Е. В. Шикин, А. И. Плис

КРИВЫЕ И ПОВЕРХНОСТИ на экране компьютера.

руководство по сплайнам для пользователей

МОСКВА ■ "ДИАЛОГ-МИФИ" ■ 1996

УДК 681.3
Ш57

Шикин Е. В., Плис А. И.

Ш57 Кривые и поверхности на экране компьютера. *Руководство по сплайнам для пользователей.* - М.: ДИАЛОГ-МИФИ, 1996. - 240 с.

ISBN 5-86404-080-0

Книга знакомит читателя со сплайнами - эффективным инструментом геометрического моделирования при проектировании гладких кривых и поверхностей. В ней описаны наиболее часто встречающиеся в задачах компьютерной графики одномерные кубические и двухмерные бикубические интерполяционные и сглаживающие сплайны.

Приведенные в книге программы могут быть использованы при решении широкого класса задач визуализации. Книгу можно рассматривать как справочное и практическое руководство, рассчитанное на студентов технических вузов и инженеров.

Ш 2404000000-023 Без объявл.
Г70(03)-96

Учебно-справочное издание
Шикин Евгений Викторович
Плис Александр Иванович
Кривые и поверхности на экране компьютера.
Руководство по сплайнам для пользователей

Редактор О. А. Голубев
Корректор В. С. Кустов
Макет О. А. Кузьминовой
Обложка Н. В. Дмитриевой

Лицензия ЛР N 070109 от 29.08.91. Подписано в печать 15.08.96.
Формат 60x84/16. Бум. офс. Печать офс. Гарнитура Таймс.
Усл. печ. л. 14.1. Уч.-изд. л. 7. Тираж 5 000 экз. Заказ935

Акционерное общество "ДИАЛОГ-МИФИ"
115409, Москва, ул. Москворечье, 31, корп. 2

Подольская типография
142110, г. Подольск, Московская обл., ул. Кирова, 25

ISBN 5-86404-080-0

Предисловие

Пути, связывающие изначальные авторские замыслы и вышедшую книгу, часто оказываются столь своеобразными, что их описание заслуживает отдельного рассказа, который, впрочем, вряд ли будет интересен читателям. К тому же удовольствие, которое обыкновенно испытывает при этом рассказчик, никак не сможет заслонить главного события - самого выхода книги.

Выпуск книги дает хороший повод для радости ее авторам. Стоит, однако, заметить, что эта радость оказывается полной лишь в тот момент, когда каждый из выпущенных экземпляров книги найдет своего читателя, то есть того, кому книга окажется и полезной и нужной. Завершая работу над рукописью, авторы рассчитывают со временем испытать и эту радость.

Мы искренне надеемся, что вместе с нами ее разделят все те, кто имеет пусть даже и малое отношение к выходу книги. Кроме издательства и авторов, явно обозначенных на ее обложке, существуют еще и другие лица, причастные к происходящим событиям. Их совокупную роль в появлении пособия не способен переоценить ни один из авторов.

Мы считаем своим приятным долгом назвать их имена и выразить испытываемое нами чувство глубокой признательности Ольге Владимировне Виноградовой, Кириллу Евгеньевичу Виноградову и Сергею Ежкову.

Москва
Март 1996 года

А. И. Плис
Е. В. Шикин

О структуре пособия

Несколько общих советов пользователю

При подготовке справочного руководства авторы исходили из того, что помещенный в него материал окажется полезным как тем, кто будет знакомиться со сплайнами впервые, так и тем, кто в своей работе сплайны уже применял. Вместе с тем это совсем не учебник, и за подробными разъяснениями и доказательствами, по-видимому, целесообразнее обратиться к соответствующим книгам и научным публикациям (часть из них указана авторами в конце пособия).

Целью авторов было создание простого по существу и доступного широкому кругу пользователей пособия, посредством которого каждый пользователь смог бы научиться достаточно успешно и эффективно решать основные задачи "на сплайны", а также получил бы определенное представление о методах, которые при этом используются.

На всех этапах работы над руководством значительное внимание уделялось выработке структуры и формы подачи материала, наиболее удобной для пользователя.

Предлагаемое пособие имеет целый ряд особенностей, на описании которых стоит остановиться подробнее.

Одна из основных идей, которую авторы постарались заложить в книгу, состоит в том, чтобы помочь пользователю в выборе именно того метода, который в наибольшей степени подходит для решения его конкретной задачи. Для ее реализации авторы используют сравнение различных подходов и методов и описание их особенностей как постоянную составляющую часть изложения. Этой же цели служит следование исторической и/или логической хронологии появления помещенных в книгу методов, последовательный показ того, как умножаются идеи и достоинства ранее найденных подходов при их модификации и развитии, как разрабатываемые методы постепенно освобождаются от недостатков.

Открыв пособие на произвольной странице, пользователь вправе ожидать, что в изложенном там ему удастся сравнительно легко разобраться, не изучая книгу целиком. Именно к реализации такой возможности стремились авторы, разрабатывая структуру пособия и отбирая материал для его наполнения. Однако в целом ряде случаев последовательность рассмотрения материала в книге достаточно жестко предопределена. Если, к примеру, читатель хочет познакомиться с интерполяционными сплайн-функциями двух переменных, то ему целесообразно сначала хотя бы просмотреть раздел, где описываются

интерполяционные сплайн-функции одной переменной. Совершенно аналогично погружение в раздел В-сплайновых поверхностей разумно предварить знакомством с соответствующим одномерным разделом, где речь идет о В-сплайновых кривых.

Хочется специально подчеркнуть, что главы, посвященные сплайн-функциям двух переменных (гл. 2) и сплайновым поверхностям (гл. 4), заметно сложнее тех, в которых рассматриваются сплайн-функции одной переменной (гл. 1) и сплайновые кривые (гл. 3). Это вполне естественно - более сложными являются сами объекты. Поэтому для того, чтобы как-то упростить пользование пособием, авторы постарались отобрать классы сплайнов так, чтобы практически каждый помещенный в пособие класс одномерных сплайнов имел в нем двумерный аналог и, в свою очередь, чтобы каждый двумерный класс опирался на одномерного предтечу.

В целом же изложение построено так, чтобы каждую главу, а во многих случаях и каждый подраздел можно было читать независимо от других. Схожесть описания методов построения различных сплайнов, равно как и минимум перекрестных ссылок, помогут пользователю лучше ориентироваться на страницах книги.

Фактически предлагаемое пособие разбито на 4 относительно независимых и тематически явно выделенных подпособия (именно так можно рассматривать главы книги). По мнению авторов, это поможет пользователю сравнительно просто получать ответы на интересующие его вопросы без того, чтобы собирать соответствующий материал по всей книге. В тех же местах, где использование материалов другой главы целесообразно (и даже необходимо), даны точные ссылки.

Отбор материала производился с тем расчетом, чтобы дать читателю возможность, познакомившись с основными фактами, понятиями и приемами, сравнительно просто войти в мир сплайнов.

Многие результаты, полученные при помощи расчетов, обладают определенной особенностью: иногда достаточно трудно бывает объяснить, почему на этом этапе вычислений нужно поступать именно так, а не иначе или какими соображениями следует пользоваться, если возникает нестандартная ситуация, и т. п. Здесь большую роль играет интуиция и опыт. С интуицией дело не всегда обстоит просто. Опыт же приобретается, причем многими способами. Настоящее пособие как раз и представляет собой один из таких путей.

Сдерживаемые не столько объемом, сколько осознанием ответственности перед читателем, мы посчитали целесообразным не заострять внимание читателя на многообразных тонкостях обращения со сплайнами, отчетливо понимая, насколько легко можно в них увяз-

нуть. Поэтому мы сознательно отобрали и жестко структурировали отобранный материал так, чтобы у пользователя сложилось в целом правильное рабочее представление о сплайновом инструментарии и его возможностях, чтобы пользователь мог достаточно свободно решать основные задачи приближения. Кроме того, мы попытались построить изложение таким образом, чтобы, опираясь на материал, помещенный в книгу, читатель имел возможность проверить, насколько хорошо он этот материал усвоил.

Книга разбита на две большие части, одна из которых посвящена сплайн-функциям, а вторая - геометрическим сплайнам. В первой части как бы главенствуют расчеты, в то время как во второй ведущая роль отводится изображению. Однако эти части довольно естественно связаны общностью основных идей. Одной из таких связующих идей является программная реализация сплайновых алгоритмов.

Почему сплайны?

В основе почти всякого созидания лежит идея составления целого из тщательно подобранных частей. Старая, почти так же как и вся наша цивилизация, она, однако, и по сей день все еще далека от окончательного воплощения в жизнь. Оно и понятно: разрушить значительно легче, чем скрупулезно собрать воедино и склеить. Да еще так, чтобы созданное радовало глаз.

На протяжении всей своей истории человечество пыталось (и временами довольно успешно) решать эту задачу. Приведем только два примера, выбрав наиболее близкие к тем задачам, разрешению которых посвящено настоящее справочное пособие.

Существует красивая легенда о том, как появился так называемый *бухарский* фарфор. В те давние времена фарфоровые изделия изготовлялись исключительно в Китае, были страшно дороги и потому особенно высоко ценились властителями больших и малых земель. Изготовленный по заявке бухарского эмира и караванными путями со всеми возможными предосторожностями доставленный к его двору роскошный фарфоровый сервиз не выдержал многодневного путешествия - и значительная часть действительно хрупких изделий оказалась в осколках. Нам в нынешнем, *XX* веке трудно в полной мере оценить страх, который испытывали подданные перед своим эмиром. А им было хорошо известно, чем грозит невыполнение желаний повелителя и к чему может привести доставка ценного груза в столь некондиционном состоянии. Нужно было срочно искать достойный выход. И как гласит легенда, он был найден. Знаменитые на всем Востоке золотых дел мастера тончайшими пластинами скрепили осколки ваз, блюд, тарелок, блюдец и чашек. Искусные ювелиры смогли не только воссоздать изящную красоту фарфоровых изделий, но и вернули им функциональность - изделия можно было использовать по прямому назначению. Проделанные операции придали многострадальному сервизу уникальность столь привлекательную, что позже выписанный из Китая фарфор употребляли только после своеобразного расчленения и последующей "золотой" сборки. И сейчас еще, путешествуя по Средней Азии, можно встретить фарфоровые или фаянсовые чайники, починенные по этому "бухарскому" образцу, но, правда, уже без золотых пластин.

А вот еще один, более известный, *корабельный* пример. Привлекаемое окружающими его водными просторами любопытное человечество не слишком долго довольствовалось плотами и долбленными лодками - их мореходные качества были малоудовлетворительны.

Стремление людей узнать, что там интересного имеется за морем, в немалой степени способствовало поиску все более совершенных средств для перемещения по воде. И как это часто случается с великими открытиями, идея создания плавсредств путем сборки возникла и начала развиваться совершенно независимо в разных частях Земли (правда, и с разным успехом). Ключевым в этой идее было осознание необходимости начинать изготовление судна с прочного каркаса. С течением времени универсальным методом проб и ошибок этот принципиальный вопрос был решен и для каркаса были найдены вполне удовлетворительные формы, позволившие придать судам плавучесть, устойчивость и нужную маневренность. И уже довольно скоро после этого на каравеллах, клиперах и шхунах человечество вплыло в эпоху великих географических открытий.

Позже эти идеи стали весьма успешно использоваться не только в кораблестроении, но и при создании автомобилей и конструировании самолетов, а также в архитектуре.

Заметим, что, несмотря на различие используемых материалов, в обоих случаях изделие воссоздавалось по заданному шаблону путем подгонки имеющихся фрагментов. Правда, при решении фарфоровой проблемы творческий коллектив был разделен на две независимые части, одна из которых (китайская) создавала, а вторая (бухарская) воссоздавала по фактически готовым образцам (тех, кто разбивал, в расчет принимать не стоит). Корабелам же приходилось делать самим и деревянный каркас, и его обшивку (благо что леса в ту пору еще обильно покрывали Землю).

Попробуем перевести рассматриваемую проблему на геометрический язык.

Геометрическая модель. Описав искомую гладкую поверхность массивом соответствующим образом выделенных опорных точек, построить по ним эту поверхность путем подбора нужных фрагментов.

Тем самым решение задачи создания искривленной поверхности разбивается на несколько этапов:

1) задание подходящего набора опорных точек,
2) выработка универсального и сравнительно несложного способа построения элементарных фрагментов поверхности,
3) отыскание эффективного механизма плавной состыковки элементарных фрагментов,
4) анализ полученных результатов.

Ясно, что решение такой задачи далеко не однозначно, так как приблизить массив точек плавноизменяющейся кривой (или поверх-

ностью) можно очень многими способами. Тем самым множество приближающих геометрических объектов практически необозримо. Вместе с тем при решении конкретных задач, как правило, требуется найти только один приближающий объект, причем ведущий себя достаточно хорошо. Поэтому разумно сначала четко понять, что будет играть роль иголки в стоге возможностей, и только потом приниматься за поиски.

Выбор элементарных фрагментов, представляющих собой вырезки гладких кривых или поверхностей, должен быть таким, чтобы их описание было простым, геометрически наглядным и универсальным. Преимущества простоты описания вполне очевидны. Универсальность позволяет добиться того, чтобы объем вычислений был сравнительно невелик. Хорошо прослеживаемые геометрические характеристики неоценимы при состыковке фрагментов и анализе полученной составной кривой или поверхности. Обычно геометрической непрерывности, то есть гладкого изменения касательной к кривой и ее кривизны или соответственно касательной плоскости к поверхности и ее кривизны, во многих случаях оказывается вполне достаточно. Если использовать для описания фрагментов полиномиальные скалярные функции, то такой непрерывности можно добиться, ограничившись многочленами 3-й степени (в двумерном случае по каждой из двух вспомогательных переменных).

Немаловажную роль играет и проблема единственности решения задачи приближения.

Конечно, удовлетворить сразу всем указанным выше ограничениям не просто. Использование известных классов кривых или поверхностей (например, задаваемых многочленами или кусочно-линейными функциями) позволяет удовлетворить только части указанных требований, правда всякий раз своей. Поэтому довольно естественным представляется желание объединить каким-нибудь удачным способом большую часть достоинств этих классов.

Укажем некоторые вполне естественные требования, которые нужно наложить на выделяемые классы кривых и поверхностей и выполнение которых необходимо для успешного разрешения задачи приближения (разумеется, есть и другие).

1. Выбираемый класс должен описываться достаточно просто.
2. Кривые или поверхности, входящие в выделенный класс, не должны иметь особенностей, то есть должны быть достаточно гладкими - нигде не рваться, иметь непрерывно изменяющуюся касательную или непрерывную кривизну.

3. Поиск нужной кривой или поверхности в выделенном классе должен быть сравнительно легким, что предполагает наличие эффективного алгоритма ее построения.
4. Для достаточно больших массивов точек найденные кривые или поверхности должны вести себя вполне предсказуемо.

Небезынтересно заметить, что кривые и поверхности, удовлетворяющие перечисленным требованиям, существуют. Они составлены из ладно пригнанных гладких кусочков.

Именно эти кривые и поверхности принято называть *сплайнами.*

Таким образом, неизбежность появления сплайнов висела в воздухе уже очень давно. По-видимому, в наиболее доходчивой и вместе с тем поэтичной форме это всеобщее ощущение сумел выразить великий Гоголь в "Женитьбе": "Если бы губы Никанора Ивановича да приставить к носу Ивана Кузьмича, да взять сколько-нибудь развязности, какая у Балтазара Балтазарыча, да, пожалуй, прибавить к этому еще дородности Ивана Павловича - *я бы тогда тотчас же решилась*" (курсив наш. - *А. П., Е. Ш.*).

Часть I
Сплайн-функции

В практике многие кривые и поверхности имеют довольно сложную форму, не допускающую универсального аналитического задания в целом при помощи элементарных функций. Поэтому их собирают из сравнительно простых гладких фрагментов - отрезков (кривых) или вырезков (поверхностей), каждый из которых может быть вполне удовлетворительно представлен в виде элементарной функции одной или двух переменных.

Для того чтобы получающаяся в результате составная кривая или поверхность была достаточно гладкой, необходимо быть особенно внимательным в местах стыковки.

Естественно потребовать, чтобы гладкие функции, графики которых используются при построении частичных кривых или поверхностей, имели схожую природу, например были многочленами одинаковой степени. Последняя выбирается из простых геометрических соображений и, как правило, невелика: для гладкого изменения касательной вдоль всей составной кривой достаточно описывать стыкуемые кривые при помощи многочленов 3-й степени. Коэффициенты этих многочленов всегда можно подобрать так, чтобы соответствующая составная функция имела непрерывную 2-ю производную.

Результаты решения одномерных задач разумно использовать при построении фрагментов соответствующих составных поверхностей. Так возникают бикубические сплайны - функции, являющиеся многочленами 3-й степени по каждой переменной. Работа с такими сплайнами требует уже значительно большего объема вычислений. Но правильно организованный процесс позволяет учесть нарастающие возможности вычислительной техники в максимально возможной степени.

Таким образом, в этой части рассматриваются гладкие кривые и поверхности, представляющие собой графики кусочно-полиномиальных функций невысоких степеней. Именно с построения таких кривых, а потом и поверхностей сплайны сравнительно недавно и начали свое бурное развитие.

Глава 1. Сплайн-функции одной переменной

Пусть на отрезке $[a, b]$ задана *сетка* ω:

$$a = x_0 < x_1 < \ldots < x_{m-1} < x_m = b .$$

Точки x_0 и x_m называются *граничными узлами* сетки ω, а точки $x_1, \ldots, x_{m-1}$ - ее *внутренними узлами* (рис. 1.1). Сетка называется *равномерной*, если расстояния между любыми двумя соседними узлами одинаковы.

Функция $S(x)$, заданная на отрезке $[a, b]$, называется *сплайном порядка $p+1$ (степени p)*, если эта функция:

1) на каждом из отрезков

$$\Delta_i = [x_i, x_{i+1}], \qquad i = 0, 1, \ldots, m-1,$$

является многочленом заданной степени $p \geq 2$, то есть может быть записана в виде

$$S(x) = S_i(x) = \sum_{k=0}^{p} a_k^{(i)} (x - x_i)^k , \qquad i = 0, 1, \ldots, m - 1;$$

и

2) $p - 1$ раз непрерывно дифференцируема на отрезке $[a, b]$, то есть

$$S(x) \in C^{p-1}[a, b].$$

Замечание

Индекс (i) у чисел $a_k^{(i)}$ указывает на то, что набор коэффициентов, которым определяется функция $S(x)$, на каждом частичном отрезке Δ_i свой.

На каждом из отрезков Δ_i, $i = 0, 1, \ldots, m - 1$, сплайн $S(x)$ является многочленом степени p и определяется на этом отрезке $p+1$ коэффициентом. Всего частичных отрезков - m. Значит, для того чтобы полностью определить сплайн, необходимо найти $(p+1)m$ чисел

$$a_k^{(i)}, \qquad k = 0, 1, \ldots, p, \qquad i = 0, 1, \ldots, m - 1.$$

Условие $S(x) \in C^{p-1}[a.b]$ означает непрерывность функции $S(x)$ и ее производных $S'(x)$, $S''(x)$, ..., $S^{p-1}(x)$ во всех внутренних узлах

сетки ω. Число таких узлов - m -1. Тем самым для отыскания коэффициентов всех многочленов получается $p(m-1)$ условий (уравнений).

Для полного определения сплайна недостает $(p+1)m - p(m-1) = m + p$ условий (уравнений). Выбор дополнительных условий определяется характером рассматриваемой задачи, а иногда и просто желанием пользователя.

Наиболее часто рассматриваются задачи интерполяции и сглаживания, когда требуется построить тот или иной сплайн по заданному массиву точек на плоскости (x_i, y_i), $i = 0, 1, ..., m$ (рис. 1.2).

В задачах *интерполяции* требуется, чтобы график сплайна проходил через точки (x_i, y_i), $i = 0, 1, ..., m$, что накладывает на его коэффициенты $m + 1$ дополнительных условий (уравнений). Остальные p -1 условий (уравнений) для однозначного построения сплайна чаще всего задают в виде значений младших производных сплайна на концах рассматриваемого отрезка $[a, b]$ - *граничных* (*краевых*) *условий*. Возможность выбора различных граничных условий позволяет строить сплайны, обладающие самыми разными свойствами.

В задачах *сглаживания* сплайн строят так, чтобы его график проходил вблизи точек (x_i, y_i), $i = 0, 1, ..., m$, а не через них. Меру этой близости можно определять по-разному, что приводит к значительному разнообразию сглаживающих сплайнов.

Описанные возможности выбора при построении сплайн-функций далеко не исчерпывают всего их многообразия. И если первоначально рассматривались только кусочно-полиномиальные сплайн-функции, то по мере расширения сферы их приложений стали возникать сплайны, "склеенные" и из других элементарных функций. Один из таких классов - класс *напряженных сплайнов*, обладающих полезными интересными свойствами, приведен в этой главе в качестве примера.

1.1. Интерполяционные кубические сплайны

1.1.1. Постановка задачи интерполяции

Пусть на отрезке $[a, b]$ задана сетка ω:

$$a = x_0 < x_1 < \ldots < x_{m-1} < x_m = b .$$

Рассмотрим набор чисел

$$y_0, y_1, \ldots, y_{m-1}, y_m.$$

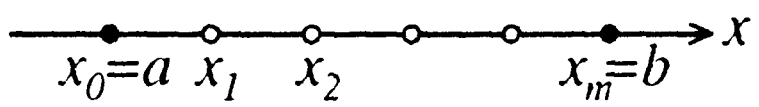


Рис. 1.1

Задача. Построить гладкую на отрезке $[a, b]$ функцию $\sigma(x)$, которая принимает в узлах сетки ω заданные значения, то есть

$$\sigma(x_i) = y_i,\ i = 0, 1, ..., m-1, m \qquad \text{(рис. 1.1).}$$

Замечание

Сформулированная задача интерполяции состоит в восстановлении гладкой функции, заданной таблично (рис. 1.2). Ясно, что такая задача имеет множество различных решений. Накладывая на конструируемую функцию дополнительные условия, можно добиться необходимой однозначности.

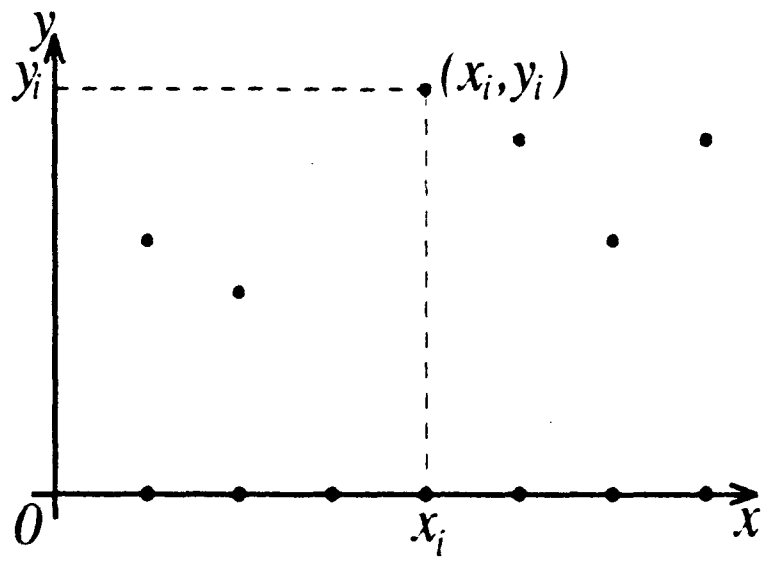


Рис. 1.2

В приложениях часто возникает необходимость приблизить функцию, заданную аналитически,

$$y = f(x),\ x \in [a,b],$$

при помощи функции с предписанными достаточно хорошими свойствами. Например, в тех случаях, когда вычисление значений заданной функции $f(x)$ в точках отрезка $[a, b]$ связано со значительными трудностями и/или заданная функция $f(x)$ не обладает требуемой гладкостью, удобно воспользоваться другой функцией, которая достаточно хорошо приближала бы заданную функцию и была лишена отмеченных ее недостатков.

Задача интерполяции функции

Построить на отрезке $[a, b]$ гладкую функцию $\sigma(x)$, совпадающую в узлах сетки ω с заданной функцией $f(x)$,

$$\sigma(x_i) = f(x_i), \qquad i = 0, 1, \ldots, m-1, m.$$

1.1.2. Определение интерполяционного кубического сплайна

Интерполяционным кубическим сплайном $S(x)$ на сетке ω называется функция, которая:

1) на каждом из отрезков $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, m-1$, представляет собой многочлен 3-й степени,

$$S(x) = S_i(x) = a_0^{(i)} + a_1^{(i)}(x - x_i) + a_2^{(i)}(x - x_i)^2 + a_3^{(i)}(x - x_i)^3;$$

2) дважды непрерывно дифференцируема на отрезке $[a, b]$, то есть принадлежит классу $C^2[a,b]$;

3) удовлетворяет условиям

$$S(x_i) = y_i, \qquad i = 0, 1, \ldots, m. \qquad (1.1)$$

На каждом из отрезков $[x_i, x_{i+1}]$, $i = 0, 1, ..., m-1$, сплайн $S(x)$ является многочленом 3-й степени и определяется на этом отрезке четырьмя коэффициентами. Всего отрезков - m. Значит, для того чтобы полностью определить сплайн, необходимо найти $4m$ чисел

$$a_0^{(i)},\ a_1^{(i)},\ a_2^{(i)},\ a_3^{(i)}, \qquad i = 0, 1, ..., m-1.$$

Условие $S(x) \in C^2[a,b]$ означает непрерывность функции $S(x)$ и ее производных $S'(x)$ и $S''(x)$ во всех внутренних узлах сетки ω. Число таких узлов - $m-1$. Тем самым для отыскания коэффициентов всех многочленов получается еще $3(m-1)$ условий (уравнений). Вместе с условиями (1.1) получается $3(m-1)+(m+1) = 4m-2$ условий (уравнений).

1.1.3. Граничные (краевые) условия

Два недостающих условия задаются в виде ограничений на значения сплайна и/или его производных на концах промежутка $[a, b]$. При построении интерполяционного кубического сплайна наиболее часто используются краевые условия следующих четырех типов.

Граничные (краевые) условия 1-го типа

$$S'(a) = f'(a), \quad S'(b) = f'(b)$$

- на концах промежутка $[a, b]$ задаются значения 1-й производной искомой функции.

Граничные (краевые) условия 2-го типа

$$S''(a) = f''(a), \quad S''(b) = f''(b)$$

- на концах промежутка $[a, b]$ задаются значения 2-й производной искомой функции.

Граничные (краевые) условия 3-го типа

$$S'(a) = S'(b), \quad S''(a) = S''(b)$$

называются *периодическими*. Выполнения этих условий естественно требовать в тех случаях, когда интерполируемая функция является периодической с периодом $T = b-a$.

Граничные (краевые) условия 4-го типа

$$S'''(y, x_1 - 0) = S'''(y, x_1 + 0), \quad S'''(y, x_{m-1} - 0) = S'''(y, x_{m-1} + 0)$$

требуют особого комментария.

Комментарий. Во внутренних узлах сетки 3-я производная функции $S(x)$, вообще говоря, разрывна. Однако число разрывов 3-й производной можно уменьшить при помощи условий 4-го типа.

В этом случае построенный сплайн будет трижды непрерывно дифференцируем на промежутках $[x_0, x_2]$ и $[x_{m-2}, x_m]$.

1.1.4. Построение интерполяционного кубического сплайна

Опишем способ вычисления коэффициентов кубического сплайна, при котором число величин, подлежащих определению, равно $m+1$, а не $4m$.

На каждом из промежутков $[x_i, x_{i+1}]$, $i = 0, 1, ..., m-1$, интерполяционная сплайн-функция ищется в следующем виде:

$$S(x) = S_i(x) = y_i(1-t)^2(1+2t) + y_{i+1}t^2(3-2t) + n_i h_i t(1-t)^2 - n_{i+1}t^2(1-t). \tag{1.2}$$

Здесь

$$h_i = x_{i+1} - x_i, \qquad t = \frac{x - x_i}{h_i},$$

а числа n_i, $i = 0, 1, ..., m$, являются решением системы линейных алгебраических уравнений, вид которой зависит от типа граничных (краевых) условий.

Для *граничных (краевых) условий 1-го и 2-го типов* эта система имеет следующий вид:

$$\begin{cases} 2n_0 + \mu_0^* n_1 = c_o^*, \\ \lambda_i n_{i-1} + 2m_i + \mu_i n_{i+1} = c_i, \\ \lambda_m^* n_{m-1} + 2n_m = c_m^*, \end{cases}$$

где $i = 1, 2, ..., m-1$;

$$c_i = 3\left(\mu_i \frac{y_{i+1} - y_i}{h_i} + \lambda_i \frac{y_i - y_{i-1}}{h_{i-1}}\right).$$

Коэффициенты μ_0^*, c_0^*, λ_m^*, c_m^* зависят от выбора граничных (краевых) условий.

Граничные (краевые) условия 1-го типа:

$$\mu_0^* = 0, \quad c_0^* = 2y_0',$$
$$\lambda_m^* = 0, \quad c_m^* = 2y_m'.$$

Граничные (краевые) условия 2-го типа:

$$\mu_0^* = 1, \quad c_0^* = 3\frac{y_1 - y_0}{h_0} - \frac{h_0}{2}y_0'',$$

$$\lambda_m^* = 1, \quad c_m^* = 3\frac{y_m - y_{m-1}}{h_{m-1}} + \frac{h_{m-1}}{2} y_m'' .$$

В случае *граничных (краевых) условий 3-го типа* система для определения чисел n_i, $i = 1, 2, \ldots, m$, записывается так:

$$\begin{cases} 2n_1 + \mu_1 n_2 + \lambda_1 n_m = c_1, \\ \lambda_i n_{i-1} + 2n_i + \mu_i n_{i+1} = c_i, \\ \mu_m n_1 + \lambda_m n_{m-1} + 2n_m = c_m. \end{cases}$$

где $i = 2, \ldots, m\text{-}1$.

Число неизвестных в последней системе равно m, так как из условия периодичности вытекает, что $n_0 = n_m$.

Для *граничных (краевых) условий 4-го типа* система для определения чисел n_i, $i = 1, 2, \ldots, m\text{-}1$, имеет вид:

$$\begin{cases} (1+\gamma_0)n_1 + \gamma_0 n_2 = c_1^*, \\ \lambda_i n_{i-1} + 2n_i + \mu_i n_{i+1} = c_i, \\ \gamma_m n_{m-2} + (1+\gamma_m)n_{m-1} = c_{m-1}^*, \end{cases}$$

где $i = 2, \ldots, m\text{-}2$;

$$\gamma_0 = \frac{h_o}{h_1}, \qquad c_1^* = \frac{1}{3}c_1 + 2\gamma_0 \frac{y_2 - y_1}{h_1},$$

$$\gamma_m = \frac{h_{m-1}}{h_{m-2}}, \qquad c_{m-1}^* = \frac{1}{3}c_{m-1} + 2\gamma_m \frac{y_{m-1} - y_{m-2}}{h_{m-2}}.$$

По найденному решению системы числа n_0 и n_m можно определить при помощи формул

$$n_0 = 2\left(\frac{y_1 - y_0}{h_0} - \gamma_0^2 \frac{y_2 - y_1}{h_1}\right) - (1 - \gamma_0^2)n_1 + \gamma_0^2 n_2 ,$$

$$n_m = 2\left(\frac{y_m - y_{m-1}}{h_{m-1}} - \gamma_m^2 \frac{y_{m-1} - y_{m-2}}{h_{m-2}}\right) - (1 - \gamma_m^2)n_{m-1} + \gamma_m^2 n_{m-2} .$$

Важное замечание

Матрицы всех трех линейных алгебраических систем являются матрицами с диагональным преобладанием. Такие матрицы невырождены, и потому каждая из этих систем имеет единственное решение.

Теорема. Интерполяционный кубический сплайн, удовлетворяющий условиям (1.1) и граничному (краевому) условию одного из перечисленных четырех типов, существует и единственен.

Таким образом, построить интерполяционный кубический сплайн - это значит найти его коэффициенты n_0, n_1, ..., n_m.

Когда коэффициенты сплайна найдены, значение сплайна $S(x)$ в произвольной точке отрезка $[a, b]$ можно найти по формуле (1.2). Однако для практических вычислений больше подходит следующий алгоритм нахождения величины $S(x)$.

Пусть $x \in [x_i, x_{i+1}]$. Сначала вычисляются величины A и B по формулам

$$A = -2\frac{y_{i+1} - y_i}{x_{i+1} - x_i} + n_i + n_{i+1},$$

$$B = -A + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + n_i,$$

а затем находится величина $S(x)$:

$$S(x) = y_i + (x - x_i)\left[n_i + t(B + tA)\right],$$

где, как обычно,

$$t = \frac{x - x_i}{x_{i+1} - x_i}.$$

Применение этого алгоритма существенно сокращает вычислительные затраты на определение величины $S(x)$.

1.1.5. Советы пользователю

Выбор граничных (краевых) условий и узлов интерполяции позволяет в известной степени управлять свойствами интерполяционных сплайнов.

А. Выбор граничных (краевых) условий

Выбор граничных (краевых) условий является одной из центральных проблем при интерполяции функций. Он приобретает особую важность в том случае, когда необходимо обеспечить высокую точность аппроксимации функции $f(x)$ сплайном $S(x)$ вблизи концов отрезка $[a, b]$. Граничные значения оказывают заметное влияние на поведение сплайна $S(x)$ вблизи точек a и b, и это влияние по мере удаления от них быстро ослабевает.

Выбор граничных (краевых) условий часто определяется наличием дополнительных сведений о поведении аппроксимируемой функции $f(x)$.

Если на концах отрезка [*a*, *b*] известны значения 1-й производной $f'(x)$, то естественно воспользоваться граничными (краевыми) условиями 1-го типа.

Если на концах отрезка [*a*, *b*] известны значения 2-й производной $f''(x)$, то естественно воспользоваться граничными (краевыми) условиями 2-го типа.

Если есть возможность выбора между граничными (краевыми) условиями 1-го и 2-го типа, то предпочтение следует отдать условиям 1-го типа.

Если $f(x)$ - периодическая функция, то следует остановиться на граничных (краевых) условиях 3-го типа.

В случае, если никакой дополнительной информации о поведении аппроксимируемой функции нет, часто используют так называемые *естественные граничные (краевые) условия* $S''(a) = 0$, $S''(b) = 0$. Однако следует иметь ввиду, что при таком выборе граничных (краевых) условий точность аппроксимации функции $f(x)$ сплайном $S(x)$ вблизи концов отрезка [*a*, *b*] резко снижается. Иногда используются граничные (краевые) условия 1-го или 2-го типа, но не с точными значениями соответствующих производных, а с их разностными аппроксимациями. Точность такого подхода невысока.

Практический опыт расчетов показывает, что в рассматриваемой ситуации наиболее целесообразным является выбор граничных (краевых) условий 4-го типа.

Б. Выбор узлов интерполяции

Если 3-я производная $f'''(x)$ функции терпит разрыв в некоторых точках отрезка [*a*, *b*], то для улучшения качества аппроксимации эти точки следует включить в число узлов интерполяции.

Если разрывна 2-я производная $f''(x)$, то для того, чтобы избежать осцилляции сплайна вблизи точек разрыва, необходимо принять специальные меры. Обычно узлы интерполяции выбирают так, чтобы точки разрыва 2-й производной попадали внутрь промежутка $[x_i, x_{i+1}]$, такого, что

$$h_i = \alpha \min\{h_{i-1}, h_{i+1}\},$$

где $\alpha << 1$. Величину α можно выбрать путем численного эксперимента (часто достаточно положить $\alpha = 0.01$).

Существует набор рецептов по преодолению трудностей, возникающих при разрывной 1-й производной $f'(x)$. В качестве одного из

самых простых рецептов можно предложить такой: разбить отрезок аппроксимации на промежутки, где производная непрерывна, и на каждом из этих промежутков построить сплайн.

1.1.6. Выбор интерполяционной функции (плюсы и минусы)

А. Интерполяционный многочлен Лагранжа

По заданному массиву

(x_i, y_i), $i = 0, 1, ..., m\text{-}1, m$

(рис. 1.3) *интерполяционный многочлен Лагранжа* определяется формулой

$$L_m(x) = \sum_{i=0}^{m} y_i \frac{\varphi_m(x)}{(x - x_i)\varphi'_m(x_i)},$$

где

$$\varphi_m(x) = \prod_{i=0}^{m} (x - x_i).$$

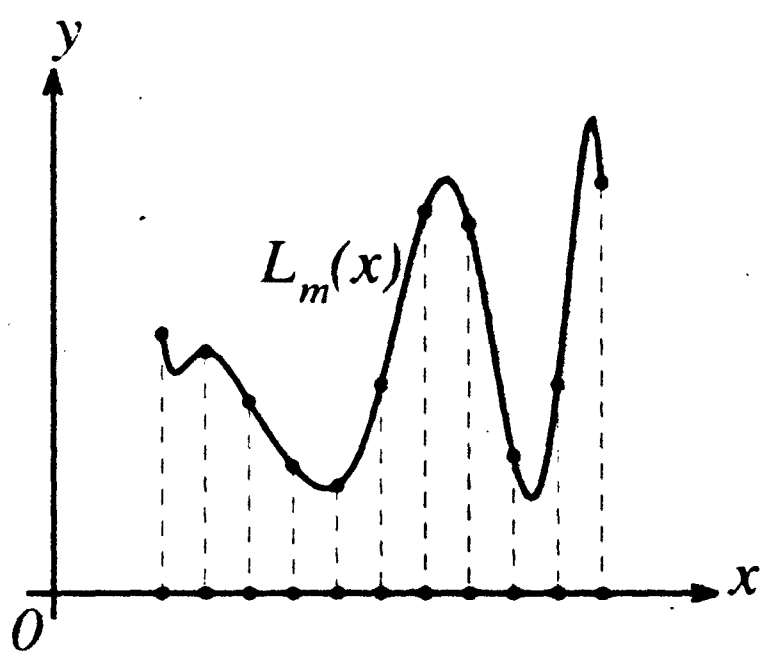


Рис. 1.3

Свойства интерполяционного многочлена Лагранжа целесообразно рассматривать с двух противоположных позиций, обсуждая основные достоинства отдельно от недостатков.

Основные достоинства 1-го подхода:

1) график интерполяционного многочлена Лагранжа проходит через каждую точку массива;
2) конструируемая функция легко описывается (число подлежащих определению коэффициентов интерполяционного многочлена Лагранжа на сетке ω равно $m + 1$);
3) построенная функция имеет непрерывные производные любого порядка;
4) заданным массивом интерполяционный многочлен определен однозначно.

Основные недостатки 1-го подхода:

1) степень интерполяционного многочлена Лагранжа зависит от числа узлов сетки, и чем больше это число, тем выше степень интерполяционного многочлена и, значит, тем больше требуется вычислений;
2) изменение хотя бы одной точки в массиве требует полного пересчета коэффициентов интерполяционного многочлена Лагранжа;

3) добавление новой точки в массив увеличивает степень интерполяционного многочлена Лагранжа на единицу и также приводит к полному пересчету его коэффициентов;
4) при неограниченном измельчении сетки степень интерполяционного многочлена Лагранжа неограниченно возрастает.

Поведение интерполяционного многочлена Лагранжа при неограниченном измельчении сетки вообще требует особого внимания.

Комментарии:

О приближении непрерывной функции многочленом

Известно (Вейерштрасс, 1885), что всякая непрерывная (а тем более гладкая) на отрезке функция может быть как угодно хорошо приближена на этом отрезке многочленом.

Опишем этот факт на языке формул. Пусть $f(x)$ - функция, непрерывная на отрезке $[a, b]$. Тогда для любого $\varepsilon > 0$ найдется такой многочлен $P_n(x)$, что для любого x из промежутка $[a, b]$ будет выполняться неравенство

$$\left|P_n(x) - f(x)\right| < \varepsilon$$

(рис. 1.4). Отметим, что многочленов даже одной степени, приближающих функцию $f(x)$ с указанной точностью, существует бесконечно много.

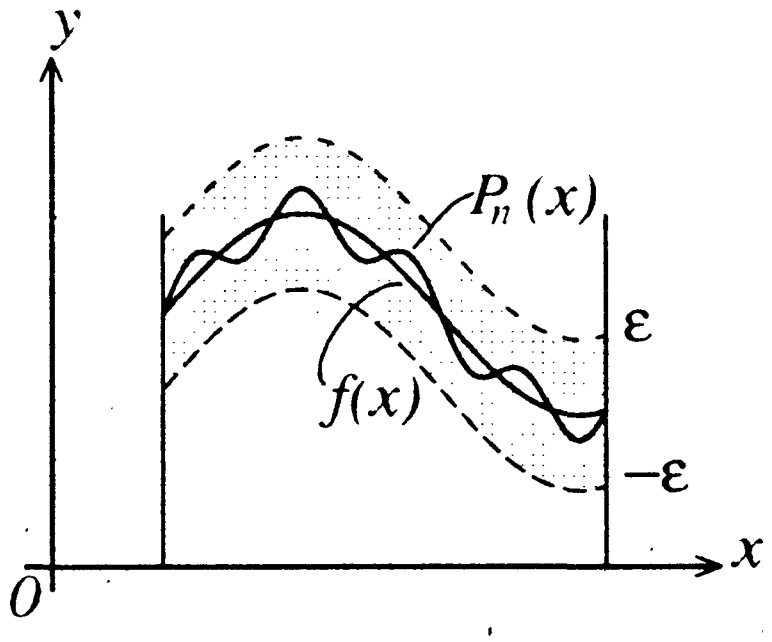


Рис. 1.4

Построим на отрезке $[a, b]$ сетку ω. Ясно, что ее узлы, вообще говоря, не совпадают с точками пересечения графиков многочлена $P_n(x)$ и функции $f(x)$ (рис. 1.5). Поэтому для взятой сетки многочлен $P_n(x)$ не является интерполяционным.

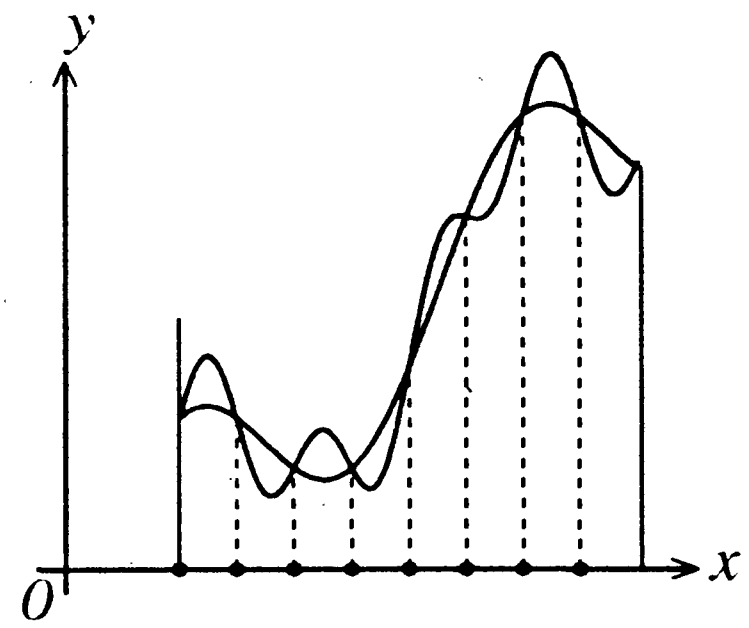


Рис. 1.5

Об интерполировании непрерывной функции.

При аппроксимации непрерывной функции интерполяционным многочленом Лагранжа его график не

только не обязан быть близким к графику функции $f(x)$ в каждой точке отрезка $[a, b]$, но может уклоняться от этой функции как угодно сильно. Приведем два примера.

Пример 1. (Рунге, 1901). При неограниченном увеличении

числа узлов для функции

$$f(x) = \frac{1}{1+25x^2}$$

на отрезке [-1, 1] (рис. 1.6) выполняется предельное равенство

$$\lim_{m\to\infty} \max_{-1\le x\le 1} \left| f(x) - L_m(x) \right| = \infty .$$

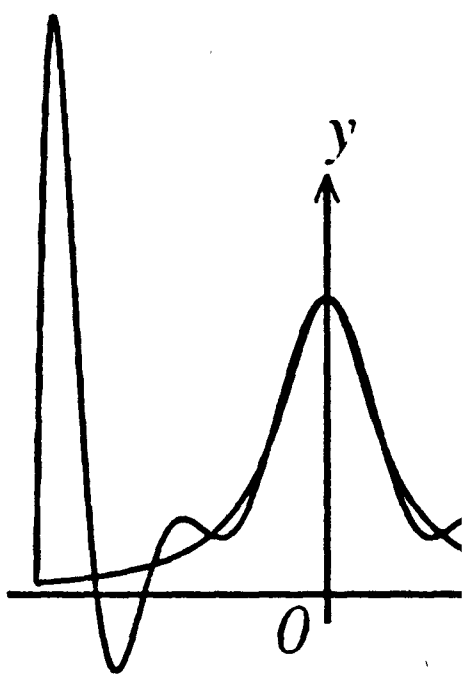


Рис. 1.6

Пример 2. (Бернштейн, 1912). Последовательность интерполяционных многочленов Лагранжа $L_m(x)$, построенных на равномерных сетках ω_m для непрерывной функции $f(x) = |x|$ на отрезке [-1, 1], с возрастанием числа узлов *m не стремится* к функции $f(x)$ (рис. 1.7).

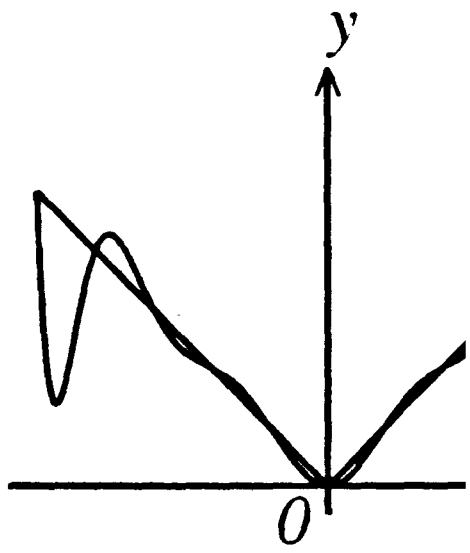


Рис. 1.7

Б. Кусочно-линейная интерполяция

При отказе от гладкости интерполируемой функции соотношение между числом достоинств и числом недостатков можно заметно изменить в сторону первых.

Построим кусочно-линейную функцию путем последовательного соединения точек (x_i, y_i) прямолинейными отрезками (рис. 1.8).

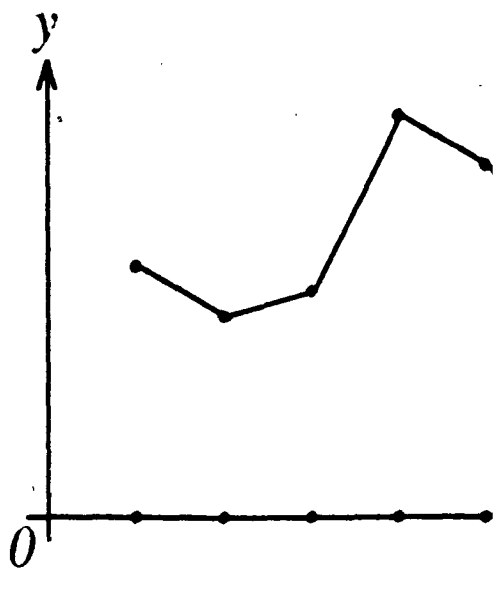


Рис. 1.8

Основные достоинства 2-го подхода:

1) график кусочно-линейной функции проходит через ка массива;

2) конструируемая функция легко описывается (число подлежащих определению коэффициентов соответствующих линейных функций для сетки ω равно $2m$);
3) заданным массивом построенная функция определена однозначно;
4) степень многочленов, используемых для описания интерполяционной функции, не зависит от числа узлов сетки (равна единице);
5) изменение одной точки в массиве требует вычисления четырех чисел (коэффициентов двух прямолинейных звеньев, исходящих из новой точки);
6) добавление дополнительной точки в массив требует вычисления четырех коэффициентов.

Кусочно-линейная функция достаточно хорошо ведет себя и при измельчении сетки.

Основной недостаток 2-го подхода:
аппроксимирующая кусочно-линейная функция не является гладкой: первые производные терпят разрыв в узлах сетки (*узлах интерполяции*).

В. Сплайн-интерполяция

Предложенные подходы можно объединить так, чтобы число перечисленных достоинств обоих подходов сохранилось при одновременном уменьшении числа недостатков. Это можно сделать путем построения гладкой интерполяционной сплайн-функции степени p.

Основные достоинства 3-го подхода:
1) график построенной функции проходит через каждую точку массива;
2) конструируемая функция сравнительно легко описывается (число подлежащих определению коэффициентов соответствующих многочленов для сетки ω равно $m(p + 1)$;
3) заданным массивом построенная функция определена однозначно;
4) степень многочленов не зависит от числа узлов сетки и, следовательно, не изменяется при его увеличении;
5) построенная функция имеет непрерывные производные до порядка $p - 1$ включительно;
6) построенная функция обладает хорошими аппроксимационными свойствами.

Краткая справка

Предложенное название - *сплайн* - не является случайным: введенные нами гладкие кусочно-полиномиальные функции и чертежные сплайны тесно связаны.

Рассмотрим гибкую идеально тонкую линейку, проходящую через расположенные на плоскости (x, y) опорные точки массива (x_i, y_i), $i = 0, 1, ..., m\text{-}1, m$ (рис. 1.9). Согласно закону Бернулли-Эйлера линеаризованное уравнение изогнутой линейки имеет вид

$$EIS''(x) = -M(x),$$

где $S(x)$ - изгиб; $M(x)$ - *изменяющийся линейно* от опоры к опоре изгибающий момент; EI - жесткость линейки.

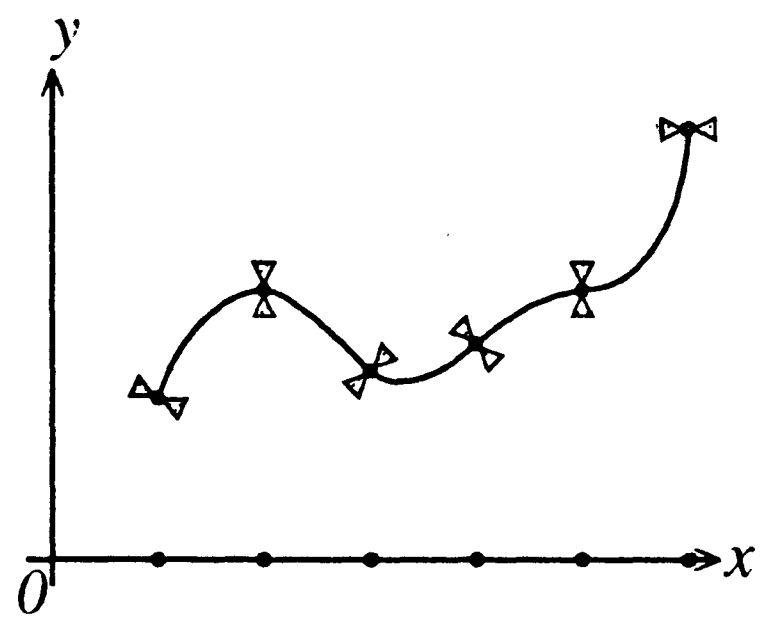


Рис. 1.9

Функция $S(x)$, описывающая форму линейки, является многочленом 3-й степени между каждыми двумя соседними точками массива (опорами) и дважды непрерывно дифференцируема на всем промежутке $[a, b]$.

Комментарий: об интерполировании непрерывной функции.

В отличие от интерполяционных многочленов Лагранжа, последовательность интерполяционных кубических сплайнов на равномерной сетке всегда сходится к интерполируемой непрерывной функции, причем с улучшением дифференциальных свойств этой функции скорость сходимости повышается.

Пример. Для функции

$$f(x) = \frac{1}{1 + 25x^2}$$

кубический сплайн на сетке с числом узлов $m = 6$ дает *погрешность аппроксимации* того же порядка, что и интерполяционный многочлен $L_5(x)$, а на сетке с числом узлов $m = 21$ эта погрешность настолько мала, что в масштабе обычного книжного рисунка просто не может быть показана (рис. 1.10) (интерполяционный многочлен $L_{20}(x)$ дает в этом случае погрешность около 10 000 %).

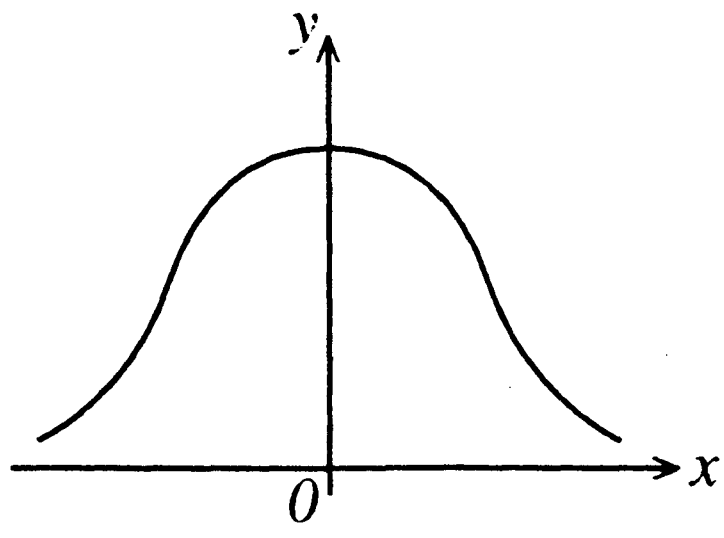


Рис. 1.10

1.1.7. Свойства интерполяционного кубического сплайна

А. Аппроксимационные свойства кубического сплайна

Аппроксимационные свойства интерполяционного сплайна зависят от гладкости функции $f(x)$ - чем выше гладкость интерполируемой функции, тем выше порядок аппроксимации и при измельчении сетки тем выше скорость сходимости.

Если интерполируемая функция $f(x)$ непрерывна на отрезке $[a, b]$, то есть $f(x) \in C^0[a,b]$, то

$$\|f(x) - S(x)\|_C = \max_{x \in [a,b]} |f(x) - S(x)| \to 0$$

при

$$h = \max_{0 \le i \le N-1} h_i \to 0 .$$

Если интерполируемая функция $f(x)$ имеет на отрезке $[a, b]$ непрерывную 1-ю производную, то есть $f(x) \in C'[a,b]$, а $S(x)$ - интерполяционный сплайн, удовлетворяющий граничным условиям 1-го или 3-го типа, то при $h \to 0$ имеем

$$\|f(x) - S(x)\|_C = o(h), \qquad \|f'(x) - S'(x)\|_C = o(1).$$

В этом случае не только сплайн сходится к интерполируемой функции, но и производная сплайна сходится к производной этой функции.

В случае если $f(x) \in C^4[a,b]$, сплайн $S(x)$ аппроксимирует на отрезке $[a, b]$ функцию $f(x)$, а его 1-я и 2-я производные аппроксимируют соответственно функции $f'(x)$ и $f''(x)$:

$$\|f(x) - S(x)\|_C = o(h^4), \qquad \|f'(x) - S'(x)\|_C = o(h^3),$$

$$\|f''(x) - S''(x)\|_C = o(h^2).$$

Б. Экстремальное свойство кубического сплайна

Интерполяционный кубический сплайн обладает еще одним полезным свойством. Рассмотрим следующую задачу.

Задача. Построить функцию $f(x)$, минимизирующую функционал

$$J(f) = \int_a^b (f''(x))^2 \, dx$$

на классе функций из пространства $C^2[a,b]$, графики которых проходят через точки массива (x_i, y_i), $i = 0, 1, ..., m$.

Решение. Среди всех функций, проходящих через опорные точки $(x_i, f(x_i))$ и принадлежащих указанному пространству, именно кубический сплайн $S(x)$, удовлетворяющий краевым условиям

$$S''(a) = S''(b) = 0,$$

доставляет экстремум (минимум) функционалу $J(f)$.

Замечания:

1. Часто именно это экстремальное свойство берут в качестве определения интерполяционного кубического сплайна.

2. Интересно отметить, что интерполяционный кубический сплайн обладает описанным выше экстремальным свойством на очень широком классе функций, а именно на классе $W_2^2[a,b]$.

В. Построение интерполяционных сплайновых кривых при помощи сплайн-функций

Выше рассматривались массивы, точки которых были занумерованы так, что их абсциссы образовывали строго возрастающую последовательность. Например, случай, изображенный на рис. 1.11, когда у разных точек массива одинаковые абсциссы, не допускался. Это обстоятельство определяло и выбор класса аппроксимирующих кривых (графики функций) и способ их построения.

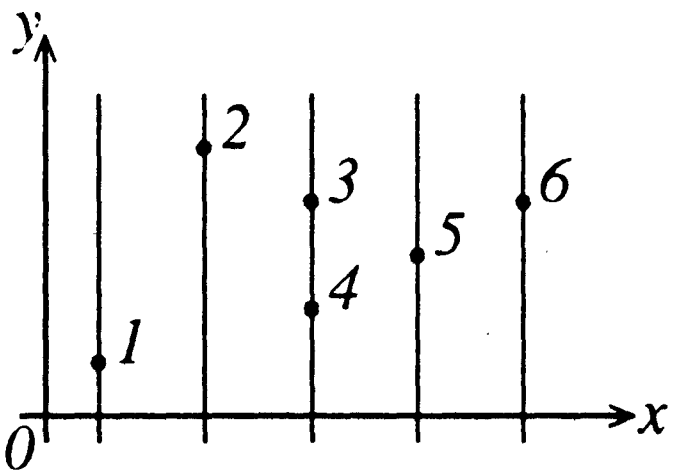


Рис. 1.11

Однако предложенный выше метод позволяет достаточно успешно строить интерполяционную кривую и в более общем случае, когда нумерация точек массива

$$\mathbf{P} = \{\mathbf{P}_i(x_i, y_i),\ i = 0,\ 1,\ \ldots,\ m\}$$

и их расположение на плоскости, как правило, не связаны (рис. 1.12). Более того, ставя задачу построения интерполяционной кривой, можно считать заданный массив неплоским, то есть

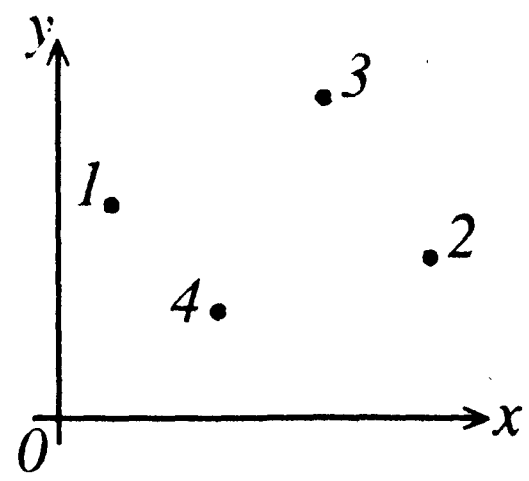


Рис. 1.12

$$\mathbf{P} = \{\mathbf{P}_i(x_i, y_i, z_i),\ i = 0,\ 1,\ \ldots,\ m\}.$$

Ясно, что для решения этой общей задачи необходимо существенно расширить класс допустимых кривых, включив в него и замкнутые

кривые, и кривые, имеющие точки самопересечения, и пространственные кривые. Такие кривые удобно описывать при помощи параметрических уравнений

$$x = x(t), \quad y = y(t), \quad z = z(t), \qquad (1.3)$$
$$\alpha \le t \le \beta .$$

Потребуем дополнительно, чтобы функции $x(t)$, $y(t)$ и $z(t)$ обладали достаточной гладкостью, например принадлежали классу $C^1[\alpha,\beta]$ или классу $C^2[\alpha,\beta]$.

Для отыскания параметрических уравнений кривой, последовательно проходящей через все точки массива, поступают следующим образом.

1-й шаг. На произвольно взятом отрезке $[\alpha, \beta]$ изменения параметра t вводится вспомогательная сетка

$$\alpha = t_0 < t_1 < \ldots < t_{m-1} < t_m = \beta,$$

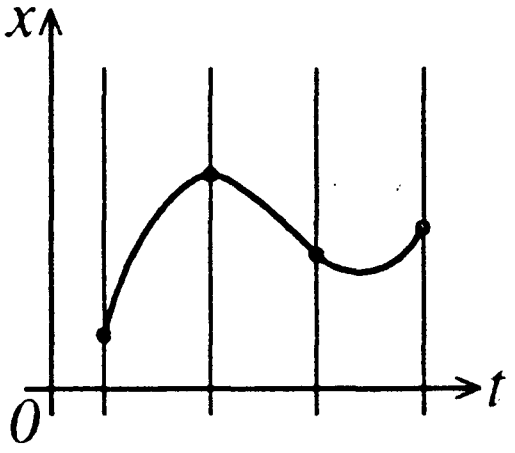
число узлов которой совпадает с числом точек в массиве **P**.

2-й шаг. По заданному массиву **P** строятся 3 новых вспомогательных массива (в плоском случае два (рис. 1.13)):

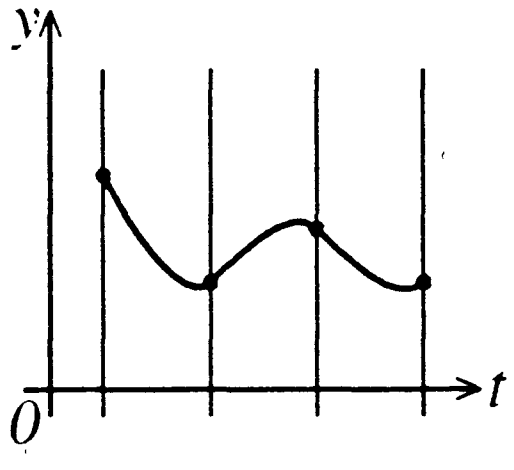
$$\mathbf{X} = \{(t_i, x_i), \ i = 0, 1, \ldots, m\},$$
$$\mathbf{Y} = \{(t_i, y_i), \ i = 0, 1, \ldots, m\},$$
$$\mathbf{Z} = \{(t_i, z_i), \ i = 0, 1, \ldots, m\}.$$

3-й шаг. Для каждого из массивов **X**, **Y** и **Z** находятся соответствующие интерполяционные сплайн-функции $x(t)$, $y(t)$ и $z(t)$.

В результате для кривой, проходящей через точки $\mathbf{P}_i$, $i = 0, 1, \ldots, m$ (рис. 1.14 - плоский случай), получаем параметрические уравнения вида (1.3).



а



б

Рис. 1.13

Замечания:

1. *Предложенный подход позволяет строить замкнутые интерполяционные кривые (при* $\mathbf{P}_0 = \mathbf{P}_m$*); для этого при построении координатных функций $x(t)$, $y(t)$ и $z(t)$ нужно использовать граничные условия 3-го типа.*

2. Полученная кривая будет гладкой, но не обязательно регулярной, так как возможность одновременного обращения в нуль производных

$$x'(t^*)=0, \quad y'(t^*)=0, \quad z'(t^*)=0$$

для некоторого $t^* \in (\alpha,\beta)$ *исключать нельзя.*

Кроме того, эта кривая может иметь точки самопересечения.

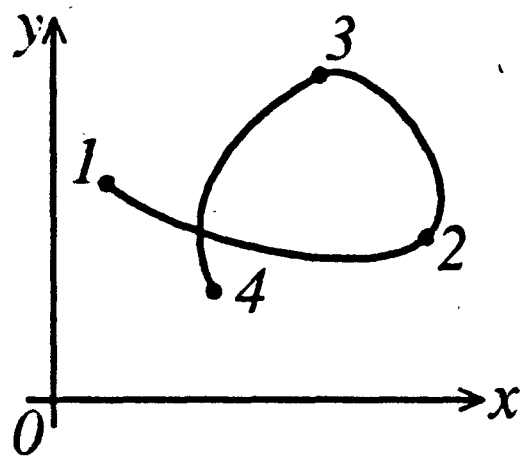


Рис. 1.14

1.1.8. Программная реализация

Описанный выше алгоритм построения интерполяционного кубического сплайна и вычисления его значений в произвольной точке реализован в виде подпрограммы Spline, написанной на языке Фортран:

```
      subroutine Spline (n,x,y,a,b,c,d,z,u,v,w,
     *                   t,s,ind,ib,ax,bx,xx,sp,dsp,d2sp)
c*****************************************************
c     Программа Spline предназначена для построения интер- *
c     поляционного кубического сплайна                     *
c                                                          *
c      n - число узлов                                     *
c      x - массив (длины n), содержащий узлы интерполяции  *
c      y - массив (длины n), содержащий значения функции   *
c          в узлах интерполяции                            *
c    a,b,                                                  *
c    c,d - рабочие массивы (длины n)                       *
c    s,t,                                                  *
c    u,v - рабочие массивы (длины n+1)                     *
c      z - массив (длины n), содержащий параметры сплайна  *
c                                                          *
c    ind - указатель режима работы:                        *
c          ind=0 - программа вычисляет параметры сплайна  *
c          ind=1 - параметры сплайна известны              *
c     xx - точка, в которой вычисляются значения сплайна   *
c          и двух его первых производных                   *
c     ib - тип граничных условий                           *
c                ind = i (i=1-4) - условия i-го типа       *
c     ax,bx - значения параметров, входящих в граничные    *
c          условия 1-го и 2-го типа                        *
c          Результат:                                      *
c     sp   - значение сплайна в точке xx                   *
c     dsp  - значение 1-й производной сплайна в            *
c            точке xx                                      *
c     d2sp - значение 2-й производной сплайна в            *
c            точке xx                                      *
c*****************************************************
        dimension x(1),y(1),a(1),b(1),c(1),d(1),
     *            z(1),u(1),v(1), w(1), s(1), t(1)
c
        ip = 1
        if (ind.eq.0) then
```

```
            a(1) =  2.
            ne   =  n
            ns   =  2
            nf   =  n - 1
      select case (ib)
c
        case (1)
          b(1 ) = 0.
          c(1) =  0.
          d(1) =  2.*ax
          a(n) =  2.
          b(n) =  0.
          c(n) =  0.
          d(n) =  2.*bx
c
        case (2)
          b(1) =  1.
          c(1) =  0.
          h1   =  x(2)-x(1)
          d(1) =  3.*(y(2)-y(1))/h1 - 0.5*h1*ax
          a(n) =  2.
          b(n) =  0.
          c(n) =  1.
          h1   =  x(n)-x(n-1)
          d(n) =  3.*(y(n)-y(n-1))/h1 + 0.5*h1*bx
c
        case (3)
          h1   = x(2) - x(1)
          h2   = x(n) - x(n-1)
          am   = h2/(h1 + h2)
          al   = 1. - am
          b(1) = am
          c(1) = al
          d(1) = 3.*(am*(y(2) - y(1))/h1 +
     *                        al*(y(1) - y(n-1))/h2)
          h1   = x(n-1) - x(n-2)
          h2   = x(n) - x(n-1)
          am   = h1/(h1 + h2)
          al   = 1. - am
          a(n-1) = 2.
          b(n-1) = am
          c(n-1) = al
          d(n-1) = 3.*(am*(y(n) - y(n-1))/h2 +
     *                      al*(y(n-1) - y(n-2))/h1)
          nf = n - 2
          ne = n - 1
c
        case (4)
          h1   = x(2)-x(1)
          h2   = x(3)-x(2)
          g0   = h1/h2
               a(2) = 1. + g0
          b(2) = g0
          c(2) = 0.
               am   = h1/(h1+h2)
               al   = 1.- am
          cc   = am*(y(3)-y(2))/h2 + al*(y(2)-y(1))/h1
          d(2) = cc + 2.*g0*(y(3)-y(2))/h2
          h2   = x(n)-x(n-1)
          h1   = x(n-1)-x(n-2)
          gn   = h1/h2
```

```
            a(n-1) = 1. + gn
            b(n-1) = 0.
            c(n-1) = gn
               am = h1/(h1+h2)
               al = 1. - am
            cc = am*(y(n)-y(n-1))/h2 +al*(y(n-1)-y(n
            d(n-1) = cc + 2.*gn*(y(n-1)-y(n-2))/h1
            ns = 3
            nf = n - 2
            ne = n - 2
               ip = 2
      end select
c
         do  j  = ns ,nf
           h1   = x(j + 1) - x(j)
           h2   = x(j) - x(j-1)
           am   = h2/(h2 + h1)
           al   = 1. - am
           c(j) = al
           a(j) = 2.
           b(j) = am
           d(j) = 3.*(am*(y(j+1) - y(j))/h1 +
     *             al*(y(j) - y(j-1))/h2)
         enddo
c
           call Progon3 (a(ip),b(ip),c(ip),d(ip),u,
     *                   v,w,s,t,z(ip),ne)
c
       select case (ib)
c
          case (3)
           z(n) = z(1)
c
          case (4)
           z(1) = g0**2*z(3)+(g0**2-1.)*z(2)+
     *             2.*((y(2)-y(1))/(x(2)-x(1))
     *            -g0**2*(y(3)-y(2))/(x(3)-x(2)))

           z(n) = gn**2*z(n-2)+(gn**2-1.)*z(n-1)+
     *             2.*((y(n)-y(n-1))/(x(n)-x(n-1))
     *            -gn**2*(y(n-1)-y(n-2))
     *             /(x(n-1)-x(n-2)))
       end select
         endif
c
         do  j = 2,n
             nj= j
             if(x(j).gt.xx) go to 1
         enddo
c
  1      j  = nj - 1
         h  = x(j+1) - x(j)
         tt = (xx - x(j))/h
         rp = (y(j+1) - y(j))/h
         aa = -2.*rp + z(j) + z(j+1)
         bb = -aa +rp - z(j)
         sp = y(j) + (xx - x(j))*(z(j) + tt*(bb + tt
         dsp= z(j) + tt*(bb + aa*tt) + tt*(bb + 2.*a
         d2sp = (2.*bb + 6.*aa*tt)/h
      return
      end
```

Обращение к программе имеет вид:

```
call Spline(n,x,y,a,b,c,d,z,u,v,w,t,s,ind,ib,ax,bx,xx,sp,dsp,dsp2)
```

Входные данные:

n - число узлов интерполяции,
x - массив размерности n, содержащий узлы интерполяции,
y - массив размерности n, содержащий значения функции в узлах интерполяции,
ax, bx - значения параметров, входящих в граничные условия 1-го и 2-го типа,
xx - точка, в которой вычисляются значения сплайна и его производных
ind - указатель режима работы:
ind = 0 - программа вычисляет коэффициенты сплайна,
ind = 1 - коэффициенты сплайна известны, программа вычисляет значения сплайна и двух его первых производных в точке xx ,
ib - указатель типа граничных условий
ib = i (i=1-4) граничные условия i-го типа

Результат:

z - массив размерности n, содержащий коэффициенты сплайна,
sp - значение сплайна в точке xx,
dsp - значение 1-й производной сплайна в точке xx ,
dsp2 - значение 2-й производной сплайна в точке xx.

Вспомогательные переменные:

a,b,c,d рабочие массивы размерности n ,
s,t,u,v - рабочие массивы размерности n+1.

Программа Spline обращается к подпрограмме решения системы линейных уравнений с трехдиагональной матрицей Progon3 (см. приложение А).

Перед обращением к программе Spline необходимо:

1) задать размерности массивов x, y, z, a, b, c, d, u, v, не забывая, что размерности массивов u и v на единицу больше размерностей массивов x, y и т. д.;
2) присвоить фактические значения элементам массивов x и y и переменным n, ib и ind (ind = 0 при первом обращении к программе, ind = 1 при повторных обращениях).

Пример. Построим на отрезке $[0,2\pi]$ равномерную сетку из 21 узла и вычислим значения функции $f(x)=\sin(x)$ в этих узлах. По этим данным построим интерполяционный кубический сплайн, удовлетворяющий граничным условиям 1-го типа, и сравним значения заданной функции и ее двух первых производных со значениями построенного сплайна и двух его первых производных в точках

$$\tilde{x}_i = \frac{x_i + x_{i+1}}{2}, \qquad i = 0,\ 1,\ \ldots,\ 19.$$

Программа для решения этой задачи может иметь следующий вид:

```
      dimension x(81),y(81),a(82),b(82),c(82),d(82),
     *             z(82),u(82),v(82),w(82),s(82),t(82)
            data pi /3.1415926/
              n = 21
              h = 2.*pi/20
     c
              do j = 1,21
                x(j) = h*(j-1)
                y(j) = sin(x(j))
              enddo
              ib = 1
              ax = 1.
              bx = 1.
            call spline (n,x,y,a,b,c,d,z,u,v,w,t,s,
           *                0,ib,ax,bx,xx,sp,dsp,d2sp)
              del0 = 0.
              del1 = 0.
              del2 = 0.
              do j = 1,41
                h  = 2.*pi/40.
                xx = h*(j-1)
                yy = sin(xx)
                d1 = cos(xx)
                d2 =-sin(xx)
              call spline (n,x,y,a,b,c,d,z,u,v,w,t,s,
           *                1,ib,ax,bx,xx,sp,dsp,d2sp)
              if (abs(yy-sp).gt.del0) del0= abs(yy-sp)
              if (abs(d1-dsp).gt.del1) del1= abs(d1-dsp)
              if (abs(d2-d2sp).gt.del2) del2= abs(d2-d2sp)
              enddo
              print *,' max|f(x)-S(x)|= ', del0
              print *,' max|f''(x)-S''(x)|= ', del1
              print *,' max|f"(x)-S"(x)|= ', del2
              stop
              end
```

Вычисления по этой программе привели к следующим результатам:

$$\max|f(x)-S(x)| = 2.568960E\text{-}05,$$
$$\max|f'(x)-S'(x)| = 6.669760E\text{-}05,$$
$$\max|f''(x)-S''(x)| = 8.250833E\text{-}03.$$

Максимум вычислялся на множестве точек $\tilde{x}_i$. Эти результаты показывают, что интерполяционный кубический сплайн действительно обладает хорошими аппроксимационными свойствами.

Следующая программа строит интерполяционный сплайн, удовлетворяющий граничным условиям 2-го типа,

```
      dimension x(81),y(81),a(82),b(82),c(82),d(82),
     *             z(82),u(82),v(82),w(82),s(82),t(82)
            data pi /3.1415926/
              n = 21
              h = 2.*pi/20
```

```
c
      do j = 1,21
          x(j) = h*(j-1)
          y(j) = sin(x(j))
      enddo
      ib = 2
      ax = 0.
      bx = 0.
      call spline (n,x,y,a,b,c,d,z,u,v,w,t,s,
     *             0,ib,ax,bx,xx,sp,dsp,d2sp)
      del0 = 0.
      del1 = 0.
      del2 = 0.
      do j = 1,41
        h  = 2.*pi/40.
        xx = h*(j-1)
        yy = sin(xx)
        d1 = cos(xx)
        d2 =-sin(xx)
      call spline (n,x,y,a,b,c,d,z,u,v,w,t,s,
     *             1,ib,ax,bx,xx,sp,dsp,d2sp)
      if (abs(yy-sp).gt.del0) del0= abs(yy-sp)
      if (abs(d1-dsp).gt.del1) del1= abs(d1-dsp)
      if (abs(d2-d2sp).gt.del2) del2= abs(d2-d2sp)
      enddo
      print *,' max|f(x)-S(x)|= ', del0
      print *,' max|f''(x)-S''(x)|= ', del1
      print *,' max|f"(x)-S"(x)|= ', del2
      stop
      end
```

Вычисления по этой программе привели к следующим результатам:

$$\max|f(x)-S(x)| = 2.574921E\text{-}05,$$
$$\max|f'(x)-S'(x)| = 5.531311E\text{-}05,$$
$$\max|f''(x)-S''(x)| = 8.254051E\text{-}03.$$

Максимум вычислялся на множестве точек $\tilde{x}_i$ из предыдущего примера.

Если требуется по тому же массиву данных построить интерполяционный сплайн, удовлетворяющий граничным условиям 3-го типа, то можно воспользоваться программой, текст которой приведен ниже.

```
      dimension x(81),y(81),a(82),b(82),c(82),d(82),
     *          z(82),u(82),v(82),w(82),s(82),t(82)
        data pi /3.1415926/
          n = 21
          h = 2.*pi/20
c
        do j = 1,21
          x(j) = h*(j-1)
          y(j) = sin(x(j))
        enddo
        ib = 3
```

```
      call spline (n,x,y,a,b,c,d,z,u,v,w,t,s,
     *                  0,ib,ax,bx,xx,sp,dsp,d2sp)
      del0 = 0.
      del1 = 0.
      del2 = 0.
      do j = 1,41
        h  = 2.*pi/40.
        xx = h*(j-1)
        yy = sin(xx)
        d1 = cos(xx)
        d2 =-sin(xx)
      call spline (n,x,y,a,b,c,d,z,u,v,w,t,s,
     *                  1,ib,ax,bx,xx,sp,dsp,d2sp)
      if (abs(yy-sp).gt.del0) del0= abs(yy-sp)
      if (abs(d1-dsp).gt.del1) del1= abs(d1-dsp)
      if (abs(d2-d2sp).gt.del2) del2= abs(d2-d2sp)
      enddo
      print *,' max|f(x)-S(x)|= ', del0
      print *,' max|f''(x)-S''(x)|= ', del1
      print *,' max|f"(x)-S"(x)|= ', del2
      stop
      end
```

Вычисления по этой программе привели к следующим результатам:

$$\max|f(x)-S(x)| = 2.574921E\text{-}05,$$
$$\max|f'(x)-S'(x)| = 5.501509E\text{-}05,$$
$$\max|f''(x)-S''(x)| = 8.254051E\text{-}03.$$

Максимум вычислялся на множестве точек $\tilde{x}_i$ из предыдущего примера.

Следующая программа строит и тестирует интерполяционный сплайн, удовлетворяющий граничным условиям 4-го типа.

```
      dimension x(81),y(81),a(82),b(82),c(82),d(82),
     *          z(82),u(82),v(82),w(82),s(82),t(82)
        data pi /3.1415926/
          n = 21
          h = 2.*pi/20
c
        do j = 1,21
           x(j) = h*(j-1)
           y(j) = sin(x(j))
        enddo
        ib = 4
        call spline  n,x,y,a,b,c,d,z,u,v,w,t,s,
     *                  0,ib,ax,bx,xx,sp,dsp,d2sp)
        del0 = 0.
        del1 = 0.
        del2 = 0.
        do j = 1,41
          h  = 2.*pi/40.
          xx = h*(j-1)
          yy = sin(xx)
          d1 = cos(xx)
          d2 =-sin(xx)
```

```
      call spline (n,x,y,a,b,c,d,z,u,v,w,t,s,
     *             1,ib,ax,bx,xx,sp,dsp,d2sp)
      if (abs(yy-sp).gt.del0) del0= abs(yy-sp)
      if (abs(d1-dsp).gt.del1) del1= abs(d1-dsp)
      if (abs(d2-d2sp).gt.del2) del2= abs(d2-d2sp)
      enddo
      print *,' max|f(x)-S(x)|= ', del0
      print *,' max|f''(x)-S''(x)|= ', del1
      print *,' max|f"(x)-S"(x)|= ', del2
      stop
      end
```

Вычисления по этой программе привели к следующим результатам:

$$\max|f(x)-S(x)| = 8.162856E\text{-}05,$$
$$\max|f'(x)-S'(x)| = 1.665711E\text{-}03,$$
$$\max|f''(x)-S''(x)| = 1.897535E\text{-}02.$$

Максимум вычислялся на множестве точек $\tilde{x}_i$ из основного примера.

Текст программы Spline находится в файле spline.for в поддиректории INTERPOL на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". В эту же поддиректорию помещены файлы, содержащие примеры применения программы Spline при других граничных условиях. Подробную информацию об именах этих файлов и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

1.2. Сглаживающие кубические сплайны

1.2.1. О постановке задачи сглаживания

Пусть заданы сетка ω и набор чисел

$$y_0, y_1, \ldots, y_{m-1}, y_m.$$

Комментарий к исходным данным. На практике часто приходится иметь дело со случаем, когда значения y_i в массиве

$$(x_i, y_i), \qquad i = 0, 1, \ldots, m,$$

заданы с некоторой погрешностью. Фактически это означает, что для каждого $i = 0, 1, \ldots, m$ указан интервал

$$(c_i, d_i) \text{ или } (y_i - \delta_i, y_i + \delta_i)$$

и любое число из этого интервала может быть взято в качестве значения y_i. Величины y_i удобно интерпретировать, например, как результаты измерений некоторой функции $y(x)$ при заданных значениях переменной x, содержащие случайную погрешность. При решении задачи восстановления функции по таким ее "экспериментальным" значениям вряд ли целесообразно использовать интерполяцию, поскольку интерполяционная функция будет послушно воспроизводить причудливые осцилляции, обусловленные случайной компонентой в массиве y_i. Более естественным является подход, основанный на процедуре сглаживания, призванной как-то уменьшить элемент случайности в результатах измерений. Обычно в таких задачах требуется найти функцию, значения которой при $x = x_i$, $i = 0, 1, \ldots, m$, попадали бы в соответствующие интервалы и которая обладала бы, кроме того, достаточно хорошими свойствами (рис. 1.15). Например, имела бы непрерывные 1-е и 2-е производные или же ее график был бы не слишком сильно искривлен, то есть не имел сильных осцилляций.

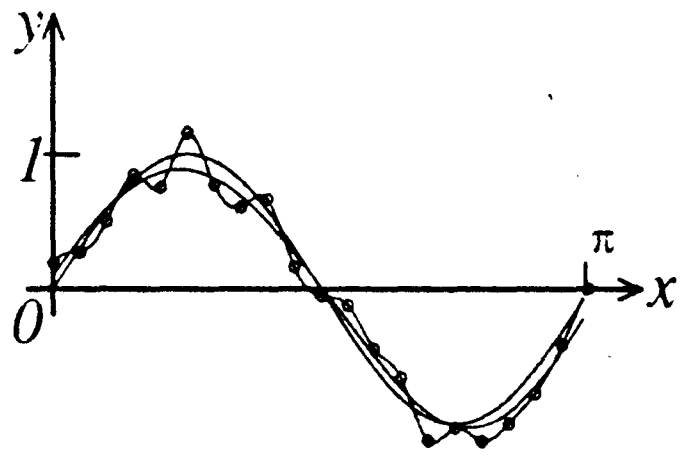


Рис. 1.15

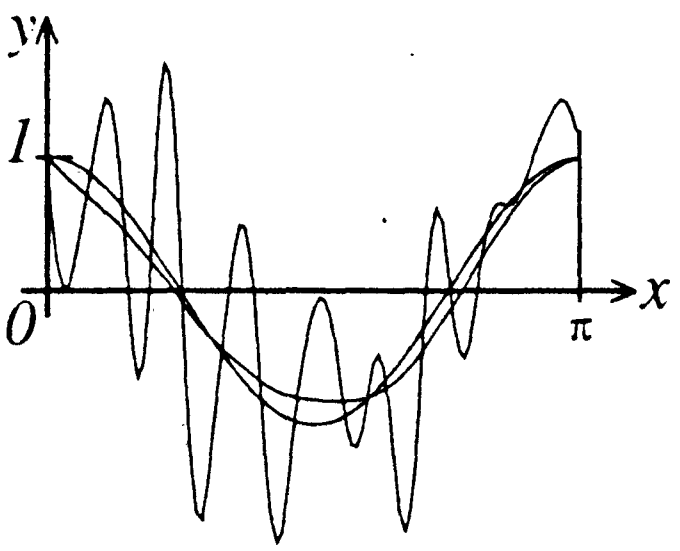


Рис. 1.16

Задача подобного рода возникает и тогда, когда по заданному (точно) массиву (x_i, y_i), $i = 0, 1, \ldots, m$, требуется построить функцию, которая проходила бы не через заданные точки, а вблизи них и к тому

же изменялась достаточно плавно (рис. 1.16). Другими словами, искомая функция как бы сглаживала заданный массив, а не интерполировала его.

Пусть заданы сетка ω и два набора чисел

$$y_0, y_1, \ldots, y_{m-1}, y_m \qquad \text{и} \qquad \Delta_0 > 0,\ \Delta_1 > 0,\ \ldots,\ \Delta_{m-1} > 0,\ \Delta_m > 0.$$

Задача. Построить на отрезке $[a, b]$ гладкую функцию $\sigma(x)$, значения которой в узлах сетки ω отличаются от чисел y_i на заданные величины $\Delta_i > 0$

$$| \sigma (x_i) - y_i | < \Delta_i,$$
$$i = 0, 1, \ldots, m\text{-}1, m.$$

Замечание

Сформулированная задача сглаживания состоит в восстановлении гладкой функции, заданной таблично. Ясно, что такая задача имеет много различных решений. Накладывая на конструируемую функцию дополнительные условия, можно добиться необходимой однозначности.

1.2.2. Определение сглаживающего кубического сплайна

Сглаживающим кубическим сплайном $S(x)$ на сетке ω называется функция, которая:

1) на каждом из отрезков $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, m\text{-}1$, представляет собой многочлен 3-й степени,
$$S(x) = S_i(x) = a_0^{(i)} + a_1^{(i)}(x - x_i) + a_2^{(i)}(x - x_i)^2 + a_3^{(i)}(x - x_i)^3;$$
2) дважды непрерывно дифференцируема на отрезке $[a, b]$, то есть принадлежит классу $C^2[a,b]$;
3) доставляет минимум функционалу
$$J(f) = \int_a^b (f''(x))^2 dx + \sum_{i=0}^{m} \frac{1}{\rho_i}(f(x_i) - y_i)^2,$$
где y_i и $\rho_i > 0$ - заданные числа;
4) удовлетворяет граничным условиям одного из трех указанных ниже типов.

1.2.3. Граничные (краевые) условия

Граничные (краевые) условия задаются в виде ограничений на значения сплайна и его производных в граничных узлах сетки ω.

Граничные (краевые) условия 1-го типа

$$S'(a) = y'_0, \quad S'(b) = y'_m$$

- на концах промежутка $[a, b]$ задаются значения 1-й производной искомой функции.

Граничные (краевые) условия 2-го типа

$$S''(a) = 0, \quad S''(b) = 0$$

- 2-е производные искомой функции на концах промежутка $[a, b]$ равны нулю.

Граничные (краевые) условия 3-го типа

$$S(a) = S(b), \quad S'(a) = S'(b), \quad S''(a) = S''(b)$$

называются *периодическими*.

Теорема. Кубический сплайн $S(x)$, минимизирующий функционал (1.1) и удовлетворяющий краевым условиям одного из указанных трех типов, определен однозначно.

Определение. Кубический сплайн, минимизирующий функционал $J(f)$ и удовлетворяющий граничным (краевым) условиям i-го типа, называется *сглаживающим сплайном i-го типа*.

Замечание

На каждом из отрезков $[x_i, x_{i+1}]$, $i = 0, 1, ..., m-1$, сплайн $S(x)$ является многочленом 3-й степени и определяется на этом отрезке четырьмя коэффициентами. Всего отрезков - m. Значит, для того, чтобы полностью определить сплайн, необходимо найти $4m$ чисел

$$a_0^{(i)}, a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, \quad i = 0, 1, ..., m-1.$$

Условие $S(x) \in C^2[a, b]$ означает непрерывность функции $S(x)$ и ее производных $S'(x)$ и $S''(x)$ во всех внутренних узлах сетки ω. Число таких узлов - m-1. Тем самым для отыскания коэффициентов всех многочленов получается $3(m-1)$ условий (уравнений).

1.2.4. Построение сглаживающего кубического сплайна

Опишем способ вычисления коэффициентов кубического сплайна, при котором число величин, подлежащих определению, равно $2m + 2$.

На каждом из промежутков $[x_i, x_{i+1}]$, $i = 0, 1, ..., m-1$, сглаживающая сплайн-функция ищется в следующем виде:

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)\left[(2-t)n_i + (1+t)n_{i+1}\right].$$

Здесь

$$h_i = x_{i+1} - x_i, \qquad t = \frac{x - x_i}{h_i},$$

а числа z_i и n_i, $i = 0, 1, ..., m$, являются решением системы линейных алгебраических уравнений, вид которой зависит от типа граничных (краевых) условий.

Опишем сначала, как находятся величины n_i.

Для *краевых условий 1-го и 2-го типа* система линейных уравнений для определения величин n_i записывается в следующем виде:

$$a_0 n_0 + b_0 n_1 + c_0 n_2 = g_0 ,$$

$$b_0 n_0 + a_1 n_1 + b_1 n_2 + c_1 n_3 = g_1 ,$$

$$c_{i-2} n_{i-2} + b_{i-1} n_{i-1} + a_i n_i + b_i n_{i+1} + c_i n_{i+2} = g_i , \qquad i = 2, 3, ..., m\text{-}2,$$

$$c_{m-3} n_{m-3} + b_{m-2} n_{m-2} + a_{m-1} n_{m-1} + b_{m-1} n_m = g_{m-1} ,$$

$$c_{m-2} n_{m-2} + b_{m-1} n_{m-1} + a_m n_m = g_m ,$$

где

$$a_i = \frac{1}{3}\left(h_{i-1} + h_i\right) + \frac{1}{h_{i-1}^2}\rho_{i-1} + \left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)^2 \rho_i + \frac{1}{h_i^2}\rho_{i+1} , \qquad i = 1, 2, ..., m\text{-}1,$$

$$b_i = \frac{1}{6}h_i - \frac{1}{h_i}\left(\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\rho_i + \left(\frac{1}{h_i} + \frac{1}{h_{i-1}}\right)\rho_{i+1}\right) , \qquad i = 1, 2, ..., m\text{-}2,$$

$$c_i = \frac{1}{h_i h_{i+1}}\rho_{i+1} , \qquad i = 1, 2, ..., m\text{-}3, \qquad (1.4)$$

$$g_i = \frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} , \qquad i = 1, 2, ..., m\text{-}1$$

($\rho_i \geq 0$ - известные числа).

Коэффициенты

$$a_0, b_0, c_0, c_{m-2}, b_{m-1}, a_m, g_m$$

зависят от выбора граничных условий.

Граничные (краевые) условия 1-го типа:

$$a_0 = \frac{h_0}{3} + \frac{1}{h_0^2}\left(\rho_0 + \rho_1\right), \qquad c_0 = \frac{1}{h_0 h_1}\rho_1 ,$$

$$b_0 = \frac{h_0}{6} + \frac{1}{h_0}\left(\frac{1}{h_0} + \frac{1}{h_1}\right)\rho_1 - \frac{1}{h_0^2}\rho_0 , \qquad g_0 = \frac{y_1 - y_0}{h_0} - y_0' ,$$

$$a_m = \frac{h_{m-1}}{3} + \frac{1}{h_{m-1}^2}\left(\rho_{m-1} + \rho_m\right), \qquad c_{m-2} = \frac{1}{h_{m-2}h_{m-1}}\rho_{m-1},$$

$$b_{m-1} = \frac{h_{m-1}}{6} - \frac{1}{h_{m-1}}\left(\frac{1}{h_{m-1}} + \frac{1}{h_{m-2}}\right)\rho_{m-2} - \frac{1}{h_{m-1}^2}\rho_m,$$

$$g_m = y'_m - \frac{y_m - y_{m-1}}{h_{m-1}}.$$

Граничные (краевые) условия 2-го типа:

$$a_0 = 1, \qquad b_0 = 1, \qquad c_0 = 0, \qquad g_0 = 0,$$

$$a_m = 1, \qquad b_{m-1} = 1, \qquad c_{m-2} = 0, \qquad g_m = 0.$$

В случае граничных (краевых) условий 3-го типа система для определения чисел n_i, $i = 1, 2, \ldots, m$, записывается так:

$$c_{i-2}n_{i-2} + b_{i-1}n_{i-1} + a_i n_i + b_i n_{i+1} + c_i n_{i+2} = g_i, \qquad i = 1, 2, \ldots, m,$$

причем все коэффициенты вычисляются по формулам (1.4) (величины с индексами k и $m + k$ считаются равными: $n_0 = n_m$, $h_0 = h_m$, $a_0 = a_m$ и т. д.).

Важное замечание

Матрицы систем невырожденны, и потому каждая из этих систем имеет единственное решение.

Если числа n_i найдены, то величины z_i легко определяются по формулам

$$z_i = y_i - \rho_i D_i, \qquad i = 1, 2, \ldots, m,$$

где

$$D_0 = \frac{1}{h_0}\left(n_1 - n_0\right), \qquad D_m = -\frac{1}{h_{m-1}}\left(n_m - n_{m-1}\right),$$

$$D_i = \frac{1}{h_i}\left(n_{i+1} - n_i\right) - \frac{1}{h_{i-1}}\left(n_i - n_{i-1}\right), \qquad i = 1, 2, \ldots, m\text{-}1.$$

В случае периодических граничных (краевых) условий

$$h_m = h_0, \qquad n_m = n_0, \qquad n_1 = n_{m+1}$$

и

$$D_0 = D_m = \frac{1}{h_m}\left(n_1 - n_m\right) - \frac{1}{h_{m-1}}\left(n_m - n_{m-1}\right).$$

1.2.5. Выбор весовых коэффициентов

Выбор весовых коэффициентов ρ_i, входящих в функционал (1.4), позволяет в известной степени управлять свойствами сглаживающих сплайнов.

Если все $\rho_i = 0$, то $z_i = y_i$ и сглаживающий сплайн оказывается интерполяционным. Это, в частности, означает, что чем точнее заданы величины y_i, тем меньше должны быть соответствующие весовые коэффициенты. Если же необходимо, чтобы сплайн прошел через точку (x_k, y_k), то отвечающий ему весовой множитель ρ_k следует положить равным нулю.

В практических вычислениях наиболее важным является выбор величин ρ_i.

Пусть Δ_i - погрешность измерения величины y_i. Тогда естественно потребовать, чтобы сглаживающий сплайн $\sigma(x)$ удовлетворял условию

$$\left| \sigma(x_i) - y_i \right| \le \Delta_i \qquad (1.5)$$

или, что то же $\rho_i \left| D_i \right| \le \Delta_i$.

В простейшем случае весовые коэффициенты ρ_i можно задать, например, формулой

$$\rho_i = c\Delta_i,$$

где c - некоторая достаточно малая постоянная. Однако такой выбор весов ρ_i не позволяет использовать "коридор", обусловленный погрешностями величин y_i. Более рациональный, но и более трудоемкий алгоритм определения величин ρ_i может выглядеть следующим образом.

Если на k-й итерации величины $D_i^{(k)}$ найдены, то полагают

$$\rho_i^{(k+1)} = \begin{cases} \dfrac{\Delta_i}{\left| D_i^{(k)} \right|} & \text{при } \left| D_i^{(k)} \right| > \varepsilon, \\ 0 & \text{при } \left| D_i^{(k)} \right| \le \varepsilon \end{cases}$$

где ε - малое число, которое выбирается экспериментально с учетом разрядной сетки компьютера, значений Δ_i и точности решения системы линейных алгебраических уравнений.

Если на k-й итерации в точке x_i нарушилось условие (1.5), то последняя формула обеспечит уменьшение соответствующего весового коэффициента ρ_i. Если же

$$\left| \sigma^{(k)}(x_i) - y_i \right| < \Delta_i,$$

то на следующей итерации $\rho_i^{(k+1)} > \rho_i^{(k)}$. Увеличение ρ_i приводит к более полному использованию "коридора" (1.5) и в конечном счете к более плавно изменяющемуся сплайну.

1.2.6. Построение сглаживающих сплайновых кривых при помощи сплайн-функций

Выше рассматривались массивы, точки которых были занумерованы так, что их абсциссы образовывали строго возрастающую последовательность. Например, случай, когда у разных точек массива одинаковые абсциссы, не допускался. Это обстоятельство определяло и выбор класса сглаживающих кривых (графики функций) и способ их построения.

Однако предложенный выше метод позволяет достаточно успешно строить сглаживающую кривую и в более общем случае, когда нумерация точек массива

$$\mathbf{P} = \left\{\mathbf{P}_i(x_i, y_i),\ i = 0,\ 1,\ \ldots,\ m\right\}$$

и их расположение на плоскости не связаны. Более того, ставя задачу построения сглаживающей кривой, можно считать заданный массив неплоским, то есть

$$\mathbf{P} = \left\{\mathbf{P}_i(x_i, y_i, z_i),\ i = 0,\ 1,\ \ldots,\ m\right\}.$$

Ясно, что для решения этой общей задачи необходимо существенно расширить класс допустимых кривых, включив в него и замкнутые кривые, и кривые, имеющие точки самопересечения, и пространственные кривые. Такие кривые удобно описывать при помощи параметрических уравнений

$$x = x(t), \qquad y = y(t), \qquad z = z(t), \tag{1.6}$$
$$\alpha \le t \le \beta.$$

Потребуем дополнительно, чтобы функции $x(t)$, $y(t)$ и $z(t)$ обладали достаточной гладкостью, например принадлежали классу $C^1[\alpha, \beta]$ или классу $C^2[\alpha, \beta]$.

Для отыскания параметрических уравнений сглаживающей кривой поступают так:

1-й шаг. На произвольно взятом отрезке $[\alpha, \beta]$ изменения параметра t вводится вспомогательная сетка

$$\alpha = t_0 < t_1 < \ldots < t_{m-1} < t_m = \beta,$$

число узлов которой совпадает с числом точек в массиве **P**.

2-й шаг. По заданному массиву **P** строится 3 новых вспомогательных массива (в плоском случае два):

$$\mathbf{X}=(t_i, x_i),\ \ i = 0, 1, ..., m,$$
$$\mathbf{Y}=(t_i, y_i),\ \ i = 0, 1, ..., m,$$
$$\mathbf{Z}=(t_i, z_i),\ \ i = 0, 1, ..., m.$$

3-й шаг. Для каждого из массивов **X**, **Y** и **Z** находятся соответствующие сглаживающие сплайн-функции $x(t)$, $y(t)$ и $z(t)$.

В результате для кривой, сглаживающей массив $\mathbf{P}_i$, $i = 0, 1, ..., m$, получаем параметрические уравнения вида (1.6) (см. рис. 1.14, плоский случай).

Замечания:

1. Предложенный подход позволяет строить замкнутые сглаживающие кривые (при $\mathbf{P}_0 = \mathbf{P}_m$); для этого при построении координатных функций x(t), y(t) и z(t) нужно использовать граничные условия 3-го типа.

2. Полученная кривая будет гладкой, но не обязательно регулярной, так как возможность одновременного обращения в нуль производных,

$$x'\left(t^*\right)=0,\quad y'\left(t^*\right)=0,\quad z'\left(t^*\right)=0$$

для некоторого $t^* \in (\alpha, \beta)$ исключать нельзя. Кроме того, эта кривая может иметь точки самопересечения.

1.2.7. Программная реализация

Описанный выше алгоритм построения сглаживающего кубического сплайна и вычисления его значений в произвольной точке реализован в виде подпрограммы Smspline, написанной на языке Фортран:

```
      subroutine Smspline(n,ib,ind,x,y,a,b,c,d,e,g,rho,
     *     ax,bx,u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
c
c*************************************************************
c     Программа Smspline строит сглаживающий сплайн         *
c                по заданному массиву точек                 *
c                                                           *
c  n   - число точек в массиве исходных данных              *
c  x   - массив длины n, содержащий x-координаты точек      *
c  y   - массив длины n, содержащий y-координаты точек      *
c  rho - массив длины n, содержащий весовые коэффициенты    *
c                                                           *
c  ind - код режима работы:                                 *
c        ind = 0 - программа вычисляет параметры сплайна   *
c        ind = 1 - параметры сплайна известны               *
c  ib  - код типа граничных условий                         *
c         ib = i  i=1-3 - условия i-го типа                 *
c                                                           *
c  a,b,c,d,e,g,zm,aa,dd - рабочие массивы (длины n)        *
c  u,v,w,p                                                  *
```

```
c  q,r,s,t - рабочие массивы (длины n+2)
c  ax, bx - параметры, входящие в граничные условия 1
c           типа
c
c           Результат:
c  sp    -  значение сплайна в точке xx
c  dsp   -  значение 1-й производной сплайна в точке x
c  d2sp  -  значение 2-й производной сплайна в точке x
c
c*************************************************
c
      real a(1),b(1),c(1),d(1),e(1),
     *     g(1),rho(1),aa(1),r(1)
      real v(1),w(1),p(1),q(1),t(1),s(1),zm(1),dd(1)
      real x(1), y(1)
c
       nn = n
       if (ind.eq.0) then
       select case (ib)
c
       case (1)
        h1      = x(2)-x(1)
        a(1)    = h1/3.+(rho(1)+rho(2))/h1**2
        h2      = x(3) - x(2)
        b(1)    = h1/6.-1./h1 +1./h2*rho(2)/h1
     *            - rho(1)/h1**2
        c(1)    = rho(2)/(h1*h2)
        g(1)    = (y(2) - y(1))/h1 - ax
        e(3)    = c(1)
        h1      = x(n) - x(n-1)
        h2      = x(n-1) - x(n-2)
        a(n)    = h1/3. +(rho(n-1)+rho(n))/h1**2
        b(n-1)  = h1/6.-(1./h1 +1./h2)*rho(n-1)/h1
     *            -rho(n)/h1**2
        c(n-2)  = rho(n-1)/(h1*h2)
        e(1)  = 0.
        e(2)  = 0.
        d(1)  = 0.
        d(2)  = b(1)
        c(n-1)  = 0.
        c(n)  = 0.
        b(n)  = 0.
        d(n)  = b(n-1)
        g(n)  = bx - (y(n) - y(n-1))/h1
        e(n)  = c(n-2)
c
       case (2)
        a(1)    = 1.
        a(n)    = 1.
        b(1)    = 0.
        b(n-1)  = 0.
        c(1)    = 0.
        c(n-2)  = 0.
        g(1)    = 0.
        g(n)    = 0.
        e(1)    = 0.
        e(2)    = 0.
        e(3)    = 0.
        d(1)    = 0.
        b(n)    = 0.
        c(n)    = 0.
```

```
          c(n-1) = 0.
c
         case (3)
          h1 = x(2) - x(1)
          h2 = x(n) - x(n-1)
          h3 = x(3) - x(2)
          a(1) = (h1 + h2)/3.
     *           + rho(n-1)/h2**2 + rho(2)/h1**2 +
     *             (1./h1 + 1./h2)**2*rho(1)
          b(1) = h1/6. -((1./h2 + 1./h1)*rho(1)+
     *                   (1./h2 + 1./h3)*rho(2))/h2
          c(1) = rho(1)/(h1*h3)
          g(1) = (y(2) - y(1))/h1 - (y(1) - y(n-1))/h2
          d(2) = b(1)
          e(3) = c(1)
          h4   = x(n-1) - x(n-2)
          c(n-2) = rho(n-1)/(h2*h4)
          c(n-1) = rho(n)/(h2*h1)
          b(n-1) = h2/3. - ((1./h4 + 1./h2)*rho(n-1) +
     *                      (1./h2 + 1./h1)*rho(n))/h2
          d(1) = b(n-1)
          e(1) = c(n-2)
          e(2) = c(n-1)
          g(n-1) = (y(n) - y(n-1))/h2 - (y(n-1) - y(n-2))/h4
          nn = n - 1
         end select
c
      do i = 2, n-1
          h1   = x(i)   - x(i-1)
       h2   = x(i+1) - x(i)
          a(i) = (h1+h2)/3.
       *          + rho(i-1)/h1**2 + rho(i+1)/h2**2 +
       *            (1./h1 + 1./h2)**2*rho(i)
       g(i) = (y(i+1)-y(i))/h2 - (y(i)-y(i-1))/h1
              if (i.le.(n-2)) then
                  h3 = (x(i+2)-x(i+1))
              b(i) = h2/6. - ((1./h1 + 1./h2)*rho(i)
     *                       +(1./h2 + 1./h3)*rho(i+1))/h2
              d(i+1) = b(i)
              endif
         if (i.le.(n-3)) then
              c(i) = rho(i+1)/(h2*h3)
              e(i+2) = c(i)
             endif
      enddo
c
          call Progon5
     *          (nn,a,b,c,d,e,g,u,v,w,p,q,r,s,t,aa,dd,zm)
          aa(1) = y(1) - rho(1)*(zm(2)-zm(1))/(x(2)-x(1))
      aa(n) = y(n) + rho(n)*(zm(n)-zm(n-1))/(x(n)-x(n-1))

          if (ib.eq.3) then
              zm(n) = zm(1)
              di = (zm(2)-zm(1))/(x(2)-x(1))
              di = di - (zm(n)-zm(n-1))/(x(n)-x(n-1))
              aa(1) = y(1) - rho(1)*di
              aa(n) = aa(1)
          endif
c
      do i = 2, n-1
             di = (zm(i+1)-zm(i))/(x(i+1)-x(i))
```

```
            di = di - (zm(i)-zm(i-1))/(x(i)-x(i-1))
            aa(i) = y(i) - rho(i)*di
        enddo
c
        else
c
            do  i = 2,n
                ni= i
                if(x(i).gt.xx) exit
            enddo
c
            i  = ni - 1
            h  = x(i+1) - x(i)
            tt = (xx - x(i))/h
            sp = aa(i)*(1.-tt) + aa(i+1)*tt
     *        - h**2/6.*tt*(1.-tt)*
     *          ((2.-tt)*zm(i) + (1.+tt)*zm(i+1))
            dsp= (aa(i+1) - aa(i))/h -
     *           h/6.*((2. - 6.*tt + 3.*tt**2)*zm(i)
     *         +(1. - 3.*tt**2)*zm(i+1))
            d2sp = (1. - tt)*zm(i) + tt*zm(i+1)
c
        endif
            return
        end
```

Обращение к программе имеет вид:

```
call Smspline(n,ib,ind,x,y,a,b,c,d,e,g,rho,ax,bx,
*        u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
```

Входные данные:

n - число точек в массиве исходных данных;

x - массив размерности n, содержащий x-координаты заданных точек;

y - массив размерности n, содержащий y-координаты заданных точек;

rho - массив размерности n, содержащий значения весовых коэффициентов;

ind - код режима работы:

ind = 0 - программа вычисляет параметры сплайна,

ind = 1 - параметры сплайна известны, и программа вычисляет значения сплайна и первых двух его производных в точке xx;

ax, bx - значения параметров, входящих в граничные условия 1-го типа;

xx - точка, в которой вычисляются значения сплайна и его производных;

ib - указатель типа граничных условий:

ib = i (i=1-3) граничные условия i-го типа.

Результат:

aa - массив размерности n, aa(i) содержит значения сглаживающего сплайна в точке x(i);

z - массив размерности n, z(i) содержит значения 2-й производной сглаживающего сплайна в точке x(i);

sp - значение сплайна в точке xx ;

dsp - значение 1-й производной сплайна в точке xx ;

dsp2 - значение 2-й производной сплайна в точке xx.

Вспомогательные переменные:

a, b, c, d, e, g, zm, aa, dd - рабочие массивы размерности n ,

p, q, r, s, t, u, v, w - рабочие массивы размерности n + 2.

Программа Smspline обращается к подпрограмме решения системы линейных уравнений с 5-диагональной матрицей Progon5 (см. приложение А).

Чтобы показать, как пользоваться программой Smspline, рассмотрим следующий пример, имитирующий сглаживание экспериментальных данных.

Пусть в файле input.dat содержатся результаты 21 "измерения" функции $f(x) = \sin(x) + \xi$ в узлах сетки

$$\varpi = \{x_i = \frac{\pi}{20}(i-1), \quad i = 1, \ldots, 21\}$$

где погрешность измерения ξ представляет собой случайную величину, равномерно распределенную на отрезке [0.2, 0.2].

По результатам "измерений" построим сглаживающий сплайн, удовлетворяющий граничным условиям 1-го типа $S'(0) = 1$, $S'(\pi) = -1$ и протабулируем в 101-й точке с координатами

$$x_j = \frac{\pi}{100}(j-1), \quad j = 1, 2, \ldots, 101.$$

В качестве весовых коэффициентов возьмем величины

$$\rho_i = 0.2, \quad i = 1, 2, \ldots, 21.$$

Программа для решения этой задачи может иметь следующий вид:

```
      real a 100 ,b 100 ,c 100 ,d 100 ,e 100 ,g 100
      real w 100 ,u 100 ,v 100 ,s 100 ,zm 100
      real r 100 ,t 100 ,p 100 ,q 100 ,rho 100
      real aa 100 ,dd 100 ,x 100 ,y 100
      data pi /3.1415926/
C
            open 1,file = 'input_1.dat'
C
```

```
      read (1,*) n
        do  i = 1,n
            read(1,*) x(i), y(i), rho(i)
        enddo
      read (1,*) ax, bx
c
      ib = 1
        call  Smspline(n,ib,0,x,y,a,b,c,d,e,g,rho,ax,bx,
     *          u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
        h = pi/100.
c
      do i = 1,101
        xx = h*(i-1)
        call  Smspline(n,ib,1,x,y,a,b,c,d,e,g,rho,ax,bx,
     *          u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
        write (2,1) xx,sp,sin(xx),dsp,cos(xx),d2sp,-sin(xx)
      enddo
c
      stop
1     format (7 f8.4, 2x)
end
```

Результаты вычислений, проведенных по этой программе, приведены на рис. 1.15. На этом рисунке точками отмечены исходные данные и представлены: график сглаживающего сплайна, график интерполяционного сплайна, удовлетворяющего тем же граничным условиям, и график синусоиды.

На рисунке видно, что осцилляции, ярко выраженные на графике интерполяционного сплайна, практически полностью отсутствуют на графике сглаживающего сплайна. Графики первых производных обоих сплайнов представлены на рис. 1.16. Разница в поведении вторых производных еще более значительна.

Если требуется построить сглаживающий сплайн, удовлетворяющий граничным условиям 2-го типа, то следует воспользоваться программой, текст которой приведен ниже.

```
real a(100),b(100),c(100),d(100),e(100),g(100)
      real w(100),u(100),v(100),s(100),zm(100)
      real r(100),t(100),p(100),q(100),rho(100)
      real aa(100),dd(100),x(100),y(100)
      data pi /3.1415926/
c
      open (1,file = 'input_2.dat')
c
      read (1,*) n
        do  i = 1,n
            read(1,*) x(i), y(i), rho(i)
        enddo
c
      ib = 2
        call  Smspline(n,ib,0,x,y,a,b,c,d,e,g,rho,ax,bx,
     *          u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
        h = 2.*pi/100.
c
      do i = 1,101
        xx = h*(i-1)
```

```
          call  Smspline(n,ib,1,x,y,a,b,c,d,e,g,rho,ax,bx,
     *           u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
          write (2,1) xx,sp,sin(xx),dsp,cos(xx),d2sp,-sin(xx)
        enddo
c
        stop
1       format(7(f8.4,2x))
        end
```

Здесь подразумевается, что исходные данные для этого примера находятся в файле input_2.dat.

Следующая программа предназначена для построения сглаживающего сплайна, удовлетворяющего граничным условиям 3-го типа.

```
      real a(100),b(100),c(100),d(100),e(100),g(100)
          real w(100),u(100),v(100),s(100),zm(100)
          real r(100),t(100),p(100),q(100),rho(100)
          real aa(100),dd(100),x(100),y(100)
          data pi /3.1415926/
c
          open (1,file = 'input_3.dat')
c
          read (1,*) n
            do i = 1,n
               read(1,*) x(i), y(i), rho(i)
            enddo
c
          ib = 3
            call  Smspline(n,ib,0,x,y,a,b,c,d,e,g,rho,ax,bx,
     *           u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
            h = 2.*pi/100.
c
          do i = 1,101
            xx = h*(i-1)-1.e-5
            call  Smspline(n,ib,1,x,y,a,b,c,d,e,g,rho,ax,bx,
     *           u,v,w,p,q,r,s,t,aa,dd,zm,xx,sp,dsp,d2sp)
            write (2,1) xx,sp,sin(xx),dsp,cos(xx),d2sp,-sin(xx)
          enddo
c
          stop
1         format(7(f8.4,2x))
          end
```

Исходные данные для этого примера находятся в файле input_3.dat.

Текст программы Smspline находится в файле smspline.for в поддиректории SMOOTH на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". В эту же поддиректорию помещены файлы, содержащие примеры применения программы Smspline при других граничных условиях. Подробную информацию об именах этих файлов и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

1.3. Другие сплайны

1.3.1. Линейное пространство кубических сплайн-функций

Множество кубических сплайнов, построенных на отрезке $[a, b]$ по сетке ω с $m + 1$ узлом, является линейным пространством размерности $m + 3$:

1) сумма двух кубических сплайнов, построенных по сетке ω, и произведение кубического сплайна, построенного по сетке ω, на произвольное число также являются кубическими сплайнами, построенными по этой сетке;
2) любой кубический сплайн, построенный по сетке ω из $m + 1$ узла, полностью определяется $m + 1$ значением величин y_i в этих узлах и двумя граничными условиями - всего $m + 3$ параметрами.

Выбрав в этом пространстве базис, состоящий из $m + 3$ линейно независимых сплайнов $\sigma_i(x)$, $i = 1, ..., m + 3$, мы можем записать произвольный кубический сплайн $\sigma(x)$ в виде их линейной комбинации

$$\sigma(x) = \sum_{i=1}^{m+3} \alpha_i \sigma_i(x),$$

причем единственным образом.

Замечание

Подобное задание сплайна широко распространено в вычислительной практике. Особенно удобным является базис, состоящий из так называемых кубических B-сплайнов (базовых, или фундаментальных, сплайнов). Применение B-сплайнов позволяет существенно снизить требования к объему памяти компьютера.

1.3.2. Кубические *B*-сплайны

Определение *B*-сплайнов

B-сплайном нулевой степени, построенным на числовой прямой по сетке ω, называется функция вида

$$B_i^{(0)}(x) = \begin{cases} 1, & x \in [x_i, x_{i+1}], \\ 0, & x \notin [x_i, x_{i+1}]. \end{cases}$$

B-сплайн степени $k \geq 1$, построенный на числовой прямой по сетке ω, определяется посредством рекуррентной формулы

$$B_i^{(k)}(x) = \frac{x - x_i}{x_{i+1} - x_i} B_i^{(k-1)}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} B_{i+1}^{(k-1)}(x).$$

Графики B-сплайнов 1-й $B_i^{(1)}(x)$ - и 2-й - $B_i^{(2)}(x)$ - степеней представлены соответственно на рис. 1.17 и 1.18.

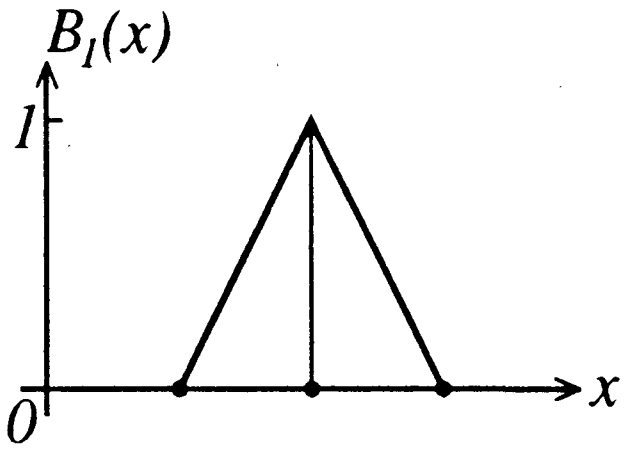


Рис. 1.17

B-сплайн произвольной степени k может быть отличен от нуля только на некотором отрезке (определяемом $k + 2$ узлами).

Кубические B-сплайны удобнее нумеровать так, чтобы сплайн $B_i^{(3)}(x)$ был отличен от нуля на отрезке $[x_{i-2}, x_{i+2}]$.

Приведем формулу для кубического сплайна 3-й степени для случая равномерной сетки (с шагом h). Имеем:

$$B_i^{(3)}(x) = \begin{cases} \frac{1}{6}\left(\frac{x-x_i}{h}+2\right)^3, & x \in [x_{i-2}, x_{i-1}], \\ \frac{2}{3} - \frac{1}{2}\left(\left(\frac{x-x_i}{h}\right)^3 + 2\left(\frac{x-x_i}{h}\right)^2\right), & x \in [x_{i-1}, x_i], \\ \frac{2}{3} + \frac{1}{2}\left(\left(\frac{x-x_i}{h}\right)^3 - 2\left(\frac{x-x_i}{h}\right)^2\right), & x \in [x_i, x_{i+1}], \\ \frac{1}{6}\left(2 - \frac{x-x_i}{h}\right)^3, & x \in [x_{i+1}, x_{i+2}], \\ 0 & \text{для друг их } x. \end{cases}$$

Функция $B_i^{(3)}(x)$

а) дважды непрерывно дифференцируема на отрезке $[a, b]$, то есть принадлежит классу $C^2[a,b]$;

б) отлична от нуля только на четырех последовательных отрезках $[x_{i-2}, x_{i-1}]$, $[x_{i-1}, x_i]$, $[x_i, x_{i+1}]$, $[x_{i+1}, x_{i+2}]$.

Отрезок $[x_{i-2}, x_{i+2}]$ называется *носителем функции* $B_i^{(3)}(x)$.

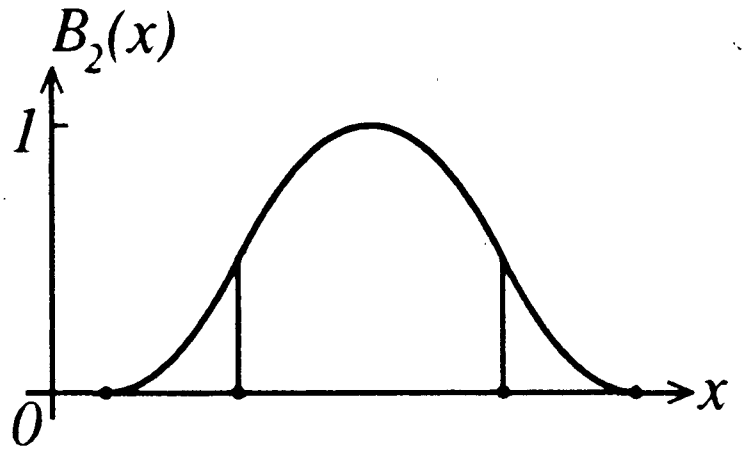


Рис. 1.18

Типичный график кубического B-сплайна представлен на рис. 1.19.

Дополним сетку ω вспомогательными узлами

$$x_{-3} < x_{-2} < x_{-1} < a$$
$$b < x_{m+1} < x_{m+2} < x_{m+3},$$

взятыми совершенно произвольно.

По расширенной сетке ω^*

$$x_{-3} < x_{-2} < x_{-1} < a < x_1 < \ldots < x_{m-1} < b < x_{m+1} < x_{m+2} < x_{m+3}$$

можно построить семейство из $m + 3$ кубических B-сплайнов:

$$B_i^{(3)}(x), \qquad i = -1, 0, \ldots, m, m+1 .$$

Это семейство образует базис в пространстве кубических сплайнов на отрезке $[a, b]$. Тем самым произвольный кубический сплайн $S(x)$, построенный на отрезке $[a, b]$ по сетке ω из $m+1$ узла, может быть представлен на этом отрезке в виде линейной комбинации

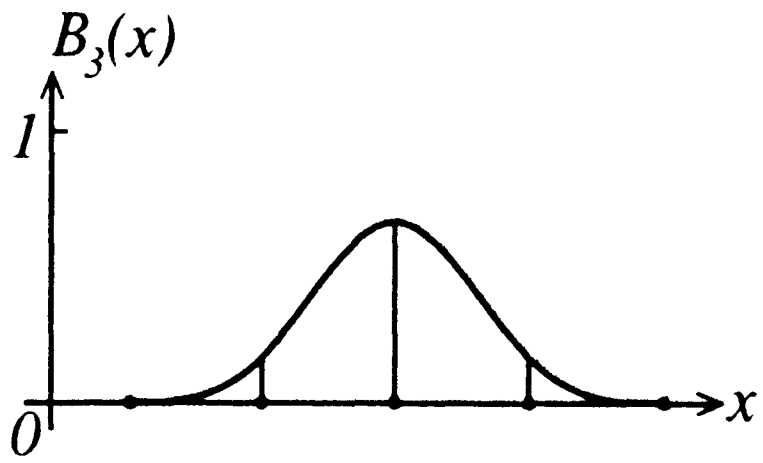


Рис. 1.19

$$S(x) = \sum_{i=-1}^{m+1} b_i B_i^{(3)}(x) .$$

Условиями задачи коэффициенты b_i этого разложения определяются однозначно.

В случае, когда заданы значения y_i функции в узлах сетки и значения y'_0 и y'_m 1-й производной функции на концах сетки (задача интерполяции с граничными условиями 1-го рода), эти коэффициенты вычисляются из системы следующего вида:

$$\begin{cases} b_{-1}B'_{-1}(x_0) + b_0 B'_0(x_0) + b_1 B'_1(x_0) = y'_0, \\ b_{i-1}B'_{i-1}(x_i) + b_i B'_i(x_i) + b_{i+1} B'_{i+1}(x_i) = y_i, \quad \text{где } i = 0, 1, \ldots, m. \\ b_{m-1}B'_{m-1}(x_m) + b_m B'_m(x_m) + b_{m+1} B'_{m+1}(x_m) = y'_m. \end{cases}$$

После исключения величин b_{-1} и b_{m+1} получается линейная система с неизвестными $b_0, \ldots, b_m$ и 3-диагональной матрицей. Условие

$$\rho = \max_{i-j=1} \frac{h_i}{h_j} < \frac{1+\sqrt{3}}{2}$$

обеспечивает диагональное преобладание и, значит, возможность применения метода прогонки для ее разрешения.

Замечания:

1. *Линейные системы аналогичного вида возникают при рассмотрении и других задач интерполяции.*

2. *В сравнении с алгоритмами, описанными в 1.1, применение B-сплайнов в задачах интерполяции позволяет уменьшить объем хранимой информации, то есть существенно снизить требования к объему памяти компьютера, хотя и приводит к увеличению числа операций.*

Глава 2. Сплайн-функции двух переменных

Пусть в прямоугольнике

$$R = [a,b] \times [c,d] = \{(x,y) \mid a \le x \le b,\ c \le y \le d\}$$

задана сетка

$$\omega = \omega_x \times \omega_y$$

где

$$\omega_x: \quad a = x_0 < x_1 < \ldots < x_{m-1} < x_m = b,$$

$$\omega_y: \quad c = y_0 < y_1 < \ldots < y_{n-1} < y_n = d$$

(рис. 2.1). Точки

(x_i, y_j), $i = 1, \ldots, m\text{-}1$, $j = 1, \ldots, n\text{-}1$,

называются *внутренними узлами сетки* ω; точки

(x_i, y_0), (x_i, y_n), $i = 1, \ldots, m - 1$,

(x_0, y_j), (x_m, y_j), $j = 1, \ldots, n - 1$

- *граничными узлами заданной сетки*, а точки

(x_0, y_0), (x_0, y_n), (x_m, y_0), (x_m, y_n)

- ее *угловыми узлами*.

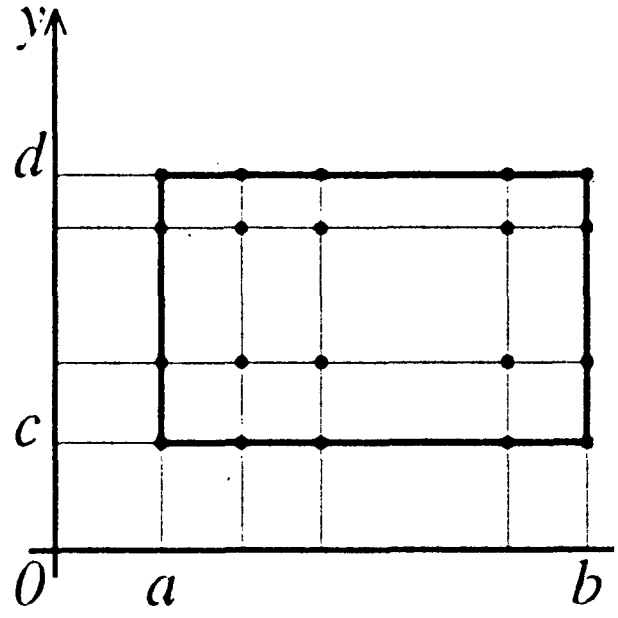


Рис. 2.1

Сетка ω делит прямоугольник R на ячейки

$$R_{ij} = \{(x,y) \mid x_i \le x \le x_{i+1},\ y_j \le y \le y_{j+1}\}$$

$i = 0, 1, \ldots, m - 1$, $j = 0, 1, \ldots, n - 1$ (рис. 2.2).

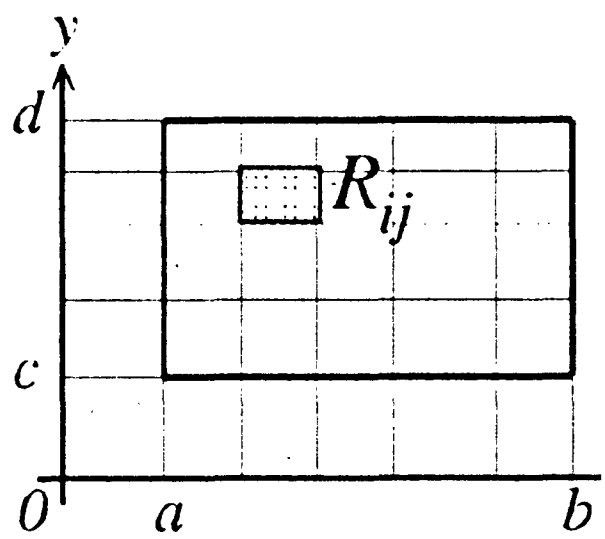


Рис. 2.2

Функция $S(x, y)$, заданная на прямоугольнике R, называется бикубическим (дваждыкубическим) сплайном или бикубической сплайн-функцией, если:

1) в каждой прямоугольной ячейке R_{ij} $S(x, y)$ функция являе многочленом 3-й степени как по x, так и по y, то есть

$$S(x,y)=\sum_{k=0}^{3}\sum_{l=0}^{3}a_{kl}^{(i,j)}(x-x_i)^k(y-y_j)^l,$$

$$i=0,\ 1,\ ...,\ m-1,\quad j=0,\ 1,\ ...,\ n-1;$$

2) сама функция $S(x, y)$ и ее производные

$$\frac{\partial S}{\partial x},\ \frac{\partial S}{\partial y},\ \frac{\partial^2 S}{\partial x^2},\ \frac{\partial^2 S}{\partial x\partial y},\ \frac{\partial^2 S}{\partial y^2},\ \frac{\partial^3 S}{\partial x^2\partial y},\ \frac{\partial^3 S}{\partial x\partial y^2},\ \frac{\partial^4 S}{\partial x^2\partial y^2}$$

непрерывны в прямоугольнике R, то есть $S(x,y)\in C^{2,2}(R)$.

Замечания:

1. *Индексы i и j у коэффициентов* $a_{kl}^{(i,j)}$ *указывают на то, что набор величин, которым определяется сплайн-функция S(x, y), в каждой ячейке* R_{ij} *свой.*

2. *В каждой прямоугольной ячейке* R_{ij} *бикубический сплайн S(x, y) можно записать в виде*

$$S(x,y)=\sum_{l=0}^{3}\left(\sum_{k=0}^{3}a_{kl}^{(i,j)}(x-x_i)^k\right)(y-y_j)^l$$

или в виде

$$S(x,y)=\sum_{k=0}^{3}\left(\sum_{l=0}^{3}a_{kl}^{(i,j)}(y-y_j)^l\right)(x-x_i)^k.$$

Первая запись позволяет говорить, что бикубический сплайн $S(x, y)$ в каждой ячейке R_{ij} представляет собой кубический многочлен по переменной $y - y_j$, коэффициенты которого суть кубические многочлены по переменной x-x_i. Аналогичный смысл имеет и вторая запись.

Это обстоятельство дает основания считать, что алгоритм построения двумерных кубических сплайнов в значительной степени опирается на алгоритм построения одномерных сплайнов.

В каждой прямоугольной ячейке R_{ij} сплайн-функция $S(x, y)$ определяется 16 коэффициентами:

$$\begin{array}{cccc} a_{00}^{(i,j)} & a_{10}^{(i,j)} & a_{20}^{(i,j)} & a_{30}^{(i,j)} \\ a_{01}^{(i,j)} & a_{11}^{(i,j)} & a_{21}^{(i,j)} & a_{31}^{(i,j)} \\ a_{02}^{(i,j)} & a_{12}^{(i,j)} & a_{22}^{(i,j)} & a_{32}^{(i,j)} \\ a_{03}^{(i,j)} & a_{13}^{(i,j)} & a_{23}^{(i,j)} & a_{33}^{(i,j)} \end{array}.$$

Всего прямоугольных ячеек - mn. Значит, для того, чтобы полностью описать бикубический сплайн на сетке ω, необходимо найти $16mn$ чисел

$$a_{pq}^{(i,j)}, \quad p=0,\ 1,\ 2,\ 3, \quad q=0,\ 1,\ 2,\ 3,$$

$$i=0,\ 1,\ \ldots,\ m\text{-}1, \quad j=0,\ 1,\ \ldots,\ n\text{-}1$$

Наиболее часто рассматриваются задачи интерполяции и сглаживания, когда требуется построить тот или иной сплайн по заданному массиву точек

$$\left(x_{ij}, y_{ij}, z_{ij}\right), \quad i=0,1,\ \ldots,\ m,$$

$$j=0,\ 1,\ldots,\ n$$

(рис. 2.3).

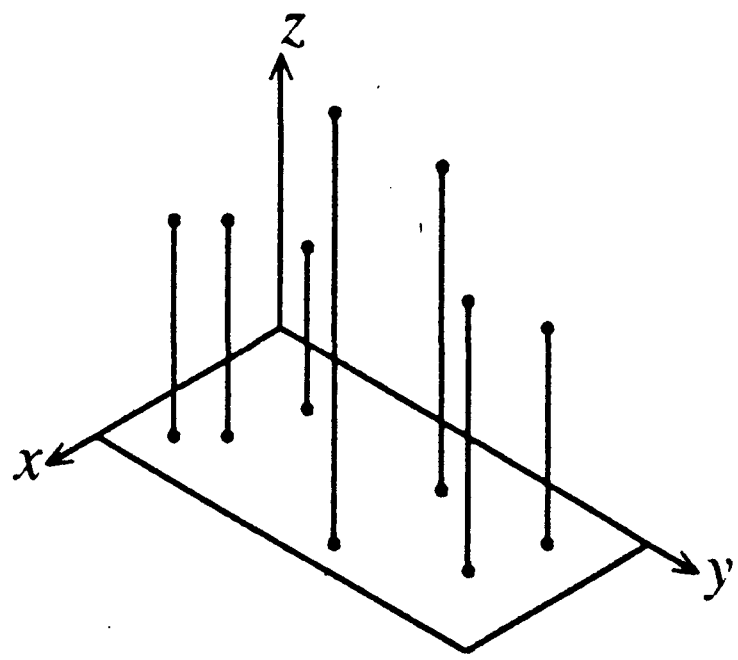


Рис. 2.3

В задачах *интерполяции* требуется, чтобы график сплайна проходил через точки $\left(x_{ij}, y_{ij}, z_{ij}\right)$, $i=0,\ 1,\ \ldots,\ m,\ \ j=0,\ 1,\ \ldots,\ n$. Это требование накладывает на его коэффициенты ряд дополнительных условий (уравнений), которых, однако, недостаточно для однозначного построения сплайна. Необходимый для этого набор условий (уравнений) чаще всего задают в виде значений младших производных сплайна в граничных узлах сетки ω - *граничных (краевых) условий.* Возможность выбора различных граничных условий позволяет строить сплайны, обладающие самыми разными свойствами.

В задачах *сглаживания* сплайн строят так, чтобы его график проходил вблизи точек $\left(x_{ij}, y_{ij}, z_{ij}\right)$, $i=0,\ 1,\ \ldots,\ m$, $j=0,\ 1,\ \ldots,\ n$, а не через них. Меру этой близости можно определять по-разному, что приводит к значительному разнообразию сглаживающих сплайнов.

В этой главе мы останавливаемся на сплайн-функциях двух переменных, которые представляют собой прямое обобщение соответствующих одномерных сплайнов. Однако рассматриваемые двумерные сплайны далеко не исчерпывают всего их многообразия, которое определяется и выбором граничных (краевых) условий, и исходным массивом точек, и классом функций, используемых для построения элементарных фрагментов, и т. д.

2.1. Интерполяционные бикубические сплайны

2.1.1. Постановка задачи интерполяции

Пусть заданы сетка ω

$$a = x_0 < x_1 < \ldots < x_{m-1} < x_m = b, \quad c = y_0 < y_1 < \ldots < y_{n-1} < y_n = d$$

и набор чисел

$$z_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n.$$

Задача. Построить на прямоугольнике $R = [a,b] \times [c,d]$ гладкую функцию $\sigma(x,y)$, которая принимает в узлах сетки ω заданные значения

$$\sigma(x_i, y_j) = z_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n.$$

Замечание

Сформулированная задача интерполяции состоит в восстановлении функции, заданной таблично. Ясно, что такая задача имеет множество различных решений. Накладывая на конструируемую функцию дополнительные условия, можно добиться необходимой однозначности.

2.1.2. Определение интерполяционного бикубического сплайна

Интерполяционным бикубическим сплайном $S(x, y)$ на сетке ω называется функция, которая:

1) в каждой ячейке

$$R_{ij} = \left\{(x,y) \,\middle|\, x_i \le x \le x_{i+1},\ y_j \le y \le y_{j+1}\right\}$$

$$i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n,$$

представляет собой многочлен вида

$$S(x,y) = \sum_{p=0}^{3} \sum_{q=0}^{3} a_{p,q}^{(i,j)} (x - x_i)^p (y - y_j)^q;$$

2) принадлежит классу $C^{(2,2)}(R)$;

3) удовлетворяет условиям

$$S(x_i, y_j) = z_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n.$$

Для построения сплайна необходимо определить $16mn$ коэффициентов $a_{p,q}^{(i,j)}$. Соотношения (*) дают $(m+1)(n+1)$ условий. Вместе с требованием $S(x,y) \in C^{(2,2)}(R)$ получается $16mn$-$2(m+n)$-8 условий (уравнений) на коэффициенты $a_{p,q}^{(i,j)}$.

2.1.3. Граничные (краевые) условия

Недостающие $2(m+n+4)$ условия задаются в виде ограничений на значения производных сплайна в граничных и угловых узлах сетки ω. При построении интерполяционного бикубического сплайна наиболее часто используются граничные условия следующих четырех типов.

Граничные (краевые) условия 1-го типа

$$\frac{\partial S}{\partial x}(x_i, y_j) = z_{ij}^{(x)}, \qquad i = 0,\ m, \qquad j = 0,\ 1,\ \ldots,\ n,$$

$$\frac{\partial S}{\partial y}(x_i, y_j) = z_{ij}^{(y)}, \qquad i = 0,\ 1,\ \ldots,\ m, \qquad j = 0,\ n,$$

$$\frac{\partial^2 S}{\partial x \partial y}(x_i, y_j) = z_{ij}^{(xy)}, \qquad i = 0,\ m, \qquad j = 0,\ n,$$

- в граничных узлах сетки ω задаются значения первых частных производных по x и по y искомой функции, а в угловых узлах - значения 2-ой смешанной производной.

Число граничных (краевых) условий 1-го типа равно $2m + 2n + 8$.

Наглядное представление о способе задания сплайна, отвечающего граничным условиям этого типа, дает рис. 2.4. На нем жирными точками отмечены узлы сетки, в которых задаются значения сплайна S, горизонтальными и вертикальными стрелками указаны узлы, в которых задаются значения первых частных производных S_x и S_y соответственно, а ромбиком - значения смешанной производной S_{xy}.

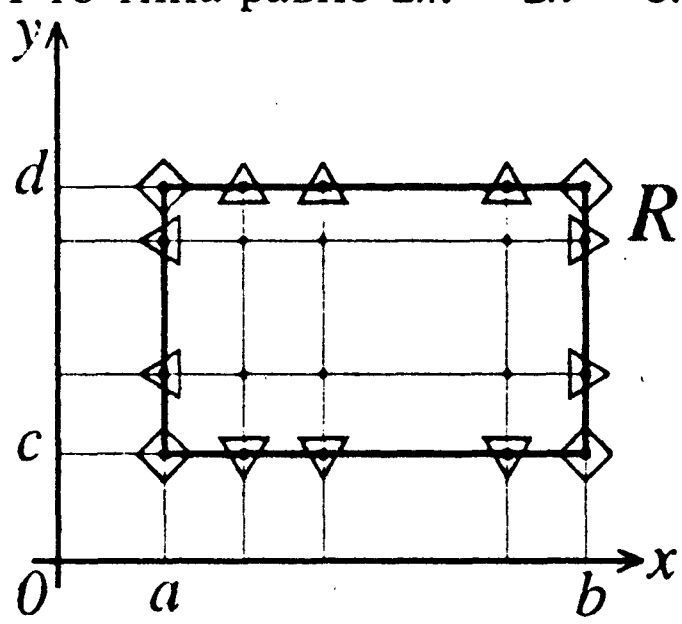


Рис. 2.4

Граничные (краевые) условия 2-го типа

$$\frac{\partial^2 S}{\partial x^2}(x_i, y_j) = z_{ij}^{(x)}, \qquad i = 0,\ m, \qquad j = 0,\ 1,\ \ldots,\ n,$$

$$\frac{\partial^2 S}{\partial^2 y}(x_i, y_j) = z_{ij}^{(y)}, \qquad i = 0, 1, ..., m, \quad j = 0, n,$$

$$\frac{\partial^4 S}{\partial x^2 \partial y^2}(x_i, y_j) = z_{ij}^{(xy)}, \quad i = 0, m, \qquad j = 0, n,$$

- в граничных узлах сетки ω задаются значения вторых частных производных по x и по y искомой функции, а в угловых узлах – значения смешанной производной S_{xxyy}.

Число граничных (краевых) условий 2-го типа равно $2m + 2n + 8$.

Наглядное представление о способе задания сплайна, отвечающего граничным условиям этого, типа также дает рис. 2.4. На нем жирными точками отмечены узлы сетки, в которых задаются значения сплайна S, горизонтальными и вертикальными стрелками указаны узлы, в которых задаются значения вторых частных производных S_{xx} и S_{yy} соответственно, а ромбиком - значения смешанной производной S_{xxyy}.

Граничные (краевые) условия 3-го типа

$$S(x_0, y_j) = S(x_m, y_j), \qquad j = 0, 1, ..., n$$

$$S(x_i, y_0) = S(x_i, y_n), \qquad i = 0, 1, ..., m,$$

$$\frac{\partial^k S}{\partial x^k}(x_0, y_j) = \frac{\partial^k S}{\partial x^k}(x_m, y_j), \qquad j = 0, 1, ..., n, \qquad k = 1, 2,$$

$$\frac{\partial^l S}{\partial y^l}(x_i, y_0) = \frac{\partial^l S}{\partial y^l}(x_i, y_n), \qquad i = 0, 1, ..., m, \qquad l = 1, 2,$$

$$\frac{\partial^{2k} S}{\partial x^k \partial y^k}(x_0, y_j) = \frac{\partial^{2k} S}{\partial x^k \partial y^k}(x_m, y_j), \quad j = 0, 1, ..., n, \qquad k = 1, 2,$$

$$\frac{\partial^{2l} S}{\partial x^l \partial y^l}(x_i, y_0) = \frac{\partial^{2l} S}{\partial x^l \partial y^l}(x_i, y_n), \qquad i = 0, 1, ..., m, \qquad l = 1, 2,$$

называются *периодическими*. В этом случае сплайн $S(x, y)$ и его частные производные должны быть двоякопериодическими функциями - с периодом $T_x = b - a$ по переменной x и с периодом $T_y = c - d$ по переменной y.

Условия на сплайн-функцию $S(x, y)$ входят в группу условий (*), а общее число условий на ее производные равно $2(2m + 2n + 4)$.

Граничные (краевые) условия 4-го типа

требуют, чтобы на линиях $x = x_1$ и $x = x_{m-1}$ были непрерывны следующие производные искомого сплайна:

$$\frac{\partial S}{\partial x},\ \frac{\partial S}{\partial y},\ \frac{\partial^2 S}{\partial x^2},\ \frac{\partial^2 S}{\partial x \partial y},\ \frac{\partial^2 S}{\partial y^2},$$

$$\frac{\partial^3 S}{\partial x^3},\ \frac{\partial^3 S}{\partial x^2 \partial y},\ \frac{\partial^3 S}{\partial x \partial y^2},\ \frac{\partial^4 S}{\partial x^3 \partial y},\ \frac{\partial^4 S}{\partial x^2 \partial y^2},\ \frac{\partial^5 S}{\partial x^3 \partial y^2},$$

а на линиях $y = y_1$ и $y = y_{n-1}$ - производные

$$\frac{\partial S}{\partial x},\ \frac{\partial S}{\partial y},\ \frac{\partial^2 S}{\partial x^2},\ \frac{\partial^2 S}{\partial x \partial y},\ \frac{\partial^2 S}{\partial y^2},$$

$$\frac{\partial^3 S}{\partial y^3},\ \frac{\partial^3 S}{\partial x^2 \partial y},\ \frac{\partial^3 S}{\partial x \partial y^2},\ \frac{\partial^4 S}{\partial x^2 \partial y^2},\ \frac{\partial^4 S}{\partial x \partial y^3},\ \frac{\partial^5 S}{\partial x^2 \partial y^3},$$

Комментарий. Сплайн, удовлетворяющий граничным (краевым) условиям 4-го типа, обладает повышенной гладкостью: в прямоугольниках

$$[x_k, x_{k+2}] \times [y_l, y_{l+2}] \qquad k = 0,\ m-2,\quad l = 0,\ n-2,$$

прилегающих к вершинам прямоугольника R, непрерывны все его производные

$$\frac{\partial^{\,p} S}{\partial x^q \partial y^r},\quad 1 \le p \le 6,\quad 0 \le q \le 3,\quad 0 \le r \le 3.$$

Замечание

Кроме перечисленных, возможны и смешанные граничные условия, то есть условия, относящиеся по разным переменным к разным типам. При этом если, например, по переменной x заданы условия 1-го типа, а по переменной y – 2-го типа, то в вершинах прямоугольника R следует задавать частные производные S_{xyy} и т. д.

2.1.4. Построение интерполяционного бикубического сплайна

Для описания алгоритма исходную информацию удобно расположить в виде табл. 2.1.

Таблица 2.1

$z_{0,n}^{(x,y)}$	$z_{0,n}^{(y)}$	$z_{1,n}^{(y)}$	...	$z_{m-1,n}^{(y)}$	$z_{m,n}^{(y)}$	$z_{m,n}^{(x,y)}$
$z_{0,n}^{(x)}$	$z_{0,n}$	$z_{1,n}$	...	$z_{m-1,n}$	$z_{m,n}$	$z_{m,n}^{(x)}$
$z_{0,n-1}^{(x)}$	$z_{0,n-1}$	$z_{1,n-1}$	...	$z_{m-1,n-1}$	$z_{m,n-1}$	$z_{m,n-1}^{(x)}$
...	...	...	...	...	...	...
$z_{0,1}^{(x)}$	$z_{0,1}$	$z_{1,1}$	...	$z_{m-1,1}$	$z_{m,1}$	$z_{m,1}^{(x)}$
$z_{0,0}^{(x)}$	$z_{0,0}$	$z_{1,0}$	...	$z_{m-1,0}$	$z_{m,0}$	$z_{m,0}^{(x)}$
$z_{0,0}^{(x,y)}$	$z_{0,0}^{(y)}$	$z_{1,0}^{(y)}$	...	$z_{m-1,0}^{(y)}$	$z_{m,0}^{(y)}$	$z_{m,0}^{(x,y)}$

Во внутренненей части таблицы располагаются значения z_{ij} сплайна в узлах сетки ω. Окаймляющие строки и столбцы заполняются численными значениями соответствующих производных (только в случае граничных условий 1-го или 2-го типа).

Алгоритм построения интерполяционного бикубического сплайна основан на том, что при фиксированном значении одной из переменных (например, y) сам сплайн и его частная производная по y являются интерполяционными кубическими сплайнами по переменной x.

1-й шаг алгоритма

По каждой строке таблицы, включая граничные (если они имеются), строятся одномерные кубические интерполяционные сплайны по переменной x с краевыми условиями, взятыми из граничных столбцов (если они имеются). Построение каждого одномерного сплайна сводится к отысканию чисел

$$\mu_{ij}^{(x)}, \quad i = 0, 1, ..., m, \quad j = 0, 1, ..., n,$$

которые совпадают со значениями производной S_x сплайна S в узлах сетки ω. Эти числа удобно расположить без окаймляющих столбцов в виде табл. 2.2.

Таблица 2

$\mu_{0,n}^{(xy)}$	$\mu_{1,n}^{(xy)}$	...	$\mu_{m,n}^{(xy)}$
$\mu_{0,n}^{(x)}$	$\mu_{1,n}^{(x)}$	...	$\mu_{m,n}^{(x)}$
...	...	...	...
$\mu_{0,0}^{(x)}$	$\mu_{1,0}^{(x)}$	...	$\mu_{m,0}^{(x)}$
$\mu_{0,0}^{(xy)}$	$\mu_{1,0}^{(xy)}$	...	$\mu_{m,0}^{(xy)}$

При граничных условиях 1-го и 2-го типа в верхней и нижней строках таблицы размещаются значения в граничных узлах сетки ω производных S_{xy} и S_{xyy} соответственно. При граничных условиях 3-го или 4-го типа по переменной y этих окаймляющих строк не будет.

2-й шаг алгоритма

По каждому столбцу табл. 2.2 строятся интерполяционные кубические сплайны $\widetilde{S}$, являющиеся частными производными по переменной x искомого сплайна S на линиях $x = x_i$:

$$\widetilde{S}(x,y) = \frac{\partial S}{\partial x}(x_i, y) \ , \qquad i = 0,\ 1,\ ...,\ m.$$

Построение каждого из сплайнов $S(x_i, y)$ сводится к определению чисел $\mu_{ij}^{(xy)}$, которые являются производными по переменной y вспомогательных сплайнов $\widetilde{S}$ и, значит, производными S_{xy} искомого сплайна в узлах сетки ω.

3-й шаг алгоритма

аналогичен 1-му шагу: по данным табл. 2.1 строятся одномерные интерполяционные сплайны $S(x_i, y)$, $i = 0,\ 1,\ ...,\ m$, по переменной y, удовлетворяющие граничным условиям соответствующего типа. В результате будут найдены значения

$$\mu_{ij}^{(y)} = \frac{\partial S}{\partial y}(x_i, y_j) \ , \qquad i = 0,1,\ ...,\ m \ , \quad j = 0,1,\ ...,\ n.$$

После выполнения 3-го шага алгоритма в каждом узле сетки ω будут известны величины

$$z_{ij} \ , \quad \mu_{ij}^{(x)} \ , \quad \mu_{ij}^{(y)} \ , \quad \mu_{ij}^{(xy)} \ ,$$

которые полностью определяют интерполяционный бикубический сплайн, удовлетворяющий заданным граничным условиям.

Алгоритм вычисления значений сплайна

Для того чтобы вычислить значение интерполяционного бикубического сплайна в произвольной точке (x, y) прямоугольника R, определим следующие параметры:

номер i такой, что $x \in [x_i, x_{i+1}]$,

номер j такой, что $y \in [y_j, y_{j+1}]$,

шаги $h_i = x_{i+1} - x_i$, $l_j = y_{j+1} - y_j$ сетки ω по x и по y соответственно,

числа $t = \dfrac{x - x_i}{h_i}$ и $u = \dfrac{y - y_j}{l_j}$.

Введем 4 вспомогательные функции:

$$\varphi_1(\xi) = (1-\xi)^2(1+2\xi), \qquad \varphi_2(\xi) = \xi^2(3-2\xi),$$

$$\varphi_3(\xi) = \xi(1-\xi)^2, \qquad \varphi_4(\xi) = -\xi^2(1-\xi)$$

- и два вектора - $\mathbf{a}_t$ и $\mathbf{b}_u$:

$$\mathbf{a}_t = \begin{bmatrix} \varphi_1(t) & \varphi_2(t) & h_i\varphi_3(t) & h_i\varphi_4(t) \end{bmatrix},$$

$$\mathbf{b}_u = \begin{bmatrix} \varphi_1(u) & \varphi_2(u) & l_j\varphi_3(u) & l_j\varphi_4(u) \end{bmatrix}.$$

Тогда искомый сплайн $S(x, y)$ можно вычислить посредством формулы

$$S(x,y) = \mathbf{a}_t \mathbf{Z} \mathbf{b}_u^T,$$

где матрица $\mathbf{Z}$ имеет следующий вид:

$$\mathbf{Z} = \begin{pmatrix} z_{ij} & z_{i+1,j} & \mu_{ij}^{(x)} & \mu_{i+1,j}^{(x)} \\ z_{i,j+1} & z_{i+1,j+1} & \mu_{i,j+1}^{(x)} & \mu_{i+1,j+1}^{(x)} \\ \mu_{ij}^{(y)} & \mu_{i+1,j}^{(y)} & \mu_{ij}^{(xy)} & \mu_{i+1,j}^{(xy)} \\ \mu_{i,j+1}^{(y)} & \mu_{i+1,j+1}^{(y)} & \mu_{i,j+1}^{(xy)} & \mu_{i+1,j+1}^{(xy)} \end{pmatrix}$$

2.1.5. Свойства интерполяционного бикубического сплайна

А. Аппроксимационное свойство

В приложениях часто требуется приблизить сплайном функцию, заданную аналитически,

$$z = f(x, y), \quad (x, y) \in R.$$

Например, такая задача возникает в случае, когда вычисление значений заданной функции в точках прямоугольника R связано со значительными трудностями.

Задача интерполяции функции. Построить в прямоугольнике R бикубический сплайн $S(x, y)$, совпадающий в узлах сетки ω с заданной функцией $f(x, y)$,

$$S(x_i, y_j) = f(x_i, y_j), \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n.$$

Аппроксимационные свойства построенного интерполяционного бикубического сплайна $S(x, y)$ зависят от гладкости интерполируемой функции $f(x, y)$.

Например, если функция $f(x, y)$ принадлежит классу $C^4(R)$, то

$$\left| f(x,y) - S(x,y) \right| = O\left(h^{3/2} + l^{3/2}\right),$$

где

$$h = \max_{i=0,\ldots,m} h_i, \quad l = \max_{j=0,\ldots,n} l_j$$

(здесь $h_i = x_{i+1} - x_i$ и $l_j = y_{j+1} - y_j$ - шаги сетки ω по x и по y соответственно).

Если же функция $f(x, y)$ принадлежит классу $C^8(R)$, то порядок аппроксимации гораздо выше:

$$\left| f(x,y) - S(x,y) \right| = O\left(h^3 + l^3\right).$$

Замечание

Приведенные формулы оценивают порядок погрешности аппроксимации заданной функции бикубическим сплайном при $h \to 0$ и $l \to 0$ и справедливы при любых соотношениях между h и l. Если же потребовать дополнительно, чтобы при $h \to 0$ и $l \to 0$ отношение h/l оставалось ограниченным, то порядок аппроксимации возрастет на единицу,

$$\left| f(x,y) - S(x,y) \right| = O\left(h^4 + l^4\right).$$

Б. Экстремальное свойство

По аналогии с одномерным случаем рассмотрим следующую вариационную задачу.

Пусть $R = [a,b] \times [c,d]$ - заданный прямоугольник.

Задача. Среди всех функций, принадлежащих классу $C^{2,2}(R)$ и принимающих в узлах сетки ω

$$a = x_0 < x_1 < \ldots < x_{m-1} < x_m = b, \quad c = y_0 < y_1 < \ldots < y_{n-1} < y_n = d$$

заданные значения

$$z_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n,$$

найти такую, которая доставляет минимум функционалу

$$J(f) = \iint_R \left(\frac{\partial^4 f}{\partial x^2 \partial y^2} \right)^2 dxdy .$$

Для того чтобы сформулированная задача имела единственное решение, необходимы дополнительные условия. Укажем некоторые варианты таких условий.

1. Если потребовать, чтобы допустимые функции удовлетворяли условиям

$$\frac{\partial^2 f}{\partial x^2}(x_i, y_j) = 0, \qquad i = 0, m, \qquad j = 0, 1, \ldots, n,$$

$$\frac{\partial^2 f}{\partial y^2}(x_i, y_j) = 0, \qquad i = 0, 1, \ldots, m, \qquad j = 0, n,$$

$$\frac{\partial^4 f}{\partial x^2 \partial y^2}(x_i, y_j) = 0, \qquad i = 0, m, \qquad j = 0, n,$$

то решение вариационной задачи будет однозначно определено и этим решением будет интерполяционный бикубический сплайн.

2. Если потребовать, чтобы допустимые функции удовлетворяли граничным условиям вида

$$\frac{\partial^2 f}{\partial x^2}(x_i, y_j) = z_{ij}^{(1,0)}, \qquad i = 0, m, \qquad j = 0, 1, \ldots, n,$$

$$\frac{\partial^2 f}{\partial y^2}(x_i, y_j) = z_{ij}^{(0,1)}, \qquad i = 0, 1, \ldots, m, \qquad j = 0, n,$$

$$\frac{\partial^2 f}{\partial x^2 \partial y^2}(x_i, y_j) = z_{ij}^{(1,1)}, \qquad i = 0, m, \qquad j = 0, n,$$

то решение вариационной задачи будет однозначно определено и этим решением будет интерполяционный бикубический сплайн.

3. Если потребовать, чтобы допустимые функции были двоякопериодическими функциями с периодами $T_x = b - a$ и $T_y = d - c$, то решение вариационной задачи будет однозначно определено и этим решением будет двоякопериодический интерполяционный бикубический сплайн.

Замечание

Перечисленные выше сплайны являются решением сформулированной вариационной задачи в гораздо более широком классе функций, а именно в классе $W_2^{2,2}(R)$.

2.1.6. Построение сплайновых поверхностей при помощи сплайн-функций

Выше рассматривались массивы, точки которых были занумерованы так, что и их абсциссы, и их ординаты образовывали строго возрастающие последовательности. Это обстоятельство определяло и выбор класса аппроксимирующих поверхностей (графики функций), и способ их построения.

Однако предложенный метод позволяет достаточно успешно строить интерполяционную поверхность и в более общем случае, когда нумерация точек массива

$$\mathbf{P} = \left\{\mathbf{P}_{ij}\left(x_{ij}, y_{ij}, z_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n\right\}$$

и их расположение в пространстве, как правило, не связаны.

Ясно, что для решения этой общей задачи необходимо существенно расширить класс допустимых поверхностей, включив в него и поверхности, которые нельзя однозначно спроектировать ни на одну из координатных плоскостей, и поверхности с самопересечениями и самоналеганиями. Такие поверхности удобно описывать при помощи параметрических уравнений

$$x = x(u,v),\quad y = y(u,v),\quad z = z(u,v),$$

$$(u,v) \in R = [\alpha,\beta] \times [\gamma,\delta].$$

Потребуем дополнительно, чтобы функции $x(u, v)$, $y(u, v)$ и $z(u, v)$ обладали достаточной гладкостью, например принадлежали классу $C^1(R)$ или классу $C^2(R)$.

Для отыскания параметрических уравнений поверхности, последовательно проходящей через все точки массива, поступим следующим образом.

1-й шаг. На произвольно взятом прямоугольнике R изменения параметров u и v вводится вспомогательная сетка

$$\alpha = u_0 < u_1 < \ \ldots\ < u_{m-1} < u_m = \beta,\quad \gamma = v_0 < v_1 < \ \ldots\ < v_{n-1} < v_n = \delta,$$

число узлов которой совпадает с числом точек в массиве **P**.

2-й шаг. По заданному массиву **P** строятся 3 новых вспомогательных массива:

$$\mathbf{X} = \left\{\left(u_i, v_j, x_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n\right\},$$

$$\mathbf{Y} = \left\{\left(u_i, v_j, y_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n\right\},$$

$$\mathbf{Z} = \left\{\left(u_i, v_j, z_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n\right\}.$$

3-й шаг. Для каждого из массивов **X**, **Y** и **Z** находятся соответствующие интерполяционные сплайн-функции $x(u, v)$, $y(u, v)$ и $z(u, v)$.

В результате мы получаем параметрические уравнения поверхности, проходящей через массив точек **P**.

Замечание

Полученная поверхность будет гладкой, но не обязательно регулярной, так как возможность

$$\operatorname{rank}\begin{pmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \end{pmatrix} < 2$$

для некоторой точки $(u^*, v^*) \in (\alpha, \beta) \times (\gamma, \delta)$ *исключать нельзя.*

Кроме того, эта поверхность может иметь линии самопересечения и участки самоналегания.

2.1.7. Программная реализация

Описанный выше алгоритм построения бикубического интерполяционного сплайна и вычисления его значений в произвольной точке реализованы в виде подпрограммы Splint2, написанной на языке Фортран:

```
      subroutine Splint2
     *  (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,ind,ibx,
     *   iby,xx,yy,a,b,c,d,u,v, w,t,s,zz,sd,sd,spl
c
c**********************************************************************
c    Программа Splint2 строит интерполяционный                        *
c              бикубический сплайн для таблично заданной              *
c              функции двух переменных                                 *
c                                                                      *
c m   - число узлов сетки по оси x                                    *
c x   - массив длины m, содержит узлы сетки по оси x                  *
c n   - число узлов сетки по оси y                                    *
c y   - массив длины n, содержит узлы сетки по оси y                  *
c z   - одномерный массив длины n*m, содержит значения                *
c       интерполируемой функции в узлах интерполяции;                 *
c       значение функции в узле сетки (i,j) содержится                 *
c       в элементе z(i+(j-1)*m)                                        *
c ibx - код типа граничных условий на прямых x=x1                      *
c       и x=xm                                                         *
c                 ibx = i (i=1-4) - условия i-го типа                  *
c zl  - массив длины n, элемент zl(j) содержит значение               *
c       i-й частной производной по x в узлах (1,j)                     *
c       для граничных условий i-го типа (i=1,2)                        *
c zr  - массив длины n, элемент zr(j) содержит значение               *
c       i-й частной производной по x в узлах (m,j)                     *
c       для граничных условий i-го типа (i=1,2)                        *
c iby - код типа граничных условий на прямых y=y1                      *
c       и y=yn                                                         *
c                 iby = i (i=1-4) - условия i-го типа                  *
c zu  - массив длины m, элемент zu(j) содержит значение               *
c       i-й частной производной по y в узлах (j,m)                     *
```

```
c         для граничных условий i-го типа (i=1,2)                *
c zd -    массив длины m, элемент zd(j) содержит значение        *
c         i-й частной производной по y в узлах (j,1)             *
c         для граничных условий i-го типа (i=1,2)                *
c zxy-    массив длины 4, содержит значения смешанных            *
c         производных в угловых точках сетки в случае гра-      *
c         ничных условий 1-го или 2-го типа                      *
c a,b,c,d   - рабочие массивы длины max(n,m)                     *
c s,t,u,v,w,zz,su,sd - рабочие массивы длины  max(n,m)+1         *
c zx,zy,zxy - одномерные массивы длины n*m, содержат             *
c               значения параметров сплайна                      *
c ind - код режима работы:                                       *
c         ind=0 - программа вычисляет параметры сплайна         *
c         ind=1 - параметры сплайна известны                     *
c                                                                *
c xx,yy - координаты точки, в которой вычисляются парамет-       *
c         ры сплайна                                             *
c                                                                *
c         Результат:                                             *
c  spl    - значение сплайна в точке с координатами (xx,yy)      *
c*****************************************************************
c
      dimension x(1) ,y(1) ,z(1) ,zl(1) ,zr(1), zu(1),
     *          zd(1),zx(1),zy(1),zxy(1),u(1) , v(1) ,
     *          w(1) ,t(1) ,s(1) ,su(1) ,sd(1), a(1) ,
     *          b(1) ,c(1) ,d(1) ,zz(1) ,f(4) , g(4) ,
     *          sum(4)
      if (ind.eq.0) then
      do j = 1,n
      call Spline (m,x,z(1+j-1)*m),a,b,c,d,zx(1+(j-1)*m),
     *       u,v,w,t,s,ind,ibx,zl(j),zr(j),xx,sp,dsp,d2sp)
      enddo
c
      if (iby.le.2) then
         call Spline (m,x,zd,a,b,c,d,sd,u,v,w,
     *             t,s,0,ibx,zxy(1),zxy(2),xx,sp,dsp,d2sp)
         call Spline (m,x,zu,a,b,c,d,su,u,v,w,
     *             t,s,0,ibx,zxy(3),zxy(4),xx,sp,dsp,d2sp)
      endif
c
      do i = 1,m
         do j = 1,n
            zz(j) = zx(i + (j-1)*m)
         enddo
       call Spline (n,y,zz,a,b,c,d,zxy(1+(i-1)*n),u,v,w,
     *             t,s,0,iby,sd(i),su(i),xx,sp,dsp,d2sp)
      enddo
c
      do i = 1,m
         do j = 1,n
            zz(j) = z(i + (j-1)*m)
         enddo
       call Spline (n,y,zz,a,b,c,d,zy(1+(i-1)*n),u,v,w,
     *             t,s,0,iby,zd(i),zu(i),xx,sp,dsp,d2sp)
      enddo
c
      endif
c
      do  i = 2,m
         ni = i
         if(x(i).gt.xx) go to 1
```

```
            enddo
    1       i = ni - 1
c
            do  j = 2,n
               nj = j
               if(y(j).gt.yy) go to 2
            enddo
    2       j = nj - 1
c
            hx = x(i+1) - x(i)
            tx = (xx - x(i))/hx
c
            hy = y(j+1) - y(j)
            ty = (yy - y(j))/hy
c
            f(1) = (1.- tx)**2*(1. + 2.*tx)
            f(2) = tx**2*(3.- 2*tx)
            f(3) = tx*(1. - tx)**2*hx
            f(4) =-tx**2*(1.- tx)*hx
c
            g(1) = (1.- ty)**2*(1. + 2.*ty)
            g(2) = ty**2*(3.- 2*ty)
            g(3) = ty*(1. - ty)**2*hy
            g(4) =-ty**2*(1.- ty)*hy
c
            sum(1) =  z(i+(j-1)*m)*f(1) +  z(i+1+(j-1)*m)
     *             + zx(i+(j-1)*m)*f(3) + zx(i+1+(j-1)*m)
c
            sum(2) =  z(i+j*m)*f(1) +  z(i+1+j*m)*f(2)
     *             + zx(i+j*m)*f(3) + zx(i+1+j*m)*f(4)
c
            sum(3) =  zy(j+(i-1)*n)*f(1) +  zy(j+i*n)*f(2)
     *             + zxy(j+(i-1)*n)*f(3) + zxy(j+i*n)*f(4)
c
            sum(4) =  zy(j+1+(i-1)*n)*f(1) +  zy(j+1+i*n)
     *             + zxy(j+1+(i-1)*n)*f(3) + zxy(j+1+i*n)
c
            spl = 0.
            do k  = 1, 4
               spl = spl + g(k)*sum(k)
            enddo
c
            return
            end
```

Обращение к программе имеет вид:

```
call Splint2
*   (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,ind,ibx,
*    iby,xx,yy,a,b,c,d,u,v, w,t,s,zz,su,sd,spl)
```

Входные данные:

m - число узлов сетки по оси x;

x - массив длины m, содержит узлы сетки по оси x;

n - число узлов сетки по оси y;

y - массив длины n, содержит узлы сетки по оси y;

z - одномерный массив длины n*m, содержит значения интерполируемой функции в узлах интерполяции; значение функции в узле сетки (i, j) содержится в элементе z(i+(j-1)*m);

ibx - код типа граничных условий на прямых x = x1 и x = xm,

ibx = i (i=1-4) - условия i-го типа;

zl - массив длины n, элемент zl(j) содержит значение i-й частной производной по x в узлах (1, j) для граничных условий i-го типа (i = 1, 2);

zr - массив длины n, элемент zr(j) содержит значение i-й частной производной по x в узлах (m, j) для граничных условий i-го типа (i = 1, 2);

iby - код типа граничных условий на прямых y = y1 и y = yn,

iby = i (i=1-4) - условия i-го типа;

zu - массив длины m, элемент zu(j) содержит значение i-й частной производной по y в узлах (j, m) для граничных условий i-го типа (i = 1, 2);

zd - массив длины m, элемент zd(j) содержит значение i-й частной производной по y в узлах (j, 1) для граничных условий i-го типа (i = 1, 2);

zxy - массив длины 4, содержит значения смешанных производных в угловых точках сетки в случае граничных условий 1-го или 2-го типа;

a,b,c,d - рабочие массивы длины max(n,m),

s,t,u,v,w,zz,su,sd - рабочие массивы длины max(n,m)+1,

zx,zy,zxy - одномерные массивы длины n*m, содержат значения параметров сплайна;

ind - код режима работы:

ind = 0 - программа вычисляет параметры сплайна,

ind = 1 - параметры сплайна известны;

xx,yy - координаты точки, в которой вычисляются параметры сплайна.

Результат:

spl - значение сплайна в точке с координатами (xx,yy).

Чтобы показать, как пользоваться программой Splint2, рассмотрим следующий пример.

Пример. Построим сетку из 11×6 узлов в прямоугольнике $[0, \pi] \times [0, \pi/2]$,

$$x_i = \frac{\pi}{10} i, \quad i = 0,1, \ ..., \ 10, \qquad y_j = \frac{\pi}{10} j, \quad j = 0,1, \ ..., \ 5,$$

и вычислим значения функции $f(x, y) = \sin(x)\sin(y)$ в этих узлах. С этими исходными данными построим интерполяционный кубический сплайн, удовлетворяющий граничным условиям 1-го типа на всей границе прямоугольника, а затем вычислим значения сплайна в точках с координатами

$$x_i = \frac{\pi}{20}(i-1), \quad i = 1, \ ..., \ 21, \qquad x_j = \frac{\pi}{20}(j-1), \quad j = 1, \ ..., \ 11.$$

Множество этих точек обозначим Ω.

Для оценки качества интерполяции сравним значения сплайна и заданной функции на множестве Ω.

Программа для решения этой задачи может иметь следующий вид:

```
      dimension x(100) , y(100) ,  z(100) , zl(100),
     *          zr(100), zu(100),  zd(100), zx(100),
     *          zy(100), zxy(100), u(100) , v(100) ,
     *          a(100) , b(100),   c(100) , d(100) ,
     *          w(100) , t(100),   su(100), sd(100),
     *          s(100) , zz(100)
c
      data pi/3.1415926/
c
      hx  = pi/10.
      hy  = pi/10
      m   = 11
      n   = 6
      ibx = 1
      iby = 1
c
      do i = 1,m
       x(i)  = hx*(i-1)
       zd(i) = sin(x(i))
       zu(i) = 0.
      enddo
c
      do j = 1,n
       y(j)  =  hy*(j-1)
       zl(j) =  sin(y(j))
       zr(j) = -sin(y(j))
      enddo
c
      do i = 1,m
      do j = 1,n
       z(i+(j-1)*m) = sin(x(i))*sin(y(j))
      enddo
      enddo
c
      zxy(1) =  1.
      zxy(2) = -1.
      zxy(3) =  0.
```

```
            zxy(4) =  0.
      c
            call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,0,
     *          ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
      c
            hx = pi/20.
            hy = pi/20.
      c
            del = 0.
            do j = 1,11
            do i = 1,21
            xx = hx*(i-1)
            yy = hy*(j-1)
            fun = sin(xx)*sin(yy)
            call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,
     *         1,ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
            if (abs(fun-spl).gt.del) del=abs(fun-spl)
            enddo
            enddo
            write (*,*) 'del= ',del
           stop
           end
```

Вычисления по этой программе привели к следующим результатам:

$$\max_{\Omega}\left|f(x,y)-S(x,y)\right| = 5.078316E-05.$$

Для построения по тому же массиву данных интерполяционного сплайна, удовлетворяющего граничным условиям 2-го типа, можно воспользоваться следующей программой:

```
            dimension x(100) , y(100) , z(100) , zl(100), zr(100),
     *                zu(100), zd(100), zx(100), zy(100),zxy(100),
     *                u(100) , v(100) , a(100) , b(100) , c(100) ,
     *                d(100) , w(100) , t(100) , su(100), sd(100),
     *                s(100) , zz(100)
      c
            data pi/3.1415926/
            hx = pi/10.
            hy = pi/10
      c
            m   = 11
            n   = 6
            ibx = 2
            iby = 2
      c
            do i = 1,m
             x(i)  = hx*(i-1)
             zd(i) = 0
             zu(i) = -sin(x(i))
            enddo
      c
            do j = 1,n
             y(j)  =  hy*(j-1)
             zl(j) =  0.
             zr(j) =  0.
            enddo
      c
```

```
      do i = 1,m
      do j = 1,n
       z(i+(j-1)*m) = sin(x(i))*sin(y(j))
      enddo
      enddo
c
      zxy(1) =  0.
      zxy(2) =  0.
      zxy(3) =  0.
      zxy(4) =  0.
c
      call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,0,
     *     ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
c
      hx = pi/20.
      hy = pi/20.
      open (1,file='res2.dat')
      del = 0.
      do j = 1,11
      do i = 1,21
      xx = hx*(i-1)
      yy = hy*(j-1)
c
      fun = sin(xx)*sin(yy)
c
      call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,
     *      1,ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
      if (abs(fun-spl).gt.del) del=abs(fun-spl)
      enddo
      enddo
      write (1,*) 'del=',del

      stop
      end
```

Вычисления по этой программе привели к следующему результату:

$$\max_{\Omega}\left|f(x,y)-S(x,y)\right| = 8.755922\text{E-}05.$$

Приведенная ниже программа строит интерполяционный сплайн, удовлетворяющий граничным условиям 3-го типа.

```
      dimension x(100) , y(100) , z(100) , zl(100), zr(100),
     *          zu(100), zd(100), zx(100), zy(100), xy(100),
     *          u(100) , v(100) , a(100) , b(100) , c(100) ,
     *          d(100) , w(100) , t(100) , su(100), sd(100),
     *          s(100) , zz(100)
c
      data pi/3.1415926/
      hx = 2.*pi/10.
      hy = 2.*pi/5.
c
      m   = 11
      n   = 6
      ibx = 3
      iby = 3
c
      do i = 1,m
       x(i)  = hx*(i-1)
      enddo
```

```
c
          do j = 1,n
           y(j)   =  hy*(j-1)
          enddo
c
          do i = 1,m
          do j = 1,n
           z(i+(j-1)*m) = sin(x(i))*sin(y(j))
          enddo
          enddo
c
          call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,0,
     *         ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
c
          hx = 2.*pi/20.
          hy = 2.*pi/20.
          open (1,file='res3.dat')

          del = 0.
          do j = 1,21
          do i = 1,21
          xx = hx*(i-1)
          yy = hy*(j-1)
          fun = sin(xx)*sin(yy)
          call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,
     *           1,ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
          if (abs(fun-spl).gt.del) del=abs(fun-spl)
          enddo
          enddo
          write (1,*) 'del=',del
          close (1)
         stop
         end
```

Вычисления по этой программе привели к следующему результату:

$$\max_{\Omega}\left|f(x,y)-S(x,y)\right| = 9.367764\text{E}-03.$$

Если необходимо построить интерполяционный сплайн, удовлетворяющий граничным условиям 4-го типа, то можно воспользоваться следующей программой:

```
          dimension x(100) , y(100) , z(100) , zl(100) , zr(100) ,
     *              zu(100) , zd(100) , zx(100) , zy(100) ,zxy(100) ,
     *              u(100)  , v(100)  , a(100)  , b(100)  , c(100)  ,
     *              d(100)  , w(100)  , t(100)  , su(100) , sd(100) ,
     *              s(100)  , zz(100)
c
          data pi/3.1415926/
          hx = 2.*pi/10.
          hy = 2.*pi/5.
c
          m   = 11
          n   = 6
          ibx = 4
          iby = 4
c
          do i = 1,m
           x(i)   = hx*(i-1)
          enddo
```

```
c
      do j = 1,n
       y(j)  =  hy*(j-1)
      enddo
c
      do i = 1,m
      do j = 1,n
       z(i+(j-1)*m) = sin(x(i))*sin(y(j))
      enddo
      enddo
c
      call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,0,
     *     ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
c
      hx = 2.*pi/20.
      hy = 2.*pi/20.
      open (1,file='res4.dat')
      del = 0.
      do j = 1,21
      do i = 1,21
      xx = hx*(i-1)
      yy = hy*(j-1)
      fun = sin(xx)*sin(yy)
      call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,
     *      1,ibx,iby,xx,yy,a,b,c,d,u,v,w,t,s,zz,su,sd,spl)
      if (abs(fun-spl).gt.del) del=abs(fun-spl)
      enddo
      enddo
      write (1,*) 'del=',del
     stop
     end
```

Вычисления по этой программе привели к следующему результату:

$$\max_{\Omega}\left|f(x,y)-S(x,y)\right| = 5.985552\text{E-}02.$$

Текст программы Splint2 находится в файле splint2.for в поддиректории INTERPOL на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". В эту же поддиректорию помещены файлы, содержащие примеры применения программы Splint2 при других граничных условиях. Подробную информацию об именах этих файлов и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

2.2. Сглаживающие бикубические сплайны

2.2.1. О постановке задачи сглаживания

Пусть заданы сетка ω

$$a = x_0 < x_1 < \ldots < x_{m-1} < x_m = b,\ c = y_0 < y_1 < \ldots < y_{n-1} < y_n = d$$

и набор чисел

$$z_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n.$$

Комментарий к исходным данным. Величины z_{ij} удобно интерпретировать, например, как результаты измерений некоторой функции $z(x, y)$ при заданных значениях переменных x и y, содержащие случайную погрешность. При решении задачи восстановления функции по таким ее экспериментальным значениям вряд ли целесообразно использовать интерполяцию, поскольку интерполяционная функция будет послушно воспроизводить причудливые осцилляции, обусловленные случайной компонентой в массиве $\{z_{ij}\}$. Более естественным является подход, основанный на процедуре сглаживании, призванной как-то уменьшить элемент случайности в результатах измерений.

Существует много разнообразных процедур сглаживания. Формально они чаще всего ставятся как задачи отыскания экстремума определяющего функционала, зависящего от сетки ω и от массива $\{z_{ij}\}$, на некотором классе допустимых функций.

2.2.2. Определение сглаживающего бикубического сплайна

Сглаживающим бикубическим сплайном $S(x, y)$ *на сетке* ω называется функция, которая:

1) в каждой ячейке

$$R_{ij} = \{(x, y) \mid x_i \le x \le x_{i+1},\ y_j \le y \le y_{j+i}\}.$$

$$i = 0, 1, \ldots, m - 1, \quad j = 0, 1, \ldots, n - 1$$

представляет собой многочлен вида

$$S(x, y) = \sum_{p=0}^{3} \sum_{q=0}^{3} a_{p,q}^{i,j} (x - x_i)^p (y - y_j)^q ;$$

2) принадлежит классу $C^{2,2}(R)$;

3) доставляет минимум функционалу

$$J(f)=\int_a^b\int_c^d\left(\frac{\partial^4 f(x,y)}{\partial x^2\partial y^2}\right)^2 dxdy+$$

$$\sum_{i=0}^{m}\frac{1}{\rho_i}\int_a^b\left(\frac{\partial^2 f(x_i,y)}{\partial y^2}\right)^2 dy+\sum_{j=0}^{n}\frac{1}{\sigma_j}\int_c^d\left(\frac{\partial^2 f(x,y_j)}{\partial x^2}\right)^2 dx+$$

$$\sum_{i=0}^{m}\sum_{j=0}^{n}\frac{1}{\rho_i\sigma_j}\left(f(x_i,y_j)-z_{ij}\right)^2,$$

где $z_{ij}\left(i=0,\ 1,\ \ldots,\ m;\ j=0,\ 1,\ \ldots,\ n\right)$, $\rho_i>0\ \left(i=0,\ \ldots,\ m\right)$ и $\sigma_j>0\ \left(j=0,\ \ldots,\ n\right)$ - заданные числа;

4) удовлетворяет краевым условиям одного из трех указанных ниже типов.

2.2.3. Граничные (краевые) условия

Граничные (краевые) условия задаются в виде ограничений на значения сплайна и его производных в граничных и угловых узлах сетки ω.

Граничные (краевые) условия 1-го типа

$$\frac{\partial S}{\partial x}(x_i,y_j)=z_{ij}^{(x)},\qquad i=0,\ m,\qquad j=0,\ 1,\ \ldots,\ n,$$

$$\frac{\partial S}{\partial y}(x_i,y_j)=z_{ij}^{(y)},\qquad i=0,\ 1,\ \ldots,\ m,\qquad j=0,\ n,$$

$$\frac{\partial^2 S}{\partial x\partial y}(x_i,y_j)=z_{ij}^{(xy)},\qquad i=0,\ m,\qquad j=0,\ n,$$

- в граничных узлах сетки ω задаются значения 1-х частных производных по x и по y искомой функции, а в угловых узлах - значения 2-ой смешанной производной.

Число граничных (краевых) условий 1-го типа равно $2m+2n+8$.

Наглядное представление о способе задания сплайна, отвечающего граничным условиям этого типа дает рис. 2.4. На нем жирными точками отмечены узлы сетки, которым соответствуют значения z_{ij}, горизонтальными и вертикальными стрелками указаны узлы, в которых задаются значения первых частных производных S_x и S_y соответственно, а ромбиком - значения смешанной производной S_{xy}.

Граничные (краевые) условия 2-го типа

$$\frac{\partial^2 S}{\partial x^2}(x_i, y_j) = 0, \qquad i = 0,\ m, \qquad j = 0,\ 1,\ \ldots,\ n,$$

$$\frac{\partial^2 S}{\partial y^2}(x_i, y_j) = 0, \qquad i = 0,\ 1,\ \ldots,\ m, \qquad j = 0,\ n,$$

$$\frac{\partial^4 S}{\partial x^2 \partial y^2}(x_i, y_j) = 0, \qquad i = 0,\ m, \qquad j = 0,\ n.$$

- в граничных узлах сетки ω задаются значения 2-х частных производных по x и по y искомой функции, а в угловых узлах - значения смешанной производной S_{xxyy}.

Число граничных (краевых) условий 2-го типа равно $2m + 2n + 8$.

Наглядное представление о способе задания сплайна, отвечающего граничным условиям этого типа, также дает рис. 2.4. На нем жирными точками отмечены узлы сетки, которым соответствуют значения z_{ij}, горизонтальными и вертикальными стрелками указаны узлы, в которых задаются значения первых частных производных S_{xx} и S_{yy} соответственно, а ромбиком - значения смешанной производной S_{xxyy}.

Граничные (краевые) условия 3-го типа

$$S(x_0, y_j) = S(x_m, y_j), \qquad j = 0,\ 1,\ \ldots,\ n,$$

$$S(x_i, y_0) = S(x_i, y_n), \qquad i = 0,\ 1\ ,\ \ldots,\ m,$$

$$\frac{\partial^k S}{\partial x^k}(x_0, y_j) = \frac{\partial^k S}{\partial x^k}(x_m, y_j), \qquad j = 0,\ 1\ ,\ \ldots,\ n, \qquad k = 1,\ 2,$$

$$\frac{\partial^l S}{\partial y^l}(x_i, y_0) = \frac{\partial^l S}{\partial y^l}(x_i, y_n), \qquad i = 0,\ 1,\ \ldots,\ m, \qquad l = 1,\ 2,$$

$$\frac{\partial^{2k} S}{\partial x^k \partial y^k}(x_0, y_j) = \frac{\partial^{2k} S}{\partial x^k \partial y^k}(x_m, y_j), \qquad j = 0,\ 1,\ \ldots,\ n, \qquad k = 1,\ 2,$$

$$\frac{\partial^{2l} S}{\partial x^l \partial y^l}(x_i, y_0) = \frac{\partial^{2l} S}{\partial x^l \partial y^l}(x_l, y_n), \qquad i = 0,\ 1,\ \ldots,\ m, \qquad l = 1,\ 2,$$

Все они и называются *периодическими*. В этом случае сплайн $S(x, y)$ и его частные производные должны быть двоякопериодическими функциями - с периодом $T_x = b - a$ по переменной x и с периодом $T_y = c - d$ по переменной y.

Теорема. Среди всех функций из класса $C^{2,2}(R)$, удовлетворяющих граничным условиям 2-го типа (*естественным условиям*), именно бикубический сплайн доставляет минимум функционалу $J(f)$.

Определение. Бикубический сплайн, минимизирующий функционал $J(f)$ и удовлетворяющий граничным условиям i-го типа, называется *сглаживающим сплайном i-го типа.*

Замечания:

1. Среди всех функций из класса $C^{2,2}(R)$, удовлетворяющих граничным условиям 2-го типа, именно бикубический сплайн доставляет минимум функционалу $J(f)$.

2. Кроме перечисленных, возможны и смешанные граничные условия, то есть условия, относящиеся по разным переменным к разным типам. При этом если, например, по переменной x заданы условия 1-го типа, а по переменной y – 2-го типа, то в вершинах прямоугольника R следует задавать частные производные S_{xyy} и т. д.

2.2.4. Построение сглаживающего бикубического сплайна

Алгоритм построения сглаживающего бикубического сплайна основан на том, что его построение можно свести к решению последовательности одномерных задач сглаживания.

1-й шаг алгоритма

На каждой линии $y = y_k$ $(k = 0, 1, ..., n)$ строятся одномерные кубические сглаживающие сплайны $S_k(x)$ по переменной x с соответствующими граничными условиями (процедура построения таких сплайнов приведена в гл. 1).

2-й шаг алгоритма

Для каждого узла (x_i, y_j) сетки ω вычисляются "исправленные" значения $z_{ij}^{(1)}$ по формуле $z_{ij}^{(1)} = S_j(x_i)$.

3-й шаг алгоритма

аналогичен 1-му шагу: на каждой из линий $x = x_l$ $(l = 0, 1, ..., m)$ для массива $z_{lj}^{(1)}$ строится сглаживающий кубический сплайн $\tilde{S}_l(y)$ по переменной y, удовлетворяющий граничным условиям соответствующего типа.

4-й шаг алгоритма

Для каждого узла сетки ω вычисляются новые "исправленные" значения $z_{ij}^{(2)}$ по формуле $z_{ij}^{(2)} = \tilde{S}_i(y_j)$.

5-й шаг алгоритма

Для новой системы значений $z_{ij}^{(2)}$ строится интерполяционный сплайн $S(x, y)$, который и будет искомым сглаживающим сплайном.

После выполнения 5-го шага алгоритма в каждом узле сетки ω будут известны величины

$$z_{ij}^{(2)},\ \mu_{ij}^{(x)},\ \mu_{ij}^{(y)},\ \mu_{ij}^{(xy)},$$

которые полностью определяют сглаживающий бикубический сплайн, удовлетворяющий заданным граничным условиям.

Алгоритм вычисления значений сплайна по заданным величинам

$$z_{ij}^{(2)},\ \mu_{ij}^{(x)},\ \mu_{ij}^{(y)},\ \mu_{ij}^{(xy)}$$

полностью совпадает с алгоритмом, приведенным в 2.1.4.

Замечания:

1. Перечисленные выше сплайны являются решениями соответствующих вариационных задач в гораздо более широком классе функций, а именно в классе $W_2^{2,2}(R)$.

2. При помощи весовых множителей ρ_i *и* σ_j *можно управлять свойствами сглаживающего сплайна, например, определяя форму слоя вблизи "экспериментальных" точек* z_{ij}, *в котором размещается сглаживающий сплайн. Так как в двойную сумму* $J(f)$ *весовые множители* ρ_i *и* σ_j *входят в виде произведений, то их выбор будет оказывать влияние на характер сглаживания не в одной точке* (x_i, y_j), *а соответственно на линиях* $x = x_i$ *и* $y = y_j$.

3. Значительно более сложный случай, когда весовые множители в двойной сумме имеют значения $\tau_{ij} \neq \rho_i \sigma_j$, *здесь не рассматривается.*

2.2.5. Построение сплайновых поверхностей при помощи сплайн-функций

Выше рассматривались массивы, точки которых были занумерованы так, что и их абсциссы, и их ординаты образовывали строго возрастающие последовательности. Это обстоятельство определяло и выбор класса аппроксимирующих поверхностей (графики функций) и способ их построения.

Однако предложенный метод позволяет достаточно успешно строить сглаживающую поверхность и в более общем случае, когда нумерация точек массива

$$\mathbf{P} = \left\{ \mathbf{P}_{ij}\left(x_{ij}, y_{ij}, z_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n \right\}$$

и их расположение в пространстве, как правило, не связаны.

Ясно, что для решения этой общей задачи необходимо существенно расширить класс допустимых поверхностей, включив в него и поверхности, которые нельзя однозначно спроектировать ни на одну из координатных плоскостей, и поверхности с самопересечениями и самоналеганиями. Такие поверхности удобно описывать при помощи параметрических уравнений

$$x = x(u,v),\quad y = y(u,v),\quad z = z(u,v),$$

$$(u,v) \in R = [\alpha,\beta] \times [\gamma,\delta].$$

Потребуем дополнительно, чтобы функции $x(u, v)$, $y(u, v)$ и $z(u, v)$ обладали достаточной гладкостью, например принадлежали классу $C^1(R)$ или классу $C^2(R)$.

Для отыскания параметрических уравнений поверхности, сглаживающей заданный массив, поступим следующим образом.

1-й шаг. На произвольно взятом прямоугольнике R изменения параметров u и v вводится вспомогательная сетка

$$\alpha = u_0 < u_1 < \ \ldots\ < u_{m-1} < u_m = \beta,\quad \gamma = v_0 < v_1 < \ \ldots\ < v_{n-1} < v_n = \delta,$$

число узлов которой совпадает с числом точек в массиве **P**.

2-й шаг. По заданному массиву **P** строятся 3 новых вспомогательных массива:

$$\mathbf{X} = \left\{ \left(u_i, v_j, x_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n \right\}$$

$$\mathbf{Y} = \left\{ \left(u_i, v_j, x_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n \right\}$$

$$\mathbf{Z} = \left\{ \left(u_i, v_j, x_{ij}\right),\ i = 0,\ 1,\ \ldots,\ m;\ j = 0,\ 1,\ \ldots,\ n \right\}$$

3-й шаг. Для каждого из массивов **X**, **Y** и **Z** методами, изложенными выше, находятся соответствующие сглаживающие сплайн-функции $x(u, v)$, $y(u, v)$ и $z(u, v)$.

В результате мы получаем параметрические уравнения сглаживающей сплайновой поверхности, порожденной точками $\mathbf{P}_{ij}$.

Замечание

Полученная поверхность будет гладкой, но не обязательно регулярной, так как возможность

$$\operatorname{rank}\begin{pmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \end{pmatrix} < 2$$

для некоторой точки $(u^*, v^*) \in (\alpha, \beta) \times (\gamma, \delta)$ *исключать нельзя.*

Кроме того, эта поверхность может иметь линии самопересечения и участки самоналегания.

2.2.6. Программная реализация

Описанный выше алгоритм построения бикубического сглаживающего сплайна и вычисления его значений в произвольной точке реализован в виде подпрограммы Smspl2, написанной на языке Фортран:

```
      subroutine Smspl2
     *   (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,zy,zxy,ind,ib,xx,
     *   yy,a,b,c,d,e,g,u,v, w,t,s,zz,su,sd,p,q,r,dd,zm,spl)
c**************************************************************
c     Программа Splint2 строит сглаживающий бикубический     *
c               сплайн для таблично заданной функции          *
c               двух переменных                               *
c z   - одномерный массив длины m*n содержит значения         *
c       сглаживающего сплайна в узлах сетки;                  *
c       значение сплайна в узле (i,j) содержится в элементе   *
c       z(i+(j-1)*m)                                          *
c m   - число узлов сетки по оси x                            *
c x   - массив длины m, содержит узлы сетки по оси x          *
c n   - число узлов сетки по оси y                            *
c y   - массив длины n, содержит узлы сетки по оси y          *
c z   - одномерный массив длины n*m, содержит значения        *
c       сглаживающего сплайна в узлах сетки;                  *
c       значение функции в узле сетки (i,j) содержится в      *
c       элементе z(i+(j-1)*m)                                 *
c ib  - код типа граничных условий на прямых x=x1, x=xm и     *
c       на прямых y=y1 и y=yn                                 *
c                ib = i (i=1-3) - условия i-го типа           *
c zl  -  массив длины n, элемент zl(j) содержит значение      *
c       i-й частной производной по x в узлах (1,j)            *
c       для граничных условий i-го типа (i=1,2)               *
c zr  -  массив длины n, элемент zr(j) содержит значение      *
c       i-й частной производной по x в узлах (m,j)            *
c       для граничных условий i-го типа (i=1,2)               *
c zu  -  массив длины m, элемент zu(j) содержит значение      *
c       i-й частной производной по y в узлах (j,m)            *
c       для граничных условий i-го типа (i=1,2)               *
c zd  -  массив длины m, элемент zd(j) содержит значение      *
c       i-й частной производной по y в узлах (j,1)            *
c       для граничных условий i-го типа (i=1,2)               *
c zxy-  массив длины 4, содержит значения смешанных           *
c       производных в угловых точках сетки в случае гра-     *
c       ничных условий 1-го или 2-го типа                     *
c rho-  массив длины m для размещения весовых коэффициен-     *
c       тов функционала                                       *
c sgm-  массив длины n для размещения весовых коэффициен-     *
c       тов функционала                                       *
c                                                             *
c a,b,c,d    - рабочие массивы длины max(n,m)                 *
c s,t,u,v,w,zz,su,sd - рабочие массивы длины  max(n,m)+1      *
c p,q,r,e,g - рабочие массивы длины  max(n,m)+2               *
c zx,zy,zm,zz,dd - рабочие массивы длины  max(n,m)            *
c                                                             *
c ind - код режима работы:                                    *
```

```
c          ind=0 - программа вычисляет параметры сплайна     *
c          ind=1 - параметры сплайна известны                 *
c                                                             *
c xx,yy - координаты точки, в которой вычисляются парамет-    *
c          ры сплайна                                         *
c                                                             *
c          Результат:                                         *
c  spl   - значение сплайна в точке с координатами (xx,yy)    *
c*************************************************************

      dimension x(1) , y(1) , z(1) , zl(1) , zr(1), zu(1),
     *          zd(1), zx(1), zy(1), zxy(1), u(1) , v(1) ,
     *          w(1) , t(1) , s(1) , su(1) , sd(1), a(1) ,
     *          b(1) , c(1) , d(1) , zz(1) , e(1) , g(1) ,
     *          p(1) , q(1) , r(1) , dd(1) , zm(1),
     *          rho(1),sgm(1)
c
      if (ind.eq.0) then
      if (ib.eq.2) then
         do i=1,m
            zu(i) = 0.
            zd(i) = 0.
         enddo
         do j=1,n
            zl(j) = 0.
            zr(j) = 0.
         enddo
         do j=1,4
            zxy(j) = 0.
         enddo
       endif
c
         do j = 1,n
         call Smspline (m,ib,0,x,z(1+(j-1)*m),a,b,c,d,e,g,
     *                  rho,zl(j),zr(j),u,v,w,p,q,r,s,t,
     *                  zx(1+(j-1)*m),dd,zm,xx,sp,dsp,d2sp)
         enddo
c
         do i = 1,m
              do j = 1,n
                 zz(j) = zx(i + (j-1)*m)
              enddo
         call Smspline (n,ib,0,y,zz,a,b,c,d,e,g,sgm,zu(i),
     *                  zd(i),u,v,w,p,q,r,s,t,zy(1+(i-1)*n),
     *                  dd,zm,xx,sp,dsp,d2sp)
         enddo
c
         do i = 1,m
         do j = 1,n
           z(i+(j-1)*m) = zy(j+(i-1)*n)
         enddo
         enddo
c
         call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,0,ib,
     *                  ib,xx,yy,a,b,c,d,u,v, w,t,s,zz,su,sd,spl)
         endif
c
         call  Splint2 (m,x,n,y,z,zl,zr,zu,zd,zx,zy,zxy,1,ib,
     *                  ib,xx,yy,a,b,c,d,u,v, w,t,s,zz,su,sd,spl)

         return
```

```
      end
```

Обращение к программе имеет вид:

```
call Smspl2
     *  (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,zy,zxy,ind,ib,xx,
     *  yy,a,b,c,d,e,g,u,v, w,t,s,zz,su,sd,p,q,r,dd,zm,spl)
```

Входные данные:

m - число узлов сетки по оси *x*,

x - массив длины m, содержит узлы сетки по оси *x*,

n - число узлов сетки по оси *y*,

y - массив длины n, содержит узлы сетки по оси *y*,

z - одномерный массив длины n*m, содержит значения сглаживающего сплайна в узлах сетки; значение сплайна в узле сетки (i, j) содержится в элементе z(i+(j-1)*m),

ib - код типа граничных условий на прямых x = x1 и x = xm и на прямых y = y1 и y = yn,

ib = i (i=1-3) - условия i-го типа ,

zl - массив длины n, элемент zl(j) содержит значение 1-ой частной производной по *x* в узлах (1, j) для граничных условий 1-го типа,

zr - массив длины n, элемент zr(j) содержит значение 1-й частной производной по x в узлах (m, j) для граничных условий 1-го типа,

zu - массив длины m, элемент zu(j) содержит значение 1-й частной производной по y в узлах (j, m) для граничных условий 1-го типа,

zd - массив длины m, элемент zd(j) содержит значение 1-й частной производной по y в узлах (j, 1) для граничных условий 1-го типа,

zxy - массив длины 4, содержит значения смешанных производных в угловых точках сетки в случае граничных условий 1-го типа,

rho - массив длины m содержит значения весовых коэффициентов,

sgm - массив длины n содержит значения весовых коэффициентов,

a,b,c,d - рабочие массивы длины max(n,m))

s,t,u,v,w,zz,su,sd - рабочие массивы длины max(n,m)+1

p,q,r,e,g - рабочие массивы длины max(n,m)+2

zx,zy,zm,zz,dd - рабочие массивы длины max(n,m)

ind - код режима работы:

ind = 0 - программа вычисляет параметры сплайна,

ind = 1 - параметры сплайна известны,

xx,yy - координаты точки, в которой вычисляются параметры сплайна.

Результат:

spl - значение сплайна в точке с координатами (xx,yy).

В процессе работы программа Smspl2 обращается к программам Smspline и Splint2.

Чтобы показать, как пользоваться программой Splint2, рассмотрим следующий пример, имитирующий сглаживание экспериментальных данных.

Пример. Предположим, что фунция $z = \sin(x + y^2)$ "измеряется" в узлах сетки

$$\varpi = \{x_i = \frac{1}{10} i, \quad y_j = \frac{1}{20} j, \quad i = 0,1, \ldots, 5, \ j = 0,1, \ldots, 20\}.$$

с погрешностью ξ, которая распределена равномерно на отрезке [0.1, 0.1]. Для весовых коэффициентов примем значения

$$\rho_i = 0.001, \ i = 0, \ldots, 5, \quad \sigma_j = 0.001, \ j = 0, \ldots, 20.$$

Программа для решения этой задачи может иметь следующий вид:

```
      dimension x(200) , y(200) , z(200) , zl(200) , zr(200) ,
     *          zu(200) , zd(200) , zx(200) , zy(200) , zxy(200),
     *          u(200) , v(200) , a(200) , b(200) , c(200) ,
     *          d(200) , w(200) , t(200) , su(200) , sd(200) ,
     *          s(200) , zz(200), rho(200), sgm(200), p(200) ,
     *          q(200) , r(200) , dd(200) , zm(200) , e(200) ,
     *          g(200)
c
      data rho / 200*0.001/
      data sgm / 200*0.001/
c
      m  = 6
      n  = 21
      hx = 0.5/(m-1)
      hy = 1./(n-1)
c
      do i = 1,m
         x(i) = hx*(i-1)
         zu(i) = 2.*cos(1. + x(i))
         zd(i) = 0.
      enddo
c
      do j = 1,n
         y(j) = hy*(j-1)
         zl(j) = cos(y(j)**2)
```

```
            zr(j)= cos(0.5 + y(j)**2)
          enddo
c
          zxy(1) = 0.
          zxy(2) = 0.
          zxy(3) = -2.*sin(1.)
          zxy(4) = -2.*sin(1.5)
          call seed(1234)
          do j = 1,n
          do i = 1,m
             call random(df)
             z(i+(j-1)*m) = sin(x(i)+y(j)**2) + 0.4*(df-0.5)
          enddo
          enddo
c
          ind = 0
          ib  = 1
c
          call  Smspl2   (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,zy,
     *                    zxy,ind,ib,xx,yy,a,b,c,d,e,g,u,v, w,
     *                    t,s,zz,su,sd,p,q,r,dd,zm,spl)
c
          ind=1
          open  1,file='res1.dat'
          hx = hx/2.
          hy = hy/5.
c
          do i = 1,11
            xx = hx*(i-1)
             do j = 51,61
              yy = hy*(j-1)
              call Smspl2 (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,
     *                     zy,zxy,ind,ib,xx,yy,a,b,c,d,e,g,
     *                     u,v, w,t,s,zz,su,sd,p,q,r,dd,zm,spl)
              write  1,*   xx, yy,  spl
             enddo
          enddo
c
          stop
          end
```

Замечание

Сравнение сглаживающего сплайна с интерполяционным, построенным на том же массиве исходных данных и удовлетворяющим тем же граничным условиям, показывает, что осциляции, присущие интерполяционному сплайну, практически полностью отсутствуют в сглаживающем сплайне.

Приведенная ниже программа сначала создает массив "экспериментальных" данных, а затем строит сглаживающий сплайн, удовлетворяющий граничным условиям 2-го типа.

```
      dimension x(200) , y(200) , z(200)  , zl(200) , zr(200) ,
     *          zu(200), zd(200), zx(200) , zy(200) , zxy(200),
     *          u(200) , v(200) , a(200)  , b(200)  , c(200)  ,
     *          d(200) , w(200) , t(200)  , su(200) , sd(200) ,
```

```
      *          s(200) , zz(200), rho(200), sgm(200), p(200)   ,
      *          q(200) , r(200) , dd(200) , zm(200) , e(200)   ,
      *          g(200)
c
          data rho / 200*0.001/
          data sgm / 200*0.001/
c
          m  = 6
          n  = 21
          hx = 0.5/(m-1)
          hy = 1./(n-1)
c
          do i = 1,m
             x(i) = hx*(i-1)
          enddo
c
          do j = 1,n
             y(j) = hy*(j-1)
          enddo
c
          call seed(1234)
          do j = 1,n
          do i = 1,m
             call random(df)
             z(i+(j-1)*m) = sin(x(i)+y(j)**2) + 0.4*(df-0.5
          enddo
          enddo
c
          ind = 0
          ib  = 2
c
          call  Smspl2  (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,zy,
      *                  zxy,ind,ib,xx,yy,a,b,c,d,e,g,u,v, w,
      *                  t,s,zz,su,sd,p,q,r,dd,zm,spl)
c
          ind=1
          open (1,file='res2.dat')
          hx = hx/2.
          hy = hy/5.
c
          do i = 1,11
            xx = hx*(i-1)
             do j = 51,61
              yy = hy*(j-1)
              call Smspl2 (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,
      *                    zy,zxy,ind,ib,xx,yy,a,b,c,d,e,g,
      *                    u,v, w,t,s,zz,su,sd,p,q,r,dd,zm,spl)
              write (1,*)  xx, yy,  spl
             enddo
          enddo
c
         stop
         end
```

Следующая программа сначала создает массив “экспериментальных” данных, а затем строит сглаживающий сплайн, удовлетворяющий граничным условиям 3-го типа,

```
      dimension x(50) , y(50)  , z(500) , zl(50) , zr(50) ,
     *          zu(50), zd(50) ; zx(500), zy(500), zxy(500)
     *          u(50) , v(50)  , a(50)  , b(500) , c(500)
     *          d(50) , w(50)  , t(50)  , su(50) , sd(50) ,
     *          s(50) , zz(500), rho(50), sgm(50), p(50)  ,
     *          q(50) , r(50)  , dd(50) , zm(50) , e(50)  ,
     *          g(50)
c
      data rho / 50*0.01/
      data sgm / 50*0.01/
      data pi  / 3.1415926/
c
      m  = 21
      n  = 21
      hx = 2.*pi/(m-1)
      hy = 2.*pi/(n-1)
c
      do i = 1,m
         x(i) = hx*(i-1)
      enddo
c
      do j = 1,n
         y(j) = hy*(j-1)
      enddo
c
      call seed(1234)
      do j = 1,n-1
      do i = 1,m-1
         call random(df)
         z(i+(j-1)*m) = sin(x(i))+sin(y(j)) + (df-0.5)
      enddo
      enddo

      do i = 2,m-1
         z(i+(n-1)*m) = z(i)
      enddo

      do j = 2,n-1
         z(j*m) = z(1+(j-1)*m)
      enddo
c
         z(m)   = z(1)
         z(m*n) = z(1)
         z(1+(n-1)*m) = z(1)
c
      ind = 0
      ib  = 3
c
      call  Smspl2  (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,zy,
     *               zxy,ind,ib,xx,yy,a,b,c,d,e,g,u,v, w,
     *               t,s,zz,su,sd,p,q,r,dd,zm,spl)

c
      ind=1
      open (1,file='res3.dat')
      hx = hx/2.
      hy = hy/2.
```

```
c
            do j = 10,21
             yy = hy*(j-1)
           do i = 20,35
             xx = hx*(i-1)
              call Smspl2 (m,x,n,y,z,zl,zr,zu,zd,rho,sgm,zx,
     *                     zy,zxy,ind,ib,xx,yy,a,b,c,d,e,g,
     *                     u,v, w,t,s,zz,su,sd,p,q,r,dd,zm,spl)
              write (1,*)  xx, yy,  spl
            enddo
           enddo
c
          close (1)
          stop
          end
```

Текст программы Smspl2 находится в файле smspl2.for в поддиректории SMOOTH на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". В эту же поддиректорию помещены файлы, содержащие примеры применения программы Smspl2 при других граничных условиях. Подробную информацию об именах этих файлов и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

Часть II
Геометрические сплайны

Во многих задачах требование того, чтобы конструируемая кривая или поверхность однозначно проектировалась соответственно на прямую или плоскость, является слишком жестким. Расширяя допустимые классы кривых и поверхностей, естественно обратиться и к более общему способу описания их частичных фрагментов. В качестве нового способа задания кривых и поверхностей удобно взять параметрический способ.

Параметрическое задание кривой или поверхности имеет известные преимущества над другими методами, в частности, потому, что оно не накладывает практически никаких ограничений на множество вершин в опорном массиве.

Вместе с тем этот метод требует и большой осторожности: для того чтобы составная кривая или поверхность, описываемая параметрическими уравнениями, была достаточно регулярной, необходимо быть очень внимательным, особенно в местах стыковки: связь между геометрическими и аналитическими свойствами не всегда оказывается простой.

Выбор многочленов 3-й степени для описания координатных функций проектируемых кривых геометрически вполне обоснован: координатные функции должны быть сравнительно простыми и одновременно обеспечивающими разумную гладкость.

Как и в I части, при построении составных поверхностей разумно использовать результаты решения соответствующих одномерных задач, в частности ограничиться полиномиальным описанием координатных функций. Это заметно упрощает решение задачи создания поверхностей сложной формы.

Глава 3. Сплайновые кривые

Общую задачу, рассматриваемую в этой главе, можно сформулировать так: по заданному множеству вершин

$$\mathbf{P} = \{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{m-1}, \mathbf{P}_m\}$$

с учетом их нумерации построить гладкую кривую, которая, плавно изменяясь, последовательно проходила бы вблизи этих вершин и удовлетворяла некоторым дополнительным условиям. Эти условия могут иметь различный характер. Например, можно потребовать, чтобы искомая кривая проходила через все заданные вершины или, проходя через заданные вершины, касалась заданных направлений, являлась замкнутой или имела заданную регулярность и т. п.

При отыскании подходящего решения задачи приближения важную роль играет ломаная, звенья которой соединяют соседние вершины заданного набора. Эту ломаную называют *контрольной* или *опорной*, а ее вершины – *контрольными* или *опорными* (рис. 3.1). Во многих случаях она довольно точно показывает, как будет проходить искомая кривая, что особенно полезно при решении задачи сглаживания. Каждая вершина заданного массива является либо *внутренней* либо *граничной (концевой)*. В массиве $\mathbf{P}$ вершины $\mathbf{P}_i$, $(i = 1, \ldots, m-1)$ внутренние, а вершины $\mathbf{P}_0$ и $\mathbf{P}_m$ – граничные (концевые).

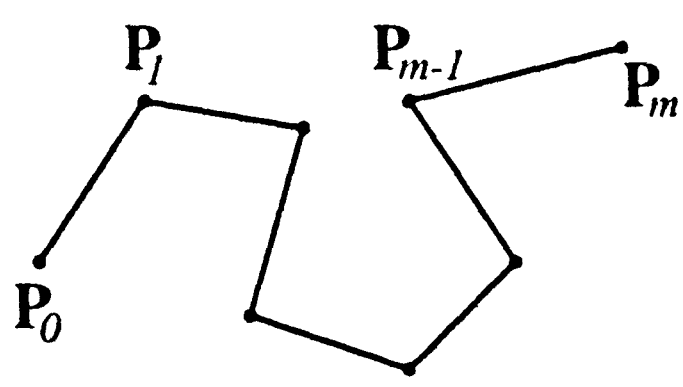


Рис. 3.1

В отличие от ситуации, рассматриваемой в гл. 1, в данном случае никаких ограничений на множество вершин не накладывается - они могут быть заданы как на плоскости, так и в пространстве, их взаимное расположение может быть совершенно произвольным, некоторые из вершин могут совпадать и т. д. Поэтому описание нужной кривой следует искать в более общей, параметрической форме, например в следующем виде

$$\mathbf{R}(t) = \sum a_i(t)\mathbf{P}_i\,, \qquad (3.1)$$

где $a_i(t)$ - некоторые функциональные коэффициенты, подлежащие определению.

Если количество вершин в заданном множестве **P** достаточно велико, то найти универсальные функциональные коэффициенты $a_i(t)$, как правило, довольно затруднительно. Если универсальные коэффициенты $a_i(t)$ все же найдены, то часто оказывается, что они наряду с нужными свойствами обладают и такими, которые не всегда удовлетворительно согласуются с ожидаемым поведением соответствующей кривой (например, кривая, описываемая уравнением (3.1) с этими коэффициентами, может осциллировать или отклоняться от заданного множества местами очень заметно).

Для успешного решения поставленной задачи приближения, весьма удобно привлечь кривые, составленные из элементарных фрагментов. В случае, когда эти элементарные фрагменты строятся по единой сравнительно простой схеме, такие составные кривые принято называть *сплайновыми кривыми.*

Параметрические уравнения каждого элементарного фрагмента ищутся в виде (*) с той лишь разницей, что всякий раз привлекается только часть заданных вершин множества **P**, а соответствующие коэффициенты имеют одинаковую природу: часто используются многочлены одинаковой степени, рациональные дроби, экспоненты и др.

Для описания элементарных кривых и вычисления их геометрических характеристик (информация о которых необходима при состыковке) в качестве функциональных коэффициентов обычно используются многочлены невысоких степеней, 2-й или 3-й, в первую очередь потому, что они сравнительно просто вычисляются. Конечно, привлекая многочлены больших степеней, можно описывать весьма сложные кривые. Однако у таких многочленов много коэффициентов, физический и геометрический смысл которых трудно понять. Кроме того, использование многочленов высокой степени может вызвать нежелательные колебания результирующей кривой.

Наибольшее распространение получили методы конструирования составных кривых, в которых используются кубические многочлены (которые, кстати, активно применялись и в гл. 1). Выбор в качестве функциональных коэффициентов кубических многочленов позволяет учесть и дифференциальные и внешнегеометрические требования, накладываемые на искомую кривую.

Высказанные соображения определяют характер изложения разделов этой главы. Глава начинается с описания минимально необходимого набора сведений из дифференциальной геометрии кривых. Затем рассматриваются различные классы кривых, при помощи которых можно решить поставленную задачу приближения. Описание каждого из классов проводится практически по одной и той же схеме. Сначала

определяются элементарные кривые (фрагменты), а затем показывается, как путем состыковки этих элементарных кривых получить составную кривую с достаточно хорошими характеристиками. Значительное внимание уделено описанию основных свойств рассматриваемых кривых. Такой подход поможет пользователю в выборе наиболее подходящего для него класса кривых. Часть из свойств формулируется в виде ответов на стандартный набор вопросов, среди которых есть, например, такие:

- каков порядок гладкости построенной кривой?
- каковы особенности расположения кривой относительно заданных вершин опорной ломаной?
- как сказываются на форме кривой изменения опорных вершин?

Полезно также знать, как связаны между собой форма кривой и простейшие геометрические преобразования (аффинные и проективные). Остановимся на этом подробнее.

Ясно, что, построив кривую, можно подвергнуть ее любому (аффинному или проективному) преобразованию. Можно поступить и по-иному: сначала подвергнуть этому преобразованию заданный набор опорных вершин и только потом построить по нему кривую из рассматриваемого класса. Если результаты оказываются одинаковыми, то говорят, что кривая *инвариантна относительно этого (аффинного или проективного) преобразования.*

Напомним, что аффинные преобразования включают в себя вращение, растяжение и сжатие, параллельный перенос и их всевозможные комбинации, а к проективным преобразованиям, кроме того, относятся еще и преобразования перспективы.

Большинство из рассматриваемых классов кривых обладают свойством аффинной инвариантности. Есть среди них и проективно-инвариантные. Хотя окончательный результат и не зависит от последовательности выполнения этих двух операций – преобразования набора опорных точек и построения кривой, выбор порядка их исполнения может заметно снизить вычислительные затраты. Легко заметить, например, что поворот массива вершин вокруг оси на некоторый угол и последующее построение кривой требуют меньшего объема вычислений, чем поворот построенной кривой.

Использование рациональных кривых вносит в решение задачи приближения нужную гибкость, ибо в описании таких кривых принимают участие свободные числовые параметры. Возможность выбора позволяет учесть многие обстоятельства, неизбежно возникающие при построении кривых с предписанными свойствами. Вместе с тем рациональные кривые сохраняют многие из свойств, которыми облада-

ют соответствующие нерациональные (полиномиальные) кривые. А то обстоятельство, что рациональные кривые проективно инвариантны, является одним из наиболее привлекательных их свойств, весьма полезных при визуализации.

Очень важным является вопрос единой параметризации составной кривой. Дело в том, что связь между аналитическим описанием составной кривой и ее геометрической формой не всегда проста и очевидна; и некоторые проблемы, которые могут возникнуть при состыковке кривых, обсуждаются в 3.1.7 на нескольких типичных примерах.

Как правило, каждый раздел, посвященный определенному классу сплайновых кривых, завершается программной реализацией и конкретными примерами вычислений.

3.1. Элементарные сведения из дифференциальной геометрии кривых

3.1.1. Параметризованные кривые

Параметрически заданной пространственной кривой называется множество γ точек **M** пространства, декартовы координаты x, y и z которых определяются посредством соотношений

$$x = x(t),\quad y = y(t),\quad z = z(t),\quad a \le t \le b,$$

где $x(t)$, $y(t)$, $z(t)$ - функции, непрерывные на отрезке $[a,b]$, или *в векторно-матричной форме:*

$$\mathbf{R} = \mathbf{R}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix},\quad a \le t \le b\ .$$

Эти соотношения называют *параметрическими уравнениями кривой* γ или просто *параметризацией кривой* γ.

3.1.2. Гладкие и регулярные кривые

Пространственная кривая γ называется *C^r-гладкой относительно заданной параметризации*, если векторная функция $\mathbf{R}(t)$ является C^r-гладкой на отрезке $[a, b]$, то есть каждая из координатных функций $x(t)$, $y(t)$, $z(t)$ имеет на отрезке $[a, b]$ непрерывные производные до порядка r включительно (в точках a и b вычисляются односторонние производные, соответственно правая и левая).

Гладкая параметризация называется *регулярной*, если

$$\dot{\mathbf{R}}(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{pmatrix} \neq \mathbf{O}, \qquad a \le t \le b,$$

где точкой обозначается дифференцирование по параметру t.

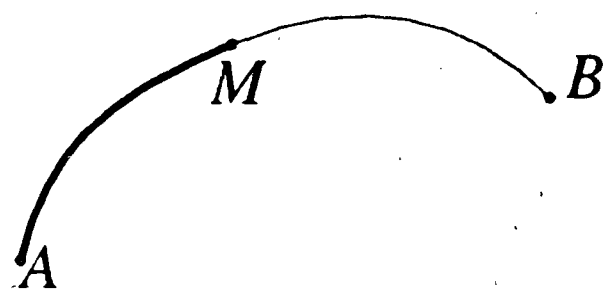


Рис. 3.2

Величина

$$\left|\dot{\mathbf{R}}(t)\right| = \sqrt{\dot{x}^2(t) + \dot{y}^2(t) + \dot{z}^2(t)}$$

называется *скоростью кривой* (относительно заданной параметризации) в точке $\boldsymbol{M} = \boldsymbol{M}(t)$, а

$$s = s(t) = \int_a^t \left| \dot{\mathbf{R}}(\xi) \right| d\xi = \int_a^t \sqrt{\dot{x}^2(\xi) + \dot{y}^2(\xi) + \dot{z}^2(\xi)}\, d\xi$$

- *длиной дуги отрезка кривой* $\cup AM$ (рис. 3.2).

3.1.3. Замена параметра

Замена параметра $\tau = \tau(t)$, где $\dot{\tau}(t) > 0$, $a \le \tau \le b$, изменяет параметрические уравнения кривой γ

$$\tilde{\mathbf{R}}(\tau) = \mathbf{R}(t(\tau)), \quad \alpha \le \tau \le \beta,$$

где $t = t(\tau)$ - функция, обратная функции $\tau = \tau(t)$, и ее скорость

$$\left| \frac{d\tilde{\mathbf{R}}}{d\tau} \right| = \left| \dot{\mathbf{R}} \right| \left| \frac{dt}{d\tau} \right|,$$

но сохраняет форму кривой и ее гладкость (последнее при условии соответствующей гладкости функции $\tau(t)$).

Параметризация, где в качестве параметра выбирается длина дуги регулярной кривой, называется *естественной параметризацией*, а сама длина дуги - *естественным параметром*. Скорость кривой относительно естественной параметризации равна единице в каждой точке,

$$\left| \mathbf{R}'(s) \right| = \sqrt{\left[x'(s)\right]^2 + \left[y'(s)\right]^2 + \left[z'(s)\right]^2} = 1$$

(здесь штрихом обозначено дифференцирование по длине дуги).

3.1.4. Трехгранник Френе

Пусть $\mathbf{R} = \mathbf{R}(t)$ - параметрическое векторное уравнение C^3-регулярной кривой γ.

Тогда

$$\mathbf{T}(t) = \frac{\dot{\mathbf{R}}(t)}{\left|\dot{\mathbf{R}}(t)\right|}$$

- *единичный вектор касательной* кривой в точке $M(t)$.

Пусть векторы $\dot{\mathbf{R}}(t)$ и $\ddot{\mathbf{R}}(t)$ неколлинеарны.

Плоскость $\mathbf{\Pi}$, проходящая через точку $M(t)$ параллельно $\dot{\mathbf{R}}(t)$ и $\ddot{\mathbf{R}}(t)$, называется *соприкасающейся плоскостью кривой γ в этой точке*, а ее нормальный вектор $\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)$ является направляющим вектором *бинормали* (рис. 3.3).

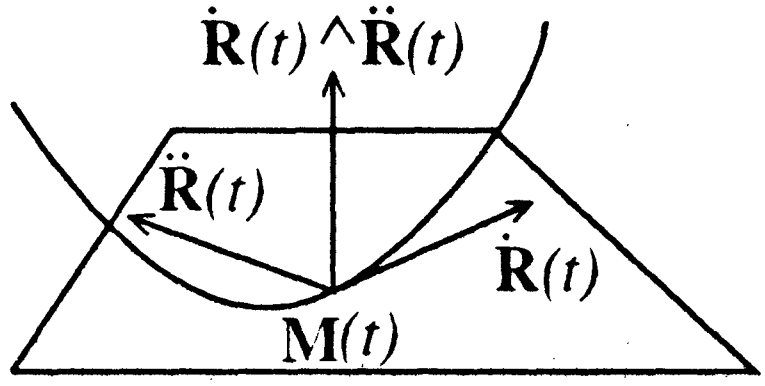


Рис. 3.3

В теории кривых важную роль играют *единичный вектор бинормали*

$$\mathbf{B}(t) = \frac{\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)}{\left|\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)\right|}$$

и *единичный вектор главной нормали*

$$\mathbf{N}(t) = \mathbf{B}(t) \times \mathbf{T}(t)$$

(он лежит в соприкасающейся плоскости).

Тройка взаимно перпендикулярных единичных векторов

$$\mathbf{T}(t),\quad \mathbf{N}(t),\quad \mathbf{B}(t)$$

называется *основным трехгранником* или *трехгранником Френе* (рис. 3.4).

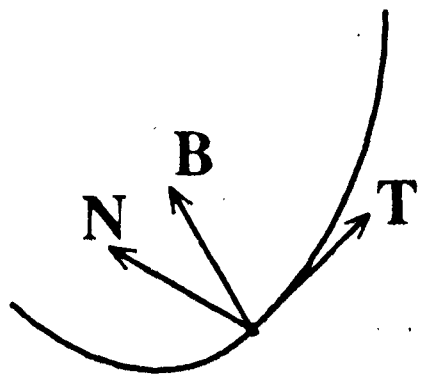


Рис. 3.4

В случае произвольной параметризации $\mathbf{R}=\mathbf{R}(t)$ векторы трехгранника Френе вычисляются по формулам

$$\mathbf{T}(t) = \frac{\dot{\mathbf{R}}(t)}{\left|\dot{\mathbf{R}}(t)\right|},\quad \mathbf{N}(t) = \frac{\left[\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)\right] \times \dot{\mathbf{R}}(t)}{\left|\left[\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)\right] \times \dot{\mathbf{R}}(t)\right|},\quad \mathbf{B}(t) = \frac{\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)}{\left|\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)\right|},$$

а в случае естественной параметризации $\mathbf{R} = \mathbf{R}(s)$ - по формулам

$$\mathbf{T}(s) = \mathbf{R}'(s),\quad \mathbf{N}(s) = \frac{\mathbf{R}''(s)}{\left|\mathbf{R}''(s)\right|},\quad \mathbf{B}(s) = \frac{\mathbf{R}'(s) \times \mathbf{R}''(s)}{\left|\mathbf{R}'(s) \times \mathbf{R}''(s)\right|}.$$

Трехгранник Френе не зависит от параметризации кривой.

Замечание

Формулы теории кривых, в которые входит естественный параметр, как правило, выглядят проще и часто более явно несут в себе отражаемый ими геометрический смысл. Однако в приложениях (в частности, при построении кривых) использование естественной параметризации весьма затруднительно, хотя бы потому, что зачастую еще нет самой кривой.

3.1.5. Кривизна и кручение кривой

Для производных векторов трехгранника Френе C^3-регулярной кривой справедлива следующая формула (*формула Серре-Френе*):

$$\begin{pmatrix} \mathbf{T}'(s) \\ \mathbf{N}'(s) \\ \mathbf{B}'(s) \end{pmatrix} = \begin{pmatrix} 0 & k_1(s) & 0 \\ -k_1(s) & 0 & k_2(s) \\ 0 & -k_2(s) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{T}(s) \\ \mathbf{N}(s) \\ \mathbf{B}(s) \end{pmatrix},$$

где $k_1(s)$ - *кривизна*, а $k_2(s)$ - *кручение кривой* в рассматриваемой точке.

Кривизну и кручение произвольной регулярной кривой можно вычислить посредством следующих формул:

$$k_1 = k_1(s) = \left| \mathbf{R}''(s) \right|, \quad k_1 = k_1(t) = \frac{\left| \dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t) \right|}{\left| \dot{\mathbf{R}}(t) \right|^3},$$

$$k_2 = k_2(s) = \frac{\left(\mathbf{R}'(s)\mathbf{R}''(s)\mathbf{R}'''(s)\right)}{\left| \mathbf{R}''(s) \right|^3}, \quad k_2 = k_2(t) = \frac{\left(\dot{\mathbf{R}}(t)\ddot{\mathbf{R}}(t)\dddot{\mathbf{R}}(t)\right)}{\left| \dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t) \right|^3}.$$

Величины k_1 и k_2 не зависят от выбора параметризации кривой γ и показывают степень ее искривленности: кривизна кривой характеризует степень отклонения кривой от прямой линии (кривизна прямой тождественно равна нулю), а кручение - от плоскости (кручение любой плоской регулярной кривой тождественно равно нулю).

Вектор $\mathbf{K}$, вычисляемый по формулам

$$\mathbf{K} = \mathbf{K}(s) = \mathbf{R}''(s), \quad \mathbf{K} = \mathbf{K}(t) = \frac{\left[\dot{\mathbf{R}}(t) \times \ddot{\mathbf{R}}(t)\right] \times \dot{\mathbf{R}}(t)}{\left| \dot{\mathbf{R}}(t) \right|^4},$$

называется *вектором кривизны кривой* γ. Этот вектор лежит в соприкасающейся плоскости кривой в точке, перпендикулярен касательной к кривой в этой точке, и его длина равна кривизне кривой в точке $M(t)$, $\left| \mathbf{K}(t) \right| = k_1(t)$.

Окружность с центром в точке

$$\mathbf{R}(t)+\frac{1}{k_1(t)}\mathbf{N}(t)=\mathbf{R}(t)+\frac{1}{k_1^2(t)}\mathbf{K}(t)$$

радиуса $1/k_1(t)$ называется *соприкасающейся окружностью кривой γ в точке $M(t)$*. Соприкасающаяся окружность лежит в соприкасающейся плоскости кривой γ, проходящей через точку $M(t)$, и имеет с кривой в этой точке касание, порядок которого не ниже второго. Центр соприкасающейся окружности называется *центром кривизны кривой*, а ее радиус - *радиусом кривизны кривой в точке $M(t)$* (рис. 3.5).

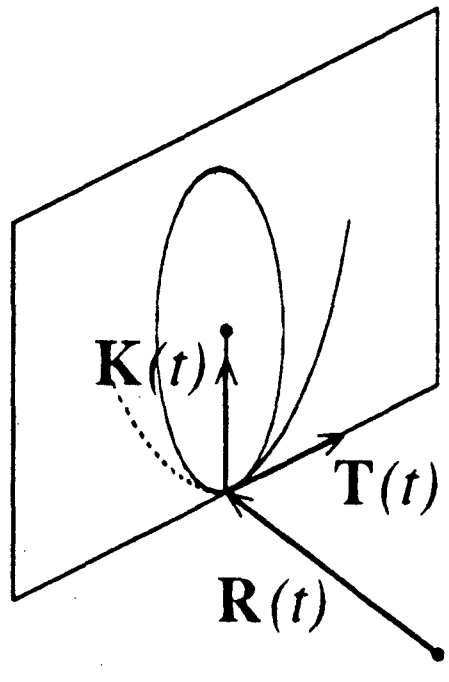


Рис. 3.5

3.1.6. Плоские кривые

А. Параметрическое задание

Запишем основные формулы для случая, когда кривая γ является плоской:

$$\mathbf{R}=\mathbf{R}(t)=\begin{pmatrix}x(t)\\ y(t)\end{pmatrix},\quad a\le t\le b\,.$$

Скорость кривой в точке $M(t)$ -

$$\left|\,\dot{\mathbf{R}}(t)\,\right|=\sqrt{\dot{x}^2(t)+\dot{y}^2(t)}\,.$$

Единичный вектор касательной -

$$\mathbf{T}(s)=\mathbf{R}'(s),\quad \mathbf{T}(t)=\frac{1}{\left|\,\dot{\mathbf{R}}(t)\,\right|}\begin{pmatrix}\dot{x}(t)\\ \dot{y}(t)\end{pmatrix}.$$

Единичный вектор нормали (рис. 3.6) -

$$\mathbf{N}(s) = \frac{\mathbf{R}''(s)}{|\mathbf{R}''(s)|},$$

$$\mathbf{N}(t) = \frac{1}{|\dot{\mathbf{R}}(t)|}\begin{pmatrix} -\dot{y}(t) \\ \dot{x}(t) \end{pmatrix}.$$

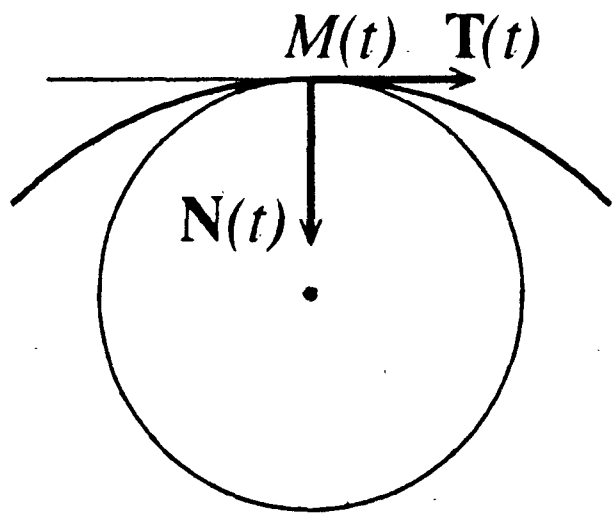


Рис. 3.6

Кривизна плоской кривой -

$$k = k(s) = x'(s)y''(s) - x''(s)y'(s), \qquad k = k(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{\left(\dot{x}^2(t) + \dot{y}^2(t)\right)^{3/2}}$$

(может менять знак).

Формулы Серре-Френе -

$$\begin{pmatrix} \mathbf{T}'(s) \\ \mathbf{N}'(s) \end{pmatrix} = \begin{pmatrix} 0 & k(s) \\ -k(s) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{T}(s) \\ \mathbf{N}(s) \end{pmatrix}.$$

Б. Неявное задание

Непустое множество γ точек $M(x, y)$ плоскости, декартовы координаты x и y которых удовлетворяют уравнению вида

$$F(x, y) = 0,$$

где $F(x, y)$ - гладкая функция своих аргументов, называется *неявно заданной кривой*, а само уравнение - ее *неявным уравнением.*

Пусть $F(x, y)$ - гладкая функция своих аргументов. Точка $M_0(x_0, y_0)$ называется *регулярной точкой неявно заданной кривой*, если выполнены следующие условия:

$$F(x_0, y_0) = 0, \quad F_x^2(x_0, y_0) + F_y^2(x_0, y_0) > 0,$$

и *особой точкой*, если

$$F(x_0, y_0) = 0, \quad F_x(x_0, y_0) = 0, \quad F_y(x_0, y_0) = 0.$$

3.1.7. Составные кривые

Часто кривую приходится строить из определенным образом подобранных частей. Для того чтобы получающаяся в результате составная кривая имела достаточно хорошие геометрические характеристики,

важна не только регулярность составляющих ее частичных кривых, но и выполнение определенных требований в стыковочных узлах.

Пусть

$$\mathbf{R} = \mathbf{R}_-(t), \quad a_- \le t \le b_-, \quad \mathbf{R} = \mathbf{R}_+(t), \quad a_+ \le t \le b_+,$$

- параметрические уравнения гладких кривых γ_- и γ_+ соответственно.

Требование 1

Для того чтобы составная кривая $\gamma_- \cup \gamma_+$ была непрерывна, необходимо, чтобы правый конец кривой γ_- совпадал с левым концом кривой γ_+:

$$\mathbf{R}_-(b_-) = \mathbf{R}_+(a_+)$$

(рис. 3.7).

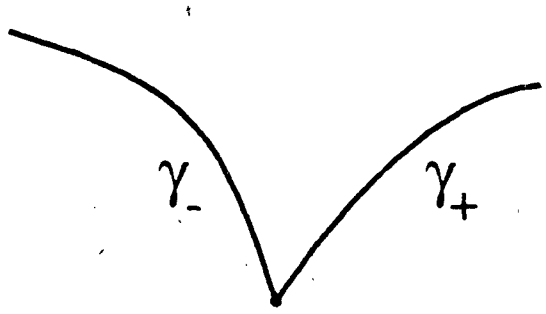


Рис. 3.7

При построении составных кривых важно помнить, что, с одной стороны, кривая как множество точек может иметь прекрасные геометрические характеристики, но описываться сравнительно плохими параметрическими уравнениями, а с другой - хорошие дифференциальные свойства координатных функций кривой не всегда приводят к регулярной кривой.

Поясним сказанное несколькими примерами.

Внимание: примеры!

Пример 1. Составная кривая $\gamma = \gamma_- \cup \gamma_+$ задана параметрическими уравнениями

$$\mathbf{R}_-(t) = \begin{pmatrix} 4t \\ 4t \end{pmatrix}, \quad 0 \le t \le \frac{1}{4},$$

$$\mathbf{R}_+(t) = \begin{pmatrix} 1+t \\ 1+t \end{pmatrix}, \quad 0 \le t \le 1$$

(рис. 3.8).

Результат: составная кривая - прямолинейный отрезок, касательные векторы в общей точке не совпадают,

$$\dot{\mathbf{R}}_-\left(\frac{1}{4}\right) = \begin{pmatrix} 4 \\ 4 \end{pmatrix}, \quad \dot{\mathbf{R}}_+(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

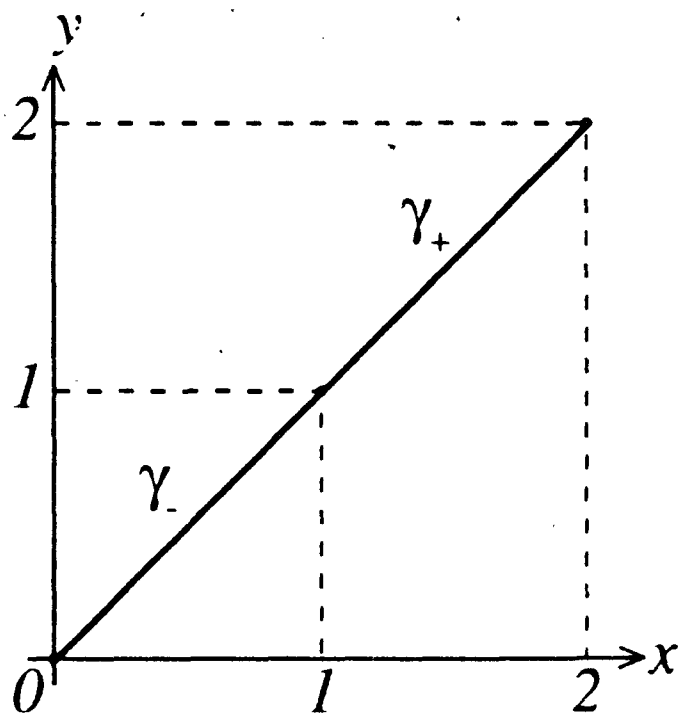


Рис. 3.8

Причина особенности: несогласованность параметризаций в точке стыковки кривых.

Требование 2

$$\dot{\mathbf{R}}_+(a_+) = \alpha \dot{\mathbf{R}}_-(b_-), \quad \alpha > 0.$$

Пример 2. Составная кривая $\gamma = \gamma_- \cup \gamma_+$ задана параметрическими уравнениями

$$\mathbf{R}_-(t) = \begin{pmatrix} -(1-t)^2 \\ (1-t)^2 \end{pmatrix}, \quad 0 \le t \le 1, \quad \mathbf{R}_+(t) = \begin{pmatrix} t^2 \\ t^2 \end{pmatrix}, \quad 0 \le t \le 1$$

(рис. 3.9).

Результат: составная кривая имеет излом, касательные векторы в общей точке равны,

$$\dot{\mathbf{R}}_-(1) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \dot{\mathbf{R}}_+(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Причина особенности: неудачная параметризация стыкующихся кривых - касательные векторы в точке стыковки равны нулю.

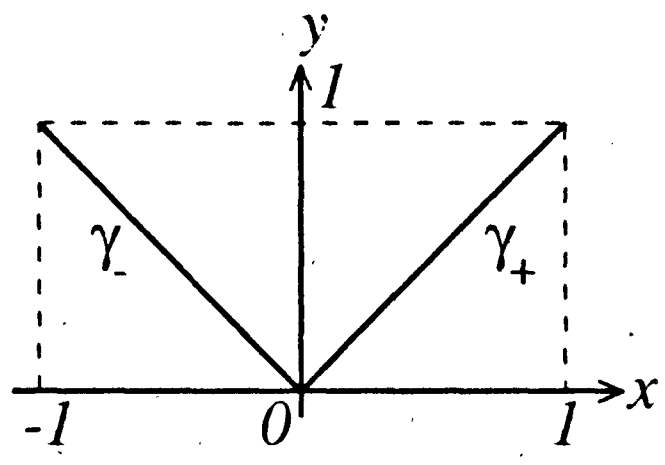


Рис. 3.9

Пример 3. Составная кривая $\gamma = \gamma_- \cup \gamma_+$ задана параметрическими уравнениями

$$\mathbf{R}_-(t) = \begin{pmatrix} \cos\left(\frac{\pi}{2}(1-t)^3\right) \\ \sin\left(\frac{\pi}{2}(1-t)^3\right) \end{pmatrix}, \quad 0 \le t \le 1,$$

$$\mathbf{R}_+(t) = \begin{pmatrix} 2 - \cos\left(\frac{\pi}{2}t^3\right) \\ -\sin\left(\frac{\pi}{2}t^3\right) \end{pmatrix}, \quad 0 \le t \le 1$$

(рис. 3.10).

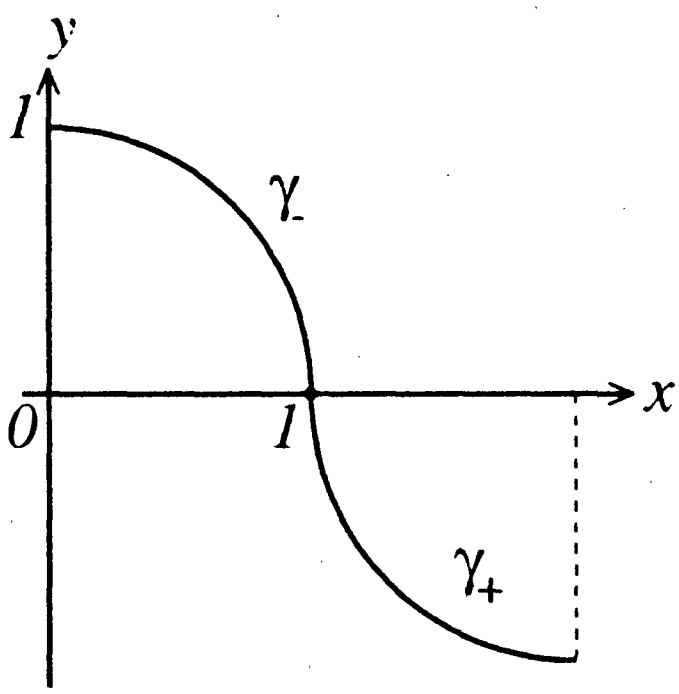


Рис. 3.10

Результат: составная кривая - круговая двузвенная ломаная, касательные векторы в точке стыка нулевые, вторые производные радиусов-векторов в точке стыка нулевые, векторы кривизны в точке стыковки различны,

$$\mathbf{K}_-(1)=\begin{pmatrix}-1\\0\end{pmatrix},\quad \mathbf{K}_+(0)=\begin{pmatrix}1\\0\end{pmatrix}.$$

Причина особенности: несогласованность параметризаций стыкуемых кривых.

Пример 4. Составная кривая $\gamma=\gamma_-\cup\gamma_+$ задана параметрическими уравнениями

$$\mathbf{R}_-(t)=\begin{pmatrix}\sin\left(\frac{\pi}{2}t^2\right)\\ \cos\left(\frac{\pi}{2}t^2\right)\end{pmatrix},\quad 0\le t\le 1,$$

$$\mathbf{R}_+(t)=\begin{pmatrix}\cos\left(\frac{\pi}{2}t^2\right)\\ -\sin\left(\frac{\pi}{2}t^2\right)\end{pmatrix},\quad 0\le t\le 1$$

(рис. 3.11).

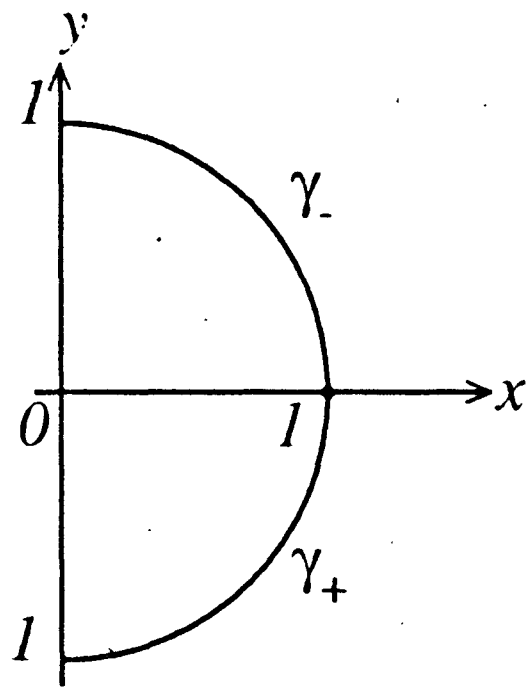


Рис. 3.11

Результат: составная кривая - полуокружность, касательные векторы в точке стыка различны,

$$\dot{\mathbf{R}}_-(1)=\begin{pmatrix}0\\ \pi\end{pmatrix},\quad \dot{\mathbf{R}}_+(0)=\begin{pmatrix}0\\0\end{pmatrix},$$

вторые производные радиусов-векторов в точке стыка различны,

$$\ddot{\mathbf{R}}_-(1)=\begin{pmatrix}-\pi^2\\ -\pi\end{pmatrix},\quad \ddot{\mathbf{R}}_+(0)=\begin{pmatrix}0\\ -\pi\end{pmatrix},$$

векторы кривизны в точке стыковки одинаковы,

$$\mathbf{K}_-(1)=\begin{pmatrix}-1\\0\end{pmatrix},\quad \mathbf{K}_+(0)=\begin{pmatrix}-1\\0\end{pmatrix}.$$

Причина особенности: один из касательных векторов в точке стыковки нулевой.

Требование 3

$$\dot{\mathbf{R}}_+(a_+)\ne\mathbf{O},\quad \dot{\mathbf{R}}_-(b_-)\ne\mathbf{O}.$$

Пример 5. Составная кривая $\gamma = \gamma_- \cup \gamma_+$ задана параметрическими уравнениями

$$\mathbf{R}_-(t) = \begin{pmatrix} 2t^2 + t - 3 \\ -t^2 + 2t \end{pmatrix}, \quad 0 \le t \le 1,$$

$$\mathbf{R}_+(t) = \begin{pmatrix} -2t^2 + 5t \\ -t^2 + 1 \end{pmatrix}, \quad 0 \le t \le 1$$

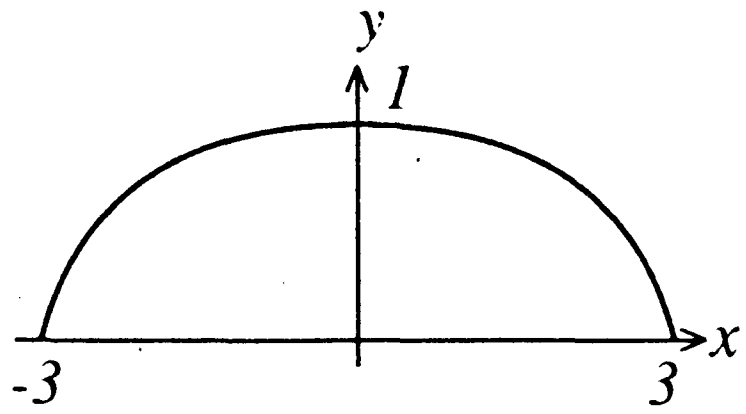


Рис. 3.12

(рис. 3.12).

Результат: касательные векторы в точке стыковки равные и не нулевые,

$$\dot{\mathbf{R}}_-(1) = \begin{pmatrix} 5 \\ 0 \end{pmatrix}, \quad \dot{\mathbf{R}}_+(0) = \begin{pmatrix} 5 \\ 0 \end{pmatrix},$$

вторые производные радиусов-векторов в точке стыковки не нулевые и неравные,

$$\ddot{\mathbf{R}}_-(1) = \begin{pmatrix} 4 \\ -2 \end{pmatrix}, \quad \ddot{\mathbf{R}}_+(0) = \begin{pmatrix} -4 \\ -2 \end{pmatrix},$$

векторы кривизны в точке стыковки равны,

$$\mathbf{K}_-(1) = \frac{1}{25}\begin{pmatrix} 0 \\ -2 \end{pmatrix}, \quad \mathbf{K}_+(0) = \frac{1}{25}\begin{pmatrix} 0 \\ -2 \end{pmatrix}.$$

Причина особенности: несогласованность параметризаций состыкованных кривых.

3.1.8. Геометрическая непрерывность

При построении составных кривых приходится сталкиваться с ситуацией, когда каждая из регулярных кривых, участвующих в процессе создания новой кривой, имеет собственную параметризацию.

Важное замечание

Случай, когда первые производные радиусов-векторов кривых в точке стыковки обращаются в нуль, из рассмотрения исключен (вследствие многообразия сложностей, которые могут встретиться).

Несогласованность параметризаций часто является причиной особенностей, возникающих в стыковочных узлах. Это - *особенности параметризации.*

Непрерывная кривая называется G^1-*непрерывной* (*геометрически непрерывной*, G^1-*кривой*), если вдоль этой кривой ее касательная изменяется непрерывно.

Любая G^1-непрерывная кривая является C^1-регулярной относительно естественной параметризации.

Кривая, изображенная на рис. 3.8 (пример 1), G^1-непрерывна.

G^1-непрерывная кривая называется G^2-*непрерывной (геометрически непрерывной,* G^2-*кривой),* если вдоль этой кривой ее вектор кривизны изменяется непрерывно.

Любая G^2-непрерывная кривая является C^2-регулярной относительно естественной параметризации.

Кривая, изображенная на рис. 3.12 (пример 5), G^2-непрерывна.

Требования к C^2-регулярным кривым

Если C^2-регулярные кривые

$$\gamma_-:\ \mathbf{R}=\mathbf{R}_-(t),\ a_-\le t\le b_-,\qquad \gamma_+:\ \mathbf{R}=\mathbf{R}_+(t),\ a_+\le t\le b_+,$$

удовлетворяют условиям:

$$\mathbf{R}_+(a_+)=\mathbf{R}_-(b_-),$$

$$\dot{\mathbf{R}}_+(a_+)=\beta_1\dot{\mathbf{R}}_-(b_-),\quad \beta_1>0,$$

$$\ddot{\mathbf{R}}_+(a_+)=\beta_1^2\ddot{\mathbf{R}}_-(b_-)+\beta_2\dot{\mathbf{R}}_-(b_-),$$

то составная кривая $\gamma=\gamma_-\cup\gamma_+$ будет G^2-кривой.

G^2-кривая - это кривая, являющаяся C^2-регулярной относительно естественной параметризации, но не обязательно C^2-регулярная относительно другой параметризации.

Если параметр кривой - длина дуги s, то в каждой точке G^2-кривой выполняются равенства

$$\mathbf{R}_+(s)=\mathbf{R}_-(s),\quad \mathbf{R}'_+(s)=\mathbf{R}'_-(s),\quad \mathbf{R}''_+(s)=\mathbf{R}''_-(s).$$

3.2. Кривые Безье

3.2.1. Параметрические уравнения кривой Безье

По заданному массиву вершин

$$\mathbf{P} = \left\{\mathbf{P}_i\left(x_i, y_i, z_i\right), \quad i = 0,\ 1,\ \ldots,\ m\right\}$$

(элементарная) кривая Безье степени m определяется при помощи векторного уравнения, имеющего следующий вид:

$$\mathbf{R}(t) = \sum_{i=0}^{m} B_i^m(t)\mathbf{P}_i, \quad 0 \le t \le 1,$$

где

$$B_i^m(t) = \binom{m}{i} t^i (1-t)^{m-i} = \frac{m!}{i!(m-i)!} t^i (1-t)^{m-i}$$

- *многочлены Бернштейна*. Здесь введено обозначение

$$\binom{m}{i} \equiv \frac{m!}{i!(m-i)!}.$$

Матричная запись параметрических уравнений, описывающих кривую Безье:

$$\begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} x_0 & \ldots & x_m \\ y_0 & \ldots & y_m \\ z_0 & \ldots & z_m \end{pmatrix} \begin{pmatrix} \mu_{00} & \ldots & \mu_{0m} \\ \ldots & \ldots & \ldots \\ \mu_{m0} & \ldots & \mu_{mm} \end{pmatrix} \begin{pmatrix} t^0 \\ \vdots \\ t^m \end{pmatrix},$$

$$0 \le t \le 1$$

где

$$\mu_{ij} = (-1)^{j-i} \binom{m}{j} \binom{j}{i}.$$

В случае, если промежуток изменения параметра произволен, $a \le t \le b$, уравнение кривой Безье имеет следующий вид:

$$\mathbf{R}(t) = \sum_{i=0}^{m} B_i^m\left(\frac{t-a}{b-a}\right)\mathbf{P}_i.$$

Важный частный случай. При $m = 3$ имеем элементарную кубическую кривую Безье, определяемую четырьмя вершинами и описываемую уравнением

$$\mathbf{R}(t) = \Big(\big((1-t)\mathbf{P}_0 + 3t\mathbf{P}_1\big)(1-t) + 3t^2\mathbf{P}_2\Big)(1-t) + t^3\mathbf{P}_3,$$

$$0 \le t \le 1.$$

или в матричной форме:

$$\mathbf{R}(t) = \mathbf{PMT}, \quad 0 \le t \le 1,$$

где

$$\mathbf{R}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} 1 & -3 & 3 & 1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{T} = \begin{pmatrix} t^0 \\ t^1 \\ t^2 \\ t^3 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \end{pmatrix}.$$

Матрица **M** называется *базисной матрицей кубической кривой Безье.*

3.2.2. Свойства кривых Безье

При описании элементарных кривых Безье в качестве функциональных весовых множителей берутся многочлены Бернштейна

$$B_i^m(t) = \binom{m}{i} t^i (1-t)^{m-i} = \frac{m!}{i!(m-i)!} t^i (1-t)^{m-i}.$$

Свойства, которыми они обладают, оказывают существенное влияние на поведение элементарных кривых Безье. Укажем некоторые из них.

Многочлены Бернштейна

1⁺ неотрицательны,

2⁺ в сумме составляют единицу:

$$\sum_{i=0}^{m} B_i^m(t) = 1,$$

3⁺ не зависят от вершин массива **P** (универсальны).

Основные свойства кривых Безье

Элементарная кривая Безье, порожденная массивом **P**:

1⁺ является гладкой кривой, в частности 1-ю и 2-ю производные радиуса-вектора **R**(*t*) можно записать так:

$$\dot{\mathbf{R}}(t) = \frac{d}{dt}\mathbf{R}(t) = m\sum_{i=0}^{m-1}(\mathbf{P}_{i+1} - \mathbf{P}_i)B_i^{m-1}(t),$$

$$\ddot{\mathbf{R}}(t) = \frac{d^2}{dt^2}\mathbf{R}(t) = m(m-1)\sum_{i=0}^{m-2}(\mathbf{P}_{i+2} - 2\mathbf{P}_{i+1} + \mathbf{P}_i)B_i^{m-2}(t);$$

2⁺ начинается в 1-й вершине $\mathbf{P}_0$ массива **P**, $\mathbf{R}(0) = \mathbf{P}_0$, касаясь отрезка $\mathbf{P}_0\mathbf{P}_1$ опорной ломаной,

$$\dot{\mathbf{R}}(0) = m(\mathbf{P}_1 - \mathbf{P}_0),$$

и заканчивается в последней его вершине $\mathbf{P}_m$, $\mathbf{R}(1) = \mathbf{P}_m$, касаясь отрезка $\mathbf{P}_{m-1}\mathbf{P}_m$ опорной ломаной,

$$\dot{\mathbf{R}}(1) = m(\mathbf{P}_m - \mathbf{P}_{m-1})$$

(рис. 3.13);

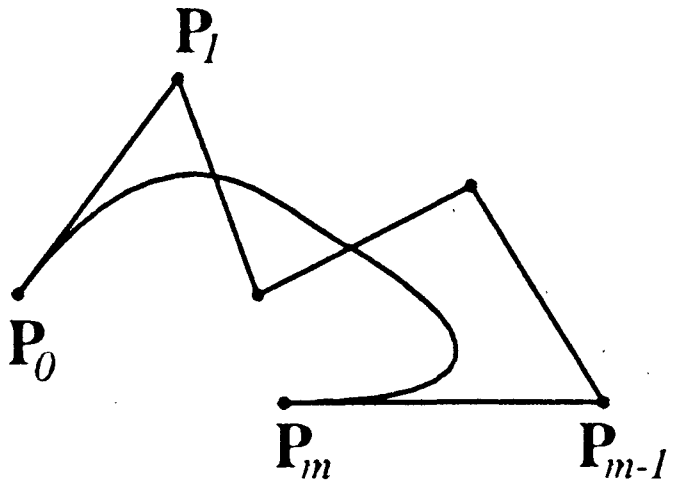


Рис. 13

3⁺ лежит в выпуклой оболочке, порожденной массивом **P** (рис. 3.14);

4⁺ симметрична - сохраняет свою форму при перемене порядка вершин массива на противоположный;

$\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{m-1}, \mathbf{P}_m \rightarrow \mathbf{P}_m, \mathbf{P}_{m-1}, \ldots, \mathbf{P}_1, \mathbf{P}_0$;

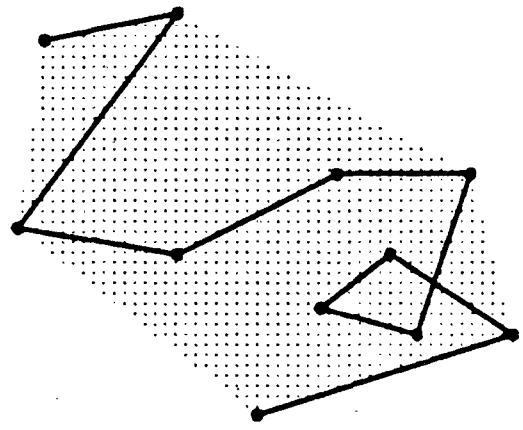
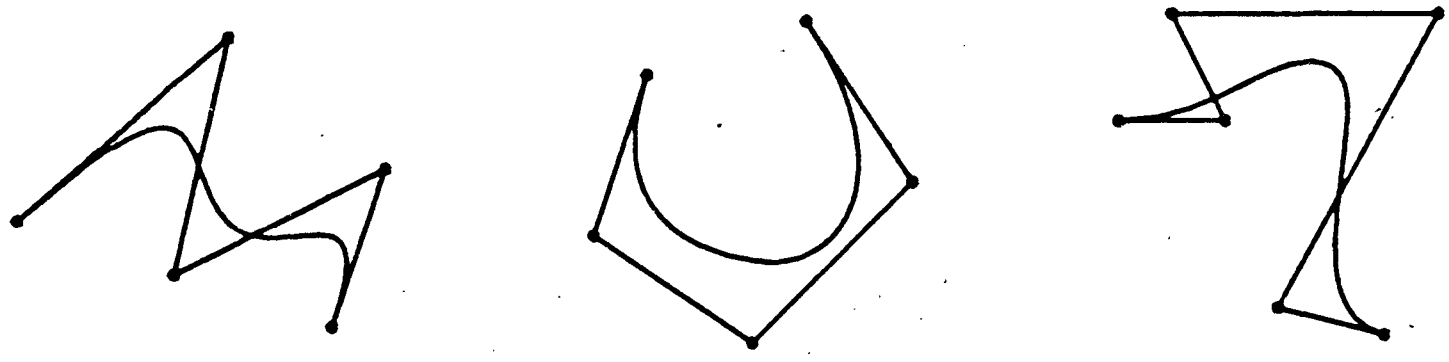


Рис. 3.14

5⁺ аффинно инвариантна;

6⁺ "повторяет" опорную ломаную (рис. 3.15) (в частности, число точек пересечения кривой Безье с произвольной прямой не больше числа точек пересечения с этой прямой опорной ломаной);



а *б* *в*

Рис. 3.15

7⁺ в случае, если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ лежат на одной прямой (коллинеарны), кривая Безье совпадает с отрезком $\mathbf{P}_0\mathbf{P}_m$;

8⁺ в случае, если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ лежат в одной плоскости (компланарны), кривая Безье лежит в этой же плоскости;

9- степень функциональных коэффициентов напрямую связана с количеством вершин в массиве (на единицу больше) и растет при его увеличении;

10- при добавлении в массив хотя бы одной вершины возникает необходимость полного пересчета параметрических уравнений элементарной кривой Безье;

11- изменение хотя бы одной вершины в массиве приводит к заметному изменению всей кривой Безье;

12- априорные сведения о расположении кривой Безье (принадлежность выпуклой оболочке заданного массива вершин) являются достаточно грубыми (на рис. 3.16 показан вид кривых Безье для массива из четырех вершин на плоскости при разном порядке их нумерации; нетрудно видеть, что, находясь в одном и том же выпуклом четырехугольнике и пытаясь повторить ход соответствующих опорных ломаных, эти кривые сильно разнятся);

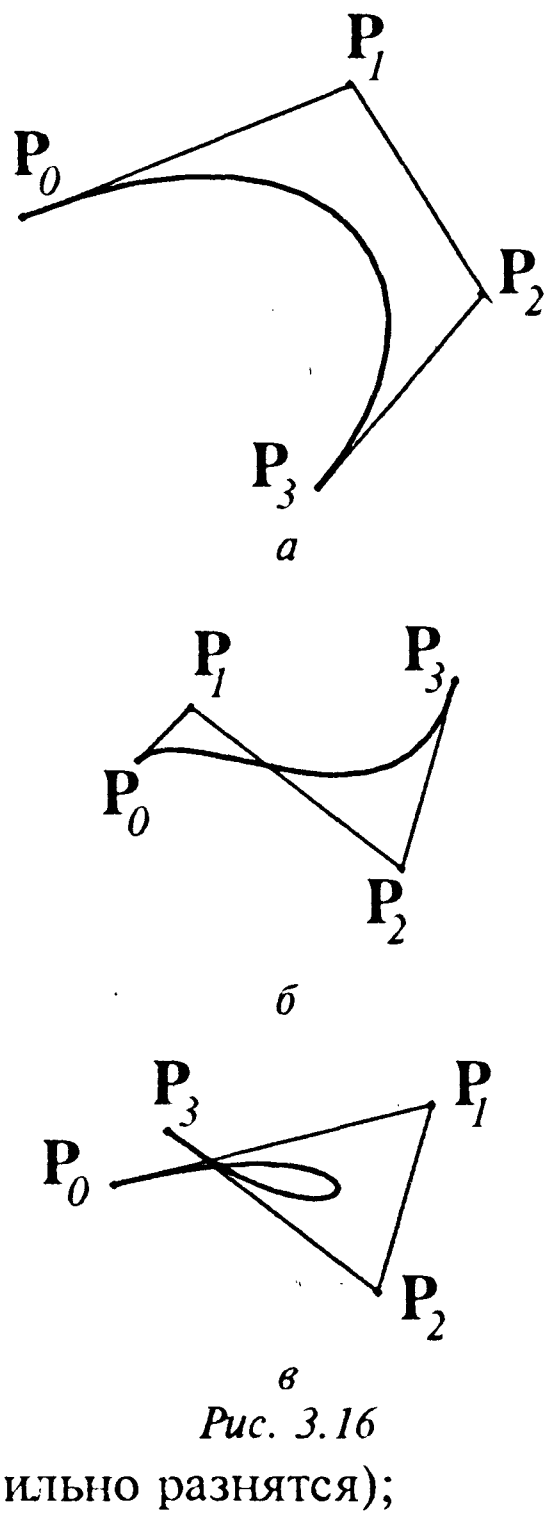


Рис. 3.16

13- в уравнении, описывающем элементарную кривую Безье, нет свободных параметров - заданный массив однозначно определяет кривую Безье, не давая возможности хоть как-то влиять на ее форму;

14- элементарная кривая Безье проективно - неинвариантна.

3.2.3. Составные кривые Безье

В этом подразделе мы подробно останавливаемся на построении составных *кубических* кривых Безье. Именно такие кривые наиболее часто используются в приложениях. Составные кривые Безье других степеней конструируются по аналогичной схеме.

Составная кубическая кривая Безье - это G^0 -непрерывная кривая γ, являющаяся объединением элементарных кубических кривых Безье $\gamma^{(1)},\ldots,\gamma^{(l)}$,

$$\gamma^{(1)} \cup \ldots \cup \gamma^{(l)}.$$

Сказанное означает, что концевая точка кривой $\gamma^{(i)}$, $i = 0, ..., l\text{-}1$, совпадает с начальной точкой кривой $\gamma^{(i+1)}$. Если кривая $\gamma^{(i)}$ задается параметрическим уравнением вида

$$\mathbf{R} = \mathbf{R}^{(i)}(t), \quad 0 \le t \le 1,$$

то это условие записывается так:

$$\mathbf{R}^{(i)}(1) = \mathbf{R}^{(i+1)}(0), \qquad i = 0, ..., l\text{-}1.$$

Условия гладкости составной кубической кривой Безье

Для того чтобы составная кубическая кривая Безье, определяемая набором $\mathbf{P}_0,...,\mathbf{P}_m$, была

1) G^1-непрерывной кривой, необходимо, чтобы тройки вершин

$$\mathbf{P}_{3i-1}, \mathbf{P}_{3i}, \mathbf{P}_{3i+1} \quad (i \ge 1)$$

были коллинеарны (лежали на одной прямой) (рис. 3.17);

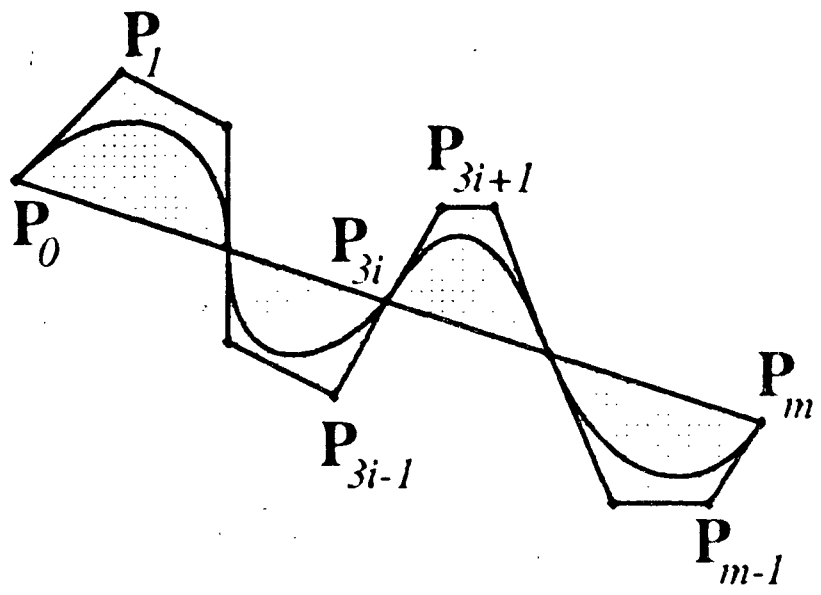


Рис. 3.17

2) G^2-непрерывной кривой, необходимо, чтобы пятерки вершин

$$\mathbf{P}_{3i-2}, \mathbf{P}_{3i-1}, \mathbf{P}_{3i}, \mathbf{P}_{3i+1}, \mathbf{P}_{3i+2} \quad (i \ge 1)$$

были компланарны (лежали в одной плоскости).

Условия замкнутости составной кривой Безье

Для того чтобы составная кубическая кривая Безье, определяемая набором вершин

$$\mathbf{P}_0, \mathbf{P}_1, ..., \mathbf{P}_{m-1}, \mathbf{P}_m,$$

была замкнутой G^1-непрерывной кривой, необходимы совпадение вершин $\mathbf{P}_0$ и $\mathbf{P}_m$ и коллинеарность тройки $\mathbf{P}_{m-1}, \mathbf{P}_m = \mathbf{P}_0, \mathbf{P}_1$ (рис. 3.18).

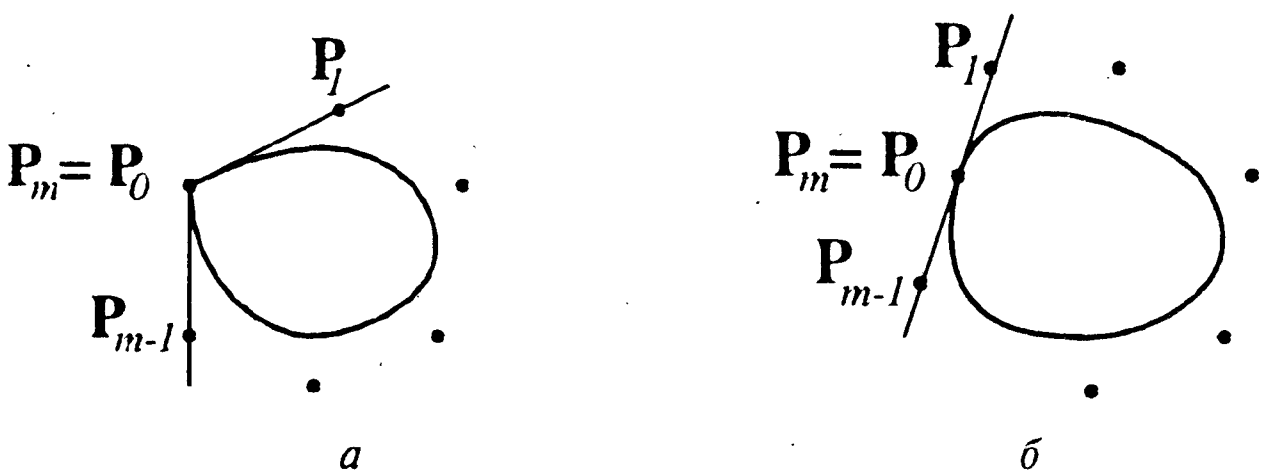


Рис. 3.18

Единая параметризация

Рассматривая составную кривую γ как целое, более естественно пользоваться единой параметризацией

$$\mathbf{R} = \mathbf{R}(t), \quad t_0 \le t \le t_l,$$

где

$$\mathbf{R} = \mathbf{R}^{(i)}(t), \quad t_{i-1} \le t \le t_i,$$

параметрическое векторное уравнение *(i)*-й элементарной кривой Безье $\gamma^{(i)}$.

Выбор промежутка $[t_0, t_l]$ и частичных сегментов

$$[t_0, t_1], \ldots, [t_{i-1}, t_i], \ldots, [t_{l-1}, t_l]$$

на нем заслуживает отдельного рассмотрения.

Простейший способ - положить $t_i = i$ - часто вполне удобен. Если же такое (равномерное) разбиение промежутка изменения параметра оказывается слишком грубым, то для выбора узлов t_i применяются другие подходы.

Опишем один из способов выбора узлов на промежутке изменения параметра для массива, в котором каждые 3 вершины

$$\mathbf{R}_{3i-1}, \mathbf{R}_{3i}, \mathbf{R}_{3i+1} \quad (i \ge 1)$$

лежат на одной прямой (коллинеарны). В этом случае составная кубическая кривая Безье будет иметь вид, указанный на рис. 3.17 (касательная изменяется вдоль кривой непрерывно). Выберем последовательность узлов t_i по следующему правилу:

$$t_0 = 0, \quad t_l = 1, \quad \frac{t_i - t_{i-1}}{t_{i-1} - t_{i-2}} = \frac{\left| \mathbf{P}_{3i+1} - \mathbf{P}_{3i} \right|}{\left| \mathbf{P}_{3i} - \mathbf{P}_{3i-1} \right|}, \qquad i = 2, \ldots, l.$$

Относительно введенного параметра t кривая γ будет C^1-гладкой, и вдоль нее касательный вектор будет изменяться непрерывно.

Замечание

Выбор параметризации не изменяет формы кривой, но может влиять на класс ее гладкости, так как наличие общей касательной в точке стыка в общем случае не влечет непрерывности касательного вектора. Класс гладкости кривой не есть чисто геометрическое свойство, а результат ее параметризации. Он зависит не только от формы кривой, но и от выбора ее параметризации.

Если в исходном массиве коллинеарных троек вершин нет, их всегда можно получить путем соответствующего расширения этого

массива, причем так, чтобы получающаяся в результате составная кривая была C^1-гладкой.

Пример. Пусть **P** - произвольный массив из шести вершин

$$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4, \mathbf{P}_5.$$

Возьмем на отрезке $\mathbf{P}_2, \mathbf{P}_3$ опорной ломаной вспомогательную вершину $\mathbf{P}_*$ (для простоты середину отрезка $\mathbf{P}_2, \mathbf{P}_3$ (рис. 19)) и построим для каждой четверки вершин $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_*$ и $\mathbf{P}_*, \mathbf{P}_3, \mathbf{P}_4, \mathbf{P}_5$ элементарные кубические кривые Безье

$$\gamma^{(1)}: \quad \mathbf{R} = \mathbf{R}^{(1)}(t), \quad 0 \le t \le 1,$$

$$\gamma^{(2)}: \quad \mathbf{R} = \mathbf{R}^{(2)}(t), \quad 1 \le t \le 2.$$

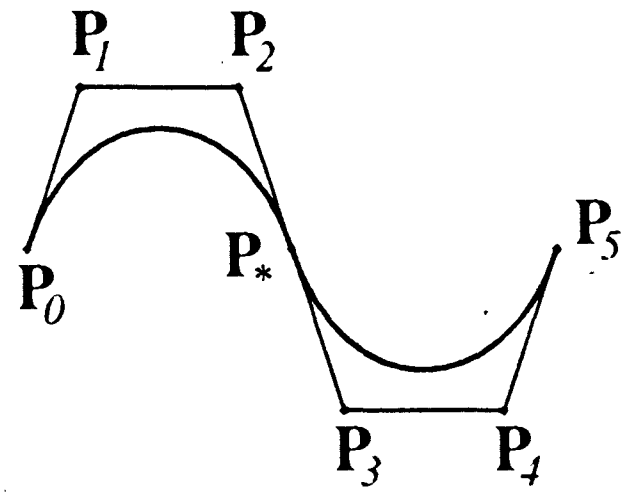


Рис. 3.19

В результате получится составная кривая

$$\gamma: \quad \mathbf{R} = \mathbf{R}(t) = \begin{cases} \mathbf{R}^{(1)}(t), & 0 \le t \le 1, \\ \mathbf{R}^{(2)}(t), & 1 \le t \le 2 \end{cases}$$

с непрерывно изменяющимся касательным вектором, но разрывной кривизной.

3.2.4. Рациональные кривые Безье

По заданному массиву вершин $\mathbf{P}_0, \ldots, \mathbf{P}_m$ *(элементарная) рациональная кривая Безье степени m* определяется уравнением следующего вида:

$$\mathbf{R}(t) = \frac{\sum_{i=0}^{m} w_i B_i^m(t) \mathbf{P}_i}{\sum_{i=0}^{m} w_i B_i^m(t)}, \quad 0 \le t \le 1,$$

где $B_i^m(t)$ - многочлены Бернштейна.

Неотрицательные числа w_i, сумма которых положительна, называются *весами.* Если все веса w_i равны между собой, получается стандартная элементарная кривая Безье m-й степени.

Свойства рациональных кривых Безье

Элементарная рациональная кривая Безье, порожденная массивом **P**:

1⁺ является гладкой кривой;

2⁺ начинается в 1-й вершине $\mathbf{P_0}$ массива **P**, $\mathbf{R}(0)=\mathbf{P_0}$, касаясь отрезка $\mathbf{P}_0\mathbf{P}_1$ опорной ломаной,

$$\dot{\mathbf{R}}(0) = m \frac{w_1}{w_0} (\mathbf{P}_1 - \mathbf{P}_0),$$

и заканчивается в последней его точке $\mathbf{P}_m$, $\mathbf{R}(1) = \mathbf{P}_m$, касаясь отрезка $\mathbf{P}_{m-1}\mathbf{P}_m$ опорной ломаной,

$$\dot{\mathbf{R}}(1) = m\frac{w_{m-1}}{w_m}(\mathbf{P}_m - \mathbf{P}_{m-1});$$

3⁺ лежит в выпуклой оболочке, порожденной массивом опорных вершин **P**;

4⁺ симметрична - при перемене порядка вершин массива на противоположный,

$$\mathbf{P}_0\mathbf{P}_1,\ldots,\mathbf{P}_{m-1}\mathbf{P}_m \rightarrow \mathbf{P}_{m-1}\mathbf{P}_m,\ldots,\mathbf{P}_1\mathbf{P}_0,$$

не изменяет своей формы;

5⁺ аффинно-инвариантна;

6⁺ "повторяет" контрольную ломаную (в частности, число точек пересечения рациональной кривой Безье с произвольной прямой не больше числа точек пересечения с этой прямой контрольной ломаной);

7⁺ в случае, если опорные вершины $\mathbf{P}_0,\ldots,\mathbf{P}_m$ лежат на одной прямой (коллинеарны), рациональная кривая Безье совпадает с отрезком $\mathbf{P}_0\mathbf{P}_m$;

8⁺ в случае, если опорные вершины $\mathbf{P}_0,\ldots,\mathbf{P}_m$ лежат в одной плоскости (компланарны), рациональная кривая Безье также лежит в этой плоскости;

9⁺ элементарная рациональная кривая Безье проективно-инвариантна;

10⁺ поведение рациональной кривой Безье определяется не только массивом вершин, но и набором свободных параметров - весов w_i, *параметров формы;* при заданном наборе вершин формой рациональной кривой Безье можно управлять, меняя весовые множители;

11⁻ степень функциональных коэффициентов напрямую связана с количеством вершин в массиве (на единицу больше) и растет при его увеличении;

12⁻ при добавлении в массив хотя бы одной вершины возникает необходимость полного пересчета параметрических уравнений элементарной кривой Безье;

13⁻ изменение хотя бы одной вершины в массиве приводит к заметному изменению всей кривой Безье.

Свойство рациональной кубической кривой Безье

Рассмотрим два набора вершин: $\mathbf{P}_0^{(1)},\mathbf{P}_1^{(1)},\mathbf{P}_2^{(1)},\mathbf{P}_3^{(1)}$ и $\mathbf{P}_0^{(2)},\mathbf{P}_1^{(2)},\mathbf{P}_2^{(2)},\mathbf{P}_3^{(2)}$.

При условии, что вершины $\mathbf{P}_3^{(1)}$ и $\mathbf{P}_0^{(2)}$ совпадают, рациональные кубические кривые Безье $\gamma^{(1)}$ и $\gamma^{(2)}$, порожденные этими наборами, имеют общую точку.

Пусть $\gamma = \gamma^{(1)} \cup \gamma^{(2)}$ - полученная составная кривая.

При условии, что вершины

$$\mathbf{P}_2^{(1)}, \mathbf{P}_3^{(1)} = \mathbf{P}_0^{(2)}, \mathbf{P}_1^{(2)}$$

коллинеарны, подбором весов

$$w^{(1)} = \left(w_0^{(1)}, w_1^{(1)}, w_2^{(1)}, w_0^{(1)}\right), \qquad w^{(2)} = \left(w_0^{(2)}, w_1^{(2)}, w_2^{(2)}, w_3^{(2)}\right)$$

можно добиться непрерывного изменения касательного вектора вдоль кривой γ. Достаточно положить

$$\frac{w_2^{(1)}}{w_3^{(1)}} \left| \mathbf{P}_3^{(1)} - \mathbf{P}_2^{(1)} \right| = \frac{w_1^{(2)}}{w_0^{(2)}} \left| \mathbf{P}_1^{(2)} - \mathbf{P}_0^{(2)} \right|.$$

При условии, что вершины

$$\mathbf{P}_1^{(1)}, \mathbf{P}_2^{(1)}, \mathbf{P}_3^{(1)} = \mathbf{P}_0^{(2)}, \mathbf{P}_1^{(2)}, \mathbf{P}_2^{(2)}$$

компланарны (лежат в одной плоскости), то есть векторы

$$\left(\mathbf{P}_2^{(1)} - \mathbf{P}_1^{(1)}\right) \times \left(\mathbf{P}_3^{(1)} - \mathbf{P}_2^{(1)}\right), \quad \left(\mathbf{P}_1^{(2)} - \mathbf{P}_0^{(2)}\right) \times \left(\mathbf{P}_2^{(2)} - \mathbf{P}_1^{(2)}\right)$$

коллинеарны, подбором весов $w^{(1)}$ и $w^{(2)}$ можно добиться непрерывности вектора кривизны вдоль кривой γ. Их нужно взять так, чтобы выполнялось равенство

$$\frac{H^{(1)}}{\left| \mathbf{P}_3^{(1)} - \mathbf{P}_2^{(1)} \right|^3} = \frac{H^{(2)}}{\left| \mathbf{P}_1^{(2)} - \mathbf{P}_0^{(2)} \right|^3},$$

где

$$H^{(i)} = \frac{w_{2-i}^{(i)} w_{4-i}^{(i)}}{\left(w_{3-i}^{(i)}\right)^2} \left| \left(\mathbf{P}_{3-i}^{(i)} - \mathbf{P}_{2-i}^{(i)}\right) \times \left(\mathbf{P}_{4-i}^{(i)} - \mathbf{P}_{3-i}^{(i)}\right) \right|, \qquad i = 1, 2.$$

Это свойство рациональных кубических кривых Безье позволяет поместить элементарную рациональную кубическую кривую Безье в разрыв между любыми двумя заданными (уже построенными) C^2-регулярными кривыми (в частности, элементарными рациональными кубическими кривыми Безье) так, что получаемая в результате составная кривая будет иметь непрерывный касательный вектор и непрерывный вектор кривизны.

Задача. По заданным набору из четырех вершин $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ и двум неотрицательным числам k_0 и k_3 найти веса w_0, w_1, w_2, w_3 так, чтобы значения кривизны рациональной кубической кривой Безье, порождаемой заданным массивом, в концах опорной ломаной $\mathbf{P}_0\ \mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3$ совпадали с заданными числами (с k_0 в вершине $\mathbf{P}_0$ и с k_3 в вершине $\mathbf{P}_3$).

Решением задачи является набор

$$w = (w_0, w_1, w_2, w_3),$$

где

$$w_0 = 1, w_1 = \frac{4}{3}\left(\frac{c_0^2 c_3}{k_0^2 k_3}\right)^{1/3}, \quad w_2 = \frac{4}{3}\left(\frac{c_0 c_3^2}{k_0 k_3^2}\right)^{1/3}, \quad w_3 = 1,$$

$$c_0 = \frac{\left|(\mathbf{P}_1 - \mathbf{P}_0) \times (\mathbf{P}_2 - \mathbf{P}_0)\right|}{2\left|\mathbf{P}_1 - \mathbf{P}_0\right|^3}, \quad c_3 = \frac{\left|(\mathbf{P}_2 - \mathbf{P}_1) \times (\mathbf{P}_3 - \mathbf{P}_1)\right|}{2\left|\mathbf{P}_3 - \mathbf{P}_2\right|^3}.$$

Замечание

Для планарного набора $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, опорная ломаная которого имеет S-форму (рис. 3.20), числа k_0 и k_3 должны иметь разные знаки (кривизна плоской кривой Безье меняет знак).

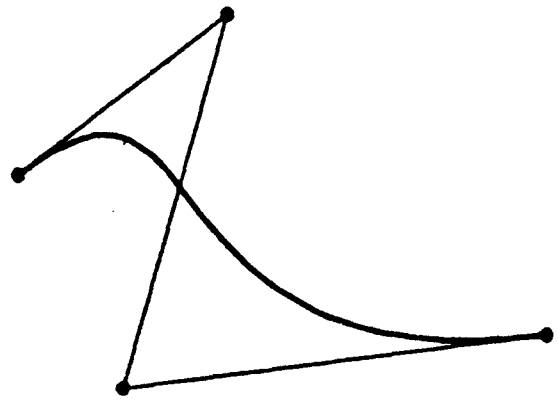


Рис. 3.20

3.2.5. Программная реализация алгоритма

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
bezier( p, segments, div, output );
```

где

```
Flt p[] :in  - опорная ломаная кривой Безье, ее элементы
               p[0]... p[m-1],  где m = 3*segments+1
int segments :in  - число звеньев кривой Безье,
int div :in  - число звеньев в аппроксимирующей ломаной,
Flt output[] :out - выходной массив с элементами
               output[0]...output[k-1], где k=segments*div+1.
```

Приведем полностью текст этой программы.

```
Flt bezier_3( Flt p[4], Flt t )
{
    Flt s = 1 - t ;
```

```
    Flt t2 = t*t ;
    Flt t3 = t2 * t ;
    return ((s*p[0] + 3*t*p[1])*s + 3*t2*p[2])*s + t3*p[3] ;
}
void bezier( Flt p[], int segments, int div, Flt output[] )
{
    Flt t, dt = 1.0 / div ;
    int s, d, first = 0 ;

    for( s = 0; s < segments; s ++ )
    {
        t = 0 ;
        for( d = 0; d <= div; d ++ )
        {
            output[ first + d ] = bezier_3( &p[ s*3 ], t ) ;
            t += dt ;
        }
        first += div ;
    }
}
```

Приведем пример использования функции для построения плоской сплайн-кривой.

```
#include "lib.h"
/* Описание вызываемой функции n */
void bezier( Flt p[], int segments, int div, Flt output[] );
/* число точек опорной ломаной  */
/* M должно быть числом вида k*3+1 */
#define  K    ( 2 )
#define  M    (K*3+1)
#define DIV   (10)
/* число точек аппроксимирующей ломаной */
#define  N    (K*DIV+1)
/* Определение опорной ломаной */
Flt xp[M] = {0.295, 0.705, 0.266, 0.206, 0.172, 0.508, 0.608};
Flt yp[M] = {0.154, 0.283, 0.413, 0.580, 0.816, 0.745, 0.515};
/* Описание выходного массива */
Flt ox[N], oy[N] ;
void main( void )
{
    /* вычисление параметров сплайна  */
    bezier( xp, K, DIV, ox ) ;
    bezier( yp, K, DIV, oy ) ;
    /* инициализация графической моды */
    SetVideoMode() ;
    /* отображение на дисплее опорной ломаной */
    DrawPolygon( xp, yp, M, 1 ) ;
    /* отображение на дисплее построенной сплайновой кривой */
    DrawPolygon( ox, oy, N, 2 ) ;
}
```

Текст программы bezier находится в файле bezier.c в поддиректории CURVES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

3.3. *B*-сплайновые кривые

3.3.1. Параметрические уравнения элементарной кубической *B*-сплайновой кривой

По заданному массиву

$$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$$

(элементарная) кубическая B-сплайновая кривая определяется при помощи векторного уравнения, имеющего следующий вид:

$$\mathbf{R}(t) = \frac{(1-t)^3}{6}\mathbf{P}_0 + \frac{3t^3 - 6t^2 + 4}{6}\mathbf{P}_1 + \frac{-3t^3 + 3t^2 + 3t + 1}{6}\mathbf{P}_2 + \frac{t^3}{6}\mathbf{P}_3,$$

$$0 \le t \le 1.$$

Матричная запись параметрических уравнений, описывающих элементарную кубическую *B*-сплайновую кривую,

$$\mathbf{R}(t) = \mathbf{PMT}, \quad 0 \le t \le 1,$$

где

$$\mathbf{R}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, \quad \mathbf{M} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{T} = \begin{pmatrix} t^0 \\ t^1 \\ t^2 \\ t^3 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \end{pmatrix}.$$

Матрица **M** называется *базисной матрицей B-сплайновой кривой.*

Свойства элементарных кубических *B*-сплайновых кривых

Свойства функциональных весовых множителей

$$n_0(t) = \frac{(1-t)^3}{6}, \quad n_1(t) = \frac{3t^3 - 6t^2 + 4}{6},$$

$$n_2(t) = \frac{-3t^3 + 3t^2 + 3t + 1}{6}, \quad n_3(t) = \frac{t^3}{6}$$

оказывают существенное влияние на поведение элементарной кубической *B*-сплайновой кривой. Укажем некоторые из них.

Функциональные коэффициенты $n_i(t)$

1⁺ неотрицательны,

2⁺ в сумме составляют единицу,

3⁺ не зависят от точек массива $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ (универсальны).

Элементарная кубическая *B*-сплайновая кривая

1⁺ лежит в выпуклой оболочке, порожденной вершинами $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ опорной ломаной, и, как правило, не проходит ни через одну из них;

2⁺ касательная в концевой точке

$$\frac{1}{6}(\mathbf{P}_0 + 4\mathbf{P}_1 + \mathbf{P}_2)$$

параллельна отрезку $\mathbf{P}_0\mathbf{P}_2$, а в концевой точке

$$\frac{1}{6}(\mathbf{P}_1 + 4\mathbf{P}_2 + \mathbf{P}_3)$$

- отрезку $\mathbf{P}_1\mathbf{P}_3$ (рис. 3.21).

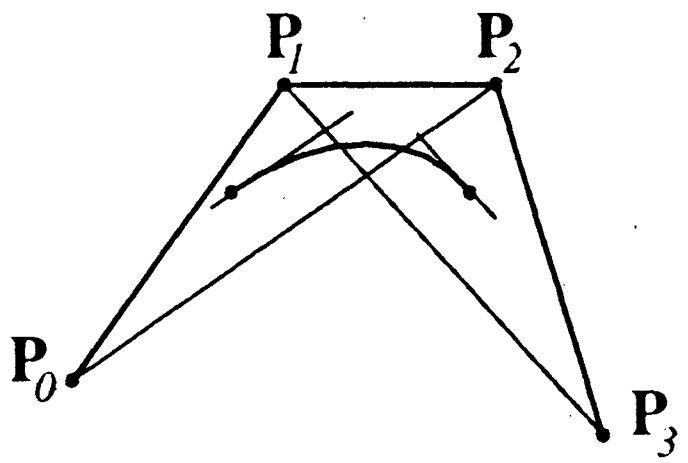


Рис. 3.21

3.3.2. Составные кубические *B*-сплайновые кривые

(Составной) кубической B-сплайновой кривой, определяемой массивом

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 3,$$

называется кривая γ, которую можно представить в виде объединения элементарных кубических *B*-сплайновых кривых $\gamma^{(1)}, \ldots, \gamma^{(m-2)}$,

$$\gamma = \gamma^{(1)} \cup \ldots \cup \gamma^{(m-2)};$$

(i)-я кривая $\gamma^{(i)}$ описывается параметрическим уравнением следующего вида:

$$\mathbf{R}^{(i)}(t) = (\mathbf{P}_{i-1} \quad \mathbf{P}_i \quad \mathbf{P}_{i+1} \quad \mathbf{P}_{i-2})\mathbf{MT},$$
$$0 \leq t \leq 1, \quad i = 1, \ldots, m-2,$$

где **M** - базисная матрица кубической *B*-сплайновой кривой.

Единая параметризация

Рассматривая составную кривую γ как целое, более естественно пользоваться единой параметризацией.

Наиболее простой является параметризация с равноотстоящими целочисленными узлами. Для массива из $m + 1$ опорных вершин составная B-сплайновая кривая строится из $m - 2$ элементарных фрагментов. Если каждый из них определен на единичном отрезке, то длина общего промежутка изменения параметра должна быть равной $m - 2$. Взяв 0 за начальную точку, получаем отрезок $[0, m-2]$. В этом случае узлы параметризации определяются по формуле

$$t_1 = 0, \quad t_{i+1} = t_i + 1, \quad i = 1, \ldots, m - 3.$$

Описанный выбор отрезка параметризации позволяет записать уравнение составной кубической B-сплайновой кривой γ следующим образом:

$$\mathbf{R} = \mathbf{R}(t), \quad t_1 \le t \le t_{m-2},$$

где

$$\mathbf{R} = \mathbf{R}^{(i)}(t) = (\mathbf{P}_{i-1} \quad \mathbf{P}_i \quad \mathbf{P}_{i+1} \quad \mathbf{P}_{i+2})\mathbf{M}\begin{pmatrix} (t-t_i)^0 \\ (t-t_i)^1 \\ (t-t_i)^2 \\ (t-t_i)^3 \end{pmatrix},$$

$$t_1 \le t \le t_{m-2}, \quad i = 1, \ldots, m-3,$$

- параметрическое векторное уравнение (i)-й элементарной кубической B-сплайновой кривой $\gamma^{(i)}$.

Свойства составной кубической B-сплайновой кривой

Составная кубическая B-сплайновая кривая, порожденная массивом $\mathbf{P}_0, \ldots, \mathbf{P}_m$, $m \ge 3$:

1$^+$ является C^2-гладкой кривой в промежутке $[0, t_{m-2}]$ $(t_{i+1} = t_i + 1, \; t_1 = 0)$, в точке стыка элементарных кривых $\gamma^{(i)}$ и $\gamma^{(i+1)}$ выполняются равенства

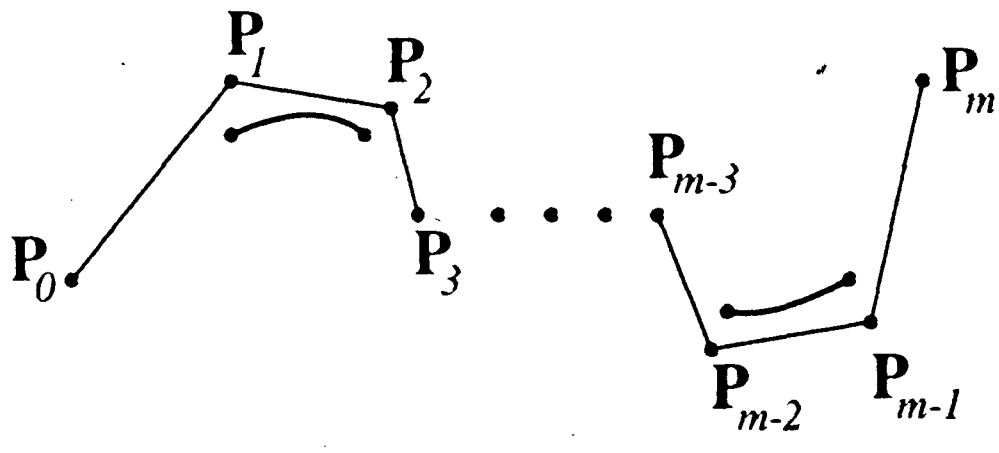


Рис. 3.22

$$\mathbf{R}^{(i)}(t_{i+1}) = \mathbf{R}^{(i+1)}(t_{i+1}) = \frac{1}{6}(\mathbf{P}_i + 4\mathbf{P}_{i+1} + \mathbf{P}_{i+2}),$$

$$\dot{\mathbf{R}}^{(i)}(t_{i+1}) = \dot{\mathbf{R}}^{(i+1)}(t_{i+1}) = \frac{1}{2}(-\mathbf{P}_i + \mathbf{P}_{i+2}),$$

$$\ddot{\mathbf{R}}^{(i)}(t_{i+1}) = \ddot{\mathbf{R}}^{(i+1)}(t_{i+1}) = \mathbf{P}_i - 2\mathbf{P}_{i+1} + \mathbf{P}_{i+2}$$

(рис. 3.22);

2+ как правило, не проходит ни через одну точку заданного массива,

3+ лежит в объединении m-2 выпуклых оболочек, порожденных четверками вершин

$$\mathbf{P}_{i-1}, \mathbf{P}_i, \mathbf{P}_{i+1}, \mathbf{P}_{i+2}, \quad i = 1, \ldots, m\text{-}2$$

(рис. 3.23);

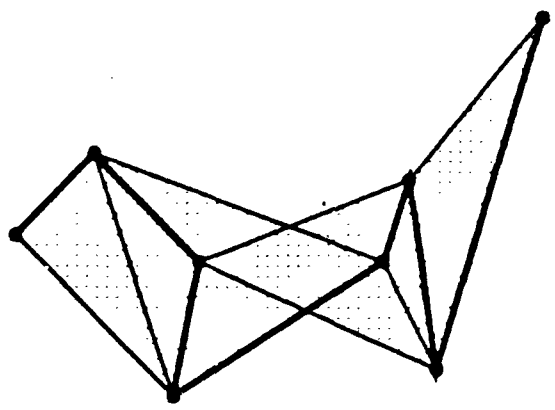


Рис. 3.23

4+ "повторяет" опорную ломаную (рис. 3.24) (в частности, число точек пересечения составной кубической B-сплайновой кривой с произвольной прямой не больше числа точек пересечения опорной ломаной с этой прямой);

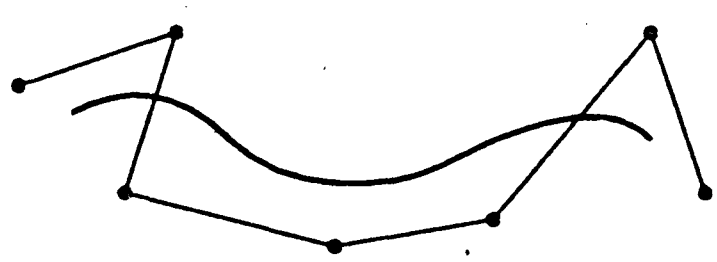


Рис. 3.24

5+ если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ массива лежат на одной прямой, то составная кубическая B-сплайновая кривая также лежит на этой прямой (между вершинами $\mathbf{P}_0$ и $\mathbf{P}_m$);

6+ если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ массива лежат в одной плоскости, то составная кубическая B-сплайновая кривая также лежит в этой плоскости;

7+ изменение одной вершины в массиве приводит к изменению только части кривой: при изменении вершины $\mathbf{P}_i$ нужно пересчитать параметрические уравнения только четырех элементарных кривых: $\gamma^{(i-2)}$, $\gamma^{(i-1)}$, $\gamma^{(i)}$, $\gamma^{(i+1)}$ (рис. 3.25);

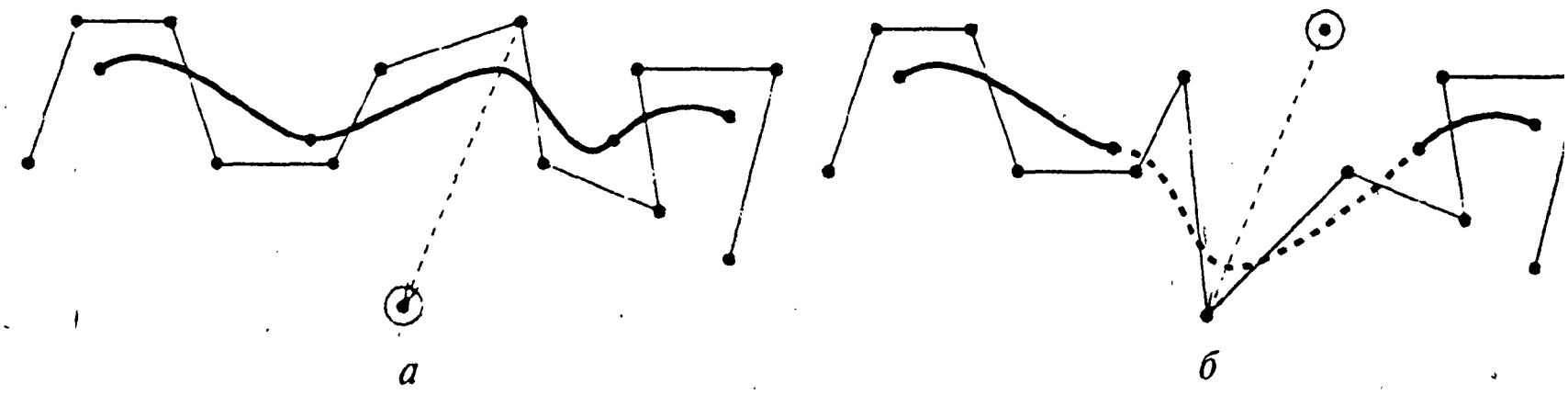


Рис. 3.25

8⁺ при добавлении в массив одной вершины возникает необходимость пересчета параметрических уравнений только четырех элементарных кривых, в формировании которых участвует эта вершина;

9⁺ составная кубическая *B*-сплайновая кривая аффинно-инвариантна;

10⁻ кубическая *B*-сплайновая кривая проективно-неинвариантна;

11⁻ в уравнениях, описывающих составную кубическую *B*-сплайновую кривую, нет свободных параметров - заданный массив однозначно определяет составную кубическую *B*-сплайновую кривую, не давая возможности изменять ее форму.

Замечание

На взаимное расположение вершин в массиве не накладывается никаких ограничений: они могут и совпадать. Однако следует иметь в виду, что в подобных случаях кривая может потерять свою регулярность. Впрочем, если номера совпадающих вершин сильно разнятся, то никакой потери регулярности не происходит.

Случай, когда совпадают две или три первые (последние) вершины, рассматривается ниже.

3.3.3. Кратные и воображаемые вершины

Составная кубическая *B*-сплайновая кривая, как правило, не проходит ни через одну вершину определяющего ее массива. Однако расположение ее начальной и конечной точек всегда известно: начальная точка $\mathbf{R}^{(1)}(0)$ составной кривой γ лежит в треугольнике $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2$, а конечная $\mathbf{R}^{(m)}(1)$ - в треугольнике $\mathbf{P}_{m-2}\mathbf{P}_{m-1}\mathbf{P}_m$. Подбором вспомогательных вершин и построением дополнительных элементарных кривых можно добиться того, чтобы начальная точка новой составной кривой выходила на отрезок $\mathbf{P}_0\mathbf{P}_1$, располагалась ближе к вершине $\mathbf{P}_0$ и даже совпадала с ней. Аналогичных результатов можно добиться и для конечной точки. Обычно это проводится путем использования *кратных* или *воображаемых* вершин.

А. Двойные вершины

Положим $\mathbf{P}_{-1} = \mathbf{P}_0$ и $\mathbf{P}_{m+1} = \mathbf{P}_m$ и построим две новые элементарные кривые $\gamma^{(0)}$ и $\gamma^{(m-1)}$, задав их параметрическими уравнениями следующего вида:

$$\mathbf{R}^{(0)}(t) = \big(n_0(t) + n_1(t)\big)\mathbf{P}_0 + n_2(t)\mathbf{P}_1 + n_3(t)\mathbf{P}_2,$$

$$\mathbf{R}^{(m-1)}(t) = n_0(t)\mathbf{P}_{m-2} + n_1(t)\mathbf{P}_{m-1} + \big(n_2(t) + n_3(t)\big)\mathbf{P}_m,$$

где $0 \le t \le 1$.

С учетом кривых $\gamma^{(0)}$ и $\gamma^{(m-1)}$ новая составная кубическая *B*-сплайновая кривая

$\gamma^* = \gamma^{(0)} \cup \gamma^{(1)} \cup \ldots \cup \gamma^{(m-2)} \cup \gamma^{(m-1)}$

будет начинаться в точке

$$\mathbf{R}^{(0)}(0) = \frac{5}{6}\mathbf{P}_0 + \frac{1}{6}\mathbf{P}_1,$$

касаясь отрезка $\mathbf{P}_0\mathbf{P}_1$,

$$\dot{\mathbf{R}}^{(0)}(0) = \frac{1}{2}(\mathbf{P}_1 - \mathbf{P}_0),$$

и заканчиваться в точке

$$\mathbf{R}^{(m-1)}(1) = \frac{1}{6}\mathbf{P}_{m-1} + \frac{5}{6}\mathbf{P}_m,$$

касаясь отрезка $\mathbf{P}_{m-1}\mathbf{P}_m$,

$$\dot{\mathbf{R}}^{(m-1)}(1) = \frac{1}{2}(\mathbf{P}_m - \mathbf{P}_{m-1})$$

(рис. 3.26, а). Кроме того, кривая γ^* будет иметь в двух этих точках нулевую кривизну.

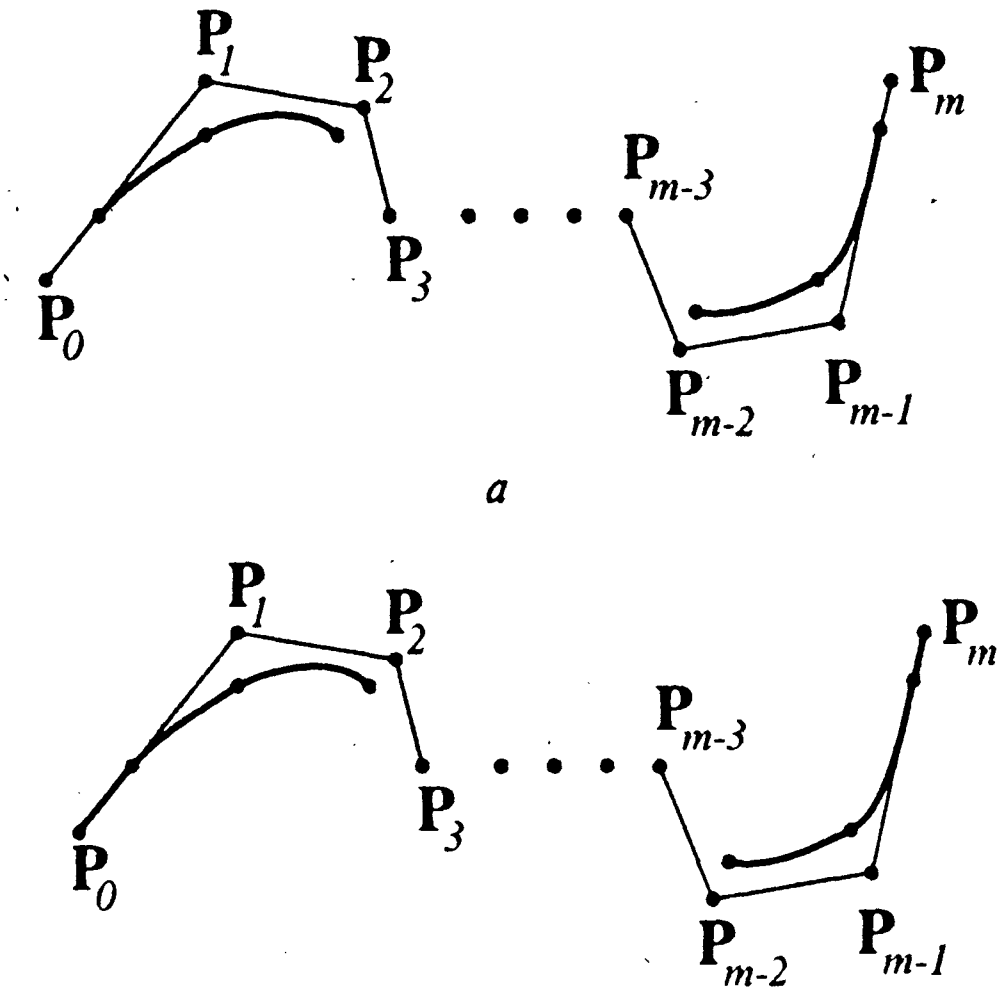


Рис. 3.26

Б. Тройные вершины

Положим $\mathbf{P}_{-2} = \mathbf{P}_{-1} = \mathbf{P}_0$ и $\mathbf{P}_{m+2} = \mathbf{P}_{m+1} = \mathbf{P}_m$ и возьмем в качестве двух новых элементарных кривых $\gamma^{(-1)}$ и $\gamma^{(m)}$ прямолинейные отрезки

$$\mathbf{R}^{(-1)}(t) = \left(1 - \frac{t^3}{6}\right)\mathbf{P}_0 + \frac{t^3}{6}\mathbf{P}_1,$$

$$\mathbf{R}^{(m)}(t) = \frac{(1-t)^3}{6}\mathbf{P}_{m-1} + \left(1 - \frac{(1-t)^3}{6}\right)\mathbf{P}_m,$$

где $0 \le t \le 1$ (рис. 3.26, б).

С учетом кривых $\gamma^{(0)}$ и $\gamma^{(m-1)}$ новая составная *B*-сплайновая кривая

$$\gamma^{**} = \gamma^{(-1)} \cup \gamma^{(0)} \cup \gamma^{(1)} \cup \ldots \cup \gamma^{(m-2)} \cup \gamma^{(m-1)} \cup \gamma^{(m)}$$

будет начинаться в вершине $\mathbf{P}_0$ и заканчиваться в вершине $\mathbf{P}_m$.

В. Воображаемые вершины

Подбором дополнительных вершин $\mathbf{P}_{-1}$ и $\mathbf{P}_{m+1}$ к массиву

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \ge 3,$$

можно добиться выполнения различных условий на концах составной кривой.

Например, составная *B*-сплайновая кривая, построенная по новому массиву

$$\mathbf{P}_{-1}, \mathbf{P}_0, \ldots, \mathbf{P}_m, \mathbf{P}_{m+1},$$

где

$$\mathbf{P}_{-1} = (\mathbf{P}_0 - \mathbf{P}_1) + \mathbf{P}_0, \quad \mathbf{P}_{m+1} = (\mathbf{P}_m - \mathbf{P}_{m-1}) + \mathbf{P}_m,$$

будет начинаться в вершине $\mathbf{P}_0$, касаясь отрезка $\mathbf{P}_0\mathbf{P}_1$,

$$\dot{\mathbf{R}}^{(0)}(0) = \mathbf{P}_1 - \mathbf{P}_0,$$

и заканчиваться в вершине $\mathbf{P}_m$, касаясь отрезка $\mathbf{P}_{m-1}\mathbf{P}_m$,

$$\dot{\mathbf{R}}^{(m-1)}(1) = \mathbf{P}_m - \mathbf{P}_{m-1}$$

(рис. 3.27). Кривизны новой кривой в точках $\mathbf{P}_0$ и $\mathbf{P}_m$, вообще говоря, отличны от нуля.

Замечание

Дополнительные вершины $\mathbf{P}_{-1}$ и $\mathbf{P}_{m+1}$ можно выбрать так, чтобы в концах новой составной кривой 1-е или 2-е производные радиусов-векторов

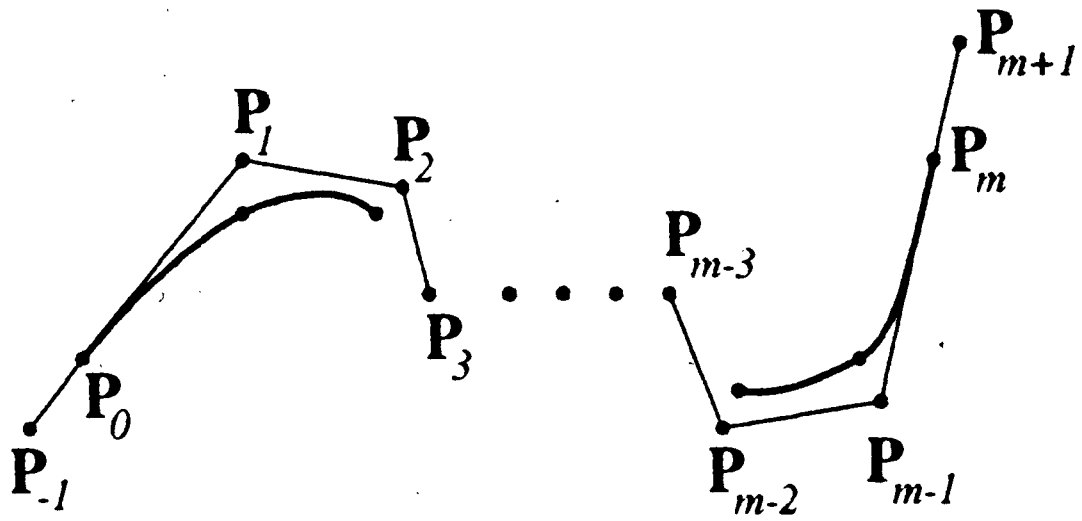


Рис. 3.27

$\mathbf{R}^{(0)}(t)$ *и* $\mathbf{R}^{(m-1)}(t)$ *кривых* $\gamma^{(0)}$ *и* $\gamma^{(m-1)}$ *совпадали с заданными значениями (соответственно при* $t = 0$ *и* $t = 1$*).*

Построение замкнутой кривой

Чтобы по заданному массиву

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 3,$$

построить C^2-гладкую замкнутую кривую, достаточно выбрать дополнительные вершины $\mathbf{P}_{m+1}, \mathbf{P}_{m+2}, \mathbf{P}_{m+3}$ по правилу

$$\mathbf{P}_{m+1} = \mathbf{P}_0, \mathbf{P}_{m+2} = \mathbf{P}_1, \mathbf{P}_{m+3} = \mathbf{P}_2$$

и рассмотреть массив

$$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_m, \mathbf{P}_{m+1} = \mathbf{P}_0, \mathbf{P}_{m+2} = \mathbf{P}_1, \mathbf{P}_{m+3} = \mathbf{P}_2$$

(при условии, что $t_{i+1} = t_i + 1$, $t_1 = 0$).

Неравномерное расположение узлов

Перейдем к случаю, когда узлы t_i расположены на отрезке изменения параметра t неравномерно,

$$t_i - t_{i-1} \neq t_{i+1} - t_i.$$

По заданному массиву

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 3,$$

составная B-сплайновая кривая определяется подобно тому, как это делается при равномерном разбиении отрезка изменения параметра:

$$\mathbf{R}_i(t) = N_{i-1}^3(t)\mathbf{P}_{i-1} + N_i^3(t)\mathbf{P}_i + N_{i+1}^3(t)\mathbf{P}_{i+1} + N_{i+2}^3(t)\mathbf{P}_{i+2},$$

$$1 \leq i \leq m-2, \quad t_i \leq t \leq t_{i+1},$$

однако теперь функциональные коэффициенты существенно зависят от промежутков между узлами и определяются рекурсивно

$$N_i^0(t) = \begin{cases} 1, & t_i \leq t \leq t_{i+1}, \\ 0, & t < t_i, t_{i+1} \leq t, \end{cases}$$

$$N_i^1(t) = \frac{t - t_i}{t_{i+1} - t_i} N_i^0(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} N_{i+1}^0(t),$$

$$N_i^2(t) = \frac{t - t_i}{t_{i+2} - t_i} N_i^1(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} N_{i+1}^1(t),$$

$$N_i^3(t) = \frac{t - t_i}{t_{i+3} - t_i} N_i^2(t) + \frac{t_{i+4} - t}{t_{i+4} - t_{i+1}} N_{i+1}^2(t).$$

Свойства функциональных коэффициентов $N_i^3(t)$

Функциональные коэффициенты $N_i^3(t)$:

1+ неотрицательны,

2+ в сумме составляют единицу:

$$\sum_{i=0}^{m} N_i^3(t) = 1.$$

Замечание

Выбор узлов параметризации может быть совершенно произвольным. Однако часто весьма удобной оказывается параметризация, в которой промежуток изменения параметра t и узлы t_i определяются длинами соответствующих хорд:

$$\begin{aligned} t_0 &= 0, \\ t_1 &= |\mathbf{P}_1 - \mathbf{P}_0|, \\ t_i &= t_{i-1} + |\mathbf{P}_i - \mathbf{P}_{i-1}|, \quad i = 2, \ldots, m-1, \\ t_m &= t_{m-1} + |\mathbf{P}_m - \mathbf{P}_{m-1}|. \end{aligned}$$

3.3.4. Рациональные кубические B-сплайновые кривые

По заданному массиву $\mathbf{P}_0, \ldots, \mathbf{P}_m$ *(элементарная) рациональная кубическая B-сплайновая кривая* определяется уравнением следующего вида:

$$\mathbf{R}(t) = \frac{\sum_{i=0}^{3} w_i n_i(t) \mathbf{P}_i}{\sum_{i=0}^{3} w_i n_i(t)}, \quad 0 \le t \le 1.$$

Неотрицательные числа w_i, сумма которых положительна, называются *весами*. В случае, если все веса w_i равны между собой, получается стандартная элементарная кубическая *B*-сплайновая кривая.

Свойства составных рациональных кубических *B*-сплайновых кривых

Составная рациональная кубическая *B*-сплайновая кривая, порожденная массивом **P**:

1+ является C^2-гладкой кривой (при условии $t_{i+1} = t_i + 1$, $t_1 = 0$);

2⁺ как правило, не проходит ни через одну точку заданного массива;

3⁺ лежит в объединении m-2 выпуклых оболочек, порожденных массивами из опорных вершин

$$\mathbf{P}_{i-3}, \mathbf{P}_{i-2}, \mathbf{P}_{i-1}, \mathbf{P}_i, \quad i = 3, \ldots, m;$$

4⁺ "повторяет" контрольную ломаную (в частности, число точек пересечения составной рациональной кубической *B*-сплайновой кривой с произвольной прямой не больше числа точек пересечения контрольной ломаной с этой прямой);

5⁺ если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ лежат на одной прямой, то составная рациональная кубическая *B*-сплайновая кривая также лежит на этой прямой;

6⁺ если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ лежат в одной плоскости, то составная рациональная кубическая *B*-сплайновая кривая также лежит в этой плоскости;

7⁺ изменение одной вершины в массиве приводит к изменению только части кривой: при изменении вершины $\mathbf{P}_i$ нужно пересчитать параметрические уравнения только четырех кривых: $\gamma^{(i-2)}$, $\gamma^{(i-1)}$, $\gamma^{(i)}$, $\gamma^{(i+1)}$;

8⁺ при добавлении в массив одной вершины возникает необходимость пересчета параметрических уравнений только четырех элементарных кривых;

9⁺ составная рациональная кубическая *B*-сплайновая кривая аффинно инвариантна;

10⁺ поведение составной рациональной кубической *B*-сплайновой кривой определяется не только массивом вершин, но и набором свободных параметров - весов w_i, *параметров формы*; при заданном наборе вершин рациональной кубической *B*-сплайновой кривой можно управлять, меняя весовые множители;

11⁺ рациональная кубическая *B*-сплайновая кривая - проективно инвариантна.

3.3.5. Форма Безье составных кубических *B*-сплайновых кривых

Одна и та же составная кривая может быть построена из элементарных фрагментов кубических кривых разных классов; правда, в этом случае для каждого класса требуется свой набор опорных вершин. Здесь мы показываем, как пересчитать вершины заданного массива **P**

на новый массив **S**, чтобы составная кубическая *B*-сплайновая кривая, построенная по массиву **P**, и составная кубическая кривая Безье, построенная по массиву **S**, совпадали. Удобство использования кривых Безье связано с тем, что кубические многочлены Бернштейна $(1-t)^3$, $3t(1-t)^2$, $3t^2(1-t)$, t^3 обладают большей устойчивостью относительно вычислений по сравнению с мономиальным набором 1, t, t^2, t^3.

Пусть заданы вершины

$$\mathbf{P}_{-1}, \mathbf{P}_0, \ldots, \mathbf{P}_m, \mathbf{P}_{m+1}$$

и последовательность узлов $t_0 < \ldots < t_m$.

Положим

$$\Delta_{i-1} = t_i - t_{i-1}, \quad i = 1, \ldots, m\text{-}1,$$

и определим вершины

$$\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_{3m-1}, \mathbf{S}_{3m}$$

по формулам

$$\mathbf{S}_0 = \mathbf{P}_{-1},$$

$$\mathbf{S}_1 = \mathbf{P}_0,$$

$$\mathbf{S}_3 = \frac{\Delta_1}{\Delta_0 + \Delta_1}\mathbf{P}_0 + \frac{\Delta_0}{\Delta_0 + \Delta_1}S_4,$$

$$\mathbf{S}_2 = \frac{\Delta_1}{\Delta_0 + \Delta_1}\mathbf{S}_2 + \frac{\Delta_0}{\Delta_0 + \Delta_1}\mathbf{P}_4,$$

$$\mathbf{S}_{3i-2} = \frac{\Delta_{i-1} + \Delta_i}{\Delta}\mathbf{P}_{i-1} + \frac{\Delta_{i-2}}{\Delta}\mathbf{P}_i,$$

$$\mathbf{S}_{3i-1} = \frac{\Delta_i}{\Delta}\mathbf{P}_{i-1} + \frac{\Delta_{i-2} + \Delta_{i-1}}{\Delta}\mathbf{P}_i,$$

$$\mathbf{S}_{3i} = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i}\mathbf{S}_{3i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i}\mathbf{S}_{3i+1},$$

$$\mathbf{S}_{3m-2} = \frac{\Delta_{m-1}}{\Delta_{m-2} + \Delta_{m-1}}\mathbf{P}_{m-1} + \frac{\Delta_{m-2}}{\Delta_{m-2} + \Delta_{m-1}}\mathbf{P}_m,$$

$$\mathbf{S}_{3m-1} = \mathbf{P}_m,$$

$$\mathbf{S}_{3m} = \mathbf{P}_{m+1},$$

где

$$\Delta = \Delta_{i-2} + \Delta_{i-1} + \Delta_i, \quad i = 2, \ldots, m\text{-}1.$$

Определение. Ломаная $\mathbf{P}_{-1}...\mathbf{P}_{m+1}$ называется *ломаной B-сплайна*, ломаная $\mathbf{S}_0...\mathbf{S}_{3m}$ - соответствующей *ломаной Безье*, а вершины $\mathbf{S}_i$ называются *вершинами Безье.*

C^2-гладкая кубическая *B*-сплайновая кривая, определяемая набором вершин $\mathbf{P}_i$, $i = -1, ..., m + 1$, совпадает с составной кубической кривой Безье, определяемой набором найденных вершин $\mathbf{S}_j$, $j = 0, ..., 3m$, при этом каждая элементарная кривая Безье определяется набором вершин $\mathbf{S}_{3k+l}$, $l = 0, 1, 2, 3$; $k = 0, ..., m - 1$.
Поступая аналогично, можно найти вершины Безье и для рациональных кубических *B*-сплайновых кривых.

3.3.6. Программная реализация алгоритма

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
bspline( p, m, div, output ),
```

где

`Flt p[]` (in) - опорная ломаная сплайна, ее элементы `p[0]...p[m-1]`,
`int m` (in) - число вершин опорной ломаной,
`int div` (in) - число звеньев аппроксимирующей ломаной,
`Flt output[]` (out) - выходной массив с элементами `output[0]...output[(m-3)*div]`.

Приведем полностью текст этой программы.

```
#include "common.h"
Flt bspline_3( Flt p[ 4 ], Flt t )
{
    Flt s = 1.0 - t ;
    Flt t2 = t * t ;
    Flt t3 = t2 * t ;

    return ( s*s*s * p[0] + ( 3*t3 - 6*t2 + 4 ) * p[1] +
      -3*t3 + 3*t2 + 3*t + 1 ) * p[2] + t3 * p[3] ) / 6.0 ;
}

void bspline( Flt p[], int m, int div, Flt output[] )
{
    Flt t, dt = 1.0 / div ;
    int i, d, first = 0 ;

    for( i = 1; i < m - 2; i ++ )
    {
        t = 0 ;
        for( d = 0; d <= div; d ++ )
        {
            output[ first + d ] = bspline_3( &p[ i-1 ], t ) ;
            t += dt ;
        }
```

```
            first += div ;
        }
    }
```

Приведем пример использования функции для построения плоской сплайн-кривой.

```
#include "lib.h"
/* Описание вызываемой функции n */
void bspline( Flt p[], int m, int div, Flt output[] ) ;
/* число точек опорной ломаной  */
#define  M     (8)
#define DIV    (10)
/* число точек аппроксимирующей ломаной */
#define  N     ((M-3)*DIV + 1)
/* Определение опорной ломаной */
Flt xp[M] = {0.056,0.287,0.655,0.716,0.228,0.269,0.666,0.929};
Flt yp[M] = {0.820,0.202,0.202,0.521,0.521,0.820,0.820,0.227};
/* Описание выходного массива */
Flt ox[N], oy[N] ;
void main( void )
{
/* вычисление параметров сплайна  */
bspline( xp, M, DIV, ox ) ;
    bspline( yp, M, DIV, oy ) ;
/* инициализация графической моды */
SetVideoMode() ;
/* отображение на дисплее опорной ломаной */
DrawPolygon( xp, yp, M, 1 ) ;
/* отображение на дисплее построенной сплайновой кривой */
DrawPolygon( ox, oy, N, 2 ) ;
}
```

Текст программы bspline находится в файле bspline.c в поддиректории CURVES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

Приведем также пример программной реализации описанного выше алгоритма построения рационального кубического *B*-сплайна. Программа написана в виде функции на языке С.

Обращение к ней имеет следующий вид:

```
nurbs( p, w, knot, m, div, output ),
```

где

```
    Flt p[]     (in)    - опорная ломаная для рационального B-
    сплайна, ее элементы p[0] .. p[ m-1 ],
    Flt w[]     (in)    - весовые коэффициенты для точек ломаной
                          соответственно w[0] .. w[ m-1 ],
    Flt knot[]  (in)    - узлы knot[0] .. knot[ m+3 ],
    int m       (in)    - число точек опорной ломаной,
    int div     (in)    - число звеньев аппроксимирующей ломаной,
```

Flt output[] (out) - аппроксимирующая ломаная для рацион
го B-сплайна, ее элементы output[0]...output[(m-3)*div]

Приведем полностью текст этой программы.

```
#include "common.h"

Flt N( int n, int i, Flt u, Flt knot[] )
{
    if( n == 0 )  return ( knot[i] <= u && u < knot[i+1]
1.0 : 0.0 ;
    return  ( u - knot[i] ) * N( n-1, i, u, knot ) /
(knot[i+n] - knot[i]) +
            ( knot[i+n+1] - u ) * N( n-1, i+1, u, knot )
(knot[i+n+1] - knot[i+1]) ;
}

Flt nurbs_3( Flt p[], Flt w[], Flt knot[], int j, Flt u
{
    Flt s1 = 0, s2 = 0 ;
    int i ;
    for( i = j-3; i < j+1; i ++ )
    {
        Flt tt = N( 3, i, u, knot ) * w[ i ] ;
        s1 += tt * p[ i ] ;
        s2 += tt ;
    }
    return s1/s2 ;
}
void nurbs( Flt p[], Flt w[], Flt knot[], int m, int div
output[] )
{
    Flt u, du ;
    int j, d ;

    for( j = 3; j < m; j ++ )
    {
        u = knot[ j ] ;
        du = ( knot[ j+1 ] - u ) / div ;
        for( d = 0; d <= div; d ++ )
        {
            output[ (j-3)*div + d ] = nurbs_3( p, w, kno
u ) ;
            u += du ;
        }
    }
}
```

Приведем пример использования функции для построения кой сплайн-кривой.

```
#include "lib.h"
/* Описание вызываемой функции n */

void nurbs( Flt p[], Flt w[], Flt knot[], int m, int div
output[] ) ;
/* число точек опорной ломаной  */
#define  M    (8)
#define DIV  (10)
/* число точек аппроксимирующей ломаной */
```

```
#define  N    ((M-3)*DIV+1)
/* Определение опорной ломаной */
Flt xp[M] = {0.056,0.287,0.655,0.716,0.228,0.269,0.666,0.929};
Flt yp[M] = {0.820,0.202,0.202,0.521,0.521,0.820,0.820,0.227};
/* Весовые коэффициенты */
Flt ww[M] = {  1,    1,    1,    1,    7,    1,    1,    1  };
/* Узлы */
Flt knot[M+4] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 } ;

/* Описание выходного массива */
Flt ox[N], oy[N] ;

void main( void )
{
    /* вычисление параметров сплайна */
    nurbs( xp, ww, knot, M, DIV, ox ) ;
    nurbs( yp, ww, knot, M, DIV, oy ) ;
    /* инициализация графической моды */
SetVidecMode() ;
/* отображение на дисплее опорной ломаной */
DrawPolygon( xp, yp, M, 1 ) ;
/* отображение на дисплее построенной аппроксимирующей
ломаной*/
DrawPolygon( ox, oy, N, 2 ) ;
}
```

Текст программы nurbs находится в файле nurbs.c в поддиректории CURVES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

3.4. Бета-сплайновые кривые

3.4.1. Параметрические уравнения элементарной Бета-сплайновой кривой

По заданному массиву

$$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2. \mathbf{P}_3$$

(элементарная) Бета-сплайновая кривая определяется при помощи векторного уравнения вида

$$\mathbf{R}(t) = b_0(t)\mathbf{P}_0 + b_1(t)\mathbf{P}_1 + b_2(t)\mathbf{P}_2 + b_3(t)\mathbf{P}_3, \quad 0 \le t \le 1,$$

функциональные коэффициенты $b_i(t)$ в котором задаются следующими формулами:

$$b_0(t) = \frac{2\beta_1^3}{\delta}(1-t)^3, \qquad b_3(t) = \frac{2t^3}{\delta},$$

$$b_1(t) = \frac{1}{\delta}\left(2\beta_1^3 t\left(t^2 - 3t + 3\right) + 2\beta_1^2\left(t^3 - 3t + 2\right) + 2\beta_1\left(t^3 - 3t + 2\right) + \beta_2\left(2t^3 - 3t^2 + 1\right)\right),$$

$$b_2(t) = \frac{1}{\delta}\left(2\beta_1^2 t^2\left(-t + 3\right) + 2\beta_1 t\left(-t^2 + 3\right) + 2\beta_2 t^2\left(-2t + 3\right) + 2\left(-t^3 + 1\right)\right),$$

где $\beta_1 > 0$ и $\beta_2 \ge 0$ и $\delta = 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + \beta_2 + 2$.

Числовые параметры β_1 и β_2 называются *параметрами формы* Бета-сплайновой кривой, при этом параметр β_1 называется *параметром скоса (смещения)*, а $\beta_2 \ge 0$ – *параметром напряжения.*

Замечание

При $\beta_1 = 1$ и $\beta_2 = 0$ получается элементарная кубическая B-сплайновая кривая.

Матричная запись параметрических уравнений, описывающих элементарную кубическую Бета-сплайновую кривую,

$$\mathbf{R}(t) = \mathbf{PMT}, \quad 0 \le t \le 1,$$

где

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \end{pmatrix},$$

$$\mathbf{R}(t)=\begin{pmatrix}x(t)\\ y(t)\\ z(t)\end{pmatrix},\qquad \mathbf{T}=\begin{pmatrix}t^0\\ t^1\\ t^2\\ t^3\end{pmatrix},$$

$$\mathbf{M}=\frac{1}{\delta}\begin{pmatrix}2\beta_1^3 & -6\beta_1^3 & 6\beta_1^3 & -2\beta_1^3\\ 4\left(\beta_1^2+\beta_1\right)+\beta_2 & 6\left(\beta_1^3-\beta_1\right) & -3\left(2\beta_1^3+\eta\right) & 2\left(3\beta_1^2+\theta\right)\\ 2 & 6\beta_1 & 3\eta & -2\left(\theta+1\right)\\ 0 & 0 & 0 & 2\end{pmatrix}$$

(здесь $\eta=2\beta_1^2+\beta_2$, $\theta=\beta_1^2+\beta_1+\beta_2$).

Матрица **M** называется *базисной матрицей Бета-сплайновой кривой.*

Свойства элементарной Бета-сплайновой кривой

Свойства функциональных весовых коэффициентов $b_0(t)$, $b_1(t)$, $b_2(t)$, $b_3(t)$ оказывают существенное влияние на поведение элементарной Бета-сплайновой кривой. Укажем некоторые из них.

Функциональные весовые коэффициенты $b_i(t)$

1$^+$ неотрицательны;

2$^+$ в сумме составляют единицу;

3$^+$ не зависят от вершин массива $\mathbf{P}_0,\mathbf{P}_1,\mathbf{P}_2,\mathbf{P}_3$ (универсальны).

Элементарная Бета-сплайновая кривая

1$^+$ лежит внутри выпуклой оболочки, порожденной вершинами $\mathbf{P}_0,\mathbf{P}_1,\mathbf{P}_2,\mathbf{P}_3$ опорной ломаной (рис. 3.21), и, как правило, не проходит ни через одну из опорных вершин;

2$^+$ в концевой точке

$$\frac{1}{\delta}\left(2\beta_1^3\mathbf{P}_0+\left(4\beta_1^2+4\beta_1+\beta_2\right)\mathbf{P}_1+2\mathbf{P}_2\right)$$

касательная Бета-сплайновой кривой параллельна вектору

$$\frac{6}{\delta}\beta_1\left(\beta_1^2\left(\mathbf{P}_1-\mathbf{P}_0\right)+\left(\mathbf{P}_2-\mathbf{P}_1\right)\right),$$

а в концевой точке

$$\frac{1}{\delta}\left(2\beta_1^3\mathbf{P}_1+\left(4\beta_1^2+4\beta_1+\beta_2\right)\mathbf{P}_2+2\mathbf{P}_3\right)$$

вектору

$$\frac{6}{\delta}\left(\beta_1^2\left(\mathbf{P}_2-\mathbf{P}_1\right)+\left(\mathbf{P}_3-\mathbf{P}_2\right)\right).$$

3.4.2. Составные Бета-сплайновые кривые

(Составной) Бета-сплайновой кривой, определяемой массивом

$$\mathbf{P}_0,\ \ldots,\ \mathbf{P}_m,\quad m\geq 3,$$

называется кривая γ, которую можно представить в виде объединения m-2 элементарных Бета-сплайновых кривых $\gamma^{(1)}$, ..., $\gamma^{(m-2)}$,

$$\gamma=\gamma^{(1)}\cup\ \ldots\ \cup\gamma^{(m-2)};$$

(i)-я кривая $\gamma^{(i)}$ описывается параметрическим уравнением следующего вида:

$$\mathbf{R}^{(i)}(t)=\left(\mathbf{P}_{i-1}\quad\mathbf{P}_i\quad\mathbf{P}_{i+1}\quad\mathbf{P}_{i+2}\right)\mathbf{MT},$$

$$0\leq t\leq 1,\qquad i=1,\ \ldots,\ m\text{-}2,$$

где **M** - базисная матрица Бета-сплайна.

Единая параметризация

Рассматривая составную кривую γ как целое, более естественно пользоваться единой параметризацией.

Наиболее простой является параметризация с равноотстоящими целочисленными узлами. Для массива из $m+1$ опорных вершин составная Бета-сплайновая кривая строится из $m-2$ элементарных фрагментов. Если каждый из них определен на единичном отрезке, то длина общего промежутка изменения параметра должна быть равной $m-2$. Взяв 0 за начальную точку, получаем отрезок $[0, m-2]$. В этом случае узлы параметризации определяются по формулам

$$t_1=0,\ \ t_{i+1}=t_i+1,\qquad i=1,\ \ldots,\ m\text{-}3.$$

Описанный выбор отрезка параметризации позволяет записать уравнение составной кубической B-сплайновой кривой γ следующим образом:

$$\mathbf{R}=\mathbf{R}(t),\quad t_1\leq t\leq t_{m-2},$$

где

$$\mathbf{R} = \mathbf{R}^{(i)}(t) = \begin{pmatrix}\mathbf{P}_{i-1} & \mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2}\end{pmatrix}\mathbf{M}\begin{pmatrix}(t-t_i)^0\\(t-t_i)^1\\(t-t_i)^2\\(t-t_i)^3\end{pmatrix},$$

$$t_i \le t \le t_{i+1}, \qquad i = 1, \ldots, m\text{-}3,$$

- параметрическое векторное уравнение (i)-й элементарной Бета-сплайновой кривой $\gamma^{(i)}$.

Свойства составной Бета-сплайновой кривой

Бета-сплайновая кривая, порожденная массивом

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \ge 3,$$

1$^+$ является G^2-гладкой кривой; в точках стыка элементарных кривых $\gamma^{(i)}$ и $\gamma^{(i+1)}$ выполняются равенства

$$\mathbf{R}^{(i+1)}(t_{i+1}) = \mathbf{R}^{(i)}(t_{i+1}) = \frac{1}{\delta}\left(2\beta_1^3\mathbf{P}_i + \left(4\beta_1^3 + 4\beta_1 + \beta_2\right)\mathbf{P}_{i+1} + 2\mathbf{P}_{i+2}\right),$$

$$\dot{\mathbf{R}}^{(i+1)}(t_{i+1}) = \beta_1\dot{\mathbf{R}}^{(i)}(t_{i+1}),$$

$$\ddot{\mathbf{R}}^{(i+1)}(t_{i+1}) = \beta_1^2\ddot{\mathbf{R}}^{(i)}(t_{i+1}) + \beta_2\dot{\mathbf{R}}^{(i)}(t_{i+1});$$

2$^+$ как правило, не проходит ни через одну вершину заданного массива;

3$^+$ лежит в объединении m-2 выпуклых оболочек, порожденных четверками вершин

$$\mathbf{P}_{i-1}, \mathbf{P}_i, \mathbf{P}_{i+1}, \mathbf{P}_{i+2}, \quad i = 1, \ldots, m\text{-}2$$

(рис. 3.23);

4$^+$ "повторяет" опорную ломаную (см. рис. 3.24) (в частности, число точек пересечения составной Бета-сплайновой кривой с произвольной прямой не больше числа точек пересечения контрольной ломаной с этой прямой);

5$^+$ если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ массива лежат на одной прямой, то составная Бета-сплайновая кривая также лежит на этой прямой (между вершинами $\mathbf{P}_0$ и $\mathbf{P}_m$);

6$^+$ если опорные вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$ массива лежат в одной плоскости, то составная Бета-сплайновая кривая также лежит в этой плоскости;

7$^+$ изменение одной вершины в массиве приводит к изменению только части кривой: при изменении вершины $\mathbf{P}_i$ нужно пересчитать параметрические уравнения только четырех элементарных кривых: $\gamma^{(i-2)}$, $\gamma^{(i-1)}$, $\gamma^{(i)}$, $\gamma^{(i+1)}$, в формировании которых принимает участие эта вершина (см. рис. 3.25);

8$^+$ при добавлении в массив одной вершины возникает необходимость пересчета параметрических уравнений только четырех элементарных кривых;

9$^+$ составная Бета-сплайновая кривая аффинно-инвариантна;

10$^-$ Бета-сплайновая кривая проективно-неинвариантна;

11$^+$ параметры β_1 и β_2 позволяют изменять форму Бета-сплайновой кривой, правда сразу всю целиком.

Замечания:

1. *Параметры формы β_1 и β_2 не обязательно должны быть одинаковыми для всех элементарных фрагментов. С учетом взаимного расположения вершин массива их можно выбирать так, чтобы:*

1) пара значений $\beta_1^{(i)}$ и $\beta_2^{(i)}$ для каждого элементарного фрагмента была своя;

2) на каждом единичном отрезке [0,1] функции $\beta_1(t)$ и $\beta_2(t)$ линейно зависели от параметра t;

3) функции $\beta_1^{i}(t))$ и $\beta_2^{i}(t)$ были разными для разных элементарных фрагментов (i = 1, ..., m - 2).

Как правило, выбор параметров формы β_1 и β_2 определяется взаимным расположением вершин в массиве. Если расстояния между соседними вершинами приблизительно равны (различаются не слишком сильно), то выбор параметров формы, одинаковых для всех частичных кривых, дает достаточно хорошее приближение. Если же взаимное расположение вершин нельзя рассматривать как равномерное, то хороших результатов можно добиться подбором переменных параметров формы.

2. *На взаимное расположение вершин в массиве не накладывается никаких ограничений: они могут и совпадать. Однако следует иметь в виду, что в подобных случаях кривая может потерять свою регулярность. Впрочем, если номера совпадающих вершин сильно разнятся, то никакой потери регулярности не происходит.*

Случай, когда совпадают две или три первые (последние) вершины, рассматривается ниже.

3.4.3. Кратные и воображаемые вершины

Составная Бета-сплайновая кривая, как правило, не проходит ни через одну вершину массива, который ее порождает. Однако заранее вполне определенно можно сказать, что начальная точка $\mathbf{R}^{(1)}(0)$ составной кривой γ лежит в треугольнике $\Delta\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2$, а $\mathbf{R}^{(m-2)}(1)$ - в треугольнике $\Delta\mathbf{P}_{m-2}\mathbf{P}_{m-1}\mathbf{P}_m$. Подбором вспомогательных вершин и построением дополнительных элементарных кривых можно добиться того, чтобы начальная точка новой составной кривой выходила на отрезок $\mathbf{P}_0\mathbf{P}_1$, располагалась ближе к вершине $\mathbf{P}_0$ и даже совпадала с ней. Аналогичных результатов можно добиться и для конечной точки. Обычно это проводится путем использования *кратных* или *воображаемых* вершин.

А. Двойные вершины

Положим $\mathbf{P}_{-1} = \mathbf{P}_0$ и $\mathbf{P}_{m+1} = \mathbf{P}_m$ и построим две новые элементарные кривые $\gamma^{(0)}$ и $\gamma^{(m-1)}$, задав их параметрическими уравнениями следующего вида:

$$\mathbf{R}^{(0)}(t) = \left(b_0(t) + b_1(t)\right)\mathbf{P}_0 + b_2(t)\mathbf{P}_1 + b_3(t)\mathbf{P}_2,$$

$$\mathbf{R}^{(m-1)}(t) = b_0(t)\mathbf{P}_{m-2} + b_1(t)\mathbf{P}_{m-1} + \left(b_2(t) + b_3(t)\right)\mathbf{P}_m,$$

где $0 \le t \le 1$.

С учетом кривых $\gamma^{(0)}$ и $\gamma^{(m-1)}$ новая составная Бета-сплайновая кривая

$$\gamma^* = \gamma^{(0)} U \gamma^{(1)} U \ldots U \gamma^{(m-2)} U \gamma^{(m-1)}$$

будет начинаться в точке

$$\mathbf{R}^{0}(0) = \left(1 - \frac{2}{\delta}\right)\mathbf{P}_0 + \frac{2}{\delta}\mathbf{P}_1,$$

касаясь отрезка $\mathbf{P}_0\mathbf{P}_1$,

$$\dot{\mathbf{R}}^{(0)}(0) = \frac{6\beta_1}{\delta}\left(\mathbf{P}_1 - \mathbf{P}_0\right),$$

и заканчиваться в точке

$$\mathbf{R}^{m-1}(1) = \frac{2\beta_1^3}{\delta}\mathbf{P}_{m-1} + \left(1 - \frac{2\beta_1^3}{\delta}\right)\mathbf{P}_m,$$

касаясь отрезка $\mathbf{P}_{m-1}\mathbf{P}_m$,

$$\dot{\mathbf{R}}^{(m-1)}(1) = \frac{6\beta_1^2}{\delta}(\mathbf{P}_m - \mathbf{P}_{m-1})$$

(см. рис. 3.26, а). Кроме того, кривая γ^* будет иметь в двух этих точках нулевую кривизну.

Б. Тройные вершины

Положим $\mathbf{P}_{-2} = \mathbf{P}_{-1} = \mathbf{P}_0$ и $\mathbf{P}_{m+2} = \mathbf{P}_{m+1} = \mathbf{P}_m$ и возьмем в качестве двух новых элементарных кривых $\gamma^{(-1)}$ и $\gamma^{(m)}$ прямолинейные отрезки

$$\mathbf{R}^{(-1)}(t) = \left(1 - \frac{2t^3}{\delta}\right)\mathbf{P}_0 + \frac{2t^3}{\delta}\mathbf{P}_1,$$

$$\mathbf{R}^{(m)}(t) = \frac{2\beta_1^3(1-t)^3}{\delta}\mathbf{P}_{m-1} + \left(1 - \frac{2\beta_1^3(1-t)^3}{\delta}\right)\mathbf{P}_m,$$

где $0 \le t \le 1$ (см. рис. 3.26, б).

С учетом кривых $\gamma^{(0)}$ и $\gamma^{(m-1)}$, построенных ранее, новая составная B-сплайновая кривая

$$\gamma^{**} = \gamma^{(-1)} \cup \gamma^{(0)} \cup \gamma^{(1)} \cup \ldots \cup \gamma^{(m)} \cup \gamma^{(m-1)} \cup \gamma^{(m)}$$

будет начинаться в вершине $\mathbf{P}_0$ и заканчиваться в вершине $\mathbf{P}_m$ (см. рис. 3.26, б).

В. Воображаемые вершины

Выбором дополнительных вершин $\mathbf{P}_{-1}$ и $\mathbf{P}_{m+1}$ к массиву

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \ge 3,$$

можно добиться выполнения различных условий на концах составной кривой.

Например, составная B-сплайновая кривая, построенная по новому массиву

$$\mathbf{P}_{-1}, \mathbf{P}_0, \ldots, \mathbf{P}_m, \mathbf{P}_{m+1},$$

где

$$\mathbf{P}_{-1} = \frac{1}{\beta_1^3}(\mathbf{P}_0 - \mathbf{P}_1) + \mathbf{P}_0, \quad \mathbf{P}_{m+1} = \beta^3(\mathbf{P}_m - \mathbf{P}_{m-1}) + \mathbf{P}_m,$$

будет начинаться в вершине $\mathbf{P}_0$, касаясь отрезка $\mathbf{P}_0\mathbf{P}_1$,

$$\dot{\mathbf{R}}^{(0)}(0) = \frac{6(\beta_1 + 1)}{\delta}(\mathbf{P}_1 - \mathbf{P}_0),$$

и заканчиваться в вершине $\mathbf{P}_m$, касаясь отрезка $\mathbf{P}_{m-1}\mathbf{P}_m$,

$$\dot{\mathbf{R}}^{(m-1)}(1) = \frac{6\beta_1^2}{\delta}(\mathbf{P}_m - \mathbf{P}_{m-1})$$

(см. рис. 3.27). Кривизны новой кривой в точках $\mathbf{P}_0$ и $\mathbf{P}_m$, вообще говоря, отличны от нуля.

Замечание

Дополнительные вершины $\mathbf{P}_{-1}$ и $\mathbf{P}_{m+1}$ можно выбрать так, чтобы в концах новой составной кривой 1-е или 2-е производные радиусов-векторов $\mathbf{R}^{(0)}(t)$ и $\mathbf{R}^{(m-1)}(t)$ кривых $\gamma^{(0)}$ и $\gamma^{(m-1)}$ совпадали с заданными значениями (соответственно при $t = 0$ и $t = 1$).

Построение замкнутой кривой

Чтобы по заданному массиву

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 3,$$

построить G^2-непрерывную замкнутую кривую, достаточно выбрать дополнительные вершины $\mathbf{P}_{m+1}, \mathbf{P}_{m+2}, \mathbf{P}_{m+3}$ по правилу

$$\mathbf{P}_{m+1} = \mathbf{P}_0, \mathbf{P}_{m+2} = \mathbf{P}_1, \mathbf{P}_{m+3} = \mathbf{P}_2$$

и рассмотреть массив

$$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_m, \mathbf{P}_{m+1} = \mathbf{P}_0, \mathbf{P}_{m+2} = \mathbf{P}_1, \mathbf{P}_{m+3} = \mathbf{P}_2.$$

(см. рис. 3.28).

3.4.4. Программная реализация алгоритма

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
betaspline( b1, b2, p, m, div, output ),
```

где

```
    Flt b1, b2  in    - параметры beta1 и beta2,
    Flt p[]     in    - опорная ломаная B-сплайна,
                        ее элементы p[0] .. p[m-1],
    int m       in    - число точек опорной ломаной,
    int div     in    - число звеньев ломаной, аппроксимирующей
                        сплайн
    Flt output[] out  - выходной массив длины (m-3)*div, содержит
                вершины аппроксимирующей ломаной для B-сплайна,
                его элементы output[0] ... output [(m-3)*div]
```

Приведем полностью текст этой программы:

```
#include "common.h"
```

```
static Flt beta1, beta2;
static Flt b12, b13, b22, b23 ; /* 2-я и 3-я степени
параметров beta1 и beta2 соответственно */
static Flt delta, d ;

Flt betaspline_3( Flt p[ 4 ], Flt t )
{
    Flt s = 1.0 - t ;
    Flt t2 = t * t ;
    Flt t3 = t2 * t ;

    Flt b0 = 2 * b13 * d * s * s * s ;
    Flt b3 = 2 * t3 * d ;
    Flt b1 = d*(2*b13*t*(t2-3*t+3)+2*b12*(t3-
3*t2+2)+2*beta1*(t3-3*t+2)+beta2*(2*t3-3*t2+1)) ;
    Flt b2 = d * ( 2*b12*t2*(-t+3) + 2*beta1*t*(-
t2+3)+beta2*t2*(-2*t+3)+2*(-t3+1)) ;

    return b0*p[0] + b1*p[1] + b2*p[2] + b3*p[3] ;
}

void betaspline( Flt b1, Flt b2, Flt p[], int m, int div, Flt
output[] )
{
    Flt t, dt = 1.0 / div ;
    int i, dd, first = 0 ;

    beta1 = b1 ;
    beta2 = b2 ;
    b12 = beta1 * beta1 ;
    b13 = b12 * beta1 ;
    b22 = beta2 * beta2 ;
    b23 = b22 * beta2 ;

    delta = 2.0 * b13 + 4.0 * b12 + 4.0 * beta1 + beta2 + 2.0 ;
    d = 1.0 / delta ;

    for( i = 1; i < m - 2; i ++ )
    {
        t = 0 ;
        for( dd = 0; dd <= div; dd ++ )
        {
         output[ first + dd ] = betaspline_3( &p[ i-1 ], t ) ;
         t += dt ;
        }
        first += div ;
    }
}
```

Приведем пример использования функции для построения плоской сплайн-кривой.

```
#include "lib.h"
/* Описание вызываемой функции */
void betaspline( Flt b1, Flt b2, Flt p[], int m, int div, Flt
output[] ) ;

/* Число точек опорной ломаной */
#define  M    (8)
#define DIV   (10)
/* Число точек аппроксимирующей ломаной */
```

```
#define  N1    ((M-3)*DIV+1)
#define  N2    ((M-1)*DIV+1)
#define  N3    ((M+1)*DIV+1)

/* Определение опорной ломаной                    */
/* первая и последняя точки утроены для           */
/* демонстрации кратных вершин.                   */
Flt xp[M+4] = { 0.056, 0.056, 0.056, 0.287, 0.655, 0.716,
                0.228, 0.269, 0.666, 0.929, 0.929, 0.929 } ;
Flt yp[M+4] = { 0.820, 0.820, 0.820, 0.202, 0.202, 0.521,
                0.521, 0.820, 0.820, 0.227, 0.227, 0.227 } ;

Flt beta1 = 1.0, beta2 = 0.0 ;
/* Описание выходных массивов    */
Flt ox1[N1], oy1[N1] ;
Flt ox2[N2], oy2[N2] ;
Flt ox3[N3], oy3[N3] ;

void main( void )
{
    /* вычисление параметров сплайна*/
    betaspline( beta1, beta2, &xp[2], M, DIV, ox1 ) ;
    betaspline( beta1, beta2, &yp[2], M, DIV, oy1 ) ;

    betaspline( beta1, beta2, &xp[1], M+2, DIV, ox2 ) ;
    betaspline( beta1, beta2, &yp[1], M+2, DIV, oy2 ) ;

    betaspline( beta1, beta2, xp, M+4, DIV, ox3 ) ;
    betaspline( beta1, beta2, yp, M+4, DIV, oy3 ) ;

/* инициализация графической моды */
    SetVideoMode() ;
/* отображение на дисплее опорной ломаной */
    DrawPolygon( &xp[2], &yp[2], M, 1 ) ;
/* отображение на дисплее построенной аппроксимирующей
ломаной*/
    DrawPolygon( ox3, oy3, N3, 2 ) ;
    DrawPolygon( ox2, oy2, N2, 3 ) ;
    DrawPolygon( ox1, oy1, N1, 4 ) ;
}
```

Текст программы betaspline находится в файле beta.c в поддиректории CURVES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

3.5. Другие сплайновые кривые

3.5.1. Интерполяционные кубические кривые Эрмита

По заданным вершинам $\mathbf{P}_0$ и $\mathbf{P}_1$ и не нулевым векторам $\mathbf{Q}_0$ и $\mathbf{Q}_1$ *(элементарная) кубическая кривая Эрмита* определяется при помощи векторного уравнения, имеющего следующий вид:

$$\mathbf{R}(t) = (1 - 3t^2 + 2t^3)\mathbf{P}_0 + t^2(3 - 2t)\mathbf{P}_1 + t(1 - 2t + t^2)\mathbf{Q}_0 - t^2(1 - t)\mathbf{Q}_1,$$

$$0 \le t \le 1.$$

Матричная запись параметрических уравнений, описывающих элементарную кубическую кривую Эрмита, -

$$\mathbf{R}(t) = \mathbf{GMT}, \quad 0 \le t \le 1,$$

где

$$\mathbf{G} = (\mathbf{P}_0 \quad \mathbf{P}_1 \quad \mathbf{Q}_0 \quad \mathbf{Q}_1) = \begin{pmatrix} x_0 & x_1 & u_0 & u_1 \\ y_0 & y_1 & v_0 & v_1 \\ z_0 & z_1 & w_0 & w_1 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}, \quad \mathbf{T} = \begin{pmatrix} t^0 \\ t^1 \\ t^2 \\ t^3 \end{pmatrix}.$$

Матрица **M** называется *базисной матрицей кубической кривой Эрмита*, а матрица **G** - ее *геометрической матрицей.*

Касательный вектор элементарной кубической кривой Эрмита в концевой точке $\mathbf{P}_0$, $\mathbf{R}(0) = \mathbf{P}_0$, совпадает с заданным вектором $\mathbf{Q}_0$, $\dot{\mathbf{R}}(0) = \mathbf{Q}_0$, а в концевой точке $\mathbf{P}_1$, $\mathbf{R}(1) = \mathbf{P}_1$, - с вектором $\mathbf{Q}_1$, $\dot{\mathbf{R}}(1) = \mathbf{Q}_1$ (рис. 3.28).

При линейной замене параметра

$$t \mapsto \tau = (1 - t)a + tb,$$

преобразующей единичный отрезок [0, 1] в промежуток [*a*, *b*], форма элементарной кривой Эрмита изменяется. Для ее сохранения необхо-

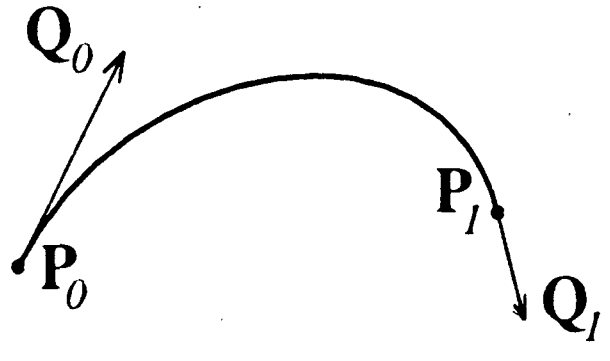


Рис. 3.28

димо заменить векторы $\mathbf{Q}_0$ и $\mathbf{Q}_1$ на

$$\frac{1}{b-a}\mathbf{Q}_0 \quad \text{и} \quad \frac{1}{b-a}\mathbf{Q}_1$$

соответственно.

Кроме предложенных форм записи элементарной кривой Эрмита используются и другие.

Запись элементарной кубической кривой Эрмита

в форме Безье -

$$\mathbf{R}(t) = B_0^3(t)\mathbf{P}_0 + B_1^3(t)\left(\mathbf{P}_0 + \frac{1}{3}\mathbf{Q}_0\right) + B_2^3(t)\left(\mathbf{P}_1 - \frac{1}{3}\mathbf{Q}_1\right) + B_3^3(t)\mathbf{P}_1,$$

где $B_i^3(t)$ - кубические многочлены Бернштейна;

в форме Эрмита -

$$\mathbf{R}(t) = H_0^3(t)\mathbf{P}_0 + H_1^3(t)\mathbf{Q}_0) + H_2^3(t)\mathbf{Q}_1 + H_3^3(t)\mathbf{P}_1,$$

где $H_i^3(t)$ - кубические многочлены Эрмита,

$$H_0^3(t) = B_0^3(t) + B_1^3(t), \qquad H_1^3(t) = \frac{1}{3}B_1^3(t),$$

$$H_2^3(t) = -\frac{1}{3}B_2^3(t), \qquad H_3^3(t) = B_2^3(t) + B_3^3(t).$$

Замечание

Вследствие большей вычислительной устойчивости многочленов Бернштейна описание эрмитовой кривой в форме Безье является более предпочтительным.

Составные кубические кривые Эрмита

(Составной) кубической кривой Эрмита, определяемой массивом

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 1,$$

и парой не нулевых векторов $\mathbf{Q}_0$ и $\mathbf{Q}_m$, называется кривая γ, которую можно представить в виде объединения элементарных кубических кривых Эрмита $\gamma^{(1)}, \ldots, \gamma^{(m)}$,

$$\gamma = \gamma^{(1)} \cup \ldots \cup \gamma^{(m)};$$

(i)-я кривая $\gamma^{(i)}$ описывается параметрическим уравнением следующего вида:

$$\mathbf{R}^{(i)}(t) = \begin{pmatrix}\mathbf{P}_{i-1} & \mathbf{P}_i & \mathbf{Q}_{i-1} & \mathbf{Q}_i\end{pmatrix}\mathbf{MT}, \qquad 0 \leq t \leq 1, \; i = 1, \ldots, m,$$

где **M** - базисная матрица эрмитова сплайна, а векторы $\mathbf{Q}_1, \ldots, \mathbf{Q}_{m-1}$ определяются из матричного уравнения следующего вида:

$$\begin{pmatrix} 1 & 4 & 1 & & & & & \\ & 1 & 4 & 1 & & & & \\ & & & \ddots & & & & \\ & & & & 1 & 4 & 1 & \\ & & & & & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{Q}_0 \\ \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_{m-1} \\ \mathbf{Q}_m \end{pmatrix} =$$

$$\begin{pmatrix} -3 & 0 & 3 & & & & & \\ & -3 & 0 & 3 & & & & \\ & & & \ddots & & & & \\ & & & & -3 & 0 & 3 & \\ & & & & & -3 & 0 & 3 \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_{m-1} \\ \mathbf{P}_m \end{pmatrix}.$$

Единая параметризация

Рассматривая составную кривую γ как целое, более естественно пользоваться единой параметризацией.

Наиболее простой является параметризация с равноотстоящими целочисленными узлами. Для массива из $m + 1$ опорных вершин составная кубическая эрмитова кривая строится из m элементарных фрагментов. Если каждый из них определен на единичном отрезке, то длина общего промежутка изменения параметра должна быть равной m. Взяв 0 за начальную точку, получаем отрезок $[0, m]$. В этом случае узлы параметризации определяются по формуле

$$t_0 = 0, \quad t_i = t_{i-1} + 1, \qquad i = 1, \ldots, m.$$

Описанный выбор отрезка параметризации позволяет записать уравнение составной кубической эрмитовой кривой γ следующим образом:

$$\mathbf{R} = \mathbf{R}(t), \quad t_0 \le t \le t_m,$$

где

$$\mathbf{R} = \mathbf{R}^{i}(t) = (\mathbf{P}_{i-1} \quad \mathbf{P}_i \quad \mathbf{Q}_{i-1} \quad \mathbf{Q}_i)\mathbf{M} \begin{pmatrix} (t - t_i)^0 \\ (t - t_i)^1 \\ (t - t_i)^2 \\ (t - t_i)^3 \end{pmatrix}, \qquad t_{i-1} \le t \le t_i, \quad i = 1, \ldots, m,$$

- параметрическое векторное уравнение (i)-й элементарной кубической кривой Эрмита γ^{i}.

Свойства составной кубической кривой Эрмита

Составная кубическая кривая Эрмита, порожденная массивом

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 1,$$

и парой не нулевых векторов $\mathbf{Q}_0$ и $\mathbf{Q}_m$:

1$^+$ является C^2-гладкой кривой (имеет непрерывную кривизну);

2$^+$ проходит через вершины $\mathbf{P}_0, \ldots, \mathbf{P}_m$, в точках стыка элементарных кривых $\gamma^{(i)}$ и $\gamma^{(i+1)}$ выполняются равенства

$$\mathbf{R}^{(i)}(t_i) = \mathbf{R}^{(i+1)}(t_i) = \mathbf{P}_i$$

(рис. 3.29);

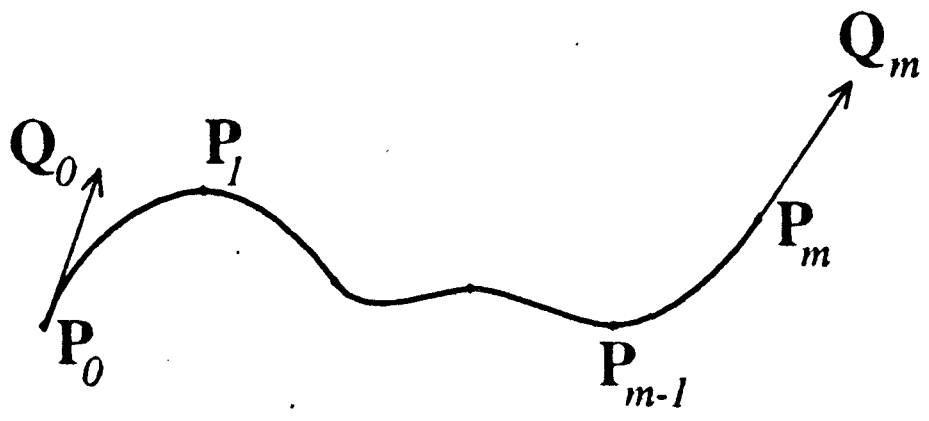


Рис. 3.29

3$^+$ касательный вектор $\mathbf{Q}_i$ во внутренней вершине $\mathbf{P}_i$ ($i = 1, \ldots, m-1$) однозначно определяется через вершины массива **P** и касательные векторы $\mathbf{Q}_0$ и $\mathbf{Q}_m$ в концевых вершинах;

4$^-$ не лежит в выпуклой оболочке, порожденной заданным массивом;

5$^-$ изменение одной вершины в массиве или одного из касательных векторов в концевых вершинах массива приводит к изменению всей кривой;

6$^-$ при добавлении в массив одной вершины возникает необходимость пересчета всех параметрических уравнений;

7$^+$ составная кубическая кривая Эрмита аффинно-инвариантна;

8$^-$ заданные векторы однозначно определяют составную кубическую кривую Эрмита, не давая возможности хоть как-то влиять на ее форму;

9$^-$ кубическая кривая Эрмита проективно-неинвариантна.

Замечание

При помощи элементарных кубических кривых Эрмита составную кривую можно построить по заданному массиву

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 1,$$

и произвольному набору не нулевых векторов

$$\mathbf{Q}_0, \ldots, \mathbf{Q}_m.$$

Каждая четверка $\mathbf{P}_{i-1}$, $\mathbf{P}_i$, $\mathbf{Q}_{i-1}$, $\mathbf{Q}_i$ задает элементарную эрмитову кривую, которая описывается матричным параметрическим уравнением вида

$$\mathbf{R} = \mathbf{R}^{(i)}(t) = (\mathbf{P}_{i-1} \quad \mathbf{P}_i \quad \mathbf{Q}_{i-1} \quad \mathbf{Q}_i)\mathbf{M}\begin{pmatrix}(t-t_i)^0\\(t-t_i)^1\\(t-t_i)^2\\(t-t_i)^3\end{pmatrix},$$

$$t_{i-1} \le t \le t_i, \qquad i = 1, \ldots, m.$$

Однако в отличие от рассмотренного выше случая получаемая в результате составная кривая является всего лишь C^1-гладкой.

Программная реализация

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
hermite( p, m, q0, qm, div, output ),
```

где

```
    Flt p[]       (in)  - опорная ломаная сплайна,
                          ее элементы p[0] .. p[ m-1 ]
    int m         (in)  - число точек опорной ломаной
    Flt q0, qm    (in)  - компоненты касательных векторов
    int div       (in)  - число звеньев аппроксимирующей ломаной
    Flt output[]  (out) - аппроксимирующая ломаная кривой Эрмита,
                          ее элементы output[0]...output[(m-1)*div]
```

Приведем полностью текст этой программы.

```
#include "common.h"

void find_q( Flt q0, Flt qm, Flt p[], int m, Flt q[] )
{
    Flt a[ 30 ], b[ 30 ] ;
    int i ;

    a[1] = 0.0 ;    b[1] = q0 ;    q[ m ] = qm ;
    for( i = 1; i <= m-1; i ++ )
    {
        a[ i+1 ] = - 1.0 / ( 4 + a[ i ] ) ;
        b[ i+1 ] = ( b[i] - 3 * ( p[i+1] - p[i-1] ) ) * a[ i+1
] ;
    }
    for( i = m-1; i >= 0; i -- )
        q[ i ] = a[i+1]*q[i+1] + b[i+1] ;
}

Flt hermite_3( Flt p[ 2 ], Flt q[ 2 ], Flt t )
{
    Flt t2 = t * t ;
    Flt t3 = t2 * t ;

    return (1-3*t2+2*t3)*p[0] + t2*(3-2*t)*p[1] +
           t*(1-2*t+t2)*q[0] - t2*(1-t)*q[1] ;
}
void hermite( Flt p[], int m, Flt q0, Flt qm, int div, Flt
output[] )
```

```
{
    Flt t, dt = 1.0 / div ;
    Flt q[ 30 ] ;
    int i, d, first = 0 ;

    find_q( q0, qm, p, m-1, q ) ; /* find intermediate values
Qi */

    for( i = 1; i < m; i ++ )
    {
        t = 0 ;
        for( d = 0; d <= div; d ++ )
        {
            output[ first + d ] = hermite_3( &p[ i-1 ], &q[ i-
1 ], t ) ;
            t += dt ;
        }
        first += div ;
    }
}
```

Приведем пример использования функции для построения плоской сплайн-кривой.

```
#include "lib.h"

/* Описание вызываемой функции */
void hermite( Flt p[], int m, Flt q0, Flt qm, int div, Flt
output[] ) ;
/* Число точек опорной ломаной */
#define  M    (8)
#define DIV   (10)
/* Число точек аппроксимирующей ломаной */
#define  N    ((M-1)*DIV+1)

/* Построение опорной ломаной                    */
Flt xp[M] = { 0.056, 0.287, 0.655, 0.716,
              0.228, 0.269, 0.666, 0.929 } ;
Flt yp[M] = { 0.820, 0.202, 0.202, 0.521,
              0.521, 0.820, 0.820, 0.227 } ;
Flt q0x = .1, q0y = .1 ;
Flt qmx = -.1, qmy = -.1 ;

/* Описание выходных массивов     */
Flt ox[N], oy[N] ;

void main( void )
{
/* вычисление параметров сплайна*/
    hermite( xp, M, q0x, qmx, DIV, ox ) ;
    hermite( yp, M, q0y, qmy, DIV, oy ) ;
/* инициализация графической моды */
SetVideoMode() ;
/* инициализация графической моды */
DrawPolygon( xp, yp, M, 1 ) ;
/* отображение на дисплее опорной ломаной */
DrawPolygon( ox, oy, N, 2 ) ;
}
```

Текст программы hermite находится в файле hermite.c в поддиректории CURVES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

3.5.2. Сплайновые кривые Catmull-Rom

По заданному массиву

$$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$$

(элементарная) сплайновая кривая Catmull-Rom определяется при помощи векторного уравнения, имеющего следующий вид:

$$\mathbf{R}(t) = \frac{1}{2}\left(-t(1-t)^2\mathbf{P}_0 + \left(2-5t^2+3t^3\right)\mathbf{P}_1 + t\left(1+4t-3t^2\right)\mathbf{P}_2 - t^2(1-t)\mathbf{P}_3\right),$$

$$0 \le t \le 1.$$

Матричная запись параметрических уравнений, описывающих элементарную сплайновую кривую Catmull-Rom, -

$$\mathbf{R}(t) = \mathbf{PMT}, \quad 0 \le t \le 1,$$

где

$$\mathbf{P} = \begin{pmatrix}\mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3\end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, \quad \mathbf{M} = \frac{1}{2}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{T} = \begin{pmatrix} t^0 \\ t^1 \\ t^2 \\ t^3 \end{pmatrix}.$$

Матрица **M** называется *базисной матрицей сплайна Catmull-Rom.*

Касательная к элементарной сплайновой кривой Catmull-Rom в концевой точке $\mathbf{P}_1$, $\mathbf{R}(0) = \mathbf{P}_1$, параллельна отрезку $\mathbf{P}_0\mathbf{P}_2$, а в концевой точке $\mathbf{P}_2$, $\mathbf{R}(1) = \mathbf{P}_2$, - отрезку $\mathbf{P}_1\mathbf{P}_3$ (рис. 3.30).

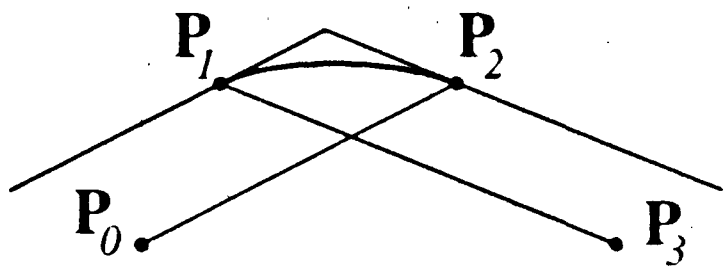


Рис. 3.30

Составные сплайновые кривые Catmull-Rom

(Составной) сплайновой кривой Catmull-Rom, определяемой массивом

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 3,$$

называется кривая γ, которую можно представить в виде объединения элементарных сплайновых кривых Catmull-Rom

$$\gamma^{(1)}, \ldots, \gamma^{(m-2)};$$

(i)-я кривая $\gamma^{(i)}$ описывается параметрическим уравнением следующего вида:

$$\mathbf{R}^{(i)}(t) = \begin{pmatrix}\mathbf{P}_{i-1} & \mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2}\end{pmatrix}\mathbf{MT},$$

$$0 \leq t \leq 1, \quad i = 1, \ldots, m - 2,$$

где **M** - базисная матрица сплайна Catmull-Rom.

Рассматривая составную кривую γ как целое, более естественно пользоваться единой параметризацией. Взяв за t_1 произвольное число (например, положив $t_1 = 0$), определим узлы параметризации по формуле

$$t_{i+1} = t_i + 1, \quad i = 1, \ldots, m - 3.$$

Тогда уравнение сплайновой кривой Catmull-Rom γ можно записать так:

$$\mathbf{R} = \mathbf{R}(t), \quad t_1 \leq t \leq t_{m-2},$$

где

$$\mathbf{R} = \mathbf{R}^{i}(t) = \begin{pmatrix}\mathbf{P}_{i-1} & \mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2}\end{pmatrix}\mathbf{M}\begin{pmatrix}(t-t_i)^0\\(t-t_i)^1\\(t-t_i)^2\\(t-t_i)^3\end{pmatrix},$$

$$t_i \leq t \leq t_{i+1}, \quad i = 1, \ldots, m - 3,$$

- параметрическое векторное уравнение (i)-й элементарной сплайновой кривой Catmull-Rom $\gamma^{(i)}$.

Свойства составной сплайновой кривой Catmull-Rom

Составная кубическая сплайновая кривая Catmull-Rom, порожденная массивом

$$\mathbf{P}_0, \ldots, \mathbf{P}_m, \quad m \geq 3:$$

1$^+$ является C^1-гладкой кривой;

2$^+$ интерполирует вершины $\mathbf{P}_1, \ldots, \mathbf{P}_{m-1}$, в точках стыка элементарных кривых $\gamma^{(i)}$ и $\gamma^{(i+1)}$ выполняются равенства

$$\mathbf{R}^{(i)}(t_{i+1}) = \mathbf{R}^{(i+1)}(t_{i+1}) = \mathbf{P}_{i-1};$$

3$^+$ касательный вектор в вершине $\mathbf{P}_{i-1}$, $i = 2, \ldots, m$, параллелен отрезку, соединяющему $\mathbf{P}_{i-2}$ и $\mathbf{P}_i$,

$$\dot{\mathbf{R}}_i(t_{i+1}) = \dot{\mathbf{R}}_{i+1}(t_{i+1}) = \frac{1}{2}(-\mathbf{P}_{i-2} + \mathbf{P}_i)$$

(рис. 3.31);

4$^-$ не лежит в выпуклой оболочке, порожденной заданным массивом;

5$^+$ изменение одной вершины в массиве приводит к изменению только части кривой: при изменении вершины $\mathbf{P}_i$ нужно пересчитать параметрические уравнения только четырех кривых: $\gamma^{(i-2)}, \gamma^{(i-1)}, \gamma^{(i)}, \gamma^{(i+1)}$;

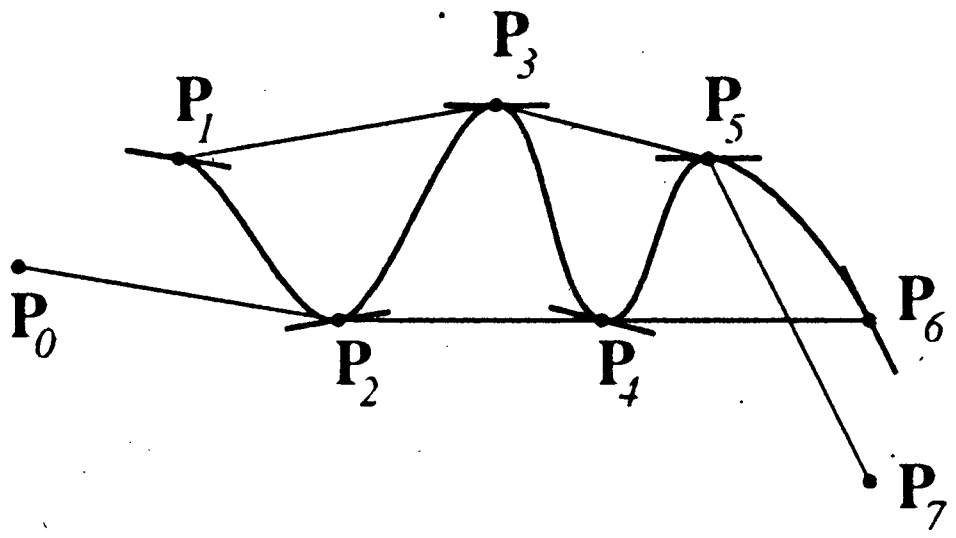


Рис. 3.31

6$^+$ при добавлении в массив одной вершины возникает необходимость пересчета параметрических уравнений только четырех элементарных кривых;

7$^+$ составная кубическая сплайновая кривая Catmull-Rom аффинно-инвариантна;

8$^-$ заданный массив однозначно определяет составную сплайновую кривую Catmull-Rom, не давая возможности хоть как-то влиять на ее форму;

9$^-$ сплайновая кривая Catmull-Rom проективно-неинвариантна.

Программная реализация

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
catmull_rom  p, m, div, output  ,
```

где

```
    Flt p[]       in   - опорная ломаная сплайна,
                         ее элементы p[0] .. p[ m-1 ]
    int m         in   - число точек опорной ломаной
    Flt q0, qm    in   - компоненты касательных векторов
    int div       in   - число звеньев аппроксимирующей ломаной
    Flt output[]  out  - аппроксимирующая ломаная кривой Эрмита,
```

ее элементы output[0]...output[(m-1)*div]

Приведем полностью текст этой программы.

```
#include "common.h"

Flt catmull_rom_3( Flt p[ 4 ], Flt t )
{
    Flt s = 1.0 - t ;
    Flt t2 = t * t ;
    Flt t3 = t2 * t ;

    return .5 * ( -t * s * s * p[0] + ( 2 - 5*t2 + 3*t3 ) *
p[1] +
                t * ( 1 + 4 * t - 3 * t2 ) * p[2] - t2 * s *
p[3] ) ;
}

void catmull_rom( Flt p[], int m, int div, Flt output[] )
{
    Flt t, dt = 1.0 / div ;
    int i, d, first = 0 ;

    for( i = 1; i < m - 2; i ++ )
    {
        t = 0 ;
        for( d = 0; d <= div; d ++ )
        {
            output[ first + d ] = catmull_rom_3( &p[ i-1 ], t
) ;
            t += dt ;
        }
        first += div ;
    }
}
```

Приведем пример использования функции для построения плоской сплайн-кривой.

```
#include "lib.h"
/* Описание вызываемой функции */
void catmull_rom( Flt p[], int m, int div, Flt output[] ) ;
/* Число точек опорной ломаной */
#define  M    (8)
#define DIV   (10)
/* Число точек аппроксимирующей ломаной */
#define  N    ((M-3)*DIV)
/* Построение опорной ломаной                    */
Flt xp[M] = { 0.056, 0.287, 0.655, 0.716, 0.228, 0.269, 0.666,
0.929 } ;
Flt yp[M] = { 0.820, 0.202, 0.202, 0.521, 0.521, 0.820, 0.820,
0.227 } ;
/* Описание выходных массивов     */
Flt ox[(M-3)*DIV], oy[(M-3)*DIV] ;
void main( void )
{
/* вычисление параметров сплайна*/
    catmull_rom( xp, M, DIV, ox ) ;
    catmull_rom( yp, M, DIV, oy ) ;
/* инициализация графической моды */
```

```
SetVideoMode() ;
/* отображение на дисплее опорной ломаной */
DrawPolygon( xp, yp, M, 1 ) ;
/* отображение на дисплее аппроксимирующей ломаной */
DrawPolygon( ox, oy, N, 2 ) ;
}
```

Текст программы catmull_rom находится в файле catmull.c в поддиректории CURVES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

3.5.3. Составные плоские кубические кривые, заданные в неявной форме

Плоские сплайновые кривые можно составлять из элементарных фрагментов алгебраических кривых, описываемых неявными уравнениями.

Геометрическая конструкция фрагмента алгебраической кривой 3-й степени

Пусть $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ - неколлинеарные точки. Положение точек плоскости относительно треугольника $\Delta = \Delta\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2$ можно описывать при помощи *барицентрических координат.*

Пусть (x_i, y_i) - декартовы координаты точек $\mathbf{P}_i$, $i = 0, 1, 2$, и $\mathbf{P}(x, y)$ - произвольная точка плоскости. Справедливо разложение

$$(x, y) = s(x_0, y_0) + t(x_1, y_1) + u(x_2, y_2),$$

где s, t и u - барицентрические координаты точки $\mathbf{P}(x, y)$ относительно точек $\mathbf{P}_0$, $\mathbf{P}_1$ и $\mathbf{P}_2$, причем $s + t + u = 1$.

Если точка $\mathbf{P}(x, y)$ лежит внутри треугольника Δ, то $s > 0$, $t > 0$, $u > 0$.

Возьмем произвольную точку $\mathbf{A}(s_0, t_0, u_0)$ внутри треугольника $\Delta(s_0 > 0, t_0 > 0, u_0 > 0)$ и точку $\mathbf{B}(s_1, 0, u_1)$, лежащую на продолжении отрезка $\mathbf{P}_0\mathbf{P}_2$ ($s_1 < 0$, $u_1 > 1$ или $s_1 > 1$, $u_i < 0$) (рис. 3.32).

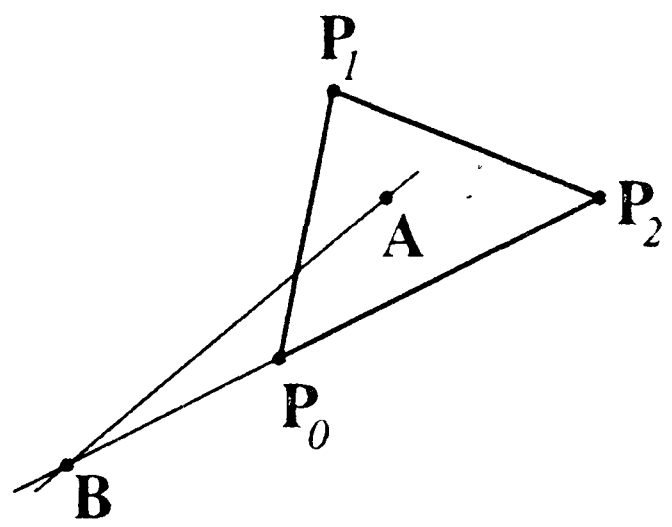


Рис. 3.32

В семействе неявно заданных алгебраических кривых 3-й степени

$$as^2 + bsu^2 + cst^2 + dt^2u + estu = 0$$

существует кривая, которая проходит через точку $\mathbf{P}_0$, касаясь отрезка $\mathbf{P}_0\mathbf{P}_1$, через точку $\mathbf{P}_2$, касаясь отрезка $\mathbf{P}_1\mathbf{P}_2$, и через точку **A**, касаясь отрезка **AB** (рис. 3.33). Чтобы записать уравнение этой кривой, достаточно положить

$$a = -h\beta_2 t_0^2 u_0, \quad b = \beta_1 s_0 t_0^2, \quad c = -(1-\beta_1) s_0 u_0^2,$$

$$d = h(1-\beta_2) s_0^2 u_0, \quad e = (1-h-2\beta_1+2h\beta_2) s_0 t_0 u_0,$$

где $0 < \beta_1, \beta_2 < 1$ - произвольные числа, а

$$h = \frac{s_0 u_1}{s_1 u_0}.$$

Кривизна части кривой, лежащей внутри треугольника, неотрицательна.

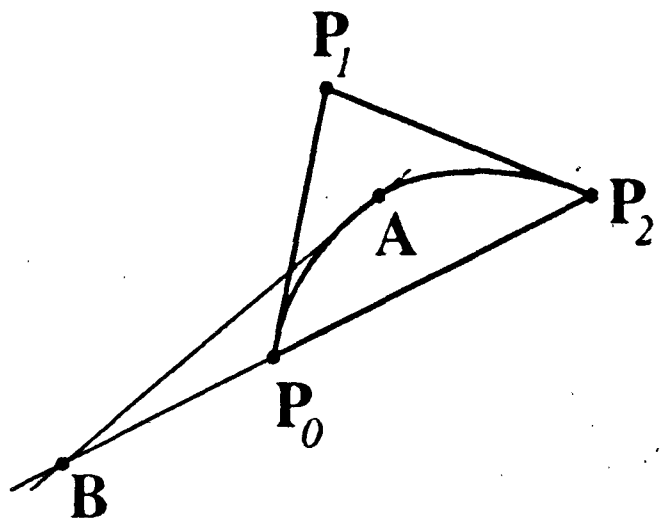


Рис. 3.33

Составная алгебраическая кривая

Пусть на плоскости задан набор точек

$$\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{2m}$$

такой, что точки $\mathbf{P}_{2i-1}, \mathbf{P}_{2i}, \mathbf{P}_{2i+1}$, $i = 1, \ldots, m-1$, коллинеарны (рис. 3.34).

Кривая, составленная из элементарных фрагментов, построенных по описанному выше правилу для каждого треугольника

$$\Delta_i = \Delta \mathbf{P}_{2i}\mathbf{P}_{2i+1}\mathbf{P}_{2i+2}$$

путем произвольного выбора интерполируемой точки $\mathbf{A}_i$, касательной к искомой кривой в этой точке, и параметров формы β_1^i и β_2^i, будет выпуклой в каждом треугольнике Δ_i и G^1-непрерывной.

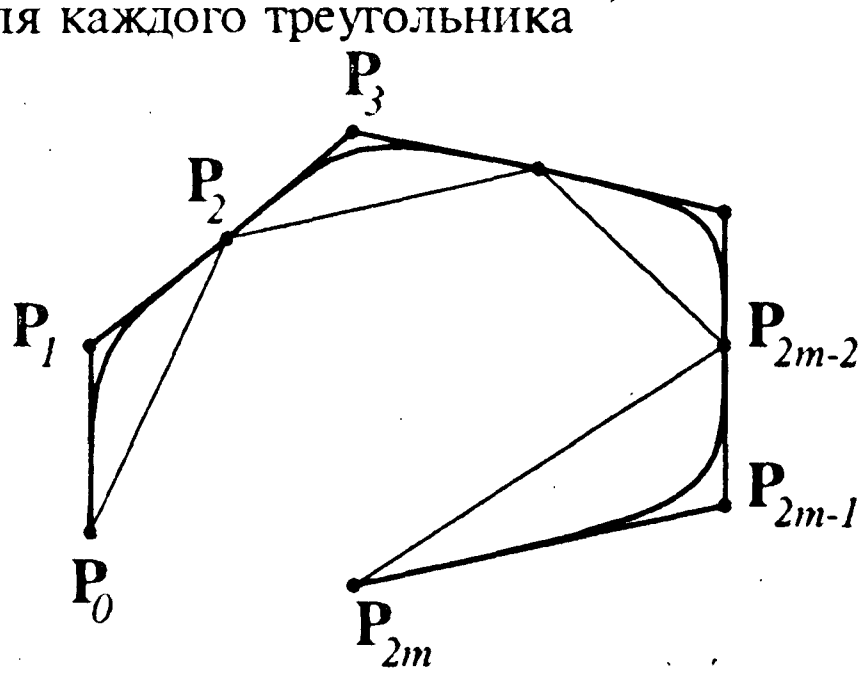


Рис. 3.34

Выбор можно сделать универсальным для всех треугольников Δ_i. Например, положить $\beta_1 = \beta_2 = 1/2$, $h = -1/3$ и взять в качестве точек $\mathbf{A}_i$ центры тяжести треугольников Δ_i.

Глава 4. Сплайновые поверхности

Общую задачу, рассматриваемую в этой главе, можно сформулировать так: по заданному множеству вершин с учетом их нумерации построить гладкую поверхность, которая, плавно изменяясь, проходила бы вблизи этих вершин и удовлетворяла некоторым дополнительным условиям. Эти условия могут иметь различный характер. Например, можно потребовать, чтобы искомая поверхность проходила через все заданные вершины или, проходя через заданные вершины, касалась заданных направлений, имела заданную регулярность и т. п.

При отыскании подходящего решения задачи приближения важную роль играет многогранная поверхность, вершины которой совпадают с точками заданного набора. Эту многогранную поверхность называют *контрольной* или *опорной*. Во многих случаях она довольно точно показывает, как будет проходить искомая поверхность, что особенно полезно при решении задачи сглаживания.

Мы будем рассматривать массивы, организованные как двумерные графы, топологически эквивалентные правильным прямоугольным сеткам. Сказанное означает, что в массиве

$$\begin{array}{cccc} \mathbf{P}_{00}, & \mathbf{P}_{01}, & \cdots & \mathbf{P}_{0n}, \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{P}_{m0}, & \mathbf{P}_{m1}, & \cdots & \mathbf{P}_{mn} \end{array}$$

наряду с вершинами $\mathbf{P}_{ij}$ учитываются еще и связывающие их ребра

$$\mathbf{P}_{ij}\mathbf{P}_{i,j+1}, \quad i = 0, 1, ..., m - 1, m; \qquad j = 0, 1, ..., n - 1,$$

$$\mathbf{P}_{ij}\mathbf{P}_{i+1,j}, \quad i = 0, 1, ..., m - 1; \qquad j = 0, 1, ..., n - 1, n$$

(рис. 4.1). Вершины

$\mathbf{P}_{ij}$, $i = 1, ..., m - 1; j = 1, ..., n - 1,$

называются *внутренними*, а вершины

$\mathbf{P}_{0j}$ $\quad j = 0, 1, ..., n - 1,$

$\mathbf{P}_{in}$ $\quad i = 0, 1, ..., m - 1,$

$\mathbf{P}_{mj}$ $\quad j = 1, 2, ..., n,$

$\mathbf{P}_{i0}$ $\quad i = 1, 2, ..., m,$

– *граничными*. Каждая внутренняя вершина имеет 4 соседние вер-

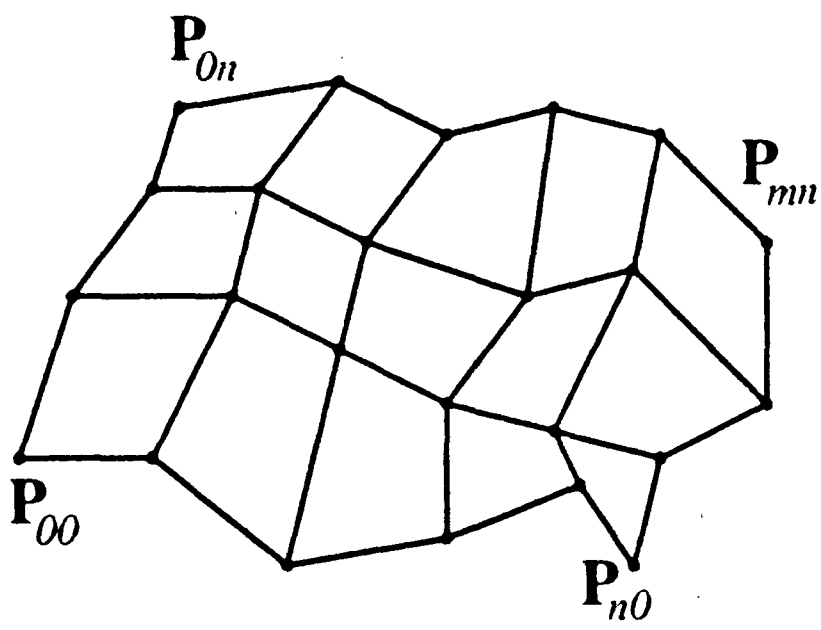


Рис. 4.1

шины, а каждая граничная - 3, за исключением четырех *угловых* вершин

$$\mathbf{P}_{00},\quad \mathbf{P}_{0n},\quad \mathbf{P}_{mn},\quad \mathbf{P}_{m0},$$

у каждой из которых только две соседние вершины.

Так организованный массив часто называют *контрольным* или *опорным графом* искомой сплайновой поверхности.

В отличие от ситуации, рассматриваемой в гл. 2, в данном случае ограничения на множество вершин являются менее жесткими. Поэтому довольно естественно искать описание поверхности в более общей, параметрической форме.

Для отыскания подходящих параметрических уравнений нужной поверхности удобно воспользоваться способом, естественно связанным с интуитивным представлением о поверхности как о множестве точек, которое "заметает" кривая, движущаяся вдоль другой фиксированной кривой и изменяющая свою форму. Кроме того, желательно учесть и структуру опорного графа.

Попытаемся прояснить геометрический смысл параметрического уравнения вида

$$\mathbf{R}(u, v) = \sum_{i=0}^{m}\sum_{j=0}^{n} a_i(u) b_j(v) \mathbf{P}_{ij}, \qquad (4.1)$$

где $a_i(u)$ и $b_j(v)$ - некоторые функциональные коэффициенты.

Положив

$$\mathbf{R}_i(v) = \sum_{j=0}^{n} b_j(v) \mathbf{P}_{ij}, \qquad i = 0, \ldots, m,$$

это уравнение можно переписать так:

$$\mathbf{R}(u, v) = \sum_{i=0}^{m} a_i(u) \mathbf{R}_i(v).$$

Первое из этих соотношений будем рассматривать как параметрическое уравнение i-й кривой ($i = 0, \ldots, m$). Зафиксировав переменную v, $v = v_0 = \mathrm{const}$, получим на каждой из этих кривых по точке $\mathbf{R}_i(v_0)$. По набору полученных точек определим кривую при помощи параметрического уравнения

$$\mathbf{R}(u, v_0) = \sum_{i=0}^{m} a_i(u) \mathbf{R}_i(v_0)$$

(рис. 4.2).

Говорят, что уравнение (4.1) есть задание поверхности в виде *тензорного произведения*. Такие поверхности имеют целый ряд преиму-

ществ, одно из которых особенно существенно для наших целей, ибо позволяет перенести на двумерный случай многие результаты и наблюдения предыдущей главы, посвященной кривым. Дело в том, что при повышении размерности задачи неизбежно возникает большое число новых проблем. Ограничения на структуру опорных массивов и выбор в качестве рабочего одного из наиболее простых классов поверхностей позволяют удержать это число в рамках, разумных для первого знакомства.

Если количество вершин в заданном множестве **P** достаточно велико, то найти универсальные функциональные коэффициенты $c_{ij}(u, v) = a_i(u)b_j(v)$, как правило, довольно затруднительно. К тому же может оказаться (а часто так и бывает), что найденные функции $c_{ij}(u, v)$ наряду с нужными свойствами обладают и такими, которые не всегда удовлетворительно согласуются с ожидаемым поведением соответствующей поверхности (например, поверхность, описываемая уравнением (4.1), с этими коэффициентами может осциллировать или отклоняться от заданного множества местами очень заметно).

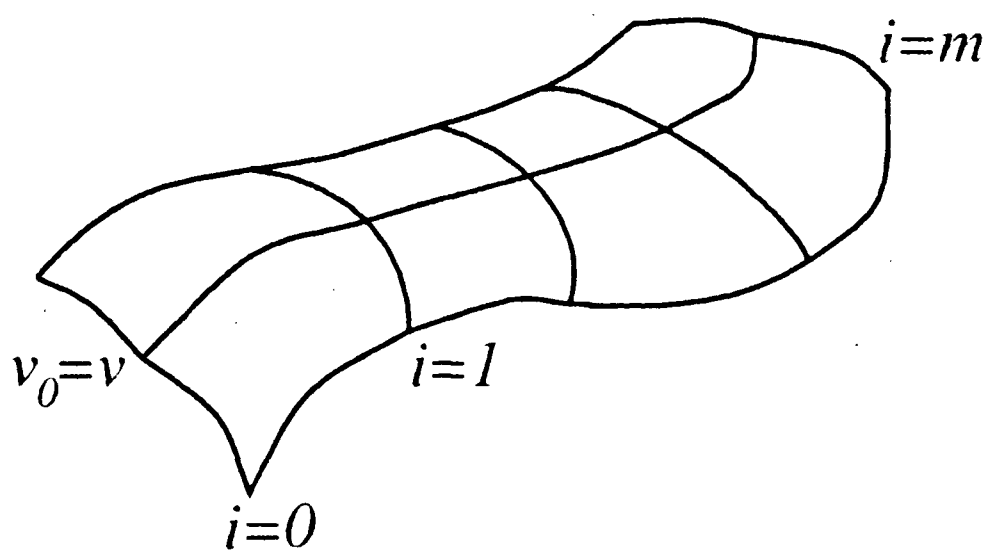


Рис. 4.2

Для успешного решения поставленной задачи приближения весьма удобно привлечь поверхности, составленные из элементарных фрагментов. В случае, когда эти элементарные фрагменты строятся по единой сравнительно простой схеме, такие составные поверхности принято называть *сплайновыми поверхностями.*

Параметрические уравнения каждого элементарного фрагмента ищутся в виде (4.1) с той лишь разницей, что всякий раз привлекается только часть заданных вершин множества **P**. При этом в качестве функциональных коэффициентов $c_{ij}(u, v)$ используются многочлены одинаковой степени.

Для описания элементарных поверхностей и вычисления их геометрических характеристик (необходимых при состыковке) часто используют функциональные коэффициенты, в которых функции $a_i(u)$ и $b_j(v)$ - многочлены, как правило, невысоких степенией, второй или третьей. Конечно, привлекая многочлены больших степеней, можно описывать весьма сложные поверхности. Однако у таких многочленов

много коэффициентов, физический и геометрический смысл которых трудно понять. Кроме того, использование многочленов высокой степени может вызвать нежелательные колебания результирующей поверхности.

Наибольшее распространение получили методы создания составных поверхностей, для построения которых используются бикубические многочлены (которые, кстати, активно использовались в гл. 2). Выбор в качестве функциональных коэффициентов бикубических многочленов позволяет учесть многие дифференциальные и внешнегеометрические требования, накладываемые на искомую поверхность.

Высказанные соображения определят характер изложения разделов этой главы.

Глава начинается с описания минимально необходимого набора сведений из дифференциальной геометрии поверхностей. Затем рассматриваются различные классы поверхностей, при помощи которых можно решить поставленную задачу приближения: определяются элементарные поверхности (будущие фрагменты) и описываются их основные свойства. Такой подход поможет пользователю в выборе наиболее подходящего для него класса поверхностей. Часть из свойств формулируется в виде ответов на стандартный набор вопросов, среди которых есть, например, такие:

- какова гладкость построенной поверхности?
- каковы особенности расположения поверхности относительно заданных вершин опорного графа?
- как сказываются на форме поверхности изменения опорных вершин?

Полезно также знать как связаны между собой форма поверхности и простейшие геометрические преобразования (аффинные и проективные).

Напомним, что аффинные преобразования включают в себя вращение, растяжение и сжатие, параллельный перенос и их всевозможные комбинации, а к проективным преобразованиям, кроме того, относятся еще и преобразования перспективы.

Большинство из рассматриваемых классов поверхностей обладают свойством аффинной инвариантности. Есть среди них и проективно инвариантные. Хотя окончательный результат и не зависит от последовательности выполнения этих двух операций - преобразования набора опорных точек и построения поверхности, выбор порядка их исполнения может заметно снизить вычислительные затраты.

Использование рациональных поверхностей вносит в решение задачи приближения нужную гибкость, ибо в их описании принимают участие свободные числовые параметры. Возможность выбора позволяет заметно влиять на форму искомых поверхностей, а также учитывать предписанные свойства. Важно и то, что рациональные поверхности сохраняют многие из свойств, которыми обладают соответствующие нерациональные (полиномиальные) поверхности. А то обстоятельство, что рациональные поверхности ко всему еще и проективно-инвариантны, является одним из наиболее привлекательных их свойств, весьма полезных при визуализации.

Кроме того, показывается, как путем состыковки этих элементарных поверхностей можно получить составную поверхность, удовлетворяющую требуемым условиям, обсуждаются различные вопросы, связанные с программной реализацией рассматриваемых подходов (их особенности, преимущества и недостатки, вопросы устойчивости).

4.1. Элементарные сведения из геометрии поверхностей

4.1.1. Параметризованные поверхности

Параметрически заданной поверхностью называется множество S точек $M(x, y, z)$ пространства, декартовы координаты x, y и z которых определяются посредством соотношений

$$x = x(u, v), \qquad y = y(u, v), \qquad z = z(u, v),$$
$$a \le u \le b, \quad c \le v \le d,$$

где $x(u, v)$, $y(u, v)$, $z(u, v)$ - функции, непрерывные в прямоугольнике

$$R = \{(u, v) \mid a \le u \le b, c \le v \le d\},$$

или *в векторно-матричной форме*:

$$\mathbf{R} = \mathbf{R}(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix},$$
$$(u, v) \in R.$$

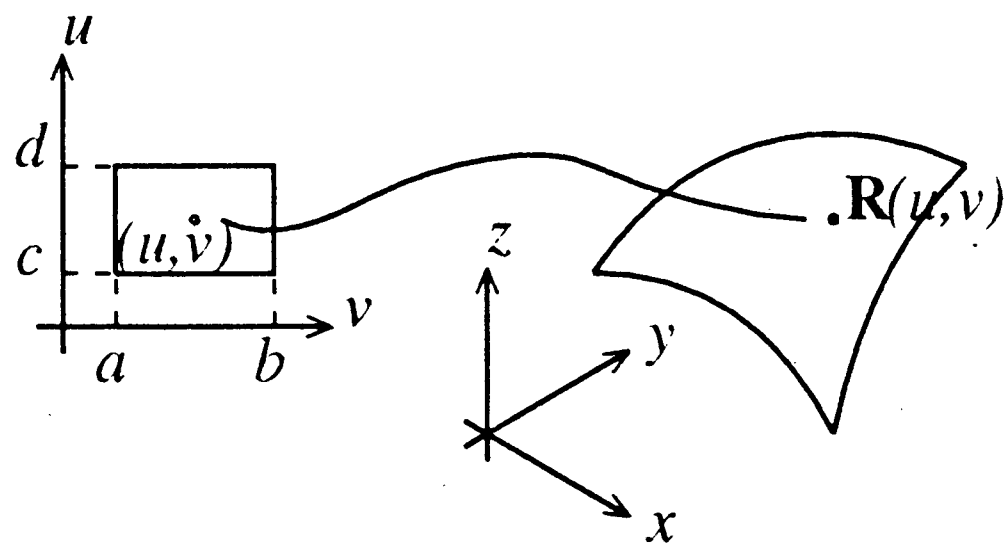


Рис. 4.3

Эти соотношения называют *параметрическими уравнениями поверхности* S или просто *параметризацией поверхности* S (рис. 4.3).

Величины u и v называются внутренними криволинейными координатами на поверхности S. При $v = v_0 = const$ векторное уравнение

$$\mathbf{R} = \mathbf{R}(u, v_0), \quad a \le u \le b,$$

описывает на поверхности S кривую - *линию u.*

Аналогично при $u = u_0 = const$ получаем векторное уравнение

$$\mathbf{R} = \mathbf{R}(u_0, v), \quad c \le v \le d,$$

линии v. Линии u и v образуют на поверхности S *координатную сеть* (рис. 4.4).

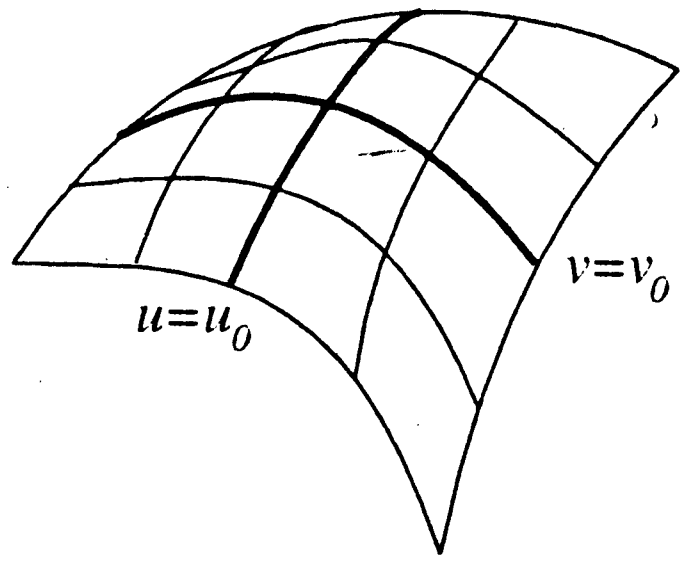


Рис. 4.4

4.1.2. Гладкие и регулярные поверхности

Поверхность S называется *C^r-гладкой относительно заданной параметризации*, если векторная функция $\mathbf{R}(u,v)$ является C^r-гладкой в прямоугольнике R, то есть каждая из координатных функций $x(u, v)$, $y(u, v)$, $z(u, v)$ имеет в этом прямоугольнике непрерывные производные до порядка r включительно.

Гладкая параметризация называется *регулярной*, если вектор

$$\mathbf{R}_u(u, v) \times \mathbf{R}_v(u, v) \neq \mathbf{O}, \quad (u, v) \in R.$$

Векторы $\mathbf{R}_u(u, v)$ и $\mathbf{R}_v(u, v)$ регулярной поверхности S лежат в ее *касательной плоскости* в точке $\mathbf{R}(u, v)$ и касаются в ней координатных линий u и v, а единичный вектор

$$\mathbf{N} = \frac{\mathbf{R}_u \times \mathbf{R}_v}{|\mathbf{R}_u \times \mathbf{R}_v|}$$

параллелен *нормали* поверхности S в этой точке (рис. 4.5).

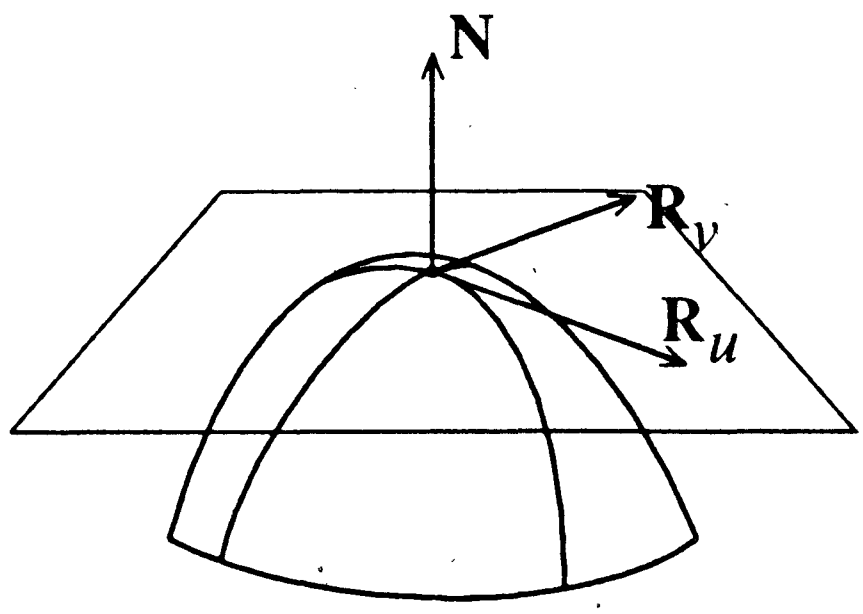


Рис. 4.5

В каждой точке регулярной поверхности определена тройка некомпланарных векторов

$$\mathbf{R}_u, \quad \mathbf{R}_v, \quad \mathbf{N}.$$

4.1.3. Первая квадратичная форма поверхности

Первой квадратичной формой регулярной поверхности S называется скалярный квадрат вектора $d\mathbf{R} = \mathbf{R}_u du + \mathbf{R}_v dv$:

$$I = d\mathbf{R}^2 = (\mathbf{R}_u du + \mathbf{R}_v dv) = Edu^2 + 2Fdudv + Gdv^2,$$

где

$$E = E(u, v) = \mathbf{R}_u^2, \quad F = F(u, v) = \mathbf{R}_u\mathbf{R}_v, \quad G = G(u, v) = \mathbf{R}_v^2$$

- *коэффициенты 1-й квадратичной формы*, связанные неравенством $EG - F^2 > 0$.

4.1.4. Кривая на поверхности

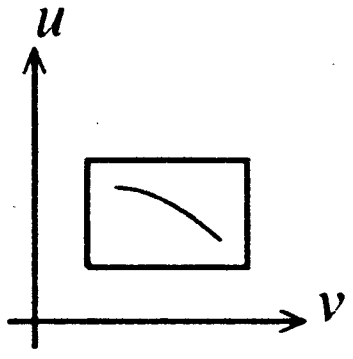
Параметрические уравнения

$$u = u(t), \quad v = v(t), \quad \alpha \le t \le \beta,$$

определяют на поверхности S кривую γ, описываемую векторным уравнением

$$\gamma: \quad \mathbf{R} = \mathbf{R}(t) = \mathbf{R}(u(t), v(t)), \quad \alpha \le t \le \beta.$$

(рис. 6). Уравнения $u = u(t)$, $v = v(t)$ называются *внутренними уравнениями кривой* γ.



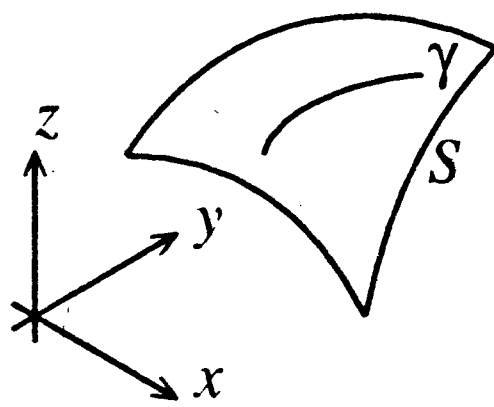


Рис. 4.6

На регулярной поверхности S кривая γ регулярна при условии, что

$$\dot{u}^2 + \dot{v}^2 > 0.$$

Касательный вектор кривой γ вычисляется по формуле

$$\dot{\mathbf{R}} = \mathbf{R}_u\dot{u} + \mathbf{R}_v\dot{v},$$

а длина дуги -

$$s = s(\xi) = \int_\alpha^\xi \left|\dot{\mathbf{R}}(t)\right| dt = \int_\alpha^\xi \sqrt{E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2}\, dt.$$

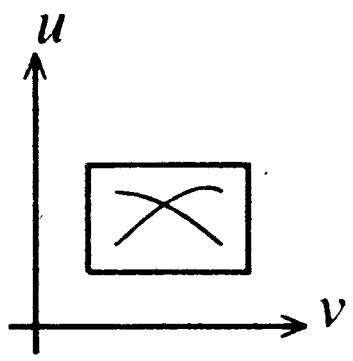
4.1.5. Угол между кривыми на поверхности

Пусть две кривые, описываемые внутренними уравнениями

$$u = u_1(t), \quad v = v_1(t),$$

$$u = u_2(t), \quad v = v_2(t),$$

пересекаются в точке $M(t)$ параметрической плоскости, отвечающей значению t параметра.



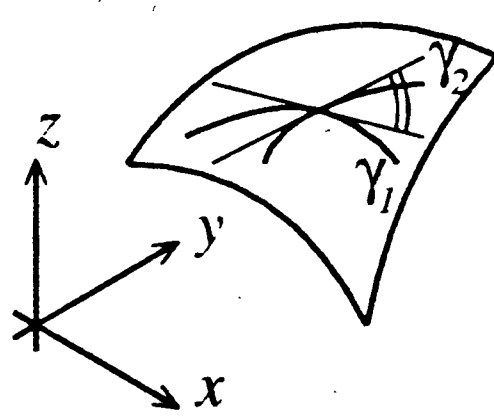


Рис. 4.7

Соответствующие кривые на поверхности S,

$$\gamma_1: \mathbf{R} = \mathbf{R}(u_1(t), v_1(t));$$

$$\gamma_2 : \mathbf{R} = \mathbf{R}(u_2(t), v_2(t)),$$

пересекаются в точке $\mathbf{R}(t)$ этой поверхности под прямым углом (рис. 4.7), если

$$E\dot{u}_1\dot{u}_2 + F(\dot{u}_1\dot{v}_2 + \dot{u}_2\dot{v}_1) + G\dot{v}_1\dot{v}_2 = 0.$$

Координатные линии u и v поверхности S ортогональны тогда и только тогда, когда $F(u, v)=0$.

4.1.6. Площадь поверхности

Площадь части поверхности S, отвечающей области Ω на параметрической плоскости (u, v) (рис. 4.8), равна

$$S = \int\int_{\Omega} \sqrt{EG - F^2}\, du dv .$$

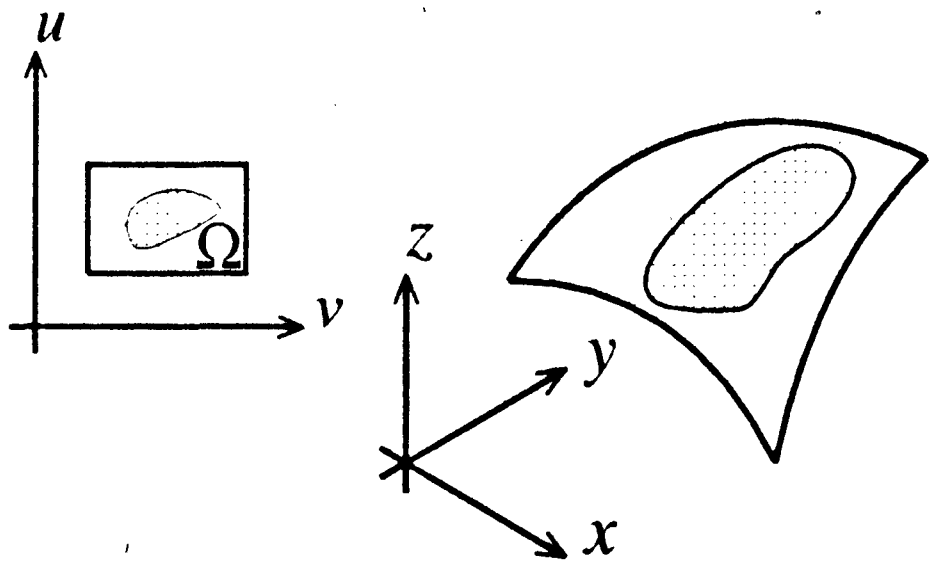


Рис. 4.8

4.1.7. Вторая квадратичная форма поверхности

Второй квадратичной формой C^2-регулярной поверхности S называется скалярное произведение векторов $d^2\mathbf{R}$ и $\mathbf{N}$:

$$II = \left(d^2\mathbf{R}\mathbf{N}\right) = Ldu^2 + 2Mdudv + Ndv^2,$$

где

$$L = \frac{(\mathbf{R}_{uu}\mathbf{R}_u\mathbf{R}_v)}{\sqrt{EG - F^2}} = (\mathbf{R}_{uu}\mathbf{N}),$$

$$M = \frac{(\mathbf{R}_{uv}\mathbf{R}_u\mathbf{R}_v)}{\sqrt{EG - F^2}} = (\mathbf{R}_{uv}\mathbf{N}),$$

$$N = \frac{(\mathbf{R}_{vv}\mathbf{R}_u\mathbf{R}_v)}{\sqrt{EG - F^2}} = (\mathbf{R}_{vv}\mathbf{N}).$$

4.1.8. Линии кривизны

Кривизна кривой, лежащей на поверхности S и проходящей через точку $\mathbf{R}(u, v)$ в направлении $\lambda = (du : dv)$ (рис. 4.9), вычисляется по формуле

$$k(\lambda) = k(\mathbf{R}, \lambda) = \frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2}.$$

Возможны два случая.

1. Функция $k(\mathbf{R}, \lambda)$ не зависит от направления λ в точке **R**. Тогда точка **R** называется либо *омбилической (шаровой) точкой* либо *точкой уплощения* поверхности *S*.
   Все точки сферы - омбилические, все точки плоскости - точки уплощения.

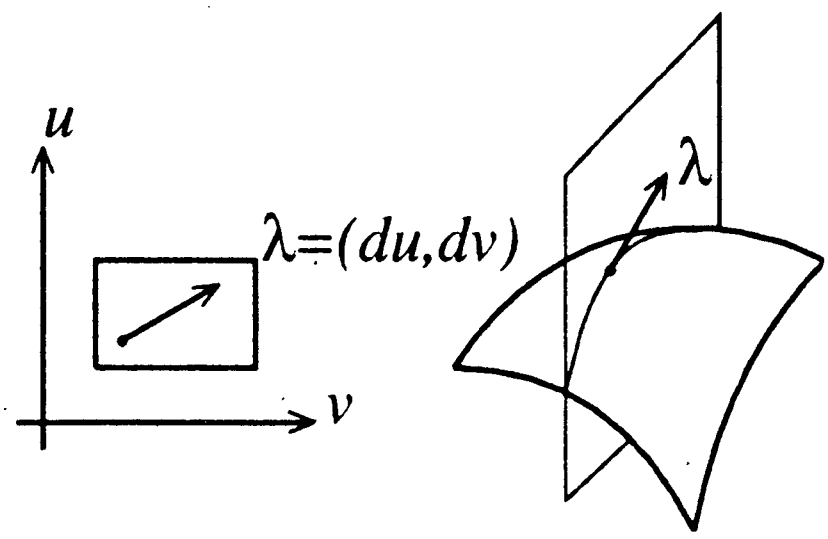


Рис. 4.9

2. Функция $k(\mathbf{R}, \lambda)$ имеет ровно два экстремума: k_- и k_+ - *главные кривизны* поверхности *S* в данной точке. Соответствующие направления на поверхности называются *главными*.

Сеть кривых, направления которых в каждой точке поверхности главные, называется *сетью линий кривизны*. Если $F = 0$ и $M = 0$, то сеть координатных линий является сетью линий кривизны.

4.1.9. Гауссова и средняя кривизны

Произведение главных кривизн в данной точке поверхности называется ее *гауссовой кривизной* в этой точке,

$$K = k_- k_+ = \frac{LN - M^2}{EG - F^2},$$

а их полусумма - ее *средней кривизной*,

$$H = \frac{1}{2} = \frac{NE - 2MF - LG}{EG - F^2}.$$

В каждой точке регулярной поверхности гауссова и средняя кривизны описывают характер и степень отклонения этой поверхности от ее касательной плоскости в рассматриваемой точке. Точка поверхности называется *эллиптической*, если в ней $K > 0$, *гиперболической*, если $K < 0$, и *параболической* при $K = 0$.

4.1.10. Геометрическая непрерывность

При работе с поверхностями важно помнить о следующих двух обстоятельствах. С одной стороны, поверхность как множество точек в пространстве может иметь прекрасные геометрические характеристики, но описываться при этом достаточно плохими параметрическими уравнениями. С другой стороны, хорошие дифференциальные свойства координатных функций, задающих поверхность, не всегда обеспечивают ее регулярность.

Рассмотрим некоторые примеры.

Пример 1

Поверхность S задана параметрическими уравнениями

$$\mathbf{R} = \mathbf{R}(u, v) = \begin{pmatrix} u^5 \cos v \\ u^5 \sin v \\ u^5 \end{pmatrix},$$

$$-1 \le u \le 1, \quad 0 \le v \le 2\pi$$

(рис. 4.10).

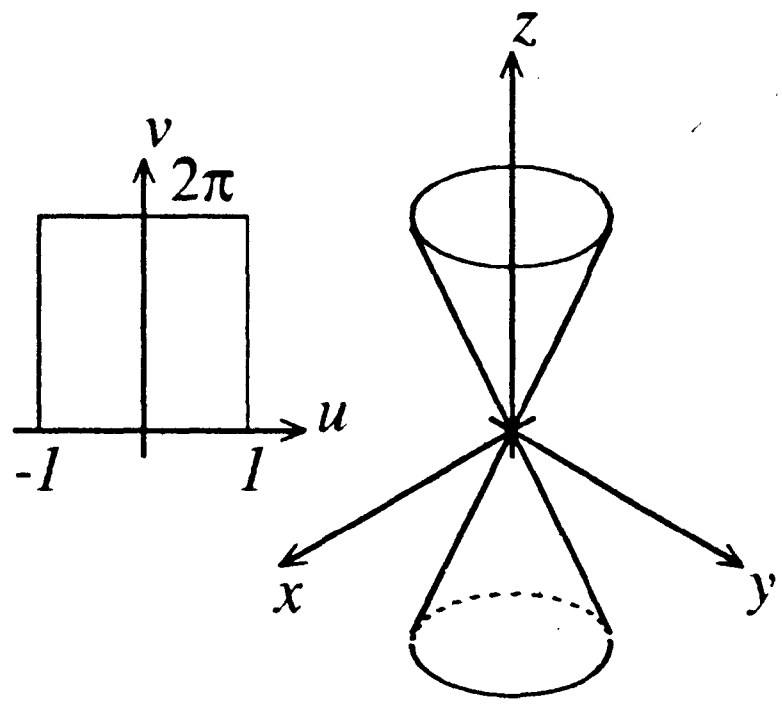


Рис. 4.10

Результат: координатные функции имеют производные всех порядков, а поверхность S имеет особенность - *коническую точку.*

Пример 2

Поверхность S задана параметрическими уравнениями

$$\mathbf{R} = \mathbf{R}(u, v) = \begin{pmatrix} v \\ u^5 \\ u^4|u| \end{pmatrix},$$

$$-1 \le u \le 1, \quad -1 \le v \le 1$$

(рис. 4.11).

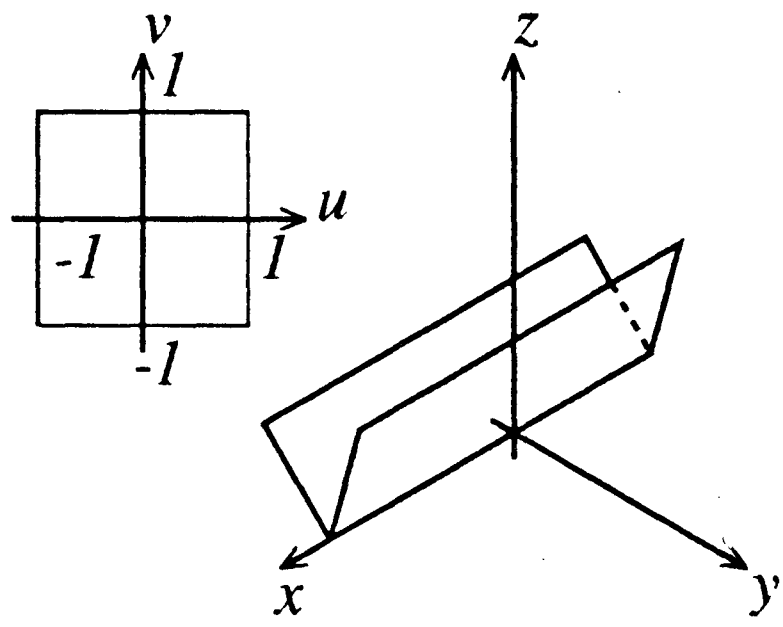


Рис. 4.11

Результат: координатные функции имеют непрерывные производные до 4-го порядка включительно, а поверхность S имеет особенность - *ребро.*

Причина особенностей - $\mathbf{R}_u \times \mathbf{R}_v = \mathbf{O}$ - в обоих примерах одна и та же.

Требование регулярности

$$\mathbf{R}_u \times \mathbf{R}_v \neq \mathbf{O}$$

обеспечивает отсутствие у поверхности и ребер, и конических точек.

При построении составных поверхностей наиболее типичной является ситуация, когда каждая из регулярных поверхностей, участвующих в процессе создания новой поверхности, имеет собственную параметризацию. Несогласованность параметризаций может явиться (и часто является) причиной особенностей, возникающих вдоль линий сопряжения. Это - *особенности параметризации.*

Предположим, что заданы две регулярные поверхности $S^{(1)}$ и $S^{(2)}$, описываемые соответственно параметрическими уравнениями

$$\mathbf{R} = \mathbf{R}^{(1)}(u, v), \quad a^{(1)} \le u \le b^{(1)}, \quad c^{(1)} \le v \le d^{(1)},$$

$$\mathbf{R} = \mathbf{R}^{(2)}(u, v), \quad a^{(2)} \le u \le b^{(2)}, \quad c^{(2)} \le v \le d^{(2)},$$

и такие, что их граничные кривые имеют общий участок (рис. 4.12). В этом случае составная поверхность $S = S^{(1)} \cup S^{(2)}$ называется *непрерывной* или G^0 *-непрерывной.*

G^0 -непрерывная составная поверхность $S = S^{(1)} \cup S^{(2)}$ называется G^1 *-непрерывной*, или G^1 *-поверхностью*, если ее касательная плоскость при переходе через общую кривую изменяется непрерывно.

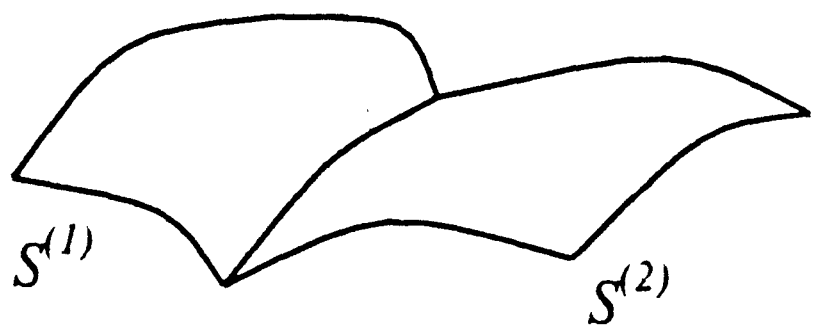


Рис. 4.12

Пример 3

Составная поверхность, заданная параметрическими уравнениями

$$\mathbf{R} = \mathbf{R}^{(1)}(u, v) = \begin{pmatrix} u \\ v \\ 0 \end{pmatrix}, \quad 0 \le u \le 1, \quad 0 \le v \le 1,$$

$$\mathbf{R} = \mathbf{R}^{(2)}(u, v) = \begin{pmatrix} (1+u)^2 \\ v \\ 0 \end{pmatrix}, \quad 0 \le u \le 1, \quad 0 \le v \le 1,$$

является G^1 -поверхностью. Это просто вырезок координатной плоскости *0xy* (рис. 4.13).

У каждой G^1 -поверхности существует такая параметризация, относительно которой эта поверхность является C^1 -регулярной.

G^1 -непрерывная составная поверхность $S = S^{(1)} \cup S^{(2)}$ называется G^2 *-непрерывной* или G^2 *-поверхностью*, если ее гауссова кривизна при переходе через

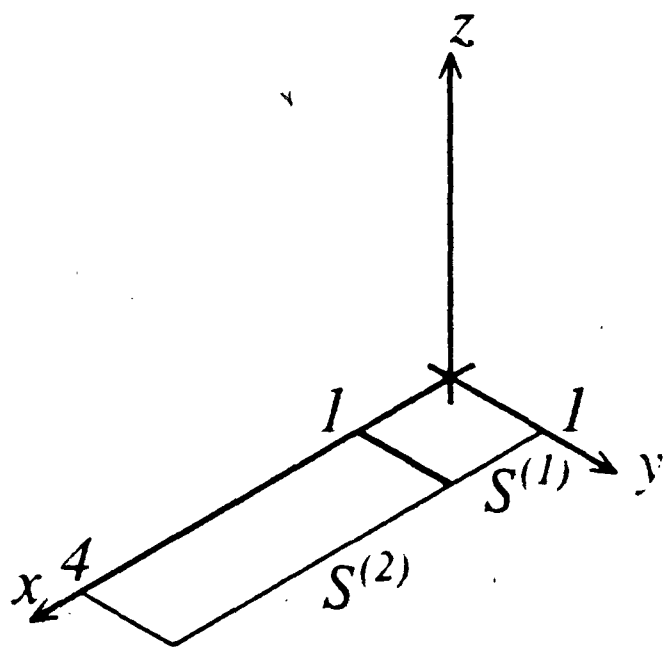


Рис. 4.13

общую кривую изменяется непрерывно.

Пример 4

Составная поверхность, заданная параметрическими уравнениями

$$\mathbf{R} = \mathbf{R}^{(1)}(u, v) = \begin{pmatrix} u \\ u^3 \\ v \end{pmatrix}, \quad 0 \le u \le 1, \quad 0 \le v \le 1,$$

$$\mathbf{R} = \mathbf{R}^{(2)}(u, v) = \begin{pmatrix} -\frac{2}{\pi}\cos\left(\frac{\pi}{2}u\right) \\ -\left(\frac{2}{\pi}\right)^3 \cos^3\left(\frac{\pi}{2}u\right) \\ \frac{1}{2}u^2 \end{pmatrix},$$

$$0 \le u \le 1, \quad 0 \le v \le 1,$$

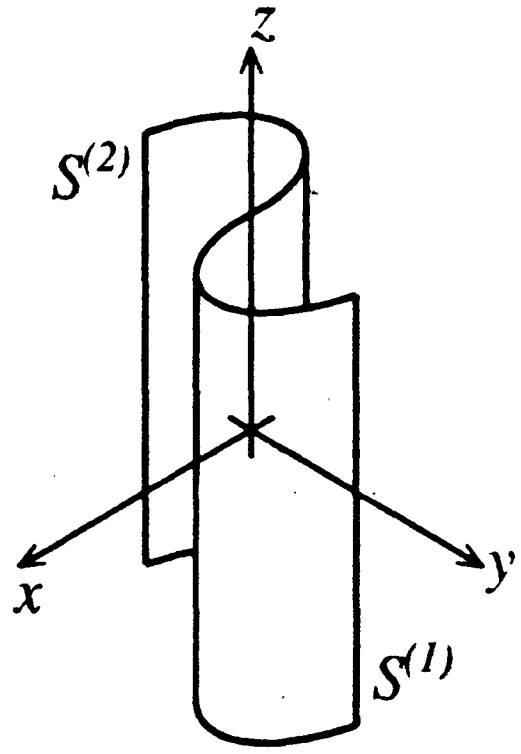


Рис. 4.14

является G^2-поверхностью (рис. 4.14).

У каждой G^2-поверхности существует такая параметризация, относительно которой эта поверхность является C^2-регулярной.

4.1.11. Вектор скручивания и билинейная поверхность

Вектором скручивания регулярной поверхности S, заданной параметрическим векторным уравнением $\mathbf{R} = \mathbf{R}(u, v)$, называется смешанная производная $\mathbf{R}_{uv}$.

Геометрический смысл введенного вектора скручивания лучше всего пояснить на примере билинейной поверхности.

Билинейной поверхностью, порожденной массивом из четырех некомпланарных вершин $\mathbf{P}_{00}$, $\mathbf{P}_{01}$, $\mathbf{P}_{10}$, $\mathbf{P}_{11}$, называется поверхность $\mathbf{B}$, описываемая параметрическими уравнениями

$$\mathbf{R}(u, v) = \begin{pmatrix} 1-u & u \end{pmatrix} \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}$$

$$0 \le u, v \le 1$$

(рис. 4.15).

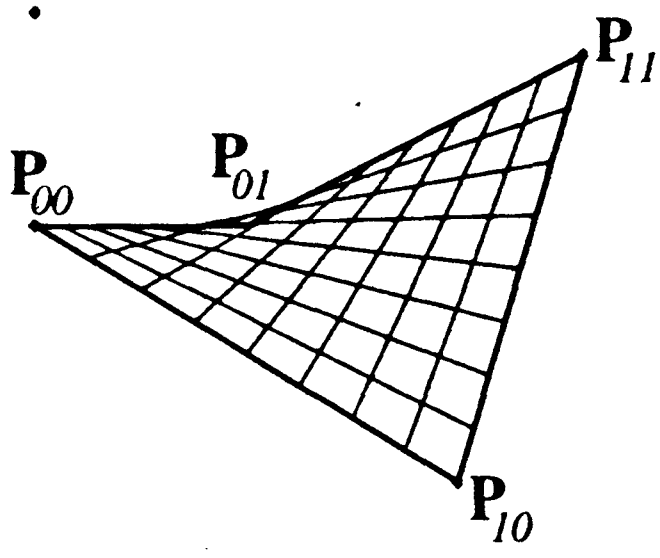


Рис. 4.15

Граничные кривые поверхности **B** суть прямолинейные отрезки

$$\mathbf{P}_{00}\mathbf{P}_{10},\quad \mathbf{P}_{10}\mathbf{P}_{11},\quad \mathbf{P}_{11}\mathbf{P}_{01},\quad \mathbf{P}_{01}\mathbf{P}_{00}.$$

Вектор скручивания билинейной поверхности **B** легко вычисляется:

$$\mathbf{R}_{uv}(u,v) = \mathbf{P}_{00} - \mathbf{P}_{01} + \mathbf{P}_{11} - \mathbf{P}_{10}.$$

Он показывает меру отклонения точки $\mathbf{P}_{11}$ от касательной плоскости к поверхности **B** в точке $\mathbf{P}_{00}$ (рис. 4.16).

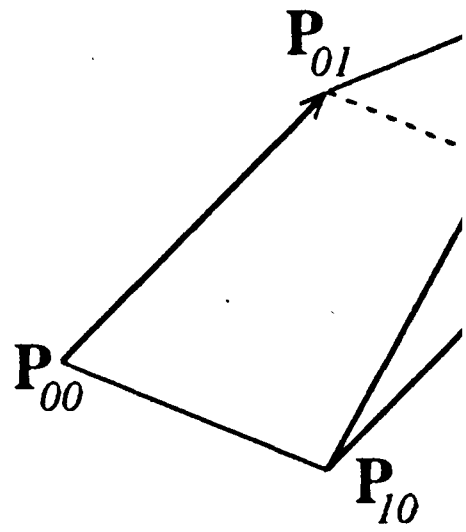


Рис. 4.16

Важное замечание

Значение смешанной производной радиуса-вектора зависит ко (и не столько) от геометрических свойств поверхности, способа ее параметризации.

Пример. Параметрическое уравнение

$$\mathbf{R}(u,v) = u\mathbf{i} + v\mathbf{j},\quad 0 \le u,v \le 1,$$

где **i**, **j** - орты координатных осей *Ox* и *Oy*, описывает единичный квадрат на координатной плоскости *Oxy* (рис. 4.17). Вектор скручивания тождественно равен нулю $\mathbf{R}_{uv}(u, v) = \mathbf{O}$.

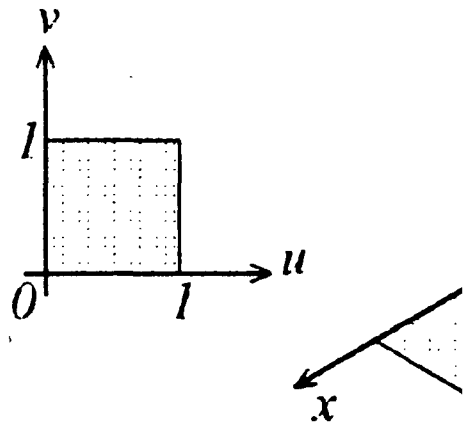


Рис. 4.17

Параметрическое уравнение

$$\mathbf{R}(u, v) = u\mathbf{i} + uv\mathbf{j},\quad 0 \le u, v \le 1,$$

описывает тот же квадрат, однако вектор $\mathbf{R}_{uv}(u, v) = \mathbf{j} \neq \mathbf{O}$.

В заключение отметим полезное соотношение

$$M = (\mathbf{R}_{uv}\mathbf{N}),$$

связывающее вектор скручивания регулярной поверхности с коэффициентов ее второй квадратичной формы.

4.2. Поверхности Безье

4.2.1. Параметрические уравнения поверхности Безье

По заданному массиву

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n\}$$

(элементарная) поверхность Безье определяется при помощи векторного уравнения, имеющего следующий вид:

$$\mathbf{R}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v) \mathbf{P}_{ij}, \quad 0 \le u, v \le 1,$$

где

$$B_i^m(u) = \binom{m}{i} u^i (1-u)^{m-i}, \quad B_j^n(v) = \binom{n}{j} v^j (1-v)^{n-j}$$

- многочлены Бернштейна.

Матричная запись параметрических уравнений, описывающих поверхность Безье, -

$$\begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \begin{pmatrix} u^0 & \ldots & u^m \end{pmatrix} \mathbf{M}^T \begin{pmatrix} \mathbf{P}_{00} & \cdots & \mathbf{P}_{0n} \\ \cdots & \cdots & \cdots \\ \mathbf{P}_{m0} & \cdots & \mathbf{P}_{mn} \end{pmatrix} \mathbf{M} \begin{pmatrix} v^0 \\ \vdots \\ v^n \end{pmatrix},$$

$$0 \le u, v \le 1,$$

где $\mathbf{M} = (\mu_{ij})$ и $\mathbf{N} = (\nu_{ij})$ - квадратные матрицы соответственно порядка m и n,

$$\mu_{ij} = (-1)^{j-i} \binom{m}{j} \binom{j}{i}, \quad \nu_{ij} = (-1)^{j-i} \binom{n}{i} \binom{j}{i}.$$

В случае, когда область изменения параметров u и v - произвольный прямоугольник вида

$$R = \{(u, v) \mid a \le u \le b, c \le v \le d\},$$

уравнение поверхности Безье принимает следующий вид:

$$\mathbf{R}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m\left(\frac{u-a}{b-a}\right) B_j^n\left(\frac{v-c}{d-c}\right) \mathbf{P}_{ij}.$$

Важный частный случай. При $m = n = 3$ имеем элементарную *бикубическую* поверхность Безье, определяемую 16 вершинами:

$$\mathbf{P}_{00},\ \mathbf{P}_{01},\ \mathbf{P}_{02},\ \mathbf{P}_{03},$$
$$\mathbf{P}_{10},\ \mathbf{P}_{11},\ \mathbf{P}_{12},\ \mathbf{P}_{13},$$
$$\mathbf{P}_{20},\ \mathbf{P}_{21},\ \mathbf{P}_{22},\ \mathbf{P}_{23},$$
$$\mathbf{P}_{30},\ \mathbf{P}_{31},\ \mathbf{P}_{32},\ \mathbf{P}_{33}.$$

Используя многочлены Бернштейна, эту поверхность можно задать так:

$$\mathbf{R}(u, v) = \sum_{i=0}^{3} B_i^3(u)\left(\sum_{j=0}^{3} B_j^3(v)\mathbf{P}_{ij},\right), \quad 0 \le u, v \le 1,$$

или в матричной форме:

$$\mathbf{R}(u, v) = (B_0^3(u)B_1^3(u)B_2^3(u)B_3^3(u)) \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{pmatrix} \begin{pmatrix} B_0^3(v) \\ B_0^3(v) \\ B_0^3(v) \\ B_0^3(v) \end{pmatrix}.$$

Последнюю формулу часто записывают и так:

$$\mathbf{R}(t) = \mathbf{U}^T\mathbf{M}^T\mathbf{PMV}, \quad 0 \le u, v \le 1,$$

где

$$\mathbf{R}(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} u^0 \\ u^1 \\ u^2 \\ u^3 \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} v^0 \\ v^1 \\ v^2 \\ v^3 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} 1 & -3 & 3 & 1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Матрица **M** называется *базисной матрицей бикубической поверхности Безье.*

4.2.2. Свойства элементарных поверхностей Безье

Свойства элементарных поверхностей Безье являются прямыми следствиями свойств элементарных кривых Безье. Укажем некоторые из них.

Основные свойства элементарной поверхности Безье

Элементарная поверхность Безье, порожденная массивом **P**:

1⁺ является гладкой поверхностью, в частности 1-е производные радиуса-вектора $\mathbf{R}(u, v)$ можно записать так:

$$\mathbf{R}_u(u, v) = \frac{\partial}{\partial u}\mathbf{R}(u, v) = m\sum_{i=0}^{m-1}\sum_{j=0}^{n}(\mathbf{P}_{i+1,j} - \mathbf{P}_{i,j})B_i^{m-1}(u)B_j^n(v),$$

$$\mathbf{R}_v(u, v) = \frac{\partial}{\partial v}\mathbf{R}(u, v) = n\sum_{i=0}^{m}\sum_{j=0}^{n-1}(\mathbf{P}_{i,j+1} - \mathbf{P}_{i,j})B_i^m(u)B_j^{n-1}(v);$$

если векторы $\mathbf{R}_u(u, v)$ и $\mathbf{R}_v(u, v)$ неколлинеарны, то определен единичный вектор нормали поверхности Безье $\mathbf{N}(u, v)$;

2⁺ граничные кривые элементарной поверхности Безье суть элементарные кривые Безье соответствующих степеней, их опорные ломаные образуют границу опорной многогранной поверхности (опорного графа), в частности граничная кривая, описываемая радиусом-вектором $\mathbf{R}(0, v)$, является элементарной кривой Безье n-й степени с опорным массивом вершин $\mathbf{P}_{00}, \ldots, \mathbf{P}_{0n}$ (рис. 4.18); все 4 угловые вершины опорного многогранника $\mathbf{P}_{00}, \mathbf{P}_{0n}, \mathbf{P}_{m0}$ и $\mathbf{P}_{mn}$ лежат на поверхности Безье,

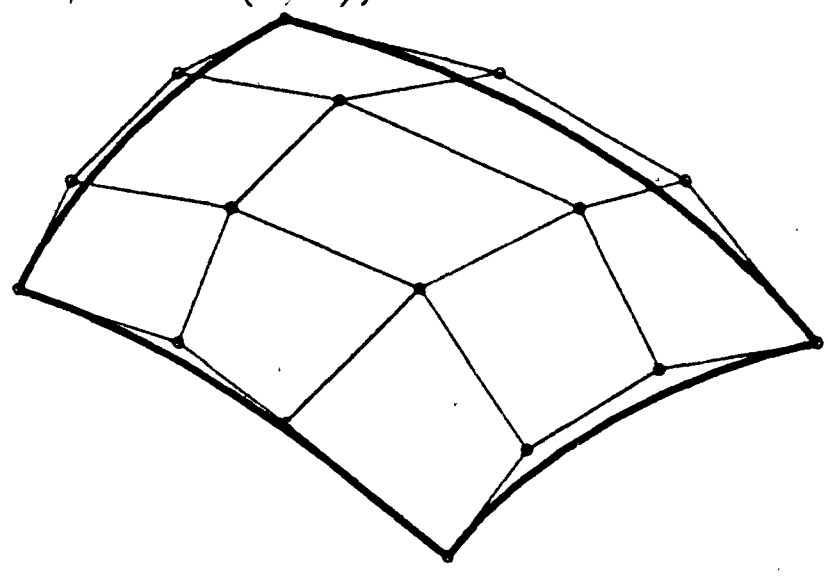


Рис. 4.18

$$\mathbf{R}(0, 0) = \mathbf{P}_{00}, \quad \mathbf{R}(1, 0) = \mathbf{P}_{m0}, \quad \mathbf{R}(0, 1) = \mathbf{P}_{0n}, \quad \mathbf{R}_u(1, 1) = \mathbf{P}_{mn},$$

и поверхность касается угловых граней опорного многогранника, например для угловой вершины $\mathbf{P}_{00}$ $(u = v = 0)$, $\mathbf{R}(0, 0) = \mathbf{P}_{00}$, выполняются равенства

$$\mathbf{R}_u(0, 0) = m(\mathbf{P}_{10} - \mathbf{P}_{00}), \quad \mathbf{R}_v(0,0) = n(\mathbf{P}_{01} - \mathbf{P}_{00})$$

(касательные векторы элементарной поверхности Безье в угловой вершине коллинеарны звеньям соответствующих граничных опорных ломаных, исходящих из этой вершины (рис. 4.19),

$$\mathbf{R}_{uv}(0, 0) = mn((\mathbf{P}_{11} - \mathbf{P}_{10}) - (\mathbf{P}_{01} - \mathbf{P}_{00}))$$

(вектор скручивания $\mathbf{R}_{uv}(0, 0)$ в угловой вершине $\mathbf{P}_{00}$ только множителем отличается от вектора скручивания билинейной поверхности, порожденной четверкой вершин

$$\mathbf{P}_{01},\ \mathbf{P}_{10},\ \mathbf{P}_{00},\ \mathbf{P}_{11},$$

и оценивает степень отклонения опорной вершины $\mathbf{P}_{11}$ от касательной плоскости элементарной поверхности Безье в вершине $\mathbf{P}_{00}$,

$$\mathbf{N}(0,0) = \frac{(\mathbf{P}_{10} - \mathbf{P}_{00}) \times (\mathbf{P}_{01} - \mathbf{P}_{00})}{\left|(\mathbf{P}_{10} - \mathbf{P}_{00}) \times (\mathbf{P}_{01} - \mathbf{P}_{00})\right|}$$

(вектор нормали перпендикулярен плоскости треугольника $\mathbf{P}_{00}\mathbf{P}_{01}\mathbf{P}_{10}$

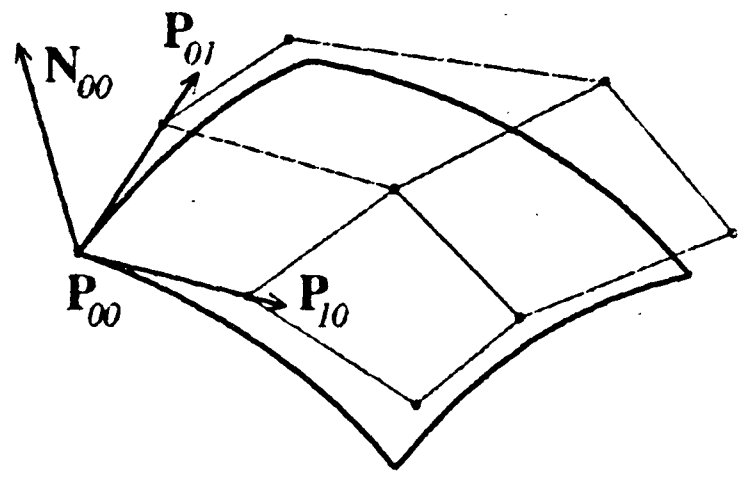


Рис. 4.19

(для каждой из трех других угловых вершин $\mathbf{P}_{01}(u=0, v=1)$, $\mathbf{P}_{10}$ $(u=1, v=0)$, $\mathbf{P}_{11}(u=v=1)$ - выполняются аналогичные соотношения);

3$^+$ элементарная поверхность Безье лежит в выпуклой оболочке, порожденной массивом **P**;

4$^+$ элементарная поверхность Безье аффинно-инвариантна;

5$^+$ элементарная поверхность Безье "повторяет" опорную многогранную поверхность;

6$^+$ если все вершины массива **P** лежат в одной плоскости, то определяемая этим массивом элементарная поверхность Безье представляет собой плоский криволинейный многоугольник, лежащий в этой плоскости;

7$^-$ степень функциональных коэффициентов напрямую связана с количеством вершин в массиве и растет при его увеличении;

8$^-$ при добавлении в массив новых вершин возникает необходимость полного пересчета параметрических уравнений элементарной поверхности Безье;

9$^-$ изменение хотя бы одной вершины в массиве приводит к заметному изменению всей поверхности Безье;

10$^-$ априорные сведения о расположении поверхности Безье (принадлежность выпуклой оболочке заданного массива вершин) являются достаточно грубыми;

11$^-$ в уравнениях, описывающих элементарную поверхность Безье, нет свободных параметров - заданный массив однозначно определяет поверхность Безье, не давая возможности хоть как-то влиять на ее форму;

12$^-$ элементарная поверхность Безье проективно-неинвариантна.

4.2.3. Составные поверхности Безье

В этом подразделе мы остановимся на построении составных бикубических поверхностях Безье вследствие того, что именно такие поверхности Безье наиболее часто используются в приложениях. Составные поверхности Безье более высоких степеней конструируются по аналогичной схеме.

Составная бикубическая поверхность Безье - это C^0-гладкая (непрерывная) поверхность, являющаяся объединением элементарных бикубических поверхностей Безье.

Пусть

$$\mathbf{R} = \mathbf{R}^{(1)}(u, v), \quad (u, v) \in [0, 1] \times [0, 1],$$

и

$$\mathbf{R} = \mathbf{R}^{(2)}(u, v), \quad (u, v) \in [0, 1] \times [0, 1],$$

- радиусы-векторы двух элементарных поверхностей Безье, соответственно $S^{(1)}$ и $S^{(2)}$, порожденные массивами

$$\mathbf{P}^{(1)} = \{\mathbf{P}_{ij}^{(1)}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n\},$$

и

$$\mathbf{P}^{(2)} = \{\mathbf{P}_{ij}^{(2)}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n\},$$

такими, что

$$\mathbf{P}_{mj}^{(1)} = \mathbf{P}_{0j}^{(2)}, \quad j = 0, \ldots, n.$$

Последнее означает, что поверхности $S^{(1)}$ и $S^{(2)}$ имеют общую граничную кривую

$$\mathbf{R}^{(1)}(1, v) = \mathbf{R}^{(2)}(0, v), \quad 0 \le v \le 1$$

(рис. 4.20).

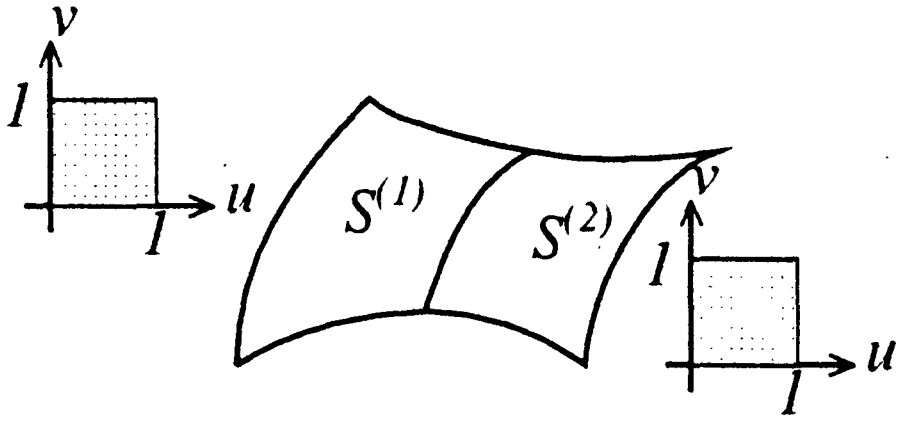


Рис. 4.20

C^1-гладкость составной поверхности $S = S^{(1)} \cup S^{(2)}$ обеспечивается коллинеарностью всех троек вида

$$\mathbf{P}_{m-1,j}^{(1)}, \quad \mathbf{P}_{mj}^{(1)} = \mathbf{P}_{0j}^{(2)}, \quad \mathbf{P}_{1j}^{(2)}$$

и выполнением условия

$$\frac{\left|\mathbf{P}_{mj}^{(1)} - \mathbf{P}_{m-1,j}^{(1)}\right|}{\left|\mathbf{P}_{1j}^{(2)} - \mathbf{P}_{0,j}^{(2)}\right|} = const$$

для всех j (рис. 4.21). Выполнение этого условия означает, что нормаль (или, что то же, касательная плоскость) составной поверхности при переходе через общую граничную кривую стыкуемых фрагментов изменяется непрерывно.

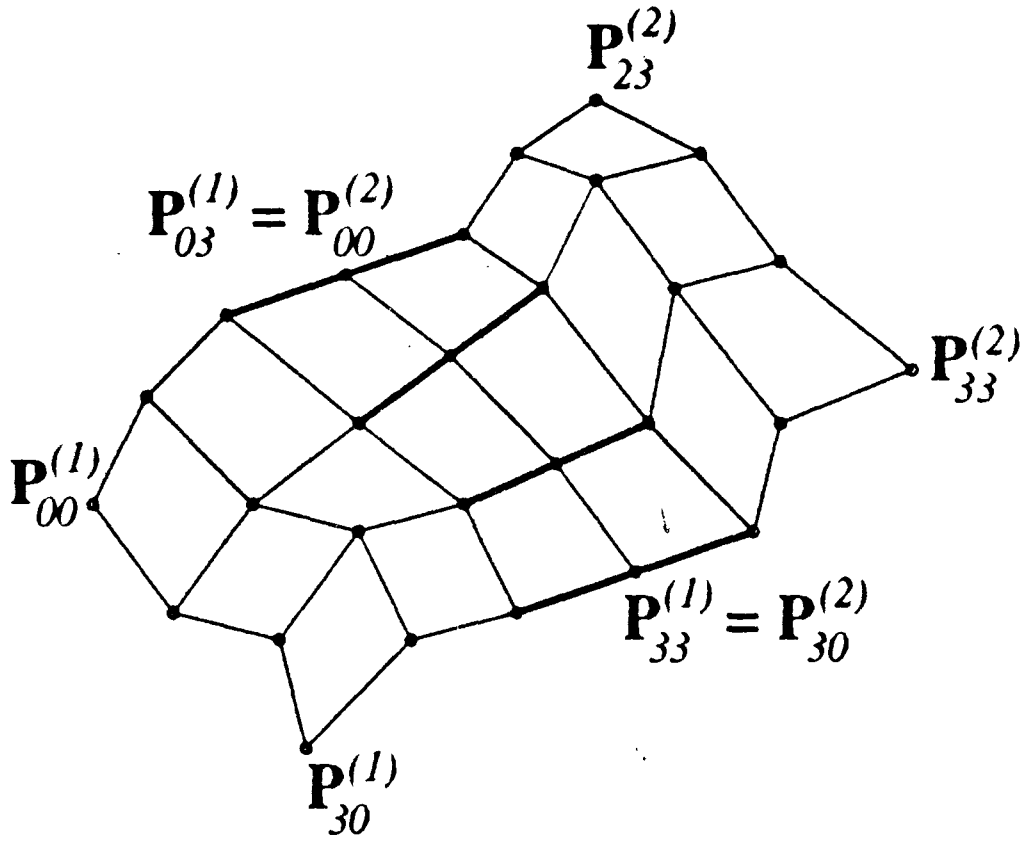


Рис. 4.21

4.2.4. Рациональные поверхности Безье

По заданному массиву

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n\}$$

(элементарная) рациональная поверхность Безье определяется уравнением следующего вида:

$$\mathbf{R}(u, v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_i^m(u) B_j^n(v) \mathbf{P}_{ij}}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_i^m(u) B_j^n(v)}, \quad 0 \le u, v \le 1,$$

где $B_i^m(u)$ и $B_j^n(v)$ - многочлены Бернштейна.

Неотрицательные числа w_{ij}, сумма которых положительна, называются *весами*. В случае, если все веса w_{ij} равны между собой, получается стандартная элементарная поверхность Безье.

Важный частный случай. При $m = n = 3$ имеем *элементарную бикубическую рациональную поверхность Безье*, порожденную массивом из 16 вершин

$$\begin{array}{cccc} \mathbf{P}_{00}, & \mathbf{P}_{01}, & \mathbf{P}_{02}, & \mathbf{P}_{03}, \\ \mathbf{P}_{10}, & \mathbf{P}_{11}, & \mathbf{P}_{12}, & \mathbf{P}_{13}, \\ \mathbf{P}_{20}, & \mathbf{P}_{21}, & \mathbf{P}_{22}, & \mathbf{P}_{23}, \\ \mathbf{P}_{30}, & \mathbf{P}_{31}, & \mathbf{P}_{32}, & \mathbf{P}_{33} \end{array}$$

и описываемую уравнением

$$\mathbf{R}(u, v) = \frac{1}{w(u, v)} \sum_{i=0}^{3} \sum_{j=0}^{3} w_{ij} B_i^3(u) B_j^3(v) \mathbf{P}_{ij}, \quad 0 \le u, v \le 1,$$

где

$$w(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} w_{ij} B_i^3(u) B_j^3(v).$$

Свойства рациональных бикубических поверхностей Безье

Элементарная рациональная бикубическая поверхность Безье, порождённая массивом P:

1⁺ является гладкой поверхностью;

2⁺ граничные кривые элементарной бикубической поверхности Безье суть элементарные рациональные кубические кривые Безье (управляемые, правда, разными наборами весовых коэффициентов); их опорные ломаные - границы опорной многогранной поверхности (опорного графа), в частности граничная кривая, описываемая радиус-вектором $\mathbf{R}(u, v)$, является элементарной рациональной кривой Безье с опорным массивом вершин $\mathbf{P}_{00}, \mathbf{P}_{01}, \mathbf{P}_{02}, \mathbf{P}_{03}$; все 4 угловые вершины опорного многогранника $\mathbf{P}_{00}, \mathbf{P}_{03}, \mathbf{P}_{30}$ и $\mathbf{P}_{33}$ лежат на поверхности Безье,

$$\mathbf{R}(0, 0) = \mathbf{P}_{00}, \quad \mathbf{R}(1, 0) = \mathbf{P}_{30}, \quad \mathbf{R}(0, 1) = \mathbf{P}_{03}, \quad \mathbf{R}(1, 1) = \mathbf{P}_{33},$$

и поверхность касается угловых граней опорного многогранника, например для угловой вершины $\mathbf{P}_{00}(u = v = 0)$, $\mathbf{R}(0, 0) = \mathbf{P}_{00}$, выполняются равенства, содержащие 3 весовых отношения

$$\frac{w_{01}}{w_{00}}, \quad \frac{w_{10}}{w_{00}}, \quad \frac{w_{11}}{w_{00}};$$

$$\mathbf{R}_u(0, 0) = 3\frac{w_{10}}{w_{00}}(\mathbf{P}_{10} - \mathbf{P}_{00}), \quad \mathbf{R}_v(0, 0) = 3\frac{w_{01}}{w_{00}}(\mathbf{P}_{01} - \mathbf{P}_{00})$$

(касательные векторы элементарной бикубической поверхности Безье в угловой вершине коллинеарны звеньям соответствующих граничных опорных ломаных, исходящих из этой вершины),

$$\mathbf{R}_{uv}(0, 0) = 9\left(\frac{w_{11}}{w_{00}}(\mathbf{P}_{11} - \mathbf{P}_{00}) + \frac{w_{01}w_{10}}{w_{00}^2}(2\mathbf{P}_{00} - \mathbf{P}_{01} - \mathbf{P}_{10})\right)$$

(вектор скручивания $\mathbf{R}_{uv}(0, 0)$ в угловой вершине $\mathbf{P}_{00}$ оценивает степень отклонения опорной вершины $\mathbf{P}_{11}$ от касательной плоскости элементарной бикубической поверхности Безье в вершине $\mathbf{P}_{00}$ (для каж-

дой из трех других угловых вершин $\mathbf{P}_{01}(u = 0, v = 1)$, $\mathbf{P}_{10}(u = 1, v = 0)$, $\mathbf{P}_{11}(u = v = 1)$ выполняются аналогичные соотношения);

3$^+$ элементарная рациональная бикубическая поверхность Безье лежит в выпуклой оболочке, порожденной массивом **P**;

4$^+$ элементарная рациональная бикубическая поверхность Безье аффинно-инвариантна;

5$^+$ элементарная рациональная бикубическая поверхность Безье "повторяет" опорную многогранную поверхность;

6$^+$ если все вершины массива **P** лежат в одной плоскости, то определяемая этим массивом элементарная рациональная бикубическая поверхность Безье представляет собой плоский криволинейный четырехугольник, лежащий в этой плоскости;

7$^+$ поведение рациональной бикубической поверхности Безье определяется не только массивом вершин, но и набором свободных параметров - весовых множителей w_{ij}: при заданном наборе вершин формой элементарной рациональной бикубической поверхности Безье можно управлять, меняя весовые коэффициенты;

8$^-$ изменение хотя бы одной вершины в массиве приводит к заметному изменению всей элементарной рациональной поверхности Безье;

9$^-$ априорные сведения о расположении элементарной рациональной бикубической поверхности Безье (принадлежность выпуклой оболочке заданного массива вершин) являются достаточно грубыми,

10$^+$ элементарная рациональная поверхность Безье проективно-инвариантна.

Замечание

При построении составной рациональной бикубической поверхности Безье следует иметь в виду, что

1) определяющий ее массив

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n\}$$

не может быть произвольным: числа m и n должны удовлетворять условию $m = 3k + 1$, $n = 3l + 1$ (впрочем, этого всегда можно добиться путем добавления в исходный массив новых вершин),

2) хотя выбором весов можно влиять на форму составной поверхности, однако в общем случае она будет всего лишь непрерывной, и кривые стыка элементарных фрагментов будут на этой поверхности ребрами.

4.2.5. Программная реализация

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
bezier_surface( p, seg_x, seg_y, div_x, div_y, output ),
```

где

```
Point **p   (in)      - точки опорного графа  Pij,
                        i = 0...(m-1),j=0...(n-1),
                        m = 3*seg_x+1,n = 3*seg_y+1,
int seg_x,seg_y (in)  - число криволинейных сегментов
                        по u и по v)
int div_x     (in)    - число звеьев в u - ломаной,
int div_y     (in)    - число звеьев в v - ломаной,
Point **output (out)  - выходной массив
                        точки Qij, i = 0...seg_x*div_x
                                   j = 0...seg_y*div_y
```

Приведем полностью текст этой программы.

```
#include "common.h"

Flt B( int i, Flt t )
{
    Flt s = 1.0 - t ;
    switch( i )
    {
        case 0: return s*s*s ;
        case 1: return 3*t*s*s ;
        case 2: return 3*t*t*s ;
        case 3: return t*t*t ;
    }
}

void bezier_3_3( Point **p, int n, Flt u, Flt v, int i, int j
Point out )
{
    int k, l ;
    Flt yy ;
    Point t = { 0, 0, 0 } ;
    for( k = 0; k < 4; k ++ )
    {
        Point s = { 0, 0, 0 } ;
        for( l = 0; l < 4; l ++ )
        {
            Flt tt = B( l, v ) ;
            VecAddS( tt, pp( p, i+k, j+l, n ), s, s ) ;
        }
        yy = B( k, u ) ;
        VecAddS( yy, s, t, t ) ;
    }
    VecCopy( t, out ) ;
}

void bezier_surface( Point **p, int seg_x, int seg_y, int
div_x, int div_y,
                     Point **output )
{
```

```
        int i, j, d1, d2 ;
        Flt u, v, du = 1.0 / div_x, dv = 1.0 / div_y ;

        int np = 3*seg_y + 1 ;
        int nc = seg_y*div_y + 1 ;

        for( j = 0; j < seg_y; j ++ )
            for( i = 0; i < seg_x; i ++ )
            {
                v = 0 ;
                for( d2 = 0; d2 <= div_y; d2 ++ )
                {
                    u = 0 ;
                    for( d1 = 0; d1 <= div_x; d1 ++ )
                    {
                        bezier_3_3( p, np, u, v, i*3, j*3,
                                    pp( output, i*div_x + d1,
j*div_y + d2, nc ) ) ;
                        u += du ;
                    }
                    v += dv ;
                }
            }
}
```

Приведем пример использования функции для построения сплайн-поверхности.

```
#include "lib.h"
/* Описание вызываемой функции */
void bezier_surface( Point **p, int seg_x, int seg_y, int
div_x, int div_y, Point **output ) ;

/* число сегментов  */
#define SEG    (2)

/* число точек в опорном графе    */
#define  M      ( SEG*3+1 )
#define DIV    (5)

/* число точек в аппроксимирующем графе */
#define  N     (SEG*DIV+1)

/* Control graph initialization */
Point p[ M ][ M ] ;
Point p30 = {  .5, -.5,  0 } ;
Point p22 = { .33, 1, -.33 } ;
Point p06 = { 0.0, 1, -1.0 } ;
Point p60 = {   1, -1,  0  } ;
Point p54 = { .83, 2, -.67 } ;
Point p36 = {  .5, 5, -1   } ;

/* вспомогательная переменная  */
Point tt = { 0, 0, 0 } ;

/* Описание выходных массивов */
Point cc[ N ][ N ] ;

Point EyePos = { .5, 6, 2 } ;
Point LookAt = { .5, 0, -3 } ;
```

```
Vec Up = { 0, 1, -1 } ;
Flt Alpha = .7 ;

void main( void )
{
    int i, j ;

    /* Заполнение массива точек */
    for( i = 0; i < M; i ++ )
        for( j = 0; j < M; j ++ )
        {
            tt[ 0 ] = (Flt )i / ((Flt )( M - 1 )) ;
            tt[ 2 ] = - (Flt )j / ((Flt )( M - 1 )) ;
            VecCopy( tt, p[ i ][ j ] ) ;
        }
    VecCopy( p30, p[ 3 ][ 0 ] ) ;
    VecCopy( p22, p[ 2 ][ 2 ] ) ;
    VecCopy( p06, p[ 0 ][ 6 ] ) ;
    VecCopy( p60, p[ 6 ][ 0 ] ) ;
    VecCopy( p54, p[ 5 ][ 4 ] ) ;
    VecCopy( p36, p[ 3 ][ 6 ] ) ;

    /* вычисление сплайна */
   bezier_surface( (Point **)p,SEG,SEG, DIV,DIV,(Point **)cc )
;

    /* инициализация графической моды*/
    SetVideoMode() ;

    /* инициализация трехмерной графики */
    Init3D( EyePos, LookAt, Up, Alpha ) ;

    /* отображение на дисплее построенного сплайна */
    PlotSurface( (Point **)cc, M, M, 1 ) ;
}
```

Текст программы bezier_surface находится в файле bezier.c в поддиректории SURFACES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

4.3. *B*-сплайновые поверхности

4.3.1. Параметрические уравнения элементарной бикубической *B*-сплайновой поверхности

По заданному массиву

$$\begin{matrix} \mathbf{P}_{00}, & \mathbf{P}_{01}, & \mathbf{P}_{02}, & \mathbf{P}_{03}, \\ \mathbf{P}_{10}, & \mathbf{P}_{11}, & \mathbf{P}_{12}, & \mathbf{P}_{13}, \\ \mathbf{P}_{20}, & \mathbf{P}_{21}, & \mathbf{P}_{22}, & \mathbf{P}_{23}, \\ \mathbf{P}_{30}, & \mathbf{P}_{31}, & \mathbf{P}_{32}, & \mathbf{P}_{33} \end{matrix}$$

(элементарная) бикубическая B-сплайновая поверхность определяется при помощи векторного уравнения

$$\mathbf{R}(u, v) = \sum_{i=0}^{3}\sum_{j=0}^{3} n_i(u) n_j(v) \mathbf{P}_{ij}, \quad 0 \le u, v \le 1,$$

функциональные коэффициенты $n_i(u)$ и $n_j(v)$ в котором задаются следующими формулами:

$$n_0(u) = \frac{1}{6}(1-u)^3, \qquad n_1(u) = \frac{1}{6}(3u^3 - 6u^2 + 4),$$

$$n_2(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1), \qquad n_3(u) = \frac{u^3}{6}$$

(формулы для многочленов $n_j(v)$ отличаются от приведенных только тем, что всюду вместо переменной *u* стоит переменная *v*).

Замечание

Запись параметрического уравнения бикубической B-сплайновой поверхности в виде

$$\mathbf{R}(u, v) = \sum_{i=0}^{3} n_i(u) \left(\sum_{j=0}^{3} n_j(v) \mathbf{P}_{ij} \right)$$

хорошо подчеркивает тесную связь, существующую между кубическими B-сплайновыми кривыми и бикубическими B-сплайновыми поверхностями.

Матричная запись параметрических уравнений, описывающих элементарную бикубическую *B*-сплайновую поверхность, -

$$\mathbf{R}(u, v) = (n_0(u)n_1(u)n_2(u)n_3(u)) \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{pmatrix} \begin{pmatrix} n_0(u) \\ n_1(u) \\ n_2(u) \\ n_3(u) \end{pmatrix},$$

$$0 \le u, v \le 1.$$

Последнюю формулу часто записывают так

$$\mathbf{R}(t) = \mathbf{U}^T \mathbf{M}^T \mathbf{P} \mathbf{M} \mathbf{V}, \quad 0 \le u, v \le 1,$$

где

$$\mathbf{R}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \qquad \mathbf{U} = \begin{pmatrix} u^0 \\ u^1 \\ u^2 \\ u^3 \end{pmatrix}, \qquad \mathbf{V} = \begin{pmatrix} v^0 \\ v^1 \\ v^2 \\ v^3 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{pmatrix}, \qquad \mathbf{M} = \frac{1}{6} \begin{pmatrix} 1 & -3 & -3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Матрица **M** называется *базисной матрицей бикубической B-сплайновой поверхности.*

4.3.2. Свойства элементарных бикубических *B*-сплайновых поверхностей

Свойства элементарных бикубических *B*-сплайновых поверхностей являются прямыми следствиями свойств элементарных кубических B-сплайновых кривых. Укажем некоторые из них.

Основные свойства элементарных бикубических *B*-сплайновых поверхностей

Элементарная бикубическая *B*-сплайновая поверхность, порожденная массивом **P**:

1⁺ является гладкой поверхностью;

2⁺ как правило, не проходит ни через одну из 16 опорных вершин массива;

3⁺ граничные кривые элементарной бикубической *B*-сплайновой поверхности суть элементарные кубические *B*-сплайновые кривые; их опорные ломаные - границы опорной многогранной поверхности

сом-вектором $\mathbf{R}(0, v)$, является элементарной кубической *B*-сплайновой кривой с опорным массивом вершин $\mathbf{P}_{00}, \mathbf{P}_{01}, \mathbf{P}_{02}, \mathbf{P}_{03}$;

4⁺ элементарная бикубическая *B*-сплайновая поверхность лежит в выпуклой оболочке, порожденной заданным массивом **P** из 16 вершин; это свойство является следствием того, что неотрицательные функциональные коэффициенты $n_i(u)$ и $n_j(v)$ $(i, j = 0, 1, 2, 3)$ разбивают единицу,

$$\sum_{i=0}^{3}\sum_{j=0}^{3} n_i(u)n_j(v) = 1;$$

5⁺ элементарная бикубическая *B*-сплайновая поверхность аффинно-инвариантна;

6⁺ элементарная бикубическая *B*-сплайновая поверхность "повторяет" опорную многогранную поверхность;

7⁺ если все вершины заданного массива **P** лежат в одной плоскости, то определяемая этим массивом элементарная бикубическая *B*-сплайновая поверхность представляет собой плоский криволинейный четырехугольник, лежащий в этой же плоскости;

8⁻ изменение хотя бы одной вершины в массиве приводит к заметному изменению всей элементарной бикубической *B*-сплайновой поверхности;

9⁻ априорные сведения о расположении элементарной бикубической *B*-сплайновой поверхности (принадлежность выпуклой оболочке заданного массива вершин) являются достаточно грубыми;

10⁺ в уравнениях, описывающих элементарную бикубическую *B*-сплайновую поверхность, нет свободных параметров - форма элементарной бикубической *B*-сплайновой поверхности однозначно определена;

11⁻ элементарная бикубическая *B*-сплайновая поверхность проективно-неинвариантна.

4.3.3. Составные бикубические *B*-сплайновые поверхности

Составной бикубической B-сплайновой поверхностью, определяемой массивом из $(m + 1)(n + 1)$ вершин

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n\} \quad m \geq 3 \quad n \geq 3,$$

называется поверхность S, которую можно представить в виде объединения $(m - 2)(n - 2)$ элементарных бикубических *B*-сплайновых поверхностей $S^{(1,1)}, \ldots, S^{(m-2,n-2)}$,

$$S = \cup_{i=1}^{m-2} \cup_{j=1}^{n-2} S^{(i,j)};$$

(i, j)-я поверхность $S^{(i,j)}$ описывается параметрическими уравнениями вида

$$\mathbf{R}^{(i,j)}(u, v) = \sum_{k=0}^{3}\sum_{l=0}^{3} n_k(u) n_l(v) \mathbf{P}_{k-1+i,l-1+j} ,$$

$$0 \le u, v \le 1, \quad 1 \le i \le m-2, \quad 1 \le j \le n-2 ,$$

или в матричной форме:

$$\mathbf{R}^{(i,j)}(u, v) = \mathbf{U}^T \mathbf{M}^T \begin{pmatrix} \mathbf{P}_{i-1,j-1} & \mathbf{P}_{i,j-1} & \mathbf{P}_{i+1,j-1} & \mathbf{P}_{i+2,j-1} \\ \mathbf{P}_{i-1,j} & \mathbf{P}_{i,j} & \mathbf{P}_{i+1,j} & \mathbf{P}_{i+2,j} \\ \mathbf{P}_{i-1,j+1} & \mathbf{P}_{i,j+1} & \mathbf{P}_{i+1,j+1} & \mathbf{P}_{i+2,j+1} \\ \mathbf{P}_{i-1,j+2} & \mathbf{P}_{i,j+2} & \mathbf{P}_{i+1,j+2} & \mathbf{P}_{i+2,j+2} \end{pmatrix} \mathbf{MV},$$

$$0 \le u, v \le 1, \quad 1 \le i \le m-2, \quad 1 \le j \le n-2 ,$$

где **M** - базисная матрица бикубического *B*-сплайна. Взаимное расположение элементарных фрагментов составной поверхности схематически представлено на рис. 4.22.

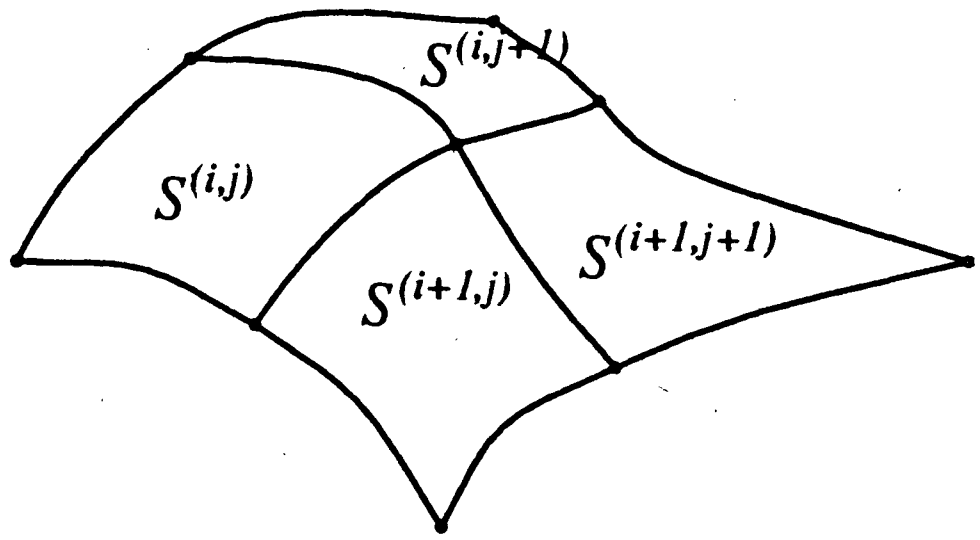


Рис. 22

Свойства бикубических B-сплайновых поверхностей

Составная бикубическая B-сплайновая поверхность, порожденная массивом **P**:

1$^+$ является C^2-непрерывной;

2$^+$ как правило, не проходит ни через одну из опорных вершин массива;

3$^+$ граничные кривые составной бикубической *B*-сплайновой поверхности суть составные кубические *B*-сплайновые кривые; их опорные ломаные - границы опорной многогранной поверхности (опорного графа), в частности граничная кривая, описываемая радиусом-вектором $\mathbf{R}(0, v)$, является составной кубической *B*-сплайновой кривой с опорным массивом вершин $\mathbf{P}_{00}, \ldots, \mathbf{P}_{0n}$;

4$^+$ составная бикубическая *B*-сплайновая поверхность лежит в объединении $(m - 2)(n - 2)$ выпуклых оболочек, порожденных наборами

$$\begin{array}{cccc}
\mathbf{P}_{i-1,j-1} & \mathbf{P}_{i,j-1} & \mathbf{P}_{i+1,j-1} & \mathbf{P}_{i+2,j-1} \\
\mathbf{P}_{i-1,j} & \mathbf{P}_{i,j} & \mathbf{P}_{i+1,j} & \mathbf{P}_{i+2,j} \\
\mathbf{P}_{i-1,j+1} & \mathbf{P}_{i,j+1} & \mathbf{P}_{i+1,j+1} & \mathbf{P}_{i+2,j+1} \\
\mathbf{P}_{i-1,j+2} & \mathbf{P}_{i,j+2} & \mathbf{P}_{i+1,j+2} & \mathbf{P}_{i+2,j+2}
\end{array}$$

(схематическое разбиение составной бикубической *B*-сплайновой поверхности показано на рис. 4.23);

(1, 1)	...	*(1, j)*	*(1, j+1)*	...	*(1, n-2)*
...	...	...	...	...	...
(i, 1)	...	*(i, j)*	*(i, j+1)*	...	*(i, n-2)*
(i+1, 1)	...	*(i+1, j)*	*(i+1, j+1)*	...	*(i+1, n-2)*
...	...	...	...	...	...
(i+1, 1)	...	*(m-2, j)*	*(m-2, j+1)*	...	*(m-2, n-2)*

Рис. 4.23

5⁺ составная бикубическая *B*-сплайновая поверхность аффинно-инвариантна;

6⁺ составная бикубическая *B*-сплайновая поверхность "повторяет" опорную многогранную поверхность;

7⁺ если все вершины массива **P** лежат в одной плоскости, то определяемая этим массивом составная бикубическая *B*-сплайновая поверхность представляет собой плоский криволинейный четырехугольник, лежащий в этой же плоскости;

8⁺ изменение одной вершины в массиве приводит к изменению только части поверхности: при изменении вершины $\mathbf{P}_{ij}$ нужно пересчитать параметрические уравнения только 16 элементарных поверхностей:

$$\begin{array}{cccc}
S^{(i-2,j-2)} & S^{(i-2,j-1)} & S^{(i-2,j)} & S^{(i-2,j+1)} \\
S^{(i-1,j-2)} & S^{(i-1,j-1)} & S^{(i-1,j)} & S^{(i-1,j+1)} \\
S^{(i,j-2)} & S^{(i,j-1)} & S^{(i,j)} & S^{(i,j+1)} \\
S^{(i+1,j-2)} & S^{(i+1,j-1)} & S^{(i+1,j)} & S^{(i+1,j+1)}
\end{array}$$

9⁻ составная бикубическая *B*-сплайновая поверхность проективно-неинвариантна;

10⁺ поведение составной бикубической *B*-сплайновой поверхности определяется только массивом вершин.

Замечание

На взаимное расположение вершин в массиве не накладывается никаких ограничений: некоторые из них могут и совпадать. Однако следует иметь в виду, что в подобных случаях поверхность может потерять свою регулярность. Впрочем, если номера совпадающих вершин сильно разнятся, то никакой потери регулярности не происходит.

Простейшие из случаев совпадения вершин рассматриваются ниже.

4.3.4. Кратные и воображаемые вершины

В приложениях часто удобно использовать массивы, в которых совпадающие (кратные) вершины являются граничными и угловыми. В этом случае в создании граничных бикубических *B*-сплайновых фрагментов участвуют опорные вершины в числе меньше 16 (без учета кратности), что сказывается на размерах фрагментов, а также позволяет получать априорную информацию о поведении составной поверхности вблизи границы. Составная бикубическая *B*-сплайновая поверхность, как правило, не содержит ни одной из вершин массива, который ее порождает. Однако подбором вспомогательных вершин и построением дополнительных элементарных поверхностей можно добиться того, чтобы граничные кривые новой составной поверхности располагались ближе к граничным и угловым вершинам. Обычно это проводится путем использования *кратных* или *воображаемых* вершин.

А. Двойные вершины

Зададим $2m + 2n + 8$ новых вершин, положив

$$\mathbf{P}_{-1,j} = \mathbf{P}_{0j}, \quad j = 0, 1, \ldots, n, \quad \mathbf{P}_{m+1,j} = \mathbf{P}_{mj}, \quad j = 0, 1, \ldots, n,$$

$$\mathbf{P}_{i,-1} = \mathbf{P}_{i0}, \quad i = 0, 1, \ldots, m, \quad \mathbf{P}_{i,n+1} = \mathbf{P}_{in}, \quad i = 0, 1, \ldots, m,$$

$$\mathbf{P}_{-1,-1} = \mathbf{P}_{0,0}, \mathbf{P}_{-1,n+1} = \mathbf{P}_{0n}, \mathbf{P}_{m+1,-1} = \mathbf{P}_{m0}, \mathbf{P}_{m+1,n+1} = \mathbf{P}_{mn}.$$

Эти вершины как бы "окаймляют" заданный массив $\mathbf{P}$ и вместе с ним образуют новый

$$\mathbf{P}^* = \{\mathbf{P}_{ij}, \quad i = -1, \ldots, m+1, \quad j = -1, \ldots, n+1\}$$

из $(m + 3)(n + 3)$ вершин.

Замечание

Как множества точек массивы $\mathbf{P}$ *и* $\mathbf{P}^*$ *неразличимы, так как все добавленные вершины отличаются от заданных только номерами - геометрически новых вершин не появилось.*

Составная бикубическая *B*-сплайновая поверхность $\mathbf{S}^*$, определяемая массивом $\mathbf{P}^*$, состоит из mn фрагментов S_{ij}, радиусы-векторы которых вычисляются по формулам

$$\mathbf{R}^{(i,j)}(u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} n_k(u) n_l(v) \mathbf{P}_{k-1+i,l-1+j},$$

$$0 \le u, v \le 1, \quad i = -1, \ldots, m-1, \; j = -1, \ldots, n-1.$$

Только $2m + 2n - 4$ из этих элементарных поверхностей

$$S^{(0,j)}, \quad j = 0, \ldots, n-2, \quad S^{(i,n-1)}, \quad i = 0, \ldots, m-2,$$

$$S^{(m-1,j)}, \quad j = 1, \ldots, n-1, \quad S^{(i,0)}, \quad i = 1, \ldots, m-1,$$

являются новыми (ясно, что в построении каждого из новых фрагментов принимает участие хотя бы одна новая вершина).

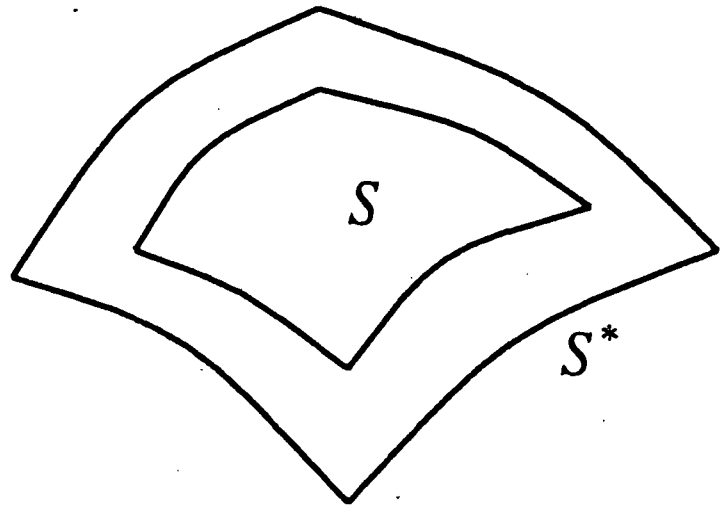


Рис. 4.24

Взаимное расположение составных бикубических *B*-сплайновых поверхностей - старой *S* и новой $\mathbf{S}^*$ - хорошо видно на рис. 4.24. Новые фрагменты схематически представлены на рис. 4.25.

Значения вектора скручивания в угловых вершинах четырех угловых фрагментов $S^{(0,0)}, S^{(0,n-1)}, S^{(m-1,n-1)}, S^{(m-1,0)}$ поверхности $\mathbf{S}^*$

$$\mathbf{R}_{uv}^{(0,0)}(0,0) = \frac{1}{4}(\mathbf{P}_{00} - \mathbf{P}_{01} + \mathbf{P}_{11} - \mathbf{P}_{10}),$$

$$\mathbf{R}_{uv}^{(0,n-1)}(0,1) = \frac{1}{4}(\mathbf{P}_{0,n-1} - \mathbf{P}_{0n} + \mathbf{P}_{1n} - \mathbf{P}_{1,n-1}),$$

$$\mathbf{R}_{uv}^{(m-1,n-1)}(1,1) = \frac{1}{4}(\mathbf{P}_{m-1,n-1} - \mathbf{P}_{m-1,n} + \mathbf{P}_{mn} - \mathbf{P}_{m,n-1}),$$

$$\mathbf{R}_{uv}^{(m-1,0)}(1,0) = \frac{1}{4}(\mathbf{P}_{m-1,0} - \mathbf{P}_{m-1,1} + \mathbf{P}_{m1} - \mathbf{P}_{m0})$$

с точностью до множителя равны значениям векторов скручивания билинейных поверхностей, определяемых соответственно четверками вершин

$$\mathbf{P}_{00},\quad \mathbf{P}_{01},\quad \mathbf{P}_{11},\quad \mathbf{P}_{10};$$
$$\mathbf{P}_{0,n-1},\quad \mathbf{P}_{0n},\quad \mathbf{P}_{0n},\quad \mathbf{P}_{1,n-1};$$
$$\mathbf{P}_{m-1,n-1},\quad \mathbf{P}_{m-1,n},\quad \mathbf{P}_{mn},\quad \mathbf{P}_{m,n-1};$$
$$\mathbf{P}_{m-1,0},\quad \mathbf{P}_{m-1,1},\quad \mathbf{P}_{11},\quad \mathbf{P}_{m0}.$$

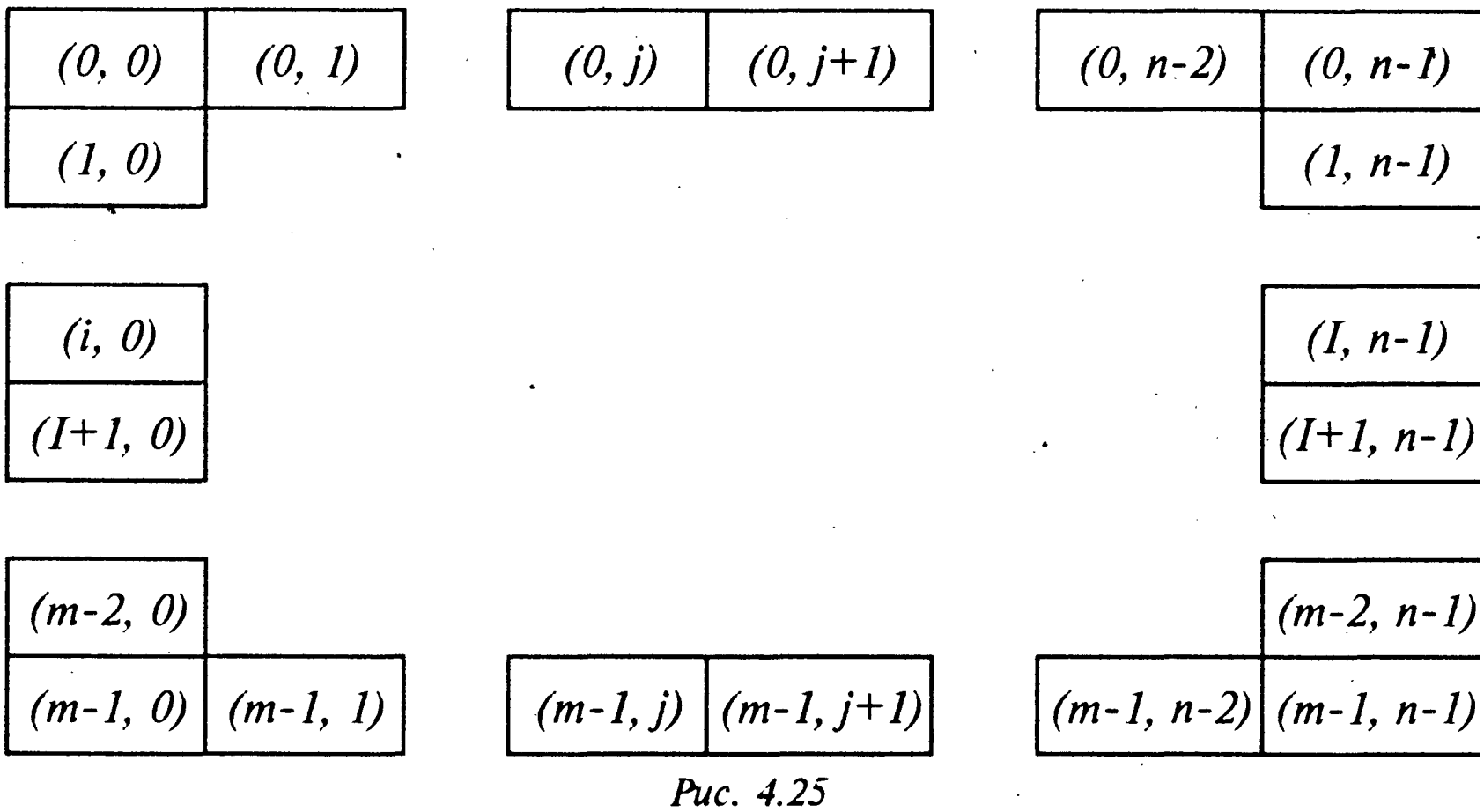


Рис. 4.25

Б. Тройные вершины

В дополнение к уже взятым $2m + 2n + 8$ вершинам зададим еще $2m + 2n + 16$ новых вершин, положив

$$\mathbf{P}_{-2,j} = \mathbf{P}_{-1,j},\ j = -1, \ldots, n+1, \quad \mathbf{P}_{m+2,j} = \mathbf{P}_{m+1,j},\ j = -1, \ldots, n+1,$$
$$\mathbf{P}_{i,-2} = \mathbf{P}_{i,-1},\ i = -1, \ldots, m+1, \quad \mathbf{P}_{i,n+2} = \mathbf{P}_{i,n+1}, \quad i = -1, \ldots, m+1,$$
$$\mathbf{P}_{-2,-2} = \mathbf{P}_{-1,-1}, \quad \mathbf{P}_{-2,n+2} = \mathbf{P}_{-1,n+1},$$
$$\mathbf{P}_{m+2,-2} = \mathbf{P}_{m+1,-1}, \quad \mathbf{P}_{m+2,n+2} = \mathbf{P}_{m+1,n+1},$$

и с учетом добавленных вершин построим $2m + 2n + 4$ новых элементарных поверхности

$$S^{(-1,j)}, \quad j = -1, \ldots, n-1,$$
$$S^{(i,n)}, \quad i = -1, \ldots, m-1,$$

$$S^{(m,j)}, \quad j = 0,\ldots,n,$$

$$S^{(i,-1)}, \quad i = 0,\ldots,m,$$

задав их параметрическими уравнениями вида

$$\mathbf{R}^{(i,j)}(u, v) = \sum_{k=0}^{3}\sum_{l=0}^{3} n_k(u) n_l(v) \mathbf{P}_{k-1+i,\, l-1+j},$$

$$0 \le u, v \le 1.$$

В результате получим новую бикубическую *B*-сплайновую поверхность $\mathbf{S}^{**}$, состоящую из $(m + 2)(n + 2)$ фрагментов (рис. 4.26).

(-1, -1)	*(-1, 0)*	*(-1, 1)*
(0, -1)	*(0, 0)*	*(0, 1)*
(1, -1)	*(1, 0)*	

Рис. 4.26

Некоторое представление о взаимном расположении новых фрагментов поверхности $\mathbf{S}^{**}$ относительно старой поверхности $\mathbf{S}^{*}$ могут дать схематические изображения, приведенные на рис. 4.27.

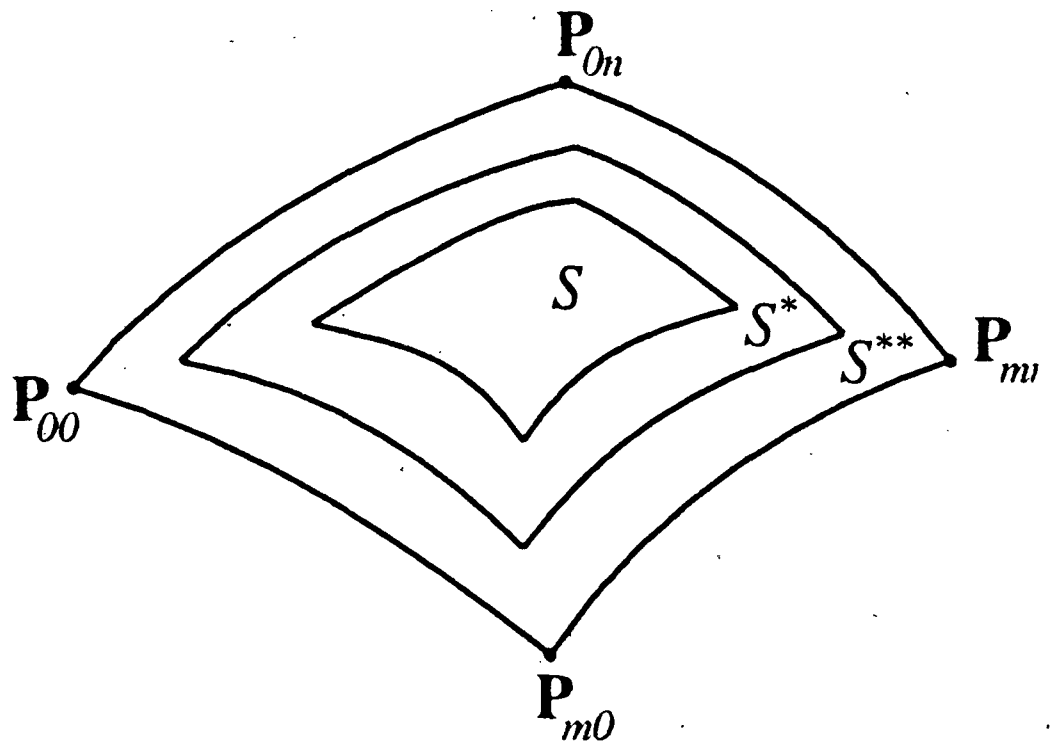


Рис. 4.27

Все 4 угловых фрагмента $S^{(-1,-1)}, S^{(-1,n)}, S^{(m,n)}, S^{(m,-1)}$, поверхности $\mathbf{S}^{**}$ лежат на билинейных поверхностях. Поэтому векторы скручивания в точках этих четырех фрагментов постоянны по направлению и соответственно имеют вид:

$$\mathbf{R}_{uv}^{(-1,-1)}(u, v) = 3u^2v^2(\mathbf{P}_{00} - \mathbf{P}_{01} + \mathbf{P}_{11} - \mathbf{P}_{10}),$$

$$\mathbf{R}_{uv}^{(-1,n)}(u, v) = 3u^2v^2(\mathbf{P}_{0,n-1} - \mathbf{P}_{0n} + \mathbf{P}_{1n} - \mathbf{P}_{1,n-1}),$$

$$\mathbf{R}_{uv}^{(m,n)}(u, v) = 3u^2v^2(\mathbf{P}_{m-1,n-1} - \mathbf{P}_{m-1,n} + \mathbf{P}_{mn} - \mathbf{P}_{m,n-1}),$$

$$\mathbf{R}_{uv}^{(m,-1)}(u, v) = 3u^2v^2(\mathbf{P}_{m-1,0} - \mathbf{P}_{m-1,1} + \mathbf{P}_{11} - \mathbf{P}_{m0}).$$

Кроме того, все 4 угловые вершины лежат в углах составной бикубической *B*-сплайновой поверхности $\mathbf{S}^{**}$, построенной при помощи тройных вершин

$$\mathbf{R}^{(-1,-1)}(0, 0) = \mathbf{P}_0, \quad \mathbf{R}^{(-1,n)}(0, 0) = \mathbf{P}_{0n},$$

$$\mathbf{R}^{(m,n)}(0,0) = \mathbf{P}_{mn}, \quad \mathbf{R}^{(m,-1)}(0,0) = \mathbf{P}_{m0}.$$

В. Воображаемые вершины

Выбором дополнительных вершин к массиву

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, \ldots, m, \; j = 0, \ldots, n\}$$

можно добиться выполнения различных условий на границе составной поверхности. Обычно новые (воображаемые) вершины ищутся в виде линейных комбинаций заданных вершин.

Рассмотрим составную бикубическую *B*-сплайновую поверхность S^*, порожденную массивом

$$\mathbf{P}^* = \{\mathbf{P}_{ij}, \quad i = -1, \ldots, m-1, \; j = -1, \ldots, n-1\},$$

$2m + 2n + 8$ вершин которого подобраны посредством следующих формул

$$\mathbf{P}_{-1,-1} = \mathbf{P}_{0,-1} - \mathbf{P}_{1,-1} + \mathbf{P}_{0,-1},$$

$$\mathbf{P}_{m+1,n+1} = \mathbf{P}_{m+1,n} - \mathbf{P}_{m+1,n-1} + \mathbf{P}_{m-1,n},$$

$$\mathbf{P}_{-1,j} = \mathbf{P}_{0,j} - \mathbf{P}_{1,j} + \mathbf{P}_{0,j}, \quad j = 0, \ldots, n,$$

$$\mathbf{P}_{i,-1} = \mathbf{P}_{i,-1} - \mathbf{P}_{i,-1} + \mathbf{P}_{i,-1}, \quad i = 0, \ldots, m,$$

$$\mathbf{P}_{-1,n+1} = \mathbf{P}_{-1,n} - \mathbf{P}_{-1,n-1} + \mathbf{P}_{-1,n},$$

$$\mathbf{P}_{m+1,-1} = \mathbf{P}_{m,-1} - \mathbf{P}_{m-1,-1} + \mathbf{P}_{m,n-1},$$

$$\mathbf{P}_{i,n+1} = \mathbf{P}_{i,n} - \mathbf{P}_{i,n-1} + \mathbf{P}_{i,n}, \quad i = 0, \ldots, m,$$

$$\mathbf{P}_{m+1,j} = \mathbf{P}_{m,j} - \mathbf{P}_{m-1,j} + \mathbf{P}_{m,j}, \quad j = 0, \ldots, n.$$

Составная бикубическая *B*-сплайновая поверхность S^*, определяемая массивом $\mathbf{P}^*$, обладает следующим свойством: кривизна общей кривой соседних фрагментов в точке, лежащей на граничной кривой этой поверхности, равна нулю.

В рассматриваемом случае векторы скручивания в угловых точках поверхности S^* равны векторам скручивания билинейных поверхностей, построенных на примыкающих к этой вершине ребрах. Например, вектор скручивания в угловой точке фрагмента $S^{(0,0)}$ равен

$$\mathbf{P}_{00} - \mathbf{P}_{01} - \mathbf{P}_{11} + \mathbf{P}_{10}.$$

4.3.5. Рациональные бикубические *B*-сплайновые поверхности

По заданному массиву

$$\mathbf{P}_{ij}, \qquad i = 0, \ldots, m, \qquad j = 0, \ldots, n,$$

(элементарная) рациональная бикубическая B-сплайновая поверхность определяется уравнением вида

$$\mathbf{R}(u, v) = \frac{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij} n_i(u) n_j(v) \mathbf{P}_{ij}}{\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij} n_i(u) n_j(v)}, \quad 0 \le u, v \le 1.$$

Неотрицательные числа w_{ij}, сумма которых положительна, называются *весами*. В случае, если все веса w_{ij} равны между собой, получается стандартная элементарная бикубическая *B*-сплайновая поверхность.

Свойства составных рациональных бикубических *B*-сплайновых поверхностей

Составная рациональная бикубическая *B*-сплайновая поверхность, порожденная массивом **P**:

1⁺ является C^2-гладкой поверхностью;

2⁺ как правило, не проходит ни через одну из опорных вершин массива;

3⁺ граничные кривые составной рациональной бикубической *B*-сплайновой поверхности суть составные рациональные кубические *B*-сплайновые кривые; их опорные ломаные - границы опорной многогранной поверхности (опорного графа), в частности, граничная кривая, описываемая радиусом-вектором **R**(0, *v*), является составной рациональной кубической *B*-сплайновой кривой с опорным массивом вершин $\mathbf{P}_{00}, \ldots, \mathbf{P}_{0n}$;

4⁺ составная рациональная бикубическая B-сплайновая поверхность лежит в объединении $(m - 2)(n - 2)$ выпуклых оболочек, порожденных наборами

$$\{\mathbf{P}_{kl}, \quad k = i - 1, i, i + 1, i + 2, \quad l = j - 1, j, j + 1, j + 2\}$$

(схематическое разбиение составной *B*-сплайновой поверхности показано на рис. 4.23);

5⁺ составная рациональная бикубическая *B*-сплайновая поверхность аффинно-инвариантна;

6⁺ составная рациональная бикубическая *B*-сплайновая поверхность "повторяет" опорную многогранную поверхность;

7⁺ если все вершины массива **P** лежат в одной плоскости, то определяемая этим массивом составная рациональная бикубическая *B*-сплайновая поверхность представляет собой плоский криволинейный четырехугольник, лежащий в этой же плоскости;

8+ изменение одной вершины в массиве приводит к изменению только части поверхности: при изменении вершины $\mathbf{P}_{ij}$ нужно пересчитать параметрические уравнения только 16 элементарных поверхностей:

$$\begin{array}{cccc} S^{(i-2,j-2)} & S^{(i-2,j-1)} & S^{(i-2,j)} & S^{(i-2,j+1)} \\ S^{(i-1,j-2)} & S^{(i-1,j-1)} & S^{(i-1,j)} & S^{(i-1,j+1)} \\ S^{(i,j-2)} & S^{(i,j-1)} & S^{(i,j)} & S^{(i,j+1)} \\ S^{(i+1,j-2)} & S^{(i+1,j-1)} & S^{(i+1,j)} & S^{(i+1,j+1)} \end{array}$$

9- составная рациональная бикубическая *B*-сплайновая поверхность проективно-инвариантна;

10+ поведение составной рациональной бикубической *B*-сплайновой поверхности определяется не только массивом вершин, но и набором свободных параметров - весов w_{ij}, *параметров формы*; при заданном наборе вершин рациональной бикубической *B*-сплайновой поверхностью можно управлять, меняя весовые множители.

4.3.6. Программная реализация

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
nurbs_surface(p, weight, uknot, vknot, m, n, div_x, div_y,output),
```

где

```
Point **p    (in)  - опорный граф сплайна, его точки  Pij,
                     i = 0...(m-1), j=0...(n-1)
Flt **w      (in)  - массив весовых коэффициентов точек, его
                     элементы  Wij, i = 0...(m-1), j=0...(n-1)
Flt uknot[]  (in)  - массив u-узлов, его элементы
                     uknot[0]...uknot[m+3],
Flt vknot[]  (in)  - массив u-узлов, его элементы
                     vknot[0]...vknot[n+3]
int m, n     (in)  - число точек в опорном графе,
int div_x    (in)  - число звеньев в u-ломаной,
int div_y    (in)  - число звеньев в u-ломаной,
Point **output (out) - граф, аппроксимирующий сплайн, его
                       точки Qij, i = 0...(m-3)*div_x,
                                  j = 0...(n-3)*div_y.
```

Приведем полностью текст этой программы.

```
#include "common.h"

Flt N( int n, int i, Flt u, Flt knot[] )
{
  if( n == 0 )  return ( knot[i] <= u && u < knot[i+1] ) ? 1.0
: 0.0 ;
  return ( u - knot[i] ) * N( n-1, i, u, knot ) / (knot[i+n]
- knot[i]) + ( knot[i+n+1] - u ) * N( n-1, i+1, u, knot ) /
(knot[i+n+1] - knot[i+1]) ;
```

```
}
void nurbs_3_3( Point **p, Flt **w, Flt uknot[], Flt vknot[],
int k, int l, int n, Flt u, Flt v, Point out )
{
    Point z = { 0, 0, 0 } ;
    Flt t, s2 = 0 ;
    int i, j ;
    for( i = k-3; i< k+1; i++ )
    {
        Flt s1 = 0 ;
        Point q = { 0, 0, 0 } ;
        for( j = l-3; j < l+1; j ++ )
        {
            t = N( 3, j, v, vknot ) * ((Flt *)w)[ i*n + j ] ;
            s1 = s1 + t ;
            VecAddS( t, pp( p, i, j, n ), q, q ) ;
        }
        t = N( 3, i, u, uknot ) ;
        s2 = s2 + t * s1 ;
        VecAddS( t, q, z, z ) ;
    }
    VecDiv( z, s2, out) ;
}
void nurbs_surface( Point **p, Flt **weight, Flt uknot[], Flt
vknot[],int m, int n, int div_x, int div_y, Point **output )
{
    int i, j, d1, d2 ;
    Flt u, v, du, dv ;
    int no = (n-3)*div_y + 1 ;
    for( j = 3; j < n; j ++ )
    {
        v = vknot[ j ] ;
        dv = (vknot[ j+1 ] - v ) / div_y ;
        for( d2 = 0; d2 <= div_y; d2 ++ )
        {
            u = uknot[ i ] ;
            du = ( uknot[ i+1 ] - u ) / div_x ;
            for( d1 = 0; d1 <= div_x; d1 ++ )
            {
                nurbs_3_3( p, weight, uknot, vknot, i, j,
n, u, v,pp( output, (i-3)*div_x+d1,
(j-3)*div_y+d2, no ) ) ;
                u += du ;
            }
            v += dv ;
        }
    }
}
```

Приведем пример использования функции для построения сплайн-поверхности.

```
#include "lib.h"

/* описание вызываемой функции */
void nurbs_surface( Point **p, Flt **weight, Flt uknot[], Flt
vknot[],
                    int m, int n, int div_x, int div_y, Point
**output ) ;

/* число точек опорного графа */
#define  M     (5)
#define DIV    (5)
/* число точек в аппроксимирующем графе */
#define  N    ((M-3)*DIV+1)

/* инициализация опорного графа */
Point p[ M ][ M ] ;
Point p11 = { .25, 1,  -.25 } ;
Point p40 = { 1.0, -.5, 0.0 } ;
Point p04 = { 0.0, 2,  -1.0 } ;
Point p33 = { .75, 2,  -.75 } ;

Flt uknot[ M + 4 ] = { 0, .1, .2, .3, .4, .5, .6, .7, .8 } ;
Flt vknot[ M + 4 ] = { 0, .1, .2, .3, .4, .5, .6, .7, .8 } ;

Flt weight[ M ][ M ] ;

/* промежуточная переменная */
Point tt = { 0, 0, 0 } ;

Flt beta1 = 1.0, beta2 = 0.0 ;

/* описание выходного массива  */
Point oo[ N ][ N ] ;

Point EyePos = { .5, 1, 1 } ;
Point LookAt = { .5, 0, -1 } ;
Vec Up = { 0, 1, 0 } ;
Flt Alpha = .7 ;

void main( void )
{
    int i, j ;

    /* заполнение массива */
    for( i = 0; i < M; i ++ )
        for( j = 0; j < M; j ++ )
        {
            tt[ 0 ] = (Flt )i / ((Flt )( M - 1 )) ;
            tt[ 2 ] = - (Flt )j / ((Flt )( M - 1 )) ;
            VecCopy( tt, p[ i ][ j ] ) ;

            weight[ i ][ j ] = 1.0 ;
        }
    VecCopy( p11, p[ 1 ][ 1 ] ) ;
    VecCopy( p40, p[ 4 ][ 0 ] ) ;
    VecCopy( p04, p[ 0 ][ 4 ] ) ;
    VecCopy( p33, p[ 3 ][ 3 ] ) ;
```

```
    /* вычисление значений сплайна */
    nurbs_surface( (Point **)p, (Flt **)weight, uknot, vknot,
                   M, M, DIV, DIV, (Point **)cc ) ;

    /* инициализация графической моды */
    SetVideoMode() ;

    /* инициализация трехмерной графики  */
    Init3D( EyePos, LookAt, Up, Alpha ) ;

    /* отображение на дисплее построенного сплайна */
    PlotSurface( (Point **)cc, N, N, 1 ) ;
}
```

Текст программы nurbs_surface находится в файле nurbs.c в поддиректории SURFACES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

4.4. Бета-сплайновые поверхности

4.4.1 Параметрические уравнения элементарной Бета-сплайновой поверхности

По заданному массиву

$$\begin{matrix} \mathbf{P}_{00}, & \mathbf{P}_{01}, & \mathbf{P}_{02}, & \mathbf{P}_{03}, \\ \mathbf{P}_{10}, & \mathbf{P}_{11}, & \mathbf{P}_{12}, & \mathbf{P}_{13}, \\ \mathbf{P}_{20}, & \mathbf{P}_{21}, & \mathbf{P}_{22}, & \mathbf{P}_{23}, \\ \mathbf{P}_{30}, & \mathbf{P}_{31}, & \mathbf{P}_{32}, & \mathbf{P}_{33} \end{matrix}$$

(элементарная) Бета-сплайновая поверхность определяется при помощи векторного уравнения

$$\mathbf{R}^{(i,j)}(u, v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u) b_j(v) \mathbf{P}_{i,j}, \quad 0 \le u, v \le 1,$$

функциональные коэффициенты $b_i(u)$ и $b_j(v)$ в котором задаются следующими формулами:

$$b_0(u) = \frac{2\beta_1^3}{\delta}(1-u)^3, \qquad b_3(u) = \frac{2u^3}{\delta},$$

$$b_1(u) = \frac{1}{\delta}(2\beta_1^3 u(u^2 - 3u + 3) + 2\beta_1^2(u^3 - 3u^2 + 2) +$$

$$+ 2\beta_1(u^3 - 3u + 2) + \beta_2(2u^3 - 3u^2 + 1)),$$

$$b_2(u) = \frac{1}{\delta}(2\beta_1^2 u^2(-u + 3) + 2\beta_1 u(-u^2 + 3) + \beta_2 u^2(-2u + 3) + 2(-u^3 + 1))$$

(формулы для многочленов $b_j(v)$ отличаются от приведенных только тем, что всюду вместо переменной *u* стоит переменная *v*), где $\beta_1 > 0$ и $\beta_2 \ge 0$ и $\delta = 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + \beta_2 + 2$. Числовые параметры β_1 и β_2 называются *параметрами формы Бета-сплайновой поверхности.*

Замечания:

1. *При $\beta_1 = 1$ и $\beta_2 = 0$ получается элементарная бикубическая B-сплайновая поверхность.*
2. *Запись параметрического уравнения Бета-сплайновой поверхности в виде*

$$\mathbf{R}(u, v) = \sum_{i=0}^{3} b_i(u) \left(\sum_{j=0}^{3} b_j(v) \mathbf{P}_{ij} \right)$$

хорошо подчеркивает тесную связь, существующую между Бета-сплайновыми кривыми и Бета-сплайновыми поверхностями.

Матричная запись параметрических уравнений, описывающих элементарную Бета-сплайновую поверхность,

$$\mathbf{R}(u, v) = (b_0(u),\ b_1(u),\ b_2(u),\ b_3(u)) \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{pmatrix} \begin{pmatrix} b_0(v) \\ b_1(v) \\ b_2(v) \\ b_3(v) \end{pmatrix},$$

$$0 \le u, v \le 1 .$$

Последнюю формулу часто записывают так:

$$\mathbf{R}(t) = \mathbf{U}^T \mathbf{M}^T \mathbf{P} \mathbf{M} \mathbf{V}, \quad 0 \le u, v \le 1,$$

где

$$\mathbf{R}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \qquad \mathbf{U} = \begin{pmatrix} u^0 \\ u^1 \\ u^2 \\ u^3 \end{pmatrix}, \qquad \mathbf{V} = \begin{pmatrix} v^0 \\ v^1 \\ v^2 \\ v^3 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{pmatrix},$$

$$\mathbf{M} = \frac{1}{\delta} \begin{pmatrix} 2\beta_1^3 & -6\beta_1^3 & 6\beta_1^3 & -2\beta_1^3 \\ 4(\beta_1^2 + \beta_1) + \beta_2 & 6(\beta_1^3 - \beta_1) & -3(2\beta_1^3 + \eta & 2(\beta_1^3 + \theta) \\ 2 & 6\beta_1 & 3\eta & -2(\theta + 1) \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

(здесь $\eta = 2\beta_1^2 + \beta_2$, $\theta = \beta_1^2 + \beta_1 + \beta_2$).

Матрица **M** называется *базисной матрицей Бета-сплайновой поверхности.*

4.4.2. Свойства элементарных Бета-сплайновых поверхностей

Свойства элементарных Бета-сплайновых поверхностей являются прямыми следствиями свойств элементарных Бета-сплайновых кривых. Укажем некоторые из них.

Свойства элементарных Бета-сплайновых поверхностей

Элементарная Бета-сплайновая поверхность, порожденная массивом **P**:

1⁺ является гладкой поверхностью;

2⁺ как правило, не проходит ни через одну из шестнадцати опорных вершин массива;

3⁺ граничные кривые элементарной Бета-сплайновой поверхности суть элементарные Бета-сплайновые кривые; их опорные ломаные - границы опорной многогранной поверхности (опорного графа), в частности, граничная кривая, описываемая радиусом-вектором **R**(0, *v*), является элементарной Бета-сплайновой кривой с опорным массивом вершин $\mathbf{P}_{00}, \mathbf{P}_{01}, \mathbf{P}_{02}, \mathbf{P}_{03}$;

4⁺ элементарная Бета-сплайновая поверхность лежит в выпуклой оболочке, порожденной заданным массивом **P** из 16 вершин; это свойство является следствием того, что неотрицательные функциональные коэффициенты $b_i(u)$ и $b_j(v)$, $i, j = 0, 1, 2, 3$, разбивают единицу,

$$\sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)b_j(v) = 1;$$

5⁺ элементарная Бета-сплайновая поверхность аффинно-инвариантна;

6⁺ элементарная Бета-сплайновая поверхность "повторяет" опорную многогранную поверхность;

7⁺ если все вершины заданного массива **P** лежат в одной плоскости, то определяемая этим массивом элементарная Бета-сплайновая поверхность представляет собой плоский криволинейный четырехугольник, лежащий в этой же плоскости;

8⁻ изменение хотя бы одной вершины в массиве приводит к заметному изменению всей элементарной Бета-сплайновой поверхности;

9⁻ априорные сведения о расположении элементарной Бета-сплайновой поверхности (принадлежность выпуклой оболочке заданного массива вершин) являются достаточно грубыми;

10⁺ в уравнениях, описывающих элементарную Бета-сплайновую поверхность, есть два свободных параметра - параметры формы β_1

и β_2, меняя которые можно управлять формой элементарной Бета-сплайновой поверхности;

11- элементарная Бета-сплайновая поверхность проективно неинвариантна.

4.4.3. Составные Бета-сплайновые поверхности

Составной Бета-сплайновой поверхностью, определяемой массивом из $(m + 1)(n + 1)$ вершин

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n\} \quad m \geq 3, n \geq 3,$$

называется поверхность S, которую можно представить в виде объединения $(m - 2)(n - 2)$ элементарных Бета-сплайновых поверхностей $S^{(1,1)}, \ldots, S^{(m-2, n-2)}$,

$$S = \cup_{i=1}^{m-2} \cup_{j=1}^{n-2} S^{(i,j)};$$

(i, j)-я поверхность $S^{(i,j)}$ описывается параметрическими уравнениями вида

$$\mathbf{R}^{(i,j)}(u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} b_k(u) b_l(v) \mathbf{P}_{k-1+i,l-1+j},$$

$$0 \leq u,v \leq 1, \quad 1 \leq i \leq m-2, \quad 1 \leq j \leq n-2,$$

или в матричной форме:

$$\mathbf{R}^{(i,j)}(u, v) = \mathbf{U}^T \mathbf{M}^T \begin{pmatrix} \mathbf{P}_{i-1,j-1} & \mathbf{P}_{i,j-1} & \mathbf{P}_{i+1,j-1} & \mathbf{P}_{i+2,j-1} \\ \mathbf{P}_{i-1,j} & \mathbf{P}_{i,j} & \mathbf{P}_{i+1,j} & \mathbf{P}_{i+2,j} \\ \mathbf{P}_{i-1,j+1} & \mathbf{P}_{i,j+1} & \mathbf{P}_{i+1,j+1} & \mathbf{P}_{i+2,j+1} \\ \mathbf{P}_{i-1,j+2} & \mathbf{P}_{i,j+2} & \mathbf{P}_{i+1,j+2} & \mathbf{P}_{i+2,j+2} \end{pmatrix} \mathbf{MV},$$

$$0 \leq u, v \leq 1, \quad 1 \leq i \leq m-2, \quad 1 \leq j \leq n-2,$$

где **M** - базисная матрица Бета-сплайна.

Некоторое представление о взаимном расположении элементарных фрагментов составной поверхности может дать схематическое изображение на рис. 4.22.

Свойства Бета-сплайновых поверхностей

Составная Бета-сплайновая поверхность, порожденная массивом **P**:

1+ является G^2-непрерывной;

2+ как правило, не проходит ни через одну из опорных вершин массива;

3⁺ граничные кривые составной Бета-сплайновой поверхности суть составные Бета-сплайновые кривые; их опорные ломаные - границы опорной многогранной поверхности (опорного графа), в частности граничная кривая, описываемая радиусом-вектором $\mathbf{R}(0, v)$, является составной Бета-сплайновой кривой с опорным массивом вершин $\mathbf{P}_{00}, \ldots, \mathbf{P}_{0n}$;

4⁺ составная Бета-сплайновая поверхность лежит в объединении $(m - 2)(n - 2)$ выпуклых оболочек, порожденных наборами

$$\begin{array}{cccc} \mathbf{P}_{i-1,j-1} & \mathbf{P}_{i,j-1} & \mathbf{P}_{i+1,j-1} & \mathbf{P}_{i+2,j-1} \\ \mathbf{P}_{i-1,j} & \mathbf{P}_{i,j} & \mathbf{P}_{i+1,j} & \mathbf{P}_{i+2,j} \\ \mathbf{P}_{i-1,j+1} & \mathbf{P}_{i,j+1} & \mathbf{P}_{i+1,j+1} & \mathbf{P}_{i+2,j+1} \\ \mathbf{P}_{i-1,j+2} & \mathbf{P}_{i,j+2} & \mathbf{P}_{i+1,j+2} & \mathbf{P}_{i+2,j+2} \end{array}$$

(схематическое разбиение составной Бета-сплайновой поверхности показано на рис. 4.23);

5⁺ составная Бета-сплайновая поверхность аффинно-инвариантна;

6⁺ составная Бета-сплайновая поверхность "повторяет" опорную многогранную поверхность;

7⁺ если все вершины массива **P** лежат в одной плоскости, то определяемая этим массивом составная Бета-сплайновая поверхность представляет собой плоский криволинейный четырехугольник, лежащий в этой же плоскости;

8⁺ изменение одной вершины в массиве приводит к изменению только части поверхности: при изменении вершины $\mathbf{P}_{i,j}$ нужно пересчитать параметрические уравнения только 16 элементарных поверхностей:

$$\begin{array}{cccc} S^{(i-2,j-2)} & S^{(i-2,j-1)} & S^{(i-2,j)} & S^{(i-2,j+1)} \\ S^{(i-1,j-2)} & S^{(i-1,j-1)} & S^{(i-1,j)} & S^{(i-1,j+1)} \\ S^{(i,j-2)} & S^{(i,j-1)} & S^{(i,j)} & S^{(i,j+1)} \\ S^{(i+1,j-2)} & S^{(i+1,j-1)} & S^{(i+1,j)} & S^{(i+1,j+1)} \end{array}$$

9⁻ составная Бета-сплайновая поверхность проективно-неинвариантна;

10⁺ поведение составной Бета-сплайновой поверхности определяется не только массивом вершин, но и набором свободных параметров β_1 и β_2: при заданном наборе вершин формой составной Бета-сплайновой поверхности можно управлять, меняя параметры формы.

Замечания:

1. Параметры формы β_1 и β_2 не обязательно должны быть одинаковыми для всех элементарных фрагментов. С учетом взаимного расположения вершин массива их можно выбирать так, чтобы

1) пара значений $\beta_1^{(ij)}$ и $\beta_2^{(ij)}$ для каждого элементарного фрагмента была своя,

2) на каждом единичном квадрате $[0, 1] \times [0, 1]$ функции $\beta_1(u, v)$ и $\beta_2(u, v)$ линейно зависели от параметров u и v или

3) функции $\beta_1^{(ij)}(u, v)$ и $\beta_2^{(ij)}(u, v)$ были разными для разных элементарных фрагментов ($i = 1, ..., m-2, j = 1, ..., n-2$).

Как правило, выбор параметров формы β_1 и β_2 определяется взаимным расположением вершин в массиве. Если размеры четырехугольников

$$\mathbf{P}_{i,j}\mathbf{P}_{i+1,j}\mathbf{P}_{i,j+1}\mathbf{P}_{i+1,j+1},$$

$$0 \le i \le m-1, \quad 0 \le j \le n-1,$$

различаются не слишком сильно, то выбор параметров, одинаковых для всех частичных поверхностей, дает достаточно хорошее приближение. Если же этого не наблюдается и разница в размерах этих четырехугольников весьма велика, то хороших результатов можно добиться подбором переменных параметров формы.

2. На взаимное расположение вершин в массиве не накладывается никаких ограничений: некоторые из них могут и совпадать. Однако следует иметь в виду, что в подобных случаях поверхность может потерять свою регулярность. Впрочем, если номера совпадающих вершин сильно разнятся, то никакой потери регулярности не происходит.

Простейшие из случаев совпадения вершин рассматриваются ниже.

4.4.4. Кратные и воображаемые вершины

В приложениях часто удобно использовать массивы, в которых совпадающие (кратные) вершины являются граничными и угловыми. В этом случае в создании граничных Бета-сплайновых фрагментов участвуют опорные вершины в числе меньше 16 (без учета кратности), что сказывается на размерах фрагментов, а также позволяет получать априорную информацию о поведении составной поверхности вблизи границы. Составная Бета-сплайновая поверхность, как правило, не содержит ни одной из вершин массива, который ее порождает.

Однако подбором вспомогательных вершин и построением дополнительных элементарных поверхностей можно добиться того, чтобы граничные кривые новой составной поверхности располагались ближе к граничным и угловым вершинам. Обычно это проводится путем использования *кратных* или *воображаемых* вершин.

А. Двойные вершины

Зададим $2m + 2n + 8$ новых вершин, положив

$$\mathbf{P}_{-1,j} = \mathbf{P}_{0,j}, \quad j = 0, 1, \ldots, n, \quad \mathbf{P}_{m+1,j} = \mathbf{P}_{m,j}, \quad j = 0, 1, \ldots, n,$$

$$\mathbf{P}_{i,-1} = \mathbf{P}_{i0}, \quad i = 0, 1, \ldots, m, \quad \mathbf{P}_{i,n+1} = \mathbf{P}_{i,n}, \quad i = 0, 1, \ldots, m,$$

$$\mathbf{P}_{-1,-1} = \mathbf{P}_{00}, \quad \mathbf{P}_{-1,n+1} = \mathbf{P}_{0n}, \quad \mathbf{P}_{m+1,-1} = \mathbf{P}_{m0}, \quad \mathbf{P}_{m+1,n+1} = \mathbf{P}_{mn}.$$

Эти вершины как бы "окаймляют" заданный массив $\mathbf{P}$ и вместе с ним образуют новый

$$\mathbf{P}^* = \{\mathbf{P}_{ij}, \quad i = -1, \ldots, m+1, \quad j = -1, \ldots, n+1\}$$

из $(m + 3)(n + 3)$ вершин.

Замечание

Как множества точек массивы $\mathbf{P}$ *и* $\mathbf{P}^*$ *неразличимы, так как все добавленные вершины отличаются от заданных только номерами - геометрически новых вершин не появилось.*

Составная Бета-сплайновая поверхность S^*, определяемая массивом $\mathbf{P}^*$, состоит из mn фрагментов $S^{(i,j)}$, радиусы-векторы которых вычисляются по формулам

$$\mathbf{R}^{(i,j)}(u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} b_k(u) b_l(v) \mathbf{P}_{k-1+i,l-1+j},$$

$$0 \le u, v \le 1, \quad i = -1, \ldots, m-1, \quad j = -1, \ldots, n-1.$$

Только $2m + 2n$ - 4 из этих элементарных поверхностей

$$S^{(0,j)}, \quad j = 0, \ldots, n-2, \quad S^{(i,n-1)}, \quad i = 0, \ldots, m-2,$$

$$S^{(m-1,j)}, \quad j = 1, \ldots, n-1, \quad S^{(i,0)}, \quad i = 1, \ldots, m-1,$$

являются новыми (ясно, что в построении каждого из новых фрагментов принимает участие хотя бы одна новая вершина).

Взаимное расположение составных Бета-сплайновых поверхностей - старой S и новой S^* - хорошо видно на рис. 4.24. Новые фрагменты схематически представлены на рис. 4.25.

Векторы скручивания в угловых вершинах четырех угловых фрагментов $S^{(0,0)}, S^{(0,n-1)}, S^{(m-1,n-1)}, S^{(m-1,0)}$ поверхности S^*

$$\mathbf{R}_{uv}^{(0,0)}(0,0) = \frac{36}{\delta^2}(\mathbf{P}_{00} - \mathbf{P}_{01} + \mathbf{P}_{11} - \mathbf{P}_{10}),$$

$$\mathbf{R}_{uv}^{(0,n-1)}(0,1) = \frac{36}{\delta^2}(\mathbf{P}_{0,n-1} - \mathbf{P}_{0n} + \mathbf{P}_{1n} - \mathbf{P}_{1,n-1}),$$

$$\mathbf{R}_{uv}^{(m-1,n-1)}(1,1) = \frac{36}{\delta^2}(\mathbf{P}_{m-1,n-1} - \mathbf{P}_{m-1,n} + \mathbf{P}_{mn} - \mathbf{P}_{m,n-1}),$$

$$\mathbf{R}_{uv}^{(m-1,0)}(1,0) = \frac{36}{\delta^2}(\mathbf{P}_{m-1,0} - \mathbf{P}_{m-1,1} + \mathbf{P}_{11} - \mathbf{P}_{m0})$$

с точностью до множителя равны векторам скручивания билинейных поверхностей, построенных соответственно на четверках вершин

$$\mathbf{P}_{00}, \quad \mathbf{P}_{01}, \quad \mathbf{P}_{11}, \quad \mathbf{P}_{10};$$

$$\mathbf{P}_{0,n-1}, \quad \mathbf{P}_{0n}, \quad \mathbf{P}_{1n}, \quad \mathbf{P}_{1,n-1};$$

$$\mathbf{P}_{m-1,n-1}, \quad \mathbf{P}_{m-1,n}, \quad \mathbf{P}_{mn}, \quad \mathbf{P}_{m,n-1};$$

$$\mathbf{P}_{m-1,0}, \quad \mathbf{P}_{m,0}, \quad \mathbf{P}_{m1}, \quad \mathbf{P}_{m-1,1}.$$

Б. Тройные вершины

В дополнение к уже взятым $2m + 2n + 8$ вершинам зададим еще $2m + 2n + 16$ новых вершин, положив

$$\mathbf{P}_{-2,j} = \mathbf{P}_{-1,j},\ j = -1, \ldots, n+1, \quad \mathbf{P}_{m+2,j} = \mathbf{P}_{m+1,j},\ j = -1, \ldots, n+1,$$

$$\mathbf{P}_{-2,j} = \mathbf{P}_{-1,j},\ j = -1, \ldots, n+1, \quad \mathbf{P}_{m+2,j} = \mathbf{P}_{m+1,j},\ j = -1, \ldots, n+1,$$

$$\mathbf{P}_{-2,-2} = \mathbf{P}_{-1,-1}, \quad \mathbf{P}_{-2,n+2} = \mathbf{P}_{-1,n+1},$$

$$\mathbf{P}_{m+2,-2} = \mathbf{P}_{m+1,-1}, \quad \mathbf{P}_{m+2,n+2} = \mathbf{P}_{m+1,n+1},$$

и с учетом добавленных вершин построим $2m + 2n + 4$ новых элементарных поверхности

$$S^{(-1,j)}, \quad j = -1, \ldots, n-1, \quad S^{(i,n)}, \quad i = -1, \ldots, m-1,$$

$$S^{(m,j)}, \quad j = 0, \ldots, n, \quad S^{(i,-1)}, \quad i = 0, \ldots, m,$$

задав их параметрическими уравнениями вида

$$\mathbf{R}^{(i,j)}(u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} b_k(u) b_l(v) \mathbf{P}_{k-1+i,\,l-1+j}, \quad 0 \le u, v \le 1.$$

В результате получим новую Бета-сплайновую поверхность S^{**}, состоящую из $(m + 2)(n + 2)$ фрагментов.

Некоторое представление о взаимном расположении новых фрагментов поверхности S^{**} относительно старой поверхности S^{*} могут дать схематические изображения, приведенные на рис. 4.26.

Все 4 угловых фрагмента $S^{(-1,-1)}$, $S^{(-1,n)}$, $S^{(m,n)}$, $S^{(m,-1)}$ поверхности S^{**} лежат на билинейных поверхностях. Векторы скручивания в точках этих четырех фрагментов постоянны по направлению и имеют соответственно вид:

$$\mathbf{R}_{uv}^{(-1,-1)}(u,v) = \frac{36u^2v^2}{\delta}(\mathbf{P}_{00} - \mathbf{P}_{01} + \mathbf{P}_{11} - \mathbf{P}_{10}),$$

$$\mathbf{R}_{uv}^{(-1,n)}(u,v) = \frac{36u^2v^2}{\delta}(\mathbf{P}_{0,n-1} - \mathbf{P}_{0n} + \mathbf{P}_{1n} - \mathbf{P}_{1,n-1}),$$

$$\mathbf{R}_{uv}^{(m,n)}(u,v) = \frac{36u^2v^2}{\delta}(\mathbf{P}_{m-1,n-1} - \mathbf{P}_{m-1,n} + \mathbf{P}_{mn} - \mathbf{P}_{m,n-1}),$$

$$\mathbf{R}_{uv}^{(m,-1)}(u,v) = \frac{36u^2v^2}{\delta}(\mathbf{P}_{m-1,0} - \mathbf{P}_{m-1,1} + \mathbf{P}_{11} - \mathbf{P}_{m0}).$$

Кроме того, все 4 угловые вершины лежат в углах составной Бета-сплайновой поверхности S^{**}, построенной при помощи тройных вершин

$$\mathbf{R}^{(-1,-1)}(0,0) = \mathbf{P}_0, \quad \mathbf{R}^{(-1,n)}(0,0) = \mathbf{P}_{0n},$$

$$\mathbf{R}^{(m,n)}(0,0) = \mathbf{P}_{mn}, \quad \mathbf{R}^{(m,-1)}(0,0) = \mathbf{P}_{m0}$$

(см. рис. 4.27).

В. Воображаемые вершины

Выбором дополнительных вершин к массиву

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, \ldots, m, \quad j = 0, \ldots, n\}$$

можно добиться выполнения различных условий на границе составной поверхности. Обычно новые (воображаемые) вершины ищутся в виде линейных комбинаций заданных вершин.

Рассмотрим составную Бета-сплайновую поверхность S^{*}, порожденную массивом

$$\mathbf{P}^{*} = \{\mathbf{P}_{ij}, \quad i = -1, \ldots, m+1, \; j = -1, \ldots, n+1\},$$

$2m + 2n + 8$ вершин которого подобраны посредством следующих формул:

$$\mathbf{P}_{-1,-1} = \frac{2\beta_1^2 + \beta_2}{2\beta_1^3}(\mathbf{P}_{0,-1} - \mathbf{P}_{1,-1}) + \mathbf{P}_{0,-1},$$

$$\mathbf{P}_{m+1,\,n+1} = \frac{2\beta_1^{\,2} + \beta_2}{2\beta_1^{\,3}}(\mathbf{P}_{m+1,n} - \mathbf{P}_{m+1,n-1}) + \mathbf{P}_{m+1,n},$$

$$\mathbf{P}_{-1,\,j} = \frac{2\beta_1^{\,2} + \beta_2}{2\beta_1^{\,3}}(\mathbf{P}_{0,j} - \mathbf{P}_{1,j}) + \mathbf{P}_{0,j}, \quad j = 0, \ldots, n,$$

$$\mathbf{P}_{i,-1} = \frac{2\beta_1^{\,2} + \beta_2}{2\beta_1^{\,3}}(\mathbf{P}_{i,-1} - \mathbf{P}_{i,-1}) + \mathbf{P}_{i,-1}, \quad i = 0, \ldots, m,$$

$$\mathbf{P}_{-1,n+1} = \frac{2\beta_1 + \beta_2}{2}(\mathbf{P}_{-1,n} - \mathbf{P}_{-1,n-1}) + \mathbf{P}_{-1,n},$$

$$\mathbf{P}_{m+1,-1} = \frac{2\beta_1 + \beta_2}{2}(\mathbf{P}_{m,-1} - \mathbf{P}_{m-1,-1}) + \mathbf{P}_{m,-1},$$

$$\mathbf{P}_{i,n+1} = \frac{2\beta_1 + \beta_2}{2}(\mathbf{P}_{i,n} - \mathbf{P}_{i,n-1}) + \mathbf{P}_{i,n}, \quad i = 0, \ldots, m,$$

$$\mathbf{P}_{m+1,j} = \frac{2\beta_1 + \beta_2}{2}(\mathbf{P}_{m,j} - \mathbf{P}_{m-1,j}) + \mathbf{P}_{m,j}, \quad j = 0, \ldots, n.$$

Составная Бета-сплайновая поверхность S^*, определяемая массивом $\mathbf{P}^*$, обладает следующим свойством: кривизна общей кривой соседних фрагментов в точке, лежащей на граничной кривой этой поверхности, равна нулю.

В рассматриваемом случае векторы скручивания в угловых точках поверхности S^* с точностью до множителя равны векторам скручивания билинейных поверхностей, построенных на определяемых этими точками четверках вершин.

Например, вектор скручивания в угловой точке фрагмента $S^{(0,\,0)}$ равен

$$\alpha(\mathbf{P}_{00} - \mathbf{P}_{01} + \mathbf{P}_{11} - \mathbf{P}_{10}),$$

где

$$\alpha = \frac{9}{\delta^{\,2}}(4\beta_1^{\,4} + 8\beta_1^{\,3} + 4\beta_1^{\,2}\beta_2 + 4\beta_1^{\,2} + 4\beta_1\beta_2 + \beta_2^{\,2}).$$

4.4.5. Программная реализация

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
beta_surface(b1, b2, p, m, n, div_x, div_y, output ),
```

где

```
Flt b1, b2 (in) - параметры beta1 и beta2
Point **p  (in) - опорный граф сплайна ,
                  его вершины Pij, i = 0...(m-1),
                                   j = 0...(n-1)
int m, n   (in) - число точек опорного графа,
int div_x  (in) - число звеньев в u-ломаной,
int div_y  (in) - число звеньев в v-ломаной,
Point **output (out)- граф, аппроксимирующий сплайн,
                  его вершины Qij, i = 0...(m-3)*div_x,
                                   j = 0...(n-3)*div_y
```

Приведем полностью текст этой программы.

```
#include "common.h"

static Flt beta1, beta2 ;
static Flt b12, b13, b22, b23 ;
static Flt delta, d ;

Flt b( int i, Flt t )
{
    Flt s = 1.0 - t ;
    Flt t2 = t * t ;
    Flt t3 = t2 * t ;
    switch( i )
    {
        case 0: return 2 * b13 * d * s * s * s ;
        case 1: return d*(2*b13*t*(t2-3*t+3)+2*b12*(t3-
                          3*t2+2)+2*beta1*(t3-3*t+2)
                          +beta2*(2*t3-3*t2+1)) ;
        case 2: return d*( 2*b12*t2*(-t+3) + 2*beta1*t*(-
                          t2+3)+beta2*t2*(-2*t+3)+2*(-t3+1)) ;
        case 3: return 2 * t3 * d ;
    }
}

void beta_3_3( Point **p, int n, Flt u, Flt v, int i, int j,
Point out )
{
    int k, l ;
    Flt yy ;
    Point t = { 0, 0, 0 } ;
    for( k = 0; k < 4; k ++ )
    {
        Point s = { 0, 0, 0 } ;
        for( l = 0; l < 4; l ++ )
        {
            Flt tt = b( l, v ) ;
            VecAddS( tt, pp( p, i+k, j+l, n ), s, s ) ;
        }
        yy = b( k, u ) ;
        VecAddS( yy, s, t, t ) ;
    }
    VecCopy( t, out ) ;
}

void beta_surface( Flt b1, Flt b2, Point **p, int m, int n,
int div_x, int div_y,Point **output )
{
    int i, j, d1, d2 ;
```

```
        Flt u, v, du = 1.0 / div_x, dv = 1.0 / div_y ;
        int no = (n-3)*div_y + 1 ;

        beta1 = b1 ; beta2 = b2 ;
        b12 = beta1 * beta1 ;
        b13 = b12 * beta1 ;
        b22 = beta2 * beta2 ;
        b23 = b22 * beta2 ;

        delta = 2 * b13 + 4 * b12 + 4 * beta1 + beta2 + 2 ;
        d = 1.0 / delta ;

        for( j = 0; j < n - 3; j ++ )
            for( i = 0; i < m - 3; i ++ )
            {
                v = 0 ;
                for( d2 = 0; d2 <= div_y; d2 ++ )
                {
                    u = 0 ;
                    for( d1 = 0; d1 <= div_x; d1 ++ )
                    {
                        beta_3_3( p, n, u, v, i, j,
                                  pp( output, i*div_x+d1,
                                  j*div_y+d2, no )) ;
                        u += du ;
                    }
                    v += dv ;
                }
            }
}
```

Приведем пример использования функции для построения Бета-сплайн-поверхности.

```
#include "lib.h"

/* описание вызываемой функции */

void beta_surface( Flt b1, Flt b2, Point **p, int m, int n,
int div_x, int div_y, Point **output ) ;

/* число точек опорного графа  */
#define  M     (5)
#define DIV    (5)
/* число точек в аппроксимирующем графе  */
#define  N    ((M-3)*DIV+1)

/* инициализация опорного графа */
Point p[ M ][ M ] ;
Point p11 = { .25, 1,  -.25 } ;
Point p40 = { 1.0, -.5, 0.0 } ;
Point p04 = { 0.0, 2,  -1.0 } ;
Point p33 = { .75, 2,  -.75 } ;
/* вспомогательная переменная */
Point tt = { 0,.0, 0 } ;

Flt beta1 = 1.0, beta2 = 0.0 ;

/* описание выходных массивов  */
Point oo[ N ][ N ] ;
```

```
Point EyePos = { .5, 1, 1 } ;
Point LookAt = { .5, 0, .5 } ;
Vec Up = { 0, 1, 0 } ;
Flt Alpha = .7 ;

void main( void )
{
    int i, j ;

    /* заполнение массива  */
    for( i = 0; i < M; i ++ )
        for( j = 0; j < M; j ++ )
        {
            tt[ 0 ] = (Flt )i / ((Flt )( M - 1 )) ;
            tt[ 2 ] = - (Flt )j / ((Flt )( M - 1 )) ;
            VecCopy( tt, p[ i ][ j ] ) ;
        }
    VecCopy( p11, p[ 1 ][ 1 ] ) ;
    VecCopy( p40, p[ 4 ][ 0 ] ) ;
    VecCopy( p04, p[ 0 ][ 4 ] ) ;
    VecCopy( p33, p[ 3 ][ 3 ] ) ;

    /* вычисление значений сплайна */
    beta_surface( beta1, beta2, (Point **)p, M, M, DIV, DIV,
(Point **)cc ) ;

    /* инициализация графической моды  */
    SetVideoMode() ;

    /* инициализация трехмерной графики */
    Init3D( EyePos, LookAt, Up, Alpha ) ;

    /* отображение на дисплее построенного сплайна */
    PlotSurface( (Point **)cc, N, N, 1 ) ;
}
```

Текст программы beta_surface находится в файле beta.c в поддиректории SURFACES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

4.5. Другие сплайновые поверхности

4.5.1. Интерполяционные бикубические поверхности Эрмита

По заданным вершинам

$$\mathbf{P}_{00},\quad \mathbf{P}_{01},\quad \mathbf{P}_{11},\quad \mathbf{P}_{10}$$

и набору векторов

$$\mathbf{X}_{00}, \mathbf{X}_{01}, \mathbf{X}_{10}, \mathbf{X}_{11},\quad \mathbf{Y}_{00}, \mathbf{Y}_{01}, \mathbf{Y}_{10}, \mathbf{Y}_{11},\quad \mathbf{Z}_{00}, \mathbf{Z}_{01}, \mathbf{Z}_{10}, \mathbf{Z}_{11}$$

(элементарная) бикубическая эрмитова поверхность определяется при помощи **матричного уравнения**

$$\mathbf{R}(t) = \mathbf{U}^T\mathbf{M}^T\mathbf{G}\mathbf{M}\mathbf{V},\quad 0 \le u, v \le 1,$$

где

$$\mathbf{R}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix},\qquad \mathbf{U} = \begin{pmatrix} u^0 \\ u^1 \\ u^2 \\ u^3 \end{pmatrix},\qquad \mathbf{V} = \begin{pmatrix} v^0 \\ v^1 \\ v^2 \\ v^3 \end{pmatrix},$$

$$G = \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{Y}_{00} & \mathbf{Y}_{01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{Y}_{10} & \mathbf{Y}_{11} \\ \mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{Z}_{00} & \mathbf{Z}_{01} \\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{Z}_{10} & \mathbf{Z}_{11} \end{pmatrix},\qquad \mathbf{M} = \begin{pmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

Матрица **M** называется *базисной матрицей бикубической эрмитовой поверхности*, а матрица **G** - ее *геометрической матрицей.*

Геометрический смысл векторов, определяющих элементарную бикубическую эрмитову поверхность

Вершины $\mathbf{P}_{ij}$ ($i = 0, 1,\ j = 0, 1$) являются угловыми точками элементарной бикубической эрмитовой поверхности (рис. 4.28).

В точке $\mathbf{P}_{ij}$ вектор $\mathbf{X}_{ij}$ касается *u*-линии эрмитовой поверхности, а вектор $\mathbf{Y}_{ij}$ касается ее *v*-линии в этой вершине (для того чтобы построенная поверхность была регулярной, естественно потребовать, чтобы каждая пара векторов $\mathbf{X}_{ij}$ и $\mathbf{Y}_{ij}$ была неколлинеарна). Что касается вектора $\mathbf{Z}_{ij}$, то это вектор скручивания в вершине $\mathbf{P}_{ij}$.

В частности, в вершине $\mathbf{P}_{00}$ элементарной эрмитовой поверхности (при $u = 0$ и $v = 0$) имеем

$$\mathbf{R}(0, 0)=\mathbf{P}_{00},$$
$$\mathbf{R}_u(0, 0)=\mathbf{X}_{00},$$
$$\mathbf{R}_v(0, 0)=\mathbf{Y}_{00},$$
$$\mathbf{R}_{uv}(0, 0)=\mathbf{Z}_{00}.$$

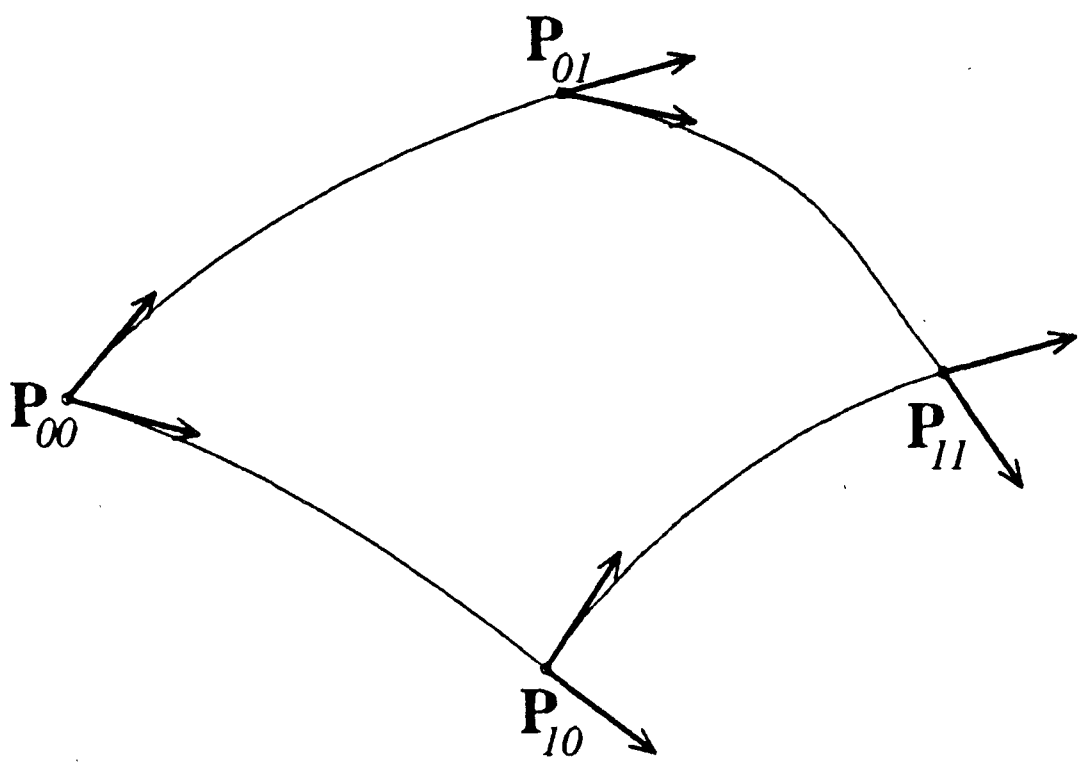


Рис. 4.28

При линейной замене параметров

$$u \mapsto \xi = (1 - u)a + ub,$$

$$u \mapsto \eta = (1 - u)c + ud,$$

преобразующей единичный квадрат $[0,1]\times[0,1]$ в прямоугольник $[a,b]\times[c,d]$, форма элементарной поверхности Эрмита изменяется. Для ее сохранения необходимо заменить векторы $\mathbf{X}_{ij}$ и $\mathbf{Y}_{ij}$ и $\mathbf{Z}_{ij}$ ($i = 0, ..., m$, $j = 0, ..., n$) соответственно на

$$\frac{1}{b-a}\mathbf{X}_{ij}, \quad \frac{1}{d-c}\mathbf{Y}_{ij}, \quad \frac{1}{(b-a)(c-d)}\mathbf{Z}_{ij}.$$

Составные бикубические поверхности Эрмита

(Составной) бикубической поверхностью Эрмита, определяемой массивом

$$\begin{array}{llll}
\mathbf{P}_{00}, & \mathbf{P}_{01}, & \cdots \mathbf{P}_{0,n-1}, & \mathbf{P}_{0n}, \\
\mathbf{P}_{10}, & \mathbf{P}_{11}, & \cdots \mathbf{P}_{1,n-1}, & \mathbf{P}_{1n}, \\
\cdots & \cdots & \cdots \cdots & \cdots \\
\mathbf{P}_{m-1,0}, & \mathbf{P}_{m-1,1}, & \cdots \mathbf{P}_{m-1,n-1}, & \mathbf{P}_{m-1,n}, \\
\mathbf{P}_{m0}, & \mathbf{P}_{m1}, & \cdots \mathbf{P}_{m,n-1}, & \mathbf{P}_{mn},
\end{array}$$

и векторами

$$\mathbf{X}_{kj}, \quad k = 0, \quad m,\ j = 0, \ldots, n; \quad \mathbf{Y}_{il}, i = 0, \ldots, m, \quad l = 0, n;$$

$$\mathbf{Z}_{00}, \quad \mathbf{Z}_{0n}, \quad \mathbf{Z}_{m0}, \quad \mathbf{Z}_{mn},$$

называется поверхность S, которую можно представить в виде объединения элементарных бикубических поверхностей Эрмита $S^{(1,1)}, \ldots, S^{(m,n)}$,

$$S = \cup_{i=1}^{m} \cup_{j=1}^{n} S^{(i,j)}$$

(рис. 4.29); (i, j)-я поверхность $S^{(i,j)}$ описывается параметрическим уравнением следующего вида:

$$\mathbf{R}^{(i,j)}(u,v) = \mathbf{U}^T \mathbf{M}^T \mathbf{G}^{(i,j)} \mathbf{M} \mathbf{V},$$

$$0 \le u,v \le 1,\ i = 1,\ldots,m,\ j = 1,\ldots,n,$$

где **M** - базисная матрица бикубической эрмитовой поверхности, а геометрическая матрица $\mathbf{G}^{(i,j)}$ имеет следующий вид:

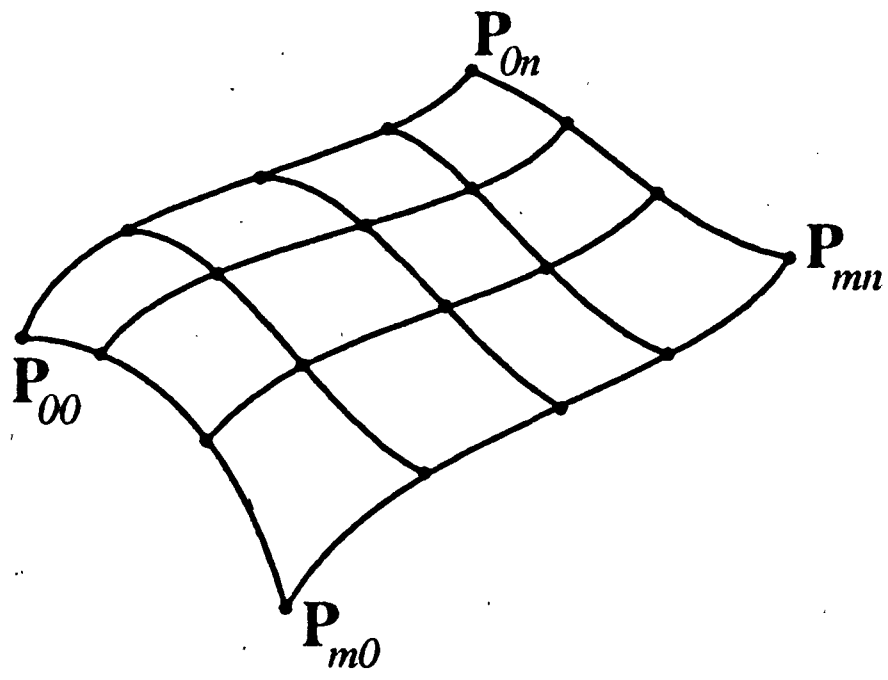


Рис. 4.29

$$G^{(i,\,j)} = \begin{pmatrix} \mathbf{P}_{i-1,\,j-1} & \mathbf{P}_{i-1,j} & \mathbf{Y}_{i-1,j-1} & \mathbf{Y}_{i-1,j} \\ \mathbf{P}_{i,j-1} & \mathbf{P}_{ij} & \mathbf{Y}_{i,j-1} & \mathbf{Y}_{ij} \\ \mathbf{X}_{i-1,j-1} & \mathbf{X}_{i-1,j} & \mathbf{Z}_{i-1,j-1} & \mathbf{Z}_{i-1,j} \\ \mathbf{X}_{i,j-1} & \mathbf{X}_{ij} & \mathbf{Z}_{i,j-1} & \mathbf{Z}_{ij} \end{pmatrix}.$$

Векторы

$$\mathbf{X}_{ij},, \quad i = 1, \ldots, m\text{-}1, \quad j = 0, \ldots, n,$$

$$\mathbf{Y}_{ij},, \quad i = 0, \ldots, m, \quad j = 1, \ldots, n\text{-}1,$$

$$\mathbf{Z}_{ij}, \quad i \ne 0, m, \quad j \ne 0, \ldots, n,$$

можно выбрать так, чтобы составная поверхность Эрмита была C^2-гладкой. Вся исходная информация, необходимая для их определения, может быть записана в виде табл. 4.1.

Таблица 4.1.

$\mathbf{Z}_{00}$	$\mathbf{X}_{00}$	$\mathbf{X}_{01}$	...	$\mathbf{X}_{0,\,n-1}$	$\mathbf{X}_{0n}$	$\mathbf{Z}_{0n}$
$\mathbf{Y}_{00}$	$\mathbf{P}_{00}$	$\mathbf{P}_{01}$	...	$\mathbf{P}_{0,\,n-1}$	$\mathbf{P}_{0n}$	$\mathbf{Y}_{0n}$
$\mathbf{Y}_{10}$	$\mathbf{P}_{10}$	$\mathbf{P}_{11}$	...	$\mathbf{P}_{1,\,n-1}$	$\mathbf{P}_{1n}$	$\mathbf{Y}_{1n}$
...	...	...	...	...	...	...
$\mathbf{Y}_{m-1,\,0}$	$\mathbf{P}_{m-1,\,0}$	$\mathbf{P}_{m-1,1}$	...	$\mathbf{P}_{m-1,\,n-1}$	$\mathbf{P}_{m-1,\,n}$	$\mathbf{Y}_{m-1,\,n}$
$\mathbf{Y}_{m0}$	$\mathbf{P}_{m0}$	$\mathbf{P}_{m1}$	...	$\mathbf{P}_{m,\,n-1}$	$\mathbf{P}_{mn}$	$\mathbf{Y}_{mn}$
$\mathbf{Z}_{m0}$	$\mathbf{X}_{m0}$	$\mathbf{X}_{m1}$	...	$\mathbf{X}_{m,\,n-1}$	$\mathbf{X}_{mn}$	$\mathbf{Z}_{mn}$

Опишем формулы, посредством которых вычисляются эти векторы.

Начнем с матричных формул для вычисления векторов $\mathbf{X}_{ij}$ и $\mathbf{Y}_{ij}$ во внутренних вершинах массива **P**. Имеем:

$$\begin{pmatrix} 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & & \ddots & & & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{X}_{00} & \cdots & \mathbf{X}_{0n} \\ \mathbf{X}_{10} & \cdots & \mathbf{X}_{1n} \\ \vdots & & \vdots \\ \mathbf{X}_{m-1,0} & \cdots & \mathbf{X}_{m-1,0} \\ \mathbf{X}_{m0} & \cdots & \mathbf{X}_{m,0} \end{pmatrix} =$$

$$\begin{pmatrix} -3 & 0 & 3 & & & & \\ & -3 & 0 & 3 & & & \\ & & & \ddots & & & \\ & & & & -3 & 0 & -3 \\ & & & & & -3 & 0 & -3 \end{pmatrix} \begin{pmatrix} \mathbf{P}_{00} & \cdots & \mathbf{P}_{0n} \\ \mathbf{P}_{10} & \cdots & \mathbf{P}_{1n} \\ \vdots & & \vdots \\ \mathbf{P}_{m-1,0} & \cdots & \mathbf{P}_{m-1,0} \\ \mathbf{P}_{m0} & \cdots & \mathbf{P}_{m,0} \end{pmatrix}$$

(прямоугольные скалярные матрицы имеют одинаковые размеры $(m-1)\times(m+1)$, элементы первой и последней строк матрицы $\mathbf{X}$ известны, матричное уравнение однозначно разрешимо),

$$\begin{pmatrix} 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & & \ddots & & & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{Y}_{00} & \cdots & \mathbf{Y}_{m0} \\ \mathbf{Y}_{01} & \cdots & \mathbf{Y}_{m1} \\ \vdots & & \vdots \\ \mathbf{Y}_{0,n-1} & \cdots & \mathbf{Y}_{0,n-1} \\ \mathbf{Y}_{n0} & \cdots & \mathbf{Y}_{mn} \end{pmatrix} =$$

$$\begin{pmatrix} -3 & 0 & 3 & & & & \\ & -3 & 0 & 3 & & & \\ & & & \ddots & & & \\ & & & & -3 & 0 & -3 \\ & & & & & -3 & 0 & -3 \end{pmatrix} \begin{pmatrix} \mathbf{P}_{00} & \cdots & \mathbf{P}_{m0} \\ \mathbf{P}_{01} & \cdots & \mathbf{P}_{m1} \\ \vdots & & \vdots \\ \mathbf{P}_{0,n-1} & \cdots & \mathbf{P}_{0,n-1} \\ \mathbf{P}_{0n} & \cdots & \mathbf{P}_{mn} \end{pmatrix}$$

(прямоугольные скалярные матрицы имеют одинаковые размеры $(n-1)\times(n+1)$, элементы первой и последней строк матрицы $\mathbf{Y}$ известны, матричное уравнение однозначно разрешимо).

Если векторы $\mathbf{X}_{ij}$ и $\mathbf{Y}_{ij}$ найдены, то векторы скручивания $\mathbf{Z}_{ij}$ во всех внутренних и граничных вершинах массива $\mathbf{P}$ (за исключением четырех уже заданных угловых $\mathbf{Z}_{00}$, $\mathbf{Z}_{0n}$, $\mathbf{Z}_{1m}$, $\mathbf{Z}_{mn}$) можно вычислить так:

сначала по формулам

$$\begin{pmatrix} 1 & 4 & 1 & & & & & \\ & 1 & 4 & 1 & & & & \\ & & & \ddots & & & & \\ & & & & 1 & 4 & 1 & \\ & & & & & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{Z}_{k0} \\ \mathbf{Z}_{k1} \\ \vdots \\ \mathbf{Z}_{k,n-1} \\ \mathbf{Z}_{kn} \end{pmatrix} =$$

$$\begin{pmatrix} -3 & 0 & 3 & & & & & \\ & -3 & 0 & 3 & & & & \\ & & & \ddots & & & & \\ & & & & -3 & 0 & -3 & \\ & & & & & -3 & 0 & -3 \end{pmatrix} \begin{pmatrix} \mathbf{X}_{k0} \\ \mathbf{X}_{k1} \\ \vdots \\ \mathbf{X}_{k,n-1} \\ \mathbf{X}_{kn} \end{pmatrix}, \quad k = 0, m,$$

вычисляются векторы скручивания в граничных вершинах $\mathbf{P}_{kj}$ ($j = 1, ..., n\text{-}1$) (прямоугольные скалярные матрицы имеют одинаковые размеры $(n\text{-}1) \times (n+1)$, первый и последний элементы столбца $\mathbf{Z}$ известны, матричное уравнение однозначно разрешимо),
а затем по формулам

$$\begin{pmatrix} 1 & 4 & 1 & & & & & \\ & 1 & 4 & 1 & & & & \\ & & & \ddots & & & & \\ & & & & 1 & 4 & 1 & \\ & & & & & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{Z}_{0j} \\ \mathbf{Z}_{1j} \\ \vdots \\ \mathbf{Z}_{m-1,j} \\ \mathbf{Z}_{mj} \end{pmatrix} =$$

$$\begin{pmatrix} -3 & 0 & 3 & & & & & \\ & -3 & 0 & 3 & & & & \\ & & & \ddots & & & & \\ & & & & -3 & 0 & -3 & \\ & & & & & -3 & 0 & -3 \end{pmatrix} \begin{pmatrix} \mathbf{Y}_{0j} \\ \mathbf{Y}_{1j} \\ \vdots \\ \mathbf{Y}_{m-1,j} \\ \mathbf{Y}_{mj} \end{pmatrix}, \qquad l = 0, n,$$

вычисляются векторы скручивания в граничных вершинах $\mathbf{P}_{il}$ ($i = 1, ..., m\text{-}1$) (прямоугольные скалярные матрицы имеют одинаковые размеры $(m - 1) \times (m + 1)$, первый и последний элементы столбца $\mathbf{Z}$ известны, матричное уравнение однозначно разрешимо).

Тем самым полностью построены массивы

производных радиуса-вектора вдоль *u*-линий

$$\begin{matrix} \mathbf{X}_{00} & \mathbf{X}_{01} & \cdots & \mathbf{X}_{0,n-1} & \mathbf{X}_{0,n} \\ \mathbf{X}_{10} & \mathbf{X}_{11} & \cdots & \mathbf{X}_{1,n-1} & \mathbf{X}_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{X}_{m-1,0} & \mathbf{X}_{m-1,1} & \cdots & \mathbf{X}_{m-1,n-1} & \mathbf{X}_{m-1,n} \\ \mathbf{X}_{m0} & \mathbf{X}_{m1} & \cdots & \mathbf{X}_{m,n-1} & \mathbf{X}_{mn} \end{matrix}$$

производных радиуса-вектора вдоль *v*-линий

$$\begin{matrix} \mathbf{Y}_{00} & \mathbf{Y}_{01} & \cdots & \mathbf{Y}_{0,n-1} & \mathbf{Y}_{0,n} \\ \mathbf{Y}_{10} & \mathbf{Y}_{11} & \cdots & \mathbf{Y}_{1,n-1} & \mathbf{Y}_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{Y}_{m-1,0} & \mathbf{Y}_{m-1,1} & \cdots & \mathbf{Y}_{m-1,n-1} & \mathbf{Y}_{m-1,n} \\ \mathbf{Y}_{m0} & \mathbf{Y}_{m1} & \cdots & \mathbf{Y}_{m,n-1} & \mathbf{Y}_{mn} \end{matrix}$$

векторов скручивания

$$\begin{matrix} \mathbf{Z}_{00} & \mathbf{Z}_{01} & \cdots & \mathbf{Z}_{0.n-1} & \mathbf{Z}_{0,n} \\ \mathbf{Z}_{10} & \mathbf{Z}_{11} & \cdots & \mathbf{Z}_{1,n-1} & \mathbf{Z}_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{Z}_{m-1,0} & \mathbf{Z}_{m-1,1} & \cdots & \mathbf{Z}_{m-1,n-1} & \mathbf{Z}_{m-1,n} \\ \mathbf{Z}_{m0} & \mathbf{Z}_{m1} & \cdots & \mathbf{Z}_{m,n-1} & \mathbf{Z}_{mn} \end{matrix}$$

Завершающим шагом в построении составной эрмитовой поверхности является вычисление радиусов-векторов $\mathbf{R}^{(i,j)}(u, v)$ элементарных фрагментов $S^{(i,j)}$ по формуле

$$\mathbf{R}^{(i,j)}(u, v) = \mathbf{U}^T \mathbf{M}^T \mathbf{G}^{(i,j)} \mathbf{M} \mathbf{V},$$

$$0 \le u, v \le 1, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Свойства составной бикубической поверхности Эрмита

Составная бикубическая поверхность Эрмита, порожденная массивом

$$\mathbf{P} = \{\mathbf{P}_{ij}, \quad i = 0, \ldots, m, \quad j = 0, \ldots, n\}$$

и векторами

$$\mathbf{X}_{kj}, \quad k = 0, m, \quad j = 0, \ldots, n,$$

$$\mathbf{Y}_{il}, \quad i = 0, \ldots, m, \quad l = 0, n,$$

$$\mathbf{Z}_{00}, \quad \mathbf{Z}_{0n}, \quad \mathbf{Z}_{m0} \quad \mathbf{Z}_{mn}:$$

1⁺ является C^2-гладкой поверхностью (имеет непрерывную кривизну);

2⁺ проходит через все вершины массива **P**;

3⁺ касательные векторы $\mathbf{X}_{ij}$ и $\mathbf{Y}_{ij}$ к координатным линиям $u=const$ и $v=const$ составной поверхности S во внутренней вершине $\mathbf{P}_{ij}$ ($i = 1, ..., m-1$, $j = 1, ..., n-1$), а также векторы скручивания $\mathbf{Z}_{ij}$ однозначно определяются через вершины массива P и векторы $\mathbf{X}_{ij}$, $\mathbf{Y}_{ij}$ и $\mathbf{Z}_{ij}$, заданные в граничных вершинах;

4⁻ не лежит в выпуклой оболочке, порожденной заданным массивом;

5⁻ изменение одной вершины в массиве или одного из заданных векторов приводит к изменению всей поверхности;

6⁻ при добавлении в массив одной вершины возникает необходимость пересчета всех параметрических уравнений;

7⁺ составная бикубическая поверхность Эрмита аффинно-инвариантна;

8⁻ заданные векторы однозначно определяют составную бикубическую поверхность Эрмита, не давая возможности хоть как-то влиять на ее форму;

9⁻ бикубическая поверхность Эрмита проективно-неинвариантна.

4.5.2. Программная реализация

Описанный алгоритм реализован в виде функции на языке С. Обращение к ней имеет следующий вид:

```
hermite_surface(p, x, y, z, m, n, div_x, div_y,output),
```

где

```
    Point **p (in) - опорный граф сплайна,
                     его точки Pij, i = 0...(m-1), j = 0...(n-1),
    Point **x (in) - массив производных по u,
                     его элементы Xij, i = 0...(m-1),
                                       j = 0...(n-1),
                     элементам Xkj, k = 0, m-1, j = 0 ... n-1,
                     следует присвоить граничные значения,
    Point **y (in) - массив производных по v,
                     его элементы Yij, i = 0...(m-1),
                                       j = 0... n-1),
                     элементам Xkj, i = 0, m-1, j = 0, n-1,
                     следует присвоить граничные значения,
    Point **z (in) - массив векторов кручения,
                     его элементы Zij, i = 0...(m-1), j=0...(n-1),
                     элементам Z[0][0], Z[0][n-1],
                               Z[m-1][0], Z[m-1][n-1]
                     следует присвоить граничные значения,
    int m, n  (in) - число точек в опорном графе,
    int div_x (in) - число звеньев в u-ломаной,
    int div_y (in) - число звеньев в v-ломаной,
    Point **output(out) - граф, аппроксимирующий сплайн,
                     его элементы Qij, i = 0...(m-1)*div_x,
                                       j = 0...(n-1)*div_y
```

Приведем полностью текст этой программы.

```
#include "common.h"

void Progonka( Point y[], Point F[], int n )
{
    static Flt alpha[ 30 ] ;
    static Point beta[ 30 ] ;
    int i ;

    alpha[ 1 ] = 0.0 ;
    VecCopy( y[0], beta[1] ) ;
    for( i = 1; i < n; i ++ )
    {
        alpha[ i+1 ] = 1.0 / ( - 4.0 - alpha[ i ] ) ;
        VecSub( beta[i], F[i], beta[i+1] ) ;
        VecMul( beta[ i+1 ], alpha[ i+1 ], beta[ i+1 ] ) ;
    }
    for( i = n-1; i > 0; i -- )
        VecAddS( alpha[ i+1 ], y[ i+1 ], beta[ i+1 ], y[ i ] )
;
}

void SolveForX( Point **x, Point **p, int m, int n )
{
    static Point y[ 30 ], F[ 30 ] ;
    int i, j ;

    for( j = 0; j < n; j ++ )
    {
        for( i = 1; i < m-1; i ++ )
        {
            VecSub( pp( p, i+1, j, n), pp( p, i-1, j, n), F[ i
] ) ;
            VecMul( F[i], 3.0, F[i] ) ;
        }
        VecCopy( pp(x,0,j,n), y[0] ) ;
        VecCopy( pp(x,m-1,j,n), y[m-1] ) ;
        Progonka( y, F, m-1 ) ;
        for( i = 1; i < m-1; i ++ )
            VecCopy( y[i], pp(x, i, j, n) ) ;
    }
}

void SolveForY( Point **y, Point **p, int m, int n )
{
    static Point yy[ 30 ], F[ 30 ] ;
    int i, j ;

    for( i = 0; i < m; i ++ )
    {
        for( j = 1; j < n-1; j ++ )
        {
            VecSub( pp( p, i, j+1, n), pp( p, i, j-1, n), F[ j
] ) ;
            VecMul( F[j], 3.0, F[j] ) ;
        }
        VecCopy( pp(y,i,0,n), yy[0] ) ;
        VecCopy( pp(y,i,n-1,n), yy[n-1] ) ;
        Progonka( yy, F, n-1 ) ;
        for( j = 1; j < n-1; j ++ )
```

```
            VecCopy( yy[j], pp(y, i, j, n) ) ;
        }
    }

    void SolveForZ( Point **x, Point **y, Point **z, int m, int n
    )
    {
        static Point z1[ 30 ], z2[ 30 ] ;
        static Point F1[ 30 ], F2[ 30 ] ;
        int i, j ;

        for( j = 1; j < n-1; j ++ )
        {
            VecSub( pp( x, 0, j+1, n), pp( x, 0, j-1, n), F1[ j ]
    ) ;
            VecMul( F1[j], 3.0, F1[j] ) ;
            VecSub( pp( x, m-1, j+1, n), pp( x, m-1, j-1, n), F2[
    j ] ) ;
            VecMul( F2[j], 3.0, F2[j] ) ;
        }
        VecCopy( pp( z, 0, 0, n ), z1[0] ) ;
        VecCopy( pp( z, 0, n-1, n ), z1[n-1] ) ;
        VecCopy( pp( z, m-1, 0, n ), z2[0] ) ;
        VecCopy( pp( z, m-1, n-1, n ), z2[n-1] ) ;
        Progonka( z1, F1, n-1 ) ;
        Progonka( z2, F2, n-1 ) ;
        for( j = 1; j < n-1; j ++ )
        {
            VecCopy( z1[j], pp(z, 0, j, n) ) ;
            VecCopy( z2[j], pp(z, m-1, j, n) ) ;
        }
        for( j = 0; j < n; j ++ )
        {
            VecCopy( pp(z, 0, j, n), z1[0] ) ;
            VecCopy( pp(z, m-1, j, n), z1[m-1] ) ;
            for( i = 1; i < m-1; i ++ )
            {
                VecSub( pp( y, i+1, j, n), pp( y, i-1, j, n), F1[
    i ] ) ;
                VecMul( F1[i], 3.0, F1[i] ) ;
            }
            Progonka( z1, F1, m-1 ) ;
            for( i = 1; i < m-1; i ++ )
                VecCopy( z1[i], pp(z, i, j, n) ) ;
        }
    }

    Flt mm( int n, Flt v )
    {
        Flt v2 = v*v ;
        Flt v3 = v2*v ;
        switch( n )
        {
            case 0: return 1 - 3*v2+2*v3 ;
            case 1: return 3*v2 - 2*v3 ;
            case 2: return v - 2*v2 + v3 ;
            case 3: return -v2 + v3 ;
        }
    }
```

```
void hermite_3_3( Point **p, Point **x, Point **y, Point **z,
int n, Flt u, Flt v, int i, int j, Point out )
{
    Flt mu[4] = { mm(0, u ), mm( 1, u), mm(2, u), mm( 3, u) }
;
    Flt mv[4] = { mm(0, v ), mm( 1, v), mm(2, v), mm( 3, v) }
;
    Point GV[4], zzz = { 0, 0, 0 } ;
    int k ;

    VecComb( mv[0], pp( p, i-1, j-1, n ), mv[1], pp( p, i-1,
j, n), GV[0] ) ;
    VecAddS( mv[2], pp( y, i-1, j-1, n ), GV[0], GV[0] ) ;
    VecAddS( mv[3], pp( y, i-1, j, n ), GV[0], GV[0] ) ;

    VecComb( mv[0], pp( p, i, j-1, n ), mv[1], pp( p, i, j,
n), GV[1] ) ;
    VecAddS( mv[2], pp( y, i, j-1, n ), GV[1], GV[1] ) ;
    VecAddS( mv[3], pp( y, i, j, n ), GV[1], GV[1] ) ;

    VecComb( mv[0], pp( x, i-1, j-1, n ), mv[1], pp( x, i-1,
j, n), GV[2] ) ;
    VecAddS( mv[2], pp( z, i-1, j-1, n ), GV[2], GV[2] ) ;
    VecAddS( mv[3], pp( z, i-1, j, n ), GV[2], GV[2] ) ;

    VecComb( mv[0], pp( x, i, j-1, n ), mv[1], pp( x, i, j,
n), GV[3] ) ;
    VecAddS( mv[2], pp( z, i, j-1, n ), GV[3], GV[3] ) ;
    VecAddS( mv[3], pp( z, i, j, n ), GV[3], GV[3] ) ;

    for( k = 0; k < 4; k ++ )
        VecAddS( mu[k], GV[k], zzz, zzz ) ;
    VecCopy( zzz, out ) ;
}

void hermite_surface( Point **p, Point **x, Point **y, Point
**z, int m, int n,
                        int div_x, int div_y, Point **output )
{
    int i, j, d1, d2 ;
    Flt u, v, du = 1.0 / div_x, dv = 1.0 / div_y ;
    int no = (n-1)*div_y + 1 ;

    SolveForX( x, p, m, n ) ;
    SolveForY( y, p, m, n ) ;
    SolveForZ( x, y, z, m, n ) ;

    for( j = 1; j < n; j ++ )
        for( i = 1; i < m; i ++ )
        {
            v = 0 ;
            for( d2 = 0; d2 <= div_y; d2 ++ )
            {
                u = 0 ;
                for( d1 = 0; d1 <= div_x; d1 ++ )
                {
                    hermite_3_3( p, x, y, z, n, u, v, i, j,
                                 pp( output, (i-1)*div_x + d1,
(j-1)*div_y + d2, no ) ) ;
                    u += du ;
                }
```

```
                    v += dv ;
                }
            }
    }
```

Приведем пример использования функции для построения сплайн-поверхности.

```
#include "lib.h"

/* описание вызываемой функции */
void hermite_surface( Point **p, Point **x, Point **y,
                      Point **z, int m, int n,int div_x,
                      int div_y, Point **output ) ;

/* число точек в опорном графе */
#define  M     (5)
#define DIV    (5)
/* число точек в аппроксимирующем графе */
#define  N    ((M-1)*DIV+1)

/* инициализация опорного графа */
Point p[ M ][ M ] ;
Point p11 = { .25, 1, -.25  } ;
Point p40 = { 1.0, -.5, 0.0 } ;
Point p04 = { 0.0, 2, -1.0  } ;
Point p33 = { .75, 2, -.75  } ;

Point x[ M ][ M ] ;
Point y[ M ][ M ] ;
Point z[ M ][ M ] ;

/* вспомогательные переменные */
Point tt = { 0, 0, 0 } ;
Point xx = { 1, 0, 0 } ;
Point yy = { 0, 0, -1 } ;
Point zz = { 0, 0, 0 } ;

/* описание выходного массива  */
Point oo[ N ][ N ] ;

Point EyePos = { .5, 3, 3 } ;
Point LookAt = { .5, 0, -5 } ;
Vec Up = { 0, 1, -1 } ;
Flt Alpha = .7 ;

void main( void )
{
    int i, j ;

    /* заполнение массива  */
    for( i = 0; i < M; i ++ )
        for( j = 0; j < M; j ++ )
        {
            tt[ 0 ] = (Flt )i / ((Flt )( M - 1 )) ;
            tt[ 2 ] = - (Flt )j / ((Flt )( M - 1 )) ;
            VecCopy( tt, p[ i ][ j ] ) ;
        }
    VecCopy( p11, p[ 1 ][ 1 ] ) ;
    VecCopy( p40, p[ 4 ][ 0 ] ) ;
```

```
    VecCopy( p04, p[ 0 ][ 4 ] ) ;
    VecCopy( p33, p[ 3 ][ 3 ] ) ;

    for( j = 0; j < M; j ++ )
    {
        VecCopy( xx, x[  0  ][ j ] ) ;
        VecCopy( xx, x[ M-1 ][ j ] ) ;
    }
    for( i = 0; i < M; i ++ )
    {
        VecCopy( y[ i ][  0  ], yy ) ;
        VecCopy( y[ i ][ M-1 ], yy ) ;
    }
    VecCopy( zz, z[  0  ][  0  ] ) ;
    VecCopy( zz, z[  0  ][ M-1 ] ) ;
    VecCopy( zz, z[ M-1 ][  0  ] ) ;
    VecCopy( zz, z[ M-1 ][ M-1 ] ) ;

    /* вычисление значений сплайна  */
    hermite_surface( (Point **)p,
                     (Point **)x,
                     (Point **)y,
                     (Point **)z,
                      M, M, DIV, DIV, (Point **)oo ) ;

    /* инициализация графической моды */
    SetVideoMode() ;

    /* инициализация трехмерной графики  */
    Init3D( EyePos, LookAt, Up, Alpha ) ;

    /* отображение на дисплее построенного сплайна */
    PlotSurface( (Point **)oo, N, N, 1 ) ;
}
```

Текст программы hermite_surface находится в файле hermite.c в поддиректории SURFACES на дискете, которую можно приобрести в издательстве "Диалог-МИФИ". Подробную информацию об именах файлов из этой директории и их содержании можно найти в файле readme.txt из директории SPLINES этой дискеты и в приложении В нашей книги.

4.5.3. Составные неявно заданные кубические поверхности

В последнее время появилось значительное число работ, посвященных построению составных поверхностей, элементарные фрагменты которых описываются неявными алгебраическими уравнениями вида $F(x, y, z) = 0$, где $F(x, y, z)$ - алгебраический многочлен 2-й или 3-й степени. Интерес к таким поверхностям связан с их активным использованием в задачах компьютерной графики. Дело в том, что указанный способ задания оказывается очень удобным при построении реалистических изображений сложных сцен методом трассировки лучей.

Задача построения составных поверхностей указанного типа весьма интересна, но совсем не проста. Не имея возможности останавливаться на этом подробно, мы отсылаем читателя к специальной литературе (несколько последних журнальных публикаций указано в конце книги).

Приложение А.
Программы метода прогонки для трёх- и пятидиагональных матриц

```
      subroutine Progon3 (a,b,c,d,u,v,w,s,t,x,n)
c*************************************************************
c         Программа Progon3 решает систему линейных          *
c         уравнений вида                                     *
c                                                            *
c         a(1)*x(1) + b(1)*x(2)       + c(1)*x(n) = d(1)     *
c         ..................................................  *
c         c(i)*x(i-1) + a(i)*x(i)  + b(i)*x(i+1) = d(i)      *
c         ..................................................  *
c         b(n)*x(1)     + a(n)*x(n-1) + b(n)*x(n)= d(n)      *
c         с трехдиагональной матрицей методом прогонки       *
c     n - число уравнений                                    *
c   a,b,                                                     *
c   c,d - массивы (длины n), содержащие элементы диагоналей *
c         матрицы и правых частей                            *
c   w,s,t,                                                   *
c   u,v - рабочие массивы (длины n+1)                        *
c         Результат:                                         *
c     x - массив (длины  n), содержащий решение системы      *
c                                                            *
c*************************************************************
         dimension a(1),b(1),c(1),d(1),u(1),
     *             v(1),x(1),w(1),s(1),t(1)
c
         u(1) = 0.
         v(1) = 0.
         w(1) = 1.
c
         do  i = 1,n
             i1 = i + 1
             z  = 1./(a(i) + c(i)*v(i))
             v(i1) = -b(i)*z
             u(i1) = (-c(i)*u(i) + d(i))*z
             w(i1) = - c(i)*w(i)*z
         enddo
c
        s(n) = 1.
        t(n) = 0.
        do  i = n-1,1, -1
            s(i) = v(i+1)*s(i+1) + w(i+1)
            t(i) = v(i+1)*t(i+1) + u(i+1)
        enddo
c
        x(n) = (d(n) - b(n)*t(1) - c(n)*t(n-1))/
     *              (a(n) + b(n)*s(1) + c(n)*s(n-1))

        do i = 1, n-1
            x(i) = s(i)*x(n) + t(i)
        enddo
c
         return
         end
```

```
      subroutine Progon5(n,a,b,c,d,e,g,u,v,w,
     *                           p,q,r,s,t,aa,dd,z)
c
c*************************************************************
c     Программа Progon5 решает линейную систему              *
c     уравнений вида                                         *
c  +                                          +  +  +  +  +  +
c  |a1   b1   c1    0   .........0   e1   d1  |  |z1|  |g1|  *
c  |d2   a2   b2   c2    0 ......    0   e2  |  |z2|  |g2|  *
c  |e3   d3   a3   b3   c3    0 ..   0    0  |  |z3|  |g3|  *
c  |0    e4   d4   a4   b4   c4   .        0  |  |z4|=|g4|  *
c  |.........................................|  |..|  |  |  *
c  |                                         |  |  |  |  |  *
c  |bn   cn   0    0  .....  0   en  dn   an |  |zn|  |gn|  *
c  +                                          +  +  +  +  +  *
c   с пятидиагональной матрицей методом прогонки            *
c   n - число уравнений                                     *
c   a,b,c,d,e - массивы (длины n), содержащие элементы      *
c               диагоналей матрицы                          *
c   g - массив (длины n), содержащий элементы правых частей *
c       уравнений                                           *
c   u,v,w,p,q,                                              *
c   r,s,t,aa,dd - рабочие массивы (длины n+2)               *
c                                                           *
c       Результат:                                          *
c   z - массив (длины n), содержащий решение системы        *
c                                                           *
c*************************************************************
c
      real a(1),b(1),c(1),d(1),e(1),g(1),u(1),aa(1),r(1)
      real v(1),w(1),p(1),q(1),t(1),s(1),z(1),dd(1)
c
            v(1) = 0.
            v(2) = 0.
            w(1) = 0.
            w(2) = 0.
            u(1) = 0.
            u(2) = 0.
c
       do i = 1,n
     dd(i)   =  d(i) + e(i)*v(i)
     aa(i)   =  a(i) + dd(i)*v(i+1) +e(i)*w(i)
     u(i+2) = (g(i) - dd(i)*u(i+1) - e(i)*u(i))/aa(i)
     v(i+2) = -(b(i)+dd(i)*w(i+1))/aa(i)
     w(i+2) = -c(i)/aa(i)
   enddo
c
         p(1) = 0.
         q(2) = 0.
         p(2) = 1.
         q(1) = 1.
c
         do i = 1,n
     p(i+2) = -(dd(i)*p(i+1) + e(i)*p(i))/aa(i)
     q(i+2) = -(dd(i)*q(i+1) + e(i)*q(i))/aa(i)
   enddo
c
            t(n-1) = 0.
            s(n-1) = 0.
            t(n)   = 0.
```

```
                r(n)    = 0.
             r(n-1) = 1.
             s(n)   = 1.
c
      do i = n-2,1,-1
         t(i) = v(i+2)*t(i+1) +w(i+2)*t(i+2) + u(i+2)
         s(i) = v(i+2)*s(i+1) +w(i+2)*s(i+2) + p(i+2)
         r(i) = v(i+2)*r(i+1) +w(i+2)*r(i+2) + q(i+2)
      enddo
c
      a11 =  1. - q(n+1) - w(n+1)*r(1)
      a12 = -(p(n+1) + v(n+1) + w(n+1)*s(1))
      a21 = -(v(n+2)*r(1) + w(n+2)*r(2) + q(n+2))
      a22 =  1. - p(n+2) - v(n+2)*s(1) - w(n+2)*s(2)
      b1  = w(n+1)*t(1) + u(n+1)
      b2  = v(n+2)*t(1) + w(n+2)*t(2) + u(n+2)
      z(n-1) = ( b1*a22 - b2*a12)/(a11*a22-a12*a21)
      z(n)   = (-b1*a21 + b2*a11)/(a11*a22-a12*a21)
c
      do i = 1,n-2
         z(i) = t(i) + s(i)*z(n) + r(i)*z(n-1)
      enddo
c
      return
      end
```

Приложение Б.
Библиотеки текстов на языке С

```
/*****************************************************************

COMMON.H

*****************************************************************/
typedef double  Flt ;
typedef Flt     Vec[3] ;
typedef Flt     Point[3] ;

/* Макросы для операций с векторами */

/* длина вектора */
#define VecLen( V )     (sqrt( sqr(V[0]) + sqr(V[1]) + sqr(V[2]) ))

/* скалярное произведение  */
#define VecDot( V1, V2 ) (V1[0]*V2[0] + V1[1]*V2[1] + V1[2]*V2[2])

/* C = t * A + B   */
#define VecAddS(t,A,B,C) { C[0] = A[0]*t+B[0]; C[1] = A[1]*t+B[1];
C[2] = A[2]*t+B[2]; }

/* C = t * A */
#define VecMul( A, t, C ) { C[0] = A[0] * t; C[1] = A[1] * t; C[2]
= A[2] * t; }

/* нормирование вектора */
#define VecUnit( V )      { Flt l = 1.0 / VecLen( V ); VecMul( V,
l, V ); }

/* C = A - B */
#define VecSub( A, B, C ) { C[0] = A[0]-B[0]; C[1] = A[1]-B[1];
C[2] = A[2]-B[2]; }

/* C = A + B */
#define VecAdd( A, B, C ) { C[0] = A[0]+B[0]; C[1] = A[1]+B[1];
C[2] = A[2]+B[2]; }

/* B = A */
#define VecCopy( A, B )         { B[0]=A[0]; B[1]=A[1]; B[2]=A[2]; }

/* C = a * A + b * B */
#define VecComb(a,A,b,B,C) { C[0] = A[0]*a + B[0]*b; C[1] = A[1]*a
+ B[1]*b; C[2] = A[2]*a + B[2]*b; }

/* B = A / t */
#define VecDiv( A, t, B ) { B[0] = A[0] / t; B[1] = A[1] / t; B[2]
= A[2] / t; }

/* макросы для доступа к элементам двумерных массивов
**  p - массив
**  i, j - индексы
**  n - вторая размерность массива
*/
#define pp( p, i, j, n )  (((Point *)p)[(i)*(n)+(j)])
```

```
/*****************************************************************
LIB.H

*****************************************************************/

#include <math.h>
#include "common.h"

void DrawPolygon( Flt x[], Flt y[], int n, int color ) ;
void Init3D( Point eyepos, Point lookat, Vec up, Flt alpha ) ;
void PlotSurface( Point **p, int m, int n, int color ) ;
void SetVideoMode( void ) ;

/*****************************************************************
LIB.C

*****************************************************************/

/*
** 1. Инициализировать SetVideoMode() перед первым обращением
**    к DrawPolygon()
** 2. Инициализировать SetVideoMode() и Init3D() перед первым
**    обращением к  PlotSurface()
*/

#ifndef __BORLANDC__
#include <graph.h>
#else
#include <alloc.h>
#include <graphics.h>
#include <stdlib.h>
#endif

#include "lib.h"

#define XMIN    0.0
#define XMAX    1.0
#define YMIN    0.0
#define YMAX    1.0

/* Для моды 640x480 */
#define SXMIN   0
#define SXMAX   639
#define SYMIN   0
#define SYMAX   479

Flt dx = XMAX - XMIN, dy = YMAX - YMIN ;
int sdx = SXMAX - SXMIN, sdy = SYMAX - SYMIN ;

static Point   eyepos ;     /* точка наблюдения  */
static Point   F ;          /* левый нижний угол экрана */
static Vec     up ;
static Vec     T ;
static Vec     n ;
static Flt     dd ;
static Flt     D ;

Flt sqr( Flt t ) { return t*t ; }

/* Векторное произведение:  C = A x B */
void CrossProduct( Vec A, Vec B, Vec C )
```

```
{
    C[0] = A[1]*B[2] - A[2]*B[1] ;
    C[1] = A[2]*B[0] - A[0]*B[2] ;
    C[2] = A[0]*B[1] - A[1]*B[0] ;
}

/* пресбразование к экранным координатам  */
int MakeX( Flt x )
{
    return ( SXMIN + (sdx * (x - XMIN))/dx ) ;
}

int MakeY( Flt y )
{
    return SYMAX - ( SYMIN + (sdy * (y - YMIN))/dy ) ;
}

/*
** Инициализация всех трехмерных переменных и функций
*/
void Init3D( Point ep, Point lookat, Vec Up, Flt alpha )
{
    Vec AB ;
    Point D1 ;

    VecCopy( ep, eyepos ) ;
    VecCopy( Up, up ) ;

    VecSub( lookat, ep, AB ) ;
    dd = VecLen( AB ) * tan( alpha ) ;
    VecAddS( -dd, up, lookat, D1 ) ;
    CrossProduct( AB, up, T ) ;
    VecUnit( T ) ;
    VecAddS( -dd, T, D1, F ) ;
    CrossProduct( T, up, n ) ;
    D = -VecDot( n, lookat ) ;
}

/* Проекция точки p на экран; */
/*  (x, y) - экранные координаты точки p  */
void Project( Point p, Flt *x, Flt *y )
{
    Vec PA, FP ;
    Flt t1 ;
    VecSub( p, eyepos, PA ) ;
    t1 = - ( VecDot( n, eyepos ) + D ) / VecDot( n, PA ) ;
    VecAddS( t1, PA, eyepos, FP ) ;
    VecSub( FP, F, FP ) ;
    *x = .5 * VecDot( FP, T ) / dd ;
    *y = .5 * VecDot( FP, up ) / dd ;
}

/*
** Рисовать поверхность, определенную массивом p (размерности n на
m)
** цветом "color"
*/
void PlotSurface( Point **p, int m, int n, int color )
{
    Flt *x = (Flt *)malloc( sizeof( Flt ) * m ) ;
    Flt *y = (Flt *)malloc( sizeof( Flt ) * m ) ;
```

```
    int i, j ;
    for( j = n-1; j >= 0; j -- )
    {
 for( i = 0; i < m; i ++ )
            Project( pp( p, i, j, n), &x[ i ], &y[ i ] ) ;
        DrawPolygon( x, y, m, color ) ;
    }
    free( x ) ;
    free( y ) ;
}

/*
** Рисовать ломаную, определенную массивами x[] и y[]
(размерности n)
** цветом "color"
*/
void DrawPolygon( Flt x[], Flt y[], int n, int color )
{
    int i ;

#ifndef __BORLANDC__
    _setcolor( color ) ;
    _moveto( MakeX( x[0] ), MakeY( y[0] ) ) ;
    for( i = 1; i < n; i ++ )
 _lineto( MakeX( x[i] ),MakeY( y[i] ) ) ;
#else
    setcolor( color ) ;
    moveto( MakeX( x[0] ), MakeY( y[0] ) ) ;
    for( i = 1; i < n; i ++ )
 lineto( MakeX( x[i] ),MakeY( y[i] ) ) ;
#endif
}

/*
** Инициализация графической моды: 640 x 480 , 16 цветов
*/
void SetVideoMode( void )
{
#ifndef __BORLANDC__
    _setvideomode( _VRES16COLOR ) ;
#else
    int grDriver, grMode;

    grDriver = VGA;
    grMode   = VGAHI;
    initgraph( &grDriver, &grMode, "" );

    if( graphresult() != grOk )
 exit( 1 );
#endif
}
```

Приложение В.
Описание дискеты

Описание дискеты "Руководство для пользователей по сплайнам"

Программы на Фортране и С:
для построения интерполяционного кубического сплайна одной переменной,
построения сглаживающего кубического сплайна одной переменной,
построения интерполяционного бикубического сплайна двух переменных,
построения сглаживающего бикубического сплайна двух переменных,
построения кубических кривых Безье,
построения B-сплайновых кривых,
построения Бета-сплайновых кривых,
построения кубических кривых Эрмита,
построения Catmull-Rom кривых,
построения бикубических поверхностей Безье,
построения бикубических B-сплайновых поверхностей,
построения Бета-сплайновых поверхностей,
построения бикубических эрмитовых поверхностей.

Для каждой программы приведены один или несколько примеров ее использования.

Для использования приведенных здесь программ необходимы:

- IBM PC XT/AT или совместимый с ним компьютер,
- цветной или монохромный EGA/VGA монитор.

Программы могут размещаться на дискете 5.25". Дополнительная память или сопроцессор не требуются.

Список файлов на дискете

\SPLINES\readme.txt

APPENDIX\progon3.for
progon5.for

\FUNCTION\ONEVARIA\INTERPOL\spline.for

exmpl1.for
exmpl2.for
exmpl3.for
exmpl4.for
res1.dat
res2.dat
res3.dat
res4.dat
\SMOOTH\smspline.for
exmpl1.for
exmpl2.for
exmpl3.for
res1.dat
res2.dat
res3.dat
input1.dat
input2.dat
input3.dat
\TWOVARIA\INTERPOL\splint2.for
exmpl1.for
exmpl2.for
exmpl3.for
exmpl4.for
res1.dat
res2.dat
res3.dat
res4.dat
\SMOOTH\smspl2.for
exmpl1.for
exmpl2.for
exmpl3.for
res1.dat
res2.dat
res3.dat

\SPLINES\GEOMETRY\CURVES\BETA\beta.c
test.c
test.doc
\BEZIER\bezier.c
test.c
test.doc
\BSPLINE\bspline.c

test.c
test.doc
\CATMULL\catmull.c
test.c
test.doc
\HERMITE\hermite.c
test.c
test.doc
\NURBS\nurbs.c
test.c
test.doc
\global.doc

\FINDC1\findc1.c

\LIB\comon.h
\lib.c
\lib.h

\SURFACES\BETA\beta.c
test.c
test.doc
\BEZIER\bezier.c
test.c
test.doc
\HERMITE\hermite.c
test.c
test.doc
\NURBS\nurbs.c
test.c
test.doc
\global.doc

Описание файлов

1. Файлы lib.h, lib.c, common.h, test.c, test.doc

Файл lib.h содержит описание вспомогательных функций:

DrawPolygon() - отображение многоугольника на экране,

Init3D() - инициализация параметров и массивов, требуемых для трехмерной графики,

PlotSurface() - отображение на экране поверхности по заданному двумерному массиву,

SetVideoMode() - инициализация графической моды с разрешением 640 на 480 и с 16 цветами.

Файл lib.c содержит описание внутренних процедур для всех вспомогательных функций из файла lib.h. С компиляторами Microsoft C или BORLAND C этот файл можно использовать не внося в него никаких изменений. При использование других компиляторов возможно потребуется модификация этого файла.

В файле common.h определены все необходимые типы данных, макросы для для работы с векторами и двумерными массивами.

Каждый файл test.c предназначен для запуска примера, демонстрирующего построение сплайна того или иного типа (см. ниже разд. 2).

В файлах test.doc описаны правила обращения к программам построения сплайнов и приведены соответствующие примеры.

2. Файлы с программами построения сплайнов. Директория FUNCTION

Все файлы вида *.for содержат исходные тексты программ на Фортране и могут компилироваться как в системе Microsoft Fortran 5.1, так и в Microsoft Fortran Power Station.

Файл spline.for (из поддиректории ONEVARIA\INTERPOL) содержит исходный текст программы Spline для построения интерполяционного кубического сплайна.

Файл exmpl1.for содержит головную программу, иллюстрирующую применение программы Spline для построения интерполяционного кубического сплайна, удовлетворяющего граничным условиям 1-го типа. Результаты вычислений находятся в файле res1.dat.

Файл exmpl2.for содержит головную программу, иллюстрирующую применение программы Spline для построения интерполяционного кубического сплайна, удовлетворяющего граничным условиям 2-го типа. Результаты вычислений находятся в файле res2.dat.

Файл exmpl3.for содержит головную программу, иллюстрирующую применение программы Spline для построения интерполяционного кубического сплайна, удовлетворяющего граничным условиям 3-го типа. Результаты вычислений находятся в файле res3.dat.

Файл exmpl4.for содержит головную программу, иллюстрирующую применение программы Spline для построения интерполяционного кубического сплайна, удовлетворяющего граничным условиям 4-го типа. Результаты вычислений находятся в файле res4.dat.

Файл smspline.for (из поддиректории ONEVARIA\SMOOTH) содержит исходный текст программы Smspline для построения сглаживающего кубического сплайна.

Файл exmpl1.for содержит головную программу, иллюстрирующую применение программы Smspline для построения сглаживающего кубического сплайна, удовлетворяющего граничным условиям 1-го типа. Исходные данные для этого примера размещены в файле input1.dat. Результаты вычислений находятся в файле res1.dat.

Файл exmpl2.for содержит головную программу, иллюстрирующую применение программы Smspline для построения сглаживающего кубического сплайна, удовлетворяющего граничным условиям 2-го типа. Исходные данные для этого примера размещены в файле input2.dat. Результаты вычислений находятся в файле res2.dat.

Файл exmpl3.for содержит головную программу, иллюстрирующую применение программы Smspline для построения сглаживающего кубического сплайна, удовлетворяющего граничным условиям 3-го типа. Исходные данные для этого примера размещены в файле input3.dat. Результаты вычислений находятся в файле res3.dat.

Файл splint2.for (из поддиректории TWOVARIA\INTERPOL) содержит исходный текст программы Splint2 для построения интерполяционного бикубического сплайна.

Файл exmpl1.for содержит головную программу, иллюстрирующую применение программы Splint2 для построения интерполяционного бикубического сплайна, удовлетворяющего граничным условиям 1-го типа. Результаты вычислений находятся в файле res1.dat.

Файл exmpl2.for содержит головную программу, иллюстрирующую применение программы Splint2 для построения интерполяционного бикубического сплайна, удовлетворяющего граничным условиям 2-го типа. Результаты вычислений находятся в файле res2.dat.

Файл exmpl3.for содержит головную программу, иллюстрирующую применение программы Splint2 для построения интерполяционного бикубического сплайна, удовлетворяющего граничным условиям 3-го типа. Результаты вычислений находятся в файле res3.dat.

Файл exmpl4.for содержит головную программу, иллюстрирующую применение программы Splint2 для построения интерполяционного бикубического сплайна, удовлетворяющего граничным условиям 4-го типа. Результаты вычислений находятся в файле res4.dat.

Файл smspl2.for (из поддиректории TWOVARIA\SMOOTH) содержит исходный текст программы Smspl2 для построения сглаживающего бикубического сплайна.

Файл exmpl1.for содержит головную программу, иллюстрирующую применение программы Smspl2 для построения сглаживающего бикубического сплайна, удовлетворяющего граничным условиям 1-го типа. Результаты вычислений находятся в файле res1.dat.

Файл exmpl2.for содержит головную программу, иллюстрирующую применение программы Smspl2 для построения сглаживающего бикубического сплайна, удовлетворяющего граничным условиям 2-го типа. Результаты вычислений находятся в файле res2.dat.

Файл exmpl3.for содержит головную программу, иллюстрирующую применение программы Smspl2 для построения сглаживающего бикубического сплайна, удовлетворяющего граничным условиям 3-го типа. Результаты вычислений находятся в файле res3.dat.

3. Файлы с программами построения сплайнов. Директория GEOMETRY

Файл beta.c (из поддиректории CURVES\BETA) содержит программу на языке C для построения Бета-сплайновой кривой по заданному двумерному массиву.

Файл beta.c (из поддиректории SURFACE\BETA) содержит программу на языке C для построения Бета-сплайновой поверхности по заданному трехмерному массиву 3D data set.

Файл bezier.c (из поддиректории CURVES\BEZIER) содержит программу на языке C для построения кубической кривой Безье по заданному двумерному массиву.

Файл bezier.c (из поддиректории SURFACES\BEZIER) содержит программу на языке C для построения бикубической поверхности Безье по заданному трехмерному массиву.

Файл bspline.c (из поддиректории CURVES\BSPLINE) содержит программу на языке C для построения кубической B-сплайновой кривой по заданному двумерному массиву.

Файл bspline.c (из поддиректории SURFACES\BSPLINE) содержит программу на языке C для построения бикубической B-сплайновой поверхности по заданному трехмерному массиву.

Файл catmull.c (из поддиректории CURVES\CATMULL) содержит программу на языке C для построения Catmull-Rom кривой по заданному двумерному массиву.

Файл hermite.c (из поддиректории CURVES\HERMITE) содержит программу на языке С для построения кубической эрмитовой кривой по заданному двумерному массиву.

Файл hermite.c (из поддиректории SURFACES\HERMITE) содержит программу на языке С для построения бикубической эрмитовой поверхности по заданному трехмерному массиву.

Файл nurbs.c (из поддиректории CURVES\NURBS) содержит программу на языке С для построения неоднородной рациональной кубической В-сплайновой кривой по заданному двумерному массиву.

Файл nurbs.c (из поддиректории SURFACES\NURBS) содержит программу на языке С для построения неоднородной рациональной бикубической В-сплайновой поверхности по заданному трехмерному массиву.

4. Другие файлы

Файл global.doc (из поддиректорий CURVES и SURFACES) содержит общую информацию о представленных сплайнах.

Файл findc1.doc (из поддиректории FINDC1}) содержит программу на языке С для построения *C1*-гладкой составной трехмерной кубической кривой Безье.

Файл progon3.for (из поддиректории APPENDIX) содержит программу на Фортране для решения системы линейных уравнений с трехдиагональной матрицей методом прогонки.

Файл progon5.for (из поддиректории APPENDIX) содержит программу на Фортране для решения системы линейных уравнений с пятидиагональной матрицей методом прогонки.

Файл readme.txt содержит описание приложенной дискеты.

ЛИТЕРАТУРА

Классические работы

Weierstrass K., Sitzungsber. Acad. Berlin 1885 S.633—639, 789—805.

Runge C. Ueber empirishe Funktionen und die Interpolation zwischen aequidistanten Ordinaten. ZAMM. 46:224 — 243, 1901.

Bernstein S. Démonstration du théorème de Weierstrass fondeé sur le calcul des probabilités. Harkov Soobs. Matem. ob-va, 13:1 — 2, 1912.

Теоретические работы по сплайнам

Альберг Дж., Нилсон Э., Уолш Дж. Теория сплайнов и их приложения. М.: Наука, 1972 (Ahlberg J., Nilson E., Walsh J. The theory of splines and their applications. Academic Press. 1967).

Стечкин С. Б., Субботин Ю. Н. Сплайны в вычислительной математике. М.: Наука, 1976.

Завьялов Ю. С., Квасов Б. И., Мирошниченко В. Л. Методы сплайн-функций. М.: Наука, 1980.

Сплайны в компьтерной графике

Barsky B. Computer Graphics and Geometric Modeling using Beta-splines. Springer Verlag. 1988.

Bartles R., Beatty J., Barsky B. An introduction to splines for use in computer graphics and geometric modeling. Morgan Kaufmann. 1987.

Farin G. Curves and surfaces for computer aided geometric design. A practical guide. Academic Press. 1990.

Foley J. D., van Dam A., Feiner S. K., Hugues J. F. Computer graphics. Principles and practice. Addison-Wesley Pub. Com. 1991.

Сплайны и проектирование

Фокс Ф., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве. М.: Мир, 1982 (Faux I. D., Pratt M. J. Computational geometry for design and manufacture. Ellis Horwood. 1979).

Mathématique et CAO (in four volumes). Hermes Publishing 1986, 1987.

Завьялов Ю. С., Леус В. А., Скороспелов Б. А. Сплайны в инженерной геометрии. М.: Машиностроение, 1985.

Yamaguchi F. Curves and surfaces in computer aided geometric design. Springer Verlag. 1988.

Практические руководства

Де Бор К. Практическое руководство по сплайнам. М.: Наука, 1983 (de Boor C. A practical guide to splines. Springer, 1978).

Недавние публикации по неявным кубическим сплайнам

Sederberg T. W. Techniques for cubic algebraic surfaces. IEEE Computer Graphics and Applications, 4 (1990), 14—25

Sederberg T. W. Techniques for cubic algebraic surfaces. IEEE Computer Graphics and Applications, 5 (1990), 12—21

Bajaj C. L. Surface fitting using implicit algebraic surface partches. In Topics in Surface Modeling. H. Hagen, Ed., SIAM. 1992.

Paluszny M., Patterson R. R. A family of tangent continuous cubic algebraic splines. Transactions on Graphics, volume 12, N 3 (1993), 209—232

Dahmen W., Thamm-Schaar T.-M. Cubicoids: modeling and visualization, Computer Aided Geometric Design, 10 (1993), 89—108

Guo B., Nonsplitting macro patches for implicit cubic spline surfaces, EUROGRAPHCS'93, volume 12, N 3 (1993), C-433 — C-445

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 3
О структуре пособия 4
Несколько общих советов пользователю 4
Почему сплайны? 7

ЧАСТЬ I. СПЛАЙН-ФУНКЦИИ 11

ГЛАВА 1. СПЛАЙН-ФУНКЦИИ ОДНОЙ ПЕРЕМЕННОЙ. 12
1.1 Интерполяционные кубические сплайны 13
1.1.1. Постановка задачи интерполяции 13
1.1.2. Определение интерполяционного кубического сплайна. 14
1.1.3. Граничные (краевые) условия 15
1.1.4. Построение интерполяционного кубического сплайна .. 16
1.1.5. Советы пользователю 18
А. Выбор граничных (краевых) условий 18
Б. Выбор узлов интерполяции. 19
1.1.6. Выбор интерполяционной функции (плюсы и минусы) .. 20
А. Интерполяционный многочлен Лагранжа. 20
Б. Кусочно-линейная интерполяция. 22
В. Сплайн-интерполяция. 23
1.1.7. Свойства интерполяционного кубического сплайна 25
А. Аппроксимационные свойства кубического сплайна. 25
Б. Экстремальное свойство кубического сплайна 25
В. Построение интерполяционных сплайновых кривых при помощи сплайн-функций. 26
1.1.8. Программная реализация 28
1.2. Сглаживающие кубические сплайны 36
1.2.1. О постановке задачи сглаживания 36
1.2.2. Определение сглаживающего кубического сплайна 37
1.2.3. Граничные (краевые) условия 37
1.2.4. Построение сглаживающего кубического сплайна 38
1.2.5. Выбор весовых коэффициентов 41
1.2.6. Построение сглаживающих сплайновых кривых при помощи сплайн-функций 42
1.2.7. Программная реализация 43
1.3. Другие сплайны 50
1.3.1. Линейное пространство кубических сплайн-функций 50
1.3.2. Кубические *B*-сплайны 50

ГЛАВА 2. СПЛАЙН-ФУНКЦИИ ДВУХ ПЕРЕМЕННЫХ 53
2.1. Интерполяционные бикубические сплайны 56
2.1.1. Постановка задачи интерполяции 56
2.1.2. Определение интерполяционного бикубического сплайна 56
2.1.3. Граничные (краевые) условия 57
2.1.4. Построение интерполяционного бикубического сплайна59
2.1.5. Свойства интерполяционного бикубического сплайна ..62
А. Аппроксимационное свойство. 62
Б. Экстремальное свойство. 63
2.1.6. Построение сплайновых поверхностей при помощи сплайн-функций. 65
2.1.7. Программная реализация 66
2.2. Сглаживающие бикубические сплайны 75
2.2.1. О постановке задачи сглаживания 75
2.2.2. Определение сглаживающего бикубического сплайна ..75
2.2.3. Граничные (краевые) условия 76
2.2.4. Построение сглаживающего бикубического сплайна 78
2.2.5. Построение сплайновых поверхностей при помощи сплайн-функций 79
2.2.6. Программная реализация 81

ЧАСТЬ II. ГЕОМЕТРИЧЕСКИЕ СПЛАЙНЫ 89

ГЛАВА 3. СПЛАЙНОВЫЕ КРИВЫЕ 90
3.1. Элементарные сведения из дифференциальной геометрии кривых 93
3.1.1. Параметризованные кривые 93
3.1.2. Гладкие и регулярные кривые 93
3.1.3. Замена параметра 94
3.1.4. Трехгранник Френе 95
3.1.5. Кривизна и кручение кривой 96
3.1.6. Плоские кривые 97
А. Параметрическое задание 97
Б. Неявное задание. 98
3.1.7. Составные кривые 98
3.1.8. Геометрическая непрерывность 102
3.2. Кривые Безье 104
3.2.1. Параметрические уравнения кривой Безье 104
3.2.2. Свойства кривых Безье 105
3.2.3. Составные кривые Безье 107
3.2.4. Рациональные кривые Безье 110
3.2.5. Программная реализация алгоритма 113

3.3. *B*-сплайновые кривые 115
3.3.1. Параметрические уравнения элементарной кубической *B*-сплайновой кривой 115
3.3.2. Составные кубические *B*-сплайновые кривые..... 116
3.3.3. Кратные и воображаемые вершины..... 119
А. Двойные вершины.....120
Б. Тройные вершины. 120
В. Воображаемые вершины.....121
3.3.4. Рациональные кубические *B*-сплайновые кривые 123
3.3.5. Форма Безье составных кубических *B*-сплайновых кривых 125
3.3.6. Программная реализация алгоритма..... 126
3.4. Бета-сплайновые кривые 130
3.4.1. Параметрические уравнения элементарной Бета-сплайновой кривой..... 130
3.4.2. Составные Бета-сплайновые кривые..... 132
3.4.3. Кратные и воображаемые вершины..... 135
А. Двойные вершины.....135
Б. Тройные вершины. 136
В. Воображаемые вершины.....136
3.4.4. Программная реализация алгоритма..... 137
3.5. Другие сплайновые кривые 140
3.5.1. Интерполяционные кубические кривые Эрмита 140
Программная реализация.....144
3.5.2. Сплайновые кривые Catmull-Rom 146
Программная реализация.....148
3.5.3. Составные плоские кубические кривые, заданные в неявной форме 150

ГЛАВА 4 СПЛАЙНОВЫЕ ПОВЕРХНОСТИ 152
4.1. Элементарные сведения из геометрии поверхностей..... 156
4.1.1. Параметризованные поверхности 156
4.1.2. Гладкие и регулярные поверхности..... 157
4.1.3. Первая квадратичная форма поверхности 157
4.1.4. Кривая на поверхности..... 158
4.1.5. Угол между кривыми на поверхности 158
4.1.6. Площадь поверхности..... 159
4.1.7. Вторая квадратичная форма поверхности..... 159
4.1.8. Линии кривизны..... 159
4.1.9. Гауссова и средняя кривизны..... 160
4.1.10. Геометрическая непрерывность..... 160
4.1.11. Вектор скручивания и билинейная поверхность..... 163

4.2. Поверхности Безье 165
4.2.1. Параметрические уравнения поверхности Безье 165
4.2.2. Свойства элементарных поверхностей Безье 166
4.2.3. Составные поверхности Безье 169
4.2.4. Рациональные поверхности Безье 170
4.2.5. Программная реализация 173
4.3. *B*-сплайновые поверхности 176
4.3.1. Параметрические уравнения элементарной бикубической *B*-сплайновой поверхности 176
4.3.2. Свойства элементарных бикубических *B*-сплайновых поверхностей 177
4.3.3. Составные бикубические *B*-сплайновые поверхности .. 178
4.3.4. Кратные и воображаемые вершины 181
А. Двойные вершины 181
Б. Тройные вершины 183
В. Воображаемые вершины 185
4.3.5. Рациональные бикубические *B*-сплайновые поверхности 186
4.3.6. Программная реализация 187
4.4. *B*-сплайновые поверхности 191
4.4.1. Параметрические уравнения элементарной Бета-сплайновой поверхности 191
4.4.2. Свойства элементарных Бета-сплайновых поверхностей 193
4.4.3. Составные Бета-сплайновая поверхности 194
4.4.4. Кратные и воображаемые вершины 196
А. Двойные вершины 197
Б. Тройные вершины 198
В. Воображаемые вершины 199
4.4.5. Программная реализация 201
4.5. Другие сплайновые поверхности 204
4.5.1. Интерполяционные бикубические поверхности Эрмита 204
4.5.2. Программная реализация 210
4.5.3. Составные неявно заданные кубические поверхности . 216

ПРИЛОЖЕНИЯ 217
Приложение А. Программы метода прогонки для трёх- и пятидиагональных матриц 218
Приложение Б. Библиотеки текстов на языке С 221
Приложение В. Описание дискеты 225

ЛИТЕРАТУРА 232

Из серии:
А. В. Фролов,
Г. В. Фролов

Библиотека системного программиста

Том 11. ОПЕРАЦИОННАЯ СИСТЕМА MICROSOFT WINDOWS 3.1.
Введение для программиста. Часть 1

Том 12. ОПЕРАЦИОННАЯ СИСТЕМА MICROSOFT WINDOWS 3.1.
Для программиста. Часть 2

Том 13. ОПЕРАЦИОННАЯ СИСТЕМА MICROSOFT WINDOWS 3.1.
Для программиста. Часть 3

Том 14. ГРАФИЧЕСКИЙ ИНТЕРФЕЙС GDI В MICROSOFT WINDOWS.

Том 15. МУЛЬТИМЕДИА ДЛЯ WINDOWS.
Руководство программиста

Том 16. МОДЕМЫ И ФАКС-МОДЕМЫ.
Программирование для MS-DOS и Windows

Том 17. MICROSOFT WINDOWS 3.1 Для программиста.
Дополнительные главы

Том 18. MS-DOS ДЛЯ ПРОГРАММИСТА.Часть 1

Том 19. MS-DOS ДЛЯ ПРОГРАММИСТА.Часть 2

Том 20. ОПЕРАЦИОННАЯ СИСТЕМА IBM OS/2 WARP

Том 21. ПРОГРАММИРОВАНИЕ ВИДЕОАДАПТЕРОВ EGA,VGA и SVGA

Том 22. ОПЕРАЦИОННАЯ СИСТЕМА WINDOWS 95. Для программиста
Том 23. ГЛОБАЛЬНЫЕ СЕТИ КОМПЬЮТЕРОВ. Практическое ведение в Internet, работа E-Mal и WWW

ПКшаг за шагом

Серия книг для самостоятельно постигающих тайны компьютерного мира. Авторы А. В. Фролов и Г. В. Фролов

Том 1. ВВЕДЕНИЕ В MS-DOS, MS Windows, MS Word for Windows.

Том 2. ОПЕРАЦИОННАЯ СИСТЕМА Microsoft Windows.
Руководство пользователя

Том 3. СЕТИ КОМПЬЮТЕРОВ В ВАШЕМ ОФИСЕ

Том 4. ЧТО ВЫ ДОЛЖНЫ ЗНАТЬ О СВОЕМ КОМПЬЮТЕРЕ

Том 5. ОСТОРОЖНО: КОМПЬЮТЕРНЫЕ ВИРУСЫ!

ДИАЛОГ-МИФИ

320-43-77, 320-43-55
факс: 324-30-55

планирует выпустить в 1996 году следующие книги

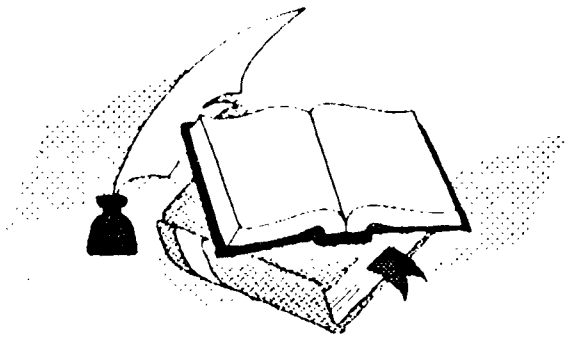
И. Ю. Баженова
Visual FoxPro 3.0

А. В. Фролов, Г. В. Фролов
БСП 24 том. **MS VISUAL C++ и MFC. Программирование для Windows 95 и Windows NT.**
БСП 25 том. **ПРОГРАММИРОВАНИЕ ДЛЯ IBM OS/2 WARP** (часть первая)

Ю. А. Кречко, В. В. Полищук
АВТОКАД 13: НОВЫЕ ВОЗМОЖНОСТИ. В 2-х частях

Сергей Дунаев
Borland-Технологии. InterBase. Paradox for MS-Windows. Delphi

Л. Б. Романова, В. Ю. Романов.
ПУТЕШЕСТВИЕ РЕБЯТ В КОМПЬЮТЕРНУЮ СКАЗКУ.



Книги издательства "Диалог - МИФИ" можно купить по адресам:

Владивосток

Салон деловой, профессиональной и образовательной информации "Оферта-Дело". 690002, Океанский пр-т, 104. 25-85-11

Кемерово

Магазин "Книжный мир", Советсткий пр-т. 43

Киев

- ТОО " В Е К " , тел. (044) 510-93-65, тел./факс 483-29-85.
  Киоск "Техническая книга", пр. Победы, 37, КПИ, уч. корп. N 7
- ТОО "Надіа - 7 ЛТД". Тел.: (044) 261-92-86, факс: 268-60-27

Краснодар

"Краснодарский дом книги", ул. Красная, 43

Минск

МНВЦ "КЕЙ". 220040, ул. М. Богдановича, 153, к.609. Тел.: 32-64-16

Москва

- "Московский дом книги", ул. Новый Арбат, дом 26.
- "Дом технической книги", Ленинский пр-т., дом 40.
- ТОО ТД "Библио-глобус", 101861, ул. Мясницкая, 6
- ТОО "КНОРУС". Тел.: 928-17-25, 928-62-69
- АОЗТ "МИДИКС". Т/ф: 955-41-01
- Книжная база из-ва "РИС". Т/ф 313-83-45, 314-31-34, 925-02-10
- ТОО "Принт". Т/ф 909-57-45
- ЗАО "РИДАС". Тел.: (095) 919-62-10, 919-62-11, факс: 919-87-03
- ТОО "ЭНТА" (рассылка по почте), ул. Каргапольская, 17. Тел.: 903-04

Новосибирск

ТОО "ГПНТБ - ПОИСК". Тел.: 66-85-67, факс: 66-33-65

Рязань

ТОО "Лексикон". Тел.: (0912) 72-22-82
ТОО "Книги - 7". Ул. Циолковского, 1/7

Самара

ТОО "Чакона", ул. Урицкого, 1. Тел.: (8462) 36-14-87, факс: 32-02-87

Санкт-Петербург

- ТОО "Диалект ". Тел.: (812) 534-45-78, факс (812) 535-56-83
- "Дом книги", 191186, Невский пр-т, 28 отдел Книга-почтой

Сургут

АОЗТ "Скейлинг - Сургут". Т/ф (3462) 77-83-51

Челябинск

ЧРЦ высшей школы, пр-т Ленина, 87, тел.: (3512) 65-49-77, 65-59-53.

Приглашаем распространителей по регионам