочение элементов поверхности согласуется с упорядочением соответствующих треугольников в области определения функции и поэтому сначала изображается элемент, соответствующий тому треугольнику в области определения, который не закрывается никаким другим, затем — элемент, соответствующий треугольнику, который может закрываться лишь первым, и т. д. В условиях принятых ограничений на взаимное расположение центра проекции, картинной плоскости и поверхности такое упорядочение всегда существует  $[1, к]$ .



Алгоритм упорядочения включает в себя алгоритм определения видимости отдельного треугольника. Будем называть сторону треугольника *условно видимой*, если угол между вектором внешней нормали и проекцией луча зрения больше  $90^\circ$ . В противном случае сторона называется *условно невидимой*. Треугольник  $ABC$  (рис. 8.18) с основанием  $AB$ , где  $AB$  — граничный отрезок, считается видимым и выбирается, если сторона  $AB$  условно видима, а две другие стороны либо условно невидимы, либо условно видимая сторона является граничной.

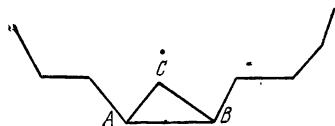


Рис. 8.18. К построению упорядоченной последовательности треугольников.

При этом никакая из условно видимых граничных сторон не должна закрываться другими условно видимыми отрезками границы.

Пусть выбран видимый треугольник  $ABC$ . Выбор очередного видимого треугольника производится следующим образом. Если сторона  $AC$  была условно невидимой, то в первую очередь исследуется треугольник с основанием  $AC$ . Если он не подходит, то проверяем треугольники, лежащие от него слева и справа вдоль границы. Если и эти треугольники не подходят, еще раз сдвигаемся влево и вправо и так до тех пор, пока не найдем подходящий треугольник, либо не обойдем весь контур, которому принадлежит отрезок  $AC$ . В этом случае выбираем ближайший к точке  $(XV, YV)$  граничный отрезок и выполняем для него аналогичную операцию. Если же с самого начала сторона  $AC$  была условно видимой (при выборе предыдущего треугольника), то этот процесс начинаем с отрезка  $CB$ .

После того как сделан выбор треугольника в области определения, элемент поверхности, соответствующий этому треугольнику, изображается с учетом текущего экрана. Затем экран изменяется и осуществляется переход к выбору очередного треугольника. И так до тех пор, пока не будут исчерпаны все треугольники.

**8.3.3. Экранирование.** По мере рисования элементов поверхности требуется запоминать ту часть картинной плоскости, которая уже занята изображением. Для проведения необходимых вычислений нужно, прежде всего, выбрать на картинной плоскости некоторую систему координат  $x_1y_1$  (рис. 8.19). Направим ось  $x_1$  вправо (если смотреть от наблюдателя, стоящего вертикально в центре проекции) параллельно линии пересечения картинной плоскости с плоскостью  $xy$ . Ось  $y_1$  выберем перпендикулярной оси  $x_1$  и направленной в сторону возрастания координаты  $z$ .

Для проведения экранирования в картинной плоскости определяется прямоугольник со сторонами, параллельными осям координат, внутри которого будет заключено изображение. Часть пря-

моугольника, которая уже занята изображением и в которой рисовать запрещается, запоминается следующим образом. Прямоугольник мысленно разбивается на равные части рядом вертикальных линий, назовем их *экранными линиями* (рис. 8.20), т. е. вводится физическая сетка (ср. с пп. 8.1.2, 8.2.1). На каждой экранной линии выделяются отрезки, принадлежащие той фигуре, которую нужно запомнить. Набор таких отрезков и является экраном, характеризующим эту фигуру.

Изображение поверхности строится последовательно по отдельным треугольникам в том порядке, как описывалось выше.

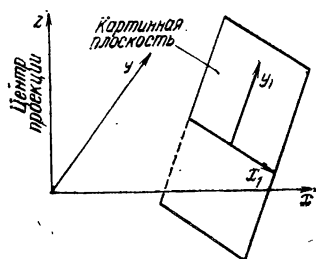


Рис. 8.19. Выбор системы координат на картинной плоскости.

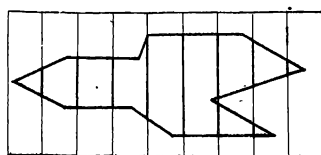


Рис. 8.20. Экранные линии. Экран.

При рисовании каждого треугольника вычерчиваются те его стороны, которые до этого еще не рисовались (каждое ребро принадлежит двум треугольникам, но вычерчивается оно всего один раз). Вычерчивание происходит при неизменном экране, затем экран модифицируется. В экран заносятся отрезки экранных линий, которые оказываются внутри спроецированного треугольника. При этом, если необходимо, происходит склеивание нескольких отрезков в один.

Для запоминания экрана служат два массива ISCR и SCR<sub>N</sub>. Массив SCR<sub>N</sub> трехмерный, граничными значениями его индексов являются следующие величины: количество экранных линий, характеризующее частоту физической сетки, число отрезков на одной экранной линии и число 2. Для оценки максимального числа отрезков, которое может встречаться на экранных линиях, поступаем следующим образом [1, к]. Спроецируем центр проекции на плоскость  $xu$  и подсчитаем, сколько раз луч, выходящий из построенной точки, может пересекать область определения, и к результату прибавим единицу. Число отрезков на каждой экранной линии не будет превышать полученной величины. Например, для области, показанной на рис. 8.21, число отрезков на экранной линии не превысит трех для центра проекции в точке 1 и двух — в точке 2.

Число 2 в описании массива SCRN обусловлено тем, что запоминаются начало и конец отрезка. Поэтому  $k$ -й отрезок на  $i$ -й экранной линии записывается в этом массиве так: его координата  $Y$  меняется в пределах  $SCRN(I, K, 1) \leq Y \leq SCRN(I, K, 2)$ . Другой массив, ISCR, одномерный, его длина определяется количеством экранных линий. В ISCR( $i$ ) запоминается число отрезков на  $i$ -й экранной линии. Перед началом рисования массив ISCR обнуляется.

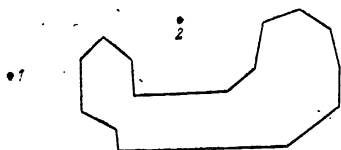
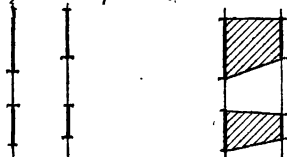


Рис. 8.21. К определению максимального числа отрезков на экранной линии.

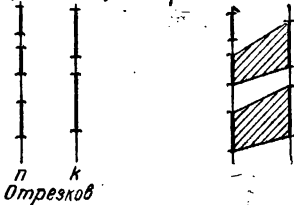
При изображении очередного треугольника приходится искать точки пересечения его ребер с границей фигуры, внутри которой рисовать запрещено. Поэтому надо уметь восстанавливать границу фигуры по ее экрану. Чтобы понять, как это делается, достаточно рассмотреть случай с двумя соседними экранными линиями.

а) Число отрезков на них одинаково



Тогда нижние отрезки соединяются между собой, верхние — между собой, остальные — аналогично.

б) Число отрезков различно



Построение границы идет как в а), только на той линии, где отрезков больше,  $n - k$  наименьших по длине отрезков в построении не участвуют.

Описать коротко сам процесс изображения отрезка с учетом экрана можно следующим образом. Будем говорить, что характеристика видимости точки на картинной плоскости равна 1, если точка лежит вне экрана, и 0 в противном случае. Для того чтобы нарисовать отрезок, нужно найти точки, где меняется характеристика видимости. Эти точки ищутся так. Экранными линиями весь отрезок разбивается на меньшие отрезки, для каждого из них определяются характеристики видимости его концов. Если они различны, то ищется точка пересечения отрезка с граничной линией экрана, в противном случае исследуется следующий малый отрезок и т. д.

### 8.3.4. Описание программ.

Программа TRSURF(X, Y, Z, N0, XV, YV, NODES, NETO, ISCR, SCRN, IDIM1, IDIM2, NET, IBOUND) управляет работой нескольких подпрограмм, с помощью которых находит последовательно треугольник за треугольником в области определения в том порядке, в каком соответствующие им элементы поверхности должны изображаться, проецирует эти элементы, рисует их с учетом текущего состояния экрана, изменяет экран. Если в процессе проведения вычислений число отрезков на одной экранной линии превысит заданную величину IDIM2, то выдается диагностическое сообщение об ошибке. Программа имеет следующие параметры:

X, Y, Z — массивы координат точек;  
N0 — количество точек;  
XV, YV — координаты центра проекции;  
NODES — массив указателей (длины N0+1);  
NETO — массив описателей сетки (длины 6\*N0);  
ISCR, SCRN — массивы описателей экрана (размером соответственно IDIM1 и (IDIM1, IDIM2, 2));  
IDIM1 — количество экранных линий;  
IDIM2 — максимальное число отрезков на экранной линии;  
NET — рабочий массив (длины 6\*N0);  
IBOUND — массив описателей условно видимых отрезков размером (2, N0).

Обращению к программе TRSURF должно предшествовать обращение к программе PREP, выполняющей ряд подготовительных операций.

Программа PREP(X, Y, Z, N0, XV, YV, ZV, XPL, YPL, ZPL, VX, VY, VZ, ISCR, IDIM1, RLYTOX) формирует матрицу однородных координат, с помощью которой будет выполняться проецирование, и устанавливает пределы изменения функции  $f(x, y)$  на картинной плоскости. Ее параметры:

X, Y, Z — массивы координат точек;  
N0 — количество точек;  
XV, YV, ZV — координаты центра проекции;  
XPL, YPL, ZPL — координаты точки на картинной плоскости;  
VX, VY, VZ — координаты вектора, перпендикулярного картинной плоскости;  
ISCR — рабочий массив характеристик экранных линий (длины IDIM1);  
IDIM1 — количество экранных линий;  
RLYTOX — коэффициент, определяющий форму математической области значений функции на картинной плоскости; он равен отношению (XMAX-XMIN)/(YMAX-YMIN), где XMIN, XMAX, YMIN, YMAX — пределы изменения функции, устанавливаемые в области рисования.

**Замечание.** Проекция точки зрения ( $XV, YV, 0$ ) на плоскость  $XU$  должна находиться вне области определения. Вектор ( $VX, VY, VZ$ ) должен быть направлен в полупространство, где находится центр проекции. В противном случае будет получено зеркальное отражение поверхности.

**8.3.5. Примеры.** Следующие рисунки получены с помощью программ, описанных в этом параграфе.

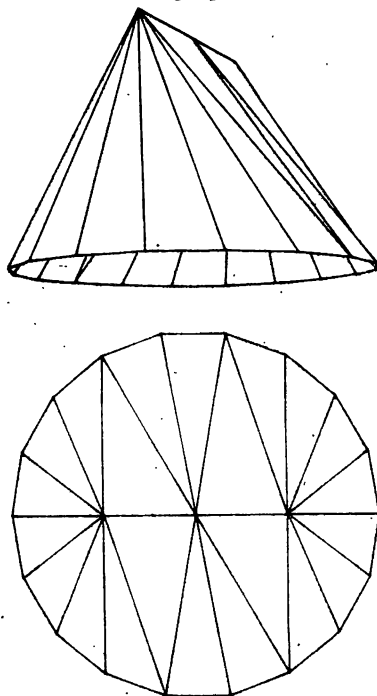


Рис. 8.22. Изображение поверхности Каталана.

1. На рис. 8.22 показана простая поверхность (поверхность Каталана), которая формируется и изображается с помощью следующей программы:

```

PARAMETER(R=2., IN=21, K=18, ID1=200, ID2=2)
DIMENSION X(N), Y(N), Z(N), NETO(6*N+2*K),
' NET(6*N)
DIMENSION SCR(N(ID1, ID2, 2), ISCR(ID1),
' NODES(N+1), IB(2, N)
DATA XV, YV, ZV/8., 5., -1./, VX, VY, VZ/8., 5., -1./
DATA XPL, YPL, ZPL, IDOM/3*0., 1/
DTH=2.*3.14159/K
    
```

### С ГРАНИЦА ОБЛАСТИ

```
DO 1 I=1, K
  IB(1, I)=1+1
  IB(2, I)=0
  X(I)=R*COS(DTH*(I-1))
  Y(I)=R*SIN(DTH*(I-1))
1 Z(I)=0.
  IB(1, K)=1
```

### С ВНУТРЕННИЕ ТОЧКИ

```
DO 2 I=1, N-K
  X(I+K)=I-2
  Y(I+K)=0.
  Z(I+K)=3.
  IB(1,I+K)=0
2 IB(2, I+K)=0
  CALL PAGE(17., 26., 0, 0, 0)
  CALL LIMITS(-2.1, 2.1, -2.1, 2.1)
  CALL REGION(3., 1., 11., 11., 0, 0, 0)
  CALL TRINGL(X, Y, N, IB, IDOM, NODES, NET, NT1)
  CALL TRGRID(X, Y, N, NET, IB, MODES, NETO, NT1)
  CALL REGION(3., 13., 11., 8., 0, 0, 0)
  CALL PREP(X, Y, Z, N, XV, YV, ZV, XPL, YPL, ZPL,
    ' VX, VY, VZ, ISCR, ID1, RLTOX)
  CALL TRSURF(X, Y, Z, N, XV, YV, NODES, NETO,
    ' ISCR, SCRN, ID1, ID2, NET, IB)
  CALL ENDPG('8.22')
END
```

Программа написана с использованием расширенного фортрана, транслировалась на машине БЭСМ-6 с помощью компилятора Форекс.

2. На рис. 8.23 показаны более сложные поверхности:

а) область определения функции не является прямоугольной (как и в предыдущем примере);

б) неодносвязная область задания функции.

#### 8.3.6. Служебные программы.

Программа TRG(L, IR, K, KL, KR, NET, NODES, IBOUND, NB, X, Y, XV, YV, IND) позволяет определить, закрывается ли чем-нибудь треугольник, в основании которого лежит граничный отрезок [L, IR]. Параметр IND = 1, если треугольник полностью видим, в противном случае IND = 0.

Программа NEXTRG(BOUND, X, Y, NET, NODES, NB, XV, YV) позволяет найти следующий треугольник и изменить сетку после его отщепления.

Программа PROJCT(XP, YP, ZP, XPJ, YPJ) позволяет спроецировать точку (XP, YP, ZP). (XPJ, YPJ) — координаты проекции.

Программа MATEVL(XV, YV, ZV, XPL, YPL, ZPL, VX, VY, VZ) вычисляет матрицу, с помощью которой осуществляется проектирование (эта матрица несколько модифицируется в программе PREP).

Программа SEE(L, IR, IBOUND, X, Y, NB, IND, IDRWN, XV, YV) определяет, закрывается ли отрезок [L, IR] одним из

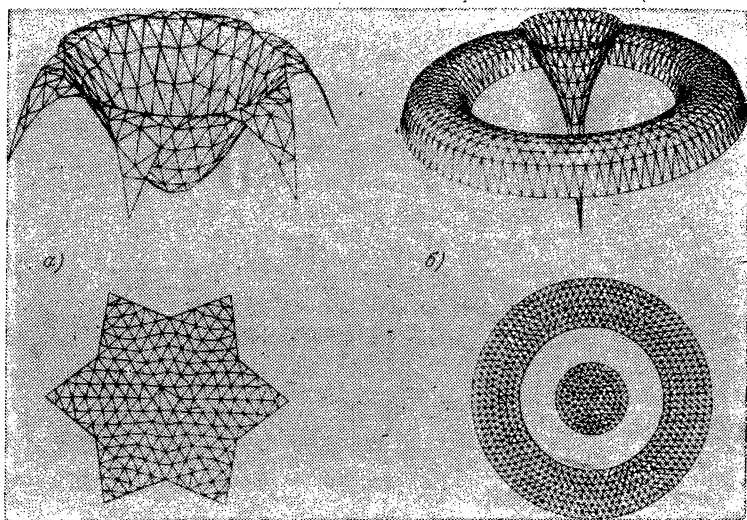


Рис. 8.23. Примеры изображений более сложных поверхностей.

условно видимых граничных отрезков, информация о которых хранится в массиве IBOUND. Кроме того, определяется, рисовался ли этот отрезок. Параметр  $IND = 1$ , если отрезок [L, IR] видим полностью, и  $IND = 0$ , если закрывается хотя бы частично. Параметр  $IDRWN = 1$ , если отрезок [L, IR] нарисован, в противном случае  $IDRWN = 0$ .

Программа SEE1(I1, I2, I3, I4, X, Y, XV, YV, IND) определяет, закрывается ли отрезок [I1, I2] отрезком [I3, I4], если смотреть из точки (XV, YV). Параметр  $IND = 1$ , если [I1, I2] полностью видим, и  $IND = 0$ , если [I1, I2] закрывается хотя бы частично.

Программа ILIMTN(L, IR, IBOUND, NB) позволяет исключить отрезок [L, IR] из списка условно видимых отрезков границы.

Программа LCROSS(X1, Y1, X2, Y2, X3, Y3, X4, Y4, C, D) вычисляет координаты точки пересечения двух отрезков [A1, A2] и [A3, A4] и заносит их в параметры C и D.

Программа SEGANG(X1, Y1, X2, Y2, A, STEP, IND) позволяет вычислить для отрезка [A1, A2] его угловой коэффициент A. Если отрезок почти вертикальный, то в параметр A заносится координата его пересечения с экранной линией (по выбранному критерию вертикальности такая линия будет единственной). В этом случае  $IND = 1$ .

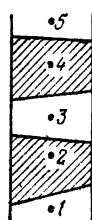
Программа SCREN1(ISCR, SCRN, IDIM1, IDIM2, Y, N, IND) позволяет определить, закрыта ли экраном точка с координатой Y, находящаяся на N-й экранной линии. Параметр  $IND = 1$ , если точка видима, в противном случае  $IND = 0$ .

Программа SCREN2(ISCR, SCRN, IDIM1, IDIM2, X, Y, NL, STEP, XBEG, A1, B1, A2, B2, IND) определяет положение точки (X, Y) относительно экрана.

A1 — угловой коэффициент гранитного отрезка, отделяющего точку (X, Y) снизу. (XBEG, B1) — точка этого отрезка;

A2 — аналогично для верхнего отрезка. (XBEG, B2) — точка верхнего отрезка.

Для точки 1	$IND = 1$
для точек 2, 4	$IND = 2$
для точки 3	$IND = -2$
для точки 5	$IND = 3$



Программа TRDRAW(ISCR, SCRN, IDIM1, IDIM2, X, Y, Z) позволяет нарисовать элемент поверхности, соответствующий треугольнику с вершинами в точках L, IR, K, и модифицировать экран.

Программа PICTUR(ISCR, SCRN, IDIM1, IDIM2, X1, Y1, X2, Y2) позволяет начертить отрезок [A1, A2], где  $A1 = (X1, Y1)$ ,  $A2 = (X2, Y2)$ .

Программа SCRMOD(ISCR, SCRN, IDIM1, IDIM2, XL, YL, XIR, YIR, XK, YK) изменяет текущее состояние экрана, добавляя к нему треугольник с координатами вершин (XL, YL), (XIR, YIR), (XK, YK).

Программа PNORDR(X, Y, NET, L, IB, IE) упорядочивает соседей L-й точки на участке от IB до IE так, чтобы при перечислении их в NET обход осуществлялся против часовой стрелки.

Функция COSIN1(X1, Y1, X2, Y2) вычисляет скалярное произведение двух векторов.

Функция COSIN2(X1, Y1, X2, Y2) вычисляет  $SIGN(\cos \varphi) * \cos^2 \varphi$ , где  $\varphi$  — угол между векторами (X1, Y1) и (X2, Y2).



## 8.4. Изображение трехмерных объектов с использованием эффекта ореола

В этом параграфе рассматривается еще один алгоритм удаления невидимых линий, позволяющий распространить на векторные устройства хорошо известный в растровой графике метод приоритетов (см. [1, л]). Этот алгоритм позволяет получать центральные и параллельные проекции произвольных графических объектов, представленных в виде неупорядоченной совокупности отрезков прямых. Для создания иллюзии глубины используется метод частичного удаления невидимых линий, основанный на *эффекте ореола*.

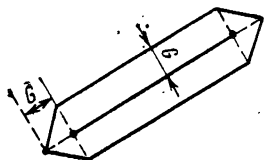


Рис. 8.24. Ореол вокруг проекции отрезка ( $G$  — ширина разрыва).

### 8.4.1. Особенности метода ореола.

Основная идея метода ореола заключается в следующем. Если в трехмерном пространстве некоторый отрезок находится впереди другого отрезка прямой, то при проецировании на картинную плоскость более удаленный отрезок изображается с разрывом. Можно считать,

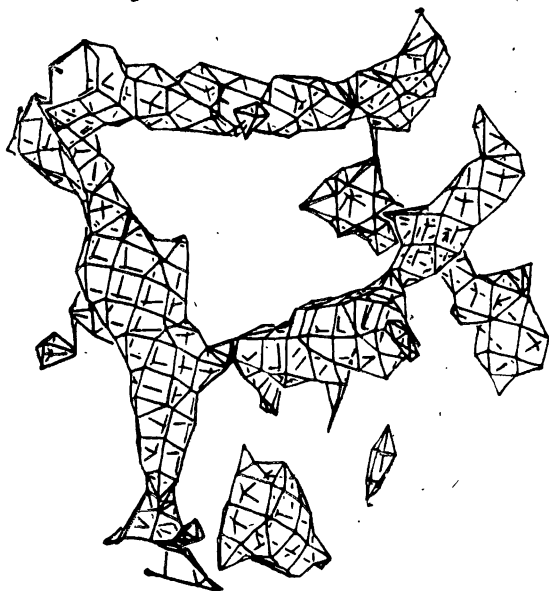


Рис. 8.25. Изображение поверхностей уровня плотности пространственного распределения галактик, построенное с использованием метода ореола.

что разрыв получается в результате создания вокруг близлежащего отрезка ореола — некоторой воображаемой непрозрачной области определенной конфигурации (например, имеющей форму шестиугольника, см. рис. 8.24).

Главное отличие метода ореола от других способов удаления невидимых линий состоит в том, что этот метод позволяет изображать неструктурированную графическую информацию. Входные данные для алгоритма окружения ореолом достаточно представить в виде неупорядоченной совокупности отрезков прямых, описывающих изображаемый объект, т. е. практически в той же форме, что и для каркасного изображения. На рис. 8.25 и 8.26 приведены примеры изображений, построенных с использованием метода ореола.

Как видно из рис. 8.25, метод ореола, вообще говоря, не позволяет полностью стереть невидимые линии. Однако когда изображаемые отрезки прямых или кривые находятся на более близком расстоянии друг от друга, чем заданная *ширина разрыва*  $G$  (рис. 8.24), то в этом случае достигается эффект, близкий к эффекту полного удаления невидимых линий (рис. 8.26). От ширины разрыва, таким образом, в значительной степени зависит «чистота» удаления невидимых линий. Влияние ширины разрыва особенно хорошо проявляется в тех случаях, когда изображаемый объект представляет собой криволинейную сетку, размер ячеек которой меньше значения этого параметра. В некоторых случаях, однако, изображения, получаемые с помощью метода ореола, оказываются более наглядными, чем при полном удалении невидимых линий. Кроме того, имеется возможность получить некое представление и о невидимой части объекта.

При реализации метода ореола в рассмотрение также вводится параметр *допуска по глубине*  $TOL$ . Необходимость его введения диктуется следующими соображениями. В довольно типичном случае, когда в пространстве пересекаются два отрезка (например, при задании сеток), между ними из-за приближенности вычислений может образоваться маленький зазор. В результате отрезки будут обрабатываться не как пересекающиеся, а как скрещивающиеся, что может привести к серьезным искажениям в изображе-

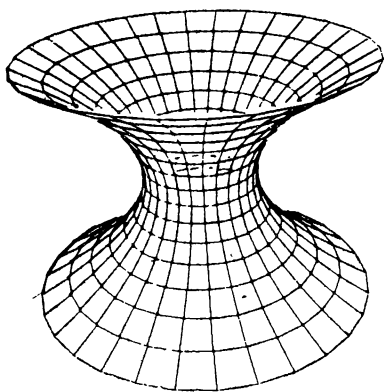
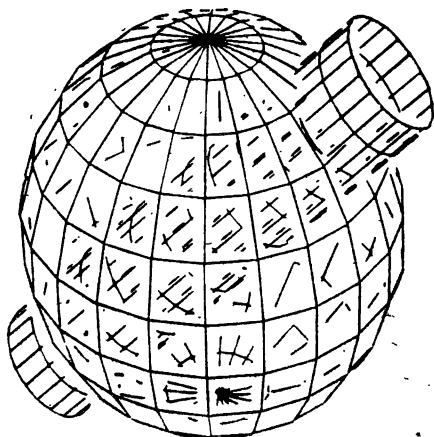


Рис. 8.26. Пример проекции алгебраической поверхности.

нии. Если же считать, что при пересечении двух отрезков один из них, скажем, первый, закрывает второй отрезок только в тех случаях, когда первый отрезок находится впереди второго на расстоянии не меньшем, чем некоторое заданное расстояние  $TOL$ , то

а)  $TOL=0.01$



б)  $TOL=0.3$

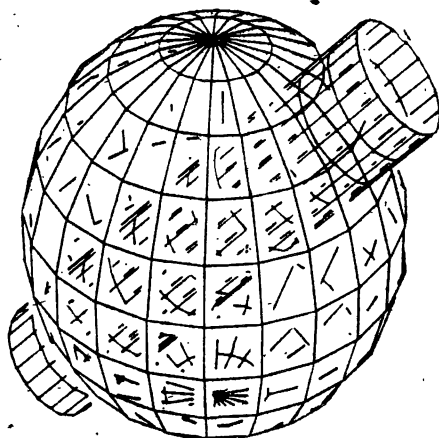


Рис. 8.27. Рисунки, показывающие влияние параметров допуска по глубине  $TOL$  на качество изображения (в результате увеличения значения  $TOL$  становятся видимыми некоторые отрезки или части отрезков, ранее скрытые от наблюдателя).

ошибок такого рода можно будет избежать. Кроме того, вводя допуск по глубине, мы получаем в свое распоряжение параметр,

с помощью которого в некоторых пределах можно управлять прозрачностью изображаемого объекта (рис. 8.27).

**8.4.2. Описание метода ореола.** Как мы уже упоминали, входная информация для алгоритма окружения ореолом должна быть представлена в виде совокупности отрезков прямых. Эти отрезки следует задавать координатами их крайних точек  $DX(K, L)$ ,  $DY(K, L)$ ,  $DZ(K, L)$ , где  $L$  — номер отрезка,  $K$  — индексы массива:  $K = 1$  для одной крайней точки и  $K = 2$  для другой.

Данные для алгоритма окружения ореолом удобно представлять в системе координат картинной плоскости. Для построения этой координатной системы можно воспользоваться практически любой схемой трехмерного проецирования. Допустимы и центральные проекции на картинную плоскость, перпендикулярную лучу зрения, проведенному из центра проекции в начало координат, и аксонометрические проекции на плоскость, перпендикулярную направлению проецирования. Без ограничения общности можем считать, что картинная плоскость всегда проходит через начало координат.

Систему координат картинной плоскости  $XYZ$  будем строить следующим образом. Пусть задана некоторая правая система координат  $X'Y'Z'$ . Будем называть ее объектной системой координат.

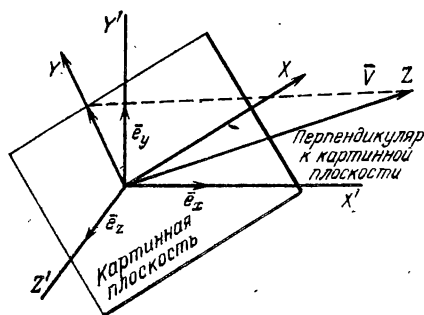


Рис. 8.28. Схема построения системы координат картинной плоскости для случая центральной проекции (здесь  $\bar{V}$  — центр проекции, а  $\bar{e}_x$ ,  $\bar{e}_y$ ,  $\bar{e}_z$  — орты объектной системы координат).

Направление оси  $Y$  системы координат картинной плоскости выберем совпадающим с направлением проекции на картинную плоскость орта оси  $Y'$  объектной системы координат (рис. 8.28). Направляющий вектор оси  $Z$  в случае центральной проекции определим как вектор, начало которого лежит в точке с координатами  $(0, 0, 0)$ , а конец — в центре проекции. В случае аксонометрической проекции будем считать, что ось  $Z$  направлена про-

типоволюжно вектору проецирования, И, наконец, орт оси  $X$  дополнит два предыдущих вектора до правой тройки.

Работу алгоритма окружения ореолом можно упрощенно представить следующим образом. Для каждого из отрезков, составляющих изображаемый объект (пусть его номер  $I$ ), отбираются все отрезки (с номерами  $J \neq I$ ), ореолы вокруг которых влияют на видимость  $I$ -го отрезка. Это означает, что проекции на картинную плоскость  $I$ -го и  $J$ -го отрезков пересекаются и  $J$ -й отрезок находится впереди  $I$ -го (относительно наблюдателя) на расстоянии большем, чем значение параметра  $TOL$ . Затем проводится отсечение проекции  $I$ -го отрезка по шестиугольным областям ореолов, окружающих отрезки с номером  $J$ . Части проекции  $I$ -го отрезка, которые попадают внутрь этих ореолов, считаются *невидимыми*, а остальные участки — *видимыми* (рис. 8.29). На проекции каждого отрезка допускается не более 15 отдельных интервалов невидимости. Всякий последующий (сверх 15) невидимый участок проекции отрезка будет склеиваться с ближайшим к нему интервалом невидимости и, следовательно, соответствующие видимые участки рисоваться не будут.

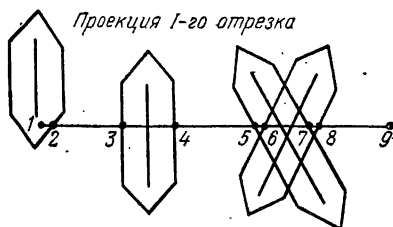


Рис. 8.29. Видимые и невидимые части проекции  $I$ -го отрезка. (Части 1—2, 3—4, 5—8 невидимы, остальные видимы.)

**8.4.3. Повышение эффективности алгоритма окружения ореолом.** Поскольку время работы алгоритма окружения ореолом в общем случае пропорционально квадрату числа отрезков, составляющих графический объект, важное значение имеют приемы, позволяющие уменьшить это время.

Разработано несколько различных способов сокращения времени вычислений, причем пользователю предоставлена возможность выбрать тот или иной из них.

Если все множество отрезков можно представить в виде двух или более независимых подмножеств, то упорядочение этих подмножеств можно использовать для сокращения времени, затрачиваемого на вычисления. Пусть, например, задана сцена, состоящая из нескольких изолированных объектов (рис. 8.30). Тогда для некоторого фиксированного центра или направления проецирования первый объект можно будет изобразить независимо от второго и третьего объектов, а второй — независимо от третьего. То есть отрезки, образующие первый объект, могут быть изображены после сравнений с отрезками, составляющими один только первый объект, отрезки, образующие второй объект, следует сравнивать с отрезками, составляющими первый и второй объекты, и только

отрезки третьего объекта необходимо сравнивать со всеми отрезками сцены. В результате можно добиться довольно существенного сокращения времени, затрачиваемого на вычисления (см. пример 2 в п. 8.4.5).

Существуют и другие возможные способы сокращения времени вычислений. Так, если отрезки, составляющие графический объект, удастся упорядочить таким образом, что проекция каждого последующего отрезка будет закрываться только ореолами проекций предыдущих отрезков, то проекцию каждого отрезка можно проверять на пересечение только с ореолами проекций тех отрезков, порядковые номера которых меньше, чем у рассматриваемого отрезка. (Подобное упорядочение применяется во многих алгоритмах удаления невидимых линий, предназначенных для изображения объектов, описываемых однозначными и непрерывными функциями двух переменных). В результате процессорное время, затрачиваемое на вычисления, уменьшается приблизительно вдвое.

С целью сокращения времени вычислений реализовано также автоматическое разбиение всего множества отрезков на два или четыре независимых подмножества с помощью одной секущей плоскости или двух взаимно перпендикулярных секущих плоскостей. При этом все отрезки, пересекающиеся с плоскостями, выделяются в отдельное (зависимое) подмножество. Секущие плоскости проходят через орт оси  $Y'$  (соответственно  $X'$ ) объектной системы координат, восстановленной из точки с координатами  $(C_x, C_y, C_z)$ , и вектор направления проецирования, помещенный в точку  $(C_x, C_y, C_z)$ , в случае параллельной проекции или вектор, соединяющий точку  $(C_x, C_y, C_z)$  с центром проекции, в случае центральной проекции (рис. 8.31). Точка  $(C_x, C_y, C_z)$  представляет собой либо центр тяжести графического объекта, либо некоторую заданную пользователем точку пространства. Центр тяжести объекта вычисляется по формулам:

$$C_x = \frac{1}{2N} \sum_{I=1}^N \sum_{J=1}^2 DX(J, I); \quad C_y = \frac{1}{2N} \sum_{I=1}^N \sum_{J=1}^2 DY(J, I);$$

$$C_z = \frac{1}{2N} \sum_{I=1}^N \sum_{J=1}^2 DZ(J, I),$$

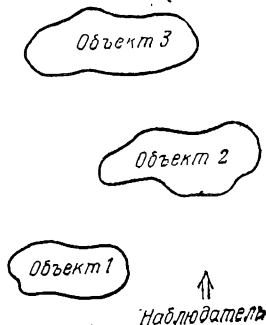


Рис. 8.30. Пример сцены, которая допускает разбиение всего множества отрезков на независимые подмножества.

где  $N$  — число отрезков, составляющих выбранный графический объект. При проведении проверок на видимость зависимое подмножество объединяется с каждым из независимых подмножеств.

В некоторых случаях может оказаться удобным добавить к графическому объекту дополнительные отрезки, которые не будут изображаться, но будут закрывать другие отрезки, т. е. будут принимать участие только в проверках на видимость. Программные

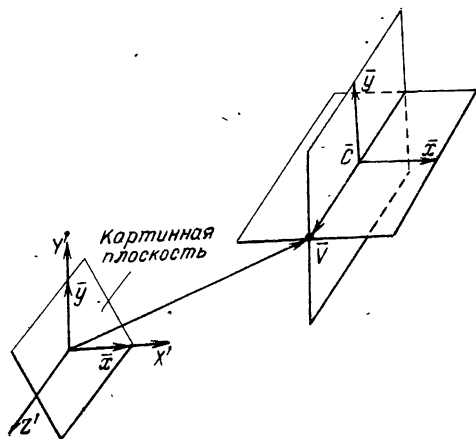


Рис. 8.31. Разбиение сцены на независимые подмножества (случай центрального проецирования). Обозначения:  $\vec{x}$ ,  $\vec{y}$  — орты соответственно осей  $X'$  и  $Y'$  объектной системы координат,  $\vec{C}$  ( $C_x$ ,  $C_y$ ,  $C_z$ ) — либо «центр тяжести» графического объекта, либо некоторая другая заданная точка,  $\vec{V}$  — центр проекции.

средства, реализующие алгоритм, дают возможность отдельно указывать отрезки, которые будут изображаться, и отрезки, которые будут принимать участие в проверках на видимость. Добавив к последнему подмножеству некоторое дополнительное количество «фиктивных» отрезков, можно добиться более полного эффекта удаления невидимых линий.

**8.4.4. Программная реализация алгоритма окружения ореолом.** При построении изображений с использованием метода ореола сохраняется обычный принятый в Графоре порядок работы: открытие страницы, заведение области и т. д. Сразу после определения области рисования необходимо задать режим проецирования, т. е. центр проекции или направление проецирования на картинную плоскость, перпендикулярную лучу зрения. Режим проецирования определяется с помощью обращения к программе PRJHL, которая формирует соответствующую матрицу преобразований размером  $4 \times 4$ .

Для задания пределов изменения координат  $X$  и  $Y$  на картинной плоскости служит программа VPLIM. Она выполняет те же функции, что и программы TDLIM и MNMX (см. пп. 8.1.2, 8.2.3), т. е. действует следующим образом. Сначала вычисляются максимальные и минимальные значения координат среди крайних точек проекций отрезков, составляющих подмножество, размер которого (т. е. число отрезков в нем) определяется самим пользователем. Далее выбирается одна из следующих возможностей: либо задаются пределы изменения, равные найденным, либо в качестве пределов изменения координат принимаются значения, общие среди найденных и пределов, полученных при предыдущих обращениях к этой программе. Используя приведенные возможности, можно в одной области рисования получать изображения сразу нескольких графических объектов с учетом их взаимного расположения в пространстве.

После того как установлен режим проецирования и определены пределы изменения координат, можно спроецировать графический объект или некоторую его часть на картинную плоскость. Для этой цели используется программа COORDT. В результате работы этой программы формируются массивы значений координат крайних точек отрезков, спроецированных на картинную плоскость. Отметим, что поскольку графические объекты, как правило, состоят из достаточно большого числа отрезков, координаты проекций крайних точек отрезков заносятся в исходные массивы  $DX$ ,  $DY$ ,  $DZ$ . Таким образом, после обращения к программе COORDT первоначальные массивы  $DX$ ,  $DY$ ,  $DZ$  со значениями координат отрезков будут изменены, поскольку на то же место будут записаны значения координат для проекций отрезков.

При обращении к программе COORDT необходимо указать способ изображения совокупности отрезков, описывающих графический объект. Здесь возможны восемь случаев. В первом из них, самом общем, проекция каждого отрезка проверяется на пересечение с ореолами проекций всех остальных отрезков, в то время как во втором случае — только с теми отрезками, порядковые номера которых меньше, чем у рассматриваемого отрезка (см. п. 8.4.3).

В остальных шести случаях выполняется автоматическое разбиение всего множества отрезков на два или четыре независимых подмножества и одно зависимое. Первые три случая отличаются от последних трех только выбором точки ( $C_x$ ,  $C_y$ ,  $C_z$ ) (см. п. 8.4.3): либо эта точка является центром тяжести объекта, либо она задается явно.

В случаях 3, 6 и 4, 7 разбиение выполняется одной секущей плоскостью, проходящей через орт оси  $Y$  (случаи 3, 6) или оси  $X$  (случаи 4, 7) объектной системы координат, в то время как в



случаях 5 и 8 разбиение производится сразу обеими этими плоскостями. Явное задание точки ( $C_x$ ,  $C_y$ ,  $C_z$ ) в случаях 6—8 производится с помощью программы SPLITP, обращение к которой должно предшествовать вызову программы COORDT.

В общем случае при построении изображений целесообразно проводить разбиение сразу обеими секущими плоскостями (относительно «центра тяжести» объекта либо относительно некоторой другой удобной точки пространства). При этом нам выгодна ситуация, когда зависимое подмножество содержит минимальное количество отрезков. Однако в случае вытянутых объектов или объектов, состоящих из «длинных» отрезков, в состав зависимого подмножества может входить чересчур много отрезков (в предельном случае оно будет содержать все отрезки объекта, в то время как независимые подмножества будут пусты). Поэтому иногда вариант с одной секущей плоскостью может оказаться предпочтительнее.

Две программы, HALLNE и HALOED, предназначены специально для построения графических объектов с удалением невидимых линий. Эти программы позволяют получить изображение всего графического объекта или некоторой его части восемью различными способами (см. описание программы COORDT). Различие между этими программами заключается в том, что при использовании программы HALLNE отпадает необходимость в предварительном вызове программы COORDT. То есть программу HALLNE можно рассматривать как объединение двух программ: COORDT и HALOED. Однако программа HALOED позволяет воспользоваться упорядочением независимых подмножеств отрезков, на которые разбивается все множество отрезков (если, конечно, такое разбиение возможно), или, например, ввести в рассмотрение отрезки, которые будут участвовать в проверках на видимость, но не будут изображаться (см. п. 8.4.3). Это достигается тем, что в описании параметров программы HALOED отдельно указывается совокупность отрезков, которые будут изображаться, и отдельно совокупность отрезков, которые будут участвовать в проверках на видимость.

В программах HALLNE и HALOED в качестве параметров задаются значения ширины разрыва  $G$  и допуска по глубине  $TOL$ . Как мы уже отмечали, конкретные значения этих параметров выбираются из эстетических соображений. При этом следует иметь в виду, что в случаях, когда ореолы соседних ребер элемента сетки, описывающей поверхность, закрывают весь или почти весь этот элемент, возможно пропадание отдельных отрезков или их частей на перегибах поверхности, ее краях и т. д. (см., например, рис. 8.27). Тем не менее использование достаточно больших (относительно элемента сетки) значений параметра  $G$  во многих слу-

чаях позволяет получить изображения даже более качественные (и с более полным удалением невидимых линий), чем при меньших значениях  $G$  (ср., например, рис. 8.27 и приведенный ниже рис. 8.32).

В качестве первого приближения рекомендуется выбирать параметр  $G$  таким образом, чтобы ореолы вокруг ребер наиболее типичных элементов поверхности закрывали эти элементы не полностью. Значение  $TOL$  вначале можно взять равным 0.01—0.1 от значения  $G$ . При таком выборе этих параметров возможные искажения на краях изображения будут сведены к минимуму, а просматриваемый задний план будет способствовать выбору наилучшего ракурса. После первой прикидки можно будет уже более уверенно подбирать значения параметров  $G$  и  $TOL$ , добиваясь желаемого качества изображения.

При работе с графическими объектами, заданными в виде последовательности отрезков прямых, в некоторых случаях может оказаться полезной возможность построить изображение всего графического объекта или некоторой его части без удаления невидимых линий. Для этой цели предназначена программа DRWHL, которая строит заданную проекцию объекта без использования метода ореола.

Программа PRJHL(IT, X, Y, Z) позволяет задать режим проецирования. Ее параметрами являются:

IT — признак проекции: IT = 0 — параллельная проекция, IT = 1 — центральная проекция;

X, Y, Z — координаты вектора направления проецирования, если выбрана параллельная проекция, или координаты центра проекции, если выбрана центральная проекция.

**З а м е ч а н и е.** При выборе положения центра проекции необходимо следить, чтобы он лежал за пределами параллелепипеда, описанного вокруг изображаемого графического объекта.

Программа VPLIM(DX, DY, DZ, NLINES, NL1, NL2, S) предназначена для задания пределов изменения координат проекций на картинную плоскость крайних точек отрезка. Установленные пределы сохраняются до очередного обращения к программе VPLIM. Параметрами этой программы являются:

DX, DY, DZ — массивы значений координат крайних точек отрезков (размером (2, NLINES));

NLINES — число отрезков в описании графического объекта;

NL1 — номер первого отрезка подмножества, для которого ищутся пределы изменения координат:

NL1 > 0 — в области рисования устанавливаются пределы, найденные для данного подмножества отрезков,

NL1 < 0 — в области рисования устанавливаются пределы, общие среди тех, которые были найдены для данного подмножества

отрезков, и тех, которые были получены при предыдущих обращениях к программе VPLIM;

NL2 — номер последнего отрезка подмножества, для которого ищутся пределы изменения координат,  $|NL1| \leq NL2$ ;

S — коэффициент, определяющий форму математической области значений координат на картинной плоскости; он равен отношению  $(XMAX - XMIN)/(YMAX - YMIN)$ , где XMAX, XMIN, YMAX, YMIN — пределы, установленные в области рисования (об использовании S см. п. 8.1.2).

Программа COORDT(DX, DY, DZ, NLINES, N1, N2, NCLUST, NLNDR, ITYP) позволяет спроецировать на картинную плоскость заданное количество отрезков. Параметры программы:

N1, N2 — номера первого и последнего отрезков, проецируемых на картинную плоскость;

NCLUST — рабочий массив, используемый при разбиении всего множества отрезков на независимые подмножества (длины NLNDR);

NLNDR — длина массива NCLUST:  $NLNDR \geq N2 - N1 + 1$ , если ITYP = 2, и  $NLNDR = 1$  в остальных случаях;

ITYP — параметр, указывающий способ изображения заданной совокупности отрезков, определяющей графический объект:

ITYP = 1 — (самый общий способ) каждый отрезок проверяется на видимость относительно всех остальных отрезков,

ITYP = 2 — (упорядоченные отрезки) каждый отрезок проверяется на видимость только по отношению к тем отрезкам, порядковые номера которых меньше, чем у рассматриваемого отрезка,

ITYP = 3, 4 — выполняется автоматическое разбиение всего множества отрезков на два независимых подмножества (разбиение выполняется плоскостью, проходящей через брт оси  $Y'$  (соответственно оси  $X'$ ) объектной системы координат, восстановленный из «центра тяжести» объекта),

ITYP = 5 — выполняется автоматическое разбиение всего множества отрезков на четыре независимых подмножества (разбиение выполняется двумя плоскостями, проходящими через орты осей  $X'$  и  $Y'$  объектной системы координат, восстановленными из «центра тяжести» объекта),

ITYP = 6, 7, 8 — то же, что и при ITYP = 3, 4, 5, только орты осей координат восстанавливаются не из «центра тяжести», а из точки, координаты которой задаются с помощью программы SPLITP (обращение к этой программе должно предшествовать вызову COORDT).

Значения остальных параметров те же, что и в программе VPLIM.

Замечание. В результате работы этой программы изменяются первоначальные значения массивов координат DX, DY, DZ,

Программа `SPLITP(X, Y, Z)` служит для задания координат точки объектного координатного пространства, относительно которой выполняется автоматическое разбиение всего множества отрезков на два или четыре независимых подмножества. Обращение к `SPLITP` имеет смысл только в тех случаях, когда значение параметра `ITYP` в обращении к программе `COORDT` равно 6, 7 или 8. Эта программа реализована как входная точка программы `COORDT`.

Программа `HALLNE(DX, DY, DZ, NLINES, N1, N2, NCLUST, NLNDR, G, TOL, ITP)` позволяет построить изображение всего графического объекта или некоторой его части с удалением невидимых линий. Ее параметрами являются:

`N1, N2` — номера первого и последнего отрезков, определяющих изображаемую часть графического объекта;

`G` — ширина разрыва;

`TOL` — допуск по глубине.

Остальные параметры те же, что и в программе `COORDT`.

Программа `HALOED(DX, DY, DZ, NLINES, NDRAW1, NDRAW2, NC1, NC2, NCLUST, NLNDR, G, TOL, ITP)` позволяет построить изображение заданной части графического объекта после проверок на видимость с указанной совокупностью отрезков. Параметры программы:

`DX, DY, DZ` — массивы значений проекций координат крайних точек отрезков (размером  $(2, NLINES)$ );

`NLINES` — количество отрезков в описании графического объекта;

`NDRAW1, NDRAW2` — номера первого и последнего в совокупности отрезков, которые будут изображаться;

`NC1, NC2` — номера первого и последнего в совокупности отрезков, которые будут участвовать в проверках на видимость.

Значения остальных параметров те же, что и в программах `COORDT` и `HALLNE`.

**З а м е ч а н и е.** Значения параметров `NC1, NC2` могут отличаться от значений параметров `NDRAW1, NDRAW2` лишь при `ITYP = 1` или `ITYP = 2`. При других значениях `ITYP`, `NC1, NC2` считаются совпадающими с `NDRAW1, NDRAW2`.

Программа `DRWHL(DX, DY, DZ, NLINES, NL1, NL2)` позволяет построить изображение всего графического объекта или некоторой его части без удаления невидимых линий. Значения параметров программы совпадают со значениями соответствующих параметров в программе `HALLNE`.

**Вспомогательные программы.** При работе с графическими объектами, представленными в виде последовательности отрезков прямых, может потребоваться повернуть объект (или его часть) на некоторый угол или перенести на некоторое расстояние.

яние. Эти задачи решаются с помощью обращений к программам ROTH и TRANH.

Кроме того, в некоторых случаях возникает необходимость представить сеточную функцию в виде последовательности отрезков прямых. Для этой цели служит программа ARRINL. Отметим, что при юго-западном положении центра проекции относительно области определения функции (рис. 8.10) формируется упорядоченная последовательность отрезков прямых, которую можно изобразить вторым способом, т. е. при значении параметра ITYP=2 (см. описание программ COORDT, HALOED, HALLNE). Порядок заполнения массивов DX, DY, DZ показан на рис. 8.10.

Программа ROTH(NAXES, FI, DX, DY, DZ, NLINES, NBEG, NEND) позволяет повернуть графический объект или некоторую его часть вокруг одной из главных координатных осей правой системы координат на заданный угол в направлении по часовой стрелке, если смотреть с конца единичного вектора этой оси на начало координат. Параметрами этой программы являются:

NAXES — номер оси, относительно которой выполняется поворот: NAXES=1 — ось X, NAXES=2 — ось Y, NAXES=3 — ось Z;

FI — угол поворота (в градусах);

DX, DY, DZ — массивы значений координат крайних точек отрезков (размером (2, NLINES));

NLINES — количество отрезков в описании графического объекта;

NBEG, NEND — номера первого и последнего в совокупности отрезков, которые подвергаются повороту.

Программа TRANH(DELX, DELY, DELZ, DX, DY, DZ, NLINES, NBEG, NEND) предназначена для переноса графического объекта или некоторой его части на заданные расстояния вдоль координатных осей. Ее параметры:

DELX, DELY, DELZ — координаты вектора переноса относительно осей X, Y и Z.

Остальные параметры те же, что и в программе ROTH.

Программа ARRINL(X, Y, A, IX, IY, IXB, IXE, IYB, IYE, DX, DY, DZ, NLINES, NLBEG) предназначена для преобразования сеточной однозначной функции в представление этой функции в виде последовательности отрезков прямых. Параметры программы:

X — массив точек сетки по координате X, расположенных в порядке возрастания;

Y — массив точек сетки по координате Y, расположенных в порядке возрастания;

A — двумерный массив значений функции в узлах сетки, определяемой массивами X и Y (размером (IX, IY));

IX, IY — размерность сетки по осям X и Y;

IXB, IXE, IYB, IYE — начальные и конечные номера столбцов и строк, определяющих на области задания функции прямоугольную подобласть, предназначенную для преобразования в последовательность отрезков;

DX, DY, DZ — массивы значений координат крайних точек отрезков (размером (2, NLINES));

NLINES — число отрезков, образующих сетку (NLINES=2\*  
\*IDX\*IDY-(IDX+IDY), где  $IDX=IXE-IXB+1$ ,  $IDY=IYE-IYB+1$ );

NLBEG — номер, начиная с которого в массивах DX, DY, DZ размещаются отрезки, составляющие рассматриваемую поверхность.

**З а м е ч а н и е.** Пользуясь параметром NLBEG, можно формировать сцены, состоящие из нескольких поверхностей, наложенных друг на друга (см. [1, л]).

**С л у ж е б н ы е п р о г р а м м ы.**

Программа HLDLN(DX, DY, DZ, NLINES, I, NC1, NC2, NCLUST, NLNDR, ITYP) позволяет построить изображение отрезка с номером I после проверок на видимость с отрезками из совокупности (NC1, NC2), если параметр ITYP=1 или ITYP=2, или с отрезками, принадлежащими подмножествам с номерами NC1, NC2, если ITYP=3÷8.

Программа CROSSP(X, Y, RC, T) предназначена для вычисления параметра T в диапазоне от 0 до 1, который соответствует точке с координатами RC(1), RC(2), лежащей на отрезке с крайними точками X(1), Y(1) и X(2), Y(2).

Программа EXTLN(X, Y, Z, G) позволяет продолжить отрезок с крайними точками X(1), Y(1), Z(1) и X(2), Y(2), Z(2) в обе стороны так, чтобы длина его проекции на картинную плоскость увеличилась в каждую сторону на расстояние, равное значению параметра G.

Программа STORIN(AL, AU) служит для запоминания невидимой части проекции I-го отрезка, определяемой параметрами AL и AU, в массиве-буфере TE(15, 2). При этом два или более пересекающихся невидимых интервала объединяются в один, отбрасываются интервалы, которые уже были учтены при предыдущих обращениях к этой программе, и т. д.

Программа SVLSEG(AL, AU, J) служит для занесения значений параметров AL и AU, определяющих невидимую часть отрезка, в J-й элемент буфера TE.

Программы SHDOWN(J) и SHUP(J) предназначены для перемещения на одну позицию соответственно вниз или вверх содержимого буфера TE, заключенного в диапазоне, определяемом параметрами J и NL. Значение NL выбирается из 31-го слова общего блока GFSV и представляет собой текущее число отдельных интервалов невидимости на проекции рассматриваемого отрезка.

Программа TESTVS предназначена для идентификации ситуаций, когда рассматриваемый отрезок оказывается полностью закрыт другими отрезками. В этом случае сбрасывается флаг INVSLN, который является 32-м словом общего блока GFSV.

Программа DRAWTE служит для изображения видимых частей рассматриваемого отрезка.

Программа DRAWEL(T, I) позволяет по заданному в интервале от 0 до 1 значению параметра прямой T вычислить координаты соответствующей точки. Если параметр I=1, то проводится отрезок из точки текущего положения пера в вычисленную точку; если же I=0, то рисование не производится.

Программа HCNIT(A) служит для формирования единичной матрицы размером  $4 \times 4$ .

Программа MXMULT(A, B) позволяет выполнить умножение двух матриц A и B размером  $4 \times 4$ . Результат умножения помещается в A.

Программа STROT3(NAXES, CS, SN, R) предназначена для формирования матрицы поворота R(4, 4). Поворот выполняется вокруг одной из главных координатных осей, номер которой определяется параметром NAXES правой системы координат. Косинус и синус угла поворота задаются параметрами CS и SN.

Программа HCNCOR(X, Y, Z, T). С помощью этой программы по заданным координатам точки в объектной системе координат определяются координаты этой точки после воздействия на нее преобразования, задаваемого матрицей T размером  $4 \times 4$ .

Программа SORTY(DX, DY, DZ, NL, N1, N2) служит для упорядочения крайних точек отрезков, заданных массивами DX(K, NL), DY(K, NL), DZ(K, NL) таким образом, что K=1 для точки с большим значением координаты DY, а для другой крайней точки K=2. Упорядочению подвергаются отрезки с номерами от N1 до N2.

**8.4.5. Примеры.** В этом разделе приводятся примеры, иллюстрирующие использование описанных выше программ проецирования с удалением невидимых линий.

1. На рис. 8.26 показана поверхность, построенная с использованием программы HALLNE. Данные для поверхности (центр которой находится в начале координат) формируются при обращении к программе GIPER1.

```
DIMENSION DX(2, 930), DY(2, 930), DZ(2, 930), NCLUST(930)
DATA N, G, TOL/930, 0.15, 0.4/
CALL GIPER1(2., -2., 2., 15, 30, DX, DY, DZ, N, 1)
CALL PAGE(14., 26., 0, 0, 0)
CALL REGION(2., 8., 10., 10., 0, 0, 0)
CALL PRJHL(1, 5, 5., -10.)
```

```
CALL VPLIM(DX, DY, DZ, N, 1, N, S)
CALL HALLNE(DX, DY, DZ, N, 1, N, NCLUST, N, G,
, TOL, 5)
CALL ENDPG('8.26')
END
```

Тот же результат получится при использовании фрагмента:

```
CALL PRJHL(1, 5., 5., -10.)
CALL VPLIM(DX, DY, DZ, N, 1, N, S)
CALL SPLTP(0., 0., 0.)
CALL HALLNE(DX, DY, DZ, N, 1, N, NCLUST, N, G,
, TOL, 8)
```

Во втором случае явно указана точка, относительно которой должно проводиться разбиение графического объекта на независимые подмножества (см. пп. 8.4.3 и 8.4.4).

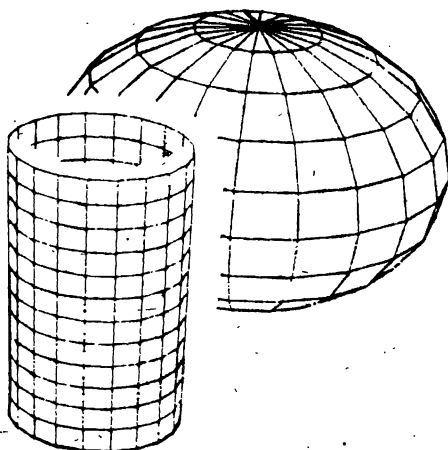


Рис. 8.32. Изображение сцены, допускающей представление в виде двух независимых подмножеств отрезков.

2. На рис. 8.32 показана сцена, состоящая из двух изолированных графических объектов: цилиндра и сферы. Цилиндр формируется с помощью обращения к подпрограмме CILIND, сфера — в результате вызова двух подпрограмм — SPHERG и SPHERV. Благодаря тому, что сцена допускает описание в виде независимых подмножеств отрезков, время, затрачиваемое на вычисления, удается заметно сократить (в данном случае на 60%).

```
DIMENSION DX(2, 660), DY(2, 660), DZ(2, 660),
, NCL(280)
DATA N, NC/660, 280/
```



С...ЦИЛИНДР (ОТРЕЗКИ 1—280):  
 CALL CILIND(0.4, -1., 0.9, 12, 20, DX, DY, DZ, N, 1)  
 CALL TRANH(1.7, 0., 0., DX, DY, DZ, N, 1, NC)

С...СФЕРА (ОТРЕЗКИ 281—660):  
 CALL SPHERG(1., 20, 10, DX, DY, DZ, N, NC+1)  
 CALL SPHERV(1., 20, 10, DX, DY, DZ, N, NC+1+180)

С...ИЗОБРАЖЕНИЕ СЦЕНЫ:  
 CALL PAGE(14., 26., 0, 0, 0)  
 CALL REGION(2., 8., 10., 10., 0, 0, 0)  
 CALL PRJHL(1, 10., 5., -3.)  
 CALL VPLIM(DX, DY, DZ, N, 1, N, S)

С...ИЗОБРАЖЕНИЕ ЦИЛИНДРА:  
 CALL COORDT(DX, DY, DZ, N, 1, NC, NCL, NC, 3)  
 CALL HALOED(DX, DY, DZ, N, 1, NC, 1, NC, NCL, NC,  
 , 0.1, 0.3, 3)

С...ИЗОБРАЖЕНИЕ СФЕРЫ:  
 CALL COORDT(DX, DY, DZ, N, NC+1, N, NCL, 1, 1)  
 CALL HALOED(DX, DY, DZ, N, NC+1, N, 1, N, NCL,  
 , 1, 0.15, 0.5, 1)  
 CALL ENDPG('8.32')  
 END

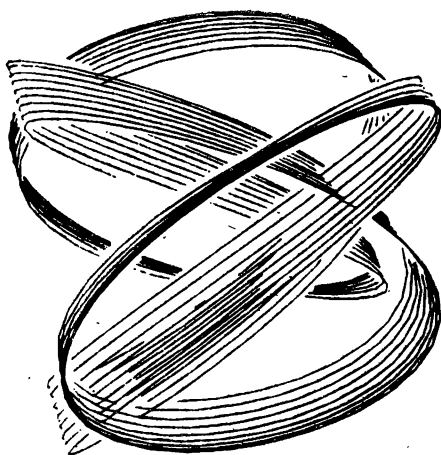


Рис. 8.33. Изображение пространственной кривой, построенное с использованием метода ореола.

3. На рис. 8.33 изображена центральная проекция пространственной кривой, построенная с использованием метода ореола.

```

DIMENSION DX(2, 880), DY(2, 880), DZ(2, 880),
, NCLUST(880)
DATA NN, N1, M, N, K/8, 110, 3, 4, 1/, G, TOL,
, ITYP/0.05, 0.2, 5/
N2=NN*N1
H1=3.5/NN
H2=4.5/NN
FI=H1
PSI=H2
DO 21 J=1, N2
FI=FI+H1/N1
PSI=PSI+H2/N1
TH=6.283*(J-1)/(N1-1)
X=SIN(TH)
Y=1.2*SIN(M*TH+FI)
Z=1.5*SIN(N*TH+PSI)
IF(K.EQ.1) GOTO 20
DX(2, K-1)=X
DY(2, K-1)=Y
DZ(2, K-1)=Z
IF(K.EQ.N2) GOTO 21
20 DX(1, K)=X
DY(1, K)=Y
DZ(1, K)=Z
K=K+1
21 CONTINUE

```

# С...РИСОВАНИЕ ПРОЕКЦИИ ПРОСТРАНСТВЕННОЙ КРИВОЙ:

```

CALL PAGE(15., 15., 0, 0, 1)
CALL REGION(1., 1., 13., 13., 0, 0, 0)
CALL PRJHL(1, 15., 8., 3.)
CALL VPLIM(DX, DY, DZ, N2, 1, N2, S)
CALL HALLNE(DX, DY, DZ, N2, 1, N2, NCLUST, N2, G,
, TOL, ITYP)
CALL ENDPG('8.33')
END

```

## ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О ГРАФОРЕ

Графтор представляет собой библиотеку программ, работающую в рамках операционной системы, имеющейся на той или иной ЭВМ. Чтобы работать с комплексом Графтор, необходимо представлять, где и в каком виде хранится библиотека программ и как ее можно подключить на стадии загрузки и объединения с главной программой.

Так, на БЭСМ-6 эта библиотека может быть либо личной, либо общей. Обращение к ней осуществляется обычными средствами мониторной системы Дубна. Из-за нестандартности подключения графических устройств к ЭВМ БЭСМ-6 библиотека драйверов не создавалась, и в каждом конкретном случае необходимо подменять программу PLOT.

На ЕС ЭВМ в рамках ОС ЕС Графтор представляет собой библиотеку загрузочных модулей с именем SYS1.GRAFLIB (или другим, заданным при подготовке библиотеки). В библиотеку включены драйверы вывода для стандартных устройств.

В настоящее время комплекс Графтор на ЕС ЭВМ обеспечивает работу следующих графических устройств: ЕС-7051, АП-7251, ЕС-7052, ЕС-7053, ЕС-7054 и ЕС-7064 (только вывод). Комплекс позволяет также подключить новые нестандартные устройства. В этом случае системному программисту необходимо будет самому написать соответствующие программы. Комплекс позволяет осуществлять вывод на графическое устройство как «напрямую» (ONLINE), так и через промежуточный носитель (перфоленту, магнитную ленту, диск).

Из-за различий в системе приказов графических устройств и канальных команд ввода/вывода не всегда удавалось найти приемлемый вариант программы нижнего уровня, который с одинаковым успехом мог бы использоваться для вывода изображения как на перфоленту, так и непосредственно на графплоттер и т. п. Вследствие этого нижний уровень раскололся еще на два: уровень генерации графических приказов (компилирующая программа) и уровень ввода/вывода (хэндлер устройства). Программа PLOT выполняет функции посредника, обращаясь к соответствующим входам компилирующей программы.

Для использования Графора в рамках ОС на машинах типа ЕС подготовлена дистрибутивная лента с двумя файлами GRAF.SYGRUS и GRAF.OBJ. В первом из них находятся тексты каталогизированных процедур и ряда программ нижнего уровня. Во втором — библиотека объектных модулей всех программ Графора. Разработана специальная система предварительной генерации комплекса Графор, результатом которой является библиотека загрузочных модулей. Сведения о генерации, об использовании программ и требования к операционной системе приведены ниже.

Вопросы организации Графора на других ЭВМ рассматриваться здесь не будут.

## А1. Генерация комплекса Графор

Генерация Графора имеет четыре стадии:

- 1) восстановление с дистрибутивной ленты дистрибутивных библиотек;
- 2) описание требуемой конфигурации (подготовка макрокоманд генерации второй стадии и получение потока заданий для третьей стадии);
- 3) формирование библиотеки загрузочных модулей (выполнение потока заданий, полученного на второй стадии);
- 4) выполнение тестовых примеров.

**А1.1. Первая стадия генерации.** Ниже приводится пример для восстановления дистрибутивных библиотек на диске с серийным номером PTOM04.

```
// RESTORE JOB
// EXEC PGM=IEHMOVE
//SYSUT1 DD VOL=REF=SYS1.SVCLIB,DISP=OLD
//A DD UNIT=2311,VOL=SER=PTOM04,DISP=OLD
//SYSPRINT DD SYSOUT=A
//TAPE DD UNIT=2400,VOL=SER=TAPE,LABEL=(, NL),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN DD *
COPY PDS=GRAF.OBJ,TO=2311=PTOM04,*
FROM=2400=(TAPE, 1),FROMDD=TAPE
COPY PDS=GRAF.SYGRUS,TO=.....
/*
// EXEC PGM=IEHPROGM
//A DD UNIT=2311,VOL=SER=PTOM04,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
CATLG DSNAME=GRAF.OBJ,VOL=2311=PTOM04
CATLG DSNAME=GRAF.SYGRUS,VOL=2311=PTOM04
/*
//
```

Следует заметить, что дистрибутивная лента изготовлена в ОС ЕС, а поэтому в OS/360 не может быть восстановлена из-за несовместимости утилиты IENMOVE в эту сторону. Если вы располагаете OS/360 и собираетесь в дальнейшем работать именно в ней, то это единственный шаг, который должен быть сделан в ОС ЕС. Все остальные шаги генерации необходимо проводить в OS/360.

**A1.2. Вторая стадия генерации.** Ниже приведено задание для запуска второй стадии генерации:

```
//STAGE2 JOB
// EXEC ASMFC,PARM.ASM=(NOLOAD, DECK)
//ASM.SYSLIB DD DSN=GRAF.SYGRUS,DISP=OLD
//ASM.SYSPUNCH DD UNIT=2400,VOL=SER=TAPE,
// LABEL=(,NL)
//ASM.SYSIN DD *
<макрокоманды генерации>-
/*
//
```

В результате будет получена лента с потоком заданий для третьей стадии. Макрокоманды генерации следующие:

- 1) GRAFOR UNIT=тип, VOL=, DSN=имя
- 2) SUPPORT DEVTYPE=(тип, )
- 3) GENERATE TYPE=, JOBSTMT=SUPPLIED

Макрокоманда GRAFOR определяет резиденцию системы: тип устройства и серийный номер тома, а также имя набора данных, если оно отлично от SYS1.GRAFLIB (последнее употребляется по умолчанию).

Макрокоманда SUPPORT описывает типы устройств, которые комплекс должен обеспечивать. В качестве типа следует указывать номера устройств 7051, 7052, 7053, 7054 или 7251.

Макрокоманда GENERATE завершает обработку макрокоманд генерации. Все ее параметры необязательны. Параметр TYPE может принимать два значения: IO и ALL. ALL означает, что проводится полная генерация. Параметр IO означает, что требуется добавить обеспечение новых устройств. При этом в макрокоманде SUPPORT необходимо указать все типы требуемых устройств, в том числе и уже обеспеченные предыдущей генерацией. Параметр JOBSTMT=SUPPLIED употребляется в том случае, если вы сами генерируете оператор JOB для 3-й стадии. Это можно сделать, например, поместив следующий оператор перед всеми макрокомандами генерации: PUNCH '//GEN JOB (3, 15), "ЛОГОВ", MSGLEVEL=0'

Еще одна макрокоманда описана в п. A2.1.

**A1.3. Третья стадия генерации Графора.** Для выполнения этой стадии необходимо запустить поток заданий, полученный на предыдущей стадии. Это можно сделать с операторской консоли посредством команды

S RDR, *сuu*, TAPE, LABEL=(, NL)  
в MVT и

S RDR.Pr, *сuu*, TAPE, LABEL=(, NL)

в MFT, где *сuu* — адрес лентопротяжного устройства, на котором установлена лента с потоком заданий, а *n* — номер доступного раздела.

**A1.4. Выполнение тестовых примеров.** Три тестовых примера находятся в библиотеке GRAF.SYGRUS в виде текстов фортран-программ. Они называются GIRL, ULITA и УТКА. Для их выполнения достаточно пропустить задания, аналогичные следующему:

```
//NAME JOB MSGLEVEL=(1, 1)
// EXEC GRAFGCLG
//FORT.SYSIN DD DSN=GRAF.SYGRUS(GIRL), DISP=SHR
//GO.SYSGRF DD UNIT = сuu
//
```

где *сuu* — адрес графического устройства или его наименование. При выводе на перфоленту или магнитную ленту для устройства EC-7054 и EC-7052 необходимо в DD-предложении с именем SYSGRF указать параметр DCB=BLKSIZE=80 или 160.

## A2. Графтор и операционная система

Для успешной генерации системы Графтор необходима операционная система не ниже 20-го релиза OS/360 или не ниже 4.0 ОС ЕС. Если имеющийся в вашем распоряжении релиз не обеспечивает ваших графических устройств (например, OS/360), то это не служит препятствием для установки комплекса Графтор. В этом случае при генерации операционной системы (см. [19]) вы описываете имеющиеся графические устройства следующим образом:

```
IODEVICE UNIT=DUMMY, ADDRESS=сuu, *
DEVTYPE=100010xx, ERTAB=eee
UNITNAME UNIT=сuu, NAME=name
```

где *сuu* — адрес устройства, определенный при его подключении, *eee* — три десятичные цифры в диапазоне 220—229, определяющие имя модуля коррекции ошибок ввода/вывода, *xx* — две шестнадцатеричные цифры, определяющие тип устройства по таблице:

xx	Тип устройства	xx	Тип устройства
FF	EC-7051	FD	EC-7053
FE	EC-7052	FC	EC-7054

где *name* — имя устройства, например 7052. Включается модуль в

генерируемую операционную систему посредством макрокоманды SVCLIB. Более подробно о генерации системы сказано в [19]. Информацию о написании модулей коррекции ошибок ввода/вывода можно получить в [20].

**A2.1. Подключение нестандартных устройств.** Для обслуживания нестандартных устройств необходимо провести генерацию операционной системы, включив в нее описание нестандартного устройства, как описано выше. В операнде DEVTYPE в поле *xx* следует указать тип устройства в виде двух шестнадцатеричных цифр в диапазоне E0—FB. В операнде ERRTAB указывается имя модуля коррекции ошибок ввода/вывода. Модули коррекции ошибок ввода/вывода для нестандартных устройств вы пишете сами. Если вы по каким-либо причинам не в состоянии написать модули коррекции (например, нет соответствующей документации), то, как оказалось на практике, их вполне могут заменить всегда имеющиеся в SYS1.SVCLIB модули с именами IGE0001A и IGE0101A. Для этого достаточно с помощью редактора связей дать каждому из них соответствующий ALIAS и выполнить утилиту IENIOSUP.

Например, если в параметре ERRTAB макрокоманды IODEVICE вы указали 220, то модулю IGE0001A нужно дать ALIAS IGE0022{ (символ { имеет код C0 и на перфокарте кодируется пробивками 12—0), а модулю IGE0101A — ALIAS IGE0122{. Следует помнить, что в имени модуля последняя цифра кодируется в десятичном записанном формате, поэтому 1 кодируется как A, 2 — как B и т. д., а 0 — как {. Включение модуля производится посредством макрокоманды SVCLIB. Далее, среди макрокоманд генерации Графора вы помещаете макрокоманду

GRDEVICE TYPE=*xx*, NAME=*наме*

где *xx* — тип устройства, соответствует указанному в поле *xx* операнда DEVTYPE макрокоманды IODEVICE генерации системы; *наме* — имя типа устройства длиной 4 символа.

Компилирующую программу, хэндлер устройства и модуль вывода на перфоленду для нового устройства необходимо ввести в библиотеку GRAF.OBJ до запуска 3-й стадии генерации комплекса.

Примеры компилирующей программы, хэндлера устройства и модуля вывода на перфоленду можно найти в библиотеке GRAF.SYGRUS под именами GRC7052, GRO7052 и GRO7052#.

**A2.2. Привязка графического устройства на стадии исполнения.** Программа GRINIT с помощью макрокоманды DEVTYPE определяет тип выводного устройства. Возможны следующие случаи: выводное устройство графическое, перфоленда, магнитная лента, устройство прямого доступа.

В случае графического устройства производится загрузка соответствующей компилирующей программы (CP) и хэндлера уст-

ройства (OУH). Имена компилирующих программ имеют вид GRCxxxx, где xxxx — наименование устройства, например, GRC7052. Имена хэндлеров устройств также стандартизованы и имеют вид GROxxxx, например, GRO7051. После этого устанавливается связь программы PLOT с компилирующей программой, а компилирующей программы с хэндлером устройства. Затем выполняются инициализирующие функции компилирующей программы и хэндлера устройства.

Если выходным устройством является перфолента или магнитная лента, то программа GRINIT выдает запрос оператору с требованием указать тип графического устройства, для которого предназначается подготавливаемая лента. Оператор посредством команды REPLY вводит тип устройства. После этого производится загрузка соответствующей компилирующей программы и модуля графического вывода на ленту. Имена этих модулей отличаются от имен, употребляемых в предыдущем случае, добавлением справа символа #. Например, GR07054#. Далее производятся те же действия, что и в предыдущем случае. Использование другой компилирующей программы обусловлено тем, что для некоторых устройств (например, 7054) форматы данных в режиме ONLINE отличаются от последних в режиме OFFLINE.

Если в качестве выходного устройства указано устройство прямого доступа, то в качестве компилирующей программы используется GRC\$\$\$\$, а в качестве модуля вывода — GRO\$\$\$\$.

Как и ранее, программа PLOT выполняет три функции: присвоение текущему положению пера указанных координат, движение с поднятым пером в указанную точку из текущего положения и вычерчивание отрезка прямой из текущего положения в указанную точку. Но теперь PLOT выполняет функции посредника, обращаясь к соответствующим входам компилирующей программы. Компилирующая программа получает на входе номер функции, которая от нее требуется, и параметры этой функции.

Компилирующая программа, выполняя эти функции, генерирует на выходе графические приказы и данные и записывает их в буфер, предоставленный соответствующим хэндлером. При заполнении буфера компилирующая программа обращается к хэндлеру, который запускает операцию вывода и предоставляет компилирующей программе новый буфер.

### **А3. Каталогизированные процедуры**

Для удобства пользования Графформ имеются следующие каталогизированные процедуры:

GRAFGCL — для компиляции и редактирования программы, использующей Граффор;



GRAFGCLG — для компиляции, редактирования и выполнения программы, использующей Графор;

GRAFGCLD — для компиляции, загрузки и выполнения программы, использующей Графор;

GRAFGLG — для редактирования и выполнения программы, использующей Графор и имеющейся в виде объектного или загрузочного модуля с неразрешенными внешними ссылками;

GRAFGO — для выполнения готовой программы, использующей Графор;

GRAFUNLD — для осуществления графического вывода, буферизованного на устройстве прямого доступа или магнитной ленте;

GRAFSCR — для уничтожения набора данных, содержащего буферизованный графический вывод.

Процедура GRAFGCL имеет два шага: FORT и LKED аналогично процедуре FORTGCL. Дополнительно описываются наборы данных:

//FORT.SYSIN DD — для исходной программы,

//LKED.SYSLMOD DD — куда поместить загрузочный модуль.

Процедура GRAFGCLG имеет три шага: FORT, LKED и GO аналогично процедуре FORTGCLG. Дополнительные описания:

//FORT.SYSIN DD — для исходной программы,

//GO.SYSGRF DD — для выходного графического набора данных.

Процедура GRAFGCLD аналогична процедуре FORTGCLD и имеет два шага.

Процедура GRAFGO имеет один шаг. Дополнительные описания:

//GO.MODLIB DD — для библиотеки, содержащей программу, предназначенную для выполнения,

//GO.SYSGRF DD — для выходного графического набора данных.

Параметр процедуры NM определяет имя модуля, предназначенного к выполнению. Например:

```
//AAAA JOB
```

```
//STEP EXEC GRAFGO, NM=MAIN
```

```
//GO.MODLIB DD .....
```

```
//GO.SYSGRF DD UNIT=7052, ...
```

Процедура GRAFUNLD имеет один шаг. Дополнительные описания:

//IEFPROC.IEFRDTER DD UNIT= — определяет графическое устройство.

Параметры каталогизированной процедуры U=, V=, D= описывают тип устройства, серийный номер тома и имя набора данных, содержащего буферизованный графический вывод.

Например,

```
//EXMPL JOB  
// EXEC GRAFUNLD, U=SYSDA, V=PTOM04, D=GRF1  
//IEFPROC.IEFRDER DD UNIT=7054
```

или с операторской консоли:

```
S GRAFUNLD, 7054, U=SYSDA, V=PTOM04, D=GRF1
```

Набор данных, содержащий буферизованный графический вывод, не уничтожается по окончании работы процедуры. Для его уничтожения достаточно воспользоваться процедурой GRAFSCR:

```
// EXEC GRAFSCR, U=SYSDA, V=PTOM04, D=GRAFI
```

или

```
S GRAFSCR, U=SYSDA, V=PTOM04, D=GRAFI
```

**Примечание 1.** Если при генерации комплекса Графор требовалось обеспечить только один тип графических устройств, то полученные в результате генерации процедуры GRAFGCLG, GRAFGCLD и GRAFGLG будут иметь на шаге GO описания наборов данных с DD-именами SYSGRF вида

```
//SYSGRF DD UNIT=xxxx
```

где xxxx — код типа устройства (7051, 7251, 7052, 7053, 7054 и 7064), которое было указано при генерации.

Аналогичное описание будет иметь и процедура GRAFUNLD. Это означает, что при использовании этих процедур можно обойтись без дополнительных описаний по отношению к графическому устройству.

**Примечание 2.** Для использования компилятора с фортрана уровня H имеются соответствующие процедуры GRAFHCL, GRAFHCLG, GRAFHCLD и GRAFHLG.

**Примечание 3.** При выводе на перфоленту или магнитную ленту для устройства EC-7054 и EC-7052 необходимо в DD-предложении с именем SYSGRF указать параметр DCB=BLKSIZE=80 или 160.

#### **A4. Сообщения системы**

```
*nn GRF001D JJJJJJJ. SSSSSSS. SPECIFY GRAPHIC DEVICE TYPE.
```

**Причина.** Для шага SSSSSSS задания JJJJJJJ необходимо указать тип графического устройства.

**Действие оператора.** Необходимо ввести команду

```
Rnn, 'xxxx'
```

где xxxx — тип графического устройства, требующегося заданию.

Например:

R 05, '7052'

Примечание. Если генерацией комплекса предусмотрено обеспечение единственного типа устройств, то это сообщение не выдается.

\*nn GRF002D INVALID DEVICE TYPE — RESPECIFY.

Причина. Это сообщение выдается, если оператор в ответ на предыдущее сообщение указал неправильный тип устройства.

Действие оператора. Указать правильный тип устройства.

\*nn GRF003I UNSUPPORTED DEVICE.

Причина. Пользователь указал графическое устройство, обеспечение которого не предусмотрено данной генерацией.

Действие системы. Выдается ABEND с кодом завершения U0004.

\*nn GRF005I REINITIATION ATTEMPT IGNORED.

Причина. Пользователь повторно обратился к программе GRINIT до обращения к GRFIN (попытка реинициализации).

Действие системы. Обращение к GRINIT игнорируется.

\*nn GRF006I I/O ERROR.

Причина. Произошла ошибка ввода/вывода при работе с графическим устройством.

Действие системы. Выдается ABEND с кодом завершения U0020.

Действие обслуживающего персонала. Необходимо провести ремонт неисправного устройства.

Сообщения GRF003—GRF006 печатаются в листинг программы в MSGCLASS, указанный оператором JOB задания или используемый по умолчанию.

## **А5. Внешние спецификации драйверов устройств для Графора в ОС ЕС ЭВМ**

Программа GRINIT обеспечивает загрузку соответствующих модулей в оперативную память и их инициализацию, т. е. обращение к модулю компилирующей программы по инициализирующему входу (GRC...). Разделение функций между компилирующей программой и хэндлером ввода/вывода (GRO...) зависит от разработчика данного драйвера и не регламентируется системой Графор. Программа GRFIN обеспечивает терминирование драйвера путем обращения по терминирующему входу модуля компилирующей программы.

**А.5.1. Инициализация и завершение драйвера.** При инициализации драйвера программа GRINIT обращается к компилирующей программе по адресу <адрес точки входа +4>, т. е. используется последовательность команд

L 15, EP  
BAL 14, 4 (15)

При этом в регистре 1 находится адрес следующего списка параметров:

R1 → 0	адрес точки входа в хэндлер ввода/вывода
4	адрес точки входа в компилирующую программу
8	адрес COMMON-блока GFTAB
12	DD-имя для графического вывода
16	

Функции инициализирующей части компилирующей программы:

1. Поместить в общий блок GFTAB со смещением 28 и 32 размеры рабочего поля графического устройства в сантиметрах по осям X и Y соответственно; данные должны представляться в виде чисел с плавающей точкой обычной точности (в формате E).

2. Инициализировать хэндлер ввода/вывода, передав ему DD-имя для графического вывода. Хэндлер должен переслать это имя в поле DCBDDNAM своего DCB для графического вывода и открыть DCB.

3. Инициализировать свои внутренние переменные (задать равными нулю счетчики, текущие координаты пера, накопители холостых перемещений и т. д.).

4. Запомнить адрес точки входа в хэндлер ввода/вывода для дальнейшего использования.

При завершении программа GRFIN обращается к компилирующей программе по адресу <адрес точки входа +8>, т. е. используется последовательность команд:

L 15, EP  
BAL 14, 8 (15)

Никакие параметры при этом не передаются.

Функции завершающей части:

1. Обеспечить компиляцию и вывод буферизованного в компилирующей программе графического вывода.

2. Обеспечить завершающую последовательность приказов для графического устройства.

3. Завершить хэндлер ввода/вывода.

**A5.2. Рабочий вход драйвера.** Обращение по данному входу производится из программы PLOT и BYPASS по адресу, совпадающему с адресом точки входа, т. е. используется последовательность команд:

L 15, EP  
BALR 14, 15

При этом передаются следующие параметры: регистр 1 указывает на адрес последовательности переменной длины:

R1 → адрес → 

код операции	операнд1	операнд2
--------------	----------	----------

 ---

Первое слово последовательности содержит код операции, следующие слова — операнды. Количество операндов зависит от кода операции.

Код операции — целое число в формате F. В настоящее время компилирующая программа должна обеспечивать интерпретацию пяти кодов операции.

Код операции	Содержание операции
0	Начальная установка
1	Перемещение пера без рисования
2	Перемещение пера с рисованием
3	Смена пера
5	Установка типа линии

Далее приведем описание отдельных операций.

*Начальная установка.* Операндов нет. Текущему положению пера присваиваются координаты  $X=0$ ,  $Y=0$ . Перо должно оставаться на месте. Это всегда первое обращение к драйверу после инициализации.

*Перемещение пера без рисования.* У этой операции два операнда — координата по оси X и координата по оси Y — представленные в виде целых чисел в формате F. Координаты задаются в шагах виртуального устройства, равных 0.01 мм, и обозначают координаты точки, в которую перо должно быть переведено без рисования.

*Перемещение пера с рисованием.* Операция аналогична предыдущей с той лишь разницей, что перо должно перемещаться с рисованием.

*Смена пера.* Если устройство имеет несколько перьев, то драйвер после инициализации должен работать с пером № 1 до тех пор, пока не будет задано другое перо. Значением операнда в этой операции является целое число в формате F, обозначающее номер пера: 1 — первое перо, 2 — второе перо и т. д. Если в обращении указан несуществующий номер пера, то драйвер должен выбрать перо № 1. Драйвер должен обеспечить смену пера и необходимое холостое перемещение пишущего элемента (если это не делается

аппаратно) так, чтобы вновь выбранное перо оказалось в текущей точке и далее рисование производилось этим пером вплоть до следующей операции смены пера.

**Установка типа линии.** Для устройств, аппаратно обеспечивающих несколько типов линии (сплошную, штриховую, штрихпунктирную и т. д.), эта операция означает, что рисование должно производиться линией заданного типа вплоть до следующей установки типа линии.

Если в устройстве аппаратно реализуется только сплошная линия, то эта операция может игнорироваться. Единственный операнд этой операции представляет собой целое число в формате F и определяет тип линии: 1 — сплошная, 2 — штрихпунктирная, 3 — штриховая, 4 и более — определяются разработчиком драйвера. Если данный тип линии не поддерживается устройством, то должна быть выбрана сплошная линия.

## А6. Графтор на ЭВМ БЭСМ-6 в ОС Диспак

Рассматривается случай использования комплекса Графтор в рамках ОС Диспак на ЭВМ БЭСМ-6 при работе с шаговым графплоттером. Преобразование отрезка прямой в последовательность осевых и диагональных шагов плоттера осуществляется в программе PLOT.

Чтобы получить наиболее плотную упаковку, учитывался тот факт, что любой отрезок прямой линии строится как последовательность не более чем двух различных шагов (из восьми возможных). Поэтому команды управления движением пера можно кодировать одним двоичным разрядом, если отдельно сообщить коды команд, которые соответствуют 0 и 1.

Программа PLOT передает в операционную систему триаду слов БЭСМ-6 одного из двух следующих видов:

$N$	шкала		$= 0$
	$S_a$	или	$S_c$
	$S_d$		$t_c$

где  $1 \leq N \leq 42$  — указывает количество графических команд, занимает 6 старших разрядов слова (поле порядка); шкала — содержит последовательность нулей и единиц, соответствующих графическим командам;  $S_a$  — команда осевого шага, которая выполняется, если очередной разряд в шкале равен 0;  $S_d$  — команда диагонального шага, которая выполняется, если очередной разряд в шкале равен 1;  $S_c$  — код управляющего движения (подъем/опускание пе-

ра или смена цвета);  $t_c$  — время, необходимое для выполнения управляющего движения.

В-триаде, которая описывает управляющее движение, первое слово тождественно равно нулю.

Таким образом, за одно обращение в операционную систему передается до 42 команд, т. е. при шаге 0.1 мм можно провести отрезок длиной 4.2 мм. Для вывода информации на графплоттер программа PLOT использует экстракод

Э50 '7600'

причем на сумматоре передается адрес триады.

Программа экстракода состоит из двух частей. В первой части осуществляется блокировка программы, в которой встретился экстракод. Информация из триады пересылается в рабочие ячейки программы экстракода. Если в триаде задано управляющее движение (первое слово нулевое), то программа экстракода заносит на сумматор код смены положения пера (второе слово триады) и передает его устройству по команде

УВВ '152' (33 152)

В ячейку 'ЧАСЫ' пересылается константа  $t_c$ , характеризующая время, необходимое для выполнения управляющего движения.

Вторая часть экстракода работает по прерываниям от часов. Прерывания поступают с частотой 250 раз в секунду. При выполнении управляющего движения производится сдвиг константы в ячейке 'ЧАСЫ' (единица 25-го разряда) на один разряд влево. Когда в ячейке 'ЧАСЫ' нуль, движение исполнено. Таким образом, в нашем случае команда управления положением пера выполняется за 25 тактов. За время, требуемое для выполнения одного управляющего движения, может быть сделано 25 шагов.

Если в триаде задано неуправляющее движение, то программа экстракода перед обращением к устройству заносит на сумматор графическую команду. В зависимости от значения крайнего правого разряда шкалы выбирается команда осевого или диагонального шага. Шкала сдвигается на один разряд вправо, и количество движений пересчитывается. По каждому прерыванию исполняется одна графическая команда. Когда выбраны все движения, работа экстракода завершена. При этом программа, обратившаяся к экстракоду, разблокируется.

# АЛФАВИТНЫЙ УКАЗАТЕЛЬ ПРОГРАММ

## ПРИЛОЖЕНИЕ Б

Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
ADDLEV (C, NCN, IS, CAD, NAD)			7.2.2
ANGLER (DEIX, DELY)			3.2.4
ANGRID (XO, YO, XS, YS, M1, N1, SLOPE)			1.6.1
APPOLY (X, Y, RO, N, CFIT, K1, COF1)			5.4
ARC (X1, Y1, X2, Y2, X3, Y3, J)		MOVE, CIRCLE	1.6.2
ARCANG (R, DELX, DELY, J, TH0A, THFA)		IVEST	1.6.3
ARC1 (XS1, YS1, XF1, YF1, XC1, YC1, NCW)		ARCIB, MOVE	1.6.2
ARCC1 (XM, YM, XF, YF, J, R, JB)	GFBFTR	IVEST	1.6.2
ARCELA (A, B, ALPHA, THS, THF)	GFBFTR	ELIPS, IVEST, GFFALS	1.6.3
ARCELB (A, B, ALPHA, THS, THF)	GFBFTR	ARCELA, AREB1, ARCANG	1.6.3
ARCIA (R, THS, THF)	GFBFTR	MOVE, IVEST, GFFALS	1.6.2
ARCIB (R, XF, YF, J)	GFTAB, GFBFTR	ARCIA, ARCANG	1.6.2
ARCIC (XM, YM, XF, YF, J)	GFBFTR	ARCIA, ARCANG, ARCC1	1.6.2
ARCID (XC, YC, PHI)	GFBFTR	ARCIA, ANGLER	1.6.2
ARCOCC (R, XT1, YT1, XT2, YT2, J)	GFGEL	ANGLER, IVEST, GFFALS	3.2.4
ARCOLC (R, XT1, YT1, XT2, YT2, J)	GFGEL, GFBFTR	ANGLER, IVEST, MOVE, STRMOD, CIRTAC, GFFALS	3.2.4
ARCOLL (R, XT1, YT1, XT2, YT2, J)	GFGEL	ANGLER, IVEST, GFFALS	3.2.4
AREB1 (THO, THF, G, THO2, THF2)			
ARRINL (X, Y, A, IX, IY, IXB, IXE, IYB, IYE, DX, DY, DZ, NL, NLBEG)			
ARROW (J)			
ASTEP (AN, AX, BS, MK, KD)			
ATDX (Z, X, Y)	GFGEL, GFBFTR	STRMOD, IVEST, ANGLER	8.4.4
ATDX1 (Z, X, Y)	GFTAB	MOVE, MOVA, ARCIA, GFFALS	3.2.3
ATDX2 (Z, X, Y)			
ATDX3 (Z, X, Y)	GFTAB		4.3
			7.3.2
			7.3.2
			7.3.2



Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
ATDY(Z, X, Y)			7.3.2
ATDY1(Z, X, Y)			7.3.2
ATDY2(Z, X, Y)			7.3.2
ATDY3(Z, X, Y)			7.3.2
ATRN2(A, B, C, D, E, F)			2.1.1
ATRST			2.1.1
AXES(NAMEX, NX, UX, KX, NAMEY, NY, UY, KY, M)	GFATRM, GFATRN	MTMPL	4.3
AXISC(NAME, NC, MMIN, MS, NM)	GFTAB	ASTER, MOVE, IBCD, BCD, SYMBOL	4.6
AXONOM(X, Y, Z)	GFTAB	MOVE, SYMBOL	8.1.1
BAR(X, Y, H, W, SH, IHAT, NP1)	GFMATR	HCR0T1	4.6
BARS(Y, YB, N, W, IHAT, NP1)	GFTAB	TMF, MOVE	4.6
BCD(A, IB, N)		BAR	4.5.2
BCDX(A1, IT, N1, NF)			1.5.2
BEGLEV	GFATRM, GFATRN	GRAFER, MTMPL	2.1.2
BEZ2(XP, YP, NPLG, XC, YC, NPTS)			5.8
BEZ3(XP, YP, ZP, NPLG, XC, YC, ZC, NPTS)			5.8
BITA(ITAGB, IXBD, IYBD, NX, NY, NB)			7.2.2
BLAN			2.3
BLANC(X, Y, N, IN)			2.3
BLANCH			2.3
BOUND(Z, M, N, X, Y, NX, NY, KX, KY, NF, XMI, XMA, YMI, YMA)	GFATRN, GFBLAN, GFBLU	GFFALS	7.3.2
BOX(X, Y, XL, YL)	GFBLAN, GFATRN, GFATBL, GFBLAW, GFBLU	ATDX, ATDY	
BRLINE(X, Y, N)	GFEBT	MOVE	1.6.1
BROKEN(A1, A2, A3, A4)	GFTAB, GFBRO	WHERE, TMF, MOVE	4.2
BROLIN(K, XS1, YS1, XF1, YF1, S1, S2)	GFBRO	MOVE	1.4
BSPLN2(K, XP, YP, NPLG, XC, YC, NPTS, WFUN, NWF, VKNOTS, NKNOTS)		KNOTV2	5.9

BSPLN3(K, XP, YP, ZP, NPLG, XC, YC, ZC, NPTS, WFUN, NWF, VKNOTS, NKNOTS)	GFL	KNOTV3	5.9
BUFL(IX1, Y1, IX2, Y2, RMAX, RMIN)	GFMATR	LININT	82.3
CABIN(I)		SHEAR	8.1.1
CCNTL(CN, L, C)			3.1.3
CCNTRP(CN, P, C)			3.1.3
CELL(PH1, N4, N2)			7.2.2
CHENSP(YM, N, YH, K, C, MM)	GFCELL		5.7
CIRC(R)	GFBFTR	ARCIA, MOVE	16.2
CIRCLE(XS, YS, THS, THF, RS, RF, L)	GFTAB	MOVE	16.2
CIRTAC(R, XT, YT, J)	GFGEL, GFBFTR	STRMOD, IVEST, ANGLER,	3.2.4
CIRTAL(R, XT, YT, J)	GFGEL, GFBFTR	GFFALS	3.2.4
CMGRID(X, Y, N4, N2, MX, MY, SZ, KP, M, N, ICM)		STRMOD, IVEST, ANGLER,	3.2.4
CMLC(LM, C1, C2)		GFFALS	7.2.3
CMS		TMF, NUMBER, MOVE	7.2.3
CNCPP(P1, P2, R, N)		PMLP	3.1.3
CNCTCC(C1, C2, R, N)	GFGMP	LPP, LPERLP, PILC	1.3
CNCTCL(C, L, R, N)	GFGMC, GFGMP	PICC	3.1.4
CNCTCP(C, P, R, N)	GFGMC, GFGMP	LPARLD, PILC	3.1.4
CNCTLL(L1, L2, R, N)	GFGMC	PICC	3.1.4
CNCTLP(L, P, R, N)	GFGMP	LPARLD, PILL	3.1.4
		LPERLP, PILL, LPARLD, PILC	3.1.4
CNTLL(L1, L2, L3, M)			
CONDEK(LX, LY, X, Y, Z, S, K, N, C)	GFGMC	PILL, LPARLD, LPARLD	3.1.4
	DRINFN, DRIN	WHERE, FNROOT, INSDEK, TFM	7.1.2
COORD(CO, J)	GFTAB		4.4
COORDT(DX, DY, DZ, NLINES, N4, N2, NCLUST, NLNDR, ITYP)	GFTAB GFMT, GFCRDN, GFPS	HCNCOR, SORTY, HCINIT, STROT3, MXMULT	8.4.4

Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
CORN(LX14, LX1, LY14, LY1, K1, M1, K2, M2, X, Y, A, LXX, LYY, XF1, YF1, RMAX, RMIN)	GFVP, GFKJ, GFL, GFX, GFB, GFTAB	MAP, TMF, MOVE, PATCH	8.2.3
COSIN(X1, Y1, X2, Y2)			8.3.6
COSIN2(X1, Y1, X2, Y2)			8.3.6
CPPP(P1, P2, P3, C)			3.1.3
CPR(P, R, C)			3.2.4
CROSS(X, Y, J)	GFGEL	ANGLER, GFFALS	8.4.4
CROSSP(X, Y, RC, T)	GFEPS		3.1.3
CTCCP(C1, C2, P, N)	GFGMC		3.1.3
CTCPP(P1, P2, C, N)	GFGMC		3.1.3
CTLPL(L1, L2, P, N)	GFGMC, GFGMP	PILL, PILC, LPERLP	3.1.3
CTLPP(P1, P2, L1, N)	GFGMC	LPP, LPERLP	3.1.3
CUBPOL(X1, X2, Y1, Y2, DY1, DY2, A)			5.3
CXYR(X, Y, R, C)			3.1.3
DARC(XC, YC, XS, YS, PHI, IC)	GFTAB, GFBFTR	ANGLER, MOVE, ARCIA	16.4
DANPPC(P1, P2, PC, D, ALPHA)			3.1.3
DASHP(X, Y, DL)	GFTAB, GFBFTR	MOVE	1.4
DDIST(X, Y)	GFGEL		3.2.4
DERIV5(DX, Y, N, I)			5.3
DIMAN(XO, YO, R, A, B)			16.4
DIMDRO(D, J)	GFTAB	MOVE, ARCIA, NUMBER	3.2.3
	GFGEL, GFTAB, GFBFTR	WRSTR, RDMVA, MOVE, MOVA, STRMOD, WRMVA, GFFALS, ARROW, RDACA, RDACB, ARCIA, RDSTR	
		NARROW, NUMBER	
DIMEN(XS, YS, DATA, SLOPE)	GFTAB	TDROT	16.4
DIMET	GFMATR	STRMOD, SECANT, IVEST,	8.1.1
DIST(J)	GFGEL, GFBFTR	ANGLER, AREB1	3.2.4
DRACON(LX, LY, X, Y, Z, S, K, N, C, FX, FY)	DRINFN, DRIN	WHERE, FNROOT, TMF, INSIDE	7.1.2
DRALIM(LX, LY, X, Y, FX, FY, R)	DRLM		7.1.2

DRAWEL(T, I)					
DRAWTE					
DRFRAM(LX, LY, X, Y, FX, FY)					
DRWHL(DX, DY, DZ, NLINES, NL4, NL2)					
ELIPS(XS, YS, A, B, ALPHA, THS, THF)					
ELPS(A, B, ALPHA)					
ENDLEV					
ENDPG(NUMB)					
EXMIMA(Z, M, N, ZMI, ZMA)					
EXTLN(X, Y, Z, G)					
EXTREM(X, Y, NX, NY, XMN, XMX, YMN, YMX,					
ICM)					
EXUDE(Z, M, N, X, Y, NX, NY, KX, KY, NF,					
MRKA, MRKI, KD, H, TH)					
FAN1(XO, YO, R1, R2, A1, A2)					
FAN2(X1, Y1, X2, Y2, X3, Y3, X4, Y4)					
FATARC(R, XF, YF, J, D)					
FATLIN(X, Y, D)					
FNROOT(LX, LY, X, Y, Z, N, R)					
FORFIT(M, A, B, XBEG, XEND, MPTS)					
FORIF(FUN, N, M, A, B, IER)					
FORIT(FNT, N, M, A, B, IER)					
FULL					
GFFALS(N)					
GRAFER(IT)					
GRID(XO, YO, XS, YS, M, N)					
GRINIT					
GRFIN					
HALLINE(DX, DY, DZ, NLINES, N1, N2, NCLUST,					
NLNDR, G, TOL, ITYP)					
GFLNI					
GFSV, GFEPS					
GGMT, GFCDRN					
GFTAB					
GFBFTR					
GFTAB, GFERR					
GFEPS					
GFTAB, GFBFTR					
GFTAB, GFBFTR					
DRINFN					
GFTAB					
GFBR					
GFGOBS					
GFERR					
ENDPG					
MOVE					
COORDT, HALOED					
TMF, MOVE					
DRAWEL					
TMF, MOVE					
HCNCOR, TMF, MOVE					
MOVE					
ELIPS, MOVE					
ATRST, RENTCH, REBLAN,					
MOVE, SYMBOL, NUMBER					
XPOL, YPOL					
ATDX, ATDY, TOKEN					
ARC1, MOVE					
ARC1, MOVE					
MOVE, MOVA, ARCIA,					
ARCANG					
MOVE					
ITPLBV					
MOVE, TMF					
ENDPG					
MOVE					
COORDT, HALOED					

Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
HALOED(DX, DY, DZ, N LINES, NDRAW1, NDRAW2, NC1, NC2, NCLUST, NLNDR, G1, TOL1, ITYP)	GFEPs, GFGTOL, GFSV	HLDLN, DRAWTE	8.4.4
HCIND(X, IST, IFN, EL)			
HCINIT(A)			8.1.2
HCINV(X, Y, Z, XP, YP, ZP)	GFMATR		8.4.4
HCLINE(X, Y, NP)			8.1.1
HCMULT(A, B)		LINEO	8.4.2
HCNCOR(X, Y, Z, T)	GFCRDN		8.1.1
HCNCRD(X, Y, Z)	GFMATR, GFCRD		8.4.4
HCPRSP(H)	GFMATR		8.1.2
HCR0T1(X, Y, Z)		HCUNIT, HCMULT	8.1.1
HCSURF(X, Y, Z, NROW, NCOL, ISTA, IFN, JSTA, JFN, LNT, MOVX, MOVY, AMXMN, AR)	GFSUR	TDROT	8.1.1
HCUNIT(A)		TDSECT	8.1.2
HISTGM(XO, DX, YO, YV, N, NP1)	GFTAB	TMF, MOVE	8.1.1
HLDLN(DX, DY, DZ, N LINES, I, NC1, NC2, NCLUST, NLNDR, ITYP)	GFEPs, GFGTOL, GFSV, GFLNI	EXTLN, GROSSP, STORIN	4.6
IADR(A)			8.4.4
IBCD(IA, IB)			2.2.2
IDENT(MD, ND)			4.5.2
ILIMTN(L, IR, IBOUND, NB)			4.3
INCHES			8.3.6
INCLIN(XBEG, DXEX, JX, Y, N, NM, JS)	GFTAB	MARKER, TMF, WHERE	4.3
INIT	GFMATR	HCUNIT	4.2
INSDEK(LX, LY, X, Y, Z)	DRINFN, DRIN	TMF, MOVE, ITPLBV, NUMBER	8.1.1
INSIDE(LX, LY, X, Y, Z, FX, FY)	DRINFN, DRIN	TMF, MOVE, NUMBER, ITPLBV	7.1.2
INTRSC(X1, Y1, X2, Y2, X3, Y3, X4, Y4, X0, Y0)			7.1.2
			8.2.3

ISOLIN(NX, NY, X, Y, Z, ITAGB, NB, IXBD, IYBD, NCN, CONT, ICM)	GFCELL, GFISOL	XPOL, YPOL, LINEO, LOOK	7.2.2
ISOMET	GFMATR	TDROT	8.1.1
ITALIC(I)			1.5.1
ITPLBV(LX, LY, X, Y, Z, N, U, V, W)			6.2
IVEST(A, B, EPS)			3.2.4
IZFLIN(Z, M, N, X, Y, ZIZ, L, NX, NY, KX, KY, NF, XI, YI, NL)		RAISE, IZLIN, LOWER	7.3.2
IZLIN(Z, M, N, X, Y, ZMI, ZIZ, NIZ, NX, NY, KX, KY, XI, YI, NL)		RECUR, XILYI, LETIZO	7.3.2
IZOLIN(Z, M, N, X, Y, ZIZ, L, XI, YI, NL)		RAISE, IZLIN, LOWER	7.3.2
KEYPO(Z, X, Y, NX, NY, ISX, ISY, NM, SZ, KP, ICM)		XPOL, YPOL, TMF, MOVE, MARKER, NUMBER	7.2.3
KNOTV2(K, XP, YP, NPLG, VKNOTS, NKNOTS)			5.9
KNOTV3(K, XP, YP, ZP, NPLG, VKNOTS, NKNOTS)			5.9
LCROSS(X1, Y1, X2, Y2, X3, Y3, X4, Y4, C, D)		SMINV	8.3.6
LESQ(X, Y, RO, M, B, N)		TRANSF, MARKBE,	5.4
LETIZO(XI, YI, ILM, ZIZ, NIZ, KIND)	GFBET, GFISOL	LINNUM	7.3.2
LETSPL(XI, YI, ILM, ZIZ, NIZ, KIND)		TDLINE	7.3.4
LEVFUN(Z, NX, NY, C, NCN)	GFNAM2	MINMAX	7.2.2
LEVMAP(C, NCN, FO, SP, SN, NP)			7.2.2
LGLINE(X, Y, N, LG, NM, JS, L)	GFTAB	WHERE, TMLGF, MOVE, MARKER	4.4
LICON(XT1, YT1, XT2, YT2, J)	GFGEL	ANGLER, IVEST, GFFALS	3.2.4
LIMITS(XMIN, XMAX, YMIN, YMAX)	GFTAB		4.1
LINEC(X, Y, N)	GFTAB	TMF, MOVE	4.2
LINEMO(X, Y, N, NM, JS)	GFTAB	WHERE, TMF, MOVE, MARKER	4.2
LINEMC(X, Y, N, NM, JS)	GFTAB	TMF, MOVE, MARKER	4.2
LINEO(X, Y, N)	GFTAB	WHERE, TMF, MOVE	4.2

Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
LINFIL(A, B, N, M)		GRAFER	5.6
LININT(JX1, Y1, JX2, Y2, RMAX, RMIN)			8.2.3
LINNUM(X, Y, N)		TMF, BRLINE, NUMBER	4.2
LINT(X, Y, XX, YY, NX, NY, KT, ICM)	GFLOOK, GFISOL,	XPOL, YPOL	7.2.3
LITAN(XT, YT, J)	GFLSZ, GFLMN	RDCRC, IVEST, ANGLER,	3.2.4
	GFCELL, GFINT	GFFALS	
	GFBFTR	PILL, PILC, PMLP, LPP,	3.1.2
LMLL(LM, L1, L2)	GFGMP	LPARLP	
LOCEXT(Z, X, Y, NX, NY, ICM)	GFLMN	XPOL, YPOL, TMF, MOVE,	7.2.3
LOOK(NX, NY, X, Y, Z, ITAGB, XX, YY, KT, ICM)	GFCELL, GFLOOK,	MARKER, NUMBER	
	GFINT	ZAPIT, LINT, CELL,	7.2.2
		LINNUM	
LOWER(Z, M, N, ZMI)		LPAX	7.3.2
LPAL(P, A, L1, L)			3.1.2
LPARD(L1, D, L2)			3.1.2
LPARLP(L1, P, L2)			3.1.2
LPARX(D, L)			3.1.2
LPARY(D, L)			3.1.2
LPAX(P, A, L)			3.1.2
LPERLP(L1, P, L2)			3.1.2
LPP(P1, P2, L)			3.1.2
LSCALE(X1, Y1, X2, Y2, R)			3.1.2
LSFIT(X, Y, RO, M, N, MPTS)			2.1.1
			5.4
MAP(X, Y, Z)	GFTAB	ROTATE, ATRAN2	
MARKBE(X1, Y1, IL)	GFVP	MOVE, TMF, PVAL, LESQ,	
MARKER(I)	GFBEF	MINMAX, GRAFER	
MATEVL(XV, YV, ZV, XPL, YPL, ZPL, VX, VY, VZ)	GFTAB, GFBFTR		8.2.3
	GFTRG2	TMF, MOVE	7.3.2
		MOVE	1.5.3
MINMAX(A, N, AMN, AMX)			8.3.6
			4.1





Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
NUMLIN(SIZE, N)	GFLMN		7.2.2
OPER(L, K, X, R)			6.1
PAC(C, A, P)			3.1.1
PAGE(XL, YL, NAME, N, J)	GFTAB	MOVE, IDENT, SET, SYMBOL, ITALIC, BOX PAC	4.3
PATA(C, A, A2, N, I, P)			3.1.1
PATC(C, N, A, I, J, P)			3.1.1
PATCH(XF1, YF1, RMAX, RMIN)	GFL, GFX, Y, GFB, GFTAB	LININT, SEARCH, BUFL, MOVE	8.2.3
PATL(PH, PK, N, I, P)	GFGMP	/	3.1.1
PCDP(PC, P, D)			3.1.1
PCNAP(PC, P1, ALPHA, P)			3.1.1
PCNITC(P1, P2, P3, PT1, PT2, PT3, C)	GFGMP	LPP, PILL, LPERLP	3.1.1
PCNP(P1, C, P, D)		LPP, PILC, SPNP	3.1.1
PCNTRC(C, P)	GFGMP		3.1.1
PICG(C1, C2, N)	GFTRG1		3.1.1
PICTUR(ISC, SCRN, IDIM1, IDIM2, X1, Y1, X2, Y2)		SEGANG, SCREN1, SCREN2, LCROSS, TMF, MOVE	8.3.6
PIEPP(P1, P2, A, B, K, P)	GFGMP		3.1.1
PILC(L, C, N)			3.1.1
PILL(L1, L2, P, N)			3.1.1
PISS(P1, P2, P3, P4, P, N)			3.1.1
PLDP(L, P, D)	GFGMP	LPP, PILL	3.1.1
PLNP(P1, L, P, D)			3.1.1
PLOT(X, Y, J)		LPERLP	3.1.1
PMIDPP(P1, P2, P)			1.4
PMLP(L, P1, P2)			3.1.1
PMP(P1, P2, J)			3.1.1
PMPP(P1, P, P2)			3.1.1
PNORDR(X, Y, NET, L, IB, IE)			8.3.6
POLG(R, M, PHI)	GFBFTR	MOVA, MOVE	1.6.1



Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
RDAEB(XO, YO, A, B, ALPHA, XF, YF)	GFGEI	STRMOD, GFFALS	3.2.2
RDCRC(XC, YC, R)	GFGEI	STRMOD, GFFALS	3.2.2
RDELP(XC, YC, A, B, ALPHA)	GFGEI	STRMOD, GFFALS	3.2.2
RDMVA(XO, YO, DL, TH)	GFGEI	STRMOD, GFFALS, ANGLER	3.2.2
RDMVB(XO, YO, DELTAX, DELTAY)	GFGEI	STRMOD, GFFALS	3.2.2
RDMVC(XO, YO, XM, YM, DL)	GFGEI	STRMOD, GFFALS	3.2.2
RDMVE(XO, YO, X, Y)	GFGEI	STRMOD, GFFALS	3.2.3
RDSTR(STORE)	GFGEI	MOVE	2.3
REBLAN	GFATR, GFBLAN	MOVE	16.1
RECT(X, Y, H, W, TH)		MOVE	7.3.2
RECUR(Z, M, N, NX, NY, KX, KY)	GFTAB	BOX, SYMBOL	4.1
REGION(X, Y, XL, YL, NAME, N, J)	GFGOBS		2.2.2
RENTCH	GFLMN		4.2
RENUM	GFATRM, GFATR		2.1.1
RESET			2.1.1
ROTATE(X, Y, PSI)	GFCRDN	ATRAN2	8.4.4
ROTH(NAXES, FI, DX, DY, DZ, NLINES, NBEG, NEND)		STROT3, HNCOR	
SAVETR(A)	GFMATR, GFTR		8.1.1
SCREN1(ISCR, SCR, IDIM2, Y, N, IND)			8.3.6
SCREN2(ISCR, SCR, IDIM2, X, Y, NL, STEP, XBEG, A1, B1, A2, B2, IND)			8.3.6
SCRMOD(ISCR, SCR, IDIM1, IDIM2, XL, YL, XIR, YIR, XK, YK)	GFTRG1	SEGAN	8.3.6
SDPG(STEP, EPS, BETA)	GFTAB	MOVE	2.4
SEARCH(IX1, Y1, IX2, Y2, XF1, YF1, RMAX, RMIN)	GFL, GFXY, GFTAB	INTRSC, BUFL, MOVE	8.2.3
SECANT(SL, ALPHA, X, Y)	GFGEI, GFBFTR		
SEE(L, IR, IBOUND, X, Y, NB, IND, IDRW, XV, YV)		RDMVA, IVEST, WRMVA, RDSTR, GFFALS	3.2.4
SEE1(I1, I2, I3, I4, X, Y, XV, YV, IND)		SEE1	8.3.6
		LCROSS	8.3.6



Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
SPLINE(X, Y, U, N, A, B, C, D, KODE, IER)		TRIDIG, TDMP, GRAFER	5.1
SPLINT(X, N, A, B, C, D, Y, M)			5.1
SPLITP(X, Y, Z)			8.4.4
SPNP(P, PN)	GFGMP	POLG, MOVE, MOVA, IVEST	3.15
STAR(R, M, ALPHA, PHI)	GFBFTR	SVLSEG, SHDOWN, SHUP	4.6.1
STORIN(AL, AU)	GFSV, GFEPS	TESTVS	8.4.4
STRMOD(J)	GFGEL		3.2.3
STROT3(NAXES, CS, SN, R)		HCINIT	8.4.4
SUPLIN(X, Y, N, NM, JS, L, ICM)		XPOL, YPOL, LINEMC, LINEMO, BRLINE	7.2.3
SVLSEG(AL, AU, I)	GFSV		8.4.4
SYMBOL(X, Y, SIZE, JTEXT, N, THETA)	GFTAB, GFATR	SYMTAB, MOVE	15.1
SYMTAB(JTEXT, I, NPLOT, KK, ISET)			15.1
TCAIC(X, Y, T, N)			5.2
TDLIM(X, Y, Z, NY, NX, ISTA, IFN, JST, JFN, S)	GFCRD	HCNCRD, LIMITS	8.1.2
TDLIML(X, Y, Z, N, S)	GFCRD	HCNCRD, LIMITS	8.1.3
TDLINE(X, Y, Z, N)	GFCRD	HCNCRD, TMF, MOVE	8.1.3
TDMP(X, Y, N, A, B, C, D, KODE)	GFMATR	HCUNIT, HCMULT	5.1
TDROT(NAXES, ALPHA)	GFMATR	HCUNIT, HCMULT	8.1.1
TDSCAL(NAXES, SCALE)	GFMATR, GFCDR, GFLIM	HCNCRD, ZINT, HCLINE	8.1.2
TDSECT(X, Y, Z, IX, IY, M, N, AMXMN, AR)			
TDTRAN(DX, DY, DZ)	GFMATR	HCUNIT, HCMULT	8.1.1
TESTVS	GFSV, GFEPS		8.4.4
TFM(XF, YF, XM, YM)	GFTAB		4.1
THAXES(NAME, N, H, UT, KT, MT, R)	GFTAB	SIGNA, TPF, MOVE, NUMBER, CIRCLE	4.5
THREED(X, Y, Z, NY, NX, ISTA, IFN, JSTA, JFN, LNTP, NGRD, AMXMN, AR)	GFMATR, GFVIEW, GFCRD, GFLIM	HCNCRD, HCSURF	8.1.2
TMF(XM, YM, XF, YF)	GFTAB		4.1
TMLGF(X, Y, LG, N, XF, YF)	GFTAB		4.4

[illegible]

Имя программы	Общие блоки	Требуемые подпрограммы	Пункты
WRMVA(DL, TH)	GFGEL, GFBFTR	WRMVE	3.2.2
WRMVB(DELTA, DELTAY)	GFBFTR	WRMVE	3.2.2
WRMVC(X, Y, DL)	GFBFTR	WRMVE	3.2.2
WRMVE(X, Y)	GFGEL, GFBFTR	STRMOD	3.2.2
WRSTR(STORE)	GFGEL		3.2.3
XAXIS(YO, NAMEX, NX, UX, KX, M, J)	GFTAB	ASTEP, TMF, MOVE, SYMBOL, IBCD, BCD	4.3
XILIY(Z1, Z2, ARG1, ARG2, ZIZOL, I, J, IUS, JUS)	GFNAM1, GFNAM2		7.3.2
XILIY1(Z1, Z2, ARG1, ARG2, ZIZOL, I, J, IUS, JUS)			7.3.4
XLGAX(YO, NAMEX, NX, JX, KX, M)	GFTAB	TMLCF, SYMBOL, NUMBER, MOVE	4.4
XPOL(R, T)			7.2.2
YAXIS(XO, NAMEY, NY, UY, KY, M, J)	GFTAB	ASTEP, TMF, MOVE, SYMBOL, IBCD, BCD	4.3
YLGAX(XO, NAMEY, NY, JS, KY, M)	GFTAB	TMLGF, SYMBOL, NUMBER, MOVE	4.4
YPOL(R, T)			7.2.2
ZAPIT(ITAGB, N1, N2)	GFCCELL		7.2.2
ZINT(X1, Z1, X2, Z2, X)			8.1.2

## ПРИЛОЖЕНИЕ В

### ОБЩИЕ БЛОКИ И ИХ ДЛИНЫ

DRIN(17)	GFBRO(5)	GFISOL(2)	GFNTCH(51)
DRINFN(4)	GFCELL(9)	GFKJ(2)	GFNU(80)
DRLM(4)	GFCRD(3)	GFL(8)	GFSUR(4)
GFATBL(6)	GFCRDN(3)	GFLIM(4)	GFSV(32)
GFATRM(13)	GFEPS(2)	GFLMN(4)	GFTAB(22)
GFATRN(7)	GFERR(5)	GFLNI(6)	GFTR(2)
GFB(7)	GFGEL(28)	GFLOOK(1)	GFTRG1(8)
GFBET(3)	GFGMC(16)	GFLSZ(1)	GFTRG2(18)
GFBFTR(2)	GFGMP(4)	GFMATR(34)	GFVFP(6)
GFBLAN(11)	GFGOBS(20)	GFMT(52)	GFVIEW(23)
GFBLAW(4)	GFGTOL(4)	GFNAM1(2)	GFX(5)
GFBLU(166)	GFINT(1)	GFNAM2(2)	



## СПИСОК ЛИТЕРАТУРЫ

1. Графор: комплекс графических программ на фортране.— М.— (Препринты/ИПМ АН СССР):  
а) 1972, № 41; б) 1973, № 52; в) 1974, № 88; г) 1974, № 79; д) 1975, № 90; е) 1975, № 109; ж) 1976, № 28; з) 1977, № 37; и) 1977, № 102; к) 1978, № 127; л) 1981, № 69; м) 1982, № 13; н) 1982, № 78; о) 1982, № 95.
2. Баяковский Ю. М., Галактионов В. А., Михайлова Т. Н. Графор: комплекс графических программ на фортране. Ч. 1, 2.— М.: Изд-во ИПМ, 1983.
3. Буликов В. Г. и др. Устройство регистрации графической информации для ЭВМ Минск-32.— М.: Изд-во ИПМ, 1977.
4. Максимов В. С. Использование машинной графики в системах проектирования на базе ЕС ЭВМ. Учебное пособие.— Горький: Изд-во ГГУ, 1979.
5. Львов В. А. и др. Комплекс программ базового математического обеспечения вывода графической информации. Часть I. Руководство программиста. Часть II. Инструкция по эксплуатации.— М.: Изд-во ЦНИПИАСС, 1979.
6. Пахомов В. Л. Адаптация Графора для устройств ADMAR, Дигиграф ЕС-7054, CALCOMP-565 на БЭСМ-6.— Сообщение ОИЯИ. Дубна, 1980.
7. Тодорой Д. Н., Романчук Л. Н., Перетятков С. М. Языки машинной графики.— Кишинев: Картя Молдовеняскэ, 1980.
8. Карпов И. И., Лазутич Ю. М., Ярошевский В. С. Графор-М: Комплекс графических программ на фортране для ЭВМ М-6000.— М., 1980.— (Препринт/ИПМ АН СССР: № 138).
9. Александров М. А., Лазутич Ю. М. Комплекс программ на фортране для работы с графическим дисплеем СИГДА на ЭВМ М-6000.— М., 1981.— (Препринт/ИПМ АН СССР: № 89).
10. Лацис А. О., Романенко С. А. Графор-БГП: Интерактивная версия системы Графор.— М., 1983.— (Препринт/ИПМ АН СССР: № 89).
11. Алберг Дж., Нильсон Э., Уолш Дж. Теория сплайнов и ее приложения.— М.: Мир, 1972.
12. Хемминг Р. В. Численные методы.— М.: Наука, 1972.
13. Рябенский В. С. Локальные формулы гладкого восполнения и гладкой интерполяции функций по их значениям в узлах неравномерной прямоугольной сетки.— М., 1974.— (Препринт/ИПМ АН СССР: № 21).
14. Akima H. A new method of interpolation and smooth curve fitting based on local procedures.— ACM, 1970, 17, № 4, p. 589—602.

15. Ньюмен У., Спрулл Р. Основы интерактивной машинной графики.— М.: Мир, 1976.
16. Роджерс Д., Адамс А. Математические основы машинной графики.— М.: Машиностроение, 1981.
17. Гилой В. Интерактивная машинная графика.— М.: Мир, 1981.
18. Фокс А., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве.— М.: Мир, 1982.
19. IBM System/360 Operating System: System Generation.— GC28—6554.
20. IBM System/360 Operating System: Input/Output Supervisor. Program Logic Manual.— GY28—6616—7.

## ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Аппроксимация 112  
 — В-сплайнами 137  
 — конечным рядом Фурье 126  
 — методом Безье 134  
 — многочленами Чебышева 132  
 —, точность 112

Базис Бернштейна 137  
 —, В-сплайн 137  
 Бергштрихи 151, 174, 177, 180, 184  
 Библиотека дистрибутивная 254,  
 255  
 Блок общий 56, 62—64, 67—71, 80,  
 156, 216, 249, 250, 262  
 БЭСМ-6 5, 11, 12, 25, 111, 112, 233,  
 254

Вектор касательный 135  
 — переноса 191  
 — свободных членов 113—116  
 — узловой 138, 140  
 Вершина кратная 139  
 Восполнение 112  
 —, гладкость 143  
 — функций двух переменных 142,  
 145, 160  
 В-сплайн 137

Генератор прямых линий 18  
 — символов 21  
 — чисел 25  
 Генерация Графора 255, 258  
 Гистограмма 5, 88, 108

Допуск по глубине 237, 244, 247  
 Драйвер 8, 10, 254, 262

Единица измерения 14, 17, 18  
 ЕС ЭВМ 5, 11, 17, 25, 254

Зависимость запаздывающая 143,  
 146  
 — опережающая 143  
 Зарубка 41—46, 54  
 —, банк 41—46

Изолиния (линия уровня) 89, 147,  
 151  
 — замкнутая 159, 175  
 —, маркировка 153

Изолиния незамкнутая 159, 175  
 — основная 161  
 — промежуточная 161  
 Интерполяция 142  
 — линейная 11, 151, 158, 164, 175,  
 179, 204, 217  
 —, методы 142  
 — функций двух переменных 142

Касание окружности 80  
 Конус зрения 214  
 Координата сдвигающая 191  
 Координаты математические 8, 88,  
 90, 101, 107  
 — однородные 37, 188, 190, 231  
 — предварительные 17, 44  
 — результирующие 17, 43  
 — страничные 8, 43, 44, 46, 90, 101  
 — устройства 8, 14  
 — — нормированные 14

Коэффициенты весовые 122, 125  
 — многочлена кубического 113, 121  
 — ряда Фурье 127, 129  
 Кривая 90, 105, 117, 188  
 — Безье 135, 139  
 —, В-сплайн 138  
 —, гладкость 112  
 — замкнутая 51, 92  
 — локальная 119, 121  
 —, параметрическое представление  
 118  
 —, порядок 137  
 — пространственная (трехмерная)  
 136, 137, 140, 174, 181, 186, 190,  
 204, 208, 252  
 Курсив 23

Лента дистрибутивная 255  
 Линия визирования 190  
 — касательная 64, 67  
 — ломаная 42, 49, 50, 80, 90, 91, 227  
 — «негативная» 92, 93  
 — пересечения поверхностей 8, 174  
 — прямая 18, 55, 65, 79  
 —, генератор 18  
 — размерная 34, 77, 87  
 — сплошная (непрерывная) 9, 13,  
 20, 31, 93, 161, 166, 176  
 —, тип 13, 20  
 —, толщина 20, 31  
 — уровня (см. изолиния)  
 — штриховая 13, 20, 31, 92, 93, 124,  
 161, 166, 176

Линия штрихпунктирная 9, 13, 20,  
92, 93, 161, 166, 176  
— экранная 229  
Литера (символ) 14, 21—23  
—, курсив 23  
—, наборы 21, 23  
—, наклон 12, 23  
—, способ кодирования 21  
—, таблица кодов 21  
Ломаная аппроксимируемая 137,  
139, 140  
— Безье 135—137  
— рельефа максимального 200, 211  
— — минимального 200, 211  
Луч зрения 188, 193, 228

Маркер 7, 26, 91, 92, 166, 180  
Маркировка 92, 106  
Матрица единичная 38, 195, 250  
— однородных координат 231  
— поворота 191, 250  
— преобразования 193, 195, 243  
— — результирующего 37, 193  
— — текущего 195  
— сдвига 191  
— симметрическая 122  
— трехдиагональная 113, 116  
— — дополненная 113, 116  
Метод Безье 134  
— В. С. Рябенского 142, 149  
— наименьших квадратов 121, 130  
— ореола 189, 236  
— приоритетов 236  
— прогонки 113, 116  
— слоев (параллельных сечений)  
188, 190, 197  
— Х. Акима 142, 149  
— ячеек 188, 197, 210  
Многоугольник 7, 26, 52, 189, 200  
— выпуклый 28  
— звездчатый 28, 29  
— правильный 28, 29  
Многочлены Бернштейна 135, 137  
— бикубические 142, 145  
— кубические 113  
— ортогональные 125  
— Чебышева 132

Область 12, 47, 52, 88, 201  
— аппроксимации 143  
— описатель 160, 162, 163  
— отображения 88  
— полярная 12, 104, 107  
Оболочка выпуклая 136  
Окно 9, 152  
Окружность 29, 55, 66, 73  
—, дуга 7, 29, 64, 73  
—, касательная 80, 81, 84  
—, определение центров 68  
Операции геометрические 55, 77  
Ореол 189, 236  
—, метод (эффект) 189, 236  
Ось, деление 96  
—, — дополнительное 96, 97  
—, — основное 96, 97  
— календарная 128  
— координат 88, 96, 193, 208  
— полярная 103  
— радиальная 106  
— симметрии 38, 66  
— угловая 106  
Отсечение 214, 239  
— (срезка) по странице 8, 19, 42, 44

Перо 13, 15, 17—19, 49, 90  
—, траектория 10, 41, 49  
—, — предварительная 42, 44  
—, — результирующая 42, 44  
Плоскость картинная 188, 190, 210,  
227, 238  
Поверхность 7, 188  
— видовая 8, 9, 10, 19  
— неоднозначная 201, 207, 213, 221  
Позиция текущая 73, 76, 78, 81  
Пределы изменения 89  
— координат сетки 154, 155  
— функции; аргумента и функции  
89, 105, 109, 163, 177, 201, 204, 214,  
231, 243, 245  
Преобразование 17, 194, 250  
— аффинное 8, 9, 37, 39, 40, 46, 52,  
155, 190  
— внешнее (глобальное) 39  
— внутреннее (локальное) 39, 40  
— линейное 9, 36, 37, 43, 49, 155  
— —, уровни вложенности 39, 41  
— масштабирования 9, 38, 190, 191  
— нелинейное 10, 36, 43, 54  
— обратное 38  
— основное 38, 190, 192  
— —, суперпозиция 38, 190, 193  
— переноса 38, 190, 191, 195  
— перспективы 190  
— поворота 9, 38, 190, 191, 195  
— проецирования 190  
— результирующее 37, 38; 194  
— —, инициализация 195  
— —, матрица 37, 193  
— сдвига 190, 191, 195  
— симметрии 38, 195  
— текущее 38, 196  
— тождественное 38  
— центрального проецирования 197  
Программа инициализации 10  
— компилирующая 11, 13, 254, 258,  
259  
Проекция 188, 190  
— аксонометрическая 188, 190, 192,  
196, 238  
— диметрическая 192, 196  
— изометрическая 192, 196, 198, 205  
— параллельная 192, 245  
— — косоугольная 192, 196  
— — прямоугольная 192  
— поверхности 6, 8, 89, 188  
— пространственной кривой (ли-  
нии) 204, 208, 252  
—, центр 188, 196, 210, 227, 238, 245  
— центральная 188, 190, 210, 227,  
236  
Проецирование 190, 192, 231  
—, направление 192, 193, 196, 238,  
245  
— ортогональное 192  
— параллельное 192  
—, режим 242, 245  
— центральное 191, 196  
Пространство математическое 88, 89,  
105  
Процедура каталогизированная 255,  
259  
Прямоугольник 28  
  
Ракурс 190, 196, 213  
—, выбор 212, 245  
Решения выбор 69  
Ряд Фурье 126

**Сглаживание** 112  
 —, линейный фильтр 129  
 —, методом наименьших квадратов 121  
 —, сдвиг, коэффициент 191, 192  
**Сектор кругового кольца** (кольцевой) 29, 32  
**Сетка** 7, 99—101, 142, 164, 248  
 —, координатная 97, 99, 105  
 —, криволинейная 188, 198  
 —, логарифмическая 100, 101  
 —, полярная 153  
 —, прямоугольная 28, 142, 145, 154, 190, 210  
 —, неравномерная 142, 145, 158, 175, 181, 210  
 —, равномерная 92, 114, 120, 132, 211  
 —, реорганизация 226  
 —, треугольная 189, 222, 227  
 —, физическая 189, 200, 212, 229  
**Сечения параллельные** 188, 190, 197  
**Симметризация** 143, 146, 155  
**Система координат** 8, 14, 88  
 —, декартова 88, 90, 151, 158, 161, 164, 177, 190  
 —, картинной плоскости 238  
 —, недекартова (произвольная) 151, 161, 174  
 —, объектная 238, 250  
 —, полярная 57, 88, 103, 158, 161, 164  
 —, сферическая 173, 178  
 —, цилиндрическая 178, 185  
 —, мониторинг Дубна 254  
 —, операционная 11, 16, 254, 257  
 —, ДИСПАН 11, 265  
 —, ОС ЕС 254, 256, 257, 262  
 —, OS/360 256, 257  
 —, супервизор графический 11  
 —, сообщения 261  
**След пера** 10, 36, 41, 48, 49, 53  
 —, предварительный 8  
 —, режим формирования 44  
 —, результирующий 8, 43  
**Сообщение** 261, 262  
 —, диагностическое 29, 39, 45, 49, 75, 79, 225, 231  
**Спираль** 29, 31, 32  
 —, дуга 31  
**Сплайн** 112  
 —, В-сплайн 137  
 —, интерполяция 5, 151  
 —, кубический 113, 114, 139  
 —, локальный 119, 120  
 —, периодический 113, 114, 116  
**Сплайн-аппроксимация** 112, 114  
**Сплайн-функция** 113, 139  
**Страница** 9, 12, 14, 17, 51, 88, 201  
**Суперпозиция** 38, 190, 193

**Текст** 7, 21, 23  
 —, строка 22, 23  
 —, —, угол наклона 21, 23  
**Точка** 25, 55—57, 61—70  
 —, зрения 196, 205  
 —, касания 63, 64, 70, 80, 81, 84  
 —, опорная 165  
 —, пересечения 56, 79, 159, 217, 234  
 —, соединения 113, 116, 119  
 —, текущая 18, 19, 78  
 —, узловая 132, 223

**Треугольник** 26, 41, 63, 223, 225  
 —, описатель 225  
 —, сторона условно видимая 228  
 —, —, невидимая 228  
**Триангуляция** 189, 222

**Удаление (стирание) невидимых линий** 188, 190  
 —, частичное 236  
**Узел параметрический** 138  
 —, сетки 115, 142, 144, 151  
**Уравнения нормальные** 122, 125  
 —, трехдиагональная система 113  
**Условие краевое** 113—119  
**Устройство виртуальное** 12, 13  
 —, графическое 7, 10—12, 15, 18, 88, 112, 254, 258  
 —, нестандартное 254, 258  
 —, подключение 258  
 —, прямого доступа 258—260  
 —, тип 256—262  
 —, хэндлер 11, 13, 254, 258, 259

**Фильтр** 126, 130  
 —, линейный 129, 130  
 —, ряд Фурье 126, 128, 129  
**Функция весовая** 138, 140  
 —, двух переменных 5, 7, 8, 151, 188  
 —, неоднозначная 118  
 —, однозначная 117, 188, 221, 248  
 —, параметрическое задание 117  
 —, периодическая 127

**Центр тяжести** 241, 243

**Числа** 21, 25, 93  
 —, вывод 25

**Ширина разрыва** 236, 244, 247  
**Шкала** 96  
 —, логарифмическая 88, 96, 100—103  
 —, полулогарифмическая 88, 100  
 —, равномерная (обычная) 88, 101  
**Шрифт** 21  
 —, курсив 23  
 —, прямой 23  
**Штриховка** 37, 46, 51, 108—110

**Экран** 8, 10, 36, 47, 189, 200, 211, 228  
 —, дисплей 9, 14, 47  
 —, инициализация 200, 212  
 —, описатель 231  
 —, результирующий 49, 50  
**Экранирование** 8, 10, 19, 36, 42, 47, 52, 162, 228  
**Экстракод** 11, 266  
**Экстремум абсолютный** 163  
 —, локальный 152, 164, 165, 180  
**Элемент** 75—77  
 —, геометрический 26, 55, 73  
 —, графический 16  
 —, пишущий 13  
 —, смена 13  
 —, поверхности 189, 211, 235  
**Эллипс** 18, 32—34, 73, 78  
 —, дуга 18, 32, 33, 46, 73, 78

1 р. 10 к.

