Глава 16. Обработка данных

Если вы ученый, инженер или студент технического вуза, то вам, несомненно, приходится сталкиваться с различными задачами, связанными с обработкой экспериментальных данных. И, наверное, вы не будете спорить с тем, что расчет регрессии или тем паче Фурье-преобразования — это довольно непростая задача, даже при решении ее в одной из систем программирования (что уж говорить о проведении подсчета на бумаге). Если у вас есть опыт обработки данных такими традиционными путями, то вы будете приятно удивлены и поражены, насколько проще и быстрее (и чаще всего — качественнее) можно выполнить эту работу в Mathcad. Все дело в том, что среди встроенных функций программы весьма заметную часть занимают функции, реализующие основные из распространенных в науке и технике алгоритмов. И для того, чтобы, например, построить кривую показательной регрессии для некоторой выборки, вам совсем не нужно помнить сложные и громоздкие методы, вроде метода наибольшего правдоподобия или алгоритма наименьших квадратов. Вам достаточно просто ввести в маркер графической области `expfit(x,y,t)` — и система, просчитав наиболее удачные параметры приближения, построит максимально корректную для ваших данных кривую. Все остальные типы обработки данных реализуются в Mathcad столь же просто.

В зависимости от специфики решаемых задач, все встроенные функции обработки данных можно разделить на несколько больших групп.

- Функции интерполяции. Соединяют точки экспериментальных данных кривыми разной степени гладкости. Кроме того, существует и многомерная интерполяция (практически важным является приближение данных некоторой поверхностью).
- Функции регрессии. Решают задачу определения вида зависимости по экспериментальным данным с учетом погрешности.
- Функции корреляции.
- Функции фильтрации. Предназначены для приближения экспериментальной зависимости при наличии некоторой шумовой компоненты.
- Функции интегральных преобразований (Фурье и волнового). Играют огромную роль в радиотехнике.

Чтобы не быть голословными, приведем пример различных типов обработки данных по отношению к сгенерированной с помощью функции случайного равномерно распределенного числа `rnd(x)` псевдоэкспериментальной линейной зависимости. Оцените, насколько просто в Mathcad решаются такие трудные, в общем, задачи, как интерполирование сплайнами или сглаживание.

Пример 16.1. Типы обработки данных

Задание псевдоэкспериментальной зависимости:

$$i := 0..16$$

$$x_i := 0.25 \cdot i \qquad y_i := 2 \cdot x_i + 2 + rnd(3)$$

Интерполирование кубическим сплайном:

$$s := cspline(x, y)$$

$$sp\_line(z) := interp(s, x, y, z)$$

Вычисление линейной регрессии:

$$reg := line(x, y) \qquad reg = \begin{pmatrix} 3.021 \\ 2.041 \end{pmatrix}$$

$$reg\_line(z) := reg_1 \cdot z + reg_0$$

Определение линейной корреляции данных:

$$corr(x, y) = 94.536\%$$

Сглаживание данных с помощью адаптивного алгоритма:

$$h := supsmooth(x, y) \qquad sm\_line(z) := interp(cspline(x, h), x, h, z)$$

Кривые, построенные на основании различных видов обработки данных, представлены на рис. 16.1.

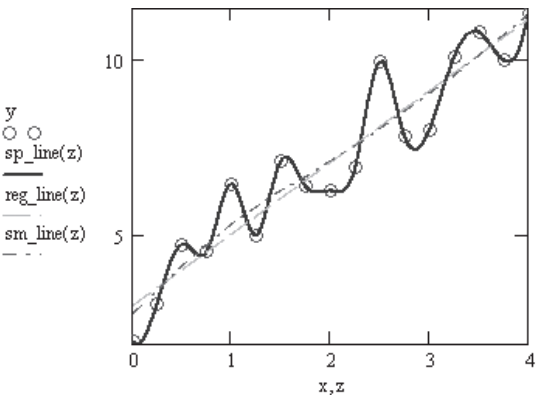


Рис. 16.1. Кривые, построенные на основании разных видов обработки данных

Приведенные примеры обработки данных отнюдь не исчерпывают возможности Mathcad в данной области. Подробному же их освещению и будет посвящена эта глава.

16.1. Интерполяция

На практике часто возникает ситуация, когда известен ряд значений функции, однако описать данную функцию аналитическим выражением невозможно. К примеру,

последовательность значений может быть получена в результате измерения какого-то сложного физического или технического явления, формул для которого не существует. Чтобы результат эксперимента можно было анализировать, через имеющиеся точки нужно провести максимально соответствующую им кривую. При этом станет возможным предсказание значения функции в промежуточных точках, в которых измерения не проводились, можно будет дифференцировать и интегрировать экспериментальную зависимость... В общем, с эмпирической функцией можно будет работать точно так же, как с заданной аналитически. Но как же провести через точки «правильную» кривую? Данная проблема может быть решена с помощью алгоритмов интерполяции. В Mathcad имеется несколько функций интерполяции, позволяющих соединить точки экспериментальных данных кривыми различной степени гладкости. Кроме того, существует возможность и трехмерной интерполяции. Обо всех потенциальных способах соединить экспериментальные точки непрерывной кривой мы и поговорим в этом разделе. Также мы продемонстрируем, как можно оперировать полученной в результате интерполяции функцией.

16.1.1. Линейная интерполяция

Самым простым видом интерполяции, заключающимся в попарном соединении точек отрезками прямых, является линейная интерполяция.

В Mathcad для задания ломаной линейной интерполяции существует специальная встроенная функция linterp(x,y,z), где x — имя вектора координат экспериментальных данных по оси абсцисс, y — имя вектора по оси ординат, z — переменная, от которой будет зависеть интерполирующая функция.

Чтобы построить график линейной интерполяции, выполните следующую последовательность действий.

1. Задайте векторы экспериментальных данных x и y. На практике это делается, как правило, либо их непосредственным внесением в таблицу ввода, либо чтением из Excel-документа или текстового файла. Мы же, проверяя эффективность встроенных алгоритмов Mathcad, будем создавать псевдоэкспериментальные зависимости с эффектом погрешности с помощью одного из генераторов случайных чисел. Например, зададим линейную зависимость с нормально распределенной ошибкой:

$$err := rnorm(10, 0, 0.1)$$

$$i := 0..9$$

$$x_i := 0.3 \cdot i \qquad y_i := (x_i + 3) \cdot (1 + err_i)$$

2. Определите функцию ломаной на основании линейной интерполяции:

$$l\_int(z) := linterp(x, y, z)$$

   Имя переменной, от которой должна зависеть функция, можно задать совершенно произвольным образом. Для того же, чтобы не спутать переменную функций интерполяции с именами векторов экспериментальных данных, во всех примерах данной главы мы будем определять ее как «z».

3. Внесите имя функции интерполяции и переменную в соответствующие маркеры графической области. Если вы хотите, чтобы также отображались и точки данных, то параллельно предыдущему определению пропишите в маркерах и имена соответствующих векторов. Тип графика при этом следует сменить с lines (Линии) на points (Точки).

Результат описанного построения приведен на рис. 16.2.

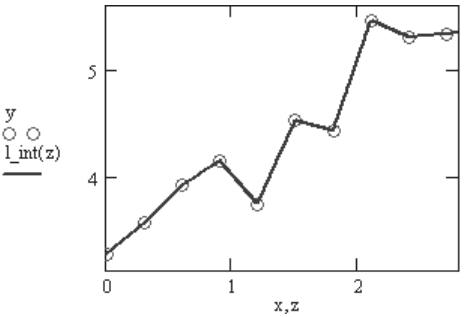


Рис. 16.2. Линейная интерполяция

Если вы попробуете построить график на основании ваших экспериментальных данных без какой-либо их предварительной обработки, то он по виду полностью совпадет с тем, который был бы визуализирован, используй вы функцию linterp. Возникает вопрос: зачем тогда вообще нужна эта функция?

Все дело в том, что по самому своему определению интерполяция служит для приближения закона изменения эмпирической зависимости между известными экспериментальными точками. То есть при ее помощи вы можете не только соединить точки данных линиями, но и вычислить соответствующие значения, которые могут использоваться для предсказания величины зависимости в тех точках, в которых она не была измерена на опыте.

А в общем можно утверждать, что никаких различий при задании графика линейной интерполяции обоими этими способами не существует. Просто при построении по немногочисленным точкам кривой в формате lines (линии) Mathcad осуществляет интерполяцию автоматически, в скрытой для вас форме, а при использовании функции linterp вы активно участвуете в этом процессе.

Принципиальным моментом при задании любой интерполирующей кривой является то, что данные в векторе x должны быть рассортированы в порядке возрастания. Чтобы провести сортировку такого рода, можно использовать специальную матричную функцию sort(x). Но нужно учесть, что точки в векторе y должны соответствовать точкам в векторе x. Поэтому все те же перестановки, что были осуществлены с элементами вектора x, должны быть проделаны и над точками вектора y. Решить данную проблему не так уж и сложно, если обратиться к встроенной функции match(z, A). Данная функция определяет индексы элемента матрицы A, который имеет значение z. Используя функцию match, мы можем узнать, какой индекс получил элемент x_i после сортировки. Имея такую информацию, перемещаем элемент y_i на ту же позицию (см. пример 16.2).

Пример 16.2. Сортировка экспериментальных данных для функций интерполяции

Генерируем псевдоэкспериментальную зависимость, в которой данные не организованы исходя из возрастания аргумента.

$$i := 0..60$$

$$x_i := rnd(-20) + 10 \qquad y_i := \sin(x_i)$$

Сортируем вектор x:

$$x\_sort := sort(x)$$

Сортируем вектор y так, чтобы он соответствовал отсортированному вектору x:

$$matcher(x, x\_sort, y) := \left| \begin{array}{l} \text{for } i \in 0..last(x) \\ \quad \left| \begin{array}{l} ind \leftarrow match\left(x_i, x\_sort\right)_0 \\ y\_sort_{ind} \leftarrow y_i \end{array} \right. \\ y\_sort \end{array} \right. \qquad y\_sort := matcher(x, x\_sort, y)$$

Задаем интерполирующую функцию и вычисляем ее значение в двух промежуточных точках:

$$sn(z) := linterp(x\_sort, y\_sort, z) \qquad sn(-\pi) = -6.696 \times 10^{-4} \qquad sn\left(\frac{\pi}{2}\right) = 0.974$$

На практике использование линейной интерполяции по очевидным причинам весьма ограничено. Гораздо большее значение имеет приближение гладкими полиномиальными сплайнами, о котором мы поговорим в следующем подразделе.

16.1.2. Интерполяция кубическими сплайнами

В том, чтобы соединить экспериментальные точки отрезками прямых, нет никакой сложности: это элементарно можно сделать, и не прибегая к помощи компьютера, используя простую линейку. Гораздо более непростая задача, требующая специальных навыков, связана с приближением данных с помощью гладкой кривой.

Конечно, можно попробовать провести через N+1 точку полином N-й степени. Технически это очень просто реализуется записью системы из N+1 линейного уравнения, коэффициентами которых являются соответствующие степени x-координат точек данных, неизвестными — коэффициенты полинома, правыми частями — y-координаты. Также можно использовать формулу Лагранжа или Ньютона. Однако при приближении полиномом возникают проблемы, связанные с тем, что при большом N интерполирующая кривая проявляет склонность к чрезмерной осцилляции. При этом вид полученного графика, как правило, совершенно не соответствует поставленной задаче максимально корректного приближения экспериментальной зависимости.

Пример 16.3. Случай неудачного интерполирования полиномом (рис. 16.3)

$$x := (0.1 \;\; 0.2 \;\; 0.3 \;\; 0.4 \;\; 0.5 \;\; 0.6 \;\; 0.7 \;\; 0.8)^T \qquad y := (1 \;\; 3 \;\; 2 \;\; -3 \;\; 5 \;\; 3 \;\; 2 \;\; 1)^T$$

$$I(x, y) := \left| \begin{array}{l} \text{for } i \in 0..last(x) \\ \quad \text{for } j \in 0..last(x) \\ \quad\quad M_{i,j} \leftarrow \left(x_i\right)^{last(x)-j} \\ M^{-1} \cdot y \end{array} \right. \qquad Int(z) := \sum_{k=0}^{last(x)} z^k \cdot I(x, y)_{last(x)-k}$$

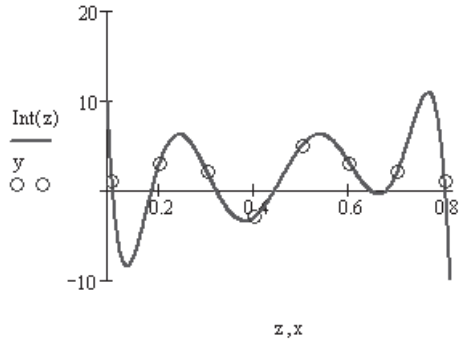


Рис. 16.3. Приближение нельзя считать хорошим из-за наличия не основанных непосредственно на данных экстремумов

Описанные трудности стали причиной того, что ни простые полиномы, ни более совершенные рациональные функции и полиномы Чебышева практически не используются на практике для интерполирования данных. Зато очень широкое распространение в различных системах проектирования и обработки экспериментальной информации получили так называемые сплайны.

Идея сплайн-интерполяции заключается в том, что, вместо того чтобы проводить капризную и непредсказуемую кривую полинома высокой степени через все опытные точки, строится непрерывная кривая из фрагментов полиномов низких порядков. Чаще всего в качестве интерполирующего полинома используется кубическая парабола, индивидуально рассчитываемая для каждого промежутка между точками (сплайн из фрагментов полиномов второй степени может иметь нежелательные изгибы в точках сшивки, что связано с тем, что функция второй производной не будет непрерывной). Достоинством сплайн-интерполяции является то, что при ее использовании на кривой приближения не появляется множественных, никак прямо не связанных со значениями координат соединяемых точек, экстремумов. А это обеспечивает куда большую эффективность приближения интерполирующей кривой вида экспериментального закона, чем при приближении его полиномами любых типов.

В Mathcad для интерполяции данных с помощью кубических сплайнов существует специальная функция interp(s,x,y,t), где s — вектор, содержащий значения вторых производных сплайна в экспериментальных точках, x и y — векторы с соответствующими координатами опытных точек, t — переменная, от которой будет зависеть создаваемая функция.

Наверное, вы задались вопросом, как можно организовать необходимый для функции interp вектор вторых производных. Безусловно, никаких специальных формул или алгоритмов для этого придумывать не стоит: в Mathcad существуют встроенные функции, позволяющие построить сплайн, не задумываясь, какие механизмы за этим стоят.

Таких функций три:

- lspline(x,y) — строит сплайн с линейным продолжением;
- pspline(x,y) — построенная кривая интерполяции будет продолжена после краевых точек параболой;
- cspline(x,y) — строит сплайн с продолжением кубической параболой.

Последовательность действий, которую нужно выполнить для интерполирования некоторой выборки опытных данных кубическими сплайнами, несколько сложнее, чем при линейной интерполяции, и заключается в следующем.

1. Данные организуются в виде отсортированных соразмерных векторов. Например:

$$x := (0.1 \;\; 0.2 \;\; 0.3 \;\; 0.4 \;\; 0.5 \;\; 0.6 \;\; 0.7 \;\; 0.8)^T \qquad y := (1 \;\; 3 \;\; 2 \;\; -3 \;\; 5 \;\; 3 \;\; 2 \;\; 1)^T$$

2. Задается вектор вторых производных с помощью одной из приведенных выше функций семейства *spline. Попробуем сравнить векторы, которые вычислят для наших данных все три эти функции:

s1 := lspline(x, y)

s1 =

	0
0	0
1	3
2	0
3	0
4	-128.203
5	$-1.287 \cdot 10^3$
6	$2.877 \cdot 10^3$
7	$-2.421 \cdot 10^3$
8	805.496
9	-201.374
10	0

s2 := pspline(x, y)

s2 =

	0
0	0
1	3
2	1
3	-101.154
4	-101.154
5	$-1.294 \cdot 10^3$
6	$2.878 \cdot 10^3$
7	$-2.418 \cdot 10^3$
8	794.231
9	-158.846
10	-158.846

s3 := cspline(x, y)

s3 =

	0
0	0
1	3
2	2
3	640.191
4	-300
5	$-1.24 \cdot 10^3$
6	$2.861 \cdot 10^3$
7	$-2.403 \cdot 10^3$
8	750.718
9	$3.903 \cdot 10^{-13}$
10	-750.718

Внимательно изучив полученные векторы, вы обнаружите, что строк в них на три больше, чем имеется точек в выборке. Это обстоятельство связано с тем, что первые три строки вектора являются служебными, и в них содержится информация относительно характеристик строящегося сплайна, необходимая для функции interp. Так, например, во второй строке отображается порядок сплайна, а в третьей — характеристика кривой, продолжающей сплайн за крайними точками (0 для прямой, 1 для параболы и 2 для полинома третьей степени).

Непосредственно значения вторых производных содержатся в строках с 3-й по 10-ю. Сравнив соответствующие элементы различных векторов, вы обнаружите, что они довольно близки по значениям. Заметно отличаются лишь элементы 3-й и 10-й строк.

Серьезные различия вторых производных для крайних точек связаны с особенностями построения кубических сплайнов. Все дело в том, что они задаются, в общем, произвольным образом, и в зависимости от типа решаемой задачи условия для концов промежутка можно определить по-разному. Так, например, если вторая производная в краевой точке задается как 0, то сплайн будет продолжен прямой. Также его можно продлить параболой и полиномом 3-й степени. Во всех этих случаях вид графика будет различен за пределами крайних точек, однако между ними интерполирующие кривые различаться практически не будут.

Принципиальным выбор типа полинома продолжения для сплайна может быть лишь в том случае, если вы его собираетесь использовать в качестве экстраполирующей функции. В основном же на практике интерес представляет качественное соединение именно внутренних точек, поэтому, скорее всего, никакой разницы от того, какую функцию вы задействуете для задания вектора вторых производных, не будет.

3. Задаем функцию interp(s,x,y,t). В нашем случае:

$$I1(z) := interp(lspline(x, y), x, y, z)$$

$$I2(z) := interp(pspline(x, y), x, y, z)$$

$$I3(z) := interp(cspline(x, y), x, y, z)$$

Обратите внимание, что вектор вторых производных может быть задан и непосредственно в скобках функции interp.

4. Прописав в соответствующих маркерах имя функции интерполяции и переменную, строим график (рис. 16.4). Обычно, помимо самой приближающей кривой, на графике отображаются и экспериментальные точки. Сделать это можно, внеся в нужные маркеры через запятую имена векторов данных и поменяв тип кривой на points (точки).

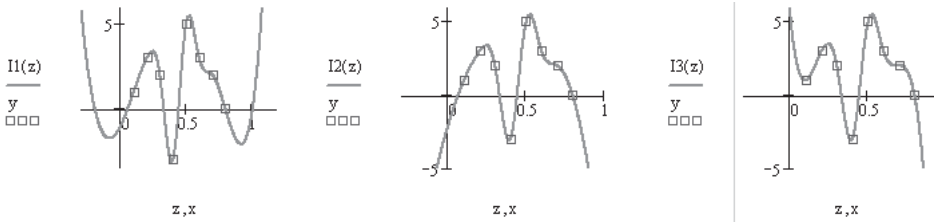


Рис. 16.4. Интерполяция кубическими сплайнами с продолжениями различных типов

На первый взгляд, кривые, приведенные на рис. 16.4, кажутся различными. Однако, присмотревшись, вы увидите, что в области локализации экспериментальных точек они полностью идентичны, и различаются лишь уходящие на бесконечность продолжения кривых сплайна.

В каком случае интерполирующая кривая будет хорошо отображать изучаемую закономерность? Для этого должны выполняться два обязательных требования. Во-первых, погрешность должна отсутствовать или быть небольшой. Если это не так, то перед построением интерполирующего сплайна нужно подвергнуть данные обработке, которая уменьшит влияние погрешности. К примеру, можно задействовать алгоритмы сглаживания. Во-вторых, экспериментальные точки не должны быть слишком далеко друг от друга. Исходя из теоремы Котельникова, дискретизация должна быть в два раза меньше минимального элемента. То есть, если пик экстремума имеет ширину в 10 единиц, то экспериментальные точки нужно получить с шагом в 5 единиц, иначе экстремум может на интерполирующей кривой и не отобразиться.

Функции интерполяции — это чрезвычайно мощный и полезный инструмент. Чтобы это продемонстрировать, решим красивую и сложную задачу — построим бутылку Клейна (см. пример 16.4).

Пример 16.4. Построение бутылки Клейна

Что такое бутылка Клейна? Эта вторая по известности, после листа Мебиуса, односторонняя поверхность. Лист Мебиуса очень просто изготовить из полоски бумаги, склеив ее концы с одной стороны. Также просто строится и график листа Мебиуса (см. разд. 6.2.3). Создать бутылку Клейна (рис. 16.5) намного сложнее. В теории для этого нужно два шага. Первый шаг: берем лист бумаги и склеиваем две его противоположные стороны. В результате получается цилиндр. Шаг второй: склеиваем концы цилиндра. Если склейку осуществить из позиции, когда концы цилиндра направлены в противоположных направлениях, то получится тор. Если же склеить их так, чтобы они были направлены в одну сторону, то выйдет бутылка Клейна. Увы, но в нашем трехмерном мире, чтобы было возможно осуществить второй шаг, требуется вблизи одного из концов цилиндра проделать отверстие, в которое можно будет направить второй конец цилиндра для того, чтобы концы сблизились с одной стороны. Очевидно также, что для того, чтобы второй конец цилиндра смог войти в отверстие вблизи первого конца и затем соединиться с ним, диаметр цилиндра должен изменяться по его длине, сначала увеличиваясь, а затем уменьшаясь. Ввиду описанных ограничений, мы можем построить лишь модель бутылки Клейна. Данная модель будет, по сути, являться проекцией бутылки Клейна в трехмерное пространство. Реальная же поверхность может существовать только в гипотетическом четырехмерном пространстве.

Рис. 16.5. Стеклянная бутылка Клейна (подобные бутылки можно купить на сайте http://www.kleinbottle.com)

Реальная бутылка Клейна — это очень удивительный объект. Во-первых, это односторонняя поверхность. Во-вторых, это непрерывная поверхность, то есть не имеющая границ (у листа Мебиуса границы есть). Эти ее свойства означают, к примеру, что можно начать движение с определенной точки «снаружи» бутылки и оказаться в той же точке, но «внутри». Довольно интересна и трехмерная модель бутылки Клейна. Если отлить ее из стекла так, чтобы точка соединения концов образующего цилиндра соответствовала горлышку, то с ее помощью можно показывать фокус: вода вливается в бутылку, однако вылить ее оттуда простым наклоном сосуда невозможно.

Как можно построить поверхность бутылки Клейна в Mathcad? Когда мы рисовали лист Мебиуса, мы использовали для этого описывающую его систему параметрических уравнений. Суще-

ствует ли аналитическое описание бутылки Клейна? В специальной литературе приводится следующее неявное уравнение:

$$\left(x^2+y^2+z^2+2y-1\right)\cdot\left[\left(x^2+y^2+z^2-2y-1\right)^2-8z^2\right]+16\cdot x\cdot z\cdot\left(x^2+y^2+z^2-2y-1\right)=0$$

Построить поверхность исходя из неявного уравнения довольно проблематично. Чтобы это можно было сделать, не прибегая к использованию сложных и ненадежных алгоритмов, неявное уравнение должно быть преобразовано в систему из трех параметрических уравнений — по одному на каждую координату. Приведенное выше неявное уравнение дает следующую параметрическую систему:

$$x=\cos(u)\cdot\cos\left(\frac{u}{2}\right)\cdot\left(\sqrt{2}+\cos(v)\right)+\sin\left(\frac{u}{2}\right)\cdot\sin(v)\cdot\cos(v)$$

$$y=\sin(u)\cdot\cos\left(\frac{u}{2}\right)\cdot\left(\sqrt{2}+\cos(v)\right)+\sin\left(\frac{u}{2}\right)\cdot\sin(v)\cdot\cos(v)$$

$$z=-1\cdot\sin\left(\frac{u}{2}\right)\cdot\left(\sqrt{2}+\cos(v)\right)+\cos\left(\frac{u}{2}\right)\cdot\sin(v)\cdot\cos(v)$$

Однако если попытаться построить исходя из приведенной системы поверхность, то она окажется совсем не похожей на классическую модель бутылки Клейна вроде изображенной на рис. 16.5. Однако полученная поверхность определенно будет замкнутой и по очертаниям схожей с бутылкой Клейна (рис. 16.6).

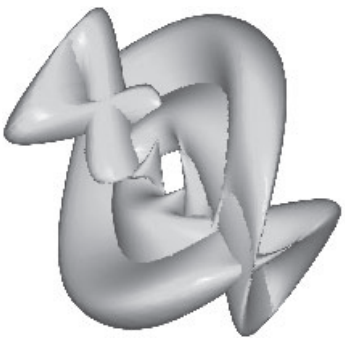
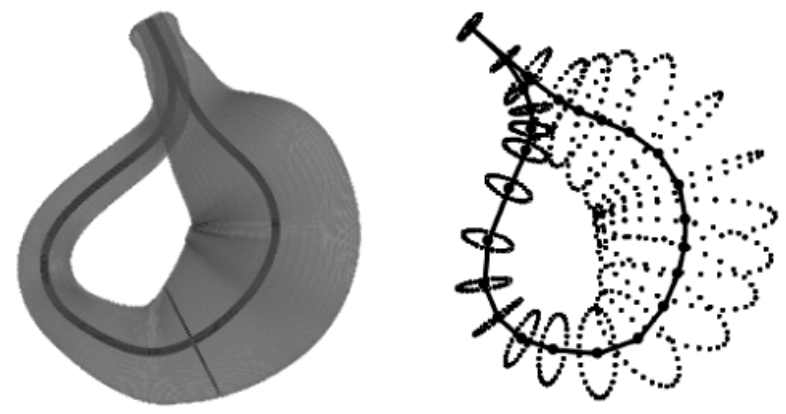


Рис. 16.6. Поверхность, построенная исходя из системы параметрических уравнений для бутылки Клейна

Почему же исходя из системы параметрических уравнений не получается построить «правильную» бутылку Клейна? Дело в том, что реальная бутылка Клейна — это четырехмерный объект. Модель же, показанная на рис. 16.5, умозрительна. В ней есть математически никак не обоснованные допущения (боковое отверстие и переменный диаметр образующего цилиндра), принятые для того, чтобы поведение модели максимально напоминало свойства ее четырехмерного прототипа. Система же параметрических уравнений описывает результат проецирования реальной бутылки Клейна в трехмерное пространство. Возникающие при этом искажения уничтожают часть информации об объекте. Но чисто математически поверхность на рис. 16.6 в большей степени соответствует бутылке Клейна, чем модель на рис. 16.5.

Но нам нужно построить пусть «неправильную», но очень эффектную традиционную модель бутылки Клейна. Сделать это с помощью аналитического описания не получится, так как соответствующих уравнений просто не существует. Тут нужен принципиально другой подход. А что если задать бутылку Клейна по точкам? Не спешите отвергать эту идею: бутылка Клейна обладает свойствами, делающими подобный подход вполне осуществимым на практике.

Что такое бутылка Клейна? Это просто особым образом изогнутый цилиндр переменного диаметра. Чтобы создать такую поверхность, совсем не обязательно знать координаты каждой ее точки. Достаточно, чтобы были известны две функции одного аргумента. Первая функция — это направляющая. Она показывает, как должен изгибаться цилиндр (рис. 16.7, *а*). Если рассечь бутылку Клейна плоскостью, перпендикулярной к касательной к направляющей, проведенной в точке пересечения направляющей и плоскости, то линия пересечения будет всегда представлять собой правильную окружность (рис. 16.7, *б*). Единственное, что нужно знать, чтобы задать точки, лежащие на этой окружности, это пространственное расположение касательной к направляющей (легко определяется из уравнения направляющей) и радиус образующего цилиндра. Так как в трехмерной модели бутылки Клейна радиус цилиндра переменный, необходимо знать функцию, задающую радиус для каждой точки направляющей.



а *б*

Рис. 16.7. Бутылка Клейна в «разрезе» (*а*) и ее точечное изображение (*б*)

Покажем, что функция направляющей и функция радиуса являются функциями от одной переменной. Так как образующий бутылку Клейна цилиндр изгибается так, что каждая точка движется в одной и той же плоскости, то и направляющая будет плоской, а не трехмерной кривой (наиболее удобно ее расположить в плоскости X0Z). Соответственно, задать ее можно функцией от одной переменной. Функция радиуса есть функция от функции направляющей. Поэтому, очевидно, они будут зависеть от одной и той же переменной.

Итак, чтобы построить бутылку Клейна, мы должны для начала задать функцию направляющей и функцию радиуса. Начнем мы с функции направляющей.

В принципе, можно попытаться написать уравнение направляющей. Однако это сложный подход. Проще действовать следующим образом. На миллиметровой бумаге или в графическом редакторе, поле которого может быть отображено в виде сетки, рисуем

направляющую. Это должна быть петля, аналогичная показанной на рис. 16.7, *а*. Ее размеры, соотношение длины и ширины и прочие параметры вы можете выбрать произвольно. Нарисовав направляющую, определяем координаты 10–20 ее точек, поставив в соответствие каждой клетке, например, отрезок длиной 1. Точки необходимо ставить так, чтобы в области быстрого изменения функции их было больше, чем на участке, где кривая изменяется плавно (рис. 16.8).

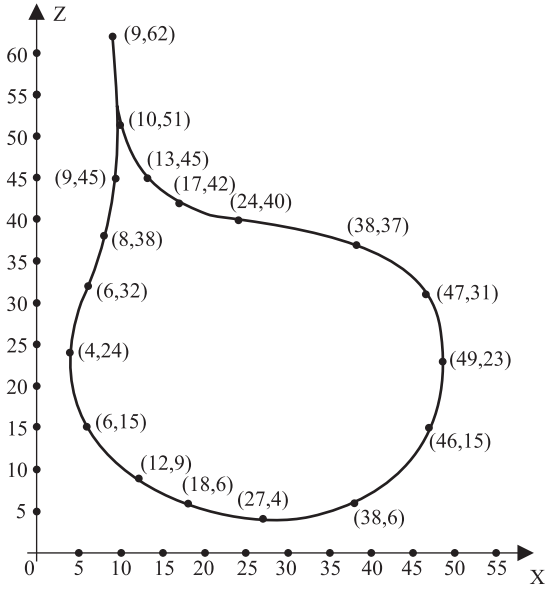


Рис. 16.8. Направляющая и описывающие ее узловые точки

Определив координаты узловых точек направляющей, заносим их в векторы trace_x и trace_z:

$$\text{trace\_x} := (9\ \ 9\ \ 8\ \ 6\ \ 4\ \ 6\ \ 12\ \ 18\ \ 27\ \ 38\ \ 46\ \ 49\ \ 47\ \ 38\ \ 24\ \ 17\ \ 13\ \ 10\ \ 9)^T$$

$$\text{trace\_z} := (62\ \ 45\ \ 38\ \ 32\ \ 24\ \ 15\ \ 9\ \ 6\ \ 4\ \ 6\ \ 15\ \ 23\ \ 31\ \ 37\ \ 40\ \ 42\ \ 45\ \ 51\ \ 62)^T$$

Теперь на основании координат узловых точек посредством интерполяции кубическим сплайном необходимо задать функцию направляющей. Однако это нельзя сделать напрямую, задав функцию z=f(x). Дело в том, что направляющая ведет себя так, что одному значению x соответствует два значения z. Описать такую зависимость явным уравнением z=f(x) невозможно. Она может быть описана или неявным уравнением F(x,y)=0 (к примеру, для окружности радиуса R неявное уравнение имеет вид $x^2+y^2-R^2=0$) или системой из двух параметрических уравнений x=f1(t) и z=f2(t) (для окружности такая система — x=R·sin(t) и y=R·cos(t), t∈0…2·π). Чисто технически проще работать с системой параметрических уравнений. Чтобы задать такую систему для направляющей, нужно провести два кубических сплайна через точки векторов trace_x и trace_z. Выбор параметра для системы нужно осуществить так, чтобы он показывал, как далеко точки отстоят друг от друга. В нашем случае значением параметра для i-й точки будет длина ломаной, последовательно соединяющей все точки от первой до i-й.

Задать вектор параметра очень просто, используя теорему Пифагора:

$$i := 0..\,last(trace\_x)$$

$$length(k) := \sum_{j=1}^{k} \sqrt{\left(trace\_x_{j-1} - trace\_x_j\right)^2 + \left(trace\_z_{j-1} - trace\_z_j\right)^2}$$

$$t_i := if(i = 0, 0, length(i))$$

t^T=(0 17 24.1 30.4 38.6 47.9 56.3 63.1 72.3 83.5 95.5 104 112.3 123.1 137.4 144)

Основываясь на векторе t и векторах trace_x и trace_z, создаем функции tr_x и tr_z, описывающие зависимость значений координат точек направляющей от длины приближающей ее ломаной:

$$tr\_x\_spl := cspline(t, trace\_x) \qquad tr\_z\_spl := cspline(t, trace\_z)$$

$$tr\_x(p) := interp(tr\_x\_spl, t, trace\_x, p)$$

$$tr\_z(p) := interp(tr\_z\_spl, t, trace\_z, p)$$

Проверяем, правильно ли мы все сделали, отобразив на одном графике построенную исходя из функций tr_x и tr_z кривую и исходные узловые точки (рис. 16.9).

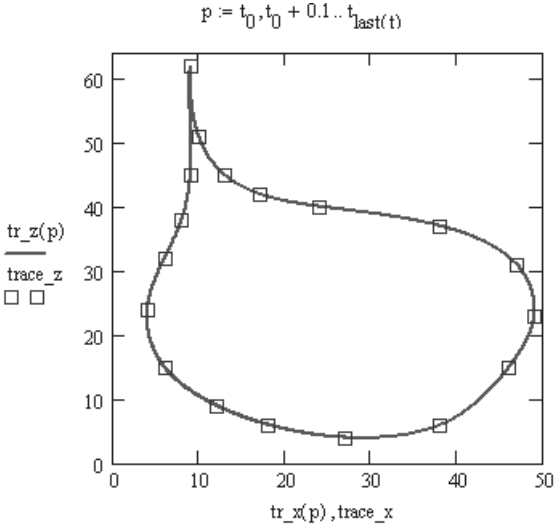


Рис. 16.9. Интерполирующая кривая проходит точно так же, как исходная, нарисованная «вручную» направляющая (сравните с рис. 16.8)

Далее нужно задать функцию, описывающую изменение радиуса образующего бутылку Клейна цилиндра. Решим мы эту задачу по тому же принципу, по которому была задана функция направляющей. Для этого нарисуем сечение поверхности плоскостью X0Z и, чтобы не было путаницы, оставим на чертеже только внешний контур и направляющую (рис. 16.10). Детали формы у бутылки могут быть произвольными. Однако нужно стремиться к тому, чтобы радиус цилиндра не был больше, чем радиус кривизны направляющей в данной точке. Иначе поверхность будет иметь складки. Чтобы бу-

тылка Клейна была похожа на реальную бутылку, в области отверстия должно быть небольшое расширение — горлышко.

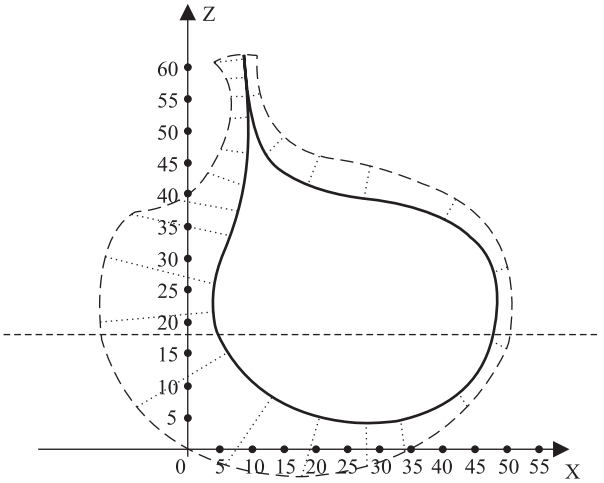


Рис. 16.10. Внешний контур бутылки Клейна с проведенными к нему от направляющей отрезками радиуса

Когда сечение бутылки Клейна будет нарисовано, начертите 10–20 отрезков, соединяющих направляющую и внешний контур. Данные отрезки должны быть перпендикулярны касательной к направляющей. Далее измерьте их длину. Полученные данные используем как узловые точки для функции радиуса. Чтобы можно было определить, какой точке направляющей соответствует какой радиус, необходимо занести в отдельный вектор координаты по X или Z точек направляющей (назовем его bord_z), относительно которых были проведены отрезки.

$$\text{bord\_z} := (62\ 61\ 58\ 46\ 52\ 47\ 42\ 37\ 34\ 26\ 22\ 15\ 8\ 5\ 4\ 5\ 9\ 17\ 28\ 36\ 40\ 41\ 47\ 55\ 62)^T$$

$$r := (3\ \ 10\ \ 5\ \ 5\ \ 5.5\ \ 6\ \ 10\ \ 15\ \ 18\ \ 21\ \ 16\ \ 13\ \ 11\ \ 10\ \ 8\ \ 7\ \ 6\ \ 5\ \ 6\ \ 4\ \ 4\ \ 4\ \ 3\ \ 3\ \ 3)^T$$

Функция радиуса должна зависеть от той же переменной, что и функция направляющей. Однако у нас есть данные относительно того, как радиус меняется по мере изменения z-координаты точек направляющей, но не относительно того, как связаны радиус образующего цилиндра и параметр t. Чтобы найти эту связь, требуется определить, каким значениям t соответствуют имеющиеся значения z-координаты. Решить эту задачу не так уж и просто — но можно. Для этого нужно для каждой имеющейся величины z-координаты Z_i искать корень уравнения $tr\_z(t)-Z_i=0$, где tr_z(t) — функция, связывающая координаты по Z направляющей и параметр t. Данное уравнение нельзя решить аналитически, но можно численно, используя функцию root. Подумаем, как чисто технически лучше реализовать соответствующий алгоритм.

В первую очередь нужно решить, численные методы какого типа лучше использовать — типа метода Больцано или типа метода секущих. Конечно, методы интервалов локализации корня надежнее. Но как этот самый интервал определить так, чтобы на нем оказался нужный корень, причем, только один? Если мы решаем уравнение «вручную», то это легко сделать, ориентируясь по графику. Если же поиском корней занимается

программа, то достоверно определить интервал локализации корня сложно. Поэтому мы будем использовать метод секущих. Однако для этого придется преодолеть массу сложностей. Во-первых, каким должно быть начальное приближение, чтобы численный метод сошелся к нужному корню? Как можно определить, что найденный корень верен? Что делать, если численный метод не сойдется к решению?

У решаемой задачи есть особенность, дающая возможность справиться с перечисленными сложностями. Нам известно, в каком интервале изменяется параметр t. Также известно, что для Z_{i+1} значение параметра будет строго больше, чем для Z_i. Поэтому мы можем разделить интервал изменения параметра t на N фрагментов, где N — количество узловых точек для функции радиуса. Будем предполагать, что решение уравнения $tr\_z(t)-Z_i=0$ локализовано (или находится вблизи) i-го интервала. Если отрезки радиуса проводились относительно равномерно, то данное предположение грубым не является. В качестве приближения к корню можно взять любую точку соответствующего Z_i интервала. Однако, чтобы уменьшить вероятность определения не того корня (а также чтобы предусмотреть возможность отсутствия сходимости к корню), будем решать уравнение $tr\_z(t)-Z_i=0$ в десяти точках i-го интервала. Получив очередное решение, будем проверять, принадлежит ли оно области изменения t. Если да, то оно должно быть занесено в вектор корней S.

Когда все интервалы будут просмотрены, вектор корней следует отсортировать. Эта необходимость связана с тем, что то, что точка начального приближения берется на i-м интервале, еще не гарантирует, что в качестве решения будет найдено именно соответствующее Z_i значение параметра. Отсортировав вектор корней, из него следует исключить повторяющиеся значения. Это легко сделать, сравнивая в цикле значение S_j с S_{j-1}. Если они окажутся разными, S_j заносится в вектор ответа T. Если же S_j с S_{j-1} равны, то просто переходим на следующую итерацию, ничего не занося в T.

Описанный выше алгоритм можно реализовать следующим кодом:

$$
\begin{aligned}
&\text{converter}(Z) := \\
&\quad \left(n \leftarrow 0 \quad TOL \leftarrow 10^{-14}\right) \\
&\quad \text{for } i \in 0..\text{last}(Z)\cdot 10 \\
&\qquad p \leftarrow t_{\text{last}(t)}\cdot\frac{i}{\text{last}(Z)\cdot 10} \\
&\qquad \text{continue} \quad \text{on error root}\left(tr\_z(p) - Z_{\text{round}\left(\frac{i}{10}\right)}, p\right) \\
&\qquad sol \leftarrow \text{root}\left(tr\_z(p) - Z_{\text{round}\left(\frac{i}{10}\right)}, p\right) \\
&\qquad \left(S_n \leftarrow sol \quad n \leftarrow n+1\right) \text{ if } sol < t_{\text{last}(t)} \wedge sol > 0 \\
&\quad \left(S \leftarrow \text{sort}(S) \quad k \leftarrow 1 \quad T_0 \leftarrow S_0\right) \\
&\quad \text{for } j \in 1..\text{last}(S) \\
&\qquad \left(T_k \leftarrow S_j \quad k \leftarrow k+1\right) \text{ if } S_j \neq S_{j-1} \\
&\quad T
\end{aligned}
$$

Обратите внимание на то, что функцию root можно вполне эффективно использовать в коде программ, задавая начальное приближение в форме локальной переменной. Также как локальную переменную можно задать и TOL.

Применим программу converter по отношению к вектору bord_z:

$$t\_bord := converter(bord\_z)$$

$$t\_bord^T = (0\ \ 1\ \ 4\ \ 10\ \ 15\ \ 16\ \ 20\ \ 25\ \ 28\ \ 37\ \ 41\ \ 48\ \ 76\ \ 81\ \ 87\ \ 88\ \ 95\ \ 98\ \ 103\ \ 109$$
$$120\ \ 137\ \ 142\ \ 152\ \ 160)$$

При определении радиусов образующего цилиндра производилось округление до целых чисел. Из-за этого в полученных значениях имеется существенная погрешность. Также нельзя гарантировать то, что отрезки радиусов были отложены под углом, мало отличающимся от истинного угла наклона перпендикуляра к касательной к направляющей. Чтобы минимизировать влияние ошибки в данных вектора r на форму поверхности, данные следует обработать одним из алгоритмов сглаживания. Мы воспользуемся для этого встроенной функцией Mathcad supsmooth. Однако нужно учесть, что сглаживание меняет величины значений исходя из величин значений по соседству. Поэтому после сглаживания значения первого и последнего элементов вектора r могут и не совпасть. Это не допустимо, так как не соответствует свойствам бутылки Клейна. Чтобы исправить эту ошибку, просто присваиваем последнему элементу вектора r значение первого элемента.

$$r := supsmooth(t\_bord, r)$$
$$r_0 := r_{last(r)}$$

На основании векторов t_bord и r посредством интерполяции кубическим сплайном создаем функцию радиуса образующего бутылку Клейна цилиндра. Назовем ее bord_f:

$$bord\_spl := cspline(t\_bord, r)$$

$$bord\_f(p) := interp(bord\_spl, t\_bord, r, p)$$

Строим график bord_f (рис. 16.11).

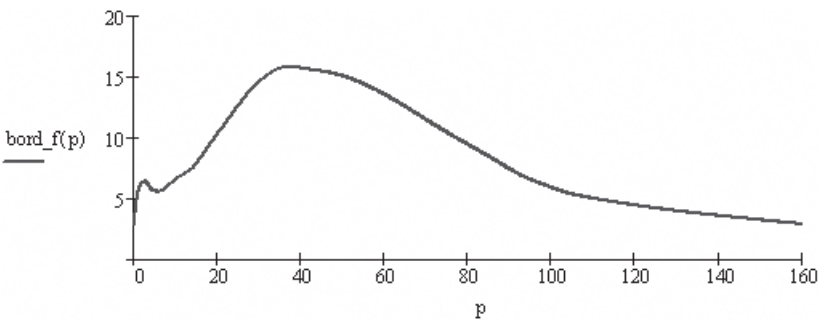


Рис. 16.11. График функции изменения радиуса бутылки Клейна

Необходимые для построения бутылки Клейна функции у нас уже есть. Осталось произвести само построение. Рисовать бутылку Клейна мы будем следующим образом. Перейдя к очередной точке направляющей, мы будем определять, под каким углом

к оси X должна располагаться плоскость с соответствующими данной точке точками образующего цилиндра. Затем мы будем располагать в этой плоскости сами точки по окружности, радиус которой задаст функция bord_f. Как видите, идейно все просто. Однако техническая реализация описанного алгоритма содержит немало тонких моментов.

Плоскость, в которой располагаются точки бутылки Клейна, соответствующие одной точке направляющей, перпендикулярна к касательной к направляющей, проведенной в точке пересечения плоскости и направляющей. Следовательно, угол наклона a плоскости к оси X будет отличаться от угла наклона касательной к оси X f на $\pi/2$. Конкретное выражение — $\alpha=\pi/2-\phi$ (его можно получить, решив элементарную геометрическую задачку). Мы можем довольно просто определить угол наклона касательной f, так как известно, что его тангенс равен производной функции в точке, к которой касательная была проведена. Тангенс же угла наклона плоскости определится как $tg(\alpha)=tg(\pi/2-\phi)=ctg(\phi)=1/tg(\phi)=1/f'(t)$, где $f'(t)$ — производная функции направляющей. Однако как рассчитать значение производной функции направляющей, если она задается системой параметрических уравнений x=tr_x(t) и z=tr_z(t)? Легко доказать (попробуйте это сделать), что если функция f(t) описывается двумя параметрическими уравнениями x(t) и y(t), то ее производную можно вычислить как $f'(t)=y'(t)/x'(t)$. С учетом данной зависимости угол наклона α плоскости к оси X можно вычислить как $\alpha=arctg(tr\_x'(t)/tr\_z'(t))$. Зададим на основании полученной формулы функцию:

$$\alpha(t) := \operatorname{atan}\left(\frac{\frac{d}{dt}tr\_x(t)}{\frac{d}{dt}tr\_z(t)}\right)$$

То, как вычислить угол, под которым должна лежать плоскость с точками поверхности, соответствующими некоторой точке направляющей, мы разобрались. Однако как можно чисто технически расположить точки по окружности определенного радиуса с центром в произвольной точке пространства и наклоненной к оси X под произвольным углом? Чтобы понять, как это можно сделать, разберемся с таким важнейшим понятием компьютерной геометрии, как матрицы преобразований.

Матрицы преобразований позволяют осуществлять над системой координат, к которой относится точка, такие модификации, как перенос, поворот, масштабирование. В общем случае подобные модификации описываются системами линейных уравнений, получаемых при решении соответствующих геометрических задач. Для примера приведем систему, определяющую поворот системы координат вокруг оси X на угол θ:

$$x' = x$$
$$y' = y\cdot\cos(\theta) - z\cdot\sin(\theta)$$
$$z' = y\cdot\sin(\theta) + z\cdot\cos(\theta)$$

Здесь: x, y, z — координаты точки до поворота системы координат, x', y', z' — ее новые координаты.

Если над системой координат должно быть проведено только одно преобразование, то вычисления по формулам вполне приемлемы. Однако если нужно выполнить одновременно несколько преобразований (а так оно чаще всего и бывает), то формулы перехода от одной системы к другой становятся крайне громоздкими. Выходом является использование матричной формы представления формул.

В общем виде систему уравнений, осуществляющую переход между двумя системами координат, можно записать как:

$$x' = a \cdot x + b \cdot y + c \cdot z + p1$$

$$y' = d \cdot x + e \cdot y + f \cdot z + p2$$

$$z' = h \cdot x + g \cdot y + i \cdot z + p3$$

Здесь a , b, c, d, e, f, h, g, i – некоторые коэффициенты; p1, p2, p3 – свободные члены.

В матричном виде данную систему можно представить следующим образом:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a & b & c & p1 \\ d & e & f & p2 \\ h & g & i & p3 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Матрица 3×4 в приведенном уравнении — это и есть матрица преобразований для трехмерной системы координат. Конкретный ее вид для каждого преобразования можно получить, переведя в матричную форму описывающую его систему уравнений. Например, матрицу поворота вокруг оси X на угол θ элементарно вывести из рассмотренной выше системы:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \end{pmatrix}$$

Если необходимо, чтобы одна матрица описывала сразу несколько преобразований, то она может быть задана простым перемножением матриц, отвечающих за каждую трансформацию. Однако, чтобы это было возможно, матрицы преобразований должны быть квадратными (то есть количество их строк должно совпадать с количеством столбцов). Для выполнения этого условия они дополняются еще одной строкой. Эта строка выбирается так, чтобы ее наличие никак не повлияло непосредственно на задаваемое преобразование. В общем виде тогда обобщенное уравнение преобразования системы координат запишется как:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & p1 \\ d & e & f & p2 \\ h & g & i & p3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

В качестве примера приведем матрицу, описывающую одновременно поворот системы координат вокруг оси X на угол θ и смещение ее точки отсчета в точку (x1, y1, z1):

$$\begin{pmatrix} 1 & 0 & 0 & x1 \\ 0 & 1 & 0 & y1 \\ 0 & 0 & 1 & z1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & x1 \\ 0 & \cos(\theta) & -\sin(\theta) & y1 \\ 0 & \sin(\theta) & \cos(\theta) & z1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матричное умножение некоммутативно (AB≠BA). Это означает, что порядок, в котором вычисляется произведение матриц, имеет принципиальное значение. Неверное

задание последовательности перемножения матриц преобразований может дать в результате либо полное отсутствие одной из трансформаций, либо то, что она будет проведена иначе, чем задумывалось (например, поворот осуществится не по часовой стрелке, а против нее). Чтобы избежать подобных ситуаций, матрицы преобразований нужно перемножать в следующей последовательности: перенос, масштабирование, поворот.

Матрицы преобразований можно найти в специальной литературе, которая есть далеко не у всех. Поэтому имеет смысл привести их. Те матрицы, которые мы не используем при построении бутылки Клейна, будут вам полезны, если вы решите самостоятельно нарисовать какую-нибудь замысловатую поверхность.

Матрица переноса точки отсчета системы координат из (0, 0, 0) в (X, Y, Z):

$$\begin{pmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица изменения масштаба оси X в scX раз, оси Y — в scY раз, оси Z — в scZ раз:

$$\begin{pmatrix} scX & 0 & 0 & 0 \\ 0 & scY & 0 & 0 \\ 0 & 0 & scZ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрицы поворота на угол f относительно оси X (rotX), Y (rotY), Z (rotZ):

$$rotX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad rotY = \begin{pmatrix} \cos(\phi) & 0 & \sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad rotZ = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Зная про матрицы трансформаций, можно, наконец, написать задающую бутылку Клейна функцию:

$$F(a,t) := \begin{pmatrix} 1 & 0 & 0 & tr\_x(t) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & tr\_z(t) \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\alpha(t)) & 0 & \sin(\alpha(t)) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha(t)) & 0 & \cos(\alpha(t)) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} bord\_f(t)\cdot\cos(a) \\ bord\_f(t)\cdot\sin(a) \\ 0 \\ 1 \end{pmatrix}$$

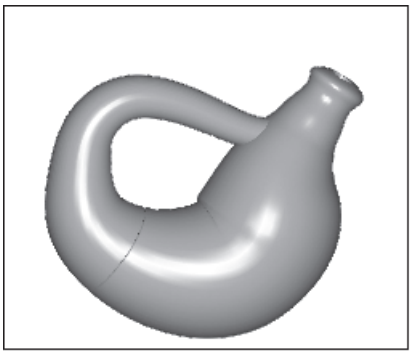
Смысл функции F(a, t) довольно прозрачен. Она сначала рисует окружность в плоскости X0Y с центром в начале координат. Для этого используется система параметрических уравнений x=R·cos(a) и y=R·sin(a) (см. правую матрицу в выражении F(a,t)). Угол a изменяется от 0 до 2·π. Радиус окружности определяется значением функции bord_f при текущем значении t. Когда окружность будет нарисована, она поворачивается относительно оси Y на угол, возвращаемый для данного t функцией α. Поворот осуществляется с помощью соответствующей матрицы трансформаций (средняя матрица в выражении для F(a,t)). Затем осуществляется перенос окружности таким образом,

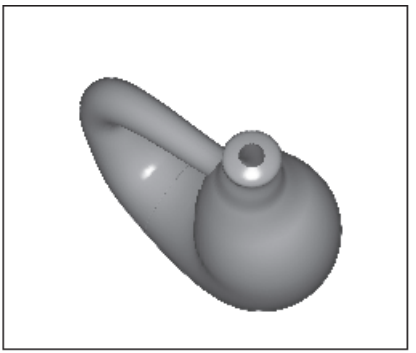
чтобы ее центр совпал с точкой направляющей, определяемой текущим значением t. Для этого используется матрица переноса (левая матрица в выражении для F(a,t)).

При вычислении значения F(a,t) в приведенной форме результатом будет вектор из четырех элементов. Четвертый элемент — это число 1, не несущее никакой информации. Поэтому его нужно вырезать (если этого не сделать, то исходя из F(a,t) не получится построить поверхность).

$$F_n(a,t) := \text{submatrix}(F(a,t),0,2,0,0)$$

Вот теперь действительно все готово для построения бутылки Клейна (рис. 16.12).





$\text{CreateMesh}\left(F_n,0,2\pi,0,t_{\text{last}(t)},50,1000\right)$ $\text{CreateMesh}\left(F_n,0,2\pi,0,t_{\text{last}(t)},50,1000\right)$

Рис. 16.12. Бутылка Клейна в двух ракурсах

16.1.3. Интерполирование B-сплайнами

Трудно спорить с тем, что интерполирование экспериментальной зависимости кубическими сплайнами является наиболее эффективным и в то же время простым методом соединения точек гладкой кривой. Однако в некоторых случаях традиционный алгоритм построения сплайна с помощью фрагментов кубических парабол со сшивкой в точках опытных данных может быть непригодным для решения поставленных задач. Если обычный кубический сплайн по тем или иным причинам будет не очень хорошо, как вам покажется, интерполировать данные, то попробуйте использовать возможности приближения B-сплайнами Mathcad.

Для построения кривой B-сплайна в программе Mathcad имеется специальная функция bspline(x,y,u,n). В общем, она предназначена для того же, что и остальные функции семейства *spline, а именно: для вычисления матрицы коэффициентов создаваемого сплайна, используемой в качестве параметра s функции interp(s,x,y,t).

Основным принципиальным отличием приближения с помощью B-сплайнов от простой кубической сплайн-интерполяции является то, что вы самостоятельно можете определить, из фрагментов графиков полиномов какого порядка будет составлена ваша интерполирующая кривая. Сделать это можно, задав соответствующим образом величину параметра n функции bspline(x,y,u,n). Так, если вы хотите приблизить опытную зависимость ломаной линией, то определите n как 1. Если вы считаете, что опытный закон скорее параболический, положите n=2. И, наконец, если вы хотите провести через точки данных кубический сплайн, то n должно быть задано как 3.

Вторым отличием, существенно расширяющим возможности B-сплайнов по сравнению с обычными кубическими, является то, что вы самостоятельно произвольным образом можете определять точки, в которых будет произведена сшивка различных фрагментов кривой. Как вы помните, при построении простого кубического сплайна, фрагменты кривых сшивались в самих экспериментальных точках. А это не всегда оправданно, хотя бы в связи с тем, что при этом точки данных очень часто оказываются точками перегиба интерполирующей кривой, что значительно снижает ее корректность в случае использования сплайна в качестве функции прогноза. Иногда бывает гораздо эффективнее сшить элементарные сплайны где-нибудь между экспериментальными точками: при этом результирующая кривая куда лучше отображает реальные процессы (что очень важно в том же моделировании).

Чтобы провести сплайн со сшивками в точках, отличных от опытных, вам нужно будет задать вектор, содержащий их координаты по оси X. При этом точки сшивки должны быть рассортированы в порядке возрастания, и значение первой из них должно быть меньше либо равно величине соответствующей координаты левой крайней точки экспериментальной выборки (и соответственно последняя сшивка должна быть расположена правее ее крайней правой точки (или совпадать с нею)). Если вам нужно построить линейный B-сплайн, то сшивок должно быть задано столько же, сколько имеется точек в векторах данных. При приближении зависимости фрагментами парабол, в векторе координат сшивок их должно быть на одну меньше, чем имеется экспериментальных точек. И на целых две точки меньше, чем имеется элементов в выборке, придется определить в рассматриваемом векторе сшивок в том случае, если используемые сплайны — кубические. Кстати, продолжением кубического B-сплайна за пределы области приближаемых данных будет именно кубическая парабола (как вы помните, в зависимости от задания краевых условий, экстраполяция за пределами экспериментальных точек может быть проведена, в общем случае, полиномами 1–3-й степеней).

Так как интерполирование с помощью B-сплайнов является технически более непростой задачей, чем построение кривой приближения с использованием обычных функций кубического сплайна, то рассмотрим его более обстоятельно.

1. Сначала нужно организовать опытные данные в виде двух векторов (элементы вектора с координатами по X должны быть отсортированы по возрастанию). Чтобы существовала возможность сравнения, используем для построения сплайна те же данные, что и в подразд. 16.1.2:

$$x := (0.1 \;\; 0.2 \;\; 0.3 \;\; 0.4 \;\; 0.5 \;\; 0.6 \;\; 0.7 \;\; 0.8)^T \qquad y := (1 \;\; 3 \;\; 2 \;\; -3 \;\; 5 \;\; 3 \;\; 2 \;\; 0)^T$$

2. Задаем вектор сшивок элементарных сплайнов. Как было указано выше, это следует сделать с учетом порядков приближающих полиномов. Мы же попробуем определить векторы сшивки для сплайнов всех трех типов (сравните количество элементов векторов):

$$u1 := (0 \;\; 0.27 \;\; 0.32 \;\; 0.41 \;\; 0.55 \;\; 0.67 \;\; 0.72 \;\; 0.81)^T$$

$$u2 := (0.09 \;\; 0.25 \;\; 0.35 \;\; 0.45 \;\; 0.55 \;\; 0.65 \;\; 0.87)^T$$

$$u3 := (0 \;\; 0.23 \;\; 0.35 \;\; 0.45 \;\; 0.67 \;\; 0.9)^T$$

3. Задаем параметры будущей сплайн-кривой с помощью функции bspline(x,y,u,n), где x и y — векторы опытных данных, u — вектор сшивок, n — порядок используемых полиномов. В нашем случае:

$B1 := bspline(x, y, u1, 1)$ $B2 := bspline(x, y, u2, 2)$ $B3 := bspline(x, y, u3, 3)$

Как было показано в предыдущем подразделе, результатом работы остальных функций семейства *spline является вектор, первые три строки которого содержат служебную информацию для функции interp, а все нижележащие являются значениями вторых производных сплайна в точках данных. Результат же, который возвращает функция B-сплайна, гораздо сложнее, так как он является не вектором, а вложенным массивом. Этот массив содержит три элемента.

Первые элементы при задании сплайн-кривой любой степени будут одинаковыми: это цифра 4. Она служит для того, чтобы функция interp могла определить, что в данном случае нужно использовать алгоритм соединения, предназначенный для B-сплайнов.

$$B1_0 = 4 \qquad B2_0 = 4 \qquad B3_0 = 4$$

Во втором элементе вложенного вектора, вычисляемого функцией bspline, расположены коэффициенты полиномов, приближающих зависимость на каждом из промежутков. Очевидно, что таких интервалов будет на один меньше, чем было задано точек в векторе сшивки. Каждому из элементарных сплайнов в матрице коэффициентов соответствует отдельный столбец. Количество же ее строк определяется тем, полиномы какой степени используются при построении интерполирующей кривой. Так, уравнение кубической параболы содержит четыре коэффициента, следовательно, в рассматриваемой матрице будет четыре строки. Нетрудно догадаться, что при интерполировании квадратичной параболой и прямой в ней будет соответственно три и две строки. Чтобы подтвердить сказанное выше, приведем рассчитанные по нашим данным матрицы коэффициентов для всех трех типов функции bspline:

$$B1_1 = \begin{pmatrix} -1 & 26 & 14 & -41.806 & 83.167 & -192.898 & 75.685 \\ 20 & -80 & -42.5 & 93.611 & -133.611 & 278.426 & -94.606 \end{pmatrix}$$

$$B2_1 = \begin{pmatrix} -3.202 & -23.152 & 162.455 & -236.818 & 100.515 & -34.343 \\ 53.03 & 212.626 & -847.98 & 926.566 & -300.101 & 114.848 \\ -110.101 & -429.293 & 1.086 \times 10^3 & -885.859 & 229.293 & -89.899 \end{pmatrix}$$

$$B3_1 = \begin{pmatrix} 154.963 & -399.584 & 776.279 & -416.027 & 1.691 \times 10^3 \\ -2.901 \times 10^3 & 4.332 \times 10^3 & -5.747 \times 10^3 & 2.202 \times 10^3 & -7.233 \times 10^3 \\ 1.653 \times 10^4 & -1.492 \times 10^4 & 1.388 \times 10^4 & -3.788 \times 10^3 & 1.029 \times 10^4 \\ -2.91 \times 10^4 & 1.648 \times 10^4 & -1.095 \times 10^4 & 2.137 \times 10^3 & -4.869 \times 10^3 \end{pmatrix}$$

В третьей строке вложенного вектора, вычисляемого функцией bspline, располагается матрица, содержащая значения границ всех элементарных сплайнов. Заполняется эта матрица исходя из определенного вами вектора сшивок, и в каждом ее столбце содержатся крайние точки для соответствующего полинома. Так как левая граница одного фрагмента полинома всегда является правой для другого (в силу

непрерывности сплайна), то диагональные элементы рассматриваемой матрицы совпадают.

Приведем пример матрицы границ элементарных сплайнов для случая интерполирования кубическими параболами:

$$B3_2 = \begin{pmatrix} 0 & 0.23 & 0.35 & 0.45 & 0.67 \\ 0.23 & 0.35 & 0.45 & 0.67 & 0.9 \end{pmatrix}$$

Обратите внимание, что, несмотря на то, что точек в выборке восемь, мы определили границы лишь пяти интервалов. Поразмышляйте, вспомнив то, о чем мы говорили в прошлом подразделе, почему их на два меньше, чем, казалось бы, должно быть.

4. Задаем функции сплайнов с помощью все той же встроенной функции interp(s,x,y,t):

$$A1(z) := \text{interp}(B1, x, y, z) \qquad A2(z) := \text{interp}(B1, x, y, z) \qquad A3(z) := \text{interp}(B1, x, y, z)$$

5. Строим графики полученных интерполирующих сплайнов (рис. 16.13). При использовании метода B-сплайнов, для большей корректности на графике стоит отобразить не только экспериментальные точки, но и точки, в которых была произведена сшивка фрагментов приближающих полиномов. Сделать это можно очень просто, внеся в нижний маркер графической области имя вектора сшивок, а в правый — результат вычисления для этого вектора значений функции сплайна.

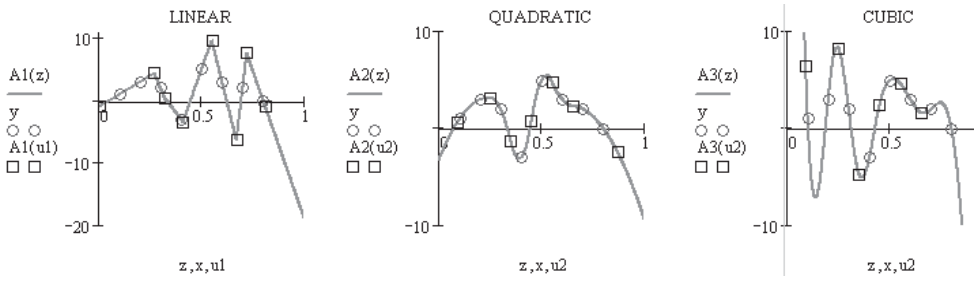


Рис. 16.13. Интерполирование B-сплайнами различных порядков

Понять принципиальное различие при использовании B-сплайнов и более простых методов интерполирования можно, сравнив график Linear рис. 16.13 с рис. 16.2. С первого взгляда в обоих случаях вы, наверное, не заметите особой разницы в ломаных, построенных по двум различным методикам. Однако, изучив их более внимательно, вы увидите, что при применении функции linterp переломы совпадают с точками данных, а при приближении линейным B-сплайном они располагаются в точках сшивок. Аналогичная ситуация наблюдается и при интерполяции гладкими кривыми, правда, при этом нужно говорить не о переломах, а о перегибах.

Какой тип интерполяции лучше использовать — на этот вопрос вы должны ответить исходя из собственного опыта и типа решаемой задачи. Однако наиболее общий совет все же можно дать: сначала попробуйте применить функции простого кубического сплайна. В том же случае, если результат его работы вас не устроит, попытайтесь поэкспериментировать с порядком используемых полиномов и точками их сшивок с помощью соответствующих функций B-сплайна.

16.1.4. Двумерная сплайн-интерполяция

Если вы читали гл. 6, то представляете, какие впечатляющие возможности открывает Mathcad в области построения трехмерных графиков. И возможности эти, равно как и аналогичные функции других математических программ, все более и более активно применяются на практике не только для визуализации каких-то теоретических выкладок, но и для наглядного представления результатов всевозможных научных и инженерных измерений (например, в дефектоскопии используются поверхности, отображающие результаты сканирования магнитного поля на поверхности детали). Если же в такого рода задачах сразу, без какой-либо предварительной обработки данных, задавать поверхность, то из-за погрешности, и довольно большой, она, скорее всего, получится совершенно не гладкой и как бы составленной из отдельных ромбов. Естественно, что такие графики некорректны, и их требуется каким-то образом сглаживать.

Построить в Mathcad на основании экспериментальных данных гладкую и ровную поверхность можно очень просто, задействовав возможности многомерной интерполяции. В общем случае для этого система использует ту же идею о кубических сплайнах, о которой мы говорили в предыдущих подразделах. А так как никаких особых отличий в технике интерполирования сплайнами между двумерным и трехмерным случаем не существует, то отдельно рассматривать механизмы, лежащие в основе задания приближающей поверхности, мы не будем.

Поскольку при интерполяции трехмерных зависимостей существует множество тонких технических деталей, то более глубоко можно изучить этот вопрос, решая какую-нибудь конкретную задачу. Например, промоделируем процесс измерения датчиком некоторой величины в узлах квадратной сетки с равным шагом по каждому из направлений (на практике измерения производятся практически исключительно с такими характеристиками сетки разбиений).

Если вы читали гл. 6, то, наверное, согласитесь, что наиболее простым способом создания в Mathcad матрицы координат для узлов сетки, на основе которой будет построена поверхность, является использование двух ранжированных переменных. В нашем случае мы зададим квадратную сетку от –5 до 5 с равной величиной шага по обеим переменным с таким расчетом, чтобы всего в ней было 25 узлов:

$$j := 0..4 \qquad i := 0..4$$

$$X_{i,j} := -5 + 5\cdot\frac{i}{2} \qquad Y_{i,j} := -5 + 5\cdot\frac{j}{2}$$

$$X = \begin{pmatrix} -5 & -5 & -5 & -5 & -5 \\ -2.5 & -2.5 & -2.5 & -2.5 & -2.5 \\ 0 & 0 & 0 & 0 & 0 \\ 2.5 & 2.5 & 2.5 & 2.5 & 2.5 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix} \qquad Y = \begin{pmatrix} -5 & -2.5 & 0 & 2.5 & 5 \\ -5 & -2.5 & 0 & 2.5 & 5 \\ -5 & -2.5 & 0 & 2.5 & 5 \\ -5 & -2.5 & 0 & 2.5 & 5 \\ -5 & -2.5 & 0 & 2.5 & 5 \end{pmatrix}$$

Далее требуется задать матрицу, содержащую значения измеряемой величины в узловых точках. Для этого определяем закон, по которому, в зависимости от X и Y, изменяется некоторая характеристика. Фактор погрешности можно промоделировать, задействовав один из генераторов случайных чисел. В нашем случае это будет генератор

равномерно распределенных величин rnd(x). Особое внимание обратите на использование индексов ранжированных переменных в рассматриваемом определении:

$$Z_{i,j} := \left(X_{i,j}\right)^2 + \left(Y_{i,j}\right)^2 + \left(X_{i,j}\right)^2 \cdot \mathrm{rnd}(0.01) - \left(Y_{i,j}\right)^2 \cdot \mathrm{rnd}(0.01)$$

$$Z = \begin{pmatrix} 49.897 & 31.347 & 25.086 & 31.487 & 49.984 \\ 31.117 & 12.501 & 6.265 & 12.503 & 31.053 \\ 24.852 & 6.222 & 0 & 6.211 & 24.861 \\ 31.029 & 12.527 & 6.271 & 12.484 & 31.08 \\ 49.907 & 31.401 & 25.163 & 31.467 & 50.112 \end{pmatrix}$$

Построим по заданным матрицам координат поверхность. В общем случае для этого нужно объединить их в один массив, что проще всего можно сделать, прописав их имена через запятую в круглых скобках в маркере графической области (рис. 16.14).

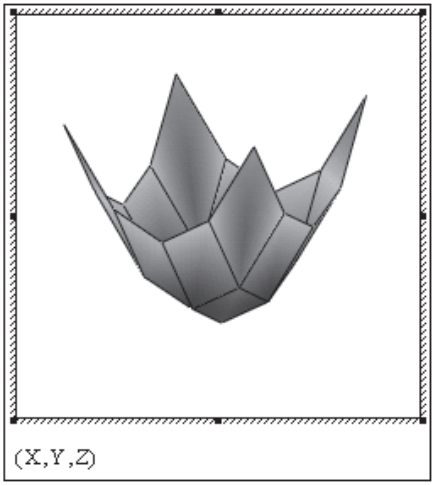


Рис. 16.14. Поверхность псевдоэкспериментальной зависимости

Рассмотрев график на рис. 16.14, вы, наверное, согласитесь, что использовать его без приближения некоторой гладкой поверхностью вряд ли целесообразно. Значит, по отношению к нему стоит применить возможности двумерной интерполяции Mathcad.

Чтобы выполнить поставленную задачу, сначала, аналогично уже рассмотренным выше методам двумерной сплайн-интерполяции, требуется задать вектор, содержащий численные значения вторых производных сплайна в узловых точках. Сделать это можно с помощью одной из трех функций семейства *spline:

- lspline(Mxy,Z) — поверхность будет построена из кубических сплайнов, однако ее экстраполирующее продолжение будет линейным;
- pspline(Mxy,Z) — поверхность из кубических сплайнов с параболическим продолжением;
- cspline(Mxy,Z) — продолжение сплайнов поверхности будет кубическим.

На практике обычно используется функция cspline. На ней остановимся и мы.

Очень интересными, однако и серьезно ограничивающими возможности двумерной интерполяции, являются особенности определения параметров функций *spline. Все дело в том, что успешно такую интерполяцию можно провести лишь в том случае, если исходные данные по результатам измерений занесены в квадратную матрицу (то есть количество замеров по каждому из направлений одинаково). Также обязательным условием является то, что шаг по каждой из переменных в каждой отдельном рядке сетки должен быть постоянным. Кроме того, некоторые сложности могут возникнуть в связи с довольно неожиданными и странными требованиями к форме задания значений координат экспериментальных точек по осям переменных.

Первый параметр функции cspline(Mxy,Z), отвечающий за характеристики сетки экспериментальных данных, должен быть определен как матрица размерности N×2, первый столбец которой содержит значения X-координат точек, определяющих своего рода главную диагональ сетки, а во второй — соответственно их координаты по Y. Именно эта особенность задания рассматриваемого параметра и является причиной описанных выше ограничений.

Так как обычно значения координат узловых точек сетки, в которых производились измерения, хранятся либо в виде векторов, либо, чаще, в виде матриц, подобных заданным нами выше, то для построения интерполирующей поверхности каким-то образом требуется организовать необходимую для функции cspline матрицу с координатами диагональных элементов. В обоих случаях эта задача предельно просто решается организацией одного лишь цикла ранжированной переменной (для диагональных элементов квадратной матрицы характерно равенство матричных индексов). Объединить же заданные таким образом векторы в одну матрицу можно с помощью специальной функции augment(A,B), где A — матрица, которая будет расположена справа, B — соответственно, присоединяемая слева матрица.

В нашем случае задать матрицу координат диагональных элементов можно следующим образом:

$$i := 0..4$$

$$R2_i := Y_{i,i} \qquad R1_i := X_{i,i}$$

$$R1 = \begin{pmatrix} -5 \\ -2.5 \\ 0 \\ 2.5 \\ 5 \end{pmatrix} \qquad R2 = \begin{pmatrix} -5 \\ -2.5 \\ 0 \\ 2.5 \\ 5 \end{pmatrix} \qquad R := \text{augment}(R1, R2) \qquad R = \begin{pmatrix} -5 & -5 \\ -2.5 & -2.5 \\ 0 & 0 \\ 2.5 & 2.5 \\ 5 & 5 \end{pmatrix}$$

В качестве второго элемента функции cspline следует прописать имя матрицы значений измеряемой характеристики (в нашем случае это матрица Z):

$$S := \text{cspline}(R, Z)$$

Результатом работы функции cspline будет вектор, три первые строки которого являются служебными, а остальные содержат значения вторых производных двумерного сплайна во всех узловых точках:

$S^T =$

	0	1	2	3	4	5	6	7	8	9
0	0	3	2	1.79	1.945	2.1	1.974	1.848	2.049	2.03

После того как вектор вторых производных будет создан, мы уже сможем непосредственно задать функцию приближающей поверхности. Сделать это можно с помощью встроенной функции interp(S,R,Z,t) точно так же, как в двумерном случае. Единственная тонкость связана с тем, что рассматриваемая функция должна быть определена как зависимость от двух переменных. Просто же ввести через запятую их имена на место параметра t ни в коем случае нельзя — при этом система выдаст сообщение о том, что функция имеет слишком много аргументов. Чтобы произвести задание корректно, переменные должны быть определены в виде вектора:

$$I(x,y) := \text{interp}\left[S, R, Z, \begin{pmatrix} x \\ y \end{pmatrix}\right]$$

Чтобы теперь визуализировать заданную интерполирующую поверхность, имя описывающей ее функции должно быть введено в соответствующий маркер графической области. Заданный таким образом график будет являться поверхностью быстрого построения. А это означает, что зависимость будет отображена для квадратной области от –5 до 5 по обеим переменным с шагом сетки 0.5 (то есть количество ее узлов будет автоматически увеличено системой с 25 до 441) (рис. 16.15).

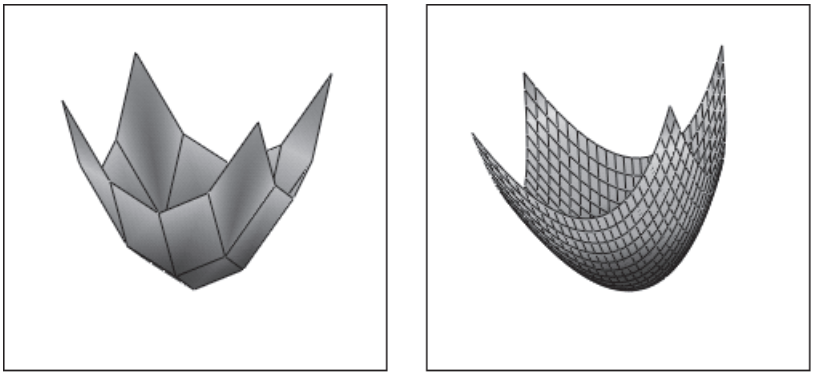


Рис. 16.15. Поверхность псевдоэкспериментальной зависимости до проведения интерполирования (график слева) и поверхность, соответствующая интерполирующей функции (график справа)

Сравнивая поверхности на рис. 16.15, нельзя не признать, что возможности многомерной интерполяции Mathcad заслуживают всяческих похвал. Действительно, системе удалось по нашим 25 экспериментальным точкам, вычисленным к тому же с некоторой погрешностью, построить абсолютно гладкий и правильный параболоид (для чего при обычном задании требуется по крайней мере несколько сотен точек). Еще лучше оценить возможности двумерной сплайн-интерполяции позволяет задание не поверхностей, а контурных графиков (рис. 16.16).

Если вы не ограничиваетесь отображением интерполирующей поверхности в области экспериментальных данных, а собираетесь ее использовать для экстраполяции опытной зависимости, принципиальным условием успеха является правильный выбор типа полинома, продолжающего сплайн. Так, например, в случае нашего параболоида наиболее корректный график, при его построении в пределах от –10 до 10 по обеим переменным, получается при применении для задания вектора вторых производных функ-

ции pspline. Если же вы решите использовать линейное продолжение сплайна (функция lspline), то форма поверхности будет весьма далека от требуемой (рис. 16.17). Определить же, какой тип продолжения лучше всего подходит для данного двумерного сплайна, проще всего можно, сравнив результат работы всех трех функций семейства *spline.

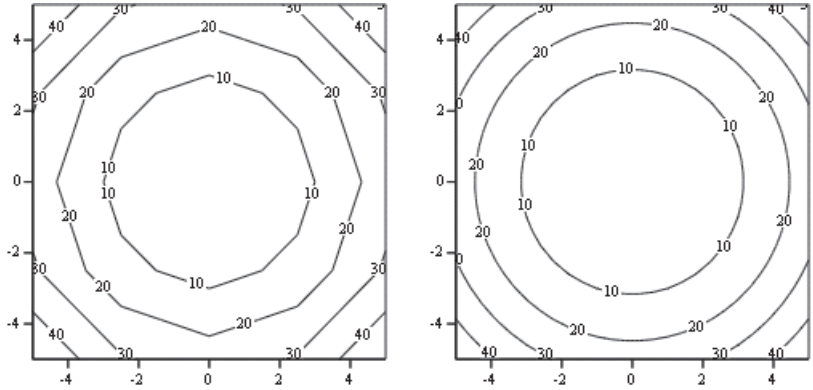


Рис. 16.16. Контурный вариант представленных на рис. 16.15 графиков

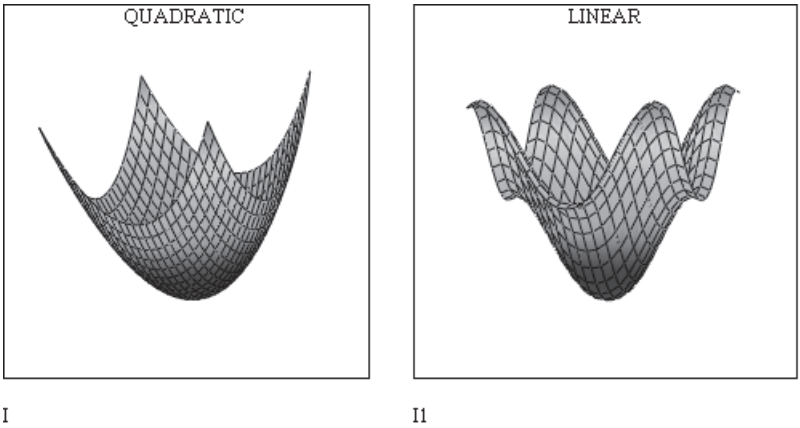


Рис. 16.17. Вид интерполирующей поверхности в зависимости от выбора типа полинома продолжения

16.1.5. Проведение математических операций над интерполирующими функциями

Функции, интерполирующие экспериментальные данные, являются непрерывными. Это означает, что по отношению к ним могут быть использованы те же математические преобразования, что и к таким «обычным» функциям, как синус или логарифм. Чаще всего экспериментальную зависимость приходится дифференцировать, интегрировать, определять ее экстремумы. Продемонстрируем на примере, как подобные операции могут быть проведены.

Пусть в результате эксперимента были получены следующие данные:

$$x := (0 \;\; 0.5 \;\; 1 \;\; 1.5 \;\; 2 \;\; 2.5 \;\; 3 \;\; 3.5 \;\; 4 \;\; 4.5 \;\; 5 \;\; 5.5 \;\; 6 \;\; 6.5 \;\; 7)^T$$

$$y := (0.1 \;\; 0.2 \;\; 0.6 \;\; 2.1 \;\; 6.1 \;\; 9.4 \;\; 10.4 \;\; 10.5 \;\; 10.6 \;\; 11.5 \;\; 14.5 \;\; 16.7 \;\; 17.4 \;\; 17.6 \;\; 17.7)^T$$

Интерполируем данные кубическим сплайном (рис. 16.18):

$$s := \mathrm{cspline}(x, y)$$

$$I(z) := \mathrm{interp}(s, x, y, z)$$

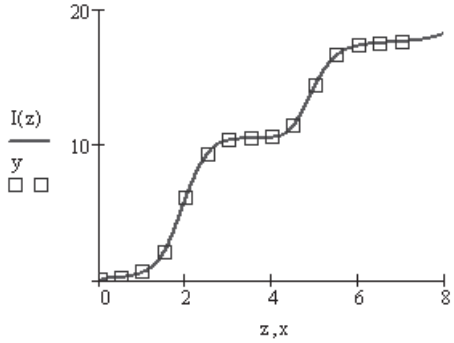


Рис. 16.18. Экспериментальные точки и проведенный через них кубический сплайн

Построенная нами интерполирующая кривая вполне удовлетворительно описывает экспериментальные точки. Однако на практике часто возникает необходимость извлечь из полученной зависимости некоторую информацию: например, вычислить значение изучаемого признака в точках перегиба функции или определить, какое значение признака появляется в выборке с наибольшей вероятностью. Зачастую проанализировать экспериментальную зависимость бывает затруднительно, поэтому для повышения «разрешающей способности» интерполирующей кривой прибегают к ее дифференцированию. Разумеется, в данном случае оно может быть проведено лишь численно, поскольку исходная зависимость не является аналитической (рис. 16.19).

$$\mathrm{der}(z) := \frac{d}{dz} I(z)$$

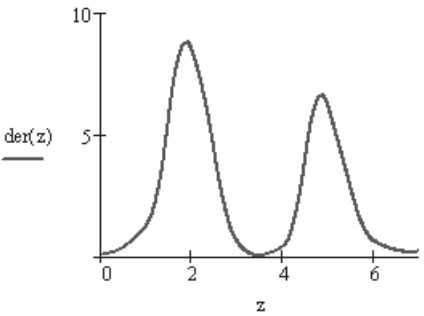


Рис. 16.19. Соответствующая экспериментальной зависимости дифференциальная кривая

После дифференцирования определить абсциссы точек перегиба стало намного проще, поскольку они же являются координатами вершин пиков по оси X. Вычислить максимумы пиков можно, задействовав функцию maximize:

$$z1 := 2 \qquad z2 := 14$$

$$\text{maximize}(\text{der}, z1) = 1.876 \qquad \text{maximize}(\text{der}, z2) = 4.841$$

Обратите внимание, для приближения z=5 (исходя из графика, второй максимум должен располагаться в окрестности этой точки) функция maximize определяет то же положение максимума, что и для z=2 (что неудивительно ввиду его большей величины). Чтобы этого избежать, нужно увеличить приближение — заметьте, в данном случае оно даже выходит за рамки рассматриваемого интервала.

Если требуется определить значение производной в некоторой точке, просто задайте ее координату x как аргумент соответствующей функции. Определим, например, величины максимумов пиков:

$$\text{der}(1.876) = 8.767 \qquad \text{der}(1.876) = 8.767$$

Во многих практических задачах возникает необходимость подсчитать площадь, ограниченную экспериментальной кривой или соответствующей ей дифференциальной кривой. В Mathcad эти элементарные операции можно провести с помощью средств численного интегрирования.

В нашем случае площади, ограниченные экспериментальной кривой I(z) и построенной на основании нее дифференциальной кривой der(z) на интервале [0; 7], равны:

$$\int_0^7 \text{I}(z)\,dz = 68.247 \qquad \int_0^7 \text{der}(z)\,dz = 17.6$$

Еще более трудоемким по сравнению с построением дифференциальной кривой является построение кривой интегральной. Однако в Mathcad эту задачу решить очень просто. Покажем, как можно построить интегральную кривую, основываясь на функции der (очевидно, что форма интегральной кривой при этом должна совпасть с формой экспериментальной зависимости) (рис. 16.20).

$$\text{Int}(p) := \int_0^p \text{der}(z)\,dz$$

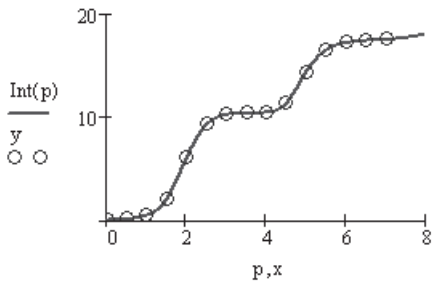


Рис. 16.20. Интегральная кривая, построенная на основании дифференциальной кривой

16.2. Предсказание поведения функции

На практике очень часто возникает задача предсказания поведения функции «в будущем» или «прошлом» на основании данных о ее поведении на каком-то отрезке. К примеру, целый ряд численных методов получил развитие в связи с необходимостью предсказания изменений на валютных рынках и рынках ценных бумаг. Вообще-то, качественный прогноз такого рода можно дать лишь в случае, если зависимость стационарна, то есть описывающий ее закон не изменяется во времени. Реально же встречаются, как правило, нестационарные зависимости, содержащие к тому же немалую случайную составляющую, обусловленную, в частности, погрешностью. По этой причине универсального метода предсказания поведения функций не существует. Однако если зависимость можно хотя бы приближенно считать стационарной (например, по причине узости интервала данных, на основании которого производится предсказание), то предсказание ее поведения в окрестности интервала наблюдения становится доступной для решения задачей.

Простейшим типом предсказания поведения зависимости является экстраполяция за счет продолжения интерполирующего полинома. С полиномиальной экстраполяцией с помощью продолжений сплайнов мы уже познакомились в предыдущих разделах этой главы. Однако вы, наверное, согласитесь, что такой тип предсказания для решения реальных задач может быть полезен не всегда, так как он является более или менее эффективным лишь в том случае, если экстраполируемые данные задаются полиномом невысокой степени. Если же исследуемая зависимость периодическая или склонна к осцилляции, то применение полиномиальной экстраполяции абсолютно бессмысленно.

Если экспериментальная зависимость ведет себя просто, то можно попытаться подобрать хорошо описывающее ее аналитическое выражение, коэффициенты которого затем можно будет найти посредством регрессивного анализа. Увы, но сделать это можно далеко не всегда. К примеру, кривая курса валюты представляет собой не гладкую кривую, а линию с непрерывной чередой острых пиков и разломов. Естественно, что описать такую зависимость каким-то простым выражением нереально.

Отдельные численные методы позволяют осуществлять более эффективное, чем при использовании полиномов, предсказание поведения разнообразных по форме зависимостей. Принцип их работы основывается на анализе поведения зависимости в нескольких ее точках, а не только на краях промежутка экспериментальных данных, как в случае применения в качестве прогнозных кривых продолжений сплайнов. В Mathcad функцией, реализующей один из алгоритмов предсказания (метод линейного предсказания Берга (Burg)), является встроенная функция predict.

Основополагающая идея всех методов линейного предсказания проста. Пусть нам известно N экспериментальных точек и необходимо предсказать, где окажется N+1 точка. Будем считать, что каждая последующая точка может быть определена, как линейное сочетание M предыдущих точек: $y_{N+1}=a_0+a_1 \cdot y_{N-M}+a_2 \cdot y_{N-M+1}+a_M \cdot y_N$. Необходимо найти оптимальные коэффициенты $a_0 \ldots a_M$, анализируя поведение функции в известных точках. О том, как это делается, можно прочитать в книгах по численным методам. Мы лишь отметим, что то, что метод Берга относится к группе методов линейного предсказания, еще не значит, что он подходит лишь для предсказания поведения линейных зависимостей. Данный метод способен предсказывать поведение периодических или осциллирующих зависимостей на промежутке значительно более широком по сравнению

с промежутком, на котором хорошее предсказание дает полиномиальная экстраполяция.

Функция predict(y,m,n) требует задания следующих параметров:

- y — вектор эмпирических значений исследуемой зависимости по оси ординат. Особенность алгоритма Берга заключается в том, что предсказание он делает только на основании y-координат выборки (при этом подразумевается, что шаг по оси абсцисс постоянен);
- m — количество ближайших к правой границе выборки точек, на основании которых проводится предсказание;
- n — количество точек в просчитываемом векторе прогноза. Сам по себе метод Берга дает возможность предсказать значение только одной точки. Но функция predict использует подход «скользящего окна»: получив на основании m экспериментальных точек предсказание для N+1-й точки, следующее предсказание рассчитывается с учетом этой точки. Соответственно, N–M-я точка в расчете уже не участвует. Если n велико, то часть точек вполне может быть рассчитана без прямого участия экспериментальных данных.

С помощью функции predict можно проводить довольно эффективное предсказание поведения непрерывных и гладких периодических или осциллирующих функций (однако обычно в относительно неширокой области). Например, попробуем предсказать поведение кривой затухающих колебаний. Для этого прежде всего зададим вектор из y-координат ее 101 точки на промежутке от 0 до 3π. Обязательным условием корректного использования функции predict является то, что шаг изменения переменной при определении вектора данных должен быть строго постоянным. Очевидно, что самым простым способом организации такого вектора является использование ранжированных переменных:

$$N := 100$$

$$n := 0 .. N$$

$$x_n := 3\pi \cdot \frac{n}{N} \qquad y_n := e^{\frac{-x_n}{5}} \cdot \cos\left(x_n\right)$$

Далее зададим векторы предсказания с помощью функции predict. Чтобы сравнить степень влияния количества анализируемых точек выборки на качество предсказания, определим сразу три вектора при различных значениях параметра m. Размерность же этих векторов зададим, например, как 150:

$$Ex := predict\left(y, \frac{N}{2}, 150\right) \qquad Ex2 := predict(y, N, 150) \qquad Ex3 := predict\left(y, \frac{N}{5}, 150\right)$$

После задания векторов приближений можно построить соответствующие графики (рис. 16.21). При этом переменная для векторов экстраполяции может быть определена прибавлением к вектору x соответствующей координаты крайнего значения в выборке. В нашем случае это 3π. Такой подход оправдывается тем, что при построении графика мы должны учитывать, что функция predict выдает точки экстраполяции с тем расчетом, что шаг между ними такой же, как между x-координатами выборки.

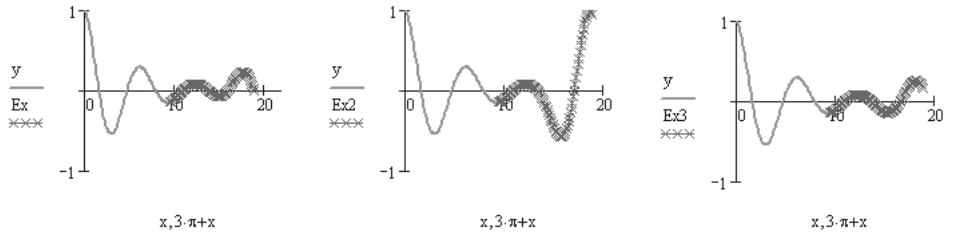


Рис. 16.21. Предсказание поведения осциллирующей функции

Внимательно изучив графики на рис. 16.21, вы, наверное, согласитесь, что предсказание поведения зависимости с помощью функции predict довольно эффективно (с учетом специфики вопроса), однако лишь на очень небольшом отрезке правее крайней точки выборки. Чем же дальше расположен фрагмент приближающей кривой от точек, на основании которых проводится предсказание, тем в меньшей степени, скорее всего, он будет соответствовать реальному поведению зависимости.

К весьма интересному и неожиданному выводу можно прийти, анализируя влияние количества точек выборки, на основании которых проводится предсказание, на точность последнего. Казалось бы, чем больше точек обрабатывается, тем точнее должна быть информация о продолжаемой кривой и тем более близким будет график экстраполирующей кривой к реальной зависимости. Однако, глядя на приведенные графики, можно обнаружить, что такой прямой зависимости не существует: в случае обработки всех точек выборки приближающая кривая очень быстро возрастает, чего не наблюдается при анализе половины экспериментальных точек.

Успех экстраполяции очень сильно зависит от типа исследуемой зависимости. Лучше всего удается предсказание поведения несложных периодических функций. Такие простые функции, как синус или косинус, корректно продолжаются функцией predict практически на сколь угодно широком интервале (рис. 16.22). Поведение непериодических функций предсказывается намного хуже, даже если это такие простые по виду зависимости, как парабола или логарифмическая кривая.

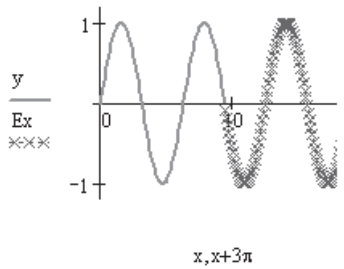


Рис. 16.22. Предсказание поведения простой периодической функции

К результатам предсказания поведения зависимости нужно относиться очень осторожно. Если в выборке не очень много точек и шаг между ними велик (или при наличии заметной погрешности), пытаться проводить предсказание по методу Берга совершенно бессмысленно. Тут нужно обращаться к более совершенным методам, например методу максимальной энтропии.

16.3. Регрессивный анализ данных

Задачей регрессивного анализа является установление параметров описывающего экспериментальную зависимость выражения с учетом того, что эмпирические точки получены с некоторой погрешностью. Обычно это делается с помощью расчета минимума тем или иным образом задаваемой функции ошибки. Очень часто назначение регрессии упрощают до построения гладкой кривой между точками опытных данных. Однако реальные возможности регрессивного анализа гораздо шире, и без этой области статистики невозможно представить ни прикладной физики, ни современной техники.

Практическая важность регрессии огромна: так, например, любой корректно построенный график экспериментальной зависимости должен быть задан при ее помощи (интерполяцию корректно использовать, лишь если данные получены с точностью не менее 4–5 знаков мантиссы). Реально задачи регрессии встречаются на практике куда чаще, чем интерполяции или, тем более, экстраполяции. Как и любые другие статистические расчеты, вычисление регрессии может быть весьма трудоемким в случае его проведения на бумаге — ввиду большого объема вычислительной работы. В Mathcad же, благодаря наличию многочисленных встроенных функций, задачи такого рода можно решать очень и очень легко. Конкретно о том, как это делается, мы и поговорим в следующих подразделах.

16.3.1. Линейная регрессия

Линейная регрессия является самым простым и в то же время наиболее часто используемым типом регрессии (так, если вы студент технического вуза, то, скорее всего, только с таким видом обработки данных вам и придется встретиться на практических занятиях по статистике). Классическим алгоритмом линейной регрессии является метод наименьших квадратов, идея которого сводится к поиску таких коэффициентов для уравнения прямой, чтобы сумма квадратов абсолютных ошибок $(b+a\cdot x_i-y_i)$ была минимальна. В Mathcad этот метод является основным. Однако существуют и другие алгоритмы для подсчета линейной регрессии, один из которых — метод медиан — также имеет в Mathcad свою встроенную функцию. Как можно вычислить в Mathcad коэффициенты экспериментальной прямой и какой алгоритм для этого предпочесть — во всех этих вопросах мы попытаемся разобраться в этом подразделе.

Рассчитать линейную регрессию по методу наименьших квадратов в Mathcad можно с помощью двух альтернативных способов. Первый, более простой, использует специальную функцию line(x,y), где x и y — соответствующие векторы экспериментальных данных. Результатом работы этой функции является вектор, первый элемент которого отвечает коэффициенту b, а второй — коэффициенту a уравнения наиболее хорошо усредняющей данную выборку прямой (функция прямой имеет общий вид $f(x)=a\cdot x+b$).

Приведем пример работы функции line(x,y). Для этого зададим на основании конкретной линейной зависимости вектор линейных псевдоэкспериментальных данных с определенным уровнем погрешности. Чтобы это сделать, просто прибавим к результатам точного расчета значений линейной функции в определенных с помощью ранжированной переменной точках значения ошибок, которые получим с помощью генератора нормально распределенных случайных чисел rnorm(N,a,σ). При этом, исходя из физического смысла погрешности, математическое ожидание определим как 0, а дисперсию, наоборот, сделаем значительнее, чем она могла бы быть на практике — это позволит

испытать рассматриваемую функцию более объективно. Сравнивая значения коэффициентов исходной линейной зависимости с результатом работы функции line, мы сможем оценить ее эффективность. Кроме того, определить степень соответствия исходной и определенной с помощью регрессии линейных функций можно, построив их графики.

Пример 16.5. Линейная регрессия

Задаем векторы псевдоэкспериментальных данных на основании известной линейной зависимости:

$$a := 1 \qquad b := -2$$

$$N := 10$$

$$Err := rnorm(N, 0, 1)$$

$$i := 0..N-1 \qquad x_i := 10 \cdot \frac{i}{N} \qquad y_i := a \cdot x_i + b + Err_i$$

Вычисляем коэффициенты линейной регрессии:

$$R := line(x, y) \qquad R = \begin{pmatrix} -1.152 \\ 0.845 \end{pmatrix}$$

Задаем на основании полученных коэффициентов функцию прямой и строим ее график. Также строим график линейной зависимости, соответствующей «экспериментальным» данным (рис. 16.23).

$$r(z) := z \cdot R_1 + R_0 \qquad f(t) := a \cdot t + b$$

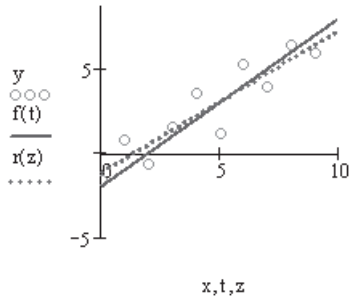


Рис. 16.23. Прямая, построенная по итогам регрессивного анализа (r(z)), и «правильная» прямая (f(t))

Сравнив значения, полученные функцией line, с изначально заданными коэффициентами линейной зависимости, вы можете возмутиться, заметив, что разница между ними весьма велика. Однако вы должны в данном случае учитывать, что для того, чтобы мы могли получить не оценочное, а достоверное значение параметров при использовании любого из регрессивных методов, уровень погрешности в выборке должен быть много ниже, чем в заданной нами, да и эмпирических точек желательно тоже проанализировать побольше. Так, например, снизив при задании уровня погрешности значение среднеквадратичного отклонения с 1 до 0,1, в качестве результата в примере 16.5 получим:

$$R = \begin{pmatrix} -2.07 \\ 1.011 \end{pmatrix}$$

Как видите, точность результата при этом возросла более чем в 10 раз. Из этого следует вывод: для получения корректного ответа эксперимент нужно проводить максимально аккуратно. Если же это невозможно, то для увеличения точности необходимо сделать больше опытов. Так, при увеличении количества измерений в примере 16.5 в три раза (с N=10 до N=30) точность возросла в 10 раз для b и в 50 (!) раз для a:

$$R = \begin{pmatrix} -2.064 \\ 0.997 \end{pmatrix}$$

Недостатком функции line является то, что результат расчета она выдает в виде одного вектора, что не всегда может быть удобно. Однако в Mathcad существуют функции, позволяющие вычислять коэффициенты уравнения линейной регрессии и по отдельности:

- intercept(x,y) — определяет значение коэффициента b (что соответствует первой строке вектора результата функции line);
- slope(x,y) — находит величину коэффициента a (равняется второму элементу вектора ответа функции line).

Так как приведенные функции и функция line используют для расчета один и тот же метод наименьших квадратов, то никакой разницы в выдаваемых ими результатах не будет. Например, если положить в примере 16.5 N=15, то расчет линейной регрессии альтернативными способами даст следующий результат:

$$\text{line}(x, y) = \begin{pmatrix} 1.902 \\ 1.029 \end{pmatrix} \qquad \text{intercept}(x, y) = 1.902 \qquad \text{slope}(x, y) = 1.029$$

Оценить, насколько хорошо экспериментальные точки ложатся на построенную с помощью регрессии линию, можно и чисто визуально. Но лучше получить некоторую количественную оценку. Оптимальной количественной оценкой (при отсутствии в данных промахов) является так называемая стандартная ошибка, равная среднеквадратичному расстоянию от точек данных до прямой. Если линейная регрессия рассчитывается по методу наименьших квадратов, то для нахождения стандартной ошибки можно использовать встроенную функцию Mathcad stderr(x,y), где x и y — вектора данных.

В любом, даже самом общем курсе математики, излагаются идеи метода наименьших квадратов. И не всегда на вузовских занятиях уместен расчет с использованием специальных встроенных функций Mathcad, поэтому будет правильным привести пример того, как можно определить регрессию с использованием более общих средств.

Пусть в результате проведения эксперимента была получена последовательность из N точек $\{x_k, y_k\}$, $k \in 0 \ldots N-1$. Пусть нам необходимо построить на основании данных точек прямую линию $f(x)=a \cdot x+b$. Для этого нужно так подобрать коэффициенты a и b, чтобы сумма квадратов отклонений экспериментальных точек от прямой (функция $E_2(a, b)$) была минимальной.

$$E_2(a, b) = \sum_{k=0}^{N-1} \left(a \cdot x_k + b - y_k \right)^2$$

Функция $E_2(a, b)$ является функцией двух переменных. Точку ее экстремума можно найти, положив ее частные производные по a и b равными нулю и решив полученную систему линейных уравнений. В результате будут получены следующие выражения для a и b:

$$a = \frac{-S_x \cdot S_y + S_{xy} \cdot N}{\Delta} \qquad b = \frac{-S_x \cdot S_{xy} + S_y \cdot S_{xx}}{\Delta}$$

$$\Delta = S_{xx} \cdot N - S_x^2$$

Переменные S_x, S_y, S_{xy}, S_{xx} соответствуют следующим суммам:

$$S_x = \sum_{k=0}^{N-1} x_k \qquad S_y = \sum_{k=0}^{N-1} y_k \qquad S_{xx} = \sum_{k=0}^{N-1} (x_k)^2 \qquad S_{xy} = \sum_{k=0}^{N-1} x_k \cdot y_k$$

Используя приведенные формулы, совсем несложно создать программу, реализующую метод наименьших квадратов (см. пример 16.6).

Пример 16.6. Проверка закона Мура

Закон Мура — это самое известное наблюдение, касающееся развития рынка высоких технологий. Его открывателем является Гордон Мур, один из основателей и руководитель компании Intel. Закон Мура гласит, что количество транзисторов на одной микросхеме удваивается за 1,5 года. Ввиду особенностей архитектуры процессоров, закон Мура можно перенести и на тактовую частоту. Попробуем, основываясь на данных о тактовой частоте процессоров Intel, выпускавшихся с 1971 по 1997 год предсказать, какая тактовая частота (в мегагерцах) должна была быть достигнута в 2005 году. О качестве прогноза можно будет легко судить, обратившись к прайс-листу любой компании, торгующей компьютерами.

Для начала зададим в виде векторов исходные данные:

$$year := (1971 \ \ 1974 \ \ 1978 \ \ 1982 \ \ 1985 \ \ 1989 \ \ 1993 \ \ 1995 \ \ 1997)^T$$

$$clock\_rate := (0.108 \ \ 2 \ \ 7 \ \ 10 \ \ 24 \ \ 65 \ \ 145 \ \ 175 \ \ 300)^T$$

Построив на основании приведенных данных график (см. ниже), мы увидим, что точки соответствуют не прямой, а, скорее, экспоненциальной кривой (скорость возрастания зависимости тем больше, чем больше аргумент). Это же заключение можно сделать и из формулировки закона Мура, который в виде аналитического выражения может быть записан как $C(1.5 \cdot n) = 2^n \cdot C_0$. Но как описать закон Мура линейной функцией, если сама исследуемая зависимость является экспоненциальной? Это очень просто сделать, тождественно преобразовав выражение для закона Мура. Пусть изменение тактовой частоты во времени может быть описано функцией $c = a \cdot e^{b \cdot x}$. Прологарифмируем обе части равенства. В результате (с учетом того, что $\ln(m \cdot n) = \ln(m) + \ln(n)$ и $\ln(m^n) = n \cdot \ln(m)$) получим: $\ln(c) = b \cdot x + \ln(a)$. Данное уравнение является линейным в координатах $x\text{-}\ln(c)$, поэтому значения b и $\ln(c)$ можно рассчитать с помощью линейного метода наименьших квадратов. Зная же $\ln(a)$, легко найти a, используя следующее тождество: $n = e^{\ln(n)}$. Дальше только останется подставить полученные значения a и c в исходное выражение.

Чтобы использовать описанную стратегию, для начала нужно прологарифмировать вектор ординат:

$$ln\_clock\_rate := \ln(clock\_rate)$$

Так как мы решили в этом примере рассчитать линейную регрессию, не прибегая к встроенным функциям, то нужно создать функцию, которая будет выполнять эту работу. Сделать это на основании приведенных выше формул проще простого:

$$\text{lin\_reg}(x, y) := \left| \begin{array}{l} N \leftarrow \text{length}(x) \\ \left[S_x \leftarrow \sum_{k=0}^{N-1} x_k \quad S_y \leftarrow \sum_{k=0}^{N-1} y_k \quad S_{xx} \leftarrow \sum_{k=0}^{N-1} (x_k)^2 \quad S_{xy} \leftarrow \sum_{k=0}^{N-1} x_k \cdot y_k \right] \\ \Delta \leftarrow S_{xx} \cdot N - S_x^{\,2} \\ \text{return} \left(\frac{-S_x \cdot S_{xy} + S_y \cdot S_{xx}}{\Delta} \quad \frac{-S_x \cdot S_y + S_{xy} \cdot N}{\Delta} \right)^T \end{array} \right.$$

Рассчитываем коэффициенты a и b с помощью программы lin_reg. Чтобы проверить эффективность нашего алгоритма, параллельно коэффициенты определяем посредством функции line:

$$\text{lin\_reg}(\text{year}, \text{ln\_clock\_rate}) = \begin{pmatrix} -515.40096 \\ 0.26111 \end{pmatrix} \quad \text{line}(\text{year}, \text{ln\_clock\_rate}) = \begin{pmatrix} -515.40096 \\ 0.26111 \end{pmatrix}$$

Результаты расчета регрессии с помощью нашей программы и встроенной функции совпали. А это означает, что формулы в основе lin_reg и line лежат одни и те же.

Задаем на основании полученных значений a и b описывающую закон Мура функцию. Строим ее график, чтобы убедиться, что она соответствует точкам данных (рис. 16.24).

$$\text{reg\_line}(x) := e^{0.261 \cdot x - 515.4}$$

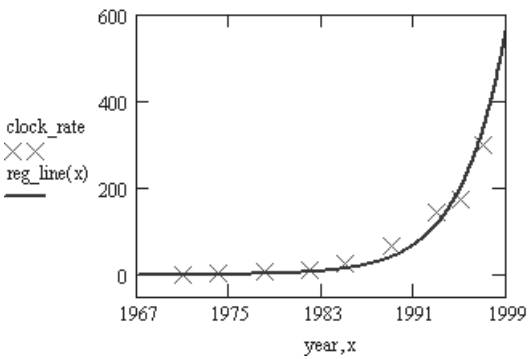


Рис. 16.24. Точки данных и кривая функции регрессии

Точки данных ложатся на кривую регрессии довольно неплохо. Значит, вполне можно рассчитывать на то, что предсказание тактовой частоты процессоров Intel в 2005 году будет хорошим:

$$\text{reg\_line}(2005) = 2710.803$$

Полученное предсказание соответствует действительности. Значит, закон Мура все еще работает. Эксперты предсказывают, что данный закон будет актуален где-то до 2020 года.

Посмотрим, какой тактовой частоты достигнут процессоры к тому времени:

$$reg\_line(2020) = 135944.229$$

В примере 16.6 мы использовали технику линеаризации по отношению к экспоненциальной функции $y=a\cdot e^{b\cdot x}$ для того, чтобы найти коэффициенты a и b посредством линейной регрессии. В Mathcad имеется функция expfit(x,y), которая позволяет решать эту задачу проще. Однако, увы, она не очень надежна. Так, найти параметры для закона Мура с ее помощью оказалось невозможно, так как при этом возникла ошибка (рис. 16.25).

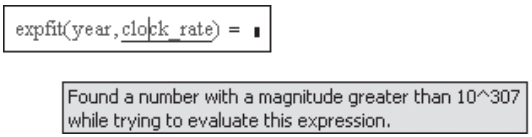


Рис. 16.25. Сообщение об ошибке гласит: «При попытке вычисления данного выражения обнаружено число со степенью, большей 10^{307}»

Помимо метода наименьших квадратов, рассчитать коэффициенты линейной регрессии вы можете с помощью другого, менее распространенного и известного алгоритма медиан (median-median regression). Для этого обратитесь к специальной встроенной функции medfit(x,y), где x и y — векторы координат точек экспериментальных данных. Результатом ее работы является точно такой же вектор, как и в случае функции line, первой строке которого соответствует значение коэффициента b, а второй — коэффициента a уравнения регрессии.

Так как метод медиан относительно мало известен, его стоит описать. Тем более что он очень прост. Чтобы найти уравнение прямой посредством медиан-медианной регрессии, нужно проделать следующие шаги.

1. Разделяем экспериментальные точки на три равные группы, исходя из величины x-координаты. В первой группе должны быть «левые» точки, во второй — «центральные», а в третьей — «правые». Если количество точек в выборке не кратно 3, то нужно действовать так, чтобы точки распределились по группам максимально равномерно. Если остаток от деления количества точек в выборке на три составляет один, то «лишнюю» точку следует поместить в центральную группу. Если «лишних» точек две, то по одной точке нужно прибавить в левую и правую группы.
2. Для каждой группы находим среднюю точку, то есть точку, координаты которой являются средним арифметическим соответствующих координат всех точек группы.
3. Находим уравнение прямой, проходящей через правую и левую средние точки.
4. Строим вторую прямую так, чтобы она была параллельна первой прямой. Располагаться вторая прямая должна между первой прямой и средней точкой центральной группы. Причем расстояние от второй прямой до первой должно быть в два раза меньше, чем от средней точки центральной группы до второй прямой. Смысл этого требования довольно очевиден: вторая прямая должна быть построена так, чтобы сумма расстояний от нее и до средних точек была минимальной.
5. Параметры уравнения второй прямой возвращаются как результат.

Технически реализовать метод медиан очень просто, решив несколько элементарных задач на уравнение прямой. Попробуйте это сделать самостоятельно. Если не получится — проанализируйте код программы mm.

$$
\begin{aligned}
mm(x,y) := \;& N \leftarrow \mathrm{length}(x) \\
& \left(b1 \leftarrow \frac{N}{3} - 1 \quad b2 \leftarrow \frac{2\cdot N}{3} - 1 \right) \text{ if } \mathrm{mod}(N,3) = 0 \\
& \left(b1 \leftarrow \frac{N}{3} - 1 \quad b2 \leftarrow \frac{2\cdot N}{3} \right) \text{ if } \mathrm{mod}(N,3) = 1 \\
& \left(b1 \leftarrow \frac{N}{3} \quad b2 \leftarrow \frac{2\cdot N}{3} \right) \text{ if } \mathrm{mod}(N,3) = 2 \\
& (x1 \leftarrow \mathrm{submatrix}(x,0,b1,0,0)\ x2 \leftarrow \mathrm{submatrix}(x,b1,b2,0,0)\ x3 \leftarrow \mathrm{submatrix}(x,b2,N-1,0,0)) \\
& (y1 \leftarrow \mathrm{submatrix}(y,0,b1,0,0)\ y2 \leftarrow \mathrm{submatrix}(y,b1,b2,0,0)\ y3 \leftarrow \mathrm{submatrix}(y,b2,N-1,0,0)) \\
& (m\_x1 \leftarrow \mathrm{mean}(x1)\ \ m\_x2 \leftarrow \mathrm{mean}(x2)\ \ m\_x3 \leftarrow \mathrm{mean}(x3)) \\
& (m\_y1 \leftarrow \mathrm{mean}(y1)\ \ m\_y2 \leftarrow \mathrm{mean}(y2)\ \ m\_y3 \leftarrow \mathrm{mean}(y3)) \\
& [k\_out \leftarrow (m\_y3 - m\_y1) \div (m\_x3 - m\_x1)\ \ b\_out \leftarrow m\_y1 - k\_out\cdot m\_x1] \\
& b\_intro \leftarrow m\_y2 - k\_out\cdot m\_x2 \\
& b\_med \leftarrow b\_out - \frac{(b\_out - b\_intro)}{3} \\
& (b\_med\ \ k\_out)^{T}
\end{aligned}
$$

Метод медиан довольно примитивен по сравнению с методом наименьших квадратов. Поэтому в большинстве случаев он дает худший результат. Однако у метода медиан есть одно преимущество: он куда менее чувствителен к наличию в выборке промахов, то есть точек, полученных с недопустимо высокой погрешностью. Поэтому к нему нужно обращаться, если данные плохо ложатся на прямую.

Пример 16.7. Расчет линейной регрессии с использованием метода медиан

Моделируем псевдоэкспериментальную линейную зависимость со средним уровнем погрешности:

$$a := 5 \qquad b := -4$$

$$err := rnorm(16,0,1)$$

$$i := 0..15$$

$$x_i := -5 + i \qquad y_i := a\cdot x_i + b + err_i$$

Делаем три точки в выборке промахами, добавляя к ним значения погрешности, в 15 раз превышающие ее нормальный уровень:

$$ERR := rnorm(3,0,15)$$

$$y_0 := y_0 + ERR_0 \qquad y_4 := y_4 + ERR_1 \qquad y_8 := y_8 + ERR_2$$

Находим коэффициенты уравнения прямой с помощью метода медиан (используя встроенную функцию medfit и программу mm) и метода наименьших квадратов:

$$medfit(x,y) = \begin{pmatrix} -3.992 \\ 5.082 \end{pmatrix} \quad mm(x,y) = \begin{pmatrix} -4.74 \\ 5.486 \end{pmatrix} \quad line(x,y) = \begin{pmatrix} -5.566 \\ 5.76 \end{pmatrix}$$

Как видите, коэффициенты, найденные по методу медиан, более соответствуют истинным, чем определенные по методу наименьших квадратов. Особенно точен результат работы функции medfit. Но и наша программа mm сработала лучше, чем функция line. Впрочем, на графике все три прямые ведут себя практически идентично.

В примере 16.6 мы, путем тождественных преобразований, смогли привести экспоненциальную функцию $y=a\cdot e^{b\cdot x}$ к такой форме, что для расчета ее параметров стало возможно использовать линейную регрессию. Подобная техника называется методом линеаризации данных. Линеаризация чрезвычайно важна для практики, так как линейная регрессия, в отличие от других видов регрессии, точна, однозначна и относительно легко рассчитывается. Существует довольно значительное количество зависимостей, которые, путем тождественных преобразований и замен переменных, можно преобразовать в линейную форму. Наиболее важные случаи приведены в табл. 16.1.

Таблица 16.1. Линеаризация некоторых зависимостей

Функция	Линеаризованная форма Y=A X+B	Необходимые замены переменных и постоянных
$y = \frac{A}{x} + B$	$y = A\cdot\frac{1}{x} + B$	$X = \frac{1}{x}, Y = y$
$y = \frac{D}{x + C}$	$y = \frac{-1}{C}\cdot x\cdot y + \frac{D}{C}$	$X = x\cdot y, Y = y, C = \frac{-1}{A}, D = \frac{-B}{A}$
$y = \frac{1}{A\cdot x + B}$	$\frac{1}{y} = A\cdot x + B$	$X = x, Y = \frac{1}{y}$
$y = \frac{x}{A\cdot x + B}$	$\frac{1}{y} = B\cdot\frac{1}{x} + A$	$X = \frac{1}{x}, Y = \frac{1}{y}$
$y = A\cdot\ln(x) + B$	$y = A\cdot\ln(x) + B$	$X = \ln(x), Y = y$
$y = C\cdot e^{A\cdot X}$	$\ln(y) = A\cdot x + \ln(C)$	$X = x, Y = \ln(y), C = e^{B}$
$y = C\cdot x^{A}$	$\ln(y) = A\cdot\ln(x) + \ln(C)$	$X = \ln(x), Y = \ln(y), C = e^{B}$
$y = \frac{1}{(A\cdot x + B)^2}$	$\frac{1}{\sqrt{y}} = A\cdot x + B$	$X = x, Y = \frac{1}{\sqrt{y}}$
$y = C\cdot x\cdot e^{-D\cdot x}$	$\ln\left(\frac{y}{x}\right) = -D\cdot x + \ln(C)$	$X = x, Y = \ln\left(\frac{y}{x}\right), C = e^{B}, D = -A$
$y = \frac{1}{1 + C\cdot e^{A\cdot x}}$	$\ln\left(\frac{1}{y} - 1\right) = A\cdot x + \ln(C)$	$X = x, Y = \ln\left(\frac{1}{y} - 1\right), C = e^{B}$

В Mathcad есть две функции, использующие технику линеаризации для нахождения параметров нелинейных зависимостей. Это уже известная нам функция expfit, рассчитывающая регрессию для экспоненциальной функции $y=a\cdot e^{b\cdot x}$, а также функция lnfit(x,y), дающая возможность определить параметры оптимально приближающей данные логарифмической кривой $y=a\cdot\ln(x)+b$. Чтобы рассчитать регрессию для других зависимостей, допускающих линеаризацию, все необходимые замены придется осуществлять самостоятельно.

16.3.2. Полиномиальная регрессия

Очень многие эффекты и явления в физике и других точных науках описываются алгебраическими полиномами. Пожалуй, в этом плане им могут составить конкуренцию лишь различного рода экспоненциальные формулы — результат решения обыкновенных дифференциальных уравнений. По этой причине практическая важность полиномиальной регрессии ничуть не уступает, а может, даже и превосходит таковую линейной регрессии, о которой мы говорили в предыдущем подразделе.

Построить полином произвольной степени по экспериментальным данным довольно просто, если использовать метод наименьших квадратов. Попробуем, чтобы лучше разобраться в вопросе, самостоятельно вывести рабочую формулу для данного метода.

Пусть в результате измерений было получено множество точек $\{x_k, y_k\}$, $k\in 0\ldots K$. Предположим, что из теории известно, что исследуемое явление может быть описано полиномом степени N ($N<K-1$). Задача — найти коэффициенты данного полинома на основании экспериментальных данных.

Для начала зададим функцию полинома в общем виде:

$$P_N(x) = c_N\cdot x^N + c_{N-1}\cdot x^{N-1} + c_{N-2}\cdot x^{N-2} + \ldots + c_1\cdot x + c_0 = \sum_{n=0}^{N} c_n\cdot x^n$$

Количественной характеристикой того, насколько правильно определены коэффициенты полинома, возьмем сумму квадратов расстояний от экспериментальных точек до кривой. Назовем соответствующую функцию E. Параметрами функции E будут являться коэффициенты полинома:

$$E(c_0, c_1, c_2, \ldots, c_n) = \sum_{k=0}^{K} \left(P_N(x_k) - y_k\right)^2 = \sum_{k=0}^{K} \left(\sum_{n=0}^{N} c_n\cdot x_k^n - y_k\right)^2$$

Оптимально подобранным коэффициентам полинома соответствует минимум функции ошибки E. Функция E является функцией N+1 переменных, где N — степень полинома. Чтобы найти ее минимум, нужно вычислить ее частные производные по всем переменным, а затем, считая их равными нулю, решить полученную систему из N+1 уравнения.

Дифференцирование функции E по любой переменной даст выражение, соответствующее следующей общей формуле:

$$\frac{\partial E(c_0, c_1, \ldots, c_n)}{\partial c_p} = \sum_{k=0}^{K} 2\cdot\left(\sum_{n=0}^{N} c_n\cdot x_k^n - y_k\right)\cdot x_k^p = 0$$

Варьируя p от 0 до N, мы получим искомые N+1 уравнения. Но как их решить? Очень просто. Дело в том, что данные уравнения являются линейными относительно коэффициентов полинома (c_0, c_1, ..., c_N). Чтобы это показать, сократим на не играющий никакой роли множитель 2, а затем раскроем скобки:

$$\sum_{k=0}^{K}\sum_{n=0}^{N} c_n \cdot x_k^n \cdot x_k^p - \sum_{k=0}^{K} y_k \cdot x_k^p = 0$$

Пользуясь свойством дистрибутивности сложения, неизвестные c_n можно вынести за знак суммы:

$$\sum_{n=0}^{N} c_n \cdot \sum_{k=0}^{K} x_k^{n+p} = \sum_{k=0}^{K} y_k \cdot x_k^p$$

Полученное уравнение является основным уравнением полиномиальной регрессии. Оно задает систему линейных уравнений, решения которой — это необходимые коэффициенты полинома. В матричном виде данную систему можно представить как:

$$\begin{pmatrix} 1 & \sum_{k=0}^{K} x_k & \sum_{k=0}^{K} x_k^2 & \cdots & \sum_{k=0}^{K} x_k^N \\ \sum_{k=0}^{K} x_k & \sum_{k=0}^{K} x_k^2 & \sum_{k=0}^{K} x_k^3 & \cdots & \sum_{k=0}^{K} x_k^{N+1} \\ \sum_{k=0}^{K} x_k^2 & \sum_{k=0}^{K} x_k^3 & \sum_{k=0}^{K} x_k^4 & \cdots & \sum_{k=0}^{K} x_k^{N+2} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum_{k=0}^{K} x_k^N & \sum_{k=0}^{K} x_k^{N+1} & \sum_{k=0}^{K} x_k^{N+2} & \cdots & \sum_{k=0}^{K} x_k^{2N} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{bmatrix} \sum_{k=0}^{K} y_k \\ \sum_{k=0}^{K} y_k \cdot x_k \\ \sum_{k=0}^{K} y_k \cdot x_k^2 \\ \vdots \\ \sum_{k=0}^{K} y_k \cdot x_k^N \end{bmatrix}$$

Итак, ключевым моментом в алгоритме полиномиальной регрессии является решение системы линейных уравнений, коэффициенты которой определяются экспериментальными данными. Справиться с этой задачей можно, например, использовав встроенную функцию Mathcad lsolve.

Формулы, используемые при расчете полиномиальной регрессии, довольно громоздкие. Однако реализовать соответствующий алгоритм на языке программирования Mathcad можно буквально несколькими строками кода:

$$\begin{aligned} &\text{pol\_reg}(x, y, n) := \\ &\quad \text{for } i \in 0..n \\ &\quad\quad \text{for } j \in 0..n \\ &\quad\quad\quad \left[M_{i,j} \leftarrow \sum_{k=0}^{\text{last}(x)} \left(x_k\right)^{i+j} \quad N_i \leftarrow \sum_{k=0}^{\text{last}(x)} y_k \cdot \left(x_k\right)^i \right] \\ &\quad \text{lsolve}(M, N) \end{aligned}$$

Программа pol_reg очень проста, однако она написана не совсем рационально. Дело в том, что при заполнении матрицы коэффициентов M сумма для каждого элемента просчитывается индивидуально. Между тем несложно заметить, что в любой строке (кроме первой) из N+1 сумм N сумм уже вычислялась при заполнении предыдущих строк. Поэтому, чтобы заполнить всю матрицу коэффициентов, достаточно вычислить 2·N+1 сумму, а не $(N+1)^2$ сумму, как это делается в программе pol_reg. Попробуйте самостоятельно так переписать программу, чтобы она каждую сумму вычисляла только один раз.

Проверим, правильными ли были наши рассуждения. Для этого рассчитаем полиномиальную регрессию, исходя из псевдоэкспериментальных данных, заданных на основании полинома третьей степени с известными коэффициентами:

$$i := 0..25 \qquad err := rnorm(26, 0, 1)$$

$$x_i := i \qquad y_i := 3\cdot(x_i)^3 - 2\cdot(x_i)^2 - 10x_i + 34 + err_i$$

$$pol\_reg(x, y, 3)^T = (34.326 \;\; -10.306 \;\; -1.969 \;\; 2.999)$$

Коэффициенты полинома найдены весьма точно. Значит, мы ни в чем не ошиблись.

Разобравшись в принципах вычисления полиномиальной регрессии по методу наименьших квадратов, изучим, как эту работу выполнить в Mathcad, применяя встроенные средства.

В Mathcad рассчитать полиномиальную регрессию можно с помощью встроенной функции regress(x,y,n), где x и y — векторы экспериментальных данных (количество точек должно быть больше порядка полинома хотя бы на 1), n — порядок полинома. В общем случае n может принимать совершенно любые значения, однако реально сталкиваться с полиномами выше 3–4 степени приходится редко.

По своим свойствам функция regress очень похожа на уже знакомые нам функции семейства *spline, используемые при интерполяции. Так, в векторе ее результата первые три строки будут служебными, а в остальных будут содержаться значения коэффициентов полинома, расположенные сверху вниз исходя из возрастания степени членов, к которым они относятся.

Для демонстрации работы функции regress решим посредством нее ту же задачу, что выше была решена с помощью нашей собственной программы pol_reg:

$$R := regress(x, y, 3) \qquad R^T = (3 \;\; 3 \;\; 3 \;\; 34.326 \;\; -10.306 \;\; -1.969 \;\; 2.999)$$

Результат, полученный функцией regress, в точности совпадает с результатом, выданным pol_reg. Следовательно, идеи в основе встроенной функции и нашей программы лежат одни и те же.

Построить кривую полинома регрессии можно, просто присвоив его коэффициентам значения соответствующих им элементов вектора R:

$$S(t) := \sum_{i=3}^{N+3} t^{i-3}\cdot R_i$$

Однако можно поступить и проще — задать полином с помощью функции interp(s,x,y,t). Для этого в качестве ее параметра s следует определить вектор коэффициентов, вычисленный функцией regress (именно для interp, кстати, в нем и имеются служебные строки).

$$S(t) := interp(R, x, y, t)$$

Когда полином регрессии будет определен, можно построить график (рис. 16.26). В нашем случае полезно сравнить кривую регрессии с кривой закономерности, исходя из которой были вычислены точки данных, для чего зададим функцию последней:

$$D(t) := 3 \cdot t^3 - 2 \cdot t^2 - 10 \cdot t + 34$$

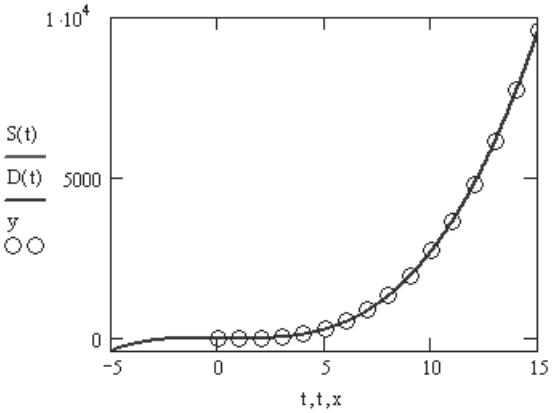


Рис. 16.26. Кривая «правильного» полинома, кривая полинома регрессии и «экспериментальные» точки

Изучив взаимное расположение кривых на рис. 16.26, вы, наверное, согласитесь, что системе удалось крайне неплохо провести среднюю кривую в области экспериментальных точек. И даже за пределами области точек кривые «правильного» полинома и полинома регрессии не расходятся. Впрочем, так бывает далеко не всегда.

Несоответствие полинома регрессии описывающему эмпирическую зависимость закону может быть связано с целым рядом причин — слабостями используемого для нахождения коэффициентов алгоритма, погрешностью, малым количеством точек. Универсального способа преодолеть все сложности не существует. Но уменьшить вероятность получения плохого результата все же можно. Для этого нужно либо увеличить точность измерений, либо расширить интервал локализации точек, либо увеличить количество точек. Например, вычислив регрессию от тех же 26 экспериментальных точек, однако распределенных на отрезке, в 10 раз менее широком, мы получим следующий вектор коэффициентов:

$$R^T = (3 \quad 3 \quad 3 \quad 34.326 \quad -13.06 \quad 1.099 \quad 2.214)$$

Как видите, уменьшение интервала локализации точек привело к существенному ухудшению в точности расчета коэффициентов полинома. При уменьшении же ширины отрезка в 100 раз результат не будет выдерживать никакой критики (рис. 16.27):

$$R^T = (3 \quad 3 \quad 3 \quad 34.326 \quad -40.6 \quad 307.897 \quad -783.419)$$

Почему же на точность полиномиальной регрессии так сильно влияет то, на каком расстоянии друг от друга находятся точки? Ответ на этот вопрос будет для вас очевиден, если в свое время вы внимательно прочитали главу, посвященную решению систем линейных уравнений. Как вы помните, точность решения СЛАУ в первую очередь за-

висит от того, насколько хорошо обусловлена матрица коэффициентов. Если матрица коэффициентов обусловлена плохо, то малые изменения в ее элементах или элементах вектора правых частей приводят к большим изменениям в решениях. В основе же вычисления полиномиальной регрессии лежит решение системы линейных уравнений. Причем матрица коэффициентов данной системы является классической плохо обусловленной матрицей (ее даже можно считать разновидностью матрицы Гильберта). Чем ближе будут лежать экспериментальные точки друг к другу, тем в меньшей степени будут отличаться соседние строки матрицы коэффициентов. Соответственно, тем хуже она будет обусловлена. Плохая же обусловленность и вызывает высокую погрешность при определении коэффициентов полинома.

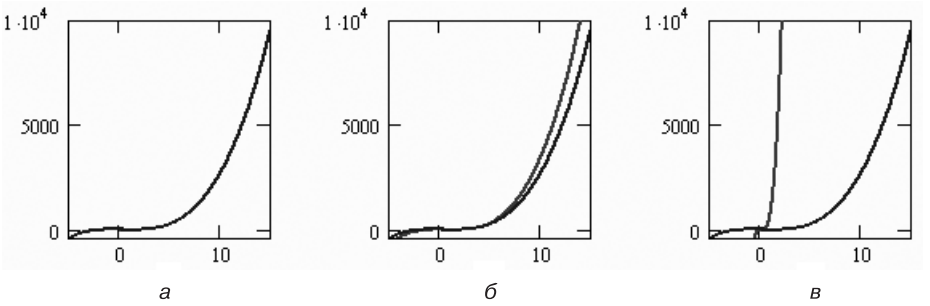


Рис. 16.27. Поведение кривых «правильного» полинома и полинома регрессии при шаге между абсциссами экспериментальных точек, равном 1 (*а*), 0.1 (*б*), 0.01 (*в*)

Плохая обусловленность матрицы коэффициентов в алгоритме расчета полиномиальной регрессии по методу наименьших квадратов — это самое слабое место данного алгоритма. Поэтому, чтобы не допустить ошибки, всегда нужно оценивать, насколько плохо обусловлена матрица коэффициентов. Для этого следует вычислить или определитель (чем он меньше, тем в большей степени вырождена матрица), или число обусловленности (чем оно больше, тем хуже обусловлена матрица). Немного модифицировав программу pol_reg, можно создать алгоритм, который будет решать эту задачу:

$$\text{pol\_reg\_cond}(x,y,n):=\left|\begin{array}{l}\text{for } i\in 0..n\\ \quad\text{for } j\in 0..n\\ \quad\quad M_{i,j}\leftarrow\sum_{k=0}^{\text{last}(x)}\left(x_k\right)^{i+j}\\ |M|\end{array}\right.$$

Расчет определителя матрицы коэффициентов для приведенного выше примера при шаге между точками, равном 1, дает значение $7\cdot10^{15}$. Если шаг равен 0.1, то значение определителя — $7\cdot10^{3}$. Если шаг уменьшить до 0.01, то определитель будет равен $7\cdot10^{-9}$. При уменьшении шага до 0.001 определитель уменьшится до 10^{-21}. Как видите, расстояние между точками очень сильно сказывается на вырожденности матрицы коэффициентов.

Если расчет определителя даст очень маленькое число, то нужно попробовать увеличить до максимума шаг между экспериментальными точками. Если же это невозможно, то можно использовать для решения СЛАУ сингулярное разложение. Часто это

дает лучший эффект по сравнению с применением стандартных методов решения систем линейных уравнений.

Кстати, обусловленность матриц типа матрицы Гильберта очень сильно зависит от их размерности. Чем больше матрица, тем она хуже обусловлена. Поэтому нужно крайне осторожно относиться к расчету регрессии для полинома степени, выше 4-й. Вероятность плохого результата при этом намного превышает вероятность ошибки при вычислении регрессии для полиномов 2–4-й степени. К счастью, полиномы высоких степеней в приложениях встречаются редко.

Если вы не знаете точного математического вида измеряемой закономерности, принципиальным условием успеха может быть правильный выбор степени полинома регрессии. Кажется очевидным то, что чем выше степень полинома, тем лучше он должен приблизить данные со сложным поведением. Однако это утверждение справедливо лишь частично. Если данные по своей природе не полиномиальные, то результатом расчета регрессии может быть сильно осциллирующая кривая. Эта осцилляция (она называется полиномиальным раскачиванием) может свести на нет физическую корректность кривой регрессии. Почти всегда полиномиальное раскачивание проявляет себя, если регрессия рассчитывается для полинома 5-й степени или выше. Поэтому не стоит применять полиномы высокой степени, если достоверно неизвестно, что данные имеют полиномиальную природу.

Продемонстрируем явление полиномиального раскачивания на примере. Зададим псевдоэкспериментальные данные, базирующиеся на экспоненциальной функции $y=e^{-x}$. Затем рассчитаем на основании их коэффициенты полинома 6-й степени. Построив графики полинома и экспоненциальной функции (рис. 16.28), посмотрим, в достаточной ли степени они себя схоже ведут хотя бы в области локализации точек.

$$i := 0..13 \qquad err := rnorm(26, 0, 0.3)$$

$$x_i := 0.2 \cdot i \qquad y_i := e^{-x_i} + err_i$$

$$R := regress(x, y, 6) \qquad S(t) := interp(R, x, y, t)$$

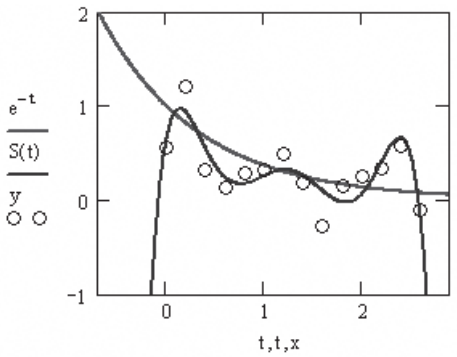


Рис. 16.28. Осцилляция полинома высокой степени при расчете регрессии на основании данных не полиномиальной природы

В том, что приближать данные не полиномиальной природы полиномом можно далеко не всегда, мы убедились. Проблема заключается в том, что полином низкой степени

не способен отобразить сложные зависимости, имеющие несколько экстремумов. Полином же высокой степени имеет ярко выраженную склонность к осцилляции. Но можно ли как-то преодолеть описанные сложности, «поймав двух зайцев»? Это реально. При построении средней кривой для экспериментальных данных, математического закона для которых вы не знаете даже приблизительно, самым оптимальным ходом будет использование регрессии с помощью отрезков полиномов. Принцип такой регрессии чем-то схож с идеями сплайн-интерполяции и заключается в построении средней кривой из фрагментов нескольких полиномов низкой степени, каждый из которых задается на основании всего нескольких точек на относительно узком интервале.

Регрессия фрагментами полиномов второй степени строится в Mathcad с помощью системной функции loess(x,y,span), где x и y — векторы эмпирических данных, span — величина, определяющая длину отрезков для полиномов. Последний параметр предельно важен, так как его правильное задание определяет эффективность регрессии. Давать какие-то точные рекомендации по выбору величины span очень сложно, так как наилучшее его значение может весьма значительно изменяться в зависимости от шага между точками (чем он меньше, тем меньше, чаще всего, должен быть span), уровня погрешности (чем он больше, тем больше должен быть span), от вида усредняемой зависимости. В справочной системе Mathcad в качестве оптимального значения span называется число 0.75, однако опыт показывает, что эта величина параметра подходит далеко не всегда. Так, если значение span выбрано слишком большим, то результат не будет отличаться от обычной регрессии полиномом 2-й степени, а при слишком малой его величине кривая будет скорее напоминать интерполяционную, нежели регрессивную (или результат вообще не будет получен). В общем случае, проводя регрессию фрагментами полиномов, вы должны поэкспериментировать с различными значениями span и выбрать из них то, которое даст наиболее подходящий, как подскажет ваш опыт и интуиция, результат.

Функция loess вычисляет только вектор с коэффициентами соответствующих полиномов. Чтобы построить на его основании кривую регрессии, нужно задействовать функцию interp. Делается это точно так же, как в случае функции regress, так что еще раз останавливаться на этом вопросе мы не будем.

Приведем пример построения кривой регрессии для синусоидально распределенных данных с помощью простой полиномиальной регрессии и регрессии фрагментами полиномов.

Пример 16.8. Регрессия фрагментами полиномов (рис. 16.29)

$$N := 30$$

$$Err := rnorm(N, 0, 0.5)$$

$$i := 0..N-1 \qquad x_i := -2\pi + 4\pi \cdot \frac{i}{N-1} \qquad y_i := 3 \cdot \sin\left(x_i\right) + Err_i$$

$$R := loess(x, y, 0.3) \qquad Rp := regress(x, y, 5)$$

$$Reg(t) := interp(R, x, y, t) \qquad Regp(t) := interp(Rp, x, y, t)$$

Изучив приведенный пример, вы, наверное, согласитесь, что в случае специфических задач регрессия отрезками полиномов намного эффективнее, чем простая полиномиальная регрессия. Так, кривая, построенная с помощью функции loess, практически идеально совпадает с графиком, соответствующим закону, по которому распределены эмпирические данные, чего нельзя сказать о полиноме 5-й степени, заданном функцией regress.

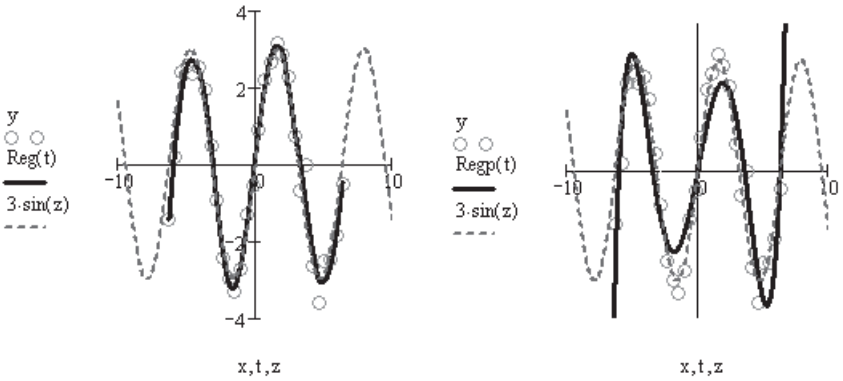


Рис. 16.29. Полиномиальная регрессия (справа) и регрессия отрезками полиномов (слева)

Впрочем, успех регрессии фрагментами полиномов в примере 16.8 оказался возможным благодаря удачному подбору параметра span. Чтобы подтвердить это, приведем примеры средних кривых для других его значений (рис. 16.30).

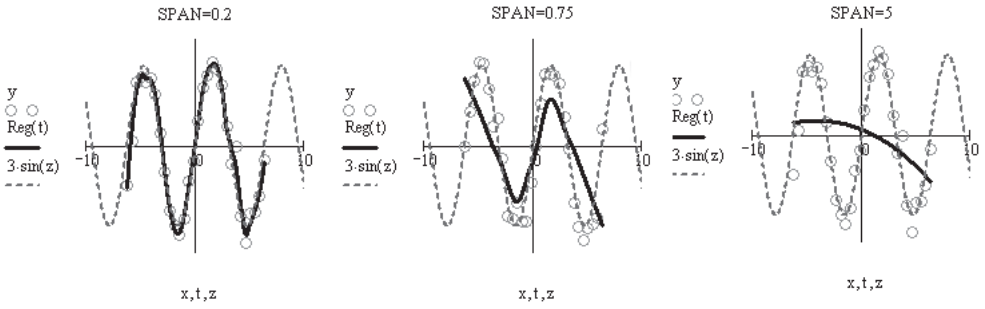


Рис. 16.30. Регрессия фрагментами полиномов при разной величине параметра span

Сравнив кривые на рис. 16.30 с левым графиком на рис. 16.29, вы, наверное, согласитесь, что регрессия, проведенная при span=0.3, была самой эффективной. Кстати, обратите внимание на то, что средняя кривая, построенная при рекомендованной в справочной системе программы величине span=0.75, не выдерживает никакой критики в плане соответствия истинному закону, по которому распределены эмпирические данные. А это говорит о том, что рекомендация создателей Mathcad в данном случае отнюдь не универсальна.

16.3.3. Многомерная полиномиальная регрессия

Обработка результатов измерения характеристики, зависящей от двух (или более) переменных, — это задача на порядок более трудоемкая, чем вычисление простой одномерной регрессии. Однако если у вас есть Mathcad, то в большинстве случаев ее можно решить очень просто, используя встроенные алгоритмы двумерной полиномиальной регрессии и регрессии отрезками полиномов.

Так как многомерная полиномиальная регрессия есть обобщение одномерной регрессии, то все, о чем мы говорили в предыдущем подразделе об особенностях расчета ко-

эффициентов полинома по методу наименьших квадратов, можно автоматически перенести на многомерный случай. Поэтому в данном подразделе мы рассмотрим лишь технику вычисления многомерной полиномиальной регрессии, не уделяя внимание математическим деталям.

С приближением выборки данных некоторой поверхностью мы уже встречались, когда разбирали многомерную сплайн-интерполяцию. Двумерная полиномиальная регрессия строится практически аналогично, однако некоторые, весьма существенные и неочевидные, технические детали при этом все же имеются. Поэтому стоит рассмотреть последовательность задания усредняющей поверхности по пунктам.

1. Для начала зададим выборку псевдоэкспериментальных данных. Так как подобную работу мы уже не раз выполняли, приведем лишь соответствующий листинг (включая рис. 16.31), не комментируя его:

$$N := 6$$

$$i := 0..N-1 \qquad j := 0..N-1$$

$$x_{i,j} := \frac{10\cdot i}{N-1} - 5 \qquad y_{i,j} := 10\cdot\frac{j}{N-1} - 5$$

$$Err := rnorm\left(N^2, 0, 0.3\right)$$

$$z_{i,j} := \left(x_{i,j}\right)^2 + \left(y_{i,j}\right)^2 + \left(x_{i,j} + y_{i,j}\right)\cdot Err_{i\cdot j+i}$$

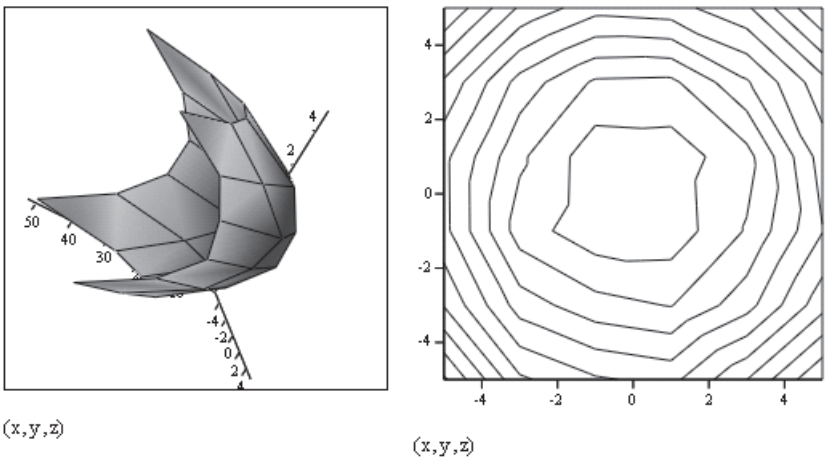


Рис. 16.31. Поверхность, построенная исходя из необработанной выборки

2. Далее требуется определить вектор коэффициентов полинома регрессии с помощью уже хорошо знакомой нам функции regress(XY,Z,n), где XY — матрица значений переменных, Z — вектор значений эмпирически определяемой величины, n — порядок соответствующего полинома. В матрице XY должно быть столько столбцов, функцией скольких переменных является искомый полином. На практике почти всегда встречаются полиномы от двух переменных.

   Если, проводя аналогию с многомерной интерполяцией, вы попытаетесь задать в матрице XY только значения элементов, лежащих на главной диагонали матриц

переменных, а Z сохранить в форме квадратной матрицы, то работать функция regress не будет. Все дело в том, что параметры в случае двумерной регрессии определяются по-другому. Так, матрица XY должна быть образована двумя столбцами, в первом из которых должны располагаться координаты экспериментальных точек по оси абсцисс, а во втором — по оси ординат (впрочем, очередность задания переменных не играет в трехмерном случае особой роли). Их последовательность следует задать такой, как будто они прочитаны из соответствующих матриц так, как мы читаем книгу: слева направо и сверху вниз. Аналогичным образом в виде вектора должна быть представлена и матрица опытных значений Z.

В том, что данные при вычислении двумерной регрессии организуются по другому принципу по сравнению с интерполяцией, есть определенный смысл. Так, например, это позволяет обрабатывать выборки, точки которых располагаются не в узлах какой-то правильной сетки, а произвольным образом. Подобный подход значительно расширяет возможности двумерной регрессии.

Если ваши данные организованы в виде матриц (а так оно чаще всего и бывает), то преобразовать их в нужную форму можно, написав две простенькие программы:

$$R := \left| \begin{array}{l} n \leftarrow 0 \\ \text{for } i \in 0..\text{last}\left(x^{\langle 0 \rangle}\right) \\ \quad \text{for } j \in 0..\text{last}\left(x^{\langle 0 \rangle}\right) \\ \quad\quad \left| \begin{array}{l} R1_n \leftarrow x_{i,j} \\ R2_n \leftarrow y_{i,j} \\ n \leftarrow n + 1 \end{array} \right. \\ R \leftarrow \text{augment}(R1, R2) \\ R \end{array} \right. \qquad Z := \left| \begin{array}{l} n \leftarrow 0 \\ \text{for } i \in 0..\text{last}\left(z^{\langle 0 \rangle}\right) \\ \quad \text{for } j \in 0..\text{last}\left(z^{\langle 0 \rangle}\right) \\ \quad\quad \left| \begin{array}{l} Z_n \leftarrow z_{i,j} \\ n \leftarrow n + 1 \end{array} \right. \\ Z \end{array} \right.$$

Когда данные будут организованы соответствующим образом, можно определить вектор коэффициентов полинома регрессии:

$$S := \text{regress}(R, Z, 2)$$

3. Чтобы теперь задать функцию, соответствующую рассчитанным параметрам регрессии, следует задействовать функцию interp. Очень важным и тонким моментом является то, что она должна быть определена как зависимость от двух переменных, что можно сделать занесением на место ее четвертого параметра вектора из двух строк, содержащих соответствующие символы:

$$I(t, k) := \text{interp}\left[S, R, Z, \begin{pmatrix} t \\ k \end{pmatrix}\right]$$

4. Далее зададим график как поверхность быстрого построения (рис. 16.32).

Вы, наверное, согласитесь, что для приведенной задачи алгоритм двумерной регрессии справился с проблемой усреднения данных безупречно. Однако в случае более сложных зависимостей полиномиальная регрессия может быть неэффективна. Если поверхность, соответствующая вашим данным, имеет многочисленные экстремумы или явно не полиномиальную природу, то для построения регрессии лучше использовать алго-

ритм отрезков полиномов. Задается такая регрессия точно так же, как полиномиальная, только вместо функции regress нужно использовать функцию loess.

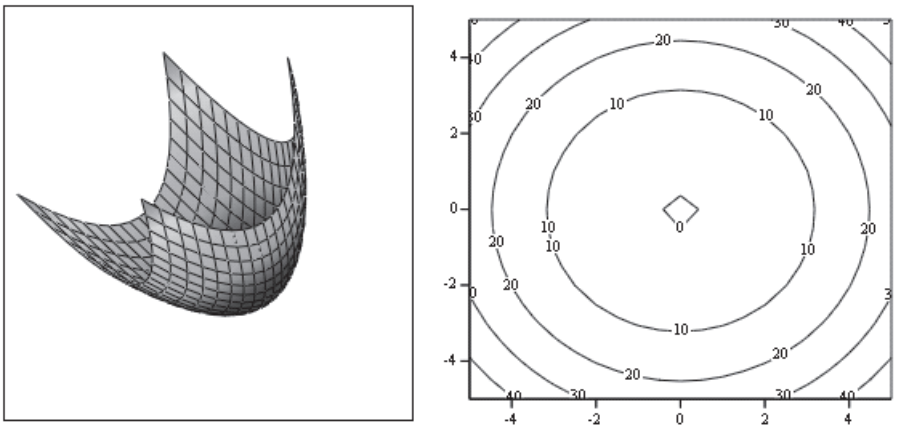


Рис. 16.32. Результат двумерной полиномиальной регрессии

16.3.4. Обобщенная линейная регрессия

Линейная регрессия предназначена для определения коэффициентов линейной функции y=a·x+b, которая наилучшим образом приближает экспериментальные данные. Задача обобщенной линейной регрессии — ответить на следующий вопрос: какие значения должны принимать коэффициенты a_0, a_1, ..., a_N, чтобы функция F(x), являющаяся линейным сочетанием N+1 произвольной функции $f_0(x)$, $f_1(x)$, ..., $f_N(x)$ (то есть $F(x)=a_0 \cdot f_0(x)+a_1 \cdot f_1(x)+...+a_N \cdot f_N(x)$), проходила между экспериментальными точками так, чтобы сумма квадратов расстояний от точек и до кривой F(x) была минимальной?

На первый взгляд задача обобщенной линейной регрессии кажется очень сложной для решения. На самом же деле это не так. Вывести ее рабочую формулу легко по тому же принципу, как раньше нами была выведена формула для расчета полиномиальной регрессии. Действительно, полином степени N есть не что иное, как линейное сочетание функций $f_0(x)=1$, $f_1(x)=x$, $f_2(x)=x^2$, ..., $f_N(x)=x^N$. Зная это, рабочую формулу полиномиальной регрессии легко преобразовать в формулу для расчета обобщенной линейной регрессии простой заменой x^k на $f_k(x)$, а x^p — на $f_p(x)$:

$$\sum_{n=0}^{N} c_n \cdot \sum_{k=0}^{K} f_n(x_k) \cdot f_p(x_k) = \sum_{k=0}^{K} y_k \cdot f_p(x_k)$$

На основании полученной формулы создадим программу для вычисления обобщенной линейной регрессии. В качестве параметров она будет принимать векторы экспериментальных данных x и y, а также вектор-функцию F, элементами которой будут являться функции, образующие линейное сочетание. Результатом работы программы будет вектор, содержащий коэффициенты для всех функций линейного сочетания. Расположены коэффициенты в векторе ответа будут в том же порядке, что и соответствующие им функции в вектор-функции F.

$$\text{gen\_lin\_reg}(x,y,F) := \left| \begin{array}{l} \text{for } i \in 0..\text{last}(F(x)) \\ \quad \text{for } j \in 0..\text{last}(F(x)) \\ \quad \left(M_{i,j} \leftarrow \sum_{k=0}^{\text{last}(x)} F(x_k)_i \cdot F(x_k)_j \quad N_i \leftarrow \sum_{k=0}^{\text{last}(x)} y_k \cdot F(x_k)_i \right) \\ \text{lsolve}(M,N) \end{array} \right.$$

Чтобы проверить эффективность написанной программы, сгенерируем последовательность псевдоэкспериментальных данных, основанных на функции, являющейся линейным сочетанием четырех разнородных функций:

$$N := 30$$

$$i := 0..N \qquad err := rnorm(31,0,3)$$

$$x_i := -5 + 0.3 \cdot i \qquad y_i := 5 \cdot \sin(x_i) + 3 \cdot \cos(2x_i) + 4 \cdot \sqrt[3]{x_i} + 5 \cdot x_i + err_i$$

Далее зададим описывающую линейное сочетание функций вектор-функцию F:

$$F(x) := \left(\sin(x) \quad \cos(2x) \quad \sqrt[3]{x} \quad x\right)^T$$

Производим расчет коэффициентов:

$$\text{gen\_lin\_reg}(x,y,F)^T = (4.104 \quad 3.054 \quad 5.728 \quad 4.251)$$

Коэффициенты, с учетом специфичности задачи и высокого уровня погрешности, были найдены довольно неплохо. Кривая, построенная на их основании, ведет себя практически так же, как «правильная» кривая (см. ниже).

В Mathcad для вычисления обобщенной линейной регрессии служит встроенная функция linfit(x,y,F), где x и y — векторы эмпирических данных, F — векторная функция, содержащая в качестве элементов функции, входящие в линейное сочетание. Попробуем с ее помощью найти коэффициенты для той же зависимости, регрессию для которой мы уже рассчитали посредством программы gen_lin_reg:

$$\text{linfit}(x,y,F)^T = (4.104 \quad 3.054 \quad 5.728 \quad 4.251)$$

Как видите, результаты, выданные gen_lin_reg и linfit, в точности совпадают. Следовательно, разработчики Mathcad использовали тот же алгоритм, что и мы.

Нам осталось только построить графики «правильной» функции и функции регрессии (рис. 16.33), чтобы сравнить, насколько схоже они себя ведут. Создать необходимые для этого функции проще всего, скалярно перемножив вектор коэффициентов (исходных или вычисленных алгоритмом регрессии) и вектор-функцию F:

$$Z(t) := \text{linfit}(x,y,F) \cdot F(t) \qquad B(t) := (5 \quad 3 \quad 4 \quad 5)^T \cdot F(t)$$

Полиномиальная регрессия — это частный случай обобщенной линейной регрессии. Соответственно, те слабости, которые присущи алгоритму полиномиальной регрессии, будут характерны и для обобщенной линейной регрессии. Чтобы не получить результат с большой ошибкой, следует контролировать то, достаточно ли хорошо обусловлена матрица коэффициентов, в программе gen_lin_reg имеющая название M. Она будет тем хуже обусловлена, чем больше функций входит в линейное сочетание и чем на бо-

лее узком интервале локализованы экспериментальные точки. Также многое зависит от природы и свойств функций, образующих линейное сочетание.

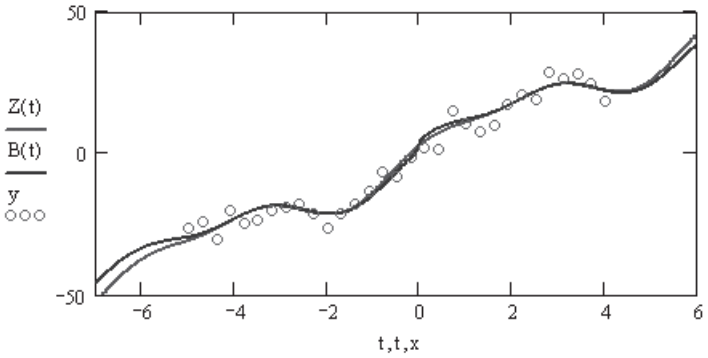


Рис. 16.33. «Правильная» кривая, кривая регрессии и экспериментальные точки

16.3.5. Регрессия общего вида

Как правило, производя какие-то измерения, например, на лабораторной работе по физике, вы знаете математический вид теоретического закона, описывающего изучаемое вами явление. Конечно, в большинстве случаев для обработки полученных данных можно использовать уже рассмотренные нами линейную и полиномиальную регрессии. Однако очень часто физические закономерности характеризуются и более сложными формулами, содержащими, например, синус или экспоненту. Можно ли методом наименьших квадратов находить параметры подобных зависимостей? В принципе, да, хотя это гораздо сложнее, а результат будет ошибочным с гораздо более высокой вероятностью. Обычно эта задача сводится к решению системы нелинейных уравнений или поиску минимума функции нескольких переменных. Соответственно, трудности в определении регрессии общего вида будут обусловлены слабостями соответствующих численных алгоритмов.

Продемонстрируем на примере то, как может быть реализован алгоритм регрессии для определения параметров зависимости произвольного вида. Представим, что в результате эксперимента была получена последовательность точек $\{x_k, y_k\}$, $k \in 0...K$, причем известно, что изучаемое явление может быть описано следующей функцией:

$$y = a \cdot \sin(w \cdot x + t)$$

Нам необходимо найти, при каких значениях параметров a, w, t сумма квадратов расстояний от кривой до экспериментальных точек будет минимальной. Чтобы ответить на этот вопрос, зададим для начала функцию ошибки приближения E:

$$E(a, w, t) = \sum_{k=0}^{K} \left(a \cdot \sin\left(x_k + t\right) + w - y_k\right)^2$$

Наилучшему приближению будет соответствовать минимум функции E. Чтобы найти координаты точки минимума, вспомним обязательное условие экстремума функции нескольких переменных: все частные производные функции должны равняться 0. Вычислим частные производные E по всем переменным:

$$\frac{\partial}{\partial a} E(a,w,t) = \sum_{k=0}^{K} 2\cdot\left(a\cdot\sin\left(x_k + t\right) + w - y_k\right)\cdot\sin\left(x_k + t\right) = 0$$

$$\frac{\partial}{\partial t} E(a,w,t) = \sum_{k=0}^{K} 2\cdot\left(a\cdot\sin\left(x_k + t\right) + w - y_k\right)\cdot a\cdot\cos\left(x_k + t\right) = 0$$

$$\frac{\partial}{\partial w} E(a,w,t) = \sum_{k=0}^{K} \left(2\cdot a\cdot\sin\left(x_k + t\right) + 2\cdot w - 2\cdot y_k\right) = 0$$

Решив полученную систему нелинейных уравнений, мы найдем оптимальные значения параметров a, w, t.

Проверим наши выводы на практике. Для этого сгенерируем на основании функции с известными параметрами псевдоэкспериментальную зависимость:

$$N := 100$$

$$a := 2 \qquad w := 1 \qquad t := 1$$

$$i := 0..N \qquad err := rnorm(N + 1, 0, 0.1)$$

$$x_i := -\pi + 2\pi\cdot\frac{i}{N} \qquad y_i := a\cdot\sin\left(x_i + t\right) + w + err_i$$

Решаем систему уравнений из частных производных функции E посредством блока Given-Find. Начальные приближения следует взять достаточно близкими к истинным значениям параметров, так как система, ввиду того, что в нее входят периодические функции, имеет бесконечное множество решений (поэтому плохие приближения могут привести к нахождению не тех корней).

$$K := last(x)$$

$$a := 3 \qquad t := 2 \qquad w := 0$$

Given

$$\sum_{k=0}^{K}\left(a\cdot\sin\left(x_k + t\right) + w - y_k\right)\cdot\sin\left(x_k + t\right) = 0 \qquad \sum_{k=0}^{K}\left(a\cdot\sin\left(x_k + t\right) + w - y_k\right)\cdot a\cdot\cos\left(x_k + t\right) = 0$$

$$\sum_{k=0}^{K}\left(a\cdot\sin\left(x_k + t\right) + w - y_k\right) = 0$$

$$find(a,t,w)^T = (1.997 \;\; 1.009 \;\; 0.986)$$

Итак, параметры зависимости удалось найти достаточно точно. Действуя аналогичным образом, можно вычислять регрессию для зависимостей совершенно произвольного вида с любым количеством параметров. Однако, конечно, нужно учитывать, что чем больше будет у зависимости параметров, тем больше в системе будет уравнений и, соответственно, тем сложнее будет ее правильно решить.

Кстати, существует еще один путь нахождения параметров нелинейной регрессии: можно искать минимум функции ошибки E непосредственно, используя один из встроенных в Mathcad численных алгоритмов нелинейной оптимизации (доступ к ним предоставляет функция Minimize). Попробуем таким способом решить ту же задачу, кото-

рая выше была решена посредством поиска корней системы уравнений из частных производных функции E:

$$a := 3 \quad w := 0 \quad t := 2$$

$$E(a, w, t) := \left[\sum_{k=0}^{last(x)} \left(y_k - a \cdot \sin\left(x_k + t \right) - w \right)^2 \right]$$

$$Minimize(E, a, t, w)^T = (1.997 \quad 0.986 \quad 1.009)$$

Результаты расчета параметров регрессии обоими способами в точности совпали. И в этом нет ничего удивительного: чисто математически данные способы идентичны, просто в случае использования функции Minimize Mathcad выполняет за вас часть операций.

Чтобы пользователям, далеким от математики, не приходилось выполнять такие сложные операции, как задание функции ошибки или системы уравнений из ее частных производных, разработчики Mathcad встроили в программу несколько функций, позволяющих вычислять регрессию методом наименьших квадратов для наиболее распространенных на практике нелинейных зависимостей. Перечислим эти функции:

- expfit(x,y,g) — регрессия экспоненциальной функцией $f(x)=a \cdot e^{b \cdot x}+c$;
- lgsfit(x,y,g) — регрессия логистической функцией $f(x)=a/(1+b \cdot e^{-c \cdot x})$;
- sinfit(x,y,g) — синусоидальная регрессия $f(x)=a \cdot \sin(x+b)+c$;
- pwrfit(x,y,g) — регрессия степенной функцией $f(x)=a \cdot x^b+c$;
- logfit(x,y,g) — регрессия логарифмической функцией $f(x)=a \cdot \ln(x+b)+c$;

Наверное, вы уже догадались, что параметры x и y приведенных функций соответствуют векторам координат эмпирических данных. А вот о том, что стоит за параметром g, стоит поговорить подробнее.

Все дело в том, что Mathcad рассчитывает регрессию общего вида точно так же, как это делали мы: используя алгоритмы численной оптимизации, основанные на численных методах решения систем нелинейных уравнений. Численные же методы решения систем нелинейных уравнений, как вы помните, требуют задания начальных приближений к корням. В случае функций регрессии эти приближения вы передаете в векторе g. Они должны быть приведены в следующей последовательности: a, b, c. К выбору начальных приближений стоит отнестись очень ответственно, так как близость значений вектора g к истинным величинам параметров зависимости как ничто другое определяет эффективность вычисления регрессии.

Приводить пример вычисления всех типов специальной регрессии мы не будем, так как для всех функций его принципы абсолютно идентичны. Мы рассчитаем лишь синусоидальную регрессию для выборки, параметры которой выше уже были определены нашим собственным алгоритмом:

$$sinfit\left[x, y, \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} 1.997 \\ 1.009 \\ 0.986 \end{pmatrix}$$

Сравнивая результаты, выданные sinfit и нашим алгоритмом, легко убедиться, что они полностью совпадают. Следовательно, в основе sinfit лежат те же идеи, которые использовали мы.

Самостоятельно создавая алгоритм расчета регрессии общего вида, можно найти параметры зависимости совершенно любого вида. В Mathcad есть встроенная функция genfit(x,y,g,F), которая дает возможность справляться с этой задачей немного проще. В качестве аргументов данная функция принимает следующие параметры:

- x — вектор опытных данных по оси абсцисс;
- y — эмпирические данные по оси ординат;
- g — вектор приближений для неизвестных параметров. Правила его задания такие же, как при вычислении регрессии специального вида;
- F(x, g) — вектор-функция из n+1 элемента, где n — количество рассчитываемых параметров. Первый элемент данной функции — это описывающая экспериментальную зависимость функция, параметры которой должны быть рассчитаны. Сами параметры должны фигурировать в выражении как соответствующие элементы вектора g (см. пример 16.9). Следующие n элементов вектор-функции F должны быть заполнены выражениями частных производных описывающей зависимость функции по искомым параметрам. Последовательность частных производных должна быть такой же, как последовательность приближений к соответствующим им параметрам в векторе g.

Из всех рассмотренных нами встроенных функций регрессии genfit является наиболее универсальной, однако и наименее точной и трудной в использовании. Как и для всех остальных регрессий специального вида, огромное влияние на точность расчета неизвестных параметров с помощью функции genfit оказывает близость значений элементов вектора g к истинным их величинам. Кроме того, в отличие от всех остальных встроенных функций регрессии, точность результата вычислений с помощью genfit зависит от значения системной переменной TOL.

Приведем пример расчета параметров эмпирической зависимости с помощью функции genfit.

Пример 16.9. Расчет регрессии для произвольной зависимости с использованием функции genfit

Попробуем применить функцию genfit для расчета параметров функции следующего вида:

$$y(x) = \frac{a}{\sin(x + b) + c}$$

Для начала сгенерируем на основании данной функции последовательность псевдоэкспериментальных данных, положив параметрам следующие значения: a=2, b=1, c=2.

$$N := 30$$

$$a := 2 \qquad b := 1 \qquad c := 2$$

$$i := 0..N \qquad err := rnorm(N + 1, 0, 0.1)$$

$$x_i := -2\pi + 4\pi \cdot \frac{i}{N} \qquad y_i := \frac{a}{\sin(x_i + b) + c} + err_i$$

Зададим вектор приближений для параметров g:

$$g := (1 \quad 2 \quad 1)^T$$

Чтобы задать вектор-функцию F, вычислим сначала частные производные функции y по всем трем параметрам:

$$\frac{d}{dg_0}\frac{g_0}{\sin(z+g_1)+g_2}\rightarrow\frac{1}{\sin(z+g_1)+g_2}\qquad \frac{d}{dg_2}\frac{g_0}{\sin(z+g_1)+g_2}\rightarrow\frac{-g_0}{(\sin(z+g_1)+g_2)^2}$$

$$\frac{d}{dg_1}\frac{g_0}{\sin(z+g_1)+g_2}\rightarrow\frac{-g_0}{(\sin(z+g_1)+g_2)^2}\cdot\cos(z+g_1)$$

Чтобы выражения производных получить в той форме, в которой они должны быть занесены в вектор-функцию F, используем для обозначения параметров имена g_0, g_1, g_2. Важной тонкостью является то, что индексы в именах должны быть текстовыми, а не матричными. Иначе расчет производных сделан не будет.

Создаем вектор-функцию F (индексы в именах параметров должны быть матричными):

$$F(z,g):=\left[\frac{g_0}{\sin(z+g_1)+g_2}\ \frac{1}{\sin(z+g_1)+g_2}\ \frac{-g_0}{(\sin(z+g_1)+g_2)^2}\cdot\cos(z+g_1)\ \frac{-g_0}{(\sin(z+g_1)+g_2)^2}\right]^T$$

Производим расчет регрессии, задействовав функцию genfit:

$$\text{genfit}(x,y,g,F)^T=(1.968\ \ 1.089\ \ 1.997)$$

Итак, значения параметров функции, которая легла в основу псевдоэкспериментальной зависимости, были определены genfit довольно неплохо. Посмотрим, достаточно ли схоже ведут себя «правильная» кривая и кривая регрессии (рис. 16.34).

$$res(z):=\frac{1.968}{\sin(z+1.089)+1.997}\qquad tr(z):=\frac{2}{\sin(z+1)+2}$$

Заканчивая разговор на тему расчета регрессии в Mathcad, хотелось бы дать один совет: выполняя обработку экспериментальных данных, старайтесь делать это двумя-тремя альтернативными способами (благо, возможности программы это позволяют). И всегда, когда это возможно, не склоняясь перед авторитетом создателей Mathcad, выполняйте проверку. Такой подход особенно важен в области обработки эмпирических данных, так как для результатов, полученных в такого рода расчетах, характерна известная доля субъективности.

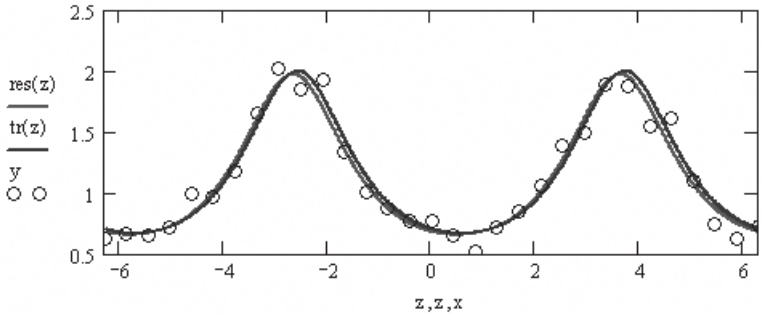


Рис. 16.34. «Правильная» кривая и кривая регрессии

16.4. Ковариация и корреляция

В статистике для оценки силы корреляционной зависимости двух случайных величин используется коэффициент корреляции r_{xy}. Определяется он отношением математического ожидания произведения отклонений случайных величин от их средних значений к произведению среднеквадратичных отклонений этих величин:

$$r_{xy} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)\cdot\left(y_i - \overline{y}\right)}{\sigma_x \cdot \sigma_y}$$

В Mathcad коэффициент корреляции двух выборок по данной формуле можно подсчитать с помощью встроенной функции corr(x,y).

Если коэффициент корреляции равен по модулю единице, то между случайными величинами существует линейная зависимость. Если же он равен нулю, то случайные величины независимы. Промежуточные значения r_{xy} говорят о том, что две выборки коррелируют в некоторой степени.

Также в статистике иногда используют понятие ковариации. Ковариация, это, по сути, не нормированная корреляция. Так как на практике ковариация используется, главным образом, для расчета корреляции, подробно о ней мы говорить не будем. В Mathcad есть функция ковариации — cvar(x,y).

Расчет корреляции полезен во многих случаях. Мы продемонстрируем, как можно использовать данный статистический параметр при решении задач регрессии.

Пример 16.10. При наблюдении за изменением значений признака y при различных величинах фактора x были получены следующие данные:

x	1	10	20	30	40	50	60	70
y	12,8	62,8	68,9	75,2	81,9	89,3	97,1	105,5

Методом наименьших квадратов определить вид кривой, описывающей зависимость y от x

Прежде всего, зададим имеющиеся данные в виде векторов:

$$x := (1 \;\; 10 \;\; 20 \;\; 30 \;\; 40 \;\; 50 \;\; 60 \;\; 70)^T$$

$$y := (12.8 \;\; 62.8 \;\; 68.9 \;\; 75.2 \;\; 81.9 \;\; 89.3 \;\; 97.1 \;\; 105.5)^T$$

Нам необходимо выяснить, к какому типу относится зависимость между полученными эмпирическими точками. Для этого нанесем их на координатную плоскость. Отдельно построив зависимости, наиболее часто встречающиеся на практике ($y=\ln(x)$, $y=e^x$, $y=x^a$, $y=a/x$), можно предположить, что лучше всего точки ложатся на логарифмическую кривую (рис. 16.35).

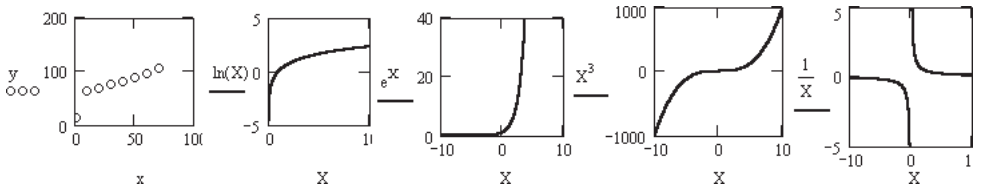


Рис. 16.35. Приближенное определение зависимости между эмпирическими точками

Таким образом, задача сводится к нахождению коэффициентов уравнения y=a·ln(b·x+c)+d. Чтобы воспользоваться функцией line, представим данную зависимость как линейную: y=a·z+b, где z= ln(b·x+c), b=d, а также зададим начальные приближения для b и c.

$$b := 2 \qquad c := 1$$

$$R := \text{line}(\ln(b \cdot x + c), y) \qquad R = \begin{pmatrix} -11.165 \\ 22.226 \end{pmatrix}$$

Меняя приближения параметров b и c, мы получаем абсолютно разные значения элементов вектора R, которые являются соответственно коэффициентами d и a в уравнении регрессии. Как же оценить, какое приближение для данной зависимости окажется оптимальным? Для этого определим коэффициент корреляции векторов z и y:

$$\text{corr}(\ln(b \cdot x + c), y) = 0.98733$$

Очевидно, существуют такие значения b и c, при которых коэффициент r_{zy} максимален. Чтобы найти их, представим коэффициент корреляции как функцию от b (рис. 16.36):

$$b := 0..50$$

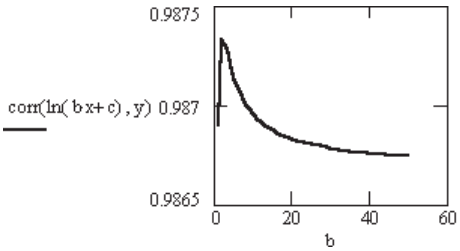


Рис. 16.36. Зависимость величины коэффициента корреляции от значения параметра b

График наглядно демонстрирует, что функция $f(b)=r_{zy}(b)$ имеет максимум. Определим его:

$$b := 2$$

$$f(b) := \text{corr}(\ln(b \cdot x + c), y)$$

$$\text{Maximize}(f, b) = 2.08$$

Значит, при c=1 наилучшим приближением для коэффициента b (с учетом округления) является 2. Если же поменять приближение параметра c, оптимум для b также изменится, но, заметьте, коэффициент корреляции при этом останется максимальным.

Осталось записать полученное уравнение регрессии в стандартном виде и визуализировать его (рис. 16.37).

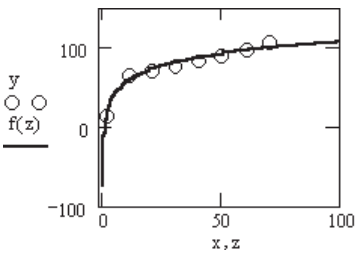


Рис. 16.37. Кривая, описывающая эмпирическую зависимость y-x

Таким образом, экспериментальные данные можно описать логарифмической зависимостью вида y(x)=22,226·ln(2·x+1)–11,165.

Обратите внимание, минимизация ошибки функцией line производилась между векторами ln(bx+c) и y, а не x и y, что могло привнести в конечный результат незначительную погрешность. Однако вероятность того, что вы получите аналогичную по величине погрешность при использовании любого другого вида регрессии, ничуть не меньше.

Проанализируем рассмотренный пример. Меняя приближения параметров уравнения регрессии, мы фактически получаем новую зависимость, аппроксимирующую, однако, эмпирические точки с такой же точностью. Возникает вопрос, какая же именно кривая наилучшим образом описывает опытные данные? С подобной проблемой можно столкнуться не только при непосредственном анализе результатов эксперимента, но и при решении задач — расхождение с ответом в последнем случае (а вероятность возникновения такой ситуации довольно высока) может вызвать ряд сомнений в правильности результата, полученного с помощью Mathcad.

В действительности же нужно помнить, во-первых, о том, что анализируемые нами случайные величины находятся между собой не в строго функциональной, а в так называемой стохастической зависимости: изменение величины X влечет за собой изменение распределения величины Y, которая подвержена влиянию всевозможных случайных факторов. Другими словами, каждому значению X мы не можем поставить в соответствие одно и только одно значение Y: определяемая нами стохастическая зависимость является корреляционной — каждому значению X соответствует не абсолютное, а некоторое среднее значение Y.

Во-вторых, представленные в учебниках экспериментальные данные, как правило, группируются в виде корреляционных таблиц. Определяя по ним коэффициенты уравнения регрессии на бумаге, удобно составлять специальные расчетные таблицы, содержащие усреднения признака Y для каждой из вариант X в отдельности, а не для всей выборки в целом. Такой подход вносит в конечный результат некоторую ошибку. Также погрешность может возникнуть при непосредственном поиске в Mathcad коэффициентов из системы уравнений, к решению которой сводится сущность метода наименьших квадратов. Дело в том, что классический метод Гаусса, с помощью которого в задачниках предлагается найти неизвестные параметры, крайне нерационально использовать в Mathcad без привлечения программных средств. Поэтому даже если вы попытаетесь решить такую систему, выполнив вручную все предварительные операции, то коэффициенты, найденные одним из численных методов Mathcad, могут отличаться от приводимых в ответе.

В-третьих, не следует забывать о том, что через конечное количество экспериментальных точек всегда можно провести бесконечное множество кривых, каждая из которых будет описывать опытные данные с удовлетворительной точностью. Поэтому любую из зависимостей, полученных как при подсчете на бумаге, так и при использовании функции line, нельзя интерпретировать содержательно: каждая из них будет являться лишь приближенной, а не истинно теоретической, поскольку за пределами интервала наблюдения она может вести себя весьма неадекватно.

Итак, в случае, если абсолютное совпадение результата с ответом является для вас принципиальным, вам придется выполнять всю рекомендуемую авторами учебников последовательность действий вручную, безусловно, потратив на это немалое количество времени. Если же вы хотите определить вид зависимости между экспериментальными точками с максимальной точностью и минимальными временными затратами, то в этом Mathcad окажет вам неоценимую помощь. Практически во всех случаях

при поиске коэффициентов уравнения криволинейной регрессии алгоритм, приведенный в примере 16.10, оказывается эффективным.

16.5. Сглаживание данных

Очень важной задачей в радиотехнике и электронике является устранение шумов — высокочастотных случайных составляющих в сигнале. Алгоритмы, позволяющие решить эту задачу, называются алгоритмами сглаживания (smoothing).

Конечно, можно попробовать удалить случайную составляющую из сигнала и с помощью вычисления, например, синусоидальной регрессии или, лучше, регрессии фрагментами полиномов. Однако такой подход не всегда может быть эффективным по той причине, что все-таки регрессия — это очень грубый вычислительный инструмент, и ее использование может уничтожить информативную составляющую в сигнале, особенно если он сложен или быстро изменяется. Немаловажным фактором является и то, что расчет регрессии — это достаточно сложная вычислительная задача, особенно при большом объеме эмпирических данных.

По изложенным выше причинам для сглаживания сигнала на практике обычно применяются не регрессивные методы, а более мягкие и простые алгоритмы фильтрации. В Mathcad имеются три встроенные функции, позволяющие произвести сглаживание некоторой выборки данных:

- `medsmooth(y,n)`, где `y` — вектор значений сигнала, `n` — параметр, определяющий количество окон сглаживания, на которое будет разбит интервал при обработке данных (`n` может быть только нечетным целым числом, строго меньшим, чем количество элементов в выборке). Эта функция реализует популярный алгоритм «бегущих» медиан (running medians), весьма подробное описание которого вы можете найти в справочной системе программы. Обязательным условием при ее применении является то, что эмпирические точки должны быть равномерно распределены на промежутке. Из всех встроенных функций сглаживания Mathcad `medsmooth` является наиболее надежной, однако и наименее универсальной функцией;
- `ksmooth(x,y,b)`, где `x` и `y` — векторы данных, `b` — ширина окна сглаживания (этот параметр по величине должен равняться общей величине нескольких промежутков, разделяющих в данной выборке соседние точки). Данная встроенная функция реализует сглаживание на основании алгоритма Гаусса (его описание вы можете прочитать в соответствующей статье справочной системы Mathcad). Лучше всего функция `ksmooth` подходит для устранения шумов в стационарном сигнале;
- `supsmooth(x,y)`, где `x` и `y` — векторы данных. Осуществляет сглаживание с помощью адаптивного алгоритма (в основе которого лежит метод наименьших квадратов), основанного на анализе взаимного расположения рассматриваемой точки и ближайших к ней (их количество зависит от особенностей поведения графика зависимости). Данная функция лучше всего подходит для сглаживания сложных нестационарных сигналов.

Все описанные выше функции возвращают в качестве ответа вектор из такого же количества элементов, как в исходных выборках. Поэтому если эмпирических точек было обработано немного, то при построении графика, для получения корректной кривой, стоит использовать возможности сплайн-интерполяции Mathcad (если же сглаживалась зависимость, образованная более чем 50–100 точками, то этого, скорее всего, делать не потребуется).

Попробуем оценить эффективность функций сглаживания Mathcad. Для этого, используя генератор нормально распределенных случайных чисел, имитируем сигнал со значительным уровнем шума. Строя кривые усреднения при разных величинах окна сглаживания, определим степень влияния этого параметра на качество результата.

Помимо сглаживания с помощью соответствующих встроенных функций, мы продемонстрируем, как эту работу можно выполнить и с помощью алгоритмов регрессии (на примере регрессии фрагментами полиномов).

Пример 16.11. Функции сглаживания (рис. 16.38–16.42)

$$N := 500 \qquad Err := rnorm(N, 0, 1)$$

$$i := 0..N-1 \qquad x_i := 5\cdot\pi\cdot\frac{i}{N-1} \qquad y_i := \sin(x_i) + \cos(x_i) + Err_i\cdot(\sin(x_i) + \cos(x_i))$$

$$f(t) := \sin(t) + \cos(t)$$

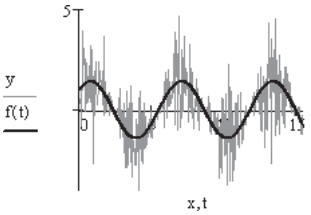


Рис. 16.38. Исходный сигнал

$$s1 := medsmooth(y, 11) \qquad s2 := medsmooth(y, 55) \qquad s3 := medsmooth(y, 101)$$

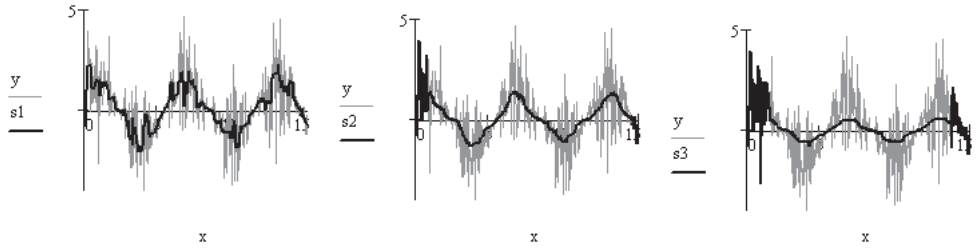


Рис. 16.39. Сглаживание по методу «бегущих» медиан

$$k1 := ksmooth(x, y, 2) \qquad k2 := ksmooth(x, y, 0.1) \qquad k3 := ksmooth(x, y, 0.5)$$

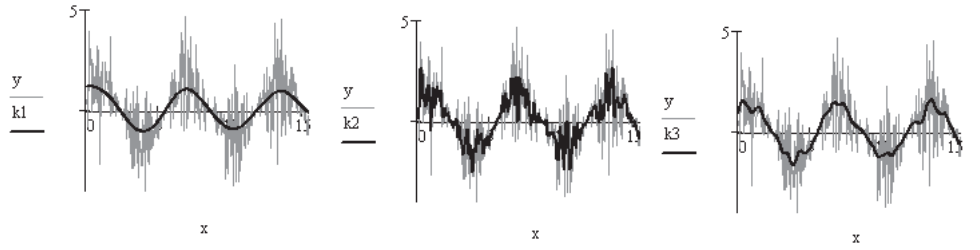


Рис. 16.40. Сглаживание на основе алгоритма Гаусса

$$S := \text{supsmooth}(x, y)$$

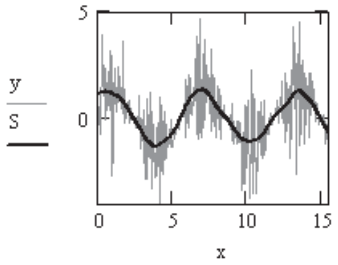


Рис. 16.41. Сглаживание с помощью адаптивного алгоритма

$$P := \text{loess}(x, y, 0.3) \qquad I(t) := \text{interp}(P, x, y, t)$$

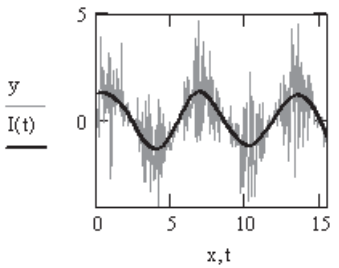


Рис. 16.42. Сглаживание с помощью регрессии отрезками полиномов

Из приведенного примера можно заключить, что принципиальным условием успеха при сглаживании сигнала может быть правильный выбор алгоритма (так, например, в случае нашей зависимости метод «бегущих» медиан оказался не на высоте). Кроме того, весьма тщательно нужно относиться и к заданию величины окна сглаживания.

Реально алгоритмов, позволяющих решать всевозможные задачи сглаживания и фильтрации, существует гораздо больше, чем встроено в систему Mathcad в виде функций. И любой из них при необходимости может быть весьма просто реализован вами благодаря высокому вычислительному потенциалу системы. Так, приведем пример создания весьма простого и эффективного метода сглаживания, известного как скользящее усреднение. Изучая его, вы оцените, насколько проще в вычислительном плане алгоритмы сглаживания по сравнению с той же регрессией.

Пример 16.12. Скользящее усреднение (рис. 16.43)

$$N := 1000 \qquad Err := rnorm(N, 0, 1)$$

$$i := 0..N - 1$$

$$x_i := 5 \cdot \pi \cdot \frac{i}{N - 1} \qquad y_i := \sin(x_i) + \cos\left[3 \cdot (x)_i\right] + Err_i \cdot \left[\sin\left[3 \cdot (x)_i\right] + \cos(3 \cdot x_i)\right]$$

$$f(t) := \sin(t) + \cos(3t)$$

$$w2 := 150 \qquad w1 := 30$$

$$r1_i := if\left(i < w1, \frac{\sum_{j=0}^{i} y_j}{i+1}, \frac{\sum_{j=i-w1+1}^{i} y_j}{w1}\right)$$

$$r2_i := if\left(i < w2, \frac{\sum_{j=0}^{i} y_j}{i+1}, \frac{\sum_{j=i-w2+1}^{i} y_j}{w2}\right)$$

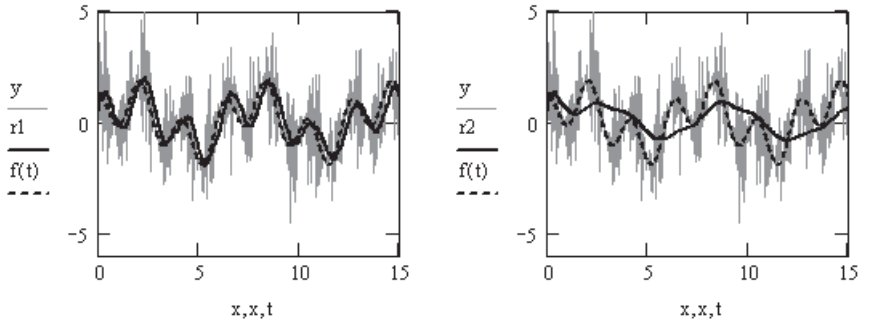


Рис. 16.43. Сглаживание скользящим усреднением

Обратите внимание, что кривая сглаживания, полученная с помощью скользящего усреднения (рис. 16.43), несколько опережает исходную кривую сигнала. При этом по форме они абсолютно идентичны. Попробуйте объяснить причину этого явления, и, может,вам удастся внести в наш алгоритм модернизации, позволяющие избавиться от этого недостатка.

16.6. Интегральные преобразования

Интегральные преобразования играют весьма значительную роль в прикладной математике. Особенно важны преобразование Фурье и вейвлетное преобразование, значимость которых для электроники и радиотехники трудно переоценить. В Mathcad имеются специальные функции, реализующие численные алгоритмы для этих преобразований. Об особенностях их использования мы и поговорим в этом разделе.

16.6.1. Преобразование Фурье

Математический смысл преобразования Фурье состоит в представлении некоторой функции f(t) в виде бесконечной суммы тригонометрических функций (спектра) вида F(v)·sin(t·v), где, в случае реальных процессов, t — это время, v — частота соответствующей составляющей сигнала f(t). Функция F(v) может быть определена как:

$$F(v) = \int_{-\infty}^{\infty} f(t) \cdot e^{i \cdot v \cdot t}\, dt$$

Приведенная формула имеет скорее теоретическое, нежели практическое значение, так как она позволяет проводить преобразование лишь над аналитически заданными зависимостями. При обработке же сигналов приходится работать с векторами числовых значений, и к ним данная формула, естественно, неприменима. Обычно используется одна из разновидностей численного алгоритма быстрого преобразования Фурье (БПФ).

Этот метод весьма несложен в вычислительном плане, однако позволяет получать очень точные и корректные результаты. Чтобы продемонстрировать, что в численном расчете спектра Фурье нет ничего трудного, приведем пример задания соответствующего алгоритма.

Пример 16.13. Алгоритм БПФ (рис. 16.44)

$$N := 2^8 \qquad N = 256$$

$$Err := rnorm(N, 0, 1) \qquad i := 0..N-1$$

$$\nu 1 := \frac{1}{10} \qquad \nu 2 := \frac{1}{5}$$

$$y_i := \sin(2\pi\cdot\nu 1\cdot i) + \cos(2\pi\cdot\nu 2\cdot i) + Err_i\cdot(\sin(2\pi\cdot\nu 1\cdot i) + \cos(2\pi\cdot\nu 2\cdot i))$$

$$n := 0..\frac{N}{2} \qquad F_n := \frac{1}{\sqrt{N}}\cdot\sum_{k=0}^{N-1} y_k\cdot e^{\frac{i\cdot 2\pi\cdot k\cdot n}{N}}$$

$$j := 0..\frac{N}{2} \qquad v_j := \frac{j}{N}$$

Обратите внимание, что полученный в примере 16.13 Фурье-спектр имеет два резко выделяющихся максимума (рис. 16.44, справа), положения которых соответствуют частотам двух периодических составляющих заданного нами сигнала. Для подобного рода разделения и используется на практике преобразование Фурье, так как это позволяет не только решать проблемы, связанные с фильтрацией, но и даже производить предсказание поведения зависимости и справляться со многими другими задачами. Более низкие пики в спектре соответствуют шумовой составляющей, и в том случае, если бы сигнал был задан без погрешности, их бы не было.

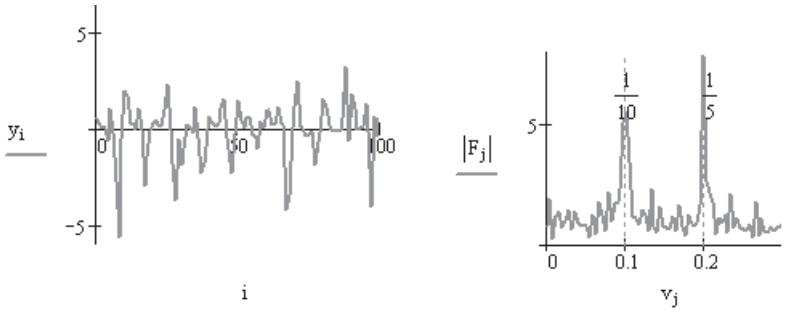


Рис. 16.44. Исходный сигнал и его Фурье-спектр

Кстати, построить график Фурье-спектра можно только на основании модулей его значений, так как они являются комплексными величинами.

Как было продемонстрировано выше, задать алгоритм БПФ в Mathcad можно предельно просто. Однако гораздо легче и лучше использовать встроенные функции прямого и обратного преобразования Фурье:

- fft(y), где y — вектор равномерно распределенных по промежутку измеренных значений изучаемого сигнала. Возвращает вектор прямого преобразования Фурье (эта функция реализует приведенную выше разновидность алгоритма БПФ);
- ifft(c), где c — вектор равномерно распределенных значений Фурье-спектра. Обратное преобразование Фурье;
- FFT(y) — эта функция реализует алгоритм БПФ в несколько другой нормировке, чем fft:

$$F_n = \frac{1}{N} \cdot \sum_{k=0}^{N-1} y_k \cdot e^{\frac{2\pi \cdot i \cdot k \cdot n}{N}}$$

- IFFT(c) — обратное преобразование Фурье для спектра, полученного на основании FFT.

Если вы собираетесь произвести анализ сигнала с помощью численного преобразования Фурье, то вы должны знать следующее.

- Количество элементов в векторе значений сигнала должно точно равняться 2^n (n — целое число, больше либо равное 2). Если данных меньше, то просто нарастите соответствующий вектор за счет нулевых значений — на корректность результата это не окажет никакого влияния.
- В качестве результата прямого преобразования Фурье система выдает вектор из $2^{n-1}+1$ элемента. Это обстоятельство следует учитывать, если вы собираетесь провести обратное преобразование или построить график спектра.
- Результатом обратного преобразования Фурье вектора спектра из $2^{n-1}+1$ элементов будет являться вектор из 2^n строк, значения элементов которого будут крайне близки к исходному сигналу (если он, конечно, был достаточно информативен).
- Причиной того, что результатом работы приведенных функций прямого преобразования Фурье является вектор, количество элементов которого в два раза меньше исходной выборки, является то, что в случае действительных данных Фурье-спектр симметричен. А это означает, что нет смысла отображать вторую половину спектра, полностью дублирующую первую.

Приведем пример расчета прямого и обратного Фурье-преобразования для сигнала с заданным с помощью генератора случайных чисел шумом с использованием соответствующих встроенных функций Mathcad.

Пример 16.14. Встроенные функции прямого и обратного преобразования Фурье (рис. 16.45, 16.46)

$$N := 2^{10} \qquad N = 1.024 \times 10^3$$

$$i := 0 .. N - 1 \qquad Err := rnorm(N, 0, 1)$$

$$v1 := \frac{1}{4} \qquad v2 := \frac{1}{5} \qquad v3 := \frac{1}{6}$$

$$y_i := \sin(2\pi \cdot v1 \cdot i) + \cos(2\pi \cdot v2 \cdot i) + \sin(2\pi \cdot v3 \cdot i) + Err_i \cdot (\sin(2\pi \cdot v1 \cdot i) + \cos(2\pi \cdot v2 \cdot i))$$

$$F := \mathrm{fft}(y)$$

$$N1 := \mathrm{last}(F) \qquad N1 = 512$$

$$j := 0..\,N1 \qquad v_j := \frac{j}{N}$$

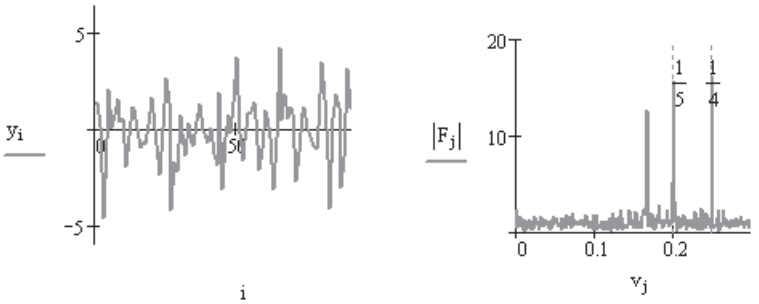


Рис. 16.45. Исходный сигнал и его Фурье-спектр

$$IF := \mathrm{ifft}(F) \qquad \mathrm{length}(IF) = 1.024\times10^{3} \qquad \max(y - IF) = 1.138\times10^{-11}$$

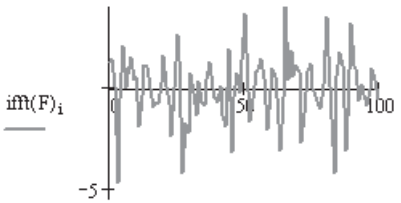


Рис. 16.46. Результат обратного преобразования Фурье

Обратите внимание, насколько точными являются численные методы преобразования Фурье: наибольшее отклонение, которое появилось в векторе ifft(F) по сравнению с вектором исходных данных, лежит в области 11-го знака после запятой. А это очень и очень мало даже для расчетов на компьютере.

Если обрабатываемые данные являются комплексными величинами, использовать описанные выше функции нельзя. Однако в Mathcad существуют встроенные функции, позволяющие проводить преобразование Фурье и над комплексным сигналом. Запомнить их очень легко, так как все они получаются с помощью добавления буквы «c» к именам соответствующих функций действительного преобразования:

- cfft(y) — комплексное преобразование Фурье в стандартной нормировке;
- CFFT(y) — вектор прямого комплексного преобразования Фурье с другим условием нормировки (см. выше функцию FFT(y));
- icfft(c) — обратное комплексное преобразование Фурье;
- ICFFT(c) — обратная функция для CFFT(y).

Как и в случае действительного преобразования Фурье, вектор данных для функций cfft(y) и CFFT(y) должен содержать строго 2^n (n>2) элементов. Однако в связи с тем, что

преобразование Фурье для комплексных данных не является симметричным, результатом работы указанных функций будет вектор, образованный 2^n, а не $2^{n-1}+1$ элементами.

При использовании функций комплексного преобразования Фурье по отношению к действительной выборке вектор результата будет образован двумя симметричными половинами, одна из которых была бы отброшена, используй вы функции действительного преобразования.

На самом же деле, если разобраться, то преобразование Фурье не совсем корректно делить на комплексное и действительное. Так, в основе функций ifft и icfft лежит один и тот же алгоритм БПФ, различно только количество подсчитываемых элементов вектора спектра.

Чтобы лучше разобраться в различиях простого и комплексного преобразования, используем соответствующие функции по отношению к действительной выборке.

Пример 16.15. Комплексное преобразование Фурье (рис. 16.47)

$$N := 2^8 \qquad N = 256$$

$$i := 0..\,N-1 \qquad Err := rnorm(N, 0, 1)$$

$$v1 := \frac{1}{10} \qquad v2 := \frac{1}{3}$$

$$y_i := \sin(2\pi \cdot v1 \cdot i) + \cos(2\pi \cdot v2 \cdot i) + Err_i \cdot \left(\sin\left(2\pi \cdot \frac{i}{10}\right) + \cos\left(\frac{i}{5}2\pi\right)\right)$$

$$F := cfft(y)$$

$$N1 := last(F) \qquad N1 = 255$$

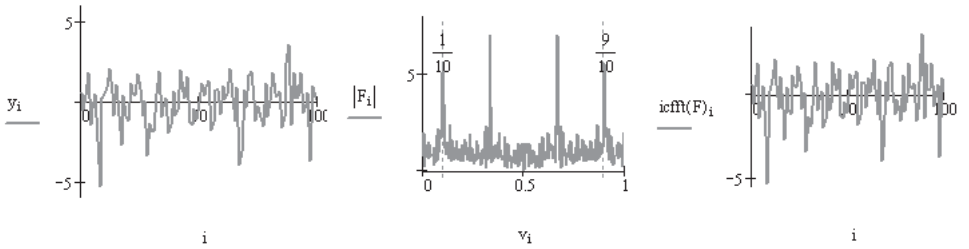


Рис. 16.47. Комплексное преобразование Фурье

$$\max(y - icfft(F)) = 2.22 \times 10^{-15} + 4.815i \times 10^{-15}$$

Как можно видеть на рис. 16.47, при комплексном преобразовании Фурье применительно к действительным данным два значимых пика зеркально отображаются относительно оси x=0.5. Подумайте, почему так происходит и какова была бы картина, если бы данные были комплексными.

В Mathcad существует возможность проведения преобразования Фурье не только над выборкой в виде вектора, но и над матрицами. Правда, использовать для этого можно

только более общие функции комплексного преобразования (cfft, CFFT, icfft, ICFFT). Функции действительного преобразования могут работать только с векторами.

16.6.2. Вейвлетное преобразование

Бурное развитие электроники и радиотехники во второй половине XX века заставило математиков искать более простые и надежные средства для различного рода обработки сигналов, чем давало использование численного преобразования Фурье. Различных аналогов ему было предложено очень много, однако наиболее широкое распространение получили так называемые вейвлетные (дискретные волновые) преобразования. При обработке нестационарных сигналов и при высоком уровне шумов такие преобразования показали себя намного более эффективными, чем преобразование Фурье.

В основу дискретных волновых преобразований положены специальные функции (называемые вейвлетообразующими), главным достоинством которых является то, что они ограничены в некоторой узкой области интервала, вне которого бесконечно приближаются к нулю. Синусоиды же, используемые в преобразовании Фурье, осциллируют на всем промежутке, что ухудшает качество обработки сигнала.

В Mathcad существуют две функции, реализующие численный расчет одного из видов вейвлетного преобразования на основе фильтра Даубечи:

- wave(y), где y — вектор значений сигнала. Прямое вейвлет-преобразование;
- iwave(c), где c — вектор значений соответствующего спектра. Обратное вейвлетное преобразование.

Количество элементов вектора значений y в случае вейвлетного преобразования, аналогично преобразованию Фурье, должно обязательно равняться 2^n (где n — целое положительное число больше 2).

Ввиду специфичности рассматриваемой проблемы подробно о ней мы говорить не будем и лишь приведем пример расчета прямого и обратного вейвлетного преобразования. Исчерпывающие сведения о применении этого интегрального преобразования вы можете получить в любом современном справочнике по спектральному анализу.

Пример 16.16. Вейвлетное преобразование (рис. 16.48, 16.49)

$$N := 1024 \qquad v2 := \frac{1}{15} \qquad v1 := \frac{1}{12}$$

$$i := 0..N-1 \qquad y_i := \sin(2\pi \cdot v1 \cdot i) + \sin(2\pi \cdot v2 \cdot i)$$

$$W := wave(y)$$

$$Nlevels := \frac{\ln(N)}{\ln(2)} - 1 \qquad Nlevels = 9$$

$$coeffs(n) := submatrix\left(W, 2^n, 2^{n+1} - 1, 0, 0\right)$$

$$k := 1..Nlevels \qquad C_{i,k} := coeffs(k)_{floor\left[\frac{i}{\left(\frac{N}{2^k}\right)}\right]}$$

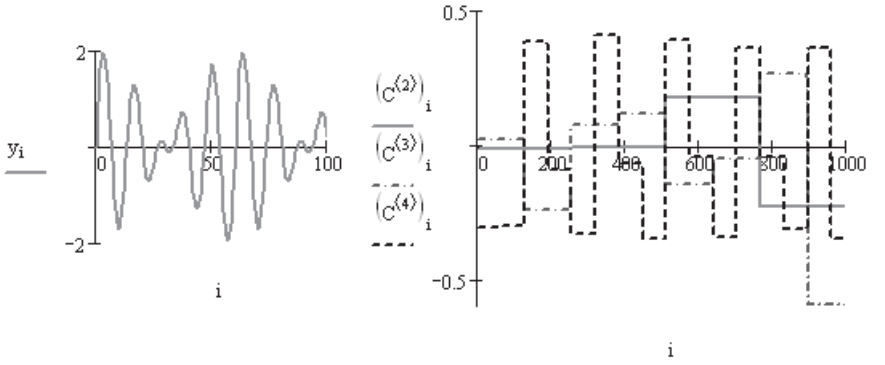


Рис. 16.48. Исходные данные и их вейвлет-спектр

$$j := 2^{Nlevels} .. N - 1 \qquad W_j := 0$$

$$Iw := iwave(W)$$

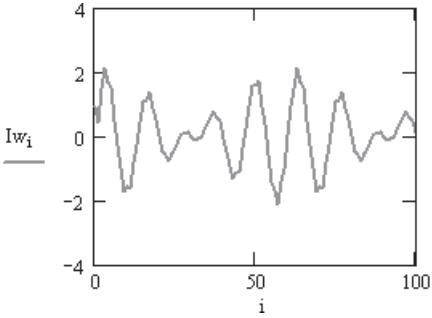


Рис. 16.49. Результат обратного вейвлетного преобразования

Кстати, приведенный пример демонстрирует одно очень интересное приложение вейвлетного преобразования — сжатие данных. Как вы можете видеть (см. рис. 16.48, 16.49), исключение одной составляющей сказывается на виде восстановленного сигнала весьма незначительно (хотя определенная потеря в гладкости все же заметна).

16.7. Импорт внешних данных

Нельзя не согласиться, что Mathcad предоставляет замечательные возможности в области обработки данных. Множество встроенных статистических функций, функции интерполяции и регрессии, разнообразие графиков и поверхностей позволяют наилучшим образом представить полученные вами результаты измерений. Самая трудная задача при этом — перевод данных в доступный для Mathcad вид. Конечно, если измерений было проведено немного, то соответствующие матрицы и векторы можно создать и непосредственно, заполнив их вручную. Однако если данных много, то такой способ может быть неэффективен.

Значительно облегчить задачу обработки данных может то, что Mathcad способен считывать информацию из текстовых файлов. Вообще, текстовые форматы очень распро-

странены и такого рода файлы создаются как при работе моделирующих алгоритмов на C или Фортране, так и программами, управляющими всевозможными измерительными приборами. Кроме того, текстовый файл вы можете создать, распознав с помощью специальной программы (например, FineReader) отсканированный документ.

Помимо текстовых файлов, Mathcad умеет отлично взаимодействовать с такими программами, как Excel, Matlab и некоторыми другими. Также в Mathcad имеются возможности по считыванию данных в битовом формате, работе с изображениями и звуками.

В рамках этой книги мы познакомимся с тем, как можно считывать данные из текстовых файлов и документов Excel. Остальные способы импорта данных, ввиду специфичности, мы разбирать не будем. При необходимости достаточную информацию вы сможете получить в справочной системе программы.

16.7.1. Функции работы с текстовыми файлами

Mathcad способен читать как структурированные, так и неструктурированные текстовые файлы. При этом в первом случае результатом будет стандартный для программы вектор, во втором — соответствующая матрица.

С помощью Mathcad можно не только прочитать, но и записать текстовый файл. Кстати, практически любая широко используемая программа может как экспортировать, так и импортировать текстовые файлы. Учитывая это, описываемая возможность Mathcad может оказаться весьма и весьма полезной.

Наиболее простым для восприятия системой форматом текстового файла является неструктурированный текстовый документ. Термин «неструктурированный» означает, что в приведенном документе важна лишь последовательность числовых значений, а не их принадлежность к тому или иному столбцу или строке. Такие текстовые файлы генерируются, например, управляющими различными измерительными установками программами, которые записывают данные в строку последовательно друг за другом. Когда лимит длины строки оказывается превышен, текстовый редактор автоматически осуществляет перевод на следующую. Открыв неструктурированный текстовый файл, вы увидите простую последовательность чисел. Более сложным случаем является чтение структурированного текстового файла, так как при этом приходится учитывать принадлежность значения к определенным строке или столбцу. Структурированные текстовые файлы встречаются на практике гораздо чаще, чем неструктурированные, поэтому особенности использования соответствующих специальных функций мы опишем исключительно на примерах.

Чтобы прочитать данные из текстового документа, используйте специальные встроенные функции READPRN("file") и READFILE("file", "type"), где file — это имя текстового файла (если он сохранен в том же каталоге, что и Mathcad) или путь к файлу (если он расположен, например, на Рабочем столе), type — тип данных (если в каждом столбце содержатся числа с одинаковым количеством знаков после запятой, параметр type следует задать как fixed (фиксированная ширина столбцов), в противном случае — как delimited (при этом столбцы данных в файле должны быть отделены друг от друга только одинаковыми разделителями, например пробелом или знаками табуляции)). С помощью этих функций можно импортировать данные не только из файлов с расширением PRN, но и из файлов любых других форматов, использующих кодировку ASCII: DAT, TXT и пр.

Чтобы в Mathcad было возможным импортирование данных из текстового файла, нужно учитывать следующие правила.

- Числа в текстовом файле должны быть разделены либо пробелами (одним или несколькими), либо запятыми (но ни в коем случае не точками), либо символами табуляции.
- Для отделения целой части числа от десятичной следует использовать точку, а не запятую. Особое внимание на этот факт стоит обратить, если вы собираетесь использовать данные, полученные в русифицированной версии Excel, так как в числах она использует принятый в странах бывшего СССР формат с запятой.
- Одинаково эффективно Mathcad может считывать как целые, так и десятичные числа. Однако следует помнить, что количество знаков в них не должно превышать 17. Числа со степенью должны быть представлены в инженерном формате (например, числу $1.234 \cdot 10^{-6}$ в нем соответствует запись 1.234E-6). Чтобы записать в текстовый файл комплексные числа, их действительные и мнимые части должны быть заданы по отдельности. Сформировать же затем по полученным при прочтении текстового файла векторам соответствующий вектор комплексных чисел можно довольно просто, организовав цикл с помощью оператора ранжированной переменной.
- Если вы собираетесь прочитать структурированный текстовый файл с помощью функции READPRN, то учтите, что количество элементов во всех строках должно быть одинаковым. В противном случае система выдаст сообщение об ошибке: Can't understand data file (Невозможно распознать данные в файле). При чтении же файла с помощью функции READFILE на месте недостающего в строке элемента появится константа NaN (рис. 16.50).
- Пустые строки и столбцы, содержащие ASCII-текст, при считывании игнорируются.

Программа Mathcad может не только импортировать, но и создавать на основании матриц структурированные PRN-файлы. Для этого в систему встроена специальная функция WRITEPRN(«file»), где file — имя создаваемого файла. Интересной особенностью этой функции является то, что не она присваивается (как было в случае всех изученных нами ранее функций), а, наоборот, ей присваивается значение некоторой матрицы. Например, запись WRITEPRN(«C:\Samples\M.prn»):=M означает, что соответствующий матрице M ASCII-файл M.prn будет создан в папке Samples диска C:.

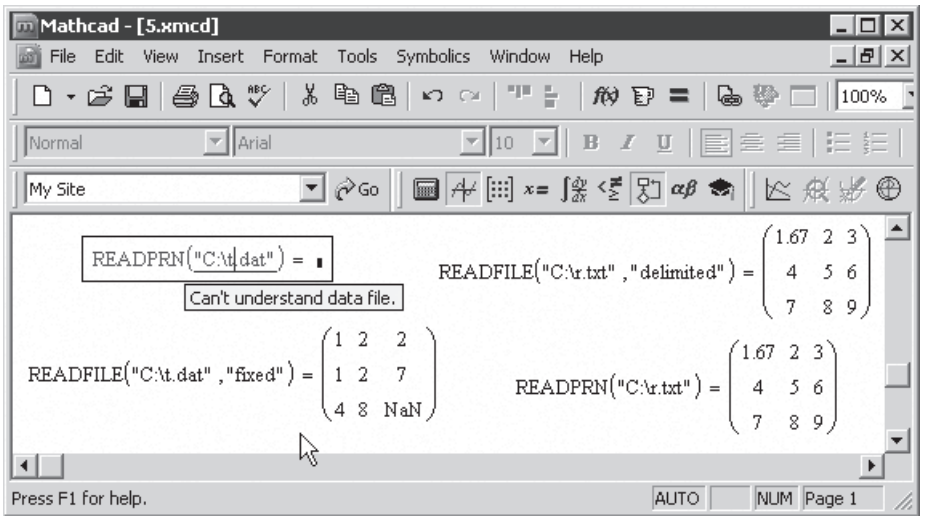


Рис. 16.50. Считывание структурированного текстового файла

Работая с функцией WRITEPRN, нужно учитывать следующие факты.

- Если вы пропишете только имя файла без пути к нему, WRITEPRN сохранит его в каталоге программы.
- WRITEPRN может создавать файлы не только с расширением prn, но и в любом другом текстовом формате, например DAT или TXT. Расширение создаваемого файла вы задаете, когда прописываете его имя в скобках рассматриваемой функции.
- Если файл с указанным для WRITEPRN именем уже существует, то он автоматически заменяется без обычного для Windows предупреждения.
- В записанном с помощью WRITEPRN текстовом файле значения будут разделены пробелами. В качестве десятичного знака будет использована точка.
- По умолчанию количество знаков импортируемых из Mathcad численных данных ограничено 4. Хотя это и соответствует обычным правилам отображения чисел в Mathcad, в ряде случаев подобный подход может быть неприемлем. Если вам нужно экспортировать данные с более высокой точностью, вы можете увеличить количество сохраняемых знаков. Чтобы это сделать, обратитесь к вкладке Builds-In Variables (Системные переменные) диалогового окна Worksheet Options (Параметры документа) меню Tools (Инструменты). Здесь, в окошке параметра PRNPRECISION, задайте необходимый уровень точности. Очевидно, что наибольшая величина PRNPRECISION должна равняться 16.
- Увеличивая точность экспортируемых данных, вы должны учитывать, что по умолчанию ширина столбца создаваемого WRITEPRN текстового файла соответствует восьми знакам. Следовательно, если PRNPRECISION превышает это значение, то ширину столбца следует увеличить (если этого не сделать, данные не сольются, однако значения в соседних столбцах будут расположены вплотную друг к другу, что значительно снижает наглядность). Сделать это можно с помощью параметра PRNCOLWIDTH вкладки Build-In Variables (Системные переменные). Величина ширины столбца в Mathcad не лимитируется.
- Соответствующие определения для PRNPRECISION и PRNCOLWIDTH можно сделать и непосредственно в документе (аналогично TOL и CTOL), что гораздо проще и техничнее, чем использование вкладки Build-In Variables (Системные переменные) диалогового окна Worksheet Options (Параметры документа).
- Если в исходной матрице содержались элементы с порядком, то в созданном с помощью WRITEPRN файле они будут представлены в инженерном формате.

Иногда требуется не создать новый текстовый файл, а лишь добавить новую информацию к уже существующему. В Mathcad задачи такого рода можно решать с помощью функции APPENDPRN(«file»), где file — имя файла (или путь к нему), к которому нужно дописать сведения. По особенностям своего синтаксиса функция APPENDPRN полностью соответствует функции WRITEPRN, поэтому останавливаться на этом вопросе мы не будем. Естественным условием при записи значений матрицы в конец некоторого текстового файла является то, что количество их столбцов должно совпадать.

16.7.2. Компоненты

Компонентами в Mathcad называются приложения или объекты из других приложений, которые могут принимать непосредственное участие в работе создаваемого вычислительного алгоритма. Учитывая глобальность распространенности таких систем, как Excel или MatLab, возможность Mathcad осуществлять с ними согласованную работу просто неоценима, так как это значительно расширяет потенциал всех этих программ.

Чтобы установить один из компонентов, прежде всего вы должны открыть специальное диалоговое окно Component Wizard (Мастер компонентов). Сделать это можно с помощью команды Component (Компонент) меню Insert (Вставить) или, лучше, нажав одноименную кнопку на панели Standard (Стандартные) (рис. 16.51).

Рис. 16.51. Кнопка Component (Компонент)

Мы ограничимся освещением наиболее часто используемого компонента Microsoft Excel и мастера импорта данных Data Import Wizard. Необходимую информацию об остальных компонентах заинтересованный читатель сможет найти в справочной системе программы.

Excel

Если вы хорошо владеете Excel, то вам совершенно не обязательно осваивать многие разделы в Mathcad. К тому же, вопросы, связанные со статистической обработкой данных, и всевозможные бизнес-задачи решаются в нем гораздо проще (и возможности Excel в этих областях гораздо шире). С другой стороны, система Mathcad обладает несравнимо более высоким потенциалом в сфере обработки экспериментальных данных (интерполяции, регрессии и экстраполяции), поэтому, даже если результаты ваших измерений были записаны в виде электронной таблицы, строить исходя из них графики и диаграммы все же лучше в ней.

Вы можете очень просто произвести обработку данных в Mathcad в том случае, если они сохранены в формате Excel. Для этого достаточно просто скопировать их и вставить затем (с помощью команды контекстного меню Paste Table (Вставить таблицу)) в пустую таблицу ввода. Однако провести обратную операцию (то есть данные из матрицы Mathcad экспортировать в Excel), используя аналогичный подход, не получится. Для выполнения этой задачи следует задействовать таблицу Excel как компонент программы Mathcad.

Чтобы вставить в документ Mathcad электронную таблицу Excel, выполните следующие операции.

- Запустите, задействовав соответствующую команду меню Insert (Вставить) или кнопку панели Standard (Стандартные), окно Component Wizard (Мастер компонентов).
- В списке появившегося окна выберите строку Microsoft Excel и нажмите кнопку Next (Далее).
- В открывшемся диалоговом окне Excel Setup Wizard (Мастер установки Excel) вы должны выбрать один из имеющихся параметров в зависимости от типа решаемой проблемы. Так, если вам требуется провести экспорт каких-то данных из Mathcad в Excel с целью их обработки, то активизируйте параметр Create an empty Excel worksheet (Создать пустой документ Excel). Если же вы собираетесь импортировать в Mathcad данные из существующего Excel-файла, то отметьте настройку Create from file (Создать из файла).
- Любой компонент может быть вставлен в документ Mathcad как в своем обычном виде (для Excel это будет таблица), так и форме маленькой картинки — значка (Icon). Никаких различий в работе с компонентом в этих двух формах не существует, поэтому выбор типа его вида должен определяться вашими предпочтениями и спецификой решаемой задачи (использование значков позволяет значительно экономить место на документе). По умолчанию компонент вставляется в своем обычном виде. Чтобы представить его в форме значка, задействуйте параметр

Display as Icon (Отобразить как значок) диалогового окна Excel Setup Wizard (Мастер установки Excel).

- Когда все необходимые настройки в окне Excel Setup Wizard (Мастер установки Excel) будут сделаны, нажмите Next (Далее).
- С помощью параметров, находящихся на появившейся второй вкладке диалогового окна Excel Setup Wizard (Мастер установки Excel), вы должны определить, из какой ячейки будут читаться (или записываться) данные из документа Excel. Практически это следует сделать из первой (A1) ячейки, так что вносить какие-либо изменения в принятые по умолчанию настройки вам почти наверняка не придется.
- Нажмите Finish (Готово).

В результате осуществления описанных выше действий в документе Mathcad появится таблица, очень похожая на обычную таблицу ввода (Input Table). Если вы вставляли пустой лист Excel, то она будет содержать оператор присваивания (для того чтобы вы могли задать ей имя и работать в дальнейшем, как с обычной матрицей) и маркер, в котором вы должны прописать имя матрицы, данные из которой следует занести в таблицу.

Прочитав матрицу значений Mathcad в таблицу Excel, провести необходимые расчеты вы можете, дважды щелкнув на ней левой кнопкой мыши. При этом все рабочие панели Mathcad будут заменены инструментами Excel, и нужные преобразования можно будет сделать точно так же, как при непосредственной загрузке программы. Выполнив затем щелчок мышью на рабочей области, вы вернетесь в нормальный режим.

Data Import Wizard (Мастер импорта данных)

О функциях, которые предназначены для чтения данных из текстового файла (READPRN и READFILE), мы довольно подробно говорили в предыдущем разделе. Однако в Mathcad подобную работу можно выполнить и гораздо проще, обратившись к специальному мастеру импорта данных Data Import Wizard. Так, например, чтобы прочитать таблицу значений из PRN-файла в матрицу, нужно сделать следующее.

1. В окне Component Wizard (Мастер компонентов) выбрать строку Data Import Wizard (Мастер импорта данных) и нажать Next (Далее).
2. В появившемся диалоговом окне File Options (Параметры файла) определить тип читаемого файла. Сделать это нужно в списке File Format (Формат файла). В нашем случае нужно выбрать пункт Mathcad PRN.
3. Заполнить строку Enter the name of the file which will be associated with this component (Введите имя файла, который будет соединен с этим компонентом). В ней нужно прописать либо имя файла (если он сохранен в том же каталоге, что и Mathcad), либо путь к нему (для чего удобно использовать кнопку Browse (Обзор)). Если вы хотите, чтобы вставляемый компонент отобразился в документе в виде значка, то установите флажок Display as Icon (Отобразить как значок).
4. Нажать Next. При этом откроется диалоговое окно Data Range (Ряды данных), в котором вам будет предложено выбрать необходимые для считывания строки и столбцы (по умолчанию выбран пункт All (Все)).
5. Нажмите Finish (Готово). После этого в документ Mathcad будет вставлен компонент с незаполненным оператором присваивания. Проименовав компонент, вы получите совершенно обычную матрицу, с которой можно будет проводить все допустимые в Mathcad преобразования.

Кстати, с помощью мастера импорта данных Data Import Wizard вы можете считывать информацию из электронных таблиц, причем делается это даже легче, чем при использовании компонента Excel.