Новосибирский государственный технический университет

Конспект лекций по курсу "Методы конечноэлементного анализа"

для магистрантов I-II курса ФПМИ

Новосибирск
2014

Составители: д.т.н., проф. М.Г. Персова,
д.т.н., проф. Ю.Г. Соловейчик

Рецензенты: к.т.н., доцент Ю.В. Тракимус
к.т.н. М.В. Гамадин

Работа подготовлена на кафедре прикладной математики

ОГЛАВЛЕНИЕ

1. СТРУКТУРЫ ДАННЫХ И ПОСТРОЕНИЕ СЕТОК В МКЭ

1.1. ОСНОВНЫЕ МЕТОДЫ ОПИСАНИЯ РАСЧЁТНОЙ ОБЛАСТИ

Описание расчётной области в МКЭ необходимо для последующего построения конечноэлементной сетки и определения значений параметров дифференциального уравнения (т.е. его коэффициентов и правой части) в подобластях. Кроме того, это описание должно содержать информацию, необходимую для обработки краевых условий.

В настоящее время разработчики базирующихся на применении МКЭ программных средств используют в основном два способа описания геометрии расчётной области. Один из них – *метод конструктивной геометрии*. В этом подходе расчётная область представляется с помощью *базовых примитивов* (например, для двумерного случая это может быть многоугольник, окружность, эллипс и т.д.) и операций над ними (объединение, пересечение, геометрические преобразования и т.п.). Достоинства этого метода очевидны:

- малый объём исходных данных, если моделируемые объекты хорошо описываются базовыми примитивами;
- возможность создания довольно простого языка для описания геометрии области.

В основном метод конструктивной геометрии применяется в комплексах конечноэлементного моделирования, предназначенных для решения задач в областях с различными (возможно, пересекающимися) телами стандартной формы (таких, как параллелепипед, цилиндр, шар, конус и т.д.).

Вторым распространенным подходом к описанию расчётной области является *метод описания границ*. В этом подходе геометрия расчётной области задаётся путём описания всех границ её подобластей. Главным достоинством этого метода является хорошая возможность стандартизации всех процедур задания границ подобластей расчётной области и конечноэлементных сеток, что позволяет созда-

вать эффективные интерактивные графические системы для решения задач с геометрией практически любой сложности.

В этом разделе мы рассмотрим один из простейших вариантов метода конструктивной геометрии, предназначенный для описания расчётной области с прямоугольными подобластями, и дадим очень короткий обзор методов, основанных на различных способах описания геометрии расчётной области через границы её подобластей. Кроме того, будут довольно детально представлены структуры данных, предназначенные для описания расчётных областей и хранения конечноэлементных сеток.

1.2. СТРУКТУРЫ ДАННЫХ ДЛЯ ОПИСАНИЯ ДВУМЕРНЫХ ЗАДАЧ. ДВУМЕРНЫЕ СЕТКИ

1.2.1. ОПИСАНИЕ ДВУМЕРНЫХ РАСЧЁТНЫХ ОБЛАСТЕЙ С ПРЯМОУГОЛЬНЫМИ ПОДОБЛАСТЯМИ

Пусть расчётная область состоит из прямоугольных подобластей $W_1, W_2, \ldots$. Построим следующую структуру данных.

Сначала сформируем массивы X^W и Y^W значений всех вертикальных и горизонтальных границ подобластей W_l. Например, для показанной на рис. 1 расчётной области $\Omega = W_1 \cup W_2 \cup W_3$ массивы X^W и Y^W будут иметь следующий вид: $X^W = \{1., 2., 6.\}$ и $Y^W = \{1., 2., 4., 5.\}$.

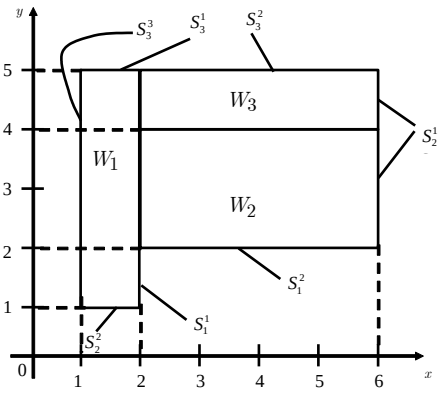


Рис. 1. Грубая конечноэлементная сетка в расчётной области Ω

Каждая из подобластей W_l представляется набором из пяти целых чисел, первое из которых означает номер подобласти, второе – номер элемента в массиве X^W, определяющего x-координату левой границы подобласти W_l, третье – номер элемента в массиве X^W, определяющего x-координату правой границы W_l, четвёртое – номер элемента в массиве Y^W, определяющего y-координату нижней границы W_l, пятое – номер элемента в массиве Y^W, определяющего y-координату верхней границы W_l.

Таким образом, полное описание расчётной области, состоящей из набора прямоугольников с различными значениями параметров дифференциального уравнения в них, может быть представлено в файле следующей структуры: первое число N_x^W (целое) означает длину массива X^W; следующие N_x^W вещественных чисел – элементы массива X^W; далее число N_y^W (целое) – длина массива Y^W; следующие N_y^W вещественных чисел – элементы массива Y^W; затем число L (целое) – количество подобластей $W_1, W_2, \ldots$, из которых состоит расчётная область Ω; далее L наборов по 5 целых чисел, описывающих все подобласти $W_1, W_2, \ldots$. При использовании такого подхода расчётная область, представленная на рис. 1, может быть описана следующим образом:

```
3
1. 2. 6.
4
1. 2. 4. 5.
3
1 1 2 1 4
2 2 3 2 3
3 2 3 3 4
```

Конечно, при описании границ отдельных подобластей вместо номеров элементов массивов X^W и Y^W можно указывать их значения. Но такое представление исходных данных может быть источником ошибок при интерпретации этих данных в ситуациях, когда границы подобластей задаются дробными числами, которые могут округляться. Действительно, если при задании вертикальной (или горизонтальной) границы двух смежных подобластей расчётной области одно и то же число (например, π) округлить по-разному (например, при описании одной подобласти задать x-координату границы равной 3.14, а при описании второй подобласти – равной 3.1415), то при интерпретации таких данных в обрабатывающей геометрию расчётной области программе эти подобласти будут иметь несовпадающую границу – в результате в расчётной области могут появиться дыры, искажающие её геометрию. Если не предусмотреть специальных мер контроля таких ситуаций, то расчёт по заданным таким образом данным может приводить к серьёзному искажению результата. Описание же границ подобластей через номера элементов массивов X^W и Y^W исключает возникновение такого рода проблем.

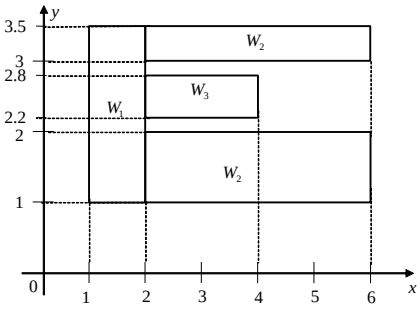


Рис. 2. Пример расчётной области, содержащей две подобласти с одинаковыми значениями параметров в них

Заметим, что номера подобластей в рассмотренном подходе к представлению расчётной области нужны фактически для того, чтобы вычислить значения параметров (коэффициентов и правой части) дифференциального уравнения – в каждой из подобластей W_l эти параметры могут быть либо различными константами, либо функциями координат или решения. Поэтому если в двух геометрически разнесенных подобластях расчётной области параметры задачи одинаковы (в том смысле, что в каждой из этих подобластей они либо равны одним и тем же константам, либо их значения могут быть вычислены по одним и тем же формулам), то у этих подобластей удобно задать один и тот же номер. Но при этом каждая из этих подобластей должна быть включена в описание расчётной области. Например, расчётная область, показанная на рис. 2, будет описана следующим образом:

```
4
1.  2.  4.  6.
6
1.  2.  2.2  2.8  3.  3.5
4
1  1  2  1  6
2  2  4  1  2
3  2  3  3  4
2  2  4  5  6
```

Как видно из приведённого выше описания расчётной области Ω, две из её четырёх подобластей имеют один и тот же номер, равный двум. Очевидно, что при таком подходе под номером подобласти фактически понимается *номер формул*, определяющих параметры дифференциального уравнения в рассматриваемой подобласти.

Аналогичный подход может быть применён для описания границ расчётной области с заданными на них краевыми условиями первого, второго или третьего рода. Каждый фрагмент границ S_1, S_2 и S_3 может быть описан шестью целыми числами:

- тип краевого условия (1 означает принадлежность фрагмента границе S_1, 2 – границе S_2, 3 – границе S_3);
- номер границы (т.е. номер формул, задающих параметры краевого условия на рассматриваемом фрагменте границы);
- номер элемента массива X^W, с которого рассматриваемый фрагмент границы начинается по оси x;
- номер элемента массива X^W, которым рассматриваемый фрагмент границы заканчивается по оси x;
- номер элемента массива Y^W, с которого рассматриваемый фрагмент границы начинается по оси y;
- номер элемента массива Y^W, которым рассматриваемый фрагмент границы заканчивается по оси y.

При использовании такого подхода границы расчётной области, представленной на рис. 1, могут быть описаны следующим образом (содержимое массивов X^W и Y^W – то же самое, что и в описании самой расчётной области на с. 6):

```
1 1 2 2 1 2
1 2 2 3 2 2
3 1 1 2 4 4
3 2 2 3 4 4
3 3 1 1 1 4
2 1 3 3 2 3
2 1 3 3 3 4
2 2 1 2 1 1
```

Очевидно, что если параметры краевых условий являются константами, то вместо номеров границ можно задавать значения этих параметров. Однако если в решаемой задаче есть параметры краевых условий в виде некоторых функций, то задать их по номеру границы гораздо проще. Возможно и комбинирование двух подходов. Например, если номер границы положителен, то он задаёт номер формулы, по которой должны вычисляться параметры краевого условия, если же отрицателен, то значения параметров краевого условия задаются в следующих за этим номером вещественных числах (т.е. одним числом для краевых условий первого и второго рода и двумя числами для краевых условий третьего рода). Аналогичные модификации возможны и при описании расчётной области – после номе-

ра подобласти можно зарезервировать место под необходимое количество вещественных чисел, которые в случае отрицательного номера подобласти интерпретируются как значения параметров.

1.2.2. ОБЩИЕ ПРИНЦИПЫ ПОСТРОЕНИЯ СТРУКТУР ДАННЫХ ДЛЯ ХРАНЕНИЯ КОНЕЧНОЭЛЕМЕНТНЫХ СЕТОК

Как было показано выше, наиболее эффективным способом формирования конечноэлементной СЛАУ является её поэлементная сборка из локальных матриц. Поэтому хранение конечноэлементной сетки должно быть таким, чтобы можно было достаточно просто организовать цикл по элементам. При обработке каждого элемента нужен простой доступ к координатам его вершин и к информации о значениях параметров дифференциального уравнения на элементе (для вычисления локальных матриц и векторов), а также нужны глобальные номера локальных базисных функций этого элемента (для занесения локальных матрицы и вектора в глобальные). Могут ещё также понадобиться глобальные номера ребер и граней конечных элементов, а также некоторая дополнительная информация об особых узлах при наличии в сетке несогласованных элементов.

Вся необходимая информация о конечноэлементной сетке хранится в специальных структурах данных. При этом организация структуры данных очень сильно зависит от типа сетки. Наиболее простыми являются структуры данных для регулярных конечноэлементных сеток.

1.2.3. ХРАНЕНИЕ РЕГУЛЯРНЫХ ПРЯМОУГОЛЬНЫХ СЕТОК

При использовании двумерных регулярных прямоугольных сеток можно довольно просто определять положения ячейки и координаты узлов *по её номеру*. Для этого дополнительно к описанию области достаточно хранить только координатные линии $x = x_p$ и $y = y_s$, определяющие прямоугольную сетку. При этом каждый конечный элемент фактически определяется двумя соседними координатными линиями по x ($x = x_p$ и $x = x_{p+1}$) и по y ($y = y_s$ и $y = y_{s+1}$). Значения же x_p и y_s, задающие координатные линии сетки, удобно хранить в двух одномерных массивах X и Y.

Эти массивы обычно формируются из массивов X^W и Y^W, с помощью которых определяются границы элементарных подобластей и которые задаются в файле описания расчётной области (см. с. 5–6). Для этого достаточно каким-либо образом определить разбиение интервалов между соседними точками в массивах

X^W и Y^W. Это можно сделать, например, задав количество подынтервалов, на которые должен быть разбит каждый интервал между соседними точками в массивах X^W и Y^W. Чтобы иметь возможность делать сгущения или разрежения узлов, на каждом интервале между соседними точками массивов X^W и Y^W можно также задавать коэффициент растяжения (сжатия) подынтервалов.

Информацию о разбиениях на подынтервалы всех интервалов между соседними точками в массивах X^W и Y^W можно задавать в файле со следующей структурой.

Пусть N_x^W – длина массива X^W, а N_y^W – длина массива Y^W. Первые $N_x^W - 1$ пар чисел задают разбиение интервалов между соседними точками в массиве X^W. При этом первое в каждой паре число (целое) – это количество подынтервалов, на которое должен быть разбит очередной интервал. Второе же число в этой паре (вещественное) – коэффициент изменения длины подынтервалов при разбиении каждого интервала.

Следующие хранящиеся в этом файле $N_y^W - 1$ пар чисел точно так же задают разбиение интервалов между соседними точками в массиве Y^W.

Приведём пример файла, определяющего разбиение показанной на рис. 2 расчётной области. Учитывая, что $N_x^W = 4$, а $N_y^W = 6$ (см. описание расчётной области на с. 7), нужно задать три пары чисел для определения сетки по x и пять пар чисел для определения сетки по y:

```
5 1.     4 1.25     4 1.
5 1.     1 1.       3 1.     1 1.     2 1.
```

Тогда массивы X и Y будут иметь вид

$$X = \{1,\ 1.2,\ 1.4,\ 1.6,\ 1.8,\ 2,\ 2.35,\ 2.78,\ 3.32\ 4,\ 4.5,\ 5,\ 5.5,\ 6\},$$

$$Y = \{1,\ 1.2,\ 1.4,\ 1.6,\ 1.8,\ 2,\ 2.2,\ 2.4,\ 2.6,\ 2.8,\ 3,\ 3.25,\ 3.5\}.$$

Обратим внимание на то, что в рассматриваемом нами примере часть узлов (x_p, y_s) и ячеек $\Omega_k = [x_p, x_{p+1}] \times [y_s, y_{s+1}]$ регулярной прямоугольной сетки не попадает в расчётную область. Тем не менее чтобы работать с этой сеткой как с *регулярной,* мы должны пронумеровать *все ячейки* и *все узлы*. Так, в рассмотренном нами примере в расчётную область не попали, например, узлы $(2.2, 4.5)$, $(2.2, 5)$ и т.д. (всего 16 узлов из 182), а также 28 ячеек из 156.

Узлы регулярной сетки, не попавшие в расчётную область, обычно называют **фиктивными.**

Поскольку не попавшие в расчётную область ячейки не являются конечными элементами, от них нет вкладов в глобальную матрицу. Поэтому для фиктивных узлов в конечноэлементной СЛАУ генерируются *полностью нулевые строки*, и, чтобы избежать вырожденности СЛАУ, на главную диагональ таких строк (соответствующих фиктивным узлам) можно, например, поставить единицу (и тогда в фиктивных узлах после решения конечноэлементной СЛАУ получатся нулевые веса).

Таким образом, данных, хранящихся в массивах X и Y и в файле описания области, достаточно для получения всей необходимой информации о конечных элементах в регулярной прямоугольной сетке при указанном выше способе нумерации узлов (с введением фиктивных узлов сетки при непопадании их в расчётную область).

Покажем, как можно получить всю необходимую информацию для построения конечноэлементной аппроксимации при использовании регулярной прямоугольной сетки с биквадратичными элементами.

Обозначим через N_x количество координатных линий $x = x_p$, являющихся вертикальными границами конечных элементов (и хранящихся в массиве X), а через N_y количество координатных линий $y = y_s$, являющихся горизонтальными границами конечных элементов (и хранящихся в массиве Y). Тогда число биквадратичных конечных элементов будет равно $\left(N_x - 1\right)\left(N_y - 1\right)$, а количество неизвестных q_i будет равно $\left(2N_x - 1\right)\left(2N_y - 1\right)$.

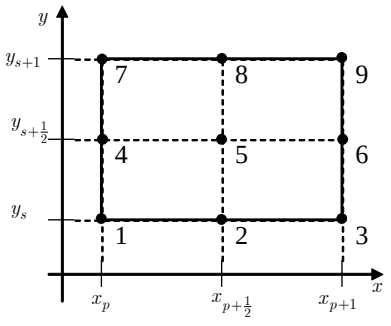


Рис. 3. Расположение узлов на прямоугольном конечном элементе с биквадратичными базисными функциями

Будем считать, что локальная нумерация узлов на биквадратичном элементе определена в соответствии с рис. 3, а глобальная нумерация всех узлов сетки (включая внутренние) выполнена по принципу возрастания их номеров сначала по x, затем по y. Тогда координаты узлов конечного элемента с номером i можно определить с помощью соотношений

$$p = \left(\left(i-1\right) \bmod \left(N_x - 1\right)\right) + 1, \qquad s = \left[\frac{i-1}{N_x - 1}\right] + 1, \tag{1.1}$$

$$x_p = X\left(p\right), \quad x_{p+1} = X\left(p+1\right), \quad y_s = Y\left(s\right), \quad y_{s+1} = Y\left(s+1\right), \tag{1.2}$$

$$x_{p+\frac{1}{2}} = \frac{x_p + x_{p+1}}{2}, \qquad y_{s+\frac{1}{2}} = \frac{y_s + y_{s+1}}{2}. \tag{1.3}$$

Номер подобласти W_m, которой принадлежит конечный элемент Ω_i, можно легко получить, проверив, в какую из подобластей попал центральный узел Ω_i с координатами $\left(x_{p+\frac{1}{2}}, y_{s+\frac{1}{2}}\right)$ (этот номер необходим для вычисления значений параметров дифференциального уравнения).

Глобальные номера девяти локальных базисных функций конечного элемента Ω_i можно легко вычислить с помощью целочисленной функции

$$K(i) = 2\left[\frac{i-1}{N_x - 1}\right](2N_x - 1) + 2\Big((i-1) \bmod (N_x - 1)\Big) + 1. \tag{1.4}$$

Так, узлу $\left(x_p, y_s\right)$ соответствует глобальная базисная функция с номером $K(i)$, узлам $\left(x_{p+\frac{1}{2}}, y_s\right)$ и $\left(x_{p+1}, y_s\right)$ – глобальные функции со следующими двумя номерами $K(i)+1$ и $K(i)+2$. Узлу $\left(x_p, y_{s+\frac{1}{2}}\right)$ соответствует глобальная базисная функция с номером $K(i)+2N_x - 1$, узлам $\left(x_{p+\frac{1}{2}}, y_{s+\frac{1}{2}}\right)$ и $\left(x_{p+1}, y_{s+\frac{1}{2}}\right)$ – глобальные функции со следующими двумя номерами. И, наконец, узлам $\left(x_p, y_{s+1}\right)$, $\left(x_{p+\frac{1}{2}}, y_{s+1}\right)$ и $\left(x_{p+1}, y_{s+1}\right)$ – глобальные базисные функции с тремя последовательными номерами, начиная с номера $K(i)+2(2N_x - 1)$.

Здесь мы рассмотрели способ, который позволяет получить всю необходимую информацию о любом конечном элементе Ω_i *по номеру* i из структур данных, описывающих расчётную область и регулярную сетку.

Однако чаще всего при работе с регулярными сетками проход по конечным элементам организуют не *одним циклом* по i (i – номер обрабатываемого конечного элемента), а *двумя* (в двумерном случае) или *тремя* (в трёхмерном) вложенными циклами. Самый внутренний цикл – по p от 1 до $N_x - 1$ (N_x – количество координатных линий, определяющих границы конечных элементов по координате x и хранящихся в массиве X). Внешний к нему цикл – по s от 1 до $N_y - 1$ (N_y – количество координатных линий по y, хранящихся в массиве Y). И если задача

трёхмерная, самый внешний цикл по r от 1 до $N_z - 1$ (N_z – количество координатных линий по z, которые должны храниться в массиве Z).

В этом случае координаты вершин обрабатываемого конечного элемента определяются совсем просто – это компоненты массивов X, Y или Z с номером, равным значению соответствующей переменной цикла p, s или r. Координаты узлов на рёбрах и внутри конечного элемента определяются через координаты его вершин по формулам вида (1.3).

Так же просто определяются глобальные номера локальных базисных функций. Для биквадратичного элемента (при нумерации узлов по принципу возрастания их номеров сначала по x, затем по y) узел $\left(x_p, y_s\right)$ и соответствующая ему глобальная базисная функция имеет номер $\left(s-1\right)\left(2N_x - 1\right) + p$, узлы $\left(x_{p+\frac{1}{2}}, y_s\right)$ и $\left(x_{p+1}, y_s\right)$ – следующие за ним номера, узел $\left(x_p, y_{s+\frac{1}{2}}\right)$ имеет номер $s\left(2N_x - 1\right) + p$, узлы $\left(x_{p+\frac{1}{2}}, y_{s+\frac{1}{2}}\right)$ и $\left(x_{p+1}, y_{s+\frac{1}{2}}\right)$ – следующие за ним номера, а узел $\left(x_p, y_{s+1}\right)$ имеет номер $\left(s+1\right)\left(2N_x - 1\right) + p$ (и соответственно узлы $\left(x_{p+\frac{1}{2}}, y_{s+1}\right)$ и $\left(x_{p+1}, y_{s+1}\right)$ – следующие за ним номера).

Номер же подобласти, которой принадлежит обрабатываемый элемент, можно, как это было описано выше, определить с помощью центрального узла сравнениями его координат с хранящимися в массивах X^W и Y^W (и в трёхмерном случае в массиве Z^W) координатами линий, определяющих границы подобластей W_m.

Однако если подобластей W_m довольно много, то количество операций, необходимое для определения номера подобласти, может стать довольно значительным. Ускорить эту процедуру можно, заменив сравнения вещественных координат на *целочисленные* сравнения переменных цикла p, s (и r в трёхмерном случае) с *номерами* координатных линий, ограничивающих подобласти W_m.

Для того чтобы определять принадлежность конечного элемента той или иной подобласти, нужны два (в двумерном случае) целочисленных массива IX^W и IY^W, сформированных следующим образом. В ячейку массива IX^W с номером j должен быть занесён номер ячейки массива X, в которой хранится та же точка, что и в j-й ячейке массива X^W (напомним, что в массиве X хранятся x-координаты *линий сетки*, а в массиве X^W хранятся x-координаты *границ подобла-*

стей W_m). Аналогично должен быть сформирован и массив IY^W (а также массив IZ^W в трёхмерном случае). Очевидно, эти массивы удобней всего формировать в тот момент, когда создаются массивы X и Y по информации из массивов X^W и Y^W и из файла, определяющего сетку (через дробление интервалов между соседними точками в массивах X^W и Y^W).

Покажем в качестве примера вид массивов IX^W и IY^W для области, изображённой на рис. 2. Описывающая эту область структура данных была приведена на с. 7. Будем считать, что разбиение этой расчётной области определяется файлом, представленным на с. 10 (сразу же за этим файлом приведены массивы X и Y с координатами линий сетки). Тогда массивы IX^W и IY^W будут иметь вид:

$$IX^W = \{1,\ 6,\ 10,\ 14\}, \qquad IY^W = \{1,\ 6,\ 7,\ 10,\ 11,\ 13\}.$$

Теперь, чтобы определить, попадает ли конечный элемент $\Omega_k = [x_p, x_{p+1}] \times [y_s, y_{s+1}]$ в подобласть W_m, нужно взять из описания расчётной области номера m_x^-, m_x^+ и m_y^-, m_y^+ ограничивающих W_m координатных линий (это со второй по пятую компоненты m-й строки массива $MW(5,*)$, описывающего все подобласти, т.е. $m_x^- = MW(2,m)$, $m_x^+ = MW(3,m)$, $m_y^- = MW(4,m)$ и $m_y^+ = MW(5,m)$), и проверить, лежат ли внутри целочисленного отрезка $[m_x^-, m_x^+]$ значения p и $p+1$, а также лежат ли значения s и $s+1$ внутри целочисленного отрезка $[m_y^-, m_y^+]$.

1.2.4. ОПИСАНИЕ ДВУМЕРНЫХ ОБЛАСТЕЙ С КРИВОЛИНЕЙНЫМИ ГРАНИЦАМИ

Чаще всего двумерные области с криволинейными границами задаются с помощью метода описания границ. При этом для описания подобластей удобно использовать понятие элементарной подобласти.

Элементарная подобластьэлементарная подобласть" – это участок области, со всех сторон окружённый границами области (внутренними или внешними) или их фрагментами и не имеющий границ или их фрагментов внутри себя. Очевидно, что в том случае, когда граница задаётся только из отрезков прямых, элементарная подобласть является многоугольником (возможно, невыпуклым). Для задания каждой элементарной подобласти расчётной области обычно указывается одна точка, лежащая внутри её. Границы же элементарной подобласти чаще всего определяются автоматически с использованием только списка всех границ расчёт-

ной области и координат одной точки, лежащей внутри этой элементарной подобласти.

Рассмотрим один из возможных вариантов структуры данных для описания областей с криволинейными границами. Будем считать, что все внутренние и внешние границы расчётной области разбиты на участки – *фрагменты границы*.

Описание каждого фрагмента границы должно содержать информацию о *координатах его концов* (удобнее всего в виде номеров точек в соответствующем массиве, где хранятся координаты этих точек), о *геометрических характеристиках* этого фрагмента (например, если фрагментом является дуга окружности, то это может быть информация о её центре и радиусе), а также о заданных на нём *краевых условиях и их параметрах*. Кроме того, для некоторых алгоритмов построения сетки требуется информация о разбиении всех границ расчётной области, и поэтому для каждого фрагмента границы довольно часто хранят и информацию о его разбиении. Например, это может быть *число интервалов*, на который требуется разбить фрагмент, и *коэффициент изменения шага* (растяжения или сжатия) от начала фрагмента к концу (т.е. коэффициент 1.2 означает, что каждый следующий шаг должен быть в 1.2 раза больше предыдущего).

Приведём пример формата текстового файла, хранящего такую структуру данных. В первой строке задаётся одно целое число N, означающее длину массива точек – концов фрагментов границ. Начиная со следующей строки задаются N пар вещественных чисел – это массив координат этих точек. Затем задаётся целое число M – число фрагментов границы. Далее следует M блоков информации о всех фрагментах границы.

Каждый блок информации о фрагменте границы начинается с целого числа, определяющего тип фрагмента (0 – отрезок прямой, 1 – дуга окружности и т.д.). Следующие два целых числа – это номера начальной и конечной точек фрагмента границы в массиве точек (концов фрагментов границ). Затем задаются вещественные числа – геометрические характеристики фрагмента, количество которых, естественно, зависит от типа фрагмента. Например, для отрезка прямой никакая дополнительная информация вообще не нужна. Для дуги же окружности достаточно хранить одно число – отношение α расстояния от середины дуги до середины её хорды (т.е. отрезка, соединяющего концы фрагмента границы) к длине этой хорды. При этом знак α может определять, какая из двух дуг задаётся – слева или справа от направленного отрезка (хорды).

Следующее целое число означает тип краевого условия, которое задано на фрагменте границы (0 – внутренняя граница или вспомогательная линия для управления построением сетки, 1 – краевое условие первого рода, 2 – краевое условие второго рода и т.д.). Затем для всех типов краевых условий (кроме типа 0) задаётся одно целое число – номер границы (т.е. номер формул, задающих параметры краевого условия на рассматриваемом фрагменте границы).

Последние два числа блока определяют параметры разбиения фрагмента границы: первое число (целое) определяет число интервалов, на которые разбивается фрагмент, а второе (вещественное) – коэффициент изменения (растяжения или сжатия) шага. Часто для удобства даётся возможность при описании разбиения фрагмента границы задавать отрицательный коэффициент, который означает, что изменение шага задано не от первой точки («начала») фрагмента ко второй («концу»), а от второй («конца») к первой («началу»).

Следующее за описанием фрагментов границы целое число K означает количество элементарных подобластей в расчётной области. Затем задаётся K троек чисел, первое из которых (целое) означает *номер формул*, определяющих параметры дифференциального уравнения в элементарной подобласти, а два остальных (вещественные) – координаты произвольной внутренней точки этой элементарной подобласти.

Приведём пример файла с такой структурой для изображённой на рис. 4 прямоугольной расчётной области с круглым отверстием, состоящей из двух подобластей W_1 и W_2. Будем считать, что точки, являющиеся концами фрагментов границ, и сами фрагменты пронумерованы так, как это показано на рис. 4. При этом номера точек указаны в прямоугольничках, а номера фрагментов – в кружочках. Будем также считать, что на границах-окружностях расчётной области заданы краевые условия третьего рода, а на всех остальных внешних границах – краевые условия первого рода. В этом случае содержимое файла будет следующим:

```
8
0. 0.     8. 0.     0. 6.     2. 6.
5. 6.     8. 6.     0. 11.    8. 11.
10
0   1 2         1 1   4  1.
0   1 3         1 1   4 -1.5
0   2 6         1 1   4 -1.5
```

```
0    3 4          0      3   1.
1    4 5   0.5    3 1    4   1.
0    5 6          0      4   1.
1    4 5  -0.5    3 1    4   1.
0    3 7          1 1    4   1.5
0    6 8          1 1    4   1.5
0    7 8          1 1    4   1.
2
1    3.     2.
2    3.    10.
```

Рассмотренная структура данных является достаточно универсальной и может быть использована в программных комплексах, которые предназначены для построения сеток с треугольными ячейками или любых других сеток, допускающих произвольные сгущения узлов в расчётной области. Естественно, эта структура данных является не единственно возможной для описания областей со сложными границами. Рассмотрим, например, структуру текстового файла данных, предназначенного для систем построения сеток с четырёхугольными ячейками.

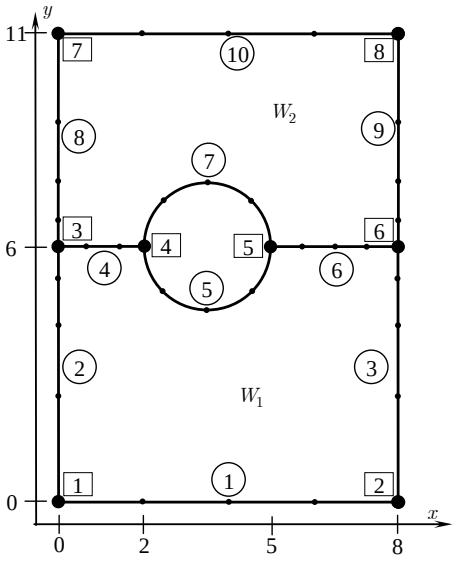


Рис. 4. Пример расчётной области с криволинейными границами

Обычно сетки с четырёхугольными ячейками используются для дискретизации областей, представляемых в виде объединения элементарных подобластей, которые сами тоже являются *псевдочетырёхугольниками*. Под псевдочетырёхугольником мы понимаем фигуру, у которой так же, как у четырёхугольника, можно выделить четыре стороны, но эти стороны могут быть не отрезками прямых, а ломаными линиями. При этом противоположные стороны таких элементарных подобластей (псевдочетырёхугольников) разбиваются одинаково (на одно и то же число интервалов). Таким образом, аналогами координатных линий в этом случае являются «горизонтальные» и «вертикальные» ломаные, а элементарные подобласти в виде псевдочетырёхугольников задаются номерами этих координатных (ломаных) линий, между которыми они находятся. Поэтому соответствующая структура данных больше похожа не на рассмотренную на с. 15–17 уни-

версальную структуру данных для описания областей с криволинейными границами, а на рассмотренную в разделе 1.2.3 структуру для описания областей с прямоугольными подобластями. Фактически всё отличие заключается лишь в том, что вместо границ, параллельных осям координат, требуется описать ломаные линии – аналоги координатных линий.

Итак, в первой строке файла, в котором хранится соответствующая структура данных, задаются два целых числа N_x^W и N_y^W. Число N_x^W означает число узлов на ломаных вдоль «горизонтального» направления (по аналогии с точками на горизонтальных координатных линиях в прямоугольной сетке), а число N_y^W – вдоль «вертикального» (по аналогии с точками на вертикальных координатных линиях). Затем следует $N_x^W \times N_y^W$ пар вещественных чисел – координат узлов ломаных, при этом N_x^W узлов, соответствующих каждой из N_y^W «горизонтальных» ломаных, *задаются последовательно*. При этом k-я «вертикальная» ломаная определяется парами координат с номерами k, $\left(N_x^W + k\right)$, $\left(2N_x^W + k\right)$,…, $\left(\left(N_y^W - 1\right)N_x^W + k\right)$.

После них в файле хранится информация об элементарных подобластях в том же виде, как и при описании прямоугольных подобластей, т.е. число L (целое) – количество подобластей $W_1, W_2, \ldots$, из которых состоит расчётная область Ω; далее L наборов по 5 целых чисел, описывающих все подобласти $W_1, W_2, \ldots$, первое из которых означает номер формул, определяющих параметры дифференциального уравнения в подобласти, второе – номер «вертикальной» ломаной, определяющей левую границу подобласти W_m, третье – номер «вертикальной» ломаной, определяющей правую границу W_m, четвёртое – номер «горизонтальной» ломаной, определяющей нижнюю границу W_m, и пятое – номер «горизонтальной» ломаной, определяющей верхнюю границу W_m.

Покажем, как будет выглядеть файл с описанной структурой для изображённой на рис. 5 расчётной области. Эта расчётная область отличается от изображённой на рис. 4 только тем, что круглое отверстие заменено восьмиугольным. Соответствующий файл будет иметь вид:

```
5 5
0. 0.    2. 0.      4. 0.    6. 0.      8. 0.
0. 5.27  2.44 4.94  3.5 4.5  4.56 4.94  8. 5.27
```

```
0. 6.     2. 6.       3.5 6.    5. 6.       8. 6.
0. 6.73   2.44 7.06   3.5 7.5   4.56 7.06   8. 6.73
0. 11.    2. 11.      4. 11.    6. 11.      8. 11.

6
1 1 2 1 3
1 2 4 1 2
1 4 5 1 3
2 1 2 3 5
2 2 4 4 5
2 4 5 3 5
```

Совсем несложно изменить последнюю структуру данных так, чтобы она позволяла более точно описывать криволинейные границы области. Для этого достаточно организовать хранение «горизонтальных» и «вертикальных» ломаных таким образом, чтобы их участками могли быть не только отрезки прямых, но и дуги окружностей (или какие-нибудь другие кривые, описываемые, например, кубическими сплайнами). В этом случае все генерируемые на сторонах псевдочетырёхугольника с криволинейными границами внутренние узлы можно ставить точно на границы (т.е. с учётом их возможного искривления). Читатель может попробовать создать соответствующую структуру данных в качестве упражнения.

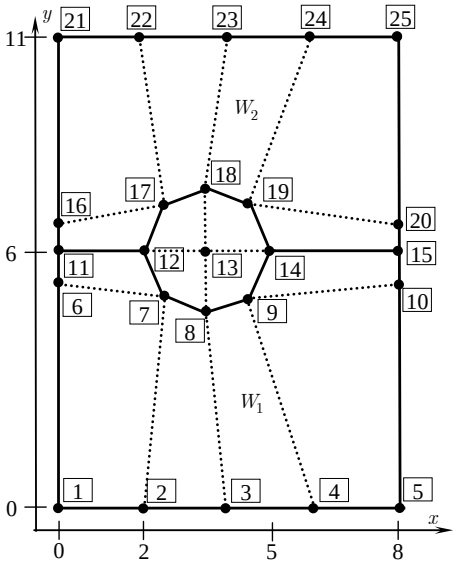


Рис. 5. Пример расчётной области с разбиением на четырёхугольные элементарные подобласти

1.2.5. РЕГУЛЯРНЫЕ СЕТКИ С ЧЕТЫРЁХУГОЛЬНЫМИ ЯЧЕЙКАМИ

Рассмотренные в разделе 1.2.3 подходы к построению регулярных прямоугольных сеток можно с не слишком большими изменениями использовать и для построения регулярных сеток с ячейками в виде четырёхугольников, если для описания расчётной области использована структура данных, рассмотренная в конце раздела 2.2.4 на с. 17-19.

В этом случае разбиение области на ячейки задаётся, как и для прямоугольной сетки (см. с. 10), с помощью файла, в котором содержится информация о разбиениях на подынтервалы всех интервалов ме-

жду соседними точками на «горизонтальных» и «вертикальных» ломаных, деляющих границы подобластей W_m.

Покажем, как можно задать четырёхугольную сетку для области, показанной на рис. 5 (файл её описания приведён на с. 18).

Пусть разбиение этой области определяется следующим файлом

```
2 1.      1 1.    1 1.    3 1.
3 -1.5    1 1.    1 1.    3 1.5
```

Содержимое этого файла определяет следующее разбиение расчётной области. Интервалы между первыми двумя узлами всех «горизонтальных» ломаных должны быть разбиты на два подынтервала (на это указывает первое число в первой паре), при этом длины подынтервалов не должны меняться (поскольку второе число в первой паре – коэффициент растяжения – равен единице). Следующие два интервала на всех горизонтальных линиях вообще не разбиваются (поскольку в следующих двух парах чисел первые числа, определяющие количество подынтервалов, равны единице). Четвёртые же интервалы на всех горизонтальных ломаных разбиваются на три подынтервала с равномерным (поскольку коэффициент растяжения равен единице) шагом.

Следующие четыре пары чисел аналогичным образом определяют разбиение интервалов «вертикальных» ломаных на подынтервалы. Каждый первый интервал всех «вертикальных» ломаных разбивается на 3 подынтервала, причём так, что каждый последующий подынтервал должен быть в 1.5 раза меньше предыдущего. На это указывает коэффициент растяжения, равный –1.5 (отрицательные коэффициенты растяжения интерпретируется нами как соответствующие коэффициенты сжатия или, что то же самое, коэффициент растяжения не от первой точки интервала ко второй, а, наоборот, от второй к первой – об этом уже говорилось на с. 16).

На рис. 6 показана регулярная четырёхугольная сетка, построенная для изображённой на рис. 5 расчётной области (с файлом её описания, приведённым на с. 18, и с учётом рассмотренного здесь файла, определяющего эту сетку). Ячейки этой сетки, как и прямоугольной, образуются соседними «горизонтальными» и «вертикальными» линиями. Однако описать эти линии уже невозможно с помощью двух независимых массивов, аналогичных массивам X и Y. Очевидно, это можно сделать так же, как были заданы «горизонтальные» и «вертикальные» линии в описании расчётной области, – для этого нужно перечислить номера всех узлов сетки последовательно вдоль «горизонтальных» линий слева направо, передвигаясь от одной «горизонтальной» линии к другой снизу вверх.

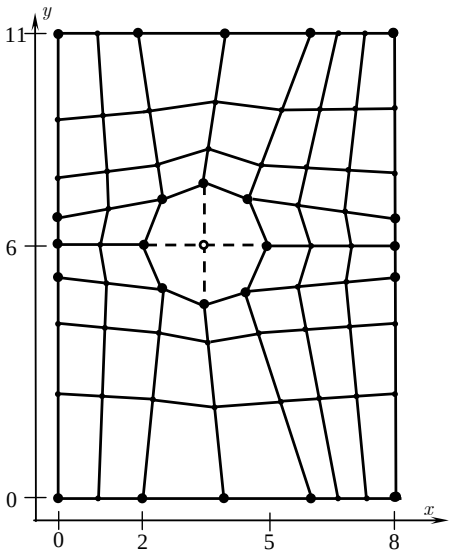


Рис. 6. Пример регулярной четырёхугольной сетки

Координаты узлов регулярной четырёхугольной сетки можно вычислить по довольно простой схеме. Сначала можно расставить узлы на «основных» «горизонтальных» и «вертикальных» линиях, заданных в файле расчётной области, с учётом количества подынтервалов и коэффициентов растяжения-сжатия. Благодаря регулярности сетки эти узлы можно ставить сразу же на нужные места в формируемом массиве координат узлов сетки. После этого не представляет никакого труда сформировать все остальные узлы. Для этого достаточно сформировать «вспомогательные» «горизонтальные» линии между узлами, поставленными на «основных» «вертикальных» линиях на предыдущем этапе, и разбить их точно так же, как и «основные», с учётом количества подынтервалов и коэффициентов растяжения-сжатия, соответствующих рассматриваемому интервалу «основной» «горизонтальной» линии.

Для примера, изображённого на рис. 6, на первом этапе в массив узлов будут по этой схеме занесено 60 узлов. Первый узел (с номером 1) совпадает с первым узлом первой «горизонтальной» ломаной из файла описания расчётной области – его координаты $x_1 = 0$, $y_1 = 0$. Координаты второго узла (с номером 2) получаем из условия, что первый интервал всех «основных» «горизонтальных» линий разбит на два подынтервала с единичным коэффициентом растяжения, – это координ-

наты $x_2 = 1$, $y_2 = 0$. Аналогично получаются координаты узлов с номерами 3-х до 8-и, от 25-и до 48-и и от 65-и до 72-х.

Координаты узла с номером 9 получаются с учётом того, что первый интервал всех «основных» «вертикальных» линий разбит на три подынтервала с коэффициентом растяжения –1.5, – его координаты $x_9 = 0$, $y_9 = 2.5$. Следующим заносится узел с номером 11. Его координаты также вычисляются с учётом разбиения первого интервала «основных» «вертикальных» линий на три подынтервала с коэффициентом растяжения –1.5. Поскольку этот интервал заключён между точками $\left(2,0\right)$ и $\left(2.44,4.94\right)$, узел с номером 11 будет иметь координаты $x_{11} = 2.21$, $y_{11} = 2.34$. Аналогично вычисляются координаты узлов с номерами 12,13,16,17,19 и т.д.

На втором этапе будут сформированы координаты узлов с номерами 10, 14, 15, 18, 22 и т.д. Например, координаты узла с номером 10 можно определить с учётом того, что интервал между уже сформированными узлами с номерами 9 и 11 должен быть разбит на два подынтервала с коэффициентом растяжения-сжатия $\alpha = 1$, как и все первые интервалы «горизонтальных» ломаных, описывающих расчётную область. Таким образом, $x_{10} = 1.115$, $y_{10} = 2.42$.

Обозначим через N_x количество точек на каждой «горизонтальной» линии сетки, а через N_y – количество точек на каждой «вертикальной» линии сетки (с учётом разбиения на подынтервалы интервалов всех «горизонтальных» и «вертикальных» линий из описания расчётной области). Таким образом, в полученной четырёхугольной сетке будет $N_x \times N_y$ узлов и $\left(N_x - 1\right)\times\left(N_y - 1\right)$ ячеек.

Проход по конечным элементам для регулярных сеток с четырёхугольными ячейками удобней всего, как и для регулярных прямоугольных сеток, осуществлять двумя вложенными циклами: внутренний цикл по p от 1 до $N_x - 1$, внешний – по s от 1 до $N_y - 1$. Определять же принадлежность ячейки той или иной подобласти (для вычисления значений параметров дифференциального уравнения) имеет смысл с использованием стратегии, основанной на целочисленных сравнениях номеров «горизонтальных» и «вертикальных» линий из описания расчётной области с номерами «горизонтальных» и «вертикальных» линий сетки. Это можно сделать с помощью массивов IX^W и IY^W, которые формируются точно так же, как это было описано для прямоугольных сеток на с. 13.

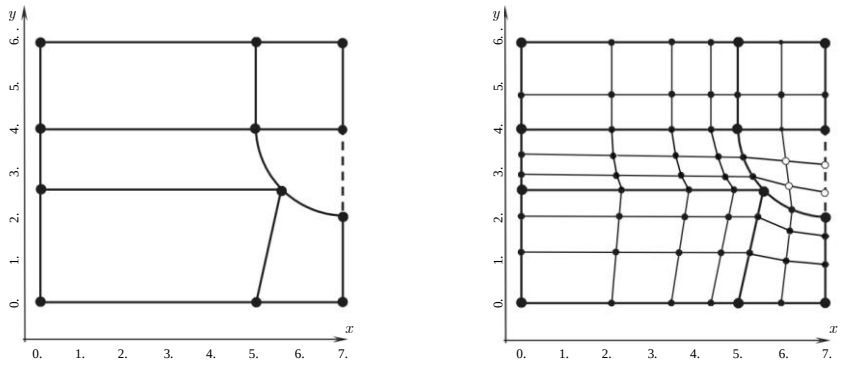
Для рассмотренного нами примера (см. рис. 6) массивы IX^W и IY^W будут иметь вид:

$$IX^W = \{1,3,4,5,8\}, \qquad IY^W = \{1,4,5,6,9\}.$$

Стратегия определения номера подобласти для четырёхугольных сеток ничем не отличается от соответствующей стратегии для прямоугольных сеток (см. с. 14).

Обратим внимание на то, что некоторые ячейки и узлы регулярной четырёхугольной сетки (так же, как и при использовании прямоугольных сеток) могут не попадать в расчётную область. Для приведённого на рис. 6 примера это четыре ячейки внутри «дырки» и узел с номером 36 (в самом центре «дырки», на рис. 6 он обозначен незакрашенным кружком). Очевидно, что для узлов регулярной сетки, лежащих вне расчётной области, не будут сгенерированы конечноэлементные уравнения, и обработка этой ситуации может быть проведена так же, как и в случае регулярных прямоугольных сеток (см. с. 11).

Ещё одна важная особенность сеток с четырёхугольными ячейками заключается в возможности уточнения описания криволинейных границ при измельчении ячеек сетки. В рассмотренном нами примере (рис. 5) круглое отверстие (см. рис. 4) было заменено правильном восьмиугольником. В этом случае дальнейшее дробление сетки не приведёт к уточнению границ расчётной области (т.е. новые узлы будут ставиться на отрезки, соединяющие вершины многоугольника). В принципе же, при описании расчётной области отдельные участки «горизонтальных» и «вертикальных» ломаных, определяющих эту область, можно хранить *как некоторые кривые* (части окружностей, парабол, сплайнов и т.д.). Это можно делать, например, введением некоторого массива, в котором задаются номера «горизонтальных» ломаных и номера тех их интервалов, которые являются дугами окружностей, и для этих интервалов задавать параметры соответствующей дуги (либо в виде координат центра окружности, либо в виде одного числа – отношения расстояния от середины дуги до середины её хорды к длине этой хорды, см. с. 15 о параметре α). Аналогично можно описывать криволинейные участки и на «вертикальных» ломаных. Задание вместо отрезков «горизонтальных» и «вертикальных» ломаных частей парабол, сплайнов и других кривых можно организовать точно так же с помощью соответствующих дополнительных массивов.



а *б*

Рис. 7. Пример расчётной области и построение в ней четырёхугольной сетки с учётом криволинейной границы

Проиллюстрируем возможность более точного учёта криволинейных границ при измельчении сеток с четырёхугольными ячейками на примере разбиения области, изображённой на рис. 7,*а*. Для описания этой области будем использовать практически такую же структуру данных, которая была рассмотрена на с. 17. В ней сначала задаются два целых числа N_x^W и N_y^W, а затем $N_x^W \times N_y^W$ пар вещественных чисел (координат узлов), определяющих «горизонтальные» и «вертикальные» ломаные:

```
3 4
0. 0.      5.   0.      7. 0.
0. 2.59    5.59 2.59    7. 2.
0. 4.      5.   4.      7. 4.
0. 6.      5.   6.      7. 6.
```

После этого задаётся информация об элементарных подобластях:

```
3
1 1 2 1 4
1 2 3 1 2
2 2 3 3 4
```

И, наконец, дополнительно (по отношению к рассмотренной на с. 17 структуре данных) зададим информацию о криволинейных участках «горизонтальных» и «вертикальных» ломаных в следующем формате. Первым идёт число M, определяющее общее количество криволинейных участков границ. Далее следуют M пятёрок чисел, определяющих каждый участок. Первое число каждой четверки – признак того, является ли линия «горизонтальной» (тогда это число равно 1) или

«вертикальной» (тогда это число равно -1). Второе число – номер ломаной, третье – номер интервала. Последние два числа – это параметры дуги (координаты центра окружности). Для рассматриваемого примера эта информация имеет вид

```
2
-1  2  2  7.  4.
 1  2  2  7.  4.
```

На рис. 7,*б* показана сетка, получаемая при её построении по файлу следующего вида:

```
4 -1.5      2 1.
3 -1.4      3 1.25      2 1.5
```

Обратим внимание на то, что при разбиении криволинейных участков узлы были поставлены не на хорды, а на сами дуги, и узлы на вспомогательных линиях генерировались с учётом этого.

Дополнительным преимуществом регулярных сеток является то, что не только для хранения их самих, но и для *хранения матриц* конечноэлементных СЛАУ, получаемых в результате аппроксимации на регулярных сетках, можно использовать самые экономичные структуры данных (такие, как диагональный формат хранения матриц, который будет рассмотрен ниже во втором разделе).

Сетки с ячейками в виде треугольников, тетраэдров и призм, как правило, являются *нерегулярными*, и для их хранения обычно используются специальные структуры данных. Довольно часто такие структуры данных используются для хранения сеток с ячейками в виде четырёхугольников и шестигранников, а иногда и для прямоугольных сеток, когда расчётная область *имеет большие «дыры»* и слишком много узлов регулярной сетки не попадает в расчётную область (т.е. при использовании регулярной сетки они должны становиться фиктивными). Кроме того, когда в регулярных сетках делаются *локальные дробления* (в частности, в прямоугольных сетках с появлением в них несогласованных ячеек), их регулярность также, как правило, нарушается.

Заметим, что идея набрасывания регулярной сетки (из прямоугольников или четырёхугольников в двумерном случае либо из параллелепипедов или шестигранников в трёхмерном случае) может быть довольно плодотворной даже при формировании нерегулярных конечноэлементных сеток в областях с существенно

изрезанными внешними границами и большими «дырами». В таких случаях в конечноэлементную сетку можно не включать ячейки и узлы исходной регулярной сетки, не попавшие в расчётную область, но для описания регулярной сетки и организации прохода по её ячейкам (для записи в конечноэлементную сетку только тех из них, которые попали в расчётную область) можно использовать рассмотренные нами выше структуры данных и алгоритмы работы с регулярными сетками.

1.2.6. ХРАНЕНИЕ НЕРЕГУЛЯРНЫХ СЕТОК

Для нерегулярных сеток наиболее удобным является *поэлементный* способ хранения. При таком способе сетка задаётся в виде *набора вершин* и *набора элементов*. Каждая вершина сетки описывается набором координат (двумя координатами x_k и y_k в двумерном случае или тремя координатами x_k, y_k и z_k в трёхмерном случае), а каждый элемент Ω_m – набором из нескольких целых чисел. Например, каждый конечный элемент можно хранить в виде набора следующих чисел:

- номеров узлов, являющихся вершинами рассматриваемого элемента;
- глобальных номеров базисных функций элемента;
- номера подобласти расчётной области, служащего для определения значений параметров дифференциального уравнения внутри данного элемента.

Естественно, в случае совпадения номеров глобальных базисных функций с номерами вершин элементов (или при возможности их простого вычисления по этим номерам) нет никакой необходимости хранить глобальные номера базисных функций элемента. Например, для элементов первого порядка, у которых число базисных функций совпадает с числом вершин, для каждого элемента хранится ровно $l+1$ целых чисел, где l – количество вершин элемента, а последнее $\left(l+1\right)$-е число – номер подобласти, которой принадлежит элемент.

Для хранения сеток с лагранжевыми конечными элементами порядка выше первого, когда количество геометрических вершин не совпадает с количеством узлов (к которым привязаны локальные базисные функции), можно использовать различные варианты структур данных. Мы рассмотрим два из них. В первом варианте массив узлов содержит все узлы сетки, включая те, которые не являются геометрическими вершинами элементов. Во втором же варианте в массиве узлов хранятся только геометрические вершины конечных элементов, а координаты узлов, к которым привязаны базисные функции, вычисляются через координаты

вершин элемента. Очевидно, что первый вариант позволяет экономить время на вычисление координат узлов, а второй – память на их хранение. Однако экономия вычислительных затрат в обоих случаях достаточно мала по сравнению с общими вычислительными затратами на решение задачи, и выбор обычно делается в пользу первого варианта как более простого для реализации.

Очевидно, что кроме информации о конечных элементах для решения задачи требуется ещё и информация о краевых условиях. Для нерегулярных сеток границы чаще всего также хранятся поэлементно – рёбрами в двумерных задачах и гранями в трёхмерных.

Краевые условия второго и третьего рода представляются наборами таких граничных элементов. Каждый граничный элемент (ребро или грань), являющийся частью границы с заданным на ней краевым условием второго или третьего рода, описывается набором целых чисел – номерами ненулевых на этом граничном элементе глобальных базисных функций и числом, которое служит для определения значений параметров краевого условия (θ для второго краевого условия или β и u_β для третьего).

Краевое же условие первого рода чаще всего представляется набором пар целых чисел, в котором первое число каждой пары является номером узла, лежащим на границе, а второе служит для определения значения параметра u_g этого краевого условия. Заметим, что для эрмитовых конечных элементов или при необходимости выполнения дополнительных дроблений сетки информацию о границе S_1 лучше хранить всё-таки в виде набора граничных элементов (рёбер или граней) так же, как и в случае границ со вторым и третьим краевыми условиями.

Рассмотрим два варианта структур данных, предназначенных для хранения сеток из треугольных элементов первого и второго порядка. Простейшие примеры таких сеток показаны на рис. 8 (с треугольниками первого порядка) и 9 (с треугольниками второго порядка). На обоих рисунках треугольники с номерами $1-3$ лежат в подобласти W_1, а треугольник с номером 4 – в подобласти W_2. Будем считать, что в обоих примерах на горизонтальных (нижней и верхней) границах расчётной области заданы краевые условия второго рода (с разными значениями параметра θ), а на левой (вертикальной) и правой (наклонной) границах – краевые условия первого рода (также с разными значениями параметра u_g).

Для первого примера массив узлов состоит из следующих шести пар вещественных чисел:

```
1. 1.   3. 1.   4. 1.   6. 1.   1. 3.   4. 3.
```

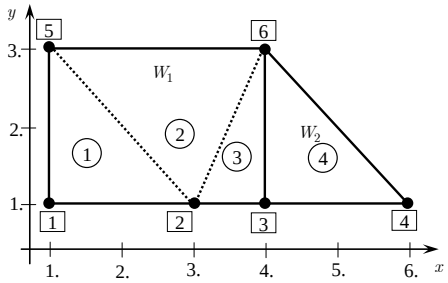


Рис. 8. Пример сетки из треугольных элементов первого порядка

Будем считать, что каждый треугольник первого порядка описывается четырьмя целыми числами: тремя номерами узлов, являющихся его вершинами, и номером подобласти, которой он принадлежит. Тогда информация о всех треугольниках сетки может быть представлена массивом, состоящим из четвёрок целых чисел, и для сетки, изображённой на рис. *8*, этот массив будет содержать следующие четвёрки чисел:

```
1 2 5 1     2 6 5 1     2 3 6 1     3 4 6 2
```

Краевые условия второго рода для элементов первого порядка можно хранить в отдельном массиве тройками целых чисел, первые два из которых – номера вершин ребра, а третье – номер формулы, по которой должно вычисляться значение параметра θ. Для рассматриваемой нами задачи (в которой краевые условия второго рода заданы на нижней и верхней границах расчётной области и параметр θ на этих границах может определяться разными формулами) и для сетки, изображённой на рис. *8*, массив с информацией о вторых краевых условиях примет вид

```
1 2 1    2 3 1    3 4 1    5 6 2
```

Информация о краевых условиях первого рода при использовании элементов первого порядка может быть задана массивом пар целых чисел, в которых первое число – номер узла, а второе – номер формулы, по которой должно вычисляться значение параметра u_g. В нашем примере (соответствующем рис. *8*) этот массив будет содержать четыре пары целых чисел:

```
1 1    5 1    4 2    6 2
```

Для сеток с квадратичными (и более высокого порядка треугольными элементами) в массиве узлов можно хранить парами координаты *всех узлов* сетки, включая узлы, не являющиеся вершинами треугольников. Тогда для сетки, изображённой на рис. 9, этот массив будет иметь вид:

```
1. 1.    2. 1.    3. 1.    3.5  1.    4.   1.
5. 1.    6. 1.    1. 2.    2.   2.    3.5  2.
4. 2.    5. 2.    1. 3.    2.5  3.    4.   3.
```

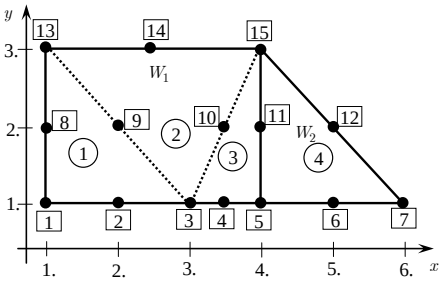


Рис. 9. Пример сетки из треугольных элементов второго порядка

Треугольники в этом случае можно хранить набором семёрок целых чисел. В каждой семёрке чисел (описывающей один треугольный конечный элемент) первые три числа – это номера вершин треугольника, являющихся вершинами треугольника, следующие три – глобальные номера узлов, расположенных на серединах рёбер. Такое перечисление узлов квадратичного треугольника соответствует локальной нумерации его узлов, приведённой на рис. 10. Последнее (седьмое) число означает номер подобласти, которой принадлежит конечный элемент.

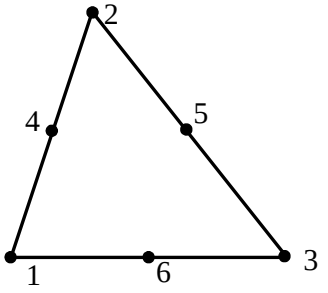


Рис. 10. Расположение узлов на треугольном конечном элементе с квадратичными базисными функциями

Для сетки, изображённой на рис. 9, массив треугольников примет вид:

```
1 3 13 2 9  8  1        3 15 13 10 14 9  1
3 5 15 4 11 10 1        5 7  15 6  12 11 2
```

Краевые условия второго рода для треугольных элементов второго порядка можно хранить в массиве четвёрками целых чисел, первые два из которых – номера вершин (граничных узлов) ребра, третье – номер центрального узла ребра, четвёртое – номер формулы, по которой должно вычисляться значение параметра θ на рассматриваемом ребре. Для сетки, изображённой на рис. 9, этот массив с информацией о краевых условиях второго рода имеет вид:

```
1 3 2 1     3 5 4 1     5 7 6 1     13 15 14 2
```

Информацию о краевом условии первого рода, как и для элементов первого порядка, можно хранить парами целых чисел (первое число – номер узла, второе – номер формулы для вычисления u_g). Для сетки, изображённой на рис. 9, соответствующий массив примет вид:

```
1 1    8 1    13 1    15 2    12 2    7 2
```

Заметим, что чаще всего программы построения сеток формируют данные, в которых содержатся только геометрические вершины элементов и ячейки как геометрические объекты (треугольники, прямоугольники, тетраэдры и т.п.), заданные только номерами геометрических вершин и номером подобласти. В этом случае для получения описанных выше структур данных необходимо внести в структуры данных, сформированные программами-построителями сеток, недостающую информацию о номерах всех ненулевых на каждом конечном элементе глобальных базисных функций. Алгоритмы нумерации глобальных базисных функций (или, что то же самое, неизвестных q_i, являющихся весами базисных функций ψ_i в представлении приближённого решения u^h в виде $\sum_i q_i \psi_i$) мы рассмотрим несколько позже.

1.2.7. ПОСТРОЕНИЕ НЕРЕГУЛЯРНЫХ ТРЕУГОЛЬНЫХ СЕТОК

Как уже говорилось выше, сетки с треугольными ячейками обычно используют в тех случаях, когда расчётная область имеет сложные границы. Кроме того, треугольные сетки позволяют довольно эффективно делать локальные дробления в отдельных местах расчётной области. Это даёт возможность уменьшать погрешность аппроксимации, не вводя в сетку так называемые «лишние» узлы, в результате чего могут быть значительно сэкономлены вычислительные ресурсы.

Существует очень много различных подходов к построению треугольных сеток. Простейшие из них основаны на формировании ячеек треугольной сетки из прямоугольных или четырёхугольных ячеек регулярной сетки, набрасываемой на расчётную область. Треугольные конечные элементы получаются разбиением каждой прямоугольной или четырёхугольной ячейки на два треугольника. При этом, естественно, не попадающие в расчётную область ячейки и узлы регулярной сетки сразу же игнорируются (из таких ячеек не формируются ячейки треугольной сетки, а не попадающие в расчётную область узлы не заносятся в список узлов).

Возможны различные модификации таких подходов, когда, например, регулярная прямоугольная сетка набрасывается на расчётную область с криволинейными или наклонными границами и вершины прямоугольников перед разбиением их на треугольники могут перемещаться на внутренние и внешние границы расчётной области, пересекающие этот прямоугольник. При перемещении вершин из прямоугольников могут получаться не только выпуклые, но и невыпуклые четы-

рёхугольники – для разбиения на треугольные ячейки это не так уж принципиально.

Однако такие подходы хотя и позволяют учитывать довольно сложную геометрию расчётной области, но могут порождать довольно большое число «лишних» узлов (из-за регулярности первоначальной сетки) и поэтому не очень экономичны.

При сравнении различных методов построения сеток самым правильным критерием качества сетки является соотношение между погрешностью решения задачи, полученного с её использованием, и требуемыми для вычисления решения затратами (т.е. оптимальная сетка должна обеспечивать либо минимальную погрешность решения при равных вычислительных затратах, либо минимум вычислительных затрат на получение решения требуемой точности). Однако на практике использовать такой критерий можно лишь для сравнения качества уже построенных сеток, да и то при решении далеко не каждой (а чаще всего модельной) задачи, поэтому для контроля качества сетки в процессе её построения нужны другие критерии.

При сравнении качества треугольных сеток (обычно их называют **триангуляциями**) с одним и тем же набором узлов довольно часто используют следующий критерий: одна сетка считается лучше другой, если значение самого острого угла среди всех треугольников у неё больше (поскольку именно слишком острый угол треугольника может существенно ухудшить свойства конечноэлементной СЛАУ). Известно, что по этому критерию для заданного набора узлов наилучшей является *триангуляция Делоне*.

Триангуляцией Делоне называется такая триангуляция, в которой для любого треугольника внутри описанной вокруг него окружности не содержится ни одного узла сетки. Для построения триангуляции Делоне по заданному набору узлов можно использовать, например, широко известный алгоритм Ватсона. Основная суть этого алгоритма состоит в следующем. В уже существующую триангуляцию (начиная с одного треугольника, охватывающего всю расчётную область) последовательно (по одной) добавляются новые точки с удалением всех треугольников, для которых новая точка находится внутри описанной вокруг них окружности. Вместо удалённых из триангуляции треугольников строятся новые путём соединения вставляемой точки со сторонами многоугольника, полученного после удаления из триангуляции ненужных треугольников.

Известно, что время работы алгоритма Ватсона пропорционально N^2 – квадрату числа узлов. Поэтому много работ посвящено его ускорениям, которые основаны на хранении дополнительной информации, ускоряющей поиск треугольников. Однако даже с ускорениями трудоёмкость соответствующих алгоритмов относительно числа узлов N по числу операций остаётся на уровне $N \ln N$.

Особо следует отметить, что при решении многих задач триангуляция Делоне не является наилучшей *конечноэлементной сеткой* для аппроксимации решения. Например, в тех случаях, когда решение очень мало меняется по одной пространственной координате и резко меняется по другой, для улучшения *аппроксимации* имеет смысл использовать вытянутые треугольники, и применение методов, требующих максимизации острых углов треугольников в процессе построения триангуляции, может не только не улучшить, но и *ухудшить* точность получаемого конечноэлементного решения. Кроме того, триангуляция Делоне по её определению не учитывает наличия в области внутренних границ (т.е. границ подобластей, в которых по-разному могут определяться параметры решаемой задачи), и поэтому при построении триангуляции в области с внутренними границами методы, основанные на получении триангуляции Делоне, нередко оказываются очень неэффективными.

И совсем малоплодотворными представляются нам попытки применить используемые при построении триангуляции Делоне принципы для генерации *тетраэдральных* сеток, так как в трёхмерном случае заложенные в её основу принципы не гарантируют даже построения невырожденных (т.е. имеющих ненулевой объём) тетраэдров. Это происходит по следующей причине. В трёхмерном случае для построения каждого тетраэдра фактически выбираются такие 4 узла, которые лежат на одной сфере и для которых внутрь этой сферы не попадает ни одного другого узла. При этом триангуляция Делоне *никак не контролирует ситуацию попадания четырёх вершин тетраэдра на одну плоскость* (или их близость к одной плоскости), что и может приводить к *вырожденности* (или *почти вырожденности*) получаемых тетраэдров.

Существуют гораздо более эффективные подходы, в которых триангуляция строится на основе алгоритмов непосредственной локальной оптимизации сетки по какому-либо критерию. Дополнительным преимуществом этих методов является то, что генерацию узлов в них можно проводить непосредственно при построении сетки. По такому принципу работает большинство алгоритмов, строящих сетки *фронтальным методом*.

Довольно часто при использовании фронтальных методов расчётная область задаётся в виде набора элементарных подобластей, сетка в которых строится отдельно с автоматической стыковкой по их границам путём задания одного и того же разбиения каждой из внутренних границ для смежных (по этой внутренней границе) элементарных подобластей. При этом разбиения границ элементарных подобластей обычно задаются как исходные данные для фронтального метода.

Продемонстрируем использование фронтального метода для генерации сетки в выпуклом четырёхугольнике, на двух противоположных сторонах которого задано одинаковое число узлов.

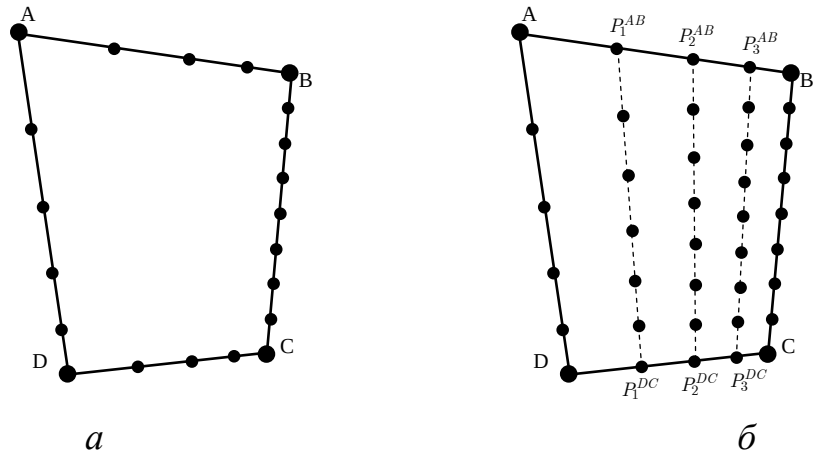
Пусть задан выпуклый четырёхугольник с вершинами A, B, C и D, на его сторонах AB и DC задано по m внутренних узлов, на стороне AD – n_{AD} узлов, на стороне BC – n_{BC} узлов. Пример такого четырёхугольника приведён на рис. 11,*а*, где $m = 3$, $n_{AD} = 4$, $n_{BC} = 7$.

Чтобы построить сетку, разобьём сначала четырёхугольник $ABCD$ на $m+1$ полосу непересекающимися отрезками $P_i^{AB}P_i^{DC}$, соединяющими узлы P_i^{AB} и P_i^{DC} на противоположных сторонах AB и DC, где задано одинаковое количество (по m) узлов (см. рис. 11,*б*).

Количество узлов n_i на i-м отрезке $P_i^{AB}P_i^{DC}$ ($i = 1 \ldots m$) будем вычислять по формуле

$$n_i = \left[\left(1 - \frac{i}{m+1}\right) n_{AD} + \frac{i}{m+1} n_{BC}\right], \qquad (1.5)$$

где $\lfloor a \rfloor$ – целая часть числа a (т.е. n_i меняется почти линейно от значения n_{AD} до значения n_{BC} при движении от стороны AD к стороне BC, см. рис. 11,*б*).



а *б*

Рис. 11. Пример генерации узлов сетки на макрочетырёхугольник

Если на сторонах AD и BC заданы коэффициенты растяжения-сжатия шага q_{AD} и q_{BC} соответственно, то коэффициент q_i растяжения-сжатия шага на i-м отрезке $P_i^{AB}P_i^{DC}$ (естественно, в направлении от стороны AB к стороне DC) будем вычислять, считая его изменение линейным при движении от стороны AD к стороне BC по аналогичной формуле

$$q_i = \left(1 - \frac{i}{m+1}\right) q_{AD} + \frac{i}{m+1} q_{BC}\,. \qquad (1.6)$$

В приведённом на рис. 11 примере заданы следующие коэффициенты растяжения-сжатия: q_{AB}=0.75, q_{BC}=1, q_{DC}=0.75 и q_{AD}=0.83. На рис. 11,*б* показаны внутренние узлы, сгенерированные по формулам (1.5)–(1.6) – эти узлы сгенерированы на отрезках $P_i^{AB}P_i^{DC}$ (обозначенных пунктирными линиями).

Треугольную сетку в четырёхугольнике $ABCD$ (с равным числом узлов на двух его противоположных сторонах AB и DC) можно построить, заполняя треугольниками полосы, на которые этот четырёхугольник был разделён отрезками $P_i^{AB}P_i^{DC}$. Это полосы между стороной AD и отрезком $P_1^{AB}P_1^{DC}$, между отрезками $P_i^{AB}P_i^{DC}$ и $P_{i+1}^{AB}P_{i+1}^{DC}$ $\left(i = 1 \ldots m-1\right)$ и между отрезком $P_m^{AB}P_m^{DC}$ и стороной BC (см. рис. 11,*б*).

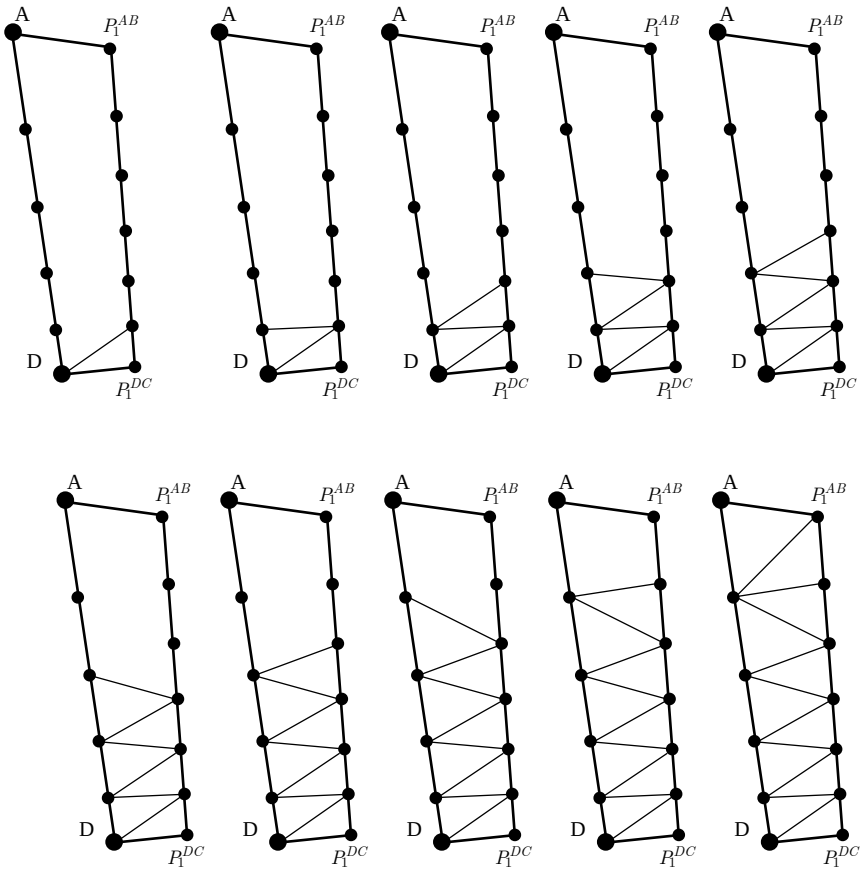


Рис. 12. Процесс заполнения треугольниками одной полосы

Рассмотрим алгоритм заполнения треугольниками одной полосы. Этот алгоритм проиллюстрирован на рис. 12 и заключается в следующем. Сначала выбирается любое из двух рёбер, являющихся короткими (т.е. без внутренних узлов) границами полосы. Первый треугольник строится присоединением к вершинам этого ребра одного из двух соседних с ним узлов. При этом выбирается такой узел, чтобы *была минимальной длина нового ребра*, соединяющего этот узел с лежащим на противоположной (длинной) стороне полосы узлом старого ребра (являющегося границей полосы).

Каждый следующий треугольник строится по тому же принципу: к вершинам ребра, соединяющего два узла на противоположных (длинных) сторонах полосы и полученного при построении предыдущего треугольника, добавляется один из двух соседних с ними узлов такой, что длина нового ребра является минимальной (см. рис. 12). Очевидно, что на последнем шаге этого алгоритма получается сразу два треугольника.

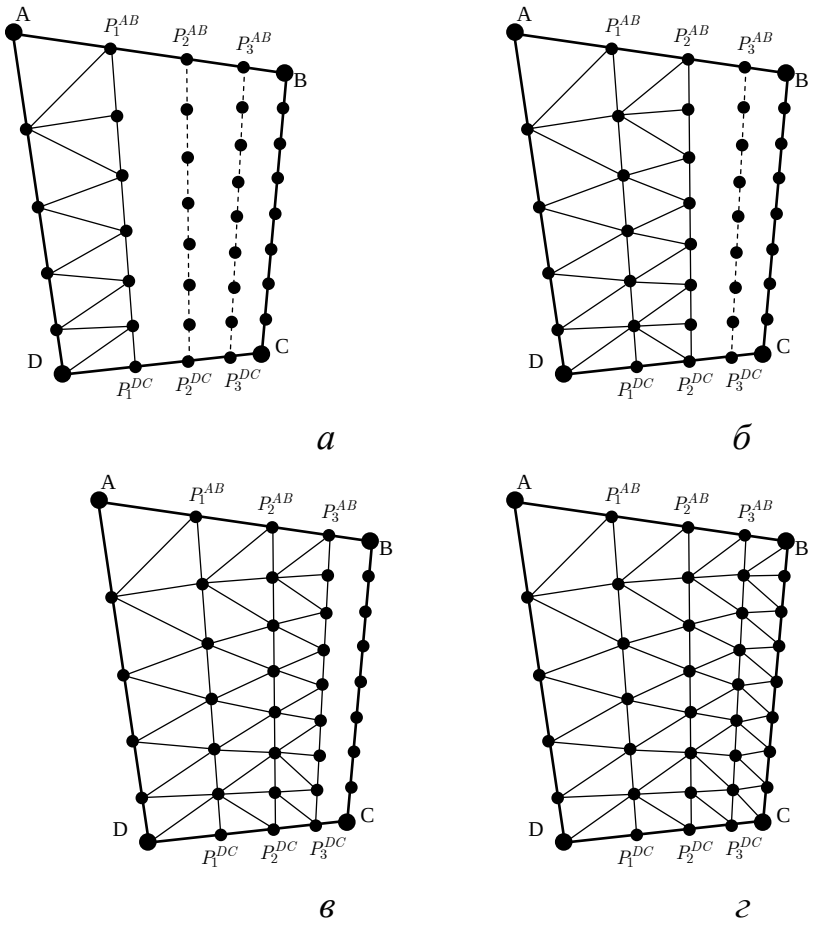
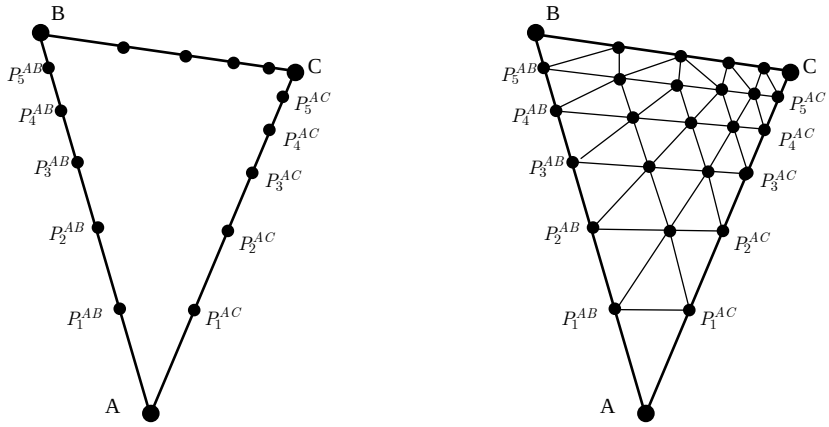


Рис. 13. Процесс построения сетки на четырёхугольнике с равным числом узлов на противоположенных сторонах

Процесс построения сетки в четырёхугольнике с равным числом узлов на двух его противоположных сторонах проиллюстрирован на рис. 13. Один шаг этого процесса вполне можно проинтерпретировать как генерацию слоя новых треугольников *на фронте* уже построенных треугольников (с перемещением фронта от стороны AD через отрезки $P_i^{AB}P_i^{DC}$ к стороне BC). Поэтому рассмотренную процедуру генерации сетки в четырёхугольнике можно отнести к классу *фронтальных* методов.

Рассмотренный метод можно достаточно просто адаптировать для генерации треугольной сетки в фигуре, представляющей собой треугольник (макротреугольник) с заданными на его сторонах узлами, если количество узлов хотя бы на двух сторонах этого макротреугольника совпадает. Будем считать, что равное количество узлов расположено на сторонах AB и AC макротреугольника ABC (см. рис. 14,*а*). Для построения сетки в этом макротреугольнике достаточно сначала отсечь ребром $P_1^{AB}P_1^{AC}$ треугольник $P_1^{AB}P_1^{AC}A$, а затем использовать рассмотренный

выше алгоритм построения треугольной сетки в четырёхугольнике $P_1^{AB}BCP_1^{AC}$ с равным числом узлов на сторонах $P_1^{AB}B$ и $P_1^{AC}C$. Пример построения такой сетки проиллюстрирован на рис. 14,*б*.



а *б*

Рис. 14. Процесс построения сетки на треугольнике
с равным числом узлов на противоположенных сторонах

С некоторыми доработками (касающимися в основном способов построения линий между узлами P_i^{AB} и P_i^{DC} для четырёхугольника и узлами P_i^{AB} и P_i^{AC} для треугольника) рассмотренные методы можно применять и для построения треугольных сеток внутри *псевдочетырёхугольников* и *псевдотреугольников*, стороны которых не являются отрезками прямых. Соответствующие примеры приведены на рис. 15.

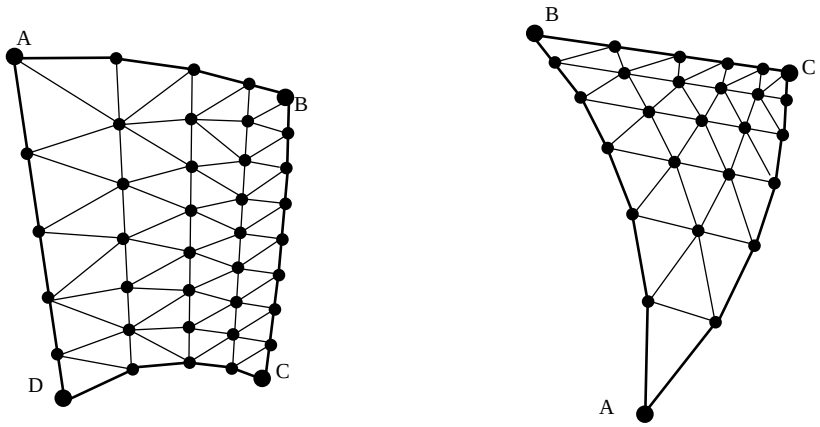
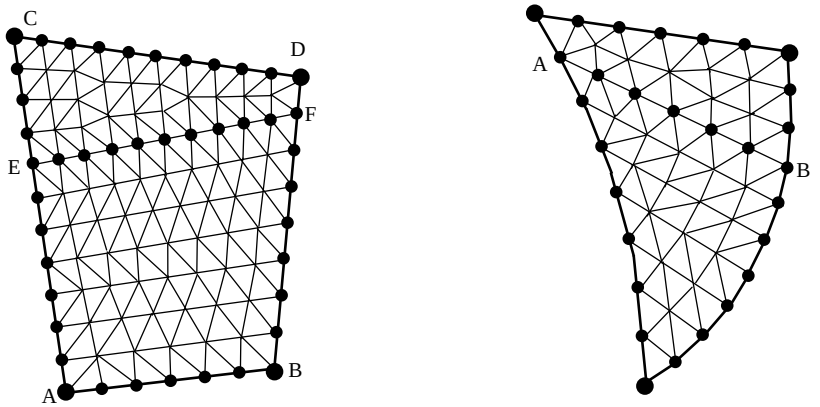


Рис. 15. Построение треугольных сеток внутри
псевдочетырёхугольников и псевдотреугольников



а *б*

Рис. 16. Построение треугольных сеток внутри фигур
с различным числом узлов на всех их сторонах

На основе рассмотренных алгоритмов построения сеток в четырёхугольниках (и псевдочетырёхугольниках) с *равным* количеством узлов на двух противоположных сторонах можно построить довольно простой алгоритм генерации треугольной сетки и в четырёхугольнике с *неодинаковым* числом узлов на его противоположных сторонах. Для этого достаточно разбить исходный четырёхугольник на два новых четырёхугольника, у которых уже есть по две стороны с равным количеством узлов. Это можно сделать, если провести через две противоположные стороны исходного четырёхугольника одну линию так, чтобы в одном из новых четырёхугольников (на которые проводимая линия разбивает исходный четырёхугольник) появилось две стороны с равным числом узлом (чего, очевидно, всегда можно добиться на двух сторонах одного из новых четырёхугольников, смежных с линией, разбивающей исходный четырёхугольник). На самой же линии (разби-

вающей исходный четырёхугольник на два новых) нужно сгенерировать столько же узлов, сколько их расположено на противоположной этой линии стороне второго нового четырёхугольника. Соответствующий пример приведён на рис. 16,*а* (ABCD – исходный четырёхугольник, ABEF и EFCD – новые четырёхугольники с *равным* числом узлов на двух противоположных сторонах).

Аналогичный подход можно применить и для построения треугольных сеток внутри фигур в виде треугольников или псевдотреугольников с различным числом узлов на всех их сторонах – соответствующий пример приведён на рис. 16,*б* (для построения сетки добавлена линия AB).

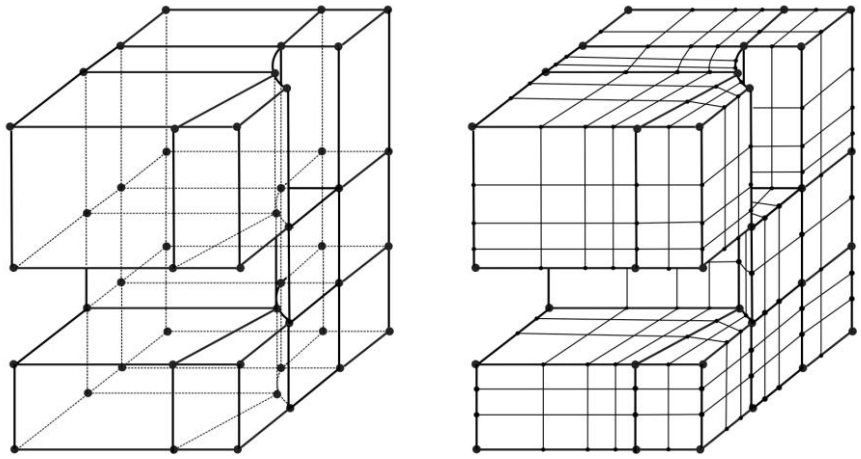
1.3. СТРУКТУРЫ ДАННЫХ ДЛЯ ОПИСАНИЯ ТРЁХМЕРНЫХ ЗАДАЧ. ТРЁХМЕРНЫЕ СЕТКИ

1.3.1. ОПИСАНИЕ ТРЁХМЕРНЫХ ОБЛАСТЕЙ И ПРИНЦИПЫ ПОСТРОЕНИЯ ТРЁХМЕРНЫХ СЕТОК С ЯЧЕЙКАМИ В ВИДЕ ПАРАЛЛЕЛЕПИПЕДОВ И ШЕСТИГРАННИКОВ

Как и в двумерном случае, способы описания трёхмерных областей довольно тесно связаны с методами построения конечноэлементных сеток. Если расчётные области содержат только подобласти в виде прямоугольных параллелепипедов и для них предполагается использовать соответствующие прямоугольные сетки, то применяются структуры данных, аналогичные описанным в разделе 1.2.1. Фактически в разделе 1.2.3 уже было сказано, как распространить алгоритм построения регулярной прямоугольной сетки на трёхмерный случай (т.е. для построения сетки с ячейками в виде прямоугольных параллелепипедов). При этом описание расчётной области кроме массивов X^W и Y^W должно содержать ещё и массив Z^W, а наборы чисел, описывающие каждую элементарную подобласть W_l (в виде параллелепипеда), должны содержать не по пять целых чисел, а по семь. Аналогично и в файл описания сетки к массивам координатных линий сетки X и Y должен быть добавлен массив Z (и при работе с сеткой к массивам IX^W и IY^W должен быть добавлен массив IZ^W). Сами же алгоритмы построения трёхмерных параллелепипеидальных сеток (как регулярных, так и нерегулярных) в целом ничем не отличаются от алгоритмов построения двумерных прямоугольных сеток, рассмотренных в разделе 1.2.3 (и в конце раздела 1.2.5 в случае генерации нерегулярных сеток).

Если расчётная область имеет не параллельные координатным плоскостям границы (в том числе в виде изогнутых поверхностей) и для её дискретизации предполагается использовать сетки с *шестигранными ячейками*, то можно использовать структуры данных, аналогичные описанным в разделе 1.2.5 для построения сеток с четырёхугольными ячейками. Рассмотрим чуть более детально возможные структуры данных для описания расчётных областей и хранения сеток с шестигранными ячейками.

Простейшую структуру данных можно построить в случае, когда все внутренние и внешние границы расчётной области либо перпендикулярны одной из осей (назовём её осью z), либо параллельны ей. В этом случае в качестве базовой можно взять структуру данных, которая была рассмотрена нами при описании двумерных областей, состоящих из четырёхугольных подобластей (см. с. 17–19 или с. 24), и добавить к ней одномерный массив Z^W, определяющий границы элементарных подобластей по оси z. Например, на базе двумерной расчётной области, изображённой на рис. 7,*а*, трёхмерная расчётная область, состоящая из пяти подобластей и изображённая на рис. 17,*а*, может быть описана с помощью следующей структуры данных (сравните со структурой данных, приведённой на с. 24).



а *б*

Рис. 17. Трёхмерная расчётная область (*а*) и шестигранная сетка в ней (*б*)

```
3 4
0. 0.        5.   0.        7. 0.
0. 2.59.     5.59 2.59      7. 2.
0. 4.        5.   4.        7. 4.
0. 6.        5.   6.        7. 6.
```

```
4
0.    3.    6.    11.
5
1  1  3  1  2  1  2
2  1  3  1  2  3  4
3  1  2  2  3  1  4
4  1  3  3  4  1  4
5  2  3  2  3  1  3
2
-1  2  2  7.  4.
 1  2  2  7.  4.
```

Первые пять строк определяют *опорные узлы базовой плоскости* (являющейся фактически некоторым обобщённым сечением расчётной области плоскостью $z = \mathrm{const}$) точно так же, как задавались узлы при описании четырёхугольной области (см. с. 24). Два целых числа в первой строке задают значения N_x^W и N_y^W, а в следующих $N_y^W = 4$ строках перечислены парами x- и y-координаты опорных узлов (определяющих «горизонтальные» и «вертикальные» ломаные) последовательно по «горизонтальным» ломаным (по $N_x^W = 3$ пар в каждой строке).

Далее (в нашем примере это шестая строка файла описания области) задаётся значение N_z^W, определяющее количество плоскостей $z = \mathrm{const}$, необходимых для задания подобластей расчётной области. В следующей строке задаются значения z-координат этих плоскостей (т.е. содержимое массива Z^W).

После этого (в нашем примере с восьмой по двенадцатую строках) задаётся информация об элементарных подобластях – сначала их количество L, а затем L семёрок чисел, определяющих каждую подобласть W_k. Первое число каждой семёрки задаёт номер формул, определяющих параметры краевой задачи в W_k. Следующие два числа определяют номера «вертикальных» ломаных (которые фактически определяют поверхности, ограничивающие подобласть W_k слева и справа, эти границы можно интерпретировать как границы «по x»), четвёртое и пятое число – номера «вертикальных» (в базовой плоскости) ломаных (границы «по y»), и, наконец, шестое и седьмое числа определяют границы подобласти W_k по z.

В самом конце файла задаётся информация о криволинейных участках «горизонтальных» и «вертикальных» ломаных в формате, который был описан при описании двумерных областей на с. 24. В нашем примере это три последние строки. Их содержимое означает следующее. Криволинейными являются два участка ломаных (поскольку первое число в этом блоке информации равно двум). Первый участок является участком «вертикальной» ломаной (поскольку имеет признак -1), расположен он на второй ломаной и является её вторым участком. Этот участок является частью окружности с центром в точке $x = 7$, $y = 4$. Второй участок – это участок «горизонтальной» ломаной (поскольку имеет признак 1). Он расположен на второй «горизонтальной» ломаной, является её вторым участком и представляет собой дугу той же окружности с центром в точке $x = 7$, $y = 4$.

Для задания трёхмерной сетки с ячейками в виде шестигранников можно использовать те же принципы, которые были описаны нами для сеток с ячейками в виде прямоугольников и четырёхугольников (см. с. 10). При этом к данным, описывающим разбиение «горизонтальных» и «вертикальных» ломаных на базовой плоскости, добавляются данные, описывающие разбиение каждого интервала по z (также в виде количества подынтервалов и коэффициентов растяжения-сжатия). Для сетки, изображённой на рис. 17,*б* (и построенной в рассмотренной выше трёхмерной области), файл её описания выглядит следующим образом:

```
4 -1.5      2 1.
3 -1.4      3 1.25      2 1.5
3 -1.2      2 1.        4 1.5
```

Теперь рассмотрим более общий случай, когда границы «по z» могут не быть плоскими и перпендикулярными оси z, а так же, как и границы «по x» и «по y» (которые в предыдущем случае определялись соответственно через «вертикальные» и «горизонтальные» ломаные базовой плоскости), могут являться некоторыми поверхностями.

По-прежнему расчётную область будем описывать с помощью *опорных узлов*. Однако теперь эти узлы будем задавать не на базовой плоскости, а сразу в трёхмерном пространстве (*тремя* их координатами). Перечислять опорные узлы будем, как и прежде, в строго установленном порядке, с помощью которого фактически будут определяться и все границы (внутренние и внешние) расчётной области «по x», «по y» и «по z». При этом сначала перечисляются (последователь-

но) N_x^W узлов первой «горизонтальной» ломаной первой поверхности «по z», затем N_x^W узлов второй «горизонтальной» ломаной и далее вплоть до N_y^W-й «горизонтальной» ломаной первой поверхности «по z». После информации обо всех опорных узлах первой поверхности «по z» перечисляются все «горизонтальные» ломаные второй поверхности «по z» (в том же порядке), третьей и далее вплоть до N_z^W-й поверхности «по z». Таким образом задаются все опорные узлы (их количество $N_x^W \times N_y^W \times N_z^W$).

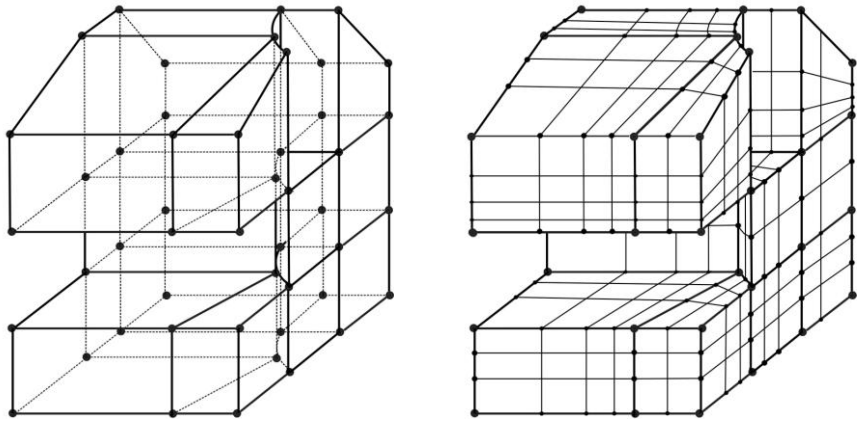
Информация об элементарных подобластях задаётся абсолютно так же, как и в предыдущем варианте (когда опорные узлы задавались на базовой плоскости, а получаемые из неё границы были строго перпендикулярны оси z). Только здесь шестое и седьмое числа каждой семёрки чисел, описывающих элементарную подобласть, будут определять номера не плоскостей, а поверхностей «по z».

Приведём пример описания расчётной области с неплоскими «по z» границами (обратим внимание, что теперь в первой строке задаётся три числа N_x^W, N_y^W и N_z^W, а за ними следует уже $N_x^W \cdot N_y^W \cdot N_z^W$ троек вещественных чисел координат всех опорных узлов):

```
3 4 4
0. 0. 0.      5.   0.   0.      7. 0. 0.
0. 2. 0.      5.59 2.59 0.      7. 2. 0.
0. 4. 0.      5.   4.   0.      7. 4. 0.
0. 6. 0.      5.   6.   0.      7. 6. 0.
0. 0. 3.      5.   0.   3.      7. 0. 3.
0. 2. 3.      5.59 2.59 3.      7. 2. 3.
0. 4. 3.      5.   4.   3.      7. 4. 3.
0. 6. 3.      5.   6.   3.      7. 6. 3.
0. 0. 6.      5.   0.   6.      7. 0. 6.
0. 2. 6.      5.59 2.59 6.      7. 2. 6.
0. 4. 6.      5.   4.   6.      7. 4. 6.
0. 6. 6.      5.   6.   6.      7. 6. 6.
0. 0. 9.      5.   0.   9.      7. 0. 9.
0. 2. 11.     5.59 2.59 11.     7. 2. 11.
0. 4. 11.     5.   4.   11.     7. 4. 11.
0. 6. 8.      5.   6.   8.      7. 6. 8.
```

```
5
1  1  3  1  2  1  2
2  1  3  1  2  3  4
3  1  2  2  3  1  4
4  1  3  3  4  1  4
5  2  3  2  3  1  3
2
-1  2  2  7.  4.
 1  2  2  7.  4.
```

Вид этой расчётной области показан на рис. 18,*а*.



а *б*

Рис. 18. Трёхмерная расчётная область (*а*) и шестигранная сетка в ней (*б*)

Структура файла задания сетки для этого варианта задания расчётной области ничем не отличается от структуры файла задания сетки для предыдущего варианта. На рис. 18,*б* изображена сетка, описываемая файлом, приведённым на с. 42.

Не представляет труда модифицировать рассмотренный вариант описания расчётной области так, чтобы можно было задавать искривлённые участки не только ломаных «вдоль x» и «вдоль y» (которые в предыдущих вариантах мы называли «горизонтальными» и «вертикальными» ломаными), но и «вдоль z». Для этого нужно ввести признак искривления линии «вдоль z». Напомним, что признак искривления линии «вдоль x» (т.е. вдоль «горизонтальной» ломаной) был введён нами ранее как значение 1, а признак искривления линии «вдоль y» («вертикальной» ломаной) – как значение -1. Введём признак искривления линии «вдоль z» как значение -10.

Для удобства определения положения искривлённой линии введём понятие номера поверхности «по z». Будем считать, что первые $N_x^W \times N_y^W$ опорных узлов определяют поверхность «по z» с номером 1, следующие $N_x^W \times N_y^W$ опорных узлов (с номерами от $N_x^W N_y^W + 1$ до $2N_x^W N_y^W$) – поверхность «по z» с номером 2 и т.д. Тогда опорный узел, из которого исходит искривлённый участок, удобно определить тремя целыми числами: первое число – это номер поверхности «по z», второе и третье числа – соответственно номера «горизонтальной» и «вертикальной» ломаных (на этой поверхности), на пересечении которых лежит определяемый нами узел.

Заметим, что раньше мы определяли местоположение криволинейных участков чуть иначе – заданием номера кривой и номера участка этой кривой. В принципе, задание искривлённых участков «вдоль x» (т.е. горизонтальных ломаных) осталось практически тем же, поскольку номер участка фактически совпадает с номером вертикальной ломаной и мы к этой информации добавили только номер поверхности «по z». Искривлённые же участки «вдоль y» теперь задаются чуть иначе (раньше при задании таких участков первым задавался номер «вертикальной» линии, а после него – номер «участка», фактически являющегося номером «горизонтальной» линии, на которой лежал опорный узел, т.е. здесь мы фактически поменяли эти номера местами). Это сделано для того, чтобы единообразно определять позицию опорного узла для искривлённых участков по любому направлению (как по старым двум направлениям – «вдоль x» и «вдоль y», так и по новому для этой структуры данных направлению – «вдоль z»).

Рассмотрим пример задания расчётной области с изогнутыми границами не только «по x» и «по y», но и «по z». Пусть файл описания расчётной области имеет вид:

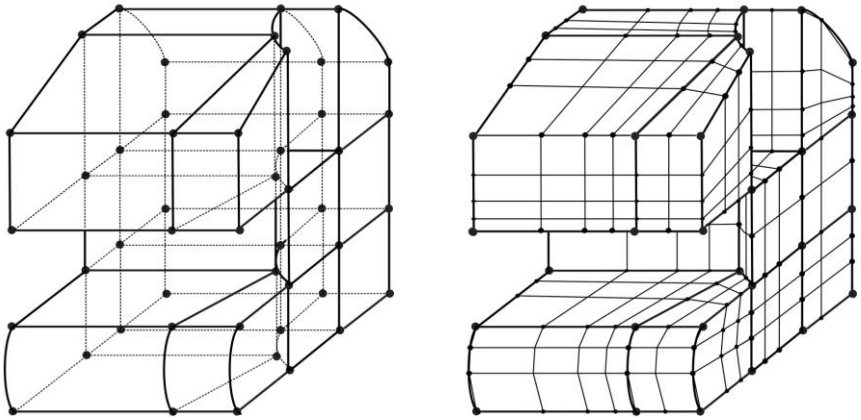
```
3 4 4
0. 0. 0.      5.   0.   0.      7. 0. 0.
0. 2. 0.      5.59 2.59 0.      7. 2. 0.
0. 4. 0.      5.   4.   0.      7. 4. 0.
0. 6. 0.      5.   6.   0.      7. 6. 0.
0. 0. 3.      5.   0.   3.      7. 0. 3.
0. 2. 3.      5.59 2.59 3.      7. 2. 3.
0. 4. 3.      5.   4.   3.      7. 4. 3.
```

```
0. 6. 3.       5.   6.   3.       7. 6. 3.
0. 0. 6.       5.   0.   6.       7. 0. 6.
0. 2. 6.       5.59 2.59 6.       7. 2. 6.
0. 4. 6.       5.   4.   6.       7. 4. 6.
0. 6. 6.       5.   6.   6.       7. 6. 6.
0. 0. 9.       5.   0.   9.       7. 0. 9.
0. 2. 11.      5.59 2.59 11.      7. 2. 11.
0. 4. 11.      5.   4.   11.      7. 4. 11.
0. 6. 8.       5.   6.   8.       7. 6. 8.
5
1 1 3 1 2 1 2
2 1 3 1 2 3 4
3 1 2 2 3 1 4
4 1 3 3 4 1 4
5 2 3 2 3 1 3
14
1   1 2 2 7. 4. 0.
1   2 2 2 7. 4. 3.
1   3 2 2 7. 4. 6.
1   4 2 2 7. 4. 11.
-1  1 2 2 7. 4. 0.
-1  2 2 2 7. 4. 3.
-1  3 2 2 7. 4. 6.
-1  4 2 2 7. 4. 11.
-1  4 3 1 0. 2. 7.5
-1  4 3 2 5. 2. 7.5
-1  4 3 3 7. 2. 7.5
-10 1 1 1 0. 3. 1.5
-10 1 1 2 5. 3. 1.5
-10 1 1 3 7. 3. 1.5
```

Вид этой расчётной области изображён на рис. 19,*а*. На рис. 19,*б* показана сетка, описываемая тем же файлом (см. с. 42).

Итак, рассмотренные способы описания расчётных областей и сеток с шестигранными ячейками позволяют решать трёхмерные задачи в областях с достаточно сложными границами. Таким образом, использование сеток с шестигранными

ячейками является разумным компромиссом, позволяющим наряду с широкими возможностями описания сложной геометрии расчётной области сохранить простоту и удобство её дискретизации, особенно если при этом реализована возможность согласования элементов на несогласованных ячейках, например, с использованием технологии **T**-преобразования, которая будет описана нами ниже, в третьем разделе.



а *б*

Рис. 19. Трёхмерная расчётная область (*а*) и шестигранная сетка в ней (*б*)

Однако в некоторых ситуациях даже шестигранные ячейки могут оказаться неудобными для дискретизации расчётной области из-за очень сложной геометрии её границ. В таких случаях в качестве ячеек дискретизации можно использовать тетраэдры. Нередко применяют и прямые призмы с треугольным основанием, если все изогнутые поверхности являются цилиндрическими с параллельными друг другу образующими, или используют комбинированные сетки в виде тетраэдров и призм.

Ниже мы рассмотрим один из наиболее часто используемых способов построения сеток с ячейками в виде призм и тетраэдров – метод тиражирования сечений.

1.3.2. МЕТОД ТИРАЖИРОВАНИЯ СЕЧЕНИЙ

В методе тиражирования сечений описание геометрии и конечноэлементной сетки практически полностью совмещено и сетка строится по *сечениям*. Под **сечением** понимается пересечение некоторой (не обязательно плоской) поверхности с расчётной областью. Трёхмерная сетка строится заполнением конечными элементами пространства между соседними сечениями, на которых построена сетка из двумерных фигур, чаще всего треугольников. Принципиально возможно

создание алгоритма, для которого треугольные сетки на соседних сечениях строятся совершенно независимо, однако невысокая скорость работы и сложность программной реализации такого алгоритма практически полностью перекрывают все его преимущества. Поэтому большинство реализаций метода тиражирования сечений требует подобия треугольных сеток и неизменности количества узлов на всех тиражируемых сечениях, за исключением случаев, когда из триангуляции выбрасываются узлы и треугольники, попавшие в полости, не входящие в расчётную область. При этом двумерная треугольная сетка, на основе которой строится трёхмерная сетка, обычно называется **базовым сечением** или **базовой плоскостью.**

Пространство между двумя сечениями с подобными треугольными сетками (т.е. допускающими соответствие каждому треугольнику одной сетки *одного и только одного* треугольника другой) заполняется сначала треугольными призмами, каждая из которых затем разбивается на три тетраэдра (с учётом согласованности разбиения соседних призм по смежным граням, более подробно это будет рассмотрено ниже). Если все призмы, построенные между двумя сечениями с треугольными сетками, оказываются прямыми, то в качестве ячеек сетки можно оставить эти призмы (не разбивая их на тетраэдры).

Кроме того, возможны достаточно простые реализации метода тиражирования сечений, когда допустимо тиражирование не всего сечения, а лишь его части.

Тиражируемые сечения в рассматриваемом подходе могут быть двух типов – *основные* и *промежуточные*. **Основными** называются сечения, которые необходимы для описания геометрии области.

Для задания основного сечения удобно использовать преобразование *базового сечения* или уже созданных других основных сечений с помощью специальных средств перемещения узлов без изменения топологии сетки (что позволяет изменять границы подобластей на сечении и делать его некоторые искривления), а также средств изменения точек привязки информации о параметрах решаемого уравнения.

Промежуточные сечения – это сечения, которые генерируются автоматически между двумя основными сечениями и обеспечивают нужную подробность сетки вдоль траектории тиражирования.

Таким образом, *процедура построения трёхмерной сетки* после задания базового сечения состоит из следующих этапов:

- описание поверхностей основных сечений и задание на них возможных смещений узлов сетки;
- задание положения основных сечений в пространстве;
- задание количества промежуточных сечений и некоторых параметров, определяющих положение этих сечений между основными сечениями (например, это могут быть параметры кривых, по которым должны генерироваться узлы между сечениями);
- непосредственное формирование конечных элементов.

1.3.3. ПОСТРОЕНИЕ ТЕТРАЭДРАЛЬНЫХ СЕТОК МЕТОДОМ ТИРАЖИРОВАНИЯ СЕЧЕНИЙ

Как уже говорилось, при использовании метода тиражируемых сечений трёхмерная конечноэлементная сетка формируется из специальной двумерной сетки, в качестве которой удобно использовать плоскую триангуляцию, заданную на так называемой *базовой плоскости*.

На базовой плоскости задаётся триангуляция с учётом того, что из неё нужно будет формировать не только ячейки сетки, но и все границы (внешние и внутренние) расчётной области. Эти границы получаются либо как некоторые участки создаваемых в процессе тиражирования сечений, либо в результате тиражирования определённых линий, заданных на базовой плоскости специально для формирования таких «боковых» границ.

Также на базовой плоскости обычно задаются сгущения или разрежения узлов в тех местах, которые соответствуют местам сгущений или разрежений узлов в трёхмерной сетке.

Рассмотрим построение согласованной тетраэдральной сетки между двумя соседними сечениями (для простоты мы не будем здесь различать основные и промежуточные сечения), полученными тиражированием одной базовой плоскости.

Для построения тетраэдральной сетки между двумя сечениями сначала строится сетка из треугольных призм с основаниями, полученными из одного и того же треугольника базовой плоскости. Если призма не вырождена (т.е. все её вершины различны), то каждая четырёхугольная грань призмы может быть разбита на два треугольника. При этом из двух возможных диагоналей этой четырёхугольной грани для её разбиения выбирается та, которая короче. Если диагонали близки по длине (т.е. грань почти прямоугольная), то строится диагональ из вер-

шины с максимальным (из четырёх вершин грани) номером. Всего возможно 8 (2^3) вариантов разбиения граней призмы. Эти варианты показаны на рис. 20 и 21. Для шести из них призму всегда можно разбить на 3 тетраэдра (см. рис. 20): для варианта на рис. 20,*а*, это тетраэдры 1 2 3 5, 1 3 5 6 и 1 4 5 6; для варианта на рис. 20,*б* – тетраэдры 1 2 3 5, 1 3 4 5 и 3 4 5 6; для варианта на рис. 20,*в* – тетраэдры 1 2 3 6, 1 2 5 6 и 1 4 5 6; для варианта на рис. 20,*г* – тетраэдры 1 2 3 4, 2 3 4 5 и 3 4 5 6; для варианта на рис. 20,*д* – тетраэдры 1 2 3 6, 1 2 4 6 и 2 4 5 6; для варианта на рис. 20,*е* – тетраэдры 1 2 3 4, 2 3 4 6 и 2 4 5 6. В двух оставшихся случаях (см. рис. 21) ставится дополнительный узел внутри призмы и строится 8 тетраэдров.

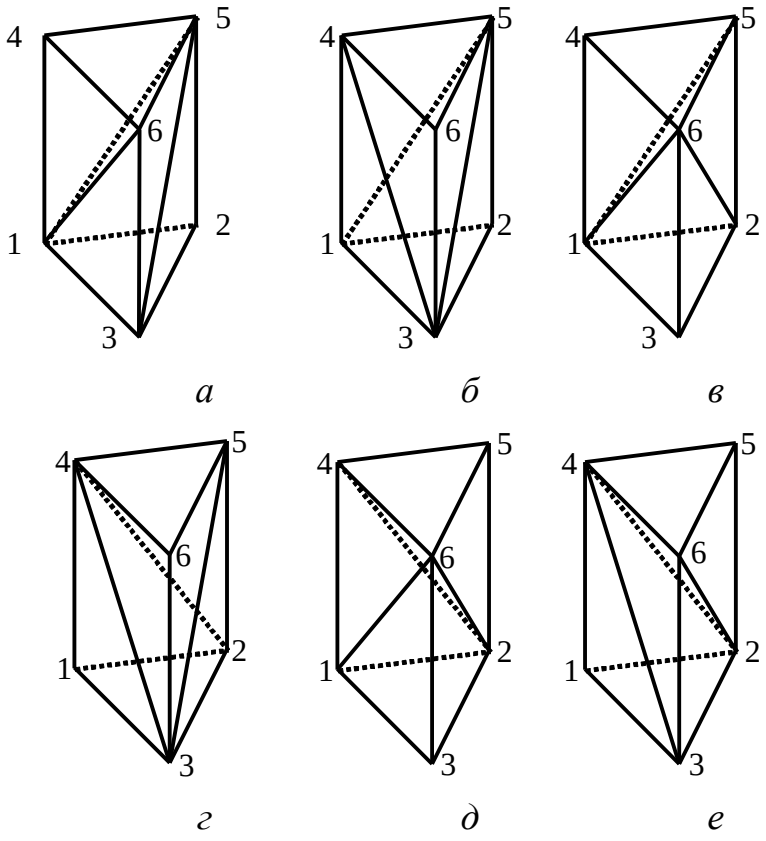


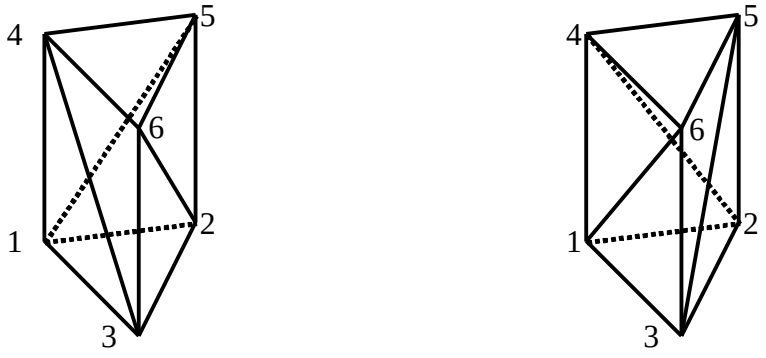
Рис. 20. Разбиение призмы на 3 тетраэдра для всех случаев разбиения граней, когда оно существует

Таким образом, в рассмотренном способе согласованная тетраэдральная сетка строится без сохранения информации о треугольниках на боковых гранях призм. При программной реализации этого алгоритма необходимо, однако, учитывать, что при обработке общей для двух призм грани в тех случаях, когда грань близка к прямоугольной, из-за погрешностей округления может оказаться, что для

одной из призм разбиение произойдет по нумерации узлов, а для другой – по длинам диагоналей. Чтобы этого не произошло, можно создать определяющую способ разбиения грани призмы подпрограмму, координаты вершин в которую передаются в порядке нумерации узлов сетки.

Информация о краевых условиях второго и третьего рода формируется в виде списка треугольников – *трёх номеров вершин* граничного треугольника и *номера формул* для определения параметров соответствующего краевого условия. При этом треугольники, заносимые в списки краевых условий второго и третьего рода, должны быть *согласованы* с тетраэдрами, т.е. треугольники, аппроксимирующие границы S_2 и S_3 с заданными на них краевыми условиями второго и третьего рода, *должны быть гранями ячеек конечноэлементной сетки*.

Рассмотренная процедура разбиения граней призмы позволяет записывать согласованные с конечноэлементной сеткой граничные треугольники независимо от обработки самих конечных элементов, если краевые условия либо приписаны к рёбрам треугольников сечения (тогда формируются по два треугольника от боковых граней призмы), либо заданы на самих треугольниках сечения.



а *б*

Рис. 21. Случаи разбиения граней призмы,
когда разбиений на 3 тетраэдра не существует

Отметим, что в МКЭ задание однородного краевого условия второго рода на некоторой грани не вносит никаких изменений в конечноэлементную СЛАУ, поэтому не имеет никакого смысла сохранять информацию о таких гранях.

2. ОСНОВНЫЕ АЛГОРИТМЫ МКЭ

2.1. СТРУКТУРЫ ДАННЫХ ДЛЯ ХРАНЕНИЯ МАТРИЦ КОНЕЧНОЭЛЕМЕНТНЫХ СЛАУ

2.1.1. ОСОБЕННОСТИ МАТРИЦ КОНЕЧНОЭЛЕМЕНТНЫХ СЛАУ

Важной особенностью матрицы $\mathbf{A}$ конечноэлементной СЛАУ является её сильная **разреженность**, т.е. небольшое (относительно общего числа компонент) число *ненулевых* компонент в каждой строке матрицы. Действительно, ненулевыми в i-й строке матрицы $\mathbf{A}$ являются только компоненты A_{ij} в столбцах с номерами j, для которых базисные функции ψ_j являются ненулевыми на тех же конечных элементах, на которых отлична от нуля базисная функция ψ_i. Поэтому при программной реализации МКЭ для хранения матриц конечноэлементных СЛАУ имеет смысл использовать форматы, позволяющие хранить только ненулевые компоненты матриц или хотя бы только те компоненты, которые могут стать ненулевыми в процессе решения СЛАУ (при использовании прямых методов решения СЛАУ).

Другой особенностью матрицы конечноэлементной СЛАУ, которую можно учитывать при её хранении, является симметричность расположения ненулевых компонент. В самом деле, если в матрице $\mathbf{A}$ ненулевой является компонента A_{ij}, то это означает, что неизвестная i и неизвестная j соответствуют одному конечному элементу (это тот конечный элемент, на котором обе глобальные базисные функции ψ_i и ψ_j отличны от нуля), и тогда компонента A_{ji} также, скорее всего, будет ненулевой. Кроме того, для большинства задач матрица конечноэлементной СЛАУ является симметричной (т.е. $A_{ij} = A_{ji}$), и поэтому хранить можно только нижний (или только верхний) треугольник матрицы.

Мы будем излагать форматы хранения и алгоритмы работы с ними для общего случая, когда матрица является несимметричной. При этом будем считать, что компоненты нижнего треугольника матрицы хранятся в массиве с именем ggl, а компоненты верхнего – в массиве с именем ggu.

Заметим, что формат хранения матрицы СЛАУ может зависеть и от метода решения СЛАУ: если предполагается использовать какой-либо из прямых методов (таких как LL^T- или LU-разложение), то в процессе решения СЛАУ необходима память для размещения *профиля* матрицы.

Под **профилем** понимаются те компоненты матрицы, которые расположены между *первой ненулевой* компонентой каждой строки нижнего треугольника и главной диагональю, и соответственно между *первой ненулевой* компонентой каждого столбца верхнего треугольника и главной диагональю.

При использовании же итерационных методов, в которых основная операция – это умножение матрицы на некоторый вектор, имеет смысл хранить только ненулевые компоненты матрицы.

Поскольку в МКЭ элементы главной диагонали матрицы СЛАУ практически всегда не равны нулю, для их хранения удобно выделить отдельный массив di длиной N (N – размерность СЛАУ). Далее при описании всех форматов мы не будем специально обращать ваше внимание на то, что они различаются только способом хранения *внедиагональных* элементов матрицы, а элементы главной диагонали всегда хранятся отдельно в массиве di.

2.1.2. ПРОФИЛЬНЫЙ ФОРМАТ ХРАНЕНИЯ МАТРИЦЫ СЛАУ

Для хранения матрицы в профильном формате нам кроме массива di потребуется ещё три массива: ggl и ggu – для хранения самих ненулевых элементов нижнего и верхнего треугольников матрицы (без главной диагонали), и ig – массив для вычисления адресов ненулевых элементов. При этом мы будем считать, что ненулевые элементы матрицы $\mathbf{A}$ расположены симметрично (т.е. если $A_{ij} \neq 0$, то и $A_{ji} \neq 0$).

В массиве ggl будем хранить все элементы *профиля нижнего треугольника* матрицы по строкам (последовательно друг за другом), а в массиве ggu – все элементы профиля верхнего треугольника матрицы по столбцам. В i-м элементе массива ig будем хранить номер элемента в массиве ggl (или ggu), с которого начинается i-я строка нижнего (или i-й столбец верхнего) треугольника профиля матрицы. При

этом длина массива ig должна быть равна $N+1$ (N – размерность СЛАУ), и последний его элемент будет содержать целое число, на единицу большее числа элементов в массиве ggl (и ggu). Например, матрица

$$\begin{bmatrix} 1. & 2. & 0 & 3. & 0 & 0 \\ 4. & 5. & 0 & 6. & 0 & 7. \\ 0 & 0 & 8. & 0 & 0 & 9. \\ 10. & 11. & 0 & 12. & 13. & 0 \\ 0 & 0 & 0 & 14. & 15. & 0 \\ 0 & 16. & 17. & 0 & 0 & 18. \end{bmatrix} \tag{2.1}$$

в профильном формате примет вид:

$$di = \{1.,5.,8.,12.,15.,19.\}, \qquad ig = \{1,1,2,2,5,6,10\},$$

$$ggl = \{4.,10.,11.,0,14.,16.,17.,0,0\}, \qquad ggu = \{2.,3.,6.,0,13.,7.,9.,0,0\}.$$

Рассмотрим некоторые приёмы работы с матрицами, хранящимися в профильном формате. В этом формате количество ненулевых элементов в i-й строке нижнего треугольника может быть вычислено как $ig(i+1)-ig(i)$, а номер столбца, в котором находится первый ненулевой элемент этой строки, – как $i-(ig(i+1)-ig(i))$. Аналогично вычисляется количество ненулевых элементов в i-м столбце (также равное $ig(i+1)-ig(i)$) и номер строки, в которой находится первый ненулевой элемент этого столбца (тоже равный $i-(ig(i+1)-ig(i))$).

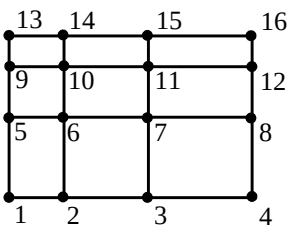
В качестве примера работы с профильным форматом приведём на языке Фортран подпрограмму занесения числа a в компоненту A_{ij} глобальной матрицы **A**.

```
subroutine AddElement(N,di,ig,ggl,ggu,i,j,a)
dimension di(N),ggl(*),ggu(*),ig(N+1)
if(i.eq.j)then
 di(i)=di(i)+a
 return
end if
if(i.lt.j)then
 ind=ig(j+1)-(j-i)
*   мы предполагаем, что портрет правильный, и
*   поэтому ошибки не контролируем (очевидно
*   ind должен быть не меньше ig(j))
 ggu(ind)=ggu(ind)+a
else
 ind=ig(i+1)-(i-j)
```

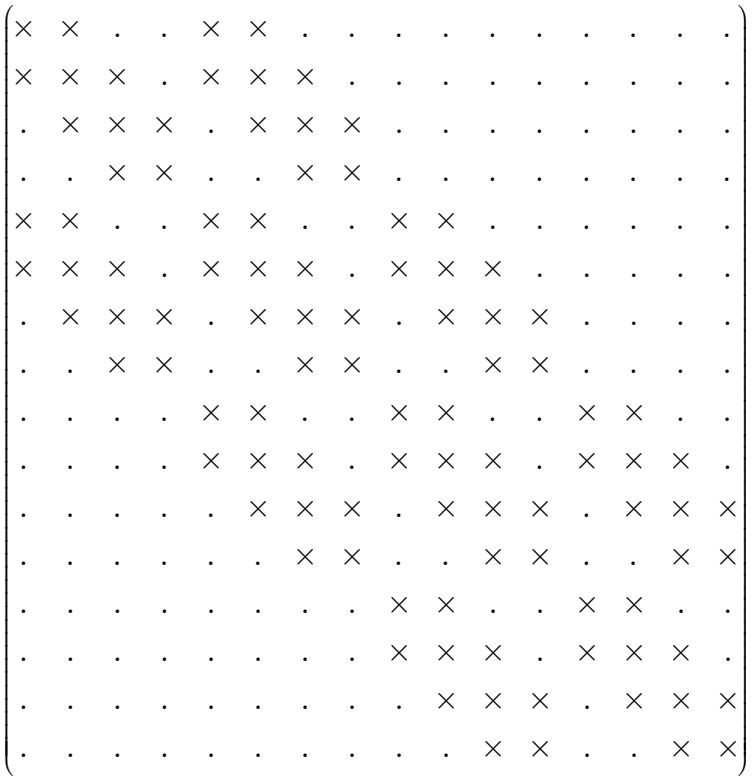
```
  ggl(ind)=ggl(ind)+a
end if
end
```

2.1.3. ЛЕНТОЧНЫЙ ФОРМАТ ХРАНЕНИЯ МАТРИЦЫ СЛАУ

При работе с регулярными сетками вместо профильного формата хранения матрицы можно использовать более простой *ленточный формат*. Например, если прямоугольная расчётная область регулярно разбивается на прямоугольные конечные элементы с билинейными базисными функциями, то при регулярной нумерации узлов, например, слева направо и снизу вверх (как, впрочем, и при нумерациях снизу вверх и слева направо, или справа налево и снизу вверх, или других, аналогичных им) матрица конечноэлементной СЛАУ будет девятидиагональной. Для сетки, представленной на рис. 22,*а*, структура матрицы конечноэлементной СЛАУ показана на рис. 22,*б* (крестиком обозначены ненулевые компоненты, точкой – нулевые).



а



б

Рис. 22. Структура матрицы конечноэлементной СЛАУ (*б*)
для регулярной прямоугольной сетки (*а*)

Поскольку в матрицах такого типа профиль имеет практически постоянный размер, отпадает необходимость в хранении массива ig, достаточно хранить одно

число – полуширину ленты lw (в приведённом примере полуширина ленты равна пяти). При вычислении адреса в массиве ggl (или ggu) любой компоненты матрицы нужно только учесть, что в первых lw строках нижнего треугольника (или столбцах верхнего треугольника) число хранимых элементов на единицу меньше номера строки (или столбца), а в остальных строках (или столбцах) число хранимых элементов равно lw.

Таким образом, при ленточном формате хранения глобальной матрицы **A** конечноэлементной СЛАУ подпрограмма занесения числа a в компоненту A_{ij} будет выглядеть следующим образом:

```
subroutine AddElement(N,lw,di,ggl,ggu,i,j,a)
dimension di(N),ggl(*),ggu(*)
if(i.eq.j)then
 di(i)=di(i)+a
 return
end if
if(i.lt.j)then
  if(j.le.lw)then
* количество значений в предыдущих столбцах
* можно посчитать как ind = sum_{k=1}^{j-1}(k-1) = (j-1)(j-2)/2
   ind=(j-1)*(j-2)/2+i
  else
   ind=lw*(lw-1)/2+lw*(j-lw-1)+i
  end if
*   мы предполагаем, что портрет правильный, и
*   поэтому ошибки не контролируем
 ggu(ind)=ggu(ind)+a
else
  if(i.le.lw)then
   ind=(i-1)*(i-2)/2+j
  else
   ind=lw*(lw-1)/2+lw*(i-lw-1)+j
  end if
 ggl(ind)=ggl(ind)+a
end if
end
```

2.1.4. РАЗРЕЖЕННЫЕ ФОРМАТЫ ХРАНЕНИЯ МАТРИЦ СЛАУ

Разреженные строчный, столбцовый и строчно-столбцовый форматы нацелены на хранение только тех элементов матрицы, которые не равны нулю. В

этих форматах хранят, как правило, матрицы таких СЛАУ, для решения которых будут использованы итерационные методы, не изменяющие **портрет** матрицы, т.е. *местоположение её ненулевых элементов*.

Здесь мы так же, как и при описании профильного формата, будем считать, что портрет матрицы симметричный, и для хранения нижнего треугольника матрицы используется *разреженный строчный формат*, а для хранения верхнего – *разреженный столбцовый формат*. Такой формат мы будем называть **разреженным строчно-столбцовым** или просто **разреженным** форматом.

Для хранения матрицы в разреженном строчно-столбцовом формате нам потребуется, кроме массива di с диагональными компонентами матрицы $\mathbf{A}$, ещё четыре массива: ggl и ggu – для хранения самих ненулевых компонент нижнего и верхнего треугольников матрицы $\mathbf{A}$, а также ig и jg – массивы для определения строк и столбцов ненулевых компонент матрицы $\mathbf{A}$.

В массиве ggl будем хранить последовательно по строкам все ненулевые элементы нижнего треугольника матрицы, а в массиве ggu – по столбцам все ненулевые элементы верхнего треугольника матрицы. При этом в i-м элементе массива ig должен храниться номер элемента в массиве ggl (и ggu), с которого начинаются компоненты i-й строки нижнего (и i-го столбца верхнего) треугольника матрицы. Длина массива ig должна быть равна $N+1$ (N –размерность СЛАУ), и последний его элемент должен содержать число, на единицу большее длины массива ggl (и ggu).

Массив jg служит для определения номеров столбцов элементов нижнего треугольника матрицы $\mathbf{A}$ и номеров строк элементов её верхнего треугольника. Длина массива jg совпадает с длиной массивов ggl и ggu. Элемент $jg\left(j\right)$ содержит номер столбца, в котором расположена компонента матрицы $ggl\left(j\right)$, и номер строки, в которой находится компонента $ggu\left(j\right)$. Например, матрица (2.1) в разреженном формате имеет вид:

$di=\{1.,5.,8.,12.,15.,19.\}$, $ig=\{1,1,2,2,4,5,7\}$, $jg=\{1,1,2,4,2,3\}$,

$ggl=\{4.,10.,11.,14.,16.,17.\}$, $ggu=\{2.,3.,6.,13.,7.,9.\}$.

Приёмы работы с матрицами, хранящимися в разреженном формате, довольно похожи на приёмы работы с матрицами в профильном формате. Как и в профильном формате, количество хранящихся элементов в i-й строке нижнего тре-

угольника (i-м столбце верхнего треугольника) может быть вычислено как $ig(i+1)-ig(i)$.

Чтобы найти адрес ячейки в массиве ggl, в которой хранится значение компоненты A_{ij} из нижнего треугольника матрицы **A** (т.е. при $i>j$), нужно сначала найти элемент со значением j в массиве jg, просматривая jg с элемента с номером $ig(i)$ до элемента с номером $ig(i+1)-1$. Индекс l элемента $jg(l)$, равного j, и будет номером искомой ячейки (содержащей значение A_{ij}) в массиве ggl. Аналогично определяются адреса ячеек массива ggu, в которых хранятся компоненты A_{ij} верхнего треугольника матрицы **A** (т.е. при $i<j$).

Алгоритм умножения матрицы, хранящейся в разреженном (строчно-столбцовом) формате, на вектор **x** выглядит следующим образом (результат заносится в вектор **y**):

```
* умножаем главную диагональ матрицы
do i = 1,N
    y(i)=di(i)*x(i)
end do
* цикл по строкам нижнего и столбцам верхнего
* треугольника матрицы
do i = 1,N
    do j = ig(i),ig(i+1)-1
* умножаем компоненту нижнего треугольника A_ik
* на x_k, где k=jg(j)
        y(i)=y(i)+ggl(j)*x(jg(j))
* умножаем компоненту верхнего треугольника A_ki
* на x_i, где k=jg(j)
        y(jg(j))=y(jg(j))+ggu(j)*x(i)
    end do
end do
```

Обратим внимание на то, как умножается верхний треугольник матрицы на вектор **x**. Поскольку при выбранном способе хранения матрицы поиск элементов одной строки её верхнего треугольника затруднён, умножение на компоненты вектора **x** верхнего треугольника матрицы проводится *проходом по его столбцам* – очередной (j-й) ненулевой элемент i-го столбца матрицы умножается на $x(i)$ и добавляется к компоненте $y(k)$ с номером $k=jg(j)$.

Очевидно, что для хранения симметричных матриц достаточно одного массива ggl, поскольку ggu в этом случае полностью совпадёт с ним. При этом рассмотренный чуть выше алгоритм умножения матрицы на вектор совершенно не изменится – достаточно лишь заменить массив ggu на ggl (если этот алгоритм оформлен в виде подпрограммы с формальными параметрами ggl и ggu, то достаточно при обращении к этой подпрограмме подставить в качестве фактического параметра название одного и того же массива и для ggl, и для ggu).

Для некоторых методов решения СЛАУ рассмотренный разреженный (строчно-столбцовый) формат хранения матриц с симметричным портретом может оказаться не очень удобным. К таким методам относится, например, метод релаксации.

Конечно, процедуру решения СЛАУ даже при использовании метода релаксации всё-таки можно организовать так, чтобы избежать поисков элементов матрицы и в случае хранения её в рассмотренном нами выше разреженном формате. Для этого нужно уравнения СЛАУ обрабатывать *последовательно снизу вверх*. При любой другой схеме обработки уравнений СЛАУ (например, при необходимости организовать выборочную релаксацию) без процедур поиска ненулевых элементов строк верхнего треугольника уже будет трудно обойтись. В этом случае можно потратить дополнительную память на увеличение (вдвое) массива jg и хранить компоненты матрицы в одном массиве gg, причём *все* компоненты каждой строки хранить последовательно (включая и компоненты из верхнего треугольника матрицы), т.е. хранить матрицу в **разреженном строчном формате**.

В некоторых же случаях самым удобным может оказаться **разреженный столбцовый формат**, в котором *все* ненулевые компоненты матрицы хранятся в массиве gg последовательно по столбцам, в массиве jg хранятся соответствующие номера строк для этих компонент, а в массиве ig хранятся адреса ячеек массивов gg и jg, с которых начинаются столбцы матрицы.

2.1.5. АЛГОРИТМЫ ЗАНЕСЕНИЯ ЛОКАЛЬНЫХ МАТРИЦ В ГЛОБАЛЬНУЮ МАТРИЦУ, ХРАНЯЩУЮСЯ В РАЗРЕЖЕННОМ ФОРМАТЕ

Рассмотрим одну из основных процедур МКЭ – процедуру сборки глобальной матрицы из локальных – для глобальной матрицы, хранящейся в разреженном формате.

В принципе, для занесения компонент локальной матрицы в глобальную можно было бы воспользоваться следующей достаточно простой подпрограммой занесения числа a в компоненту A_{ij} глобальной матрицы (аналогичную той, что была приведена в качестве примера для профильного формата в конце раздела 2.1.2):

```
subroutine AddElement(N,di,ig,jg,ggl,ggu,i,j,a)
dimension di(N),ggl(*),ggu(*),ig(N+1),jg(*)
if(i.eq.j)then
 di(i)=di(i)+a
 return
end if
if(i.lt.j)then
 do ind=ig(j),ig(j+1)-1
  if(jg(ind).eq.i)exit
 end do
*   мы предполагаем, что портрет правильный, и
*   поэтому ошибки, связанные с отсутствием
*   нужной компоненты в портрете матрицы, не
*   контролируем;
*   на самом деле, попадание в это место
*   подпрограммы означает, что нужной
*   компоненты нет в портрете
 ggu(ind)=ggu(ind)+a
else
 do ind=ig(i),ig(i+1)-1
  if(jg(ind).eq.j)exit
 end do
 ggl(ind)=ggl(ind)+a
end if
end
```

Однако этот способ не всегда эффективен. Конечно, основные затраты машинного времени при генерации конечноэлементной матрицы приходятся на вещественные операции, которые необходимо сделать при вычислении компонент локальных матриц. Но для некоторых типов конечных элементов, для которых в строке глобальной матрицы может оказаться достаточно много ненулевых компонент (как, например, для элементов высоких порядков), цикл поиска индекса в массиве jg может занимать заметное время, не только сравнимое со временем вычисления компонент локальных матриц, но и даже превышающее его. В этом

случае процедуру занесения локальной матрицы в глобальную можно ускорить следующими способами.

Во-первых, можно сделать так, чтобы элементы массива jg, соответствующие одной и той же строке нижнего треугольника (и, естественно, одному и тому же столбцу верхнего треугольника) были упорядочены по возрастанию. Такое упорядочение никак не изменит процедуры умножения матрицы на вектор, но позволит ускорить поиск в массиве jg нужного индекса при занесении одной компоненты локальной матрицы в глобальную (например, можно использовать простейший способ ускоренного поиска в упорядоченном массиве – дихотомию).

Во-вторых, можно ускорить процедуру занесения компонент локальной матрицы в глобальную, если заносить эти компоненты не по одной, а все вместе. При этом без всяких дополнительных действий (и даже без упорядочения массива jg) для несимметричных матриц поиск легко ускоряется практически в два раза, поскольку для симметрично расположенных компонент локальной матрицы индекс в jg можно будет искать один раз. Но гораздо большего ускорения можно добиться в том случае, если на конечном элементе номера локальных базисных функций упорядочить по возрастанию соответствующих им глобальных номеров, т.е. если пронумеровать локальные базисные функции так, чтобы базисная функция с минимальным глобальным номером имела на конечном элементе локальный номер 1, базисная функция со следующим по величине глобальным номером – локальный номер 2 и т.д. Рассмотрим алгоритм, реализующий такое занесение локальной матрицы в глобальную.

Будем считать, что массив jg соответствующим образом упорядочен, а локальная матрица $\hat{\mathbf{A}}$ размера $k \times k$ конечного элемента с упорядоченными локальными номерами базисных функций представлена двумя массивами: двумерным вещественным массивом `A` размера `k×k` и целочисленным массивом `L` длины `k`. При этом элемент `L(i)` – это номер глобальной базисной функции, соответствующей локальной базисной функции с номером `i`. Тогда подпрограмма занесения локальной матрицы в глобальную может быть следующей:

```
subroutine AddLocal(N,di,ig,jg,ggl,ggu,k,L,A)
dimension di(N),ggl(*),ggu(*),ig(N+1),jg(*)
dimension L(k),A(k,k)
* заносим диагональные элементы
do i=1,k
 di(L(i))=di(L(i))+A(i,i)
```

```
end do
* начинаем цикл по строкам нижнего (и
* одновременно по столбцам верхнего)
* треугольника локальной матрицы
do i=1,k
* устанавливаем начальное значение нижней
* границы поиска. Поскольку локальные номера
* упорядочены, для каждого последующего
* элемента строки начинать поиск можно с уже
* найденного индекса
 ibeg=ig(L(i))
* начинаем поиск дихотомией по строке нижнего
* (и по столбцу верхнего) треугольника
do j=1,i-1
 iend=ig(L(i)+1)-1
  do while(jg(ibeg).ne.L(j))
   ind=(ibeg+iend)/2
    if(jg(ind).le.L(j))then
     ibeg=ind+1
    else
     iend=ind
    end if
  end do
  ggl(ibeg)=ggl(ibeg)+A(i,j)
  ggu(ibeg)=ggu(ibeg)+A(j,i)
  ibeg=ibeg+1
 end do
end do
end
```

Заметим, что упорядочение локальных номеров базисных функций по возрастанию их глобальных номеров необходимо *только для ускорения* работы с глобальной матрицей, т.е. ускорения процесса сборки и алгоритма построения портрета матрицы, который будет рассмотрен позже. Для вычисления же самой локальной матрицы упорядочение локальных базисных функций может оказаться даже нежелательным (например, при использовании шаблонных элементов со «своей» нумерацией базисных функций). Поэтому в некоторых случаях имеет смысл использовать сразу две локальные нумерации базисных функций конечного элемента: одну в соответствии с нумерацией базисных функций шаблонного элемента, вторую – по возрастанию глобальных номеров базисных функций.

Программа же занесения локальной матрицы в глобальную может очень мало отличаться от рассмотренной выше. Для этого достаточно (на каждом конечном элементе) ввести целочисленный массив `LL` длины `k` такой, что в первой его ячейке хранится локальный номер базисной функции с минимальным глобальным номером, во второй – локальный номер базисной функции со следующим по величине глобальным номером и т.д. Тогда в рассмотренной выше программе нужно просто заменить обращение к компонентам `L(i)`, `L(j)`, `A(i,i)`, `A(i,j)` и `A(j,i)` обращениями `L(LL(i))`, `L(LL(j))`, `A(LL(i),LL(i))`, `A(LL(i),LL(j))` и `A(LL(j),LL(i))` соответственно, что и обеспечит занесение компонент локальной матрицы в глобальную строго по возрастанию глобальных номеров базисных функций.

2.1.6. ДИАГОНАЛЬНЫЙ ФОРМАТ ХРАНЕНИЯ МАТРИЦЫ СЛАУ

Разреженные форматы могут оказаться не самыми эффективными для матриц СЛАУ, полученных при решении задачи на конечноэлементной сетке с регулярной структурой. Для матриц такого типа более эффективными являются форматы, учитывающие закономерности расположения ненулевых компонент. К таким форматам относится диагональный формат хранения разреженных матриц, имеющих регулярную структуру.

По-прежнему будем считать, что матрица имеет симметричный портрет. В диагональном формате в двумерных массивах ggl и ggu размером $Nd \times \left(N-1\right)$ хранятся элементы ненулевых побочных диагоналей матрицы $\mathbf{A}$ (Nd – число таких диагоналей в одном из треугольников матрицы $\mathbf{A}$, N – размерность СЛАУ). Во вспомогательном массиве ig длиной Nd хранятся смещения ненулевых диагоналей относительно главной (т.е. фактически уменьшенные на единицу номера строк, с которых начинаются диагонали нижнего треугольника, или уменьшенные на единицу номера столбцов, с которых начинаются диагонали верхнего треугольника).

Например, матрица

$$\begin{bmatrix} 1. & 2. & 0 & 3. & 0 & 0 \\ 4. & 5. & 6. & 0 & 7. & 0 \\ 0 & 8. & 9. & 0. & 0 & 11. \\ 12. & 0 & 13. & 14. & 15. & 0 \\ 0 & 16. & 0 & 0. & 18. & 19. \\ 0 & 0 & 20. & 0 & 21. & 22. \end{bmatrix} \qquad (2.2)$$

в диагональном формате имеет вид (на место несуществующих элементов в массивы ggl и ggu занесены нули):

$$di = \{1.,5.,9.,14.,18.,22.\}, \qquad ig = \{1,3\},$$

$$ggu = \begin{Bmatrix} 2. & 6. & 0. & 15. & 19. \\ 3. & 7. & 11. & 0. & 0. \end{Bmatrix}, \qquad ggl = \begin{Bmatrix} 4. & 8. & 13. & 0. & 21. \\ 12. & 16. & 20. & 0. & 0. \end{Bmatrix}.$$

Приёмы работы с матрицей в таком формате достаточно просты. Например, программа умножения матрицы на вектор для этого формата может иметь следующий вид:

```
do i = 1,N
    y(i)=di(i)*x(i)
end do
do i = 1,Nd
    do j = 1,N-ig(i)
        ir=j+ig(i)
        y(ir)=y(ir)+ggl(i,j)*x(j)
        y(j)=y(j)+ggu(i,j)*x(ir)
    end do
end do
```

Рассмотрим, как можно найти местоположение компоненты A_{ij} матрицы **A** в массиве ggl (это необходимо, например, при сборке глобальной матрицы). Сначала в массиве ig нужно найти значение $i-j$ (заметим, что если в ig такого значения нет, то это означает, что $A_{ij}=0$ и эта компонента не хранится). Индекс элемента массива ig, равного $i-j$, обозначим через k (т.е. $ig\left(k\right)=i-j$). Тогда $ggl(k,j)$ будет искомым элементом массива ggl, в котором должно храниться значение компоненты A_{ij}.

2.2. ПОСТРОЕНИЕ ПОРТРЕТА КОНЕЧНОЭЛЕМЕНТНОЙ МАТРИЦЫ

Для работы с матрицей в разреженном формате нужно сначала построить её портрет (т.е. сформировать массивы ig и jg).

Построение портрета матрицы фактически означает нахождение в конечно-элементной сетке для каждой глобальной базисной функции ψ_i номеров j всех базисных функций, для которых $\int\limits_{\Omega} \psi_i \psi_j \neq 0$ или $\int\limits_{\Omega} \operatorname{grad} \psi_i \cdot \operatorname{grad} \psi_j \neq 0$, т.е. номеров всех глобальных базисных функций, не равных нулю на хотя бы одном из тех конечных элементов, на которых не равна нулю глобальная базисная функция ψ_i.

В дальнейшем две глобальные базисные функции, которые *обе не равны нулю* хотя бы на одном конечном элементе, будем называть **связанными** (так же будем называть и соответствующие таким базисным функциям веса – компоненты вектора неизвестных конечноэлементной СЛАУ).

Алгоритм построения портрета матрицы может быть реализован с помощью массива списков. Его основная идея заключается в следующем. Создаётся массив из N списков, где N – число глобальных базисных функций. При прохождении по конечным элементам добавляем в каждый список, соответствующий ненулевой на этом элементе глобальной базисной функции, номера всех остальных ненулевых на этом элементе глобальных базисных функций. Естественно, при каждом добавлении в список следует проверять, нет ли уже в нём добавляемого номера базисной функции. Кроме того, если портрет симметричный и строится только для одного из треугольников матрицы, например для нижнего треугольника с хранением компонент матрицы конечноэлементной СЛАУ по строкам, то добавлять в каждый список нужно *только те* глобальные номера функций, которые *меньше* номера самого списка.

Однако прямая программная реализация описанного алгоритма с использованием списков является не самой эффективной, причём не столько из-за затрат дополнительной памяти на реализацию списков, сколько из-за затрат времени на работу с ними (поскольку операции динамического выделения/освобождения памяти в языках высокого уровня являются достаточно затратными). Поэтому мы продемонстрируем реализацию этого алгоритма с однократным выделением памяти под весь список.

Будем считать, что для обращения к структурам данных определены следующие функции: функция `NumberOfUnknowns(I)`, которая возвращает число глобальных базисных функций, ненулевых на элементе с номером i, и функция `IndexOfUnknown(I,J)`, которая возвращает глобальный номер j-й локальной базисной функции на элементе с номером i.

Тогда программа на языке Фортран, реализующая алгоритм построения портрета (т.е. формирующая массивы ig и jg), может быть следующей:

```
subroutine GeneratePortrait(ig,jg,N,Kel,list)
* N – число глобальных базисных функций в
* конечноэлементной сетке
* Kel – число конечных элементов в сетке
dimension ig(N+1),jg(*),list(2,*)
* list – временный массив для хранения списка.
* для этого массива должно быть выделено
* вдвое больше памяти, чем для массива jg.
* Для упрощения программы мы не будем
* проверять, что алгоритм не выходит за границы
* выделенной памяти, хотя в реальной программе
* это всё-таки надо бы делать.
* Элементы list(1,:) будут хранить
* номер базисной функции, а list(2,:) – номер
* следующего элемента в массиве list.
* Номер 0 означает конец списка

ALLOCATABLE listbeg(N)
ALLOCATE listbeg
* временный массив listbeg будет хранить
* номера элементов массива list, с которых
* начинаются списки для каждой из базисных
* функций

listsize=0
*listsize – текущая длина массива list
do i=1,N
listbeg (i) = 0
end do
* цикл формирования списка одним проходом по
* элементам.
do ielem=1,Kel
 do i=1,NumberOfUnknowns(ielem)
  k=IndexOfUnknown(ielem,i)
  do j=i+1,NumberOfUnknowns(ielem)
   ind1=k
   ind2=IndexOfUnknown(ielem,j)
   if(ind2.lt.ind1)then
    ind1=ind2
    ind2=k
   end if
```

```
* заносить будем связь большего номера с
* меньшим, т.е. ind2 с ind1
   iaddr=listbeg(ind2);
   if(iaddr.eq.0)then
* списка ещё не было, создаём его
    listsize=listsize+1
    listbeg(ind2)=listsize
    list(1,listsize)=ind1
    list(2,listsize)=0
   else
* список уже был. Ищем в нём ind1.
    do while(list(1,iaddr).lt.ind1.and.
                list(2,iaddr).gt.0)
        iaddr=list(2,iaddr)
    end do
    if(list(1,iaddr).gt.ind1)then
* не нашли и встретили элемент с большим
* номером. Добавляем перед ним, чтобы список
* оставить упорядоченным
    listsize=listsize+1
    list(1,listsize)=list(1,iaddr)
    list(2,listsize)=list(2,iaddr)
    list(1,iaddr)=ind1
    list(2,iaddr)=listsize
    else
    if(list(1,iaddr).lt.ind1)then
* не нашли, а список закончился.
* Добавляем в конец списка
        listsize=listsize+1
        list(2,iaddr)=listsize
        list(1,listsize)=ind1
        list(2,listsize)=0
    end if
    end if
   end if
  end do
 end do
end do

* Создание портрета по списку
ig(1)=1
do i=1,N
 ig(i+1)=ig(i)
* ig(i+1) фактически указывает на ячейку
```

```
* массива jg, куда надо поместить следующий
* элемент
 iaddr=listbeg(i)
 do while(iaddr.ne.0)
    jg(ig(i+1))=list(1,iaddr)
    ig(i+1)=ig(i+1)+1
    iaddr=list(2,iaddr)
 end do
end do
```

Заметим, что приведённый алгоритм является практически универсальным в том смысле, что он не привязан к типу конечных элементов, и позволяет использовать в одной конечноэлементной сетке элементы с различным числом базисных функций. Единственным ограничением, делающим этот алгоритм не совсем универсальным, является предположение о том, что все локальные базисные функции на одном элементе являются связанными между собой (т.е. предполагается, что для любых двух функций с локальными номерами i и j элемент $\hat{A}_{ij}$ локальной матрицы $\hat{\mathbf{A}}$ не равен нулю). Однако это ограничение достаточно просто ликвидируется введением дополнительной функции, проверяющей перед занесением связи в список её наличие у обрабатываемого элемента. Соответствующие изменения в программе мы предоставляем сделать читателю в качестве упражнения.

Обратим внимание, что приведённый алгоритм сразу строит массив jg так, что номера элементов одной строки нижнего треугольника упорядочены в нём по возрастанию, что удобно не только для приведённого выше алгоритма сборки матрицы, но и для методов решения СЛАУ с использованием неполной факторизации.

Кроме того, рассмотренный нами алгоритм формирования портрета разреженной матрицы может быть положен в основу алгоритма построения упорядоченного массива рёбер конечных элементов, который будет рассмотрен в разделе 2.3.3.

2.3. АЛГОРИТМЫ НУМЕРАЦИИ БАЗИСНЫХ ФУНКЦИЙ, ГРАНЕЙ И РЁБЕР КОНЕЧНЫХ ЭЛЕМЕНТОВ

2.3.1. НУМЕРАЦИЯ ГЛОБАЛЬНЫХ БАЗИСНЫХ ФУНКЦИЙ

Все базисные функции рассмотренных ранее типов конечных элементов можно условно подразделить на следующие основные классы:

1) функции, связанные с геометрическими вершинами элементов сетки, – это функции, отвечающие за значения в узлах, совпадающих с геометрическими вершинами сетки для лагранжевых элементов, и все привязанные к вершинам элементов функции эрмитовых элементов;
2) функции, ненулевые внутри только одного элемента, – это все базисные функции, ассоциированные с внутренними узлами конечных элементов;
3) функции, ассоциированные с рёбрами элементов, – это все базисные функции, привязанные к узлам сетки, лежащим на рёбрах элементов и не совпадающим с геометрическими вершинами элементов;
4) функции, ассоциированные с гранями трёхмерных элементов, – это все базисные функции, привязанные к узлам сетки, лежащим на гранях элементов и не совпадающим с геометрическими вершинами элементов и узлами на их рёбрах.

Очевидно, что нумерация базисных функций первого класса не представляет особой сложности, поскольку исходная информация о сетке уже содержит номера геометрических вершин каждого конечного элемента. Для простейшего (и наиболее частого) случая, когда каждой вершине соответствует одно и то же число k базисных функций, никакой алгоритм нумерации вообще не нужен, поскольку для i-й вершины глобальный номер её j-й базисной функции ($j = 1 \ldots k$) может определяться по номеру вершины i как $k\left(i-1\right)+j$. Для остальных случаев возможен очень простой алгоритм нумерации, когда первой функции, соответствующей $\left(i+1\right)$-й вершине, присваивается номер $n_{i+1} = n_i + k_i$, где k_i – число функций, соответствующих i-й вершине.

Так же просто пронумеровать базисные функции второго класса. Для простейшего случая, когда число базисных функций k_i, ассоциированных с внутренними узлами любого конечного элемента, одинаково и равно k, глобальный номер первой локальной базисной функции элемента Ω_i можно определить как $n_0 + k\left(i-1\right)+1$, где n_0 – общее число уже пронумерованных функций других классов. Для остальных случаев возможен, как и для функций первого класса, алгоритм нумерации, когда первой функции, ассоциированной с внутренним узлом элемента Ω_{i+1}, присваивается номер $n_{i+1} = n_i + k_i$, где k_i – число функций, ассоциированных с внутренними узлами элемента Ω_i.

Нумерация же функций третьего и четвёртого классов может быть выполнена аналогично только в том случае, если предварительно пронумерованы соответ-

ствующие *граничные элементы* (*рёбра и грани*) всех конечных элементов. Тогда в массивах, размерности которых определяются числом соответствующих граничных элементов, можно сохранить значения k_i – число функций, привязанных к внутренним узлам i-го граничного элемента, и, как и ранее, определить номер первой функции на $i+1$-й границе как $n_{i+1} = n_i + k_i$. Однако подсчёт и нумерация рёбер конечных элементов и их трёхмерных граней являются отдельной проблемой, которую мы рассмотрим чуть ниже.

Необходимо отметить, что рассмотренные здесь методы могут приводить к нумерации базисных функций, которая является далеко не лучшей для решения СЛАУ. Дело в том, что при использовании этих методов связанные между собой базисные функции могут получить далёкие друг от друга номера, что, в свою очередь, может заметно осложнить процесс решения конечноэлементной СЛАУ из-за роста размера профиля при использовании как прямых, так и некоторых итерационных методов (например, с предобусловливанием на основе неполной факторизации). Поэтому построенная нами нумерация должна рассматриваться, в общем-то, как первичная, позволяющая лишь *дать одинаковые* (и вместе с тем уникальные) *номера* базисным функциям, ненулевым на нескольких элементах. Для получения же оптимальной нумерации должны быть применены какие-либо алгоритмы сортировки.

Подробное рассмотрение алгоритмов сортировки неизвестных для уменьшения профиля матрицы СЛАУ выходит за рамки данного курса лекций. Мы рассмотрим только простейшую идею перенумерации базисных функций, ассоциированных с узлами сетки (причём *с любыми узлами*, а не только с геометрическими вершинами).

Запишем в массив точек все узлы сетки, с которыми ассоциированы базисные функции, так, что адрес точки в массиве будет определяться номером ассоциированной с ней функции (т.е. каждый узел запишем столько раз, сколько функций к нему привязано). Пересортируем точки в этом массиве по возрастанию каждой из их координат (или любым другим способом, при котором узлы, расположенные в конечноэлементной сетке недалеко друг от друга, окажутся и достаточно близко расположенными друг к другу в сформированном массиве узлов). Новый номер каждой базисной функции определим как адрес соответствующей этой функции точки в пересортированном массиве (естественно, для этого у каж-

дой хранящейся в массиве точки должен быть известен её старый номер, т.е. порядковый номер этой точки в массиве узлов до его пересортировки).

На самом деле, этот очень простой алгоритм нумерации базисных функций является достаточно универсальным, поскольку практически всегда можно считать, что базисные функции, ассоциированные с рёбрами, гранями или самими элементами, ассоциированы с определёнными узлами на этих рёбрах или гранях или с узлами внутри элементов. Однако с ростом числа базисных функций (при дроблении сетки или повышения порядка элемента) он может стать довольно затратным как по объёму памяти, так и по количеству машинных операций. Поэтому ниже (на с. 76) мы рассмотрим гораздо более эффективный алгоритм нумерации базисных функций, основанный на использовании упорядоченных списков рёбер, граней и конечных элементов.

Упорядочение рёбер, граней и конечных элементов необходимо также и при использовании так называемых *иерархических* базисных функций, которые будут рассмотрены нами в разделе 3.7. Эти базисные функции порождают подпространство, совпадающее с подпространством, натянутым на базисные функции лагранжева типа. Иерархические базисные функции удобно ассоциировать с рёбрами, гранями и самими конечными элементами. Поэтому для работы с иерархическими базисными функциями необходимы соответствующие списки рёбер и граней (список же конечных элементов создаётся на этапе построения сетки, и он является основой для формирования списков граней и рёбер). Кроме того, при наличии упорядоченных списков рёбер и граней можно гораздо быстрее упорядочить и пронумеровать и лагранжевы базисные функции на элементах высоких порядков – соответствующий алгоритм будет изложен на с. 76. Поэтому ниже (в разделах 2.3.2 и 2.3.3) мы рассмотрим алгоритмы, позволяющие из списка конечных элементов (заданных номерами их вершин) построить список всех граней и список всех рёбер.

2.3.2. ПОСТРОЕНИЕ СПИСКОВ ГРАНЕЙ ТРЁХМЕРНЫХ КОНЕЧНЫХ ЭЛЕМЕНТОВ. ИСПОЛЬЗОВАНИЕ УПОРЯДОЧЕННОГО СПИСКА ГРАНЕЙ ДЛЯ УСКОРЕНИЯ ПРОЦЕДУРЫ НУМЕРАЦИИ БАЗИСНЫХ ФУНКЦИЙ

Кроме того, что построение списка граней элементов может оказаться необходимым для первичной нумерации базисных функций, ассоциированных с этой гранью (или её узлами, не являющимися вершинами элемента или узлами его рё-

бер), список граней может потребоваться для построения поверхности, разделяющей подобласти с различными свойствами. Это используется, например, в постпроцессорах для отрисовки поверхностей отдельных подобластей расчётной области. Список граней также может понадобиться при решении задач, вариационная постановка которых включает поверхностные интегралы по внутренней границе.

Построить список граней можно многими способами. Мы кратко опишем один из них.

Сначала нужно приписать каждой грани каждого конечного элемента номер смежного по этой грани конечного элемента (что требует, естественно, дополнительного объёма памяти для хранения почти 2κ целых чисел, где κ – общее число граней конечных элементов). Для регулярных сеток, а также для некоторых алгоритмов построения нерегулярных сеток информация о соседях каждого конечного элемента известна на этапе построения сетки и может быть сохранена соответствующим препроцессором. Мы же рассмотрим, как можно получить эту информацию в наиболее часто встречающемся случае, когда она не была сформирована на этапе генерации сетки, а конечноэлементная сетка хранится поэлементно и каждый элемент задаётся *только номерами* своих узлов.

Поставим в центр каждой грани по одной точке на каждом из конечных элементов (естественно, центр одной и той же грани на двух смежных конечных элементах должен определяться по одному и тому же алгоритму, чтобы соответствующие точки совпали или почти совпали). Проходя по списку конечных элементов, запишем эти точки в массив (назовём этот массив P). Затем упорядочим массив P любым способом по значениям координат (например, сначала по возрастанию координаты x, при одинаковых значениях координаты x – по возрастанию значений y и затем по возрастанию z). В упорядоченном массиве P одинаковые точки окажутся на соседних местах, и поэтому повторяющиеся точки легко можно обнаружить и удалить. Теперь номером грани можно считать номер точки в полученном после удаления повторов массиве P.

Теперь рассмотрим, как можно приписать к каждой грани по два смежных по ней конечных элемента. Создадим для этого целочисленный массив ind номеров элементов длиной κ, где κ – общее число граней конечных элементов (которое равно длине массива P после удаления повторов). Запишем в массив ind нулевые значения. Затем пройдём по конечным элементам и для каждой грани каждого

конечного элемента (обозначим номер элемента через m) определим номер i точки – центра этой грани (в упорядоченном массиве P это можно сделать достаточно быстро). Если $ind(i)$ равно нулю, то присвоим $ind(i)$ значение m, а если $ind(i)$ не равно нулю, то $ind(i)$ – номер элемента, смежного по рассматриваемой грани с элементом m, и в структуру данных, хранящую информацию о конечных элементах, нужно внести соответствующую информацию (о номерах соседей) для обоих элементов.

Недостатком такого способа поиска соседей является не столько существенный объём дополнительной памяти (если считать, что вещественное число занимает вдвое больше памяти, чем целое, то может потребоваться до $2 \cdot 3 \cdot 2 \cdot \kappa = 12\kappa$ целых чисел под массив P), сколько затраты на упорядочение вещественного массива координат «центров» граней.

Несколько более эффективным по памяти и быстродействию является следующий алгоритм.

Для каждого узла сетки подсчитаем число конечных элементов, которые его содержат. Для этого нам понадобится массив ig целых чисел длиной n, где n – число узлов в сетке. Присвоим элементам этого массива нулевые значения. Затем пройдём в цикле по всем конечным элементам, и в нём будем добавлять единицу к тем элементам $ig(i)$ массива ig, у которых значение индекса i является номером вершины обрабатываемого конечного элемента.

Теперь создадим массив jg из n подмассивов переменной длины, причём длина i-го подмассива должна быть равна $ig(i)$ (т.е. общая длина массива jg – это сумма значений всех элементов массива ig). Мы не случайно дали массивам такие имена, структура данных действительно похожа на структуру для хранения портрета разреженной матрицы.

Значения элементов массива $ig(i)$ пересчитаем так, чтобы они стали не длинами массивов, а адресами начал этих массивов в массиве jg.

Заполним массив jg номерами конечных элементов, содержащих каждую (i-ю) вершину, записав эти номера в jg по адресам от $ig(i)$ до $ig(i+1)-1$. Для этого пройдём в цикле по всем конечным элементам, записывая номер конечного элемента (равный, естественно, значению индекса цикла) в каждый из подмассивов массива jg с номерами, равными номерам вершин обрабатываемого конечного

элемента. Для формирования этой информации можно ввести ещё один целочисленный массив длины n, в i-й ячейке которого должно храниться количество номеров конечных элементов, уже занесённых в i-й подмассив массива jg. Процесс формирования массива jg завершим упорядочением каждого из n его подмассивов.

С помощью созданной структуры данных поиск соседей конечного элемента по любой из его граней можно выполнять следующим образом.

Пусть p – число вершин обрабатываемой грани m-го конечного элемента. Берём из первой ячейки подмассива массива jg с номером, равным номеру первой вершины обрабатываемой грани, хранящееся там значение номера конечного элемента. Обозначим этот номер через α. Проверяем, присутствует ли номер α во всех остальных $\left(p-1\right)$-м подмассивах с номерами, равными остальным номерам вершин обрабатываемой грани (для ускорения этого поиска и было сделано упорядочение номеров конечных элементов в подмассивах массива jg). Если хотя бы в одном из подмассивов значение α не найдено, то переходим к следующему элементу первого подмассива. Если α найдено во всех подмассивах, то α и есть номер конечного элемента, соседнего с m-м по обрабатываемой грани (естественно, значение α, совпадающее с m, должно пропускаться).

Очевидно, что параллельно с поиском смежных конечных элементов можно без всякого труда и пронумеровать все грани, давая новый номер только той грани обрабатываемого конечного элемента, для которой ещё не был найден номер смежного по ней элемента (естественно, при обработке других конечных элементов).

Сравним оба рассмотренных нами алгоритма формирования списка смежных (по граням) конечных элементов по требуемому для их работы объёму памяти. Подсчитаем объём памяти, необходимый для работы последнего алгоритма. Для простоты будем считать, что все элементы содержат одно и то же число вершин μ и граней ν. Обозначим через θ общее число элементов. Тогда длина массива jg равна $\mu\theta$, поскольку в нём номер каждого элемента хранится столько раз, сколько у него вершин. Очевидно, что если большинство граней внутренние, то их общее число можно оценить как $\kappa \approx \dfrac{\nu\theta}{2}$. Это означает, что массив jg для любого из рассмотренных нами типов конечных элементов занимает меньше памяти, чем массив P из первого алгоритма, поскольку массив P занимал около $12\kappa = 6\nu\theta$ ячеек

целочисленного массива, а для конечных элементов практически любого типа (тетраэдров, призм, шестигранников) значение 6ν в несколько раз превышает значение μ. Кроме того, последний алгоритм имеет преимущество и в скорости. Действительно, вместо *полного* упорядочения большого *вещественного* массива и поиска в нём нужных значений (координат точек) для определения номера обрабатываемой грани в первом алгоритме во втором алгоритме нужно сделать лишь *частичное* упорядочение *целочисленного* массива и относительно небольшое число *целочисленных* сравнений при поиске смежного по обрабатываемой грани конечного элемента.

Однако второй алгоритм имеет и один существенный недостаток – в нём грани нумеруются без учёта их расположения в пространстве, в то время как в первом алгоритме они уже в процессе их нумерации упорядочиваются именно по геометрическому принципу. Геометрическая же упорядоченность граней является довольно важным фактором в том случае, когда по номерам граней определяются номера ассоциированных с ними базисных функций, и тогда предпочтительными являются такие нумерации, у которых базисные функции с пересекающимися носителями имеют близкие глобальные номера. В то же время этот принцип должен выполняться *для всех базисных функций сразу* (ассоциированных с вершинами, рёбрами, гранями и самими конечными элементами), и поэтому нумерацию базисных функций с упорядочением их по геометрическому принципу всё равно нужно делать отдельно. Однако это упорядочение можно сделать гораздо быстрее, если вершины, рёбра и грани *предварительно* упорядочены по геометрическому принципу. При этом, естественно, нужно использовать специальный (очень быстрый) алгоритм, который упорядочивает узлы, хранящиеся в нескольких упорядоченных массивах.

На самом деле, упорядочить глобальные функции по геометрическому принципу можно не только путём геометрического упорядочения сразу всех узлов, с которыми эти базисные функции ассоциированы. Очень хороший результат (по критерию геометрической близости носителей базисных функций, имеющих близкие глобальные номера) можно получить и другими способами, основанными на работе, в основном, с целочисленными массивами.

Например, можно по значениям координат упорядочить *только вершины* конечных элементов, а грани, рёбра и сами конечные элементы упорядочивать уже по возрастанию *минимальных номеров* определяющих их вершин. Если же две

грани (так же, как два ребра или два конечных элемента) имеют вершины с одинаковыми минимальными номерами, то их можно упорядочить по номерам других вершин (также по возрастанию их номеров). Конечно, прежде чем упорядочивать таким способом все грани (рёбра или конечные элементы), нужно сначала упорядочить по возрастанию номера определяющих их вершин. После этого упорядочение всего массива граней (рёбер или конечных элементов) можно выполнить очень быстро и с использованием *только целочисленных* сравнений номеров вершин.

Окончательная же нумерация *всех* базисных функций может быть выполнена уже совсем просто. Первый номер даётся базисной функции, ассоциированной с вершиной с первым номером. Далее нумеруются все базисные функции, ассоциированные с рёбрами, гранями и конечными элементами, содержащими эту вершину. Следующий после них номер даётся базисной функции, ассоциированной со второй вершиной. За ней нумеруются все базисные функции, ассоциированные с рёбрами, гранями и конечными элементами, у которых *минимальный* номер определяющих их вершин равен двум и т.д. (т.е. после нумерации базисной функции, ассоциированной с i-й вершиной, нумеруются все базисные функции, ассоциированные с рёбрами, гранями и конечными элементами, у которых *минимальный* номер определяющих их вершин равен i (если таковые ещё есть)). Последний n_ψ-й номер получит базисная функция, ассоциированная с n_B-й вершиной (где n_B – количество *вершин* в сетке).

Заметим, что при использовании такого алгоритма глобальной нумерации базисных функций имеет смысл использовать и соответствующие алгоритмы работы с гранями, рёбрами и конечными элементами, которые упорядочивают их по *минимальным номерам* вершин, определяющих эти грани, рёбра и конечные элементы. Очевидно, что рассмотренный в данном разделе первый алгоритм построения списка смежных по граням конечных элементов очень легко модифицировать с учётом того, что грань определяется не координатами расположенной на ней точки, а номерами вершин. В этом случае при упорядочении граней будут выполняться только целочисленные сравнения, а после работы этого алгоритма мы получим *уже упорядоченный* массив граней, готовый к использованию в рассмотренном только что алгоритме упорядочения *всех* базисных функций (ассоциированных с вершинами, рёбрами, гранями и конечными элементами). Это тре-

бование (упорядоченность по минимальным номерам вершин) может быть без особого труда выполнено и при построении списка рёбер.

2.3.3. ПОСТРОЕНИЕ СПИСКА РЁБЕР КОНЕЧНЫХ ЭЛЕМЕНТОВ И ИХ НУМЕРАЦИЯ

Рассмотренный в разделе 2.3.2 метод нумерации граней через геометрическое упорядочение узлов, ассоциированных с этими гранями, в принципе, можно было бы использовать и для нумерации рёбер конечных элементов. Но даже в двумерном случае такой алгоритм не очень эффективен как по памяти, так и по быстродействию. В трёхмерном же случае он оказывается крайне неэффективным, поскольку каждое ребро может принадлежать большому количеству конечных элементов, что приведёт к резкому увеличению длины массива узлов из-за повторных их записей (три вещественные координаты каждого узла будут записаны столько раз, скольким конечным элементам принадлежит ребро), и, следовательно, к чрезмерно большим вычислительным затратам на упорядочение очень длинного *вещественного* массива. Поэтому мы рассмотрим другой, гораздо более эффективный алгоритм нумерации ребер, который пригоден как в двумерном, так и в трёхмерном случае, очень быстр и экономичен по памяти.

Обратим внимание на то, что портрет матрицы конечноэлементной СЛАУ в разреженном строчно-столбцовом формате (для элементов первого порядка) фактически содержит информацию о наличии связей между базисными функциями. Действительно, если для некоторых i и j (пусть для определённости $i > j$) между глобальными функциями ψ_i и ψ_j существует связь (т.е. существует конечный элемент, на котором их произведение или произведения их производных не равны нулю), то элемент A_{ij} конечноэлементной матрицы $\mathbf{A}$ не равен нулю. Тогда в массиве jg по адресам от $ig(i)$ до $ig(i+1)-1$ (включительно) должно быть значение, равное j. Обозначим через r (лежащее в диапазоне $ig\left(i\right) \le r < ig\left(i+1\right)$) такой номер элемента в массиве jg, что $jg\left(r\right) = j$. Очевидно, что для каждой связи между базисными функциями существует своё значение r, и поэтому это значение может рассматриваться как *номер* соответствующей связи. Количество таких связей, очевидно, совпадает с длиной массива jg.

Таким образом, алгоритм построения портрета является фактически алгоритмом подсчёта и нумерации связей между глобальными базисными функциями. Поэтому его достаточно легко модифицировать так, чтобы он подсчитывал и ну-

меровал ребра элементов, которые в каком-то смысле тоже являются «связями» между вершинами элементов. Для этого нужно везде в приведённой на с. 66–68 программе заменить вызов функции `NumberOfUnknowns(m)`, возвращающей число локальных базисных функций на элементе с номером `m`, на вызов функции, возвращающей число вершин элемента, а вызов функции `IndexOfUnknown(m,i)`, возвращающей глобальный номер базисной функции с локальным номером i, на вызов функции, возвращающей номер геометрической вершины в конечноэлементной сетке. Кроме того, для каждой пары вершин элемента нужно проверить, есть ли между ними ребро, для чего в двух местах программы после строки

```
ind2=IndexOfUnknown(NumElem,j)
```

достаточно вставить строку

```
IF(.NOT.IsEdgeExist(NumElem,ind1,ind2))CYCLE,
```

где логическая функция `IsEdgeExist` возвращает значение `.TRUE.` в том случае, если у элемента с номером `NumElem` есть ребро между вершинами `ind1` и `ind2`.

Итак, если нам нужно определить номер ребра, вершины которого имеют номера i и j ($i > j$), то с помощью массивов ig и jg мы можем сделать это так же, как, например, определялся номер ячейки массива gg, содержащей значение компоненты A_{ij} матрицы, хранящейся в разреженном формате. В этом случае номером ребра будет номер r (лежащей в диапазоне $ig\big(i\big) \le r < ig\big(i+1\big)$) ячейки массива jg, в котором хранится значение j.

При этом дополнительную информацию о каждом из рёбер сетки (например, о количестве базисных функций, ассоциированных с этим ребром, или о номерах этих базисных функций) можно хранить в структурах данных, аналогичных тем, которые используются для хранения матриц в разреженных форматах.

Очевидно, что рассмотренный алгоритм (практически совпадающий с алгоритмом построения портрета нижнего треугольника матрицы в разреженном *строчном* формате) даёт упорядочение рёбер по *максимальным номерам* определяющих их вершин. Мы предлагаем читателю в качестве упражнения написать программу формирования рёбер, упорядоченных по минимальным номерам определяющих их вершин.

2.4. АЛГОРИТМЫ РАБОТЫ С КОНЕЧНОЭЛЕМЕНТНЫМИ РЕШЕНИЯМИ

Под конечноэлементным решением u^h мы, как обычно, понимаем функцию, являющуюся линейной комбинацией базисных функций, причём веса в этой линейной комбинации определяются из решения конечноэлементной СЛАУ. Естественно, не возникает никаких проблем с получением значений в точках, являющихся вершинами элементов, поскольку для всех рассмотренных ранее типов элементов значение в вершине элемента совпадает с одним из весов (для лагранжевых столь же легко получаются значения во всех узлах – это значение компоненты вектора весов, номер которой совпадает с номером узла).

Однако очень часто требуется получить значение решения краевой задачи не в узле сетки, а в произвольной точке пространства, и при этом может понадобиться получить даже не само решение, а его производные или интегралы от него или его частных производных по отдельным подобластям или их границам. Здесь мы рассмотрим именно такого рода проблемы, связанные с обработкой конечноэлементных решений, и возможные пути их решения.

2.4.1. ПОИСК КОНЕЧНОГО ЭЛЕМЕНТА В НЕРЕГУЛЯРНОЙ СЕТКЕ

При использовании *нерегулярных* конечноэлементных сеток основной проблемой выдачи значений решения в произвольной точке пространства (после того как найден вектор весов разложения по базисным функциям) является поиск конечного элемента, в который попадает эта точка. Действительно, когда элемент в сетке найден, то значение решения в любой его точке вычисляется как значение в ней линейной комбинации локальных базисных функций найденного конечного элемента, причём коэффициентами этой линейной комбинации являются компоненты вектора весов $\mathbf{q}$ с номерами, определяемыми по соответствию глобальной и локальной нумерации узлов (точнее, базисных функций) на найденном конечном элементе.

Естественно, делать полный перебор всех конечных элементов с проверкой, в какой из них попала точка, можно только для сеток с небольшим числом элементов. При работе же с более или менее подробными сетками, вычислительные затраты при использовании такого подхода могут стать неприемлемо большими.

Повысить эффективность поиска элементов позволяет создание специальной вспомогательной структуры данных для хранения информации о местоположении

ячеек конечноэлементной сетки в пространстве. Покажем, как можно реализовать такой подход с использованием специальной структуры для двумерной конечно-элементной сетки.

Ускорить поиск конечных элементов можно, основываясь на следующем очевидном факте. Необходимым условием попадания точки в сам конечный элемент является условие попадания её в содержащий этот элемент прямоугольник, стороны которого параллельны осям координат.

Создадим специальный список непересекающихся прямоугольников, каждому из которых будет соответствовать массив ссылок на определённое число конечных элементов (в виде их номеров). Эти непересекающиеся прямоугольники (из списка) далее будем называть *основными*. Основной прямоугольник должен содержать ссылку на конечный элемент, если центр масс этого элемента попадает в него. Проходя по всем конечным элементам, будем приписывать ссылки на них к основным прямоугольникам. Сами же основные прямоугольники будем динамически переформировывать следующим образом. В случае появления слишком большого количества ссылок (превосходящего некоторое заранее фиксированное число) на конечные элементы в соответствующем основному прямоугольнику массиве этот прямоугольник разбивается пополам прямой, параллельной одной из координатных осей. При этом оси должны чередоваться – при каждом новом разбиении прямоугольника должна выбираться та ось, которая не была использована при разбиении того прямоугольника, из которого данный прямоугольник был получен. В качестве начального списка основных прямоугольников при таком способе их формирования можно взять список, содержащий один прямоугольник, описанный вокруг всей расчётной области.

Для того чтобы можно было определить попадание точки внутрь самих конечных элементов, после формирования списка основных прямоугольников припишем к каждому основному прямоугольнику ещё и внешний прямоугольник. Границы этого внешнего прямоугольника зададим так, чтобы он содержал все прямоугольники, *описанные* вокруг каждого конечного элемента, соответствующего основному прямоугольнику. Теперь непопадание точки внутрь описанного вокруг конечного элемента прямоугольника можно предварительно контролировать через непопадание этой точки во внешний прямоугольник вспомогательной структуры (непопадание точки во внешний прямоугольник гарантирует непопадание этой точки внутрь любого конечного элемента, ссылка на который хранится в соответствующем этому внешнему прямоугольнику основном прямоугольнике).

Очевидно, что рассматриваемый метод хранения двумерных элементов требует значительных накладных расходов, поскольку при его реализации необходима дополнительная память для хранения как самих прямоугольников, так и соответствующих им массивов ссылок на конечные элементы. Кроме того, при попадании точки во внешний прямоугольник необходимо просматривать все конечные элементы, ссылки на которые содержатся в соответствующем массиве. Однако если точек, в которых требуется выдать конечноэлементное решение, достаточно много, накладные расходы компенсируются существенным ускорением поиска нужных конечных элементов.

Отметим, что рассмотренный алгоритм построения вспомогательного списка прямоугольников очень удобно использовать для сеток с локальными сгущениями, к которым он адаптируется автоматически. Если же конечноэлементная сетка в достаточной степени регулярная, то этот алгоритм может быть существенно упрощен, например, за счёт генерации вспомогательных прямоугольников одинаковых размеров.

Обратим также внимание на то, что рассмотренный алгоритм поиска конечных элементов с использованием вспомогательной структуры данных вполне пригоден и для трёхмерных конечноэлементных сеток, только вместо прямоугольников в нём должны быть использованы прямоугольные параллелепипеды.

И конечно, очень эффективным способом выдачи значений конечноэлементного решения, полученного на нерегулярной сетке, является способ выдачи решения, полученного на сетке с четырёхугольными ячейками, с переинтерполцией его на регулярную прямоугольную сетку. Этот способ, очевидно, с тем же успехом может быть использован и при работе с ячейками другой формы – треугольными, тетраэдральными, шестигранными (в трёхмерном случае переинтерполяция делается, естественно, на регулярную сетку с ячейками в виде прямоугольных параллелепипедов). Он практически снимает проблему поиска нужной ячейки сетки в исходной нерегулярной сетке. При этом на регулярной сетке из прямоугольников или параллелепипедов можно использовать более гладкие базисные функции (эрмитовы), что позволит сразу же и существенно улучшить гладкость получаемого решения.

Таким же способом можно выдавать и значения производных конечноэлементного решения.

3. КОМБИНИРОВАННЫЕ И НЕСОГЛАСОВАННЫЕ СЕТКИ

3.1. СОГЛАСОВАННЫЕ И НЕСОГЛАСОВАННЫЕ СЕТКИ И КОНЕЧНЫЕ ЭЛЕМЕНТЫ

Несмотря на то, что у прямоугольных билинейных и треугольных линейных элементов порядок аппроксимации одинаков (поскольку одинаков максимальный порядок полностью представимых через их базисные функции полиномов), тем не менее при использовании сеток с одинаковым числом узлов (т.е. имеются в виду треугольные сетки, полученные из прямоугольных разбиением каждого прямоугольника на два треугольника) конечноэлементное решение на прямоугольных элементах довольно часто оказывается заметно точнее, чем на треугольных. Однако с помощью треугольных элементов можно существенно точнее (по сравнению с прямоугольниками) описать криволинейные границы, и поэтому для задач, решаемых в областях с криволинейными границами, гораздо выгоднее использовать треугольные элементы.

В ситуациях же, когда расчётная область состоит в основном из прямоугольников, но при этом имеет и криволинейные границы, оказывается выгодно использовать комбинированные сетки, в которых внутри областей используются прямоугольные элементы, а около криволинейных границ – треугольные. Метод конечных элементов может быть без особых проблем применён для решения задач на такого рода сетках. При этом для так называемых *согласованных* сеток не возникает никаких проблем с построением соответствующих конечноэлементных аппроксимаций.

До сих пор мы неявно полагали, что все ячейки конечноэлементных сеток являются геометрически одинаковыми (например, треугольниками или прямоугольниками) и на них однотипно определено одинаковое количество локальных базисных функций. В принципе же, при решении многих задач иногда удобно использовать конечные элементы не только с ячейками разного типа (прямоуголь-

ники с треугольниками, тетраэдры с призмами или с параллелепипедами и т.п.), но даже и с разнотипными локальными базисными функциями. В связи с этим имеет смысл уточнить понятия «сетка» и «конечный элемент», которые мы до сих пор использовали интуитивно.

В дальнейшем будем считать, что **сетка** – это объединение непересекающихся по своим внутренним точкам *ячеек*. При этом под **ячейкой сетки** будем понимать некоторую подобласть расчётной области, у которой есть *границы* и *узлы*. Под **узлами** будем понимать множество выделенных точек ячейки, используемых как для описания геометрии ячейки, так и для определения базисных функций на ней. Часть узлов, определяющих геометрию ячейки, будем называть **геометрическими вершинами**. И, наконец, под **конечным элементом** будем понимать ячейку сетки с заданными на ней локальными базисными функциями. Здесь, как и ранее, под **расчётной областью** мы понимаем (в зависимости от контекста использования этого термина) либо область Ω, в которой решается краевая задача, либо её аппроксимацию Ω^h в виде объединения всех ячеек сетки. При этом подразумевается, что границы Ω^h аппроксимируют соответствующие границы Ω (либо полностью совпадают с ними), т.е. в Ω^h нет дополнительных границ, появившихся за счёт «дырок» в сетке, а некоторое несовпадение границ Ω^h и Ω может получиться исключительно из-за искривлённости границ Ω.

Заметим, что ранее мы называли ячейки сетки конечными элементами, явно не разделяя этих понятий, поскольку на всех ячейках базисные функции определялись однотипно (тем не менее мы не считали эти понятия полностью эквивалентными, всегда подразумевая, что конечный элемент не определяется без заданных на нём базисных функций). Такое разделение становится необходимым при работе именно с *несогласованными сетками*, поскольку на несогласованной сетке можно за счёт специального определения базисных функций определить *согласованные* конечные элементы.

В дальнейшем под **согласованной сеткой** будем понимать сетку, для ячеек которой выполняется условие: любая граница одной ячейки либо точно совпадает с границей только одной другой ячейки, либо является фрагментом внешней границы расчётной области. При этом под границами ячеек, например, у треугольников, понимаются три его стороны, у прямоугольника – четыре его стороны, у многогранника – его грани и т.д.

На рис. 23 приведены примеры согласованной и несогласованной треугольной сетки. Сетка на рис. 23,*б* является несогласованной потому, что сторона Γ_1 треугольника Ω_1 не совпадает (точно) с какой-либо *одной* стороной другого треугольника. Заметим, что, как и ранее, в дальнейшем мы будем называть стороны ячеек двумерной сетки (треугольников и прямоугольников), также и *рёбрами* (как и стороны граней многогранников), т.е. под **ребром** мы всегда будем понимать линию, соединяющую две вершины ячейки сетки как в двумерном, так и в трёхмерном случае.

В этом разделе мы уделим особое внимание возможности использования *несогласованных сеток* для решения краевых задач. При этом конечноэлементная аппроксимация на *несогласованных сетках* будет строиться путём *согласования конечных элементов*, т.е. глобальные базисные функции будут формироваться из локальных функций конечных элементов так, чтобы все функции порождаемого ими конечномерного пространства были непрерывными во всей расчётной области и поэтому *допустимыми* для вариационной формулировки. Фактически для выполнения этого требования необходимо будет добиваться того, чтобы глобальные базисные функции были непрерывными на границах конечных элементов.

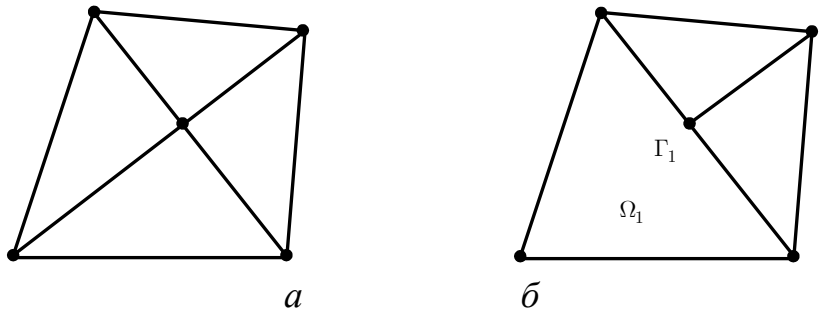


Рис. 23. Примеры согласованной и несогласованной сеток

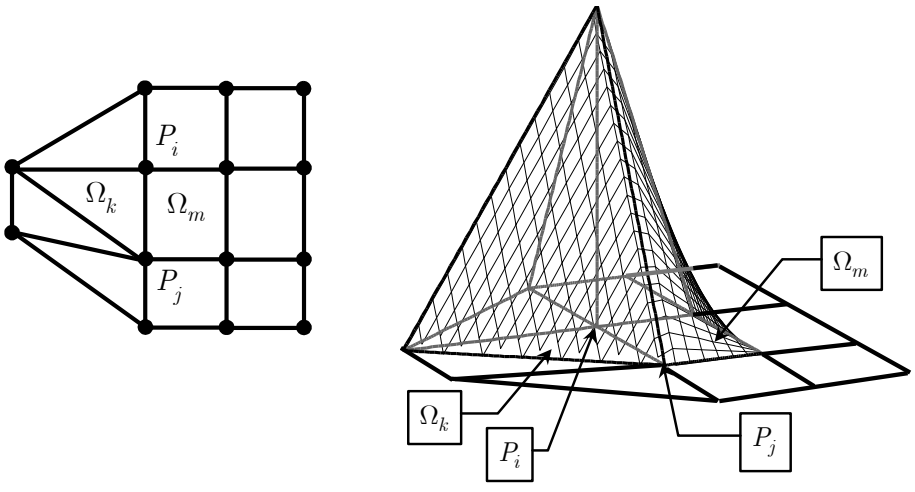
3.2. КОМБИНИРОВАННЫЕ СОГЛАСОВАННЫЕ СЕТКИ ИЗ ТРЕУГОЛЬНИКОВ И ПРЯМОУГОЛЬНИКОВ С БАЗИСНЫМИ ФУНКЦИЯМИ ОДНОГО ПОРЯДКА

Рассмотрим ситуацию, когда согласованная конечноэлементная сетка состоит из треугольников и прямоугольников с определёнными на них локальными базисными функциями первого порядка, т.е. на треугольниках заданы линейные

функции, а на прямоугольниках – билинейные. Как и в случае сеток с однотипными элементами (т.е. состоящих из только треугольных или только прямоугольных ячеек) каждая глобальная базисная функция привязывается к узлу сетки и определяется через локальные базисные функции прилегающих к этому узлу конечных элементов.

Тогда очевидно, что базисная функция ψ_i в узле P_i, к которому примыкают как треугольные, так и прямоугольные конечные элементы (фрагмент сетки с таким узлом P_i и вид глобальной базисной функции ψ_i, например, изображён на рис. 24), окажется непрерывной на всех границах между элементами. Действительно, на ребре Γ_{ij}, соединяющем узлы P_i и P_j и являющемся границей между треугольником Ω_k и прямоугольником Ω_m, обе соответствующие узлу P_i локальные базисные функции элементов Ω_k и Ω_m являются одномерными линейными функциями, равными единице в узле P_i и нулю в узле P_j.

Поэтому глобальная базисная функция ψ_i, определяемая на каждом из элементов Ω_k и Ω_m как соответствующая локальная базисная функция этих элементов (т.е. так же, как это обычно делается для сеток с однотипными элементами), является непрерывной на границе Γ_{ij} между элементами Ω_k и Ω_m.



а *б*

Рис. 24. Фрагмент комбинированной сетки (*а*)
и вид глобальной базисной функции ψ_i (*б*)

Таким образом, глобальная матрица и вектор конечноэлементной СЛАУ могут быть стандартным образом собраны из локальных матриц и векторов тре-

угольных и прямоугольных элементов по соответствию локальных номеров базисных функций глобальным (или локальных номеров узлов элементов глобальным номерам узлов сетки).

То же самое справедливо и при использовании комбинированной сетки с треугольными и прямоугольными элементами более высокого порядка (т.е. с квадратичными базисными функциями на треугольниках и биквадратичными базисными функциями на прямоугольниках, кубическими на треугольниках и бикубическими на прямоугольниках и т.д.). В этом случае глобальные базисные функции также оказываются непрерывными на рёбрах конечных элементов при сшивке соответствующих одному и тому же узлу локальных функций, поскольку эти локальные функции являются на ребре одинаковыми одномерными полиномами. Обратим внимание на то, что при построении квадратичных и кубических лагранжевых базисных функций введённые нами на рёбрах треугольников и прямоугольников дополнительные узлы были поставлены одинаково – это и обеспечивает совпадение соответствующих одномерных полиномов на ребре, по которому граничат треугольник и прямоугольник.

3.3. ОБ ИСПОЛЬЗОВАНИИ НЕСОГЛАСОВАННЫХ СЕТОК В МКЭ

На практике несогласованные сетки применяются обычно тогда, когда в качестве ячеек сетки используются прямоугольники, параллелепипеды, четырёхугольники, шестигранники или призмы. Для двумерных сеток с треугольными ячейками несогласованные сетки, как правило, используются очень редко, поскольку вместо процедуры согласования конечных элементов на несогласованной треугольной сетке можно без особого труда построить согласованную сетку, доразбив ячейки с вершинами внутри рёбер на более мелкие треугольники (и избавиться тем самым от вершин внутри рёбер). То же самое можно сказать и о сетках с тетраэдральными ячейками.

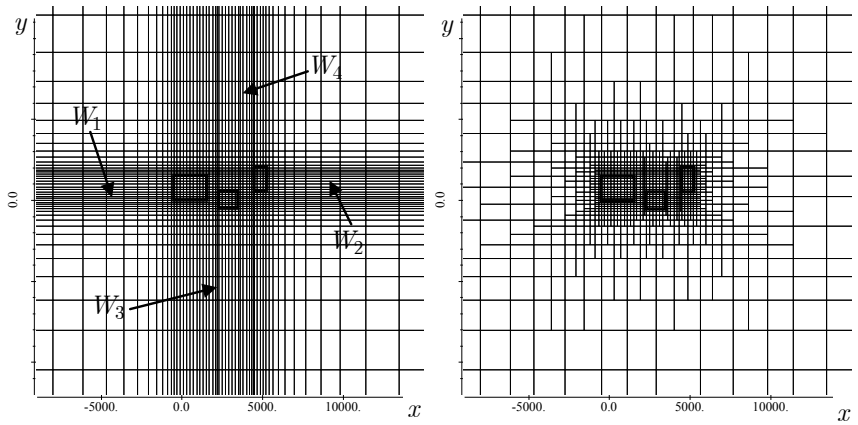
Далее мы рассмотрим процедуры построения согласованных конечных элементов для сеток с прямоугольными ячейками. При этом мы рассмотрим два способа построения согласованных конечных элементов, которые будут продемонстрированы на соответствующих примерах. Первый из них базируется на том, что используются *локальные матрицы стандартных* прямоугольных элементов и согласование элементов осуществляется фактически на этапе сборки глобальной матрицы, а второй – на введении так называемых *переходных* конечных элементов

специального вида, для которых строятся специальные локальные базисные функции и соответственно локальные матрицы.

3.4. НЕСОГЛАСОВАННЫЕ СЕТКИ С ПРЯМОУГОЛЬНЫМИ ЯЧЕЙКАМИ

Для решения практических задач, как правило, используются неравномерные сетки, поскольку они позволяют заметно уменьшить вычислительные затраты на получение решения необходимой точности. При этом если используются сетки с прямоугольными ячейками, то можно дополнительно снизить вычислительные затраты, избавившись от так называемых «*лишних*» *узлов*, которые практически не влияют на точность решения задачи. Такие («лишние») узлы часто появляются при использовании регулярных неравномерных сеток с прямоугольными ячейками. Причины появления такого рода узлов поясним на следующем примере.

На рис. 25,*а* показана регулярная прямоугольная сетка для решения задачи с тремя прямоугольными объектами в прямоугольной расчётной области (подобласти-объекты изображены на этом рисунке жирными линиями). Эта сетка была получена следующим образом. На координатных осях были поставлены точки $\{x_i\}$ и $\{y_j\}$, через них проведены координатные линии $x = x_i$ и $y = y_j$ и получены ячейки сетки в виде прямоугольников $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$. При этом точки $\{x_i\}$ и $\{y_j\}$ были поставлены так, чтобы объекты были разбиты на мелкие ячейки, а с удалением от объектов шаг дискретизации плавно увеличивался (с коэффициентом разрядки $k_r = 1.2$, т.е. с удалением от объектов каждый последующий шаг больше предыдущего в k_r раз). Очевидно, что при использовании такой регулярной прямоугольной сетки в отдельных местах расчётной области появляются ячейки, имеющие довольно большой размер по одной координате и очень маленький по другой – это подобласти W_1, W_2, W_3, W_4. Таким образом, изображённая на рис. 25,*а* сетка содержит довольно много узлов, которые никак не улучшают качество аппроксимации и только увеличивают вычислительные затраты. Эти узлы могут быть удалены, как это, например, показано на рис. 25,*б*, но получаемая сетка будет несогласованной.

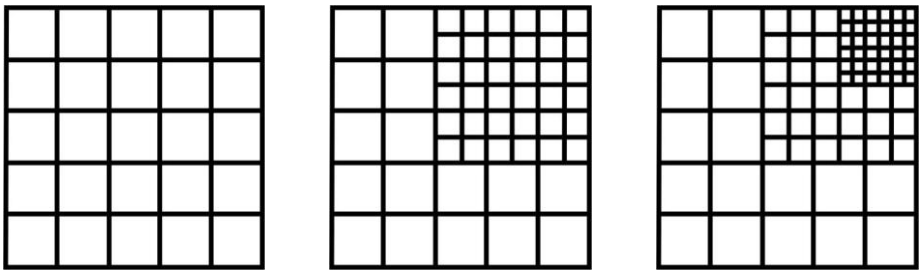


а *б*

Рис. 25. Удаление «лишних» узлов из регулярной прямоугольной сетки: *а* – стандартная регулярная сетка; *б* – нерегулярная прямоугольная сетка, полученная после удаления «лишних узлов»

Несогласованные прямоугольные сетки используются и в тех случаях, когда необходимо повысить точность конечноэлементного решения за счёт локальных вложенных дроблений сетки. Пример исходной регулярной сетки показан на рис. 26,*а*, а двух сеток, полученных после одного и двух вложенных локальных дроблений – на рис. 26,*б* и рис. 26,*в* соответственно.

Очевидно, что в получившейся несогласованной сетке присутствуют узлы двух типов: обычные узлы, являющиеся *только* узлами ячеек (одной или нескольких), и так называемые *терминальные* узлы. **Терминальный узел** узел сетки:– терминальный"– это узел, находящийся на границе хотя бы одной стандартной ячейки, узлом которой он не является (см. рис. 27).



а *б* *в*

Рис. 26. Пример регулярной сетки (*а*) и нерегулярных сеток, полученных после одного (*б*) и двух (*в*) вложенных локальных дроблений

Заметим, что если в двумерном случае несогласованные сетки с прямоугольными ячейками можно легко преобразовать к согласованным, разбив отдельные ячейки сетки на треугольники, и после этого строить конечноэлементное решение на комбинированной сетке, то в трёхмерном случае сделать совсем непросто. Поэтому имеет смысл рассмотреть возможности построения конечноэлементных аппроксимаций на несогласованной сетке с прямоугольными ячейками (в *трёхмерном случае* сетка с ячейками в виде *прямоугольных параллелепипедов* часто тоже называется *прямоугольной*, мы уже пользовались этим и будем пользоваться далее).

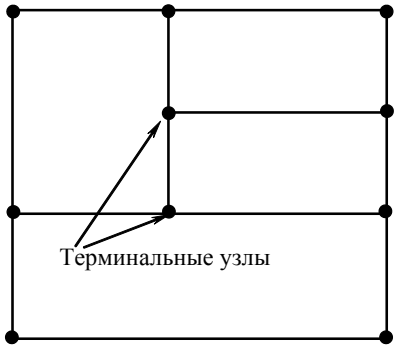


Рис. 27. Пример сетки с терминальными узлами

Способы согласования конечных элементов на несогласованных сетках, в принципе, мало различаются в двумерном и трёхмерном случаях. Поэтому идею такого согласования мы рассмотрим для двумерного случая. Вначале же на небольшом примере поясним, почему на несогласованной сетке при использовании обычной процедуры сборки конечноэлементной СЛАУ из стандартных локальных матриц получаются несогласованные конечные элементы.

3.5. ПРИМЕР НЕСОГЛАСОВАННОЙ СТЫКОВКИ ЭЛЕМЕНТОВ НА НЕСОГЛАСОВАННОЙ ПРЯМОУГОЛЬНОЙ СЕТКЕ

Пусть сетка состоит из трёх прямоугольных элементов Ω_1, Ω_2 и Ω_3, как это изображено на рис. 28. На этих элементах введём стандартные билинейные локальные базисные функции $\hat{\psi}_i^{\Omega_k}$, $i=1\ldots4$, $k=1\ldots3$. Номера этих локальных базисных функций соответствуют локальным номерам узлов на каждом конечном элементе, которые показаны на рис. 28 в кружках. При этом глобальные номера узлов сетки указаны в прямоугольничках.

Очевидно, что на правой границе Γ элемента Ω_1 (т.е. на линии, соединяющей узлы сетки с глобальными номерами 2 и 7) локальные базисные функции элементов Ω_1 и граничащих с ним элементов Ω_2 и Ω_3, соответствующие одним и тем же узлам сетки, не совпадают. Действительно, функция $\hat{\psi}_2^{\Omega_1}$ линейно изменяется от значения 1 в узле с глобальным номером 2 до значения 0 в узле с глобальным но-

мером 7. Соответствующая этому же узлу (с глобальным номером 2) функция $\hat{\psi}_1^{\Omega_2}$ на Γ линейно изменяется от значения 1 в узле с глобальным номером 2 до значения 0 в узле с глобальным номером 4 (т.е., например, в узле с глобальным номером 4 функция $\hat{\psi}_2^{\Omega_1}$ равна 0.5, а функция $\hat{\psi}_1^{\Omega_2}$ – нулю). Для узла же с глобальным номером 4, который является *терминальным*, на элементе Ω_1 вообще нет соответствующей локальной базисной функции.

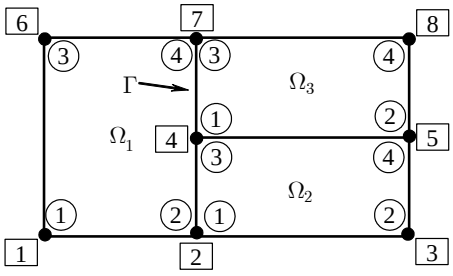


Рис. 28. Пример несогласованной сетки

Если для рассматриваемого случая собрать матрицу и вектор конечноэлементной СЛАУ из локальных матриц и векторов элементов Ω_1, Ω_2 и Ω_3 обычным способом (т.е. по соответствию глобальной и локальной нумерации), то полученная конечноэлементная СЛАУ фактически будет аппроксимировать исходную вариационную задачу на пространстве с разрывными базисными функциями. Действительно, такой сборке соответствует глобальная базисная функция ψ_2^{nc}, линейная на границе Γ со стороны элемента Ω_1 и кусочно-линейная со стороны элементов Ω_2 и Ω_3 (верхний индекс «nc» у ψ_2^{nc} означает, что эта функция *не непрерывна*).

То же самое можно сказать и о глобальной функции ψ_7^{nc}. Базисная же функция ψ_4^{nc} на ребре Γ со стороны Ω_1 вообще равна нулю, поскольку на этом элементе нет соответствующей рассматриваемому узлу локальной функции.

Таким образом, соответствующий стандартной сборке базис $\left\{\psi_i^{nc}\right\}$ содержит разрывные функции, и поэтому функции порождаемого им пространства *не являются допустимыми* для вариационной задачи, а конечные элементы становятся из-за этого *несогласованными*.

Существует два способа, позволяющих на такой сетке построить базис $\left\{\psi_i^{c}\right\}$ из непрерывных функций и тем самым согласовать конечные элементы. Первый способ будет заключаться в построении базиса $\left\{\psi_i^{c}\right\}$ без изменения локальных базисных функций элементов (т.е. терминальным узлам на элементе не будут ставиться в соответствие никакие локальные базисные функции), а второй – во введении стыковочных элементов со специальными локальными базисными функциями, соответствующими терминальным узлам.

3.6. ПОСТРОЕНИЕ СОГЛАСОВАННЫХ КОНЕЧНЫХ ЭЛЕМЕНТОВ С НЕСОГЛАСОВАННЫМИ ЛОКАЛЬНЫМИ БАЗИСНЫМИ ФУНКЦИЯМИ

Здесь мы рассмотрим первый из упомянутых в конце раздела 3.5 способов построения согласованных конечных элементов.

3.6.1. ДЕМОНСТРАЦИЯ РАССМАТРИВАЕМОГО СПОСОБА СОГЛАСОВАНИЯ НА ПРОСТЕЙШЕЙ ПРЯМОУГОЛЬНОЙ СЕТКЕ

Для сетки, изображённой на рис. 28, построим непрерывные глобальные базисные функции, соответствующие узлам с глобальными номерами 2 и 7, следующим образом. Для простоты будем считать, что терминальный узел с глобальным номером 4 стоит точно посередине ребра Γ. Функцию ψ_2^c определим как $\psi_2^{nc} + \frac{1}{2}\psi_4^{nc}$, а функцию ψ_7^c определим как $\psi_7^{nc} + \frac{1}{2}\psi_4^{nc}$. Нетрудно убедиться, что полученные глобальные базисные функции ψ_2^c и ψ_7^c являются линейными функциями на ребре Γ как со стороны Ω_1, так и со стороны $\Omega_2 \cup \Omega_3$, значения каждой из них в узлах с глобальными номерами 2 и 7 со стороны Ω_1 совпадают со значениями со стороны $\Omega_2 \cup \Omega_3$, и поэтому эти функции непрерывны на Γ.

Базисные же функции ψ_i^c для $i = 1,3,5,6,8$ определим равными ψ_i^{nc}. Таким образом, в базисе $\left\{\psi_i^c\right\}$ содержится только 7 базисных функций. При этом для того чтобы одинаковые функции в базисах $\left\{\psi_i^c\right\}$ и $\left\{\psi_i^{nc}\right\}$ имели одинаковые номера, в базисе $\left\{\psi_i^c\right\}$ базисные функции пронумерованы не подряд – в нём отсутствует базисная функция с номером 4. Соответственно и вектор искомых весов разложения по этому базису будет иметь 7 компонент, при этом их номера, естественно, должны быть равны номерам базисных функций, и поэтому компонента q_4 вектора весов $\mathbf{q} = \left(q_1,\ldots,q_8\right)^{\mathrm{T}}$ фактически будет фиктивной.

Построим конечноэлементную аппроксимацию с использованием базиса $\left\{\psi_i^c\right\}$. Для этого, как обычно, в эквивалентной вариационной постановке (в форме Галёркина) искомое решение u заменим линейной комбинацией базисных функций $u^h = \sum_{i,\, i\neq 4} q_i \psi_i^c$ и вместо пробной функции в вариационное уравнение будем

подставлять поочерёдно базисные функции $\left\{\psi_i^c\right\}$. Очевидно, что если считать номера уравнений совпадающими с номерами базисных функций, то в получаемой конечноэлементной СЛАУ будет отсутствовать уравнение с номером 4.

Таким образом, мы получим следующую систему уравнений (мы полагаем, что вкладов от естественных краевых условий нет, поскольку они никак не повлияют на наши рассуждения – на внешних границах все ψ_i^c просто совпадают с ψ_i^{nc}):

$$\sum_{\substack{j=1\\ j\neq 4}}^{8}\left(\int_\Omega \lambda \operatorname{grad}\psi_i^c\cdot\operatorname{grad}\psi_j^c d\Omega+\int_\Omega \gamma\psi_i^c\psi_j^c d\Omega\right)q_j=\int_\Omega f\psi_i^c d\Omega,\ \ i=1\ldots 8, i\neq 4\,.$$

Очевидно, что если номера i и j не равны 2 или 7, то компонента A_{ij}^c матрицы $\mathbf{A}^c$ этой СЛАУ совпадает с соответствующей компонентой A_{ij}^{nc} матрицы $\mathbf{A}^{nc}$ СЛАУ, собранной из локальных матриц по стандартной технологии (для уменьшения выкладок будем считать, что $\gamma=0$):

$$A_{ij}^c=A_{ij}^{nc}=\int_\Omega \lambda \operatorname{grad}\psi_i^{nc}\cdot\operatorname{grad}\psi_j^{nc}d\Omega,\ \ i,j=1,3,4,5,6,8\,.$$

Если же хотя бы один из номеров i или j равен 2 или 7, то A_{ij}^c не совпадает с A_{ij}^{nc}, но при этом A_{ij}^c может быть вычислен через компоненты матрицы $\mathbf{A}^{nc}$. Покажем это.

Рассмотрим случай, когда $i=2$, а $j\neq 2$ и $j\neq 7$.

$$A_{2j}^c=\int_\Omega \lambda \operatorname{grad}\psi_2^c\cdot\operatorname{grad}\psi_j^c d\Omega=\int_\Omega \lambda \operatorname{grad}\left(\psi_2^{nc}+\frac{1}{2}\psi_4^{nc}\right)\operatorname{grad}\psi_j^{nc}d\Omega=$$

$$=\sum_{k=1}^{3}\int_{\Omega_k} \lambda \operatorname{grad}\left(\psi_2^{nc}+\frac{1}{2}\psi_4^{nc}\right)\cdot\operatorname{grad}\psi_j^{nc}d\Omega\,.$$

Теперь, когда мы перешли на интегрирование по отдельным конечным элементам, мы можем заменить $\operatorname{grad}\left(\psi_2^{nc}+\frac{1}{2}\psi_4^{nc}\right)$ на $\operatorname{grad}\psi_2^{nc}+\frac{1}{2}\operatorname{grad}\psi_4^{nc}$. При этом обратим особое внимание на то, что

$$\int_\Omega \lambda \operatorname{grad}\psi_2^{nc}\cdot\operatorname{grad}\psi_j^{nc}d\Omega\neq\sum_{k=1}^{3}\int_{\Omega_k} \lambda \operatorname{grad}\psi_2^{nc}\cdot\operatorname{grad}\psi_j^{nc}d\Omega$$

из-за разрывности ψ_2^{nc} на границе Γ (между Ω_1 и $\Omega_2 \cup \Omega_3$). То же самое справедливо и для функции ψ_4^{nc}. Таким образом,

$$A_{2j}^c = \sum_{k=1}^{3} \int_{\Omega_k} \lambda \operatorname{grad} \psi_2^{nc} \cdot \operatorname{grad} \psi_j^{nc} d\Omega + \frac{1}{2} \sum_{k=1}^{3} \int_{\Omega_k} \lambda \operatorname{grad} \psi_4^{nc} \cdot \operatorname{grad} \psi_j^{nc} d\Omega \quad .$$

Учитывая, что компоненты матрицы $\mathbf{A}^{nc}$ были определены нами как

$$A_{ij}^{nc} = \sum_{k=1}^{3} \left(\int_{\Omega_k} \lambda \operatorname{grad} \psi_i^{nc} \cdot \operatorname{grad} \psi_j^{nc} d\Omega \right), \tag{3.1}$$

получаем

$$A_{2j}^c = A_{2j}^{nc} + \frac{1}{2} A_{4j}^{nc}, \qquad j \neq 2 \text{ и } j \neq 7. \tag{3.2}$$

Аналогично

$$A_{7j}^c = A_{7j}^{nc} + \frac{1}{2} A_{4j}^{nc}, \qquad j \neq 2 \text{ и } j \neq 7, \tag{3.3}$$

$$A_{i2}^c = A_{i2}^{nc} + \frac{1}{2} A_{i4}^{nc}, \qquad i \neq 2 \text{ и } i \neq 7, \tag{3.4}$$

$$A_{i7}^c = A_{i7}^{nc} + \frac{1}{2} A_{i4}^{nc}, \qquad i \neq 2 \text{ и } i \neq 7, \tag{3.5}$$

$$A_{22}^c = A_{22}^{nc} + \frac{1}{2} A_{24}^{nc} + \frac{1}{2} A_{42}^{nc} + \frac{1}{4} A_{44}^{nc}, \tag{3.6}$$

$$A_{27}^c = A_{27}^{nc} + \frac{1}{2} A_{24}^{nc} + \frac{1}{2} A_{47}^{nc} + \frac{1}{4} A_{44}^{nc}, \tag{3.7}$$

$$A_{72}^c = A_{72}^{nc} + \frac{1}{2} A_{74}^{nc} + \frac{1}{2} A_{42}^{nc} + \frac{1}{4} A_{44}^{nc}, \tag{3.8}$$

$$A_{77}^c = A_{77}^{nc} + \frac{1}{2} A_{74}^{nc} + \frac{1}{2} A_{47}^{nc} + \frac{1}{4} A_{44}^{nc}. \tag{3.9}$$

Таким образом, все компоненты матрицы $\mathbf{A}^c$ могут быть вычислены через компоненты матрицы $\mathbf{A}^{nc}$ (собранной из локальных матриц несогласованных элементов *стандартным способом*). Очевидно, что это возможно и для любой другой аналогичной ситуации, когда элементы непрерывного базиса $\left\{\psi_i^c\right\}$ формируются как линейные комбинации разрывных функций ψ_j^{nc}, получаемых формальным объединением локальных базисных функций соседних конечных элементов.

В разделе 3.6.2 мы получим соотношения, позволяющие вычислять компоненты матрицы $\mathbf{A}^c$ через компоненты матрицы $\mathbf{A}^{nc}$ для общего случая.

3.6.2. ОБЩАЯ СХЕМА ПОСТРОЕНИЯ КОНЕЧНОЭЛЕМЕНТНОЙ СЛАУ ПРИ СОГЛАСОВАНИИ ЭЛЕМЕНТОВ НА НЕСОГЛАСОВАННЫХ СЕТКАХ С ИСПОЛЬЗОВАНИЕМ МАТРИЦЫ ПЕРЕХОДА

Пусть базисные функции ψ_i^c выражаются через функции ψ_i^{nc} с помощью соотношения

$$\boldsymbol{\psi}^c = \mathbf{T}\boldsymbol{\psi}^{nc}, \tag{3.10}$$

где $\boldsymbol{\psi}^{nc}$ – вектор, составленный из функций ψ_i^{nc}, $i = 1\ldots n$ (n – число узлов в сетке), $\boldsymbol{\psi}^c$ – вектор, составленный из функций ψ_i^c, $i = 1\ldots n_c$, а $\mathbf{T}$ – матрица размера $n_c \times n$ перехода от базиса $\left\{\psi_i^{nc}\right\}$ к базису $\left\{\psi_i^c\right\}$. Тогда

$$\mathbf{A}^c = \int\limits_\Omega \left\{\lambda \operatorname{grad}\left(\boldsymbol{\psi}^c\right)\operatorname{grad}\left(\boldsymbol{\psi}^c\right)^{\mathrm{T}} + \gamma\left(\boldsymbol{\psi}^c\right)\left(\boldsymbol{\psi}^c\right)^{\mathrm{T}}\right\} d\Omega =$$

$$= \int\limits_\Omega \left\{\lambda \operatorname{grad}\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)\operatorname{grad}\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}} + \gamma\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}}\right\} d\Omega.$$

Чтобы вынести матрицу $\mathbf{T}$ (компоненты которой являются константами) за интеграл, необходимо перейти от интеграла по всей расчётной области к сумме интегралов по конечным элементам, поскольку $\boldsymbol{\psi}^c$ – непрерывные, а $\boldsymbol{\psi}^{nc}$ – разрывные функции, и для них (!)

$$\int\limits_\Omega \lambda \operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)\operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}} d\Omega \neq \sum_k \int\limits_{\Omega_k} \lambda \operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)\operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}} d\Omega.$$

Таким образом,

$$\mathbf{A}^c = \sum_k \int\limits_{\Omega_k} \left\{\lambda \operatorname{grad}\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)\operatorname{grad}\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}} + \gamma\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)\left(\mathbf{T}\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}}\right\} d\Omega =$$

$$= \sum_k \int\limits_{\Omega_k} \left\{\lambda \mathbf{T} \operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)\operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}} \mathbf{T}^{\mathrm{T}} + \gamma \mathbf{T}\boldsymbol{\psi}^{nc}\left(\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}} \mathbf{T}^{\mathrm{T}}\right\} d\Omega =$$

$$= \mathbf{T}\left(\sum_k \int\limits_{\Omega_k} \left\{\lambda \operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)\operatorname{grad}\left(\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}} + \gamma \boldsymbol{\psi}^{nc}\left(\boldsymbol{\psi}^{nc}\right)^{\mathrm{T}}\right\} d\Omega\right)\mathbf{T}^{\mathrm{T}}.$$

Учитывая, что матрица $\mathbf{A}^{nc}$ фактически была нами определена как

$$\mathbf{A}^{nc} = \sum_k \int_{\Omega_k} \left\{ \lambda \operatorname{grad}\left(\psi^{nc}\right) \operatorname{grad}\left(\psi^{nc}\right)^{\mathrm{T}} + \gamma \psi^{nc} \left(\psi^{nc}\right)^{\mathrm{T}} \right\} d\Omega, \tag{3.11}$$

окончательно получим

$$\mathbf{A}^{c} = \mathbf{T}\mathbf{A}^{nc}\mathbf{T}^{\mathrm{T}}. \tag{3.12}$$

Аналогично преобразуется и вектор правой части:

$$\mathbf{b}^{c} = \mathbf{T}\mathbf{b}^{nc}. \tag{3.13}$$

Скажем несколько слов о том, как можно вычислить компоненты матрицы перехода. Для этого сначала поясним, как *в общем случае* может быть построена функция ψ_i^c *при стыковке двух элементов с одним*. На ребре Γ, являющемся границей между одним элементом («большим») и объединением нескольких других («маленьких»), функция ψ_i^c определяется как базисная функция «большого» элемента. Поэтому фактически на этой границе мы должны представить каждую *ненулевую* базисную функцию ψ_i^c (которая на Γ является локальной базисной функцией «большого» элемента) через базисные функции ψ_j^{nc}, полученные стандартной сшивкой локальных базисных функций «маленьких» элементов. А это уже сделать довольно просто. Действительно, весами разложения линейной (как, впрочем, и любой полиномиальной) базисной функции «большого» элемента по кусочно-линейным (или соответственно по кусочно-полиномиальным *лагранжевым*) базисным функциям, полученным стандартной сшивкой локальных базисных функций «маленьких» элементов, являются *значения разлагаемой функции в узлах* «маленьких» элементов.

Например, для несогласованной сетки, изображённой на рис. 28, функция ψ_2^c в узлах, лежащих на границе Γ ($\Gamma = \Omega_1 \bigcap \left(\Omega_2 \bigcup \Omega_3\right)$), принимает значения $\psi_2^c\left(x_2, y_2\right) = 1$, $\psi_2^c\left(x_4, y_4\right) = 0.5$, $\psi_2^c\left(x_7, y_7\right) = 0$, и поэтому через базисные функции ψ_j^{nc} она выражается следующим образом:

$$\begin{aligned}\psi_2^c\left(x,y\right) &= \psi_2^c\left(x_2,y_2\right)\psi_2^{nc}\left(x,y\right) + \psi_2^c\left(x_4,y_4\right)\psi_4^{nc}\left(x,y\right) + \\ &+ \psi_2^c\left(x_7,y_7\right)\psi_7^{nc}\left(x,y\right) = \psi_2^{nc}\left(x,y\right) + 0.5\psi_4^{nc}\left(x,y\right).\end{aligned}$$

Отсюда становятся очевидными значения *ненулевых* компонент *второй* строки матрицы перехода $\mathbf{T}$: $T_{22} = 1$ и $T_{24} = 0.5$.

Аналогично вычисляются и компоненты седьмой строки матрицы $\mathbf{T}$.

Таким образом, для рассматриваемого нами примера (т.е. для сетки, изображённой на рис. 28) матрицу перехода $\mathbf{T}$ можно записать в виде:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (3.14)$$

Нетрудно убедиться, что соотношения (3.2)-(3.9) могут быть получены из (3.12) с помощью матрицы (3.14).

В матрице (3.14) четвёртая строка является полностью нулевой – это произошло потому, что мы хотели оставить у функций ψ_i^c (и соответственно у весов q_i) номера, равные номерам узлов сетки (заметим, что ψ_i^{nc} имеют именно такие номера). Очевидно, что в этом случае $n_c = n$ и матрица $\mathbf{A}^c$ имеет размер $n \times n$ (где n – число узлов в сетке) и при этом у неё имеются нулевые строки и нулевые столбцы, соответствующие терминальным узлам.

Рассмотрим, как можно преобразовать получаемую конечноэлементную СЛАУ, сделав её матрицу невырожденной. Простейший способ – поставить на места диагональных элементов нулевых строк единицы. Однако тогда значения весов с номерами, соответствующими нулевым строкам, получатся равными нулю (т.е. при вычислении решения на конечном элементе в виде линейной комбинации его локальных базисных функций этими весами всё равно нельзя воспользоваться, а в матрице СЛАУ остались «лишние» строки и столбцы).

Другой способ заключается в том, что матрица $\mathbf{T}$ сразу задаётся прямоугольной (т.е. $n_c \neq n$ и n_c – число нетерминальных узлов). Тогда номера функций ψ_i^c (и, естественно, весов) становятся *не равными* номерам узлов. В этом случае для рассматриваемого нами примера матрица $\mathbf{T}$ имела бы вид:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (3.15)$$

При таком способе построения преобразования $\mathbf{T}$ матрица $\mathbf{A}^c$ конечноэлементной СЛАУ уже не будет иметь нулевых строк и столбцов. И хотя в этом случае номера глобальных функций ψ_i^c (и соответственно компонент q_i^c вектора весов $\mathbf{q}^c = \left(\mathbf{A}^c\right)^{-1} \mathbf{b}^c$) становятся не равными номерам узлов, это не является большой проблемой для нахождения значений компонент вектора весов $\mathbf{q}^{nc}$, номера которых соответствуют глобальным номерам узлов, поскольку $\mathbf{q}^{nc}$ может быть получен из $\mathbf{q}^c$ с помощью матрицы преобразования $\mathbf{T}$:

$$\mathbf{q}^{nc} = \mathbf{T}^{\mathrm{T}} \mathbf{q}^c . \qquad (3.16)$$

Докажем справедливость соотношения (3.16). Действительно, нам нужно получить такой вектор $\mathbf{q}^{nc}$, который бы при скалярном умножении на ψ^{nc} давал конечноэлементное решение u^h, которое имеет вид $u^h = \left(\mathbf{q}^c\right)^{\mathrm{T}} \psi^c$ ($\mathbf{q}^c$ – вектор решения конечноэлементной СЛАУ), т.е. такой $\mathbf{q}^{nc}$, что

$$\left(\mathbf{q}^{nc}\right)^{\mathrm{T}} \psi^{nc} = \left(\mathbf{q}^c\right)^{\mathrm{T}} \psi^c . \qquad (3.17)$$

Нетрудно убедиться, что вектор $\mathbf{q}^{nc}$, вычисленный с помощью соотношения (3.16), удовлетворяет равенству (3.17) (с учётом связи (3.10)):

$$\left(\mathbf{q}^{nc}\right)^{\mathrm{T}} \psi^{nc} = \left(\mathbf{T}^{\mathrm{T}} \mathbf{q}^c\right)^{\mathrm{T}} \psi^{nc} = \left(\mathbf{q}^c\right)^{\mathrm{T}} \mathbf{T} \psi^{nc} = \left(\mathbf{q}^c\right)^{\mathrm{T}} \psi^c .$$

После вычисления же $\mathbf{q}^{nc}$ (с помощью соотношения (3.16)) не представляет никакой сложности вычислить значение конечноэлементного решения внутри любого конечного элемента – нужное нам значение будет линейной комбинацией значений *локальных* базисных функций (в этой точке), коэффициенты которой – это соответствующие компоненты вектора весов $\mathbf{q}^{nc}$ (полученные по соответствию локальных номеров базисных функций рассматриваемого конечного элемента номерам функций ψ_i^{nc}).

3.6.3. АЛГОРИТМ СБОРКИ МАТРИЦЫ КОНЕЧНОЭЛЕМЕНТНОЙ СЛАУ ИЗ ЛОКАЛЬНЫХ МАТРИЦ ЭЛЕМЕНТОВ С НЕСОГЛАСОВАННЫМИ ЛОКАЛЬНЫМИ БАЗИСНЫМИ ФУНКЦИЯМИ

Заметим, что при практической реализации довольно неудобно пытаться вычислять матрицу $\mathbf{A}^c$ напрямую по формуле (3.12) через предварительно собран-

ную матрицу $\mathbf{A}^{nc}$. Конечноэлементные матрицы обычно хранятся в упакованных форматах, учитывающих местоположение их ненулевых элементов, и поэтому форматы упаковки $\mathbf{A}^{c}$ и $\mathbf{A}^{nc}$ могут не совпадать. В принципе, можно вообще не собирать матрицу $\mathbf{A}^{c}$, а при реализации процедуры решения СЛАУ учитывать, что её матрица является произведением трёх матриц (см. (3.12)). Естественно, это приводит к заметному увеличению вычислительных затрат и определённым неудобствам при использовании некоторых схем предобусловливания. Поэтому более выгодными являются процедуры, позволяющие сразу (без формирования $\mathbf{A}^{nc}$) генерировать $\mathbf{A}^{c}$.

Покажем, как может быть организована процедура генерации матрицы $\mathbf{A}^{c}$, когда $\mathbf{A}^{c}$ собирается сразу из локальных матриц конечных элементов.

Пусть узлы сетки пронумерованы так, что первые n_c функций ψ_i^{nc} соответствуют тем же самым узлам, что и функции ψ_i^c, т.е. терминальные узлы (которым не соответствует ни одной функции ψ_i^c) будут иметь номера с n_c+1 до n (для рассматриваемого нами примера узел, который имел номер 4, в новой нумерации должен иметь номер 8, а узлы с номерами от 5 до 8 в новой нумерации должны иметь номера на единицу меньше).

Тогда, во-первых, матрица $\mathbf{T}$ имеет более простой вид (её первые n_c столбцов образуют единичную подматрицу). Во-вторых, на каждом конечном элементе очень легко определить по глобальному номеру узла, какая из его локальных базисных функций нестандартным образом участвует в построении глобальной (как $\hat{\psi}_3^{\Omega_2}$ и $\hat{\psi}_1^{\Omega_3}$ в нашем примере).

В этом случае не представляет особого труда построить алгоритм генерации глобальной матрицы $\mathbf{A}^{c}$, чтобы из стандартных локальных матриц формировались непосредственно её компоненты, а не компоненты матрицы $\mathbf{A}^{nc}$. Например, этот алгоритм может быть следующим.

- Каждая компонента $\hat{A}_{\hat{i}\hat{j}}$ локальной матрицы $\hat{\mathbf{A}}$ добавляется *стандартным образом* в компоненту A_{ij}^c глобальной матрицы $\mathbf{A}^{c}$ (по соответствию локальных номеров $\hat{i}$ и $\hat{j}$ глобальным i и j), если $i \le n_c$ и $j \le n_c$.
- Если $i > n_c$ и $j \le n_c$, то компонента $\hat{A}_{\hat{i}\hat{j}}$ добавляется с весами $T_{\mu i}$ ко всем компонентам $A_{\mu j}^c$ с такими значениями индекса μ, для которых $T_{\mu i} \neq 0$.

- Если $i \le n_c$ и $j > n_c$, то компонента $\hat{A}_{\hat{i}\hat{j}}$ добавляется с весами $T_{\nu j}$ ко всем компонентам $A^c_{i\nu}$ с такими значениями индекса ν, для которых $T_{\nu j} \neq 0$.
- Если $i > n_c$ и $j > n_c$, то компонента $\hat{A}_{\hat{i}\hat{j}}$ добавляется с весами $T_{\mu i} \cdot T_{\nu j}$ ко всем компонентам $A^c_{\mu\nu}$ с такими значениями индексов μ и ν, для которых $T_{\mu i} \neq 0$ и $T_{\nu j} \neq 0$.

В заключение скажем несколько слов о возможных процедурах сборки правой части конечноэлементной СЛАУ. В принципе, можно получить вектор правой части умножением матрицы перехода на вектор $\mathfrak{b}^{nc}$ (см. (3.13)) – в отличие от генерации матрицы, эта процедура не требует больших дополнительных вычислительных затрат. Тем не менее мы рекомендуем использовать всё-таки процедуру прямой сборки, аналогичную сборке матрицы:

- Если $i \le n_c$, каждая компонента $\hat{b}_{\hat{i}}$ локального вектора $\hat{\mathfrak{b}}$ добавляется *стандартным образом* в компоненту b^c_i глобального вектора $\mathfrak{b}^c$ (по соответствию локального номера $\hat{i}$ глобальному i).
- Если $i > n_c$, то компонента $\hat{b}_{\hat{i}}$ добавляется с весами $T_{\mu i}$ ко всем компонентам b^c_μ с такими значениями индекса μ, для которых $T_{\mu i} \neq 0$.

3.6.4. СПОСОБЫ ХРАНЕНИЯ МАТРИЦЫ ПЕРЕХОДА И АЛГОРИТМ ЕЁ ПОСТРОЕНИЯ

Очевидно, что для рассмотренного нами способа занесения локальных матриц *не нужно* (да и неудобно) хранить матрицу $\mathbf{T}$ в виде прямоугольного массива. Достаточно хранить *только ненулевые компоненты* её последних $n - n_c$ столбцов (с номерами $j > n_c$) вместе с номерами строк, в которых эти компоненты находятся. Поэтому для работы с матрицей $\mathbf{T}$ удобно использовать *разреженный столбцовый формат* (см. раздел 2.1.4) для хранения её последних $n - n_c$ столбцов. Для этого нужно три одномерных массива: массив ig указателей на начала столбцов (длины $n - n_c + 1$), массив gg для хранения *только ненулевых* компонент последних $n - n_c$ столбцов матрицы $\mathbf{T}$ и соответствующий ему массив номеров строк jg, в которых находятся эти ненулевые элементы.

Для рассмотренного нами примера матрица $\mathbf{T}$ после соответствующей перенумерации узлов примет вид

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.18}$$

и в разреженном столбцовом формате (без учёта единичной подматрицы) она может быть представлена массивами

$$ig = \{1,3\}, \quad jg = \{2,6\}, \quad gg = \{0.5,0.5\}.$$

Вообще говоря, рассмотренную процедуру согласования конечных элементов обычно применяют для несогласованных сеток, в которых каждое граничное ребро или грань любой ячейки сетки либо является частью одного граничного ребра (грани) другой ячейки, либо может быть объединением нескольких граничных рёбер (граней) других ячеек. Примеры таких двумерных сеток приведены на рис. 29 и 30.

Обратим внимание на то, что для сетки с «перехлёстом» границ, пример которой приведён на рис. 31, хотя формально рассмотренная процедура согласования и может быть применена, результат может получиться довольно плохим по точности, поскольку конечноэлементное решение в этом случае будет как бы игнорировать все узлы сетки между вершинами с номерами 2 и 10 (для билинейных базисных функций при $x = 5$ оно будет линейно по y на всём отрезке $1 \le y \le 13$).

Построим матрицу перехода для сетки, изображённой на рис. 29. В этой сетке из 21 узла только с 15 ассоциированы базисные функции ψ_i^c (т.е. $n_c = 15$). Эти 15 узлов в соответствии с приведённым выше описанием алгоритма прямой генерации матрицы $\mathbf{A}^c$ (без использования матрицы $\mathbf{A}^{nc}$) пронумерованы первыми. Поэтому для матрицы перехода $\mathbf{T}$ необходимо сформировать только пять последних столбцов. Покажем, как могут быть вычислены компоненты этих четырёх последних столбцов матрицы $\mathbf{T}$.

Обоснование метода вычисления компонент матрицы перехода $\mathbf{T}$ было приведено на с. 95. Но там фактически был описан способ формирования матрицы $\mathbf{T}$ *по строкам*, т.е. именно компоненты i-й *строки* $\mathbf{T}$ вычислялись как значения глобальной базисной функции ψ_i^c в узлах сетки (очевидно, что для всех узлов с номерами $j \le n_c$ функция ψ_i^c принимает ненулевое значение только в узле с номе-

ром $j = i$, что и делает левую подматрицу матрицы $\mathbf{T}$ единичной матрицей). Однако технологически гораздо эффективнее формировать матрицу $\mathbf{T}$ по столбцам, поскольку нам нужно фактически вычислять только ненулевые компоненты $\mathbf{T}$, а их гораздо проще находить именно по столбцам. Поэтому ниже на примере мы продемонстрируем, как работает алгоритм формирования матрицы $\mathbf{T}$ по столбцам.

Сначала сформируем столбец с номером 16 матрицы $\mathbf{T}$. Поскольку узел с номером 16 находится на ребре между узлами с номерами 2 и 9, ненулевые значения в 16-м столбце $\mathbf{T}$ должны попасть во 2-ю и 9-ю строки и соответствующие компоненты должны быть значениями глобальных базисных функций ψ_i^c в узле с номером 16. Таким образом, $T_{2,16} = \dfrac{y_9 - y_{16}}{y_9 - y_2} = \dfrac{9-6}{9-1} = 0.375$ и $T_{9,16} = \dfrac{y_{16} - y_2}{y_9 - y_2} = \dfrac{6-1}{9-1} = 0.625$ (здесь y_i – это значение y-координаты узла сетки с номером i).

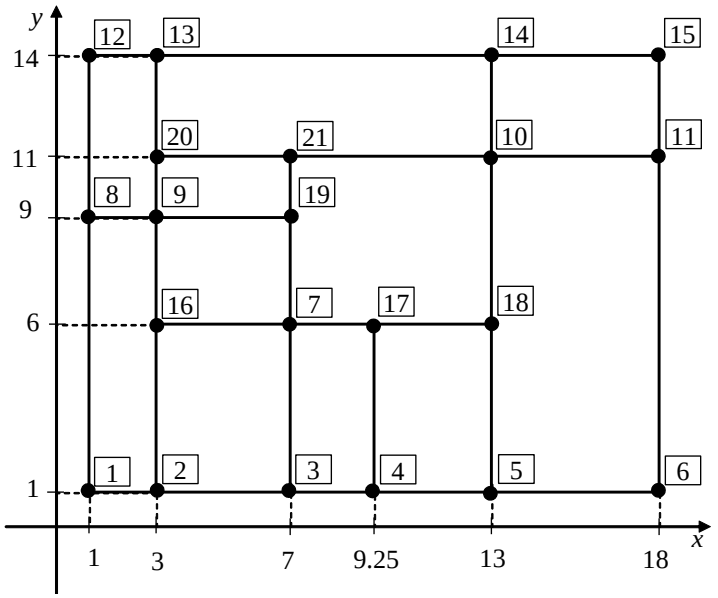


Рис. 29. Несогласованная сетка, в которой каждое граничное ребро конечного элемента контактирует не более чем с двумя граничными рёбрами других элементов

Теперь сформируем столбец с номером 17 матрицы $\mathbf{T}$. Поскольку терминальный узел с номером 17 находится на ребре между узлами 7 и 18, ненулевые значения в 17 столбце должны были бы попасть в 7-ю и 18-ю строки и соответствующие компоненты должны были бы быть значениями глобальных базисных

функций ψ_i^c в узле с номером 17. Таким образом, $T_{7,17} = \dfrac{x_{18} - x_{17}}{x_{18} - x_7} = \dfrac{13 - 9.25}{13 - 7} = 0.625$ (x_i – значение x-координаты узла сетки с номером i). Если бы узел с номером 18 не был терминальным (и ему соответствовала бы одна из функций базиса $\left\{\psi_i^c\right\}$), то соответствующая компонента матрицы $\mathbf{T}$ (обозначим её $\tilde{T}_{18,17}$) приняла бы значение $\dfrac{x_{17} - x_7}{x_{18} - x_7} = \dfrac{9.25 - 7}{13 - 7} = 0.375$. Но в матрице $\mathbf{T}$ только 15 строк, поскольку в базисе $\left\{\psi_i^c\right\}$ только 15 функций. Поэтому теперь нам нужно определить, какие базисные функции из $\left\{\psi_i^c\right\}$ принимают ненулевые значения в терминальном узле с номером 18 (и, следовательно, и в узле с номером 17). Очевидно, что для рассматриваемого нами примера это функции ψ_5^c и ψ_{10}^c. Значение в узле с номером 17 функции ψ_5^c определяется как произведение значения $T_{5,18} = \dfrac{y_{10} - y_{18}}{y_{10} - y_5} = \dfrac{11 - 6}{11 - 1} = 0.5$ на значение $\tilde{T}_{18,17}$.

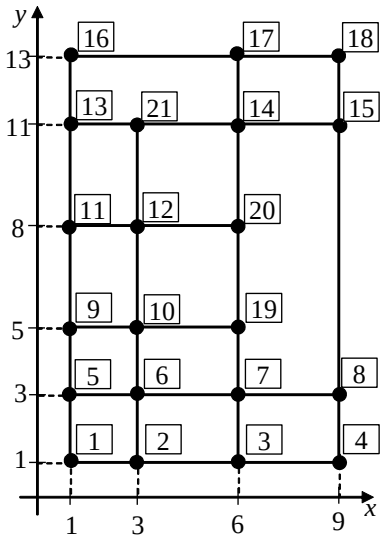


Рис. 30. Несогласованная сетка, в которой граничное ребро одного конечного элемента контактирует с граничными рёбрами сразу трёх конечных элементов

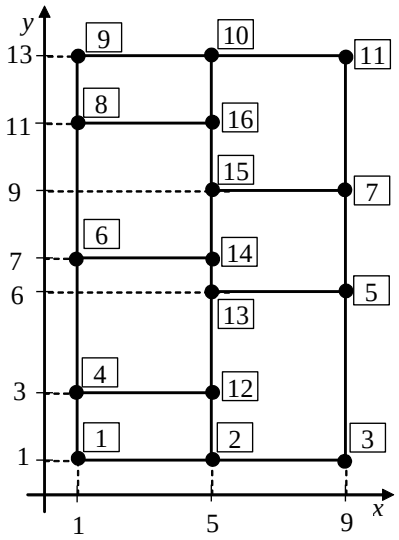


Рис. 31. Несогласованная сетка с «перехлёстами»

Действительно, функция ψ_5^c была построена так, что она линейно изменяется от значения 1 в узле с номером 5 до нулевого значения в узле с номером 10 (именно поэтому её значение в узле с номером 18 равно $\frac{y_{10}-y_{18}}{y_{10}-y_5}$). Вдоль ребра, соединяющего узлы с номерами 18 и 7, эта функция также является линейной и меняется от того значения, которое она приняла в узле с номером 18 (напомним, что это значение $T_{5,18}$) до нулевого значения в узле с номером 7. И поэтому её значение в узле с номером 17 определяется как произведение её значения в узле с номером 18 на значение $\tilde{T}_{18,17}$ линейной функции $\chi(x)$, принимающей единичное значение в узле с номером 18 и нулевое значение в узле с номером 7 (т.е. на значение $\chi(x_{18}) = \frac{x_{17}-x_7}{x_{18}-x_7} = \tilde{T}_{17,18}$). В результате мы получаем $T_{5,17} = T_{5,18} \cdot \tilde{T}_{18,17} = 0.5 \cdot 0.375 = 0.1875$.

Аналогично:

$$T_{10,17} = T_{10,18} \cdot \tilde{T}_{18,17} = \frac{y_{18}-y_5}{y_{10}-y_5} \cdot \tilde{T}_{18,17} = 0.5 \cdot 0.375 = 0.1875.$$

Таким образом, в 17-м столбце матрицы $\mathbf{T}$ три ненулевых элемента: $T_{5,17} = 0.1875$, $T_{7,17} = 0.625$ и $T_{10,17} = 0.1875$.

В столбце с номером 18 матрицы $\mathbf{T}$ ненулевыми являются только два элемента: $T_{5,18}$ и $T_{10,18}$. Они уже были получены нами при вычислении элементов 17-го столбца: $T_{5,18} = 0.5$ и $T_{10,18} = 0.5$.

Определим, в каких строках должны будут находиться ненулевые компоненты 19-го столбца. Поскольку терминальный узел с номером 19 стоит на ребре, соединяющем узлы с номерами 7 и 21, положение одной из ненулевых компонент очевидно: она должна находиться в 7-й строке. Строки же с номером 21 в матрице $\mathbf{T}$ нет (так как $21 > n_c = 15$), и поэтому мы должны определить номера вершин ребра, на котором стоит 21-й узел. Очевидно, это узлы с номерами 10 и 20. Поскольку $20 > n_c$, на данном этапе очевидно положение только ещё одной ненулевой компоненты 19-го столбца – она находится в 10-й строке (т.е. на данный момент известно, что в 19-м столбце ненулевые компоненты находятся как минимум в 7-й и 10-й строках). А для узла с номером 20 должны быть определены номера вершин ребра, на котором он расположен. Это узлы с номерами 9 и 13. Таким об-

разом, ненулевыми в 19-м столбце $\mathbf{T}$ являются компоненты строк с номерами 7, 9, 10 и 13.

Соответственно могут быть вычислены и значения компонент $T_{9,19}$, $T_{10,19}$ и $T_{13,19}$ (компонента $T_{7,19}$ вычисляется сразу же как значение в узле с номером 19 линейной на ребре между узлами с номерами 7 и 21 функции, равной единице в узле с номером 7 и нулю в узле с номером 21: $T_{7,19} = \frac{y_{21} - y_{19}}{y_{21} - y_7} = 0.4$). Компонента $T_{9,19}$ вычисляется в виде (по цепочке, которая привела от узла с номером 19 к узлу с номером 9):

$$T_{9,19} = \tilde{T}_{21,19} \cdot \tilde{T}_{20,21} \cdot T_{9,20}, \qquad (3.19)$$

т.е.

$$T_{9,19} = \frac{y_{19} - y_7}{y_{21} - y_7} \cdot \frac{x_{10} - x_{21}}{x_{10} - x_{20}} \cdot \frac{y_{13} - y_{20}}{y_{13} - y_9} = \frac{3}{5} \cdot \frac{6}{10} \cdot \frac{3}{5} = 0.216.$$

Аналогично компонента $T_{10,19}$ вычисляется как

$$T_{10,19} = \tilde{T}_{21,19} \cdot T_{10,21} = \frac{y_{19} - y_7}{y_{21} - y_7} \cdot \frac{x_{21} - x_{20}}{x_{10} - x_{20}} = \frac{3}{5} \cdot \frac{4}{10} = 0.24.$$

а компонента $T_{13,19}$ – как

$$T_{13,19} = \tilde{T}_{21,19} \cdot \tilde{T}_{20,21} \cdot T_{13,20} = \frac{y_{19} - y_7}{y_{21} - y_7} \cdot \frac{x_{10} - x_{21}}{x_{10} - x_{20}} \cdot \frac{y_{20} - y_9}{y_{13} - y_9} = \\ = \frac{3}{5} \cdot \frac{6}{10} \cdot \frac{2}{5} = 0.144 \quad .$$

Компоненты остальных столбцов (20-го и 21-го) фактически нами уже получены (при вычислении компонент 19-го столбца):

$T_{9,20} = 0.6, \qquad T_{13,20} = 0.4;$

$T_{9,21} = \tilde{T}_{20,21} \cdot T_{9,20} = 0.36, \quad T_{10,21} = 0.4, \quad T_{13,21} = \tilde{T}_{20,21} \cdot T_{13,20} = 0.24.$

Таким образом, матрица $\mathbf{T}$ в разреженном столбцовом формате (без единичной левой подматрицы) будет иметь вид:

$ig = \{1,3,6,8,12,14,17\},$

$jg = \{2,9,5,7,10,5,10,7,9,10,13,9,13,9,10,13\},$

$gg = \{0.375,0.625,0.1875,0.625,0.1875,$

$$0.5,0.5,0.4,0.216,0.24,0.144,0.6,0.4,0.36,0.4,0.24\}. \quad (3.20)$$

Обратим внимание на следующую деталь при формировании ненулевых компонент матрицы $\mathbf{T}$. Напомним, что при формировании i-й строки $\mathbf{T}$ мы должны вычислить значение функции ψ_i^c во всех узлах сетки, где она принимает ненулевое значение. То, как определить номера узлов, в которых ψ_i^c принимает ненулевые значения, рассмотрим на примере вычисления компонент 9-й строки для сетки, изображённой на рис. 29.

Так как узлы с номерами 16 и 20 стоят на рёбрах, содержащих вершину с номером 9, то ψ_9^c принимает в этих узлах ненулевые значения – это и есть значения $T_{9,16}$ и $T_{9,20}$. Узел с номером 16 *не является вершиной никакого ребра, на котором стояли бы терминальные узлы*, и поэтому идущая через него *цепочка* ненулевых элементов 9-й строки на нём заканчивается.

Узел же с номером 20 является вершиной ребра, на котором стоит терминальный узел с номером 21, и поэтому в 9-й строке $\mathbf{T}$ ненулевой будет и компонента $T_{9,21}$, причём её значение вычисляется по ведущей к ней цепочке: $T_{9,21} = T_{9,20} \cdot \tilde{T}_{20,21}$.

Узел с номером 21 также является вершиной, из которой исходит ребро, содержащее терминальный узел. Это узел с номером 19, и поэтому значение очередной ненулевой компоненты 9-й строки $T_{9,19}$ также может быть вычислено по ведущей от 9-го узла к 19-му (через 20-й и 21-й узлы) цепочке:

$$T_{9,19} = T_{9,20} \cdot \tilde{T}_{20,21} \cdot \tilde{T}_{21,19}. \quad (3.21)$$

Поскольку узел с номером 19 не является вершиной никакого ребра, содержащего терминальные узлы, на нём заканчивается и эта цепочка, поэтому в 9-й строке больше ненулевых элементов нет. Таким образом, мы показали, как матрицу $\mathbf{T}$ можно было бы сформировать по строкам.

Однако для работы этой процедуры, как мы убедились, требуется такая структура данных, которая бы позволяла по номеру узла определять исходящие из него рёбра, содержащие терминальные узлы. А поскольку чаще всего конечноэлементная сетка хранится в виде множества ячеек, определяемых узлами, гораздо проще и удобнее сформировать структуру данных, которая для каждого терминального узла позволяет определить номера вершин ребра, на котором он находится. Кроме того, заносить компоненты локальных матриц сразу (напрямую) в матрицу $\mathbf{A}^c$ на-

много удобнее, если элементы матрицы $\mathbf{T}$ хранятся по столбцам, а не по строкам. Именно поэтому сначала мы привели подробный пример, в котором матрица $\mathbf{T}$ строилась по столбцам, с движением по цепочке узлов от терминального узла к узлам, с которыми ассоциированы базисные функции, не равные нулю в рассматриваемом терминальном узле. Очевидно, что в этом случае элементы цепочки, определяющей значение искомой компоненты матрицы, получались в обратном порядке – сравните (3.19) с (3.21).

Обобщим всё сказанное выше о формировании матрицы $\mathbf{T}$. Будем считать, что в качестве исходных данных мы имеем конечноэлементную сетку в виде множества ячеек, каждая из которых задана перечислением номеров её узлов. При этом в локальной нумерации узлов ячейки перечисляются не только узлы, определяющие локальные базисные функции этого элемента, но и терминальные узлы, попавшие на границы этого элемента. Будем также считать, что все терминальные узлы имеют глобальные номера, большие n_c.

При хранении сетки в таком виде проходом по конечным элементам очень легко сформировать *структуру данных* (обозначим её $\mathfrak{S}$), в которой для каждого терминального узла с номером j хранятся те глобальные номера узлов i конечного элемента (содержащего этот терминальный узел), которым соответствуют локальные базисные функции $\hat{\psi}_{\hat{i}}$ (где $\hat{i}$ – локальный номер узла с глобальным номером i), принимающие ненулевые значения в рассматриваемом терминальном узле с глобальным номером j. Напомним, что значения $\hat{\psi}_{\hat{i}}$ в терминальном узле с номером j мы обозначали как T_{ij} (если узел с номером i не терминальный) или как $\tilde{T}_{ij}$ (если узел с номером i терминальный). Очевидно, что это как раз те значения, которые нам потребуются для формирования всех ненулевых компонент матрицы $\mathbf{T}$, и поэтому их имеет смысл сохранить вместе с номерами узлов i. Когда такая структура данных построена, алгоритм формирования $\mathbf{T}$ может быть следующим.

Выбираем очередной номер столбца j (начиная с $j = n_c + 1$). Для него определяем номера узлов i и берём соответствующие им значения T_{ij} или $\tilde{T}_{ij}$ из сформированной ранее структуры данных $\mathfrak{S}$. Если $i \le n_c$, то T_{ij} заносится в матрицу $\mathbf{T}$. Для тех же номеров i, которые больше n_c, начинается построение цепочек номеров узлов, необходимых для вычисления последующих ненулевых компонент столбца j. При этом для каждой цепочки выделяется ячейка памяти $\mathfrak{M}_i$, в кото-

рой будет накапливаться произведение вида (3.19), и в неё заносится значение $\tilde{T}_{ij}$. Далее, по структуре данных $\mathfrak{S}$ для номера узла i определяем номера узлов l. Взятые из $\mathfrak{S}$ значения T_{li} (при $l \le n_c$) или $\tilde{T}_{li}$ (при $l > n_c$) домножаем на содержимое соответствующей ячейки памяти $\mathfrak{M}_i$. Для тех l, которые не превосходят значения n_c (т.е. для $l \le n_c$), полученные значения произведений T_{li} на содержимое соответствующей ячейки памяти $\mathfrak{M}_i$ заносим в матрицу $\mathbf{T}$ как окончательно сформированные новые (не хранящиеся в $\mathfrak{S}$) компоненты T_{lj}. Для тех же номеров l, которые больше n_c, заводим «свои» ячейки памяти $\mathfrak{M}_l$ и в них переносим содержимое ячейки $\mathfrak{M}_i$, умноженное на $\tilde{T}_{li}$. Далее выбираем любую из неоконченных цепочек и продолжаем её, обращаясь к структуре $\mathfrak{S}$ и определяя новые номера, соответствующие последнему в этой цепочке номеру. При этом, очевидно, цепочка оканчивается, если все номера, полученные из структуры $\mathfrak{S}$, окажутся меньшими либо равными n_c.

После того, как все начатые цепочки будут окончены, j-й столбец матрицы $\mathbf{T}$ будет окончательно сформирован и можно переходить к формированию следующего столбца.

Очевидно, что описанная нами процедура фактически является разновидностью алгоритма обхода по дереву (который может быть реализован как обход по дереву в ширину или в глубину).

Обратим особое внимание, что описанный здесь алгоритм формирования матрицы $\mathbf{T}$ построен так, что он вполне пригоден не только для несогласованных двумерных конечноэлементных сеток с элементами первого порядка, но и для несогласованных двумерных и даже трёхмерных конечноэлементных сеток с элементами любого порядка. Заметим, что для рассмотренных нами примеров с двумерными конечными элементами первого порядка обходимое в алгоритме дерево всегда является двоичным, а при использовании элементов более высоких порядков или трёхмерных элементов из каждой вершины дерева может исходить больше двух цепочек (в соответствии со структурой $\mathfrak{S}$, в которой для каждого терминального узла хранятся значения всех базисных функций, не равных нулю в этом узле).

Заметим также, что рассмотренный алгоритм формирования матрицы $\mathbf{T}$ применим только в том случае, когда цепочки, формируемые по структуре $\mathfrak{S}$, образуют именно *дерево*, т.е. граф без циклов. Поэтому этот алгоритм не будет ра-

ботать для несогласованных сеток «с перехлёстом» (такая сетка приведёна, например, на рис. 31). На самом деле, такие сетки лучше не применять вообще (главным образом по причине резкого ухудшения аппроксимационных свойств сетки в местах «перехлёста», о чём было сказано выше, на с. 100), и поэтому, в общем-то, не имеет какого-либо смысла усложнять рассмотренный алгоритм для учёта возможных «перехлёстов» в сетке. Тем не менее контроль ситуации «перехлёста», которая приведёт к зацикливанию рассмотренного алгоритма, лучше включить в реализацию этого алгоритма, тем более что сделать это чрезвычайно просто: достаточно проверять, не появилась ли в цепочке какая-либо вершина второй раз.

В заключение в качестве ещё одного примера приведём вид матрицы перехода $\mathbf{T}$ (в разреженном столбцовом формате) для сетки, изображённой на рис. 30:

$ig = \{1,3,5,7\}$, $jg = \{7,14,7,14,13,14\}$, $gg = \{0.75,0.25,0.375,0.625,0.6,0.4\}$.

3.7. ИЕРАРХИЧЕСКИЕ БАЗИСЫ. СОГЛАСОВАНИЕ КОНЕЧНЫХ ЭЛЕМЕНТОВ РАЗНЫХ ПОРЯДКОВ

В принципе, если конечноэлементная сетка содержит лагранжевы элементы разных порядков, то их можно согласовывать с помощью специальных технологий, таких, как технология $\mathbf{T}$-преобразования. Однако элементы разных порядков можно согласовывать и гораздо проще, если использовать на них в качестве базисных функции специального вида – так называемые *иерархические базисные функции*. Определяется иерархический базис следующим образом.

На конечном элементе Ω_e **иерархическим** называют такой базис, который содержит все базисные функции этого элемента более низкого порядка.

3.7.1. ИЕРАРХИЧЕСКИЕ БАЗИСЫ ОДНОМЕРНЫХ ЭЛЕМЕНТОВ

При построении иерархических базисов в качестве шаблонного мы будем использовать одномерный элемент $\Omega^E = [-1,1]$. На этом элементе линейные базисные функции имеют вид

$$\hat{\varphi}_1(\xi) = \frac{1-\xi}{2}, \qquad \hat{\varphi}_2(\xi) = \frac{1+\xi}{2}, \tag{3.22}$$

т.е. иерархические базисные функции конечного элемента первого порядка просто совпадают с лагранжевыми базисными функциями.

Иерархический базис конечного элемента второго порядка определим добавкой к функциям (3.22) квадратичной функции, равной нулю на концах Ω^E:

$$\hat{\varphi}_3\left(\xi\right) = 1 - \xi^2. \tag{3.23}$$

Для построения иерархического базиса каждого следующего порядка p нужно добавить к базису элемента предыдущего порядка любой полином степени p, равный нулю на концах Ω^E, например полином

$$\hat{\varphi}_{p+1}\left(\xi\right) = \xi^{p-2}\left(1 - \xi^2\right). \tag{3.24}$$

Очевидно, что эта формула определяет и функцию (3.23) элемента второго порядка (при $p = 2$), и поэтому построенный иерархический базис определяется соотношениями (3.22) и соотношением (3.24) при $p \geq 2$.

Рассмотренный иерархический базис не является единственно возможным, поскольку существует множество полиномов степени $p \geq 3$, равных нулю на концах Ω^E. Полиномы вида (3.24) имеют очень простую форму, но коэффициенты при них в разложении конечноэлементного решения по такому иерархическому базису не имеют какой-либо очевидной интерпретации (как, например, значение этого решения в узлах или значение его каких-либо производных). Как мы покажем позже, при согласовании иерархических элементов на несогласованных сетках это не очень удобно, поскольку может приводить к довольно ощутимым дополнительным вычислительным затратам. Поэтому в таких ситуациях лучше использовать такие базисы, которые позволяют находить коэффициенты разложения по базисным функциям относительно простыми способами. Например, это может быть базис, в котором функции для $p \geq 2$ определяются формулой:

$$\hat{\varphi}_{p+1}\left(\xi\right) = \begin{cases} \dfrac{1}{p!}\left(\xi^p - 1\right), & \text{если } p \text{ чётно,} \\ \dfrac{1}{p!}\left(\xi^p - \xi\right), & \text{если } p \text{ нечётно.} \end{cases} \tag{3.25}$$

Нетрудно убедиться, что у функций, определяемых соотношением (3.25), в точке $\xi = 0$ все производные не ниже второго порядка, кроме производной порядка p, равны нулю, и только производная порядка p равна единице. Это позволяет легко вычислять коэффициенты разложения любого полинома по такому базису. Позже мы покажем, как удобно это свойство базисных функций при согласовании

иерархических элементов на несогласованных сетках в двумерном и трёхмерном случаях.

Можно также построить такой иерархический базис на одномерных элементах, что почти у всех базисных функций первые производные будут ортогональны друг другу (т.е. интегралы по Ω^E от произведений первых производных разных базисных функций будут равны нулю). Базис такого типа можно построить с помощью ортогональных полиномов Лежандра.

Полином Лежандра степени p может быть определён на Ω^E с помощью соотношения

$$P_p\left(\xi\right)=\frac{1}{p!}\frac{1}{2^p}\frac{d^p}{d\xi^p}\left(\left(\xi^2-1\right)^p\right). \tag{3.26}$$

Кроме того, для полиномов Лежандра справедливо рекуррентное соотношение

$$P_p\left(\xi\right)=\frac{2p-1}{p}\xi P_{p-1}\left(\xi\right)-\frac{p-1}{p}P_{p-2}\left(\xi\right), \tag{3.27}$$

которое позволяет вычислить любой полином Лежандра, если известны первые два из них:

$$P_0\left(\xi\right)=1,\qquad P_1\left(\xi\right)=\xi\,. \tag{3.28}$$

Приведём ещё несколько полиномов Лежандра:

$$\begin{aligned} &P_2\left(\xi\right)=\tfrac{1}{2}\left(3\xi^2-1\right), && P_3\left(\xi\right)=\tfrac{1}{2}\left(5\xi^3-3\xi\right),\\ &P_4\left(\xi\right)=\tfrac{1}{2}\left(35\xi^4-30\xi^2+3\right), && P_5\left(\xi\right)=\tfrac{1}{8}\left(63\xi^5-70\xi^3+15\xi\right). \end{aligned} \tag{3.29}$$

Иерархические базисные функции с ортогональными производными (начиная с квадратичной) могут быть получены из полиномов Лежандра по формуле

$$\hat{\varphi}_{p+1}\left(\xi\right)=c_{p+1}\int_{-1}^{\xi}P_{p-1}\left(\eta\right)d\eta\,,\qquad p\geq 2\,, \tag{3.30}$$

где c_{p+1} – произвольная ненулевая константа. Можно также определить эти функции по формуле

$$\hat{\varphi}_{p+1}\left(\xi\right)=c'_{p+1}\left(P_p\left(\xi\right)-P_{p-2}\left(\xi\right)\right),\qquad p\geq 2\,. \tag{3.31}$$

Коэффициент c_{p+1} в соотношении (3.30) (так же как и коэффициент c'_{p+1} в (3.31)) можно выбрать так, чтобы базисная функция имела удобный вид. Приве-

дём вид нескольких (первых) базисных функций с ортогональными производными:

$$\hat{\varphi}_3\left(\xi\right)=\xi^2-1, \qquad \hat{\varphi}_4\left(\xi\right)=\xi^3-\xi,$$
$$\hat{\varphi}_5\left(\xi\right)=5\xi^4-6\xi^2+1, \quad \hat{\varphi}_6\left(\xi\right)=7\xi^5-10\xi^3+3\xi. \tag{3.32}$$

Заметим, что производные этих функций ортогональны также и производным линейных функций (3.22). Неортогональными в системе функций (3.22), (3.30) являются производные только двух линейных функций (3.22). Кроме того, обратим внимание читателя на то, что функции $\hat{\varphi}_3$ и $\hat{\varphi}_4$ из (3.32) с точностью до коэффициента совпадают с соответствующими функциями, определяемыми соотношением (3.25), т.е. в системе функций (3.25) квадратичная и кубическая функции также имеют производные, *ортогональные* друг другу и производным линейных функций (3.22). Это свойство ортогональности производных может быть учтено при построении локальных матриц жёсткости. Если с помощью таких функций строятся базисные функции двумерных или трёхмерных элементов, то и в их локальных матрицах жёсткости будет *достаточно много нулевых компонент*, которые *не нужно хранить* в глобальной матрице *жёсткости*.

3.7.2. ИЕРАРХИЧЕСКИЕ БАЗИСЫ НА ПРЯМОУГОЛЬНИКАХ

На прямоугольных элементах иерархические базисы высоких порядков можно строить из одномерных иерархических базисов точно так же, как это было сделано при построении лагранжевых (и эрмитовых) элементов. Так, биквадратичный иерархический базис на шаблонном элементе $\Omega^E=\left[-\frac{h_x}{2},\frac{h_x}{2}\right]\times\left[-\frac{h_y}{2},\frac{h_y}{2}\right]$ содержит следующие функции:

$$\hat{\psi}_1\left(x,y\right)=\hat{\varphi}_1\left(\frac{x}{h_x}\right)\hat{\varphi}_1\left(\frac{y}{h_y}\right),\ \hat{\psi}_2\left(x,y\right)=\hat{\varphi}_2\left(\frac{x}{h_x}\right)\hat{\varphi}_1\left(\frac{y}{h_y}\right), \tag{3.33}$$

$$\hat{\psi}_3\left(x,y\right)=\hat{\varphi}_1\left(\frac{x}{h_x}\right)\hat{\varphi}_2\left(\frac{y}{h_y}\right),\ \hat{\psi}_4\left(x,y\right)=\hat{\varphi}_2\left(\frac{x}{h_x}\right)\hat{\varphi}_2\left(\frac{y}{h_y}\right), \tag{3.34}$$

$$\hat{\psi}_5\left(x,y\right)=\hat{\varphi}_1\left(\frac{x}{h_x}\right)\hat{\varphi}_3\left(\frac{y}{h_y}\right),\ \hat{\psi}_6\left(x,y\right)=\hat{\varphi}_2\left(\frac{x}{h_x}\right)\hat{\varphi}_3\left(\frac{y}{h_y}\right), \tag{3.35}$$

$$\hat{\psi}_7\left(x,y\right)=\hat{\varphi}_3\left(\frac{x}{h_x}\right)\hat{\varphi}_1\left(\frac{y}{h_y}\right),\ \hat{\psi}_8\left(x,y\right)=\hat{\varphi}_3\left(\frac{x}{h_x}\right)\hat{\varphi}_2\left(\frac{y}{h_y}\right), \tag{3.36}$$

$$\hat{\psi}_9\left(x,y\right)=\hat{\varphi}_3\left(\frac{x}{h_x}\right)\hat{\varphi}_3\left(\frac{y}{h_y}\right). \tag{3.37}$$

Очевидно, что первые четыре функции – это стандартные билинейные функции на прямоугольнике, и поэтому этот элемент действительно является иерархическим.

Функции $\hat{\psi}_5$, $\hat{\psi}_6$, $\hat{\psi}_7$ и $\hat{\psi}_8$ являются линейными по одной координате и квадратичными по другой. Эти четыре функции на границах элемента отличны от нуля каждая только на одном «своём» ребре: $\hat{\psi}_5$ на левом, $\hat{\psi}_6$ на правом, $\hat{\psi}_7$ на нижнем и $\hat{\psi}_8$ на верхнем. Поэтому функции $\hat{\psi}_5$, $\hat{\psi}_6$, $\hat{\psi}_7$ и $\hat{\psi}_8$ удобно ассоциировать *не с узлами элемента, а с соответствующими рёбрами* прямоугольника (на которых они отличны от нуля).

Функция же $\hat{\psi}_9$ является биквадратичной и отлична от нуля внутри элемента и равна нулю на всех его границах.

Так же как иерархический базис прямоугольного элемента второго порядка был получен добавкой к базису (3.33)–(3.34) элемента первого порядка пяти функций (3.35)–(3.37), иерархический базис прямоугольного элемента третьего порядка должен быть построен добавлением базисных функций к базису (3.33)–(3.37). Очевидно, что, как и в случае лагранжевых элементов, полный бикубический базис должен содержать 16 функций.

Добавляемые функции строятся в виде произведения одномерных иерархических функций предыдущих порядков от одной координаты на кубические от другой и произведения двух одномерных кубических функций от x и y:

$$\hat{\psi}_{10}\left(x,y\right)=\hat{\varphi}_1\left(\frac{x}{h_x}\right)\hat{\varphi}_4\left(\frac{y}{h_y}\right),\ \hat{\psi}_{11}\left(x,y\right)=\hat{\varphi}_2\left(\frac{x}{h_x}\right)\hat{\varphi}_4\left(\frac{y}{h_y}\right), \tag{3.38}$$

$$\hat{\psi}_{12}\left(x,y\right)=\hat{\varphi}_4\left(\frac{x}{h_x}\right)\hat{\varphi}_1\left(\frac{y}{h_y}\right),\ \hat{\psi}_{13}\left(x,y\right)=\hat{\varphi}_4\left(\frac{x}{h_x}\right)\hat{\varphi}_2\left(\frac{y}{h_y}\right), \tag{3.39}$$

$$\hat{\psi}_{14}\left(x,y\right)=\hat{\varphi}_3\left(\frac{x}{h_x}\right)\hat{\varphi}_4\left(\frac{y}{h_y}\right),\ \hat{\psi}_{15}\left(x,y\right)=\hat{\varphi}_4\left(\frac{x}{h_x}\right)\hat{\varphi}_3\left(\frac{y}{h_y}\right), \tag{3.40}$$

$$\hat{\psi}_{16}\left(x,y\right)=\hat{\varphi}_4\left(\frac{x}{h_x}\right)\hat{\varphi}_4\left(\frac{y}{h_y}\right). \tag{3.41}$$

Функции $\hat{\psi}_{10}$, $\hat{\psi}_{11}$, $\hat{\psi}_{12}$ и $\hat{\psi}_{13}$ (так же, как и $\hat{\psi}_5$, $\hat{\psi}_6$, $\hat{\psi}_7$ и $\hat{\psi}_8$) отличны от нуля каждая только на одном «своём» ребре, и поэтому они также *ассоциируются* с этими *рёбрами*. Остальные функции, очевидно, равны нулю на всех границах элемента (они *ассоциируются*, как и $\hat{\psi}_9$, *с внутренней частью элемента*).

Аналогично строятся иерархические базисы на прямоугольных элементах более высоких порядков: добавляются функции, являющиеся произведениями одномерных иерархических функций предыдущих порядков от одной координаты (x или y) на одномерный иерархический полином высшей степени от другой координаты (y или x), и ещё одна функция, являющаяся произведением двух одномерных полиномов высшей степени от x и y соответственно.

Локальные матрицы иерархических элементов так же, как и для лагранжевых и эрмитовых элементов, удобно строить из локальных матриц одномерных элементов.

Заметим, что определённые таким способом иерархические базисы порождают те же самые конечномерные пространства функций, что и базисы из лагранжевых функций. Но иногда используют и иерархические базисы с меньшим количеством функций. Например, на элементе второго порядка можно не вводить функцию $\hat{\psi}_9$, поскольку при удалении этой функции хотя и перестаёт быть представимым полный *биквадратичный полином*, но при этом не нарушается представимость любых полиномов до *второй степени*. Точно так же на элементе третьего порядка можно не вводить функции $\hat{\psi}_9$, $\hat{\psi}_{14}$, $\hat{\psi}_{15}$ и $\hat{\psi}_{16}$. Но при этом функция $\hat{\psi}_9$ должна присутствовать в базисе элемента четвёртого порядка, поскольку без неё уже не будет представим произвольный полином четвёртой степени.

Очевидно, что иерархические базисы на прямоугольнике с уменьшенным числом функций имеют как достоинства, так и недостатки по сравнению с полными биквадратичными, бикубическими и т.п. базисами. К очевидным достоинствам относится уменьшение размерности конечноэлементной СЛАУ, а к недостаткам – возможное снижение точности получаемого конечноэлементного решения.

3.7.3. ИЕРАРХИЧЕСКИЕ БАЗИСЫ НА ПРЯМОУГОЛЬНЫХ ПАРАЛЛЕЛЕПИПЕДАХ

Покажем, как строятся иерархические базисы на прямоугольных параллелепипедах. Для этого введём шаблонный элемент $\Omega^E = \left[-\frac{h_x}{2}, \frac{h_x}{2}\right] \times \left[-\frac{h_y}{2}, \frac{h_y}{2}\right] \times \left[-\frac{h_z}{2}, \frac{h_z}{2}\right]$. Иерархические базисные функции на этом элементе так же, как и лагранжевы базисные функции, могут быть построены через одномерные шаблонные иерархические функции $\hat{\varphi}_k\left(\xi\right)$ (определённые соотношением (3.22) и любым из соотношений (3.24), или (3.25), или (3.30) или (3.31)) с помощью формулы

$$\hat{\psi}_i\left(x,y,z\right) = \hat{\varphi}_{\mu(i)}\left(\frac{x}{h_x}\right) \cdot \hat{\varphi}_{\nu(i)}\left(\frac{y}{h_y}\right) \cdot \hat{\varphi}_{\vartheta(i)}\left(\frac{z}{h_z}\right), \qquad (3.42)$$

где $\mu\left(i\right)$, $\nu\left(i\right)$ и $\vartheta\left(i\right)$ – целочисленные функции, определяемые для элементов до третьего порядка включительно следующей табл. 1:

Таблица 1

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
μ	1	2	1	2	1	2	1	2	1	2	3	3	1	2	3	3
ν	1	1	2	2	1	1	2	2	3	3	1	2	3	3	1	2
ϑ	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2

i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
μ	1	2	1	2	1	2	3	3	3	3	3	1	2	4	4	1
ν	1	1	2	2	3	3	1	2	3	3	3	4	4	1	2	4
ϑ	3	3	3	3	3	3	3	3	1	2	3	1	1	1	1	2

i	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
μ	2	4	4	1	2	1	2	1	1	1	2	2	2	3	4	4
ν	4	1	2	1	1	2	2	3	4	4	3	4	4	1	1	1
ϑ	2	2	2	4	4	4	4	4	3	4	4	3	4	4	3	4

i	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
μ	3	4	4	3	4	4	3	4	4	4	3	3	4	4	3	4
ν	2	2	2	4	3	4	4	3	4	3	4	3	4	3	4	4
ϑ	4	3	4	1	1	1	2	2	2	3	3	4	3	4	4	4

Таким образом, первые 8 из функций (3.42) являются базисными для элемента первого порядка. Эти 8 и следующие 19 (т.е. первые 27) функций являются базисными для элемента второго порядка (триквадратичного). Базис элемента третьего порядка (трикубического) получается добавлением к 27 функциям триквадратичного элемента следующих за ними 37 функций (т.е. этот базис содержит 64 функции). Для элемента четвёртого порядка добавляется 61 функция (в нём $5^3 = 125$ функций) и т.д.

Обратим внимание, что в трёхмерном случае иерархические базисные функции удобно ассоциировать не только с узлами и рёбрами, но и с гранями элементов. Принцип ассоциирования остаётся тем же: если базисная функция равна нулю на всех гранях, кроме одной, то она с этой гранью и ассоциируется. Функции, равные нулю на всех границах элемента, можно ассоциировать с центром элемента. Информация о том, с каким узлом, ребром или гранью ассоциирована базисная функция, необходима для глобальной нумерации.

Табл. 1 фактически определяет следующее соответствие базисных функций узлам, рёбрам и граням параллелепипеда, с которыми эти базисные функции ассоциируются (для иерархических элементов до третьего порядка). Первые 8 функций ассоциируются с вершинами (и их номера совпадают с локальными номерами вершин). Информация о том, какие функции соответствуют каким рёбрам, приведена в табл. 2. В табл. 3 приведена информация о функциях, ассоциированных с гранями. Оставшиеся функции ассоциированы с центром элемента – это функции с номерами 27, 58÷64 (индексы μ, ν и ϑ в (3.42) для этих функций принимают значения 3 или 4).

Таблица 2

рёбра		$\begin{cases} x=-h_x/2 \\ y=-h_y/2 \end{cases}$	$\begin{cases} x=h_x/2 \\ y=-h_y/2 \end{cases}$	$\begin{cases} x=-h_x/2 \\ y=h_y/2 \end{cases}$	$\begin{cases} x=h_x/2 \\ y=h_y/2 \end{cases}$
значения индексов	μ	1	2	1	2
	ν	1	1	2	2
	ϑ	3, 4	3, 4	3, 4	3, 4
номера базисных функций		17, 36	18, 37	19, 38	20, 39

рёбра		$\begin{cases} y=-h_y/2 \\ z=-h_z/2 \end{cases}$	$\begin{cases} y=h_y/2 \\ z=-h_z/2 \end{cases}$	$\begin{cases} y=-h_y/2 \\ z=h_z/2 \end{cases}$	$\begin{cases} y=h_y/2 \\ z=h_z/2 \end{cases}$
значения индексов	μ	3, 4	3, 4	3, 4	3, 4
	ν	1	2	1	2
	ϑ	1	1	2	2
номера базисных функций		11, 30	12, 31	15, 34	16, 35

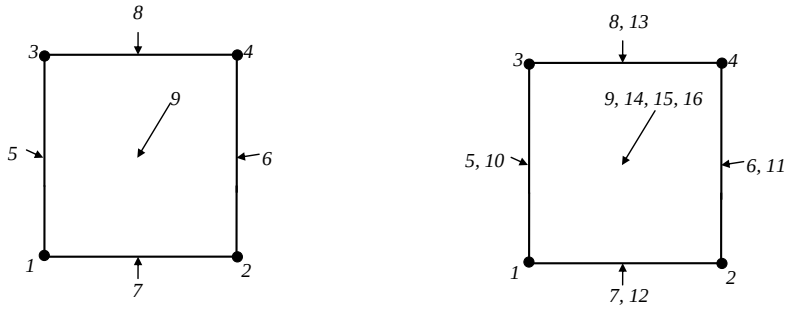
рёбра		$\begin{cases} x=-h_x/2 \\ z=-h_z/2 \end{cases}$	$\begin{cases} x=h_x/2 \\ z=-h_z/2 \end{cases}$	$\begin{cases} x=-h_x/2 \\ z=h_z/2 \end{cases}$	$\begin{cases} x=h_x/2 \\ z=h_z/2 \end{cases}$
значения индексов	μ	1	2	1	2
	ν	3, 4	3, 4	3, 4	3, 4
	ϑ	1	1	2	2
номера базисных функций		9, 28	10, 29	13, 32	14, 33

Таблица 3

грани		$x=-h_x/2$	$x=h_x/2$	$y=-h_y/2$	$y=h_y/2$	$z=-h_z/2$	$z=h_z/2$
значения индексов	μ	1	2	3, 4	3, 4	3, 4	3, 4
	ν	3, 4	3, 4	1	2	3, 4	3, 4
	ϑ	3, 4	3, 4	3, 4	3, 4	1	2
номера базисных функций		21, 40, 41, 42	22, 43, 44, 45	23, 46, 47, 48	24, 49, 50, 51	25, 52, 53, 54	26, 55, 56, 57

3.7.4. СОГЛАСОВАНИЕ ИЕРАРХИЧЕСКИХ ЭЛЕМЕНТОВ РАЗНЫХ ПОРЯДКОВ НА ПРЯМОУГОЛЬНИКАХ И ПАРАЛЛЕЛЕПИПЕДАХ

Иерархические элементы удобны тем, что позволяют очень легко согласовывать элементы разных порядков. В качестве примера покажем, как согласуются элементы второго и третьего порядка на прямоугольнике.



а *б*

Рис. 32. Локальная нумерация базисных функций на элементах второго (*а*) и третьего (*б*) порядка

На рис. 32 слева изображён элемент второго порядка, а справа – элемент третьего порядка. Для согласования этих элементов достаточно на элементе третьего порядка удалить из базиса одну функцию, ассоциированную с левым ребром и являющуюся на этом ребре кубической. В соответствии с принятой нами локальной нумерацией (см. раздел 3.7.2) это функция с номером 10. Таким образом, ненулевые на этом ребре глобальные базисные функции получаются сшивкой локальных базисных функций $\hat{\psi}_2$, $\hat{\psi}_4$ и $\hat{\psi}_6$ левого элемента с локальными базисными функциями $\hat{\psi}_1$, $\hat{\psi}_3$ и $\hat{\psi}_5$ правого элемента соответственно. Полученные в результате такой сшивки глобальные базисные функции являются непрерывными, и поэтому конечные элементы оказываются согласованными.

На рис. 33 перечислены все глобальные базисные функции (с привязкой к вершинам, рёбрам и центрам элементов, с которыми они ассоциированы) после согласования квадратичного и кубического элемента. Таким образом, с левым ребром кубического элемента (по которому этот элемент стыкуется с квадратичным) ассоциирована только одна (квадратичная) глобальная базисная функция (в отличие от остальных его рёбер, с которыми ассоциировано по две функции). В результате вся процедура согласования выродилась в исключение из базиса правого элемента локальной базисной функции $\hat{\psi}_{10}$ (и поэтому ей и не нужно было давать никакого глобального номера).

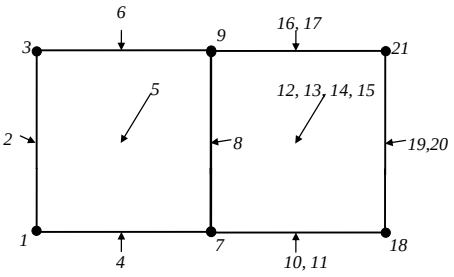


Рис. 33. Глобальная нумерация после согласования биквадратичного и бикубического элементов

Согласование иерархических элементов разных порядков на параллелепипедах идейно выполняется так же. Однако в этом случае согласование нужно проводить по объектам двух типов: рёбрам и граням.

Рассмотрим изображённую на рис. 34 ситуацию, когда контактируют элемент Ω_1 второго порядка (с вершинами {7, 8, 10, 11, 13, 14, 15, 16}) и два элемента третьего порядка (это элемент Ω_2 с вершинами {1, 2, 4, 5, 7, 8, 10, 11} и элемент Ω_3 с вершинами {2, 3, 5, 6, 8, 9, 11, 12}). Поскольку грань {7, 8, 10, 11} является общей для элементов разных порядков, с ней должны быть ассоциированы только глобальные базисные функции, соответствующие элементу более низкого порядка. В нашем случае это означает, что с ней должна быть ассоциирована только одна базисная функция (из иерархического базиса элемента второго порядка, являющаяся биквадратичной на этой грани), а из базиса элемента Ω_2 должна быть исключена ассоциированная с этой гранью базисная функция элемента третьего порядка (являющаяся на ней бикубической). При этом ассоциированная с гранью {2,5,8,11} бикубическая (на ней) функция остаётся в глобальном базисе (и соответственно как локальная базисная функция элементов Ω_2 и Ω_3).

На всех четырёх рёбрах, которые являются общими для элементов третьего и второго порядка (это рёбра {7,8}, {7,10}, {8,11} и {10,11}), не должно быть ассоциированных с ними кубических базисных функций. Именно поэтому из базиса элемента Ω_2 должны быть удалены 4 кубические (на каждом из рёбер {7,8}, {7,10}, {8,11} и {10,11}) локальные функции. Поскольку ребро {8,11} принадлежит ещё и элементу Ω_3, из его локального базиса тоже должна быть удалена кубическая функция, ассоциированная с этим ребром.

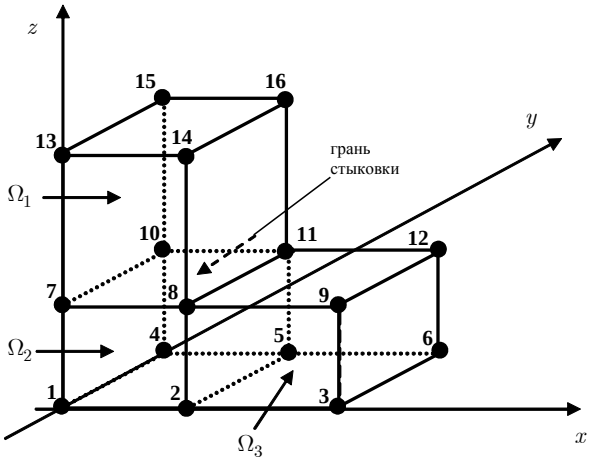


Рис. 34. К согласованию иерархических элементов
разных порядков на параллелепипедах

Итак, процедура согласования иерархических базисов разных порядков и в трёхмерном случае фактически свелась к исключению из локального иерархического базиса элемента некоторых функций, ассоциированных с рёбрами и гранями (для тех рёбер и граней, по которым рассматриваемый элемент граничит с элементами более низкого порядка). Поэтому фактически при занесении локальной матрицы элемента в глобальную соответствующие строки и столбцы в ней должны быть просто проигнорированы (и поэтому они могут даже и не генерироваться).

Таким образом, основной проблемой при согласовании иерархических базисов является создание процедуры, позволяющей для каждого элемента определять глобальные номера всех ненулевых на нём базисных функций (т.е. установление соответствия локальных номеров его базисных функций глобальным). Это проблема может решаться разными путями.

Можно, например, для каждого элемента наряду со списком всех его вершин и номеров глобальных базисных функций хранить номера всех элементов, смежных с ним по граням и рёбрам. В этом случае процедура глобальной нумерации может быть реализована следующим достаточно простым способом. Первые N номеров получают базисные функции, ассоциированные с вершинами элементов (N – общее число вершин элементов). Затем нумеруем все базисные функции первого элемента и номера функций, ассоциированных с его гранями и рёбрами, заносим в соответствующие места списка номеров глобальных базисных функций смежных элементов. То же самое делаем для всех остальных элементов, но при

этом новые глобальные номера получают только те локальные базисные функции, которые не были пронумерованы ранее.

Другой путь решения этой проблемы базируется на том, что наряду с глобальной нумерацией вершин вводится глобальная нумерация всех ребер, граней и самих элементов. Для каждого элемента в этом случае хранятся не глобальные номера его базисных функций, а глобальные номера его вершин, рёбер и граней. Глобальные же номера базисных функций приписываются к рёбрам и граням, что автоматически обеспечивает их совпадение для смежных элементов. Способы нумерации рёбер и граней были рассмотрены нами в разделах 2.3.2 и 2.3.3.

4. ВЕКТОРНЫЙ МКЭ ДЛЯ РЕШЕНИЯ ЗАДАЧ ЭЛЕКТРОМАГНЕТИЗМА

4.1. ВЕКТОРНЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ ВТОРОГО ПОРЯДКА С РАЗРЫВНЫМИ РЕШЕНИЯМИ

Для описания электромагнитного поля можно использовать модель в виде одного векторного уравнения

$$\mathrm{rot}\left(\frac{1}{\mu}\,\mathrm{rot}\,\vec{\mathbf{A}}\right)+\sigma\frac{\partial\vec{\mathbf{A}}}{\partial t}+\varepsilon\frac{\partial^2\vec{\mathbf{A}}}{\partial t^2}=\vec{\mathbf{J}}^{\mathrm{ст}}. \qquad (4.1)$$

Модель электромагнитного поля на основе векторного уравнения (4.1) позволяет решать самые сложные задачи электромагнетизма. Она корректно описывает электромагнитные поля в ситуациях, когда среда содержит любые неоднородности с изменёнными электрическими и магнитными свойствами. На основе этой модели можно рассчитывать распространение электромагнитных волн с учётом конечной скорости их распространения и отражения от объектов с изменёнными свойствами как в непроводящих, так и в проводящих и в смешанных (т.е. содержащих непроводящие и проводящие подобласти) средах. Уравнение (4.1) позволяет решать и нелинейные электромагнитные задачи, в которых коэффициенты (чаще всего магнитная проницаемость μ) зависят от значений вычисляемого поля. Таким образом, при изучении процессов электромагнетизма уравнение (4.1) играет одну из самых важных ролей.

Однако векторное уравнение (4.1) имеет определённые особенности (к ним относится, например, возможная разрывность $\vec{\mathbf{A}}$), приводящие к серьёзным трудностям при попытках воспользоваться рассмотренными ранее схемами построения конечноэлементных аппроксимаций. Тем не менее в последние годы были предложены новые подходы, которые позволяют успешно применять МКЭ и для решения этого уравнения. Изучению таких подходов и будет посвящён данный раздел.

Прежде чем приступить к построению конечноэлементной аппроксимации начально-краевой задачи для уравнения (4.1) по пространству, мы, как обычно, выполним аппроксимацию уравнения (4.1) по времени.

Обозначим через $\vec{\mathbf{A}} = \vec{\mathbf{A}}\left(x, y, z, t_j\right)$ значение вектор-потенциала на текущем временном слое, а через $\vec{\mathbf{A}}^{\Leftarrow 1} = \vec{\mathbf{A}}\left(x, y, z, t_{j-1}\right)$, $\vec{\mathbf{A}}^{\Leftarrow 2} = \vec{\mathbf{A}}\left(x, y, z, t_{j-2}\right)$ и $\vec{\mathbf{A}}^{\Leftarrow 3} = \vec{\mathbf{A}}\left(x, y, z, t_{j-3}\right)$ – значение вектор-потенциала на трёх предыдущих временных слоях. Тогда в результате аппроксимации по времени уравнения (4.1) получим векторное уравнение

$$\operatorname{rot}\left(\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}}\right) + \gamma\vec{\mathbf{A}} = \vec{\mathbf{F}}, \tag{4.2}$$

где коэффициент γ и вектор-функция $\vec{\mathbf{F}}$ определяются схемой аппроксимации по времени. Например, для трёхслойной полностью неявной схемы коэффициент γ и вектор-функция $\vec{\mathbf{F}}$ имеют вид

$$\gamma = \frac{\sigma\left(2t_j - t_{j-1} - t_{j-2}\right) + 2\varepsilon}{\left(t_j - t_{j-1}\right)\left(t_j - t_{j-2}\right)}, \tag{4.3}$$

$$\vec{\mathbf{F}} = \vec{\mathbf{J}}^{\text{ст}} + \frac{\sigma\left(t_j - t_{j-2}\right) + 2\varepsilon}{\left(t_j - t_{j-1}\right)\left(t_{j-1} - t_{j-2}\right)}\vec{\mathbf{A}}^{\Leftarrow 1} - \frac{\sigma\left(t_j - t_{j-1}\right) + 2\varepsilon}{\left(t_j - t_{j-2}\right)\left(t_{j-1} - t_{j-2}\right)}\vec{\mathbf{A}}^{\Leftarrow 2}, \tag{4.4}$$

а для четырёхслойной полностью неявной схемы – вид

$$\gamma = \frac{\sigma\left(\left(t_j - t_{j-1}\right)\left(t_j - t_{j-2}\right) + \left(t_j - t_{j-1}\right)\left(t_j - t_{j-3}\right)\right)}{\left(t_j - t_{j-1}\right)\left(t_j - t_{j-2}\right)\left(t_j - t_{j-3}\right)} + \\ + \frac{\sigma\left(\left(t_j - t_{j-2}\right)\left(t_j - t_{j-3}\right)\right) + 2\varepsilon\left(3t_j - t_{j-1} - t_{j-2} - t_{j-3}\right)}{\left(t_j - t_{j-1}\right)\left(t_j - t_{j-2}\right)\left(t_j - t_{j-3}\right)}, \tag{4.5}$$

$$\vec{\mathbf{F}} = \vec{\mathbf{J}}^{\text{ст}} + \frac{\sigma\left(t_j - t_{j-2}\right)\left(t_j - t_{j-3}\right) + 2\varepsilon\left(2t_j - t_{j-2} - t_{j-3}\right)}{\left(t_j - t_{j-1}\right)\left(t_{j-2} - t_{j-1}\right)\left(t_{j-3} - t_{j-1}\right)}\vec{\mathbf{A}}^{\Leftarrow 1} + \\ + \frac{\sigma\left(t_j - t_{j-1}\right)\left(t_j - t_{j-3}\right) + 2\varepsilon\left(2t_j - t_{j-1} - t_{j-3}\right)}{\left(t_j - t_{j-2}\right)\left(t_{j-1} - t_{j-2}\right)\left(t_{j-3} - t_{j-2}\right)}\vec{\mathbf{A}}^{\Leftarrow 2} \\ + \frac{\sigma\left(t_j - t_{j-1}\right)\left(t_j - t_{j-2}\right) + 2\varepsilon\left(2t_j - t_{j-1} - t_{j-2}\right)}{\left(t_j - t_{j-3}\right)\left(t_{j-1} - t_{j-3}\right)\left(t_{j-2} - t_{j-3}\right)}\vec{\mathbf{A}}^{\Leftarrow 3}. \tag{4.6}$$

Заметим, что при решении задач с несущественными токами смещения $\varepsilon\dfrac{\partial^2\vec{\mathbf{A}}}{\partial t^2}$ (в этих задачах обычно полагают $\varepsilon = 0$) лучше использовать трёхслойную полностью неявную схему. При решении же задач с существенным влиянием токов смещения целесообразнее использовать четырёхслойную полностью неявную схему.

Особенность векторного уравнения (4.2) состоит в том, что, хотя оно и является дифференциальным уравнением второго порядка, его решение $\vec{\mathbf{A}}$ *имеет разрывные нормальные составляющие* на границах разрыва коэффициента γ (а значит, и коэффициентов σ и ε, т.е. на границах сред с различной проводимостью или диэлектрической проницаемостью). Поэтому при решении векторного уравнения (4.2) имеет смысл использовать такие базисные функции, которые позволяют достаточно хорошо аппроксимировать разрывные вектор-функции. Ниже мы покажем, как могут быть построены базисные функции такого типа.

4.2. ВАРИАЦИОННАЯ ПОСТАНОВКА

Будем считать, что на границе $S = S_1 \cup S_2$ расчётной области Ω, в которой определено уравнение (4.2), заданы краевые условия двух типов:

$$\left(\vec{\mathbf{A}} \times \vec{\mathbf{n}}\right)\Big|_{S_1} = \vec{\mathbf{A}}^g \times \vec{\mathbf{n}}, \tag{4.7}$$

$$\left(\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}} \times \vec{\mathbf{n}}\right)\Bigg|_{S_2} = \vec{\mathbf{H}}^\Theta \times \vec{\mathbf{n}}, \tag{4.8}$$

где «$\times$» – знак векторного произведения, $\vec{\mathbf{n}}$ – вектор нормали к поверхности S_1 или S_2, а $\vec{\mathbf{A}}^g$ и $\vec{\mathbf{H}}^\Theta$ – известные вектор-функции.

Условие (4.7) фактически задаёт равенство касательных составляющих искомого вектор-потенциала $\vec{\mathbf{A}}$ касательным составляющим заданной на S_1 вектор-функции $\vec{\mathbf{A}}^g$, а условие (4.8) – равенство касательных составляющих напряжённости магнитного поля $\vec{\mathbf{H}}$ (с учётом представления вектора индукции магнитного поля $\vec{\mathbf{B}}$ в виде $\vec{\mathbf{B}} = \operatorname{rot}\vec{\mathbf{A}}$ и связи $\vec{\mathbf{B}}$ с $\vec{\mathbf{H}}$ в виде соотношения $\vec{\mathbf{B}} = \mu\vec{\mathbf{H}}$) касательным составляющим заданной на S_2 вектор-функции $\vec{\mathbf{H}}^\Theta$.

Нетрудно убедиться, что условия равенства нулю касательных либо нормальных составляющих вектора магнитной индукции $\vec{\mathbf{B}}$ (а это наиболее часто ис-

пользуемые условия симметрии и условия, задаваемые на границах большого объёма) являются частным случаем краевых условий (4.7) и (4.8).

Действительно, поскольку $\vec{\mathbf{B}}$ представляется в виде $\vec{\mathbf{B}} = \operatorname{rot}\vec{\mathbf{A}}$, условие

$$\vec{\mathbf{B}} \times \vec{\mathbf{n}}\Big|_{S_2} = \mathbf{0} \tag{4.9}$$

равенства нулю касательных составляющих вектора индукции магнитного поля $\vec{\mathbf{B}}$ на участке S_2 границы S расчётной области Ω является частным случаем краевого условия (4.8): условие (4.9) будет выполнено, если в (4.8) положить $\vec{\mathbf{H}}^{\Theta} = \mathbf{0}$.

Условие же равенства нулю нормальных составляющих индукции магнитного поля

$$\vec{\mathbf{B}} \cdot \vec{\mathbf{n}}\Big|_{S_1} = 0 \tag{4.10}$$

на участке S_1 границы S выполняется при постоянном значении касательных составляющих $\vec{\mathbf{A}}$ на S_1 (что сразу следует из определения операции rot), т.е. для выполнения (4.10) достаточно в (4.7) взять $\vec{\mathbf{A}}^{g} = \mathbf{0}$.

Эквивалентная вариационная формулировка в форме Галёркина для уравнения (4.2) с краевыми условиями (4.7) и (4.8) имеет вид

$$\int_{\Omega} \frac{1}{\mu} \operatorname{rot}\vec{\mathbf{A}} \cdot \operatorname{rot}\vec{\mathbf{\Psi}}\, d\Omega + \int_{\Omega} \gamma \vec{\mathbf{A}} \cdot \vec{\mathbf{\Psi}}\, d\Omega = \\ = \int_{\Omega} \vec{\mathbf{F}} \cdot \vec{\mathbf{\Psi}}\, d\Omega + \int_{S_2} \left(\vec{\mathbf{H}}^{\Theta} \times \vec{\mathbf{n}}\right) \cdot \vec{\mathbf{\Psi}}\, dS \quad \forall \vec{\mathbf{\Psi}} \in H_0^{\mathrm{rot}}, \tag{4.11}$$

где запись $\vec{\mathbf{R}} \cdot \vec{\mathbf{P}}$ означает скалярное произведение вектор-функций $\vec{\mathbf{R}}$ и $\vec{\mathbf{P}}$, а «$\times$» по-прежнему – знак векторного произведения.

Решение $\vec{\mathbf{A}}$ и пробные вектор-функции $\vec{\mathbf{\Psi}}$ должны принадлежать пространству $\mathbb{H}^{\mathrm{rot}}$, содержащему все вектор-функции $\vec{\mathbf{\Phi}}$, которые сами суммируемы с квадратом на Ω (т.е. $\int_{\Omega} \vec{\mathbf{\Phi}}^2 d\Omega \equiv \int_{\Omega} \vec{\mathbf{\Phi}} \cdot \vec{\mathbf{\Phi}}\, d\Omega < \infty$) и для которых на Ω существуют и суммируемы с квадратом вектор-функции $\operatorname{rot}\vec{\mathbf{\Phi}}$ (т.е. $\int_{\Omega} \left(\operatorname{rot}\vec{\mathbf{\Phi}}\right)^2 d\Omega \equiv \int_{\Omega} \operatorname{rot}\vec{\mathbf{\Phi}} \cdot \operatorname{rot}\vec{\mathbf{\Phi}}\, d\Omega < \infty$).

Пространство же $\mathbb{H}_0^{\mathrm{rot}}$ пробных вектор-функций $\vec{\mathbf{\Psi}}$ является подмножеством всех вектор-функций из пространства $\mathbb{H}^{\mathrm{rot}}$ таких, что их касательные составляю-

щие равны нулю на границе S_1 (т.е. для всех $\vec{\mathbf{\Psi}} \in \mathbb{H}_0^{\mathrm{rot}}$ должно выполняться $\left(\vec{\mathbf{\Psi}} \times \vec{\mathbf{n}}\right)\Big|_{S_1} = \mathbf{0}$, сравните с (4.7)).

Решение $\vec{\mathbf{A}}$ вариационного уравнения (4.11) следует искать на множестве $\mathbb{H}_g^{\mathrm{rot}}$ всех тех вектор-функций из пространства $\mathbb{H}^{\mathrm{rot}}$, которые удовлетворяют краевому условию (4.7). Заметим, что при $\vec{\mathbf{A}}^g = \mathbf{0}$ на S_1 пространство $\mathbb{H}_g^{\mathrm{rot}}$ тождественно совпадает с $\mathbb{H}_0^{\mathrm{rot}}$, а если в решаемой задаче отсутствуют краевые условия вида (4.7), то оба пространства $\mathbb{H}_g^{\mathrm{rot}}$ и $\mathbb{H}_0^{\mathrm{rot}}$ тождественно совпадают с пространством $\mathbb{H}^{\mathrm{rot}}$.

Докажем эквивалентность вариационного уравнения (4.11) векторной краевой задаче (4.2), (4.7), (4.8).

Преобразуем первое слагаемое вариационного уравнения (4.11) с использованием векторной формулы Грина:

$$\int_\Omega \mathrm{rot}\left(\frac{1}{\mu}\mathrm{rot}\,\vec{\mathbf{A}}\right)\cdot\vec{\mathbf{\Psi}}\,d\Omega + \int_S \left(\frac{1}{\mu}\mathrm{rot}\,\vec{\mathbf{A}}\times\vec{\mathbf{n}}\right)\cdot\vec{\mathbf{\Psi}}\,dS + \int_\Omega \gamma\vec{\mathbf{A}}\cdot\vec{\mathbf{\Psi}}\,d\Omega =$$
$$= \int_\Omega \vec{\mathbf{F}}\cdot\vec{\mathbf{\Psi}}\,d\Omega + \int_{S_2}\left(\vec{\mathbf{H}}^{\Theta}\times\vec{\mathbf{n}}\right)\cdot\vec{\mathbf{\Psi}}\,dS \qquad \forall\vec{\mathbf{\Psi}} \in \mathbb{H}_0^{\mathrm{rot}}. \tag{4.12}$$

Поскольку пробные вектор-функции $\vec{\mathbf{\Psi}}$ являются произвольными элементами $\mathbb{H}_0^{\mathrm{rot}}$, мы можем выбирать из них вектор-функции $\vec{\mathbf{\Psi}}^0$, касательные составляющие которых равны нулю на всей границе S. Для таких вектор-функций поверхностные интегралы в уравнении (4.12) оказываются равными нулю, и это уравнение принимает вид

$$\int_\Omega \left(\mathrm{rot}\left(\frac{1}{\mu}\mathrm{rot}\,\vec{\mathbf{A}}\right) + \gamma\vec{\mathbf{A}} - \vec{\mathbf{F}}\right)\cdot\vec{\mathbf{\Psi}}^0\,d\Omega = \mathbf{0} \quad \forall\vec{\mathbf{\Psi}}^0 \in \mathbb{H}_0^{\mathrm{rot}} : \vec{\mathbf{\Psi}}^0\times\vec{\mathbf{n}}\Big|_S = \mathbf{0}, \tag{4.13}$$

откуда следует, что для вектор-потенциала $\vec{\mathbf{A}}$ почти всюду в области Ω выполняется дифференциальное уравнение (4.2).

Мы показали, что для вектор-функции $\vec{\mathbf{A}}$, удовлетворяющей вариационному уравнению (4.11), почти всюду выполняется дифференциальное уравнение (4.2). Тогда, учитывая что $S = S_1 \cup S_2$ и то, что сумма объёмных интегралов в уравнении (4.12) равна нулю (поскольку нами уже доказана справедливость уравнения (4.2)), из уравнения (4.12) получим:

$$\int\limits_{S_1}\left(\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}}\times\vec{\mathbf{n}}\right)\cdot\vec{\mathbf{\Psi}}\,dS+$$

$$+\int\limits_{S_2}\left(\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}}\times\vec{\mathbf{n}}-\vec{\mathbf{H}}^{\Theta}\times\vec{\mathbf{n}}\right)\cdot\vec{\mathbf{\Psi}}\,dS=0\quad\forall\vec{\mathbf{\Psi}}\in\mathbb{H}_0^{\mathrm{rot}}. \tag{4.14}$$

Так как касательная составляющая $\vec{\mathbf{\Psi}}$ на границе S_1 равна нулю (по определению пространства $\mathbb{H}_0^{\mathrm{rot}}$), то с учётом того, что

$$\left(\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}}\times\vec{\mathbf{n}}\right)\cdot\vec{\mathbf{\Psi}}=-\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}}\cdot\left(\vec{\mathbf{\Psi}}\times\vec{\mathbf{n}}\right)=-\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}}\cdot\mathbf{0}=\mathbf{0}\,,$$

первое слагаемое в (4.14) обращается в нуль. Поэтому

$$\int\limits_{S_2}\left(\frac{1}{\mu}\operatorname{rot}\vec{\mathbf{A}}\times\vec{\mathbf{n}}-\vec{\mathbf{H}}^{\Theta}\times\vec{\mathbf{n}}\right)\cdot\vec{\mathbf{\Psi}}\,dS=0\qquad\forall\vec{\mathbf{\Psi}}\in\mathbb{H}_0^{\mathrm{rot}}. \tag{4.15}$$

Отсюда следует, что на границе S_2 выполняется (почти всюду) краевое условие (4.8).

Краевое же условие (4.7) выполняется по определению пространства $\mathbb{H}_g^{\mathrm{rot}}$, в котором ищется решение $\vec{\mathbf{A}}$.

Итак, мы доказали, что решение $\vec{\mathbf{A}}\in\mathbb{H}_g^{\mathrm{rot}}$ вариационного уравнения (4.11) действительно удовлетворяет и уравнению (4.2), и краевым условиям (4.7) и (4.8). Чтобы доказать обратное утверждение, достаточно провести рассмотренные выше рассуждения в обратном порядке (и мы предоставляем читателю провести такое доказательство самостоятельно в качестве несложного упражнения). Это и завершит доказательство эквивалентности вариационного уравнения (4.11) векторной краевой задаче (4.2), (4.7), (4.8).

4.3. ПРИБЛИЖЕНИЕ РЕШЕНИЯ НА ПОДПРОСТРАНСТВАХ

Чтобы использовать метод Галёркина для нахождения решения уравнения (4.11), мы должны прежде всего определить конечномерное подпространство $\mathbb{V}^{\mathrm{rot}}$ бесконечномерного пространства $\mathbb{H}^{\mathrm{rot}}$. Соответственно обозначим через $\mathbb{V}_g^{\mathrm{rot}}$ подмножество всех функций из $\mathbb{V}^{\mathrm{rot}}$, удовлетворяющих главному краевому условию.

Приближённое решение уравнения Галёркина (4.11) определяется как вектор-функция $\vec{\mathbf{A}}^h$, принадлежащая подмножеству $\mathbb{V}_g^{\mathrm{rot}}$ конечномерного подпространст-

ва $\mathbb{V}^{\mathrm{rot}}$ пространства $\mathbb{H}^{\mathrm{rot}}$, такая, что соответствующее вариационное уравнение должно выполняться для любой вектор-функции $\vec{\mathbf{\Psi}}^h$ из подпространства $\mathbb{V}_0^{\mathrm{rot}}$ пространства $\mathbb{H}_0^{\mathrm{rot}}$, т.е.

$$\int_\Omega \frac{1}{\mu}\mathrm{rot}\,\vec{\mathbf{A}}^h \cdot \mathrm{rot}\,\vec{\mathbf{\Psi}}^h d\Omega + \int_\Omega \gamma \vec{\mathbf{A}}^h \cdot \vec{\mathbf{\Psi}}^h d\Omega = \\ = \int_\Omega \vec{\mathbf{F}} \cdot \vec{\mathbf{\Psi}}^h d\Omega + \int_{S_2} \left(\vec{\mathbf{H}}^\Theta \times \vec{\mathbf{n}}\right) \cdot \vec{\mathbf{\Psi}}^h dS \quad \forall \vec{\mathbf{\Psi}}^h \in \mathbb{V}_0^{\mathrm{rot}}. \tag{4.16}$$

Как правило, *конечномерное* пространство $\mathbb{V}^{\mathrm{rot}}$ определяется как линейное пространство с базисом из n вектор-функций $\vec{\psi}_i$ и, естественно, любые функции из $\mathbb{V}_g^{\mathrm{rot}}$ и $\mathbb{V}_0^{\mathrm{rot}}$ могут быть представлены в виде линейной комбинации базисных вектор-функций $\vec{\psi}_i$. Поэтому приближённое решение $\vec{\mathbf{A}}^h$ ищется в виде линейной комбинации базисных вектор-функций $\vec{\psi}_i$:

$$\vec{\mathbf{A}}^h = \sum_{j=1}^{n} q_j \vec{\psi}_j . \tag{4.17}$$

Функция $\vec{\mathbf{A}}^h$ в виде линейной комбинации (4.17) подставляется в вариационное уравнение (4.11), а пробная вектор-функция $\vec{\mathbf{\Psi}}$ в этом уравнении заменяется поочерёдно на все базисные функции пространства $\mathbb{V}_0^{\mathrm{rot}}$. В результате получается система линейных уравнений вида

$$\sum_{j\in N}\left(\int_\Omega \frac{1}{\mu}\mathrm{rot}\,\vec{\psi}_j \cdot \mathrm{rot}\,\vec{\psi}_i d\Omega + \int_\Omega \gamma \vec{\psi}_j \cdot \vec{\psi}_i d\Omega\right) q_j = \\ = \int_\Omega \vec{\mathbf{F}} \cdot \vec{\psi}_i d\Omega + \int_{S_2} \left(\vec{\mathbf{H}}^\Theta \times \vec{\mathbf{n}}\right) \cdot \vec{\psi}_i dS, \quad i \in N_0, \tag{4.18}$$

где N – множество всех номеров от 1 до n (напомним, что n – размерность пространства $\mathbb{V}^{\mathrm{rot}}$), а N_0 – подмножество N с удалёнными из него $n - n_0$ номерами i тех базисных вектор-функций $\vec{\psi}_i$ конечномерного пространства $\mathbb{V}^{\mathrm{rot}}$, которые не входят в пространство $\mathbb{V}_0^{\mathrm{rot}}$ (т.е. размерность конечномерного подпространства $\mathbb{V}_0^{\mathrm{rot}}$ равна n_0, а его базисные функции – это базисные вектор-функции $\vec{\psi}_i \in \mathbb{V}^{\mathrm{rot}}$ с номерами $i \in N_0$).

Краевые условия (4.8) являются *естественными*, поскольку учитываются автоматически в вариационной постановке. Вклады от них, как и от краевых усло-

вий второго рода в узловом МКЭ, идут только в правую часть СЛАУ (4.18) (в компоненты её вектора правой части с такими номерами i, для которых базисные вектор-функции $\vec{\psi}_i$ имеют на S_2 *отличные от нуля касательные* к S_2 составляющие).

Краевое же условие (4.7), как и в узловом МКЭ, называется *главным*. Здесь мы рассмотрим, как можно учесть главное краевое условие (4.7) для наиболее часто встречающегося на практике случая, когда оно является однородным (т.е. $\vec{\mathbf{A}}^g \times \vec{\mathbf{n}} \equiv \mathbf{0}$).

Как мы увидим чуть позже, при решении уравнения (4.2) с использованием векторного МКЭ базисные вектор-функции $\vec{\psi}_i$ строятся так, что *все* базисные функции конечномерного пространства $\mathbb{V}^{\mathrm{rot}}$ *с индексами* $i \in N_0$ *имеют нулевые касательные к* S_1 *составляющие* (они же и образуют базис подпространства $\mathbb{V}_0^{\mathrm{rot}}$). Поэтому для выполнения однородных главных краевых условий достаточно к n_0 уравнениям системы (4.18) добавить $n - n_0$ уравнений вида

$$q_i = 0, \qquad i \in N \setminus N_0. \tag{4.19}$$

Таким образом, если конечноэлементная СЛАУ была сгенерирована с фиктивными уравнениями (т.е. наряду с уравнениями (4.18) для $i \in N_0$ генерировались аналогичные уравнения и для $i \in N \setminus N_0$), то учесть однородные главные краевые условия можно одним из следующих двух очень простых способов. Можно либо вместо сгенерированных строк с индексами $i \in N \setminus N_0$ вставить строки вида (4.19) (т.е. с единицей на диагонали матрицы и нулями во всех внедиагональных элементах этих строк и в соответствующей компоненте вектора правой части). После этого можно очень легко симметризовать матрицу $\mathbf{C}$ полученной системы, обнулив все её компоненты C_{ij} с индексами $j \in N \setminus N_0$ (в этом случае мы сохраним соответствие номеров компонент вектора весов $\mathbf{q}$ номерам базисных функций). А можно просто перенумеровать неизвестные q_i так, чтобы в получаемой СЛАУ остались только те веса q_i, которые в исходной нумерации имели номера $i \in N_0$. В этом случае номера компонент вектора весов после решения полученной СЛАУ с квадратной матрицей размерности n_0 не будут соответствовать номерам «своих» базисных функций, и поэтому нужно либо после решения конечноэлементной СЛАУ вернуться к исходной нумерации весов, либо перенуме-

ровать базисные функции так, чтобы все те функции, которые в исходной нумерации имели номера $i \in N \setminus N_0$, после перенумерации получили бы номера $i \le n_0$ (напомним, что n_0 – количество номеров в множестве N_0).

4.4. ЭНЕРГЕТИЧЕСКОЕ СКАЛЯРНОЕ ПРОИЗВЕДЕНИЕ И ПОГРЕШНОСТЬ ПРИБЛИЖЁННОГО РЕШЕНИЯ В ЭНЕРГЕТИЧЕСКОЙ НОРМЕ

Рассмотрим билинейную форму, определяющую левую часть уравнения Галёркина (4.11):

$$L\left(\vec{\mathbf{A}}, \vec{\mathbf{\Psi}}\right) = \int_{\Omega} \frac{1}{\mu} \operatorname{rot} \vec{\mathbf{A}} \cdot \operatorname{rot} \vec{\mathbf{\Psi}} d\Omega + \int_{\Omega} \gamma \vec{\mathbf{A}} \cdot \vec{\mathbf{\Psi}} d\Omega . \tag{4.20}$$

Нетрудно убедиться, что при $\gamma > 0$ эта билинейная форма удовлетворяет всем аксиомам скалярного произведения. Поэтому мы можем определить скалярное произведение двух произвольных вектор-функций $\vec{\mathbf{\Phi}}$ и $\vec{\mathbf{\Psi}}$ из множества $\mathbb{H}^{\mathrm{rot}}$ в виде

$$(\vec{\mathbf{\Phi}}, \vec{\mathbf{\Psi}})_э = \int_{\Omega} \frac{1}{\mu} \operatorname{rot} \vec{\mathbf{\Phi}} \cdot \operatorname{rot} \vec{\mathbf{\Psi}} d\Omega + \int_{\Omega} \gamma \vec{\mathbf{\Phi}} \cdot \vec{\mathbf{\Psi}} d\Omega . \tag{4.21}$$

Это скалярное произведение будем называть **энергетическим**. Соответственно **энергетической нормой** вектор-функции $\vec{\mathbf{\Phi}}$ будем называть норму, порождаемую скалярным произведением (4.21):

$$\left\| \vec{\mathbf{\Phi}} \right\|_э = \sqrt{(\vec{\mathbf{\Phi}}, \vec{\mathbf{\Phi}})_э} . \tag{4.22}$$

Определённые соотношениями (4.21) и (4.22) энергетическое скалярное произведение и энергетическая норма очень удобны для изучения погрешности $\left\| \vec{\mathbf{A}}^h - \vec{\mathbf{A}} \right\|$ приближённого решения $\vec{\mathbf{A}}^h$, которое ищется как решение уравнения (4.16). Чуть ниже мы изложим технику оценки этой погрешности.

Запишем уравнения (4.11) и (4.16) в виде

$$\left(\vec{\mathbf{A}}, \vec{\mathbf{\Psi}}\right)_э = f\left(\vec{\mathbf{\Psi}}\right) \quad \forall \vec{\mathbf{\Psi}} \in \mathbb{H}_0^{\mathrm{rot}}, \tag{4.23}$$

$$\left(\vec{\mathbf{A}}^h, \vec{\mathbf{\Psi}}^h\right)_э = f\left(\vec{\mathbf{\Psi}}^h\right) \quad \forall \vec{\mathbf{\Psi}}^h \in \mathbb{V}_0^{\mathrm{rot}}, \tag{4.24}$$

где f – линейная форма, определяемая следующим соотношением (сравните с правыми частями уравнений (4.11) и (4.16)):

$$f\left(\vec{\mathbf{\Phi}}\right) = \int_{\Omega} \vec{\mathbf{F}} \cdot \vec{\mathbf{\Phi}} d\Omega + \int_{S_2} \left(\vec{\mathbf{H}}^{\Theta} \times \vec{\mathbf{n}}\right) \cdot \vec{\mathbf{\Phi}} dS \,. \tag{4.25}$$

Поскольку $\mathbb{V}_0^{\mathrm{rot}}$ является подпространством пространства $\mathbb{H}_0^{\mathrm{rot}}$, уравнение Галёркина (4.11) справедливо и для всех пробных вектор-функций $\vec{\mathbf{\Psi}}^h$ из $\mathbb{V}_0^{\mathrm{rot}}$:

$$\left(\vec{\mathbf{A}}, \vec{\mathbf{\Psi}}^h\right)_{\text{э}} = f\left(\vec{\mathbf{\Psi}}^h\right) \quad \forall \vec{\mathbf{\Psi}}^h \in \mathbb{V}_0^{\mathrm{rot}} \,. \tag{4.26}$$

Тогда, из (4.24) и (4.26) получаем

$$\left(\vec{\mathbf{A}}^h - \vec{\mathbf{A}}, \vec{\mathbf{\Psi}}^h\right)_{\text{э}} = 0 \quad \forall \vec{\mathbf{\Psi}}^h \in \mathbb{V}_0^{\mathrm{rot}} , \tag{4.27}$$

т.е. приближённое решение $\vec{\mathbf{A}}^h$ является *ортогональной проекцией* точного решения $\vec{\mathbf{A}}$ на подпространство $\mathbb{V}_g^{\mathrm{rot}}$ ($\mathbb{V}_g^{\mathrm{rot}}$ подмножество вектор-функций из $\mathbb{V}^{\mathrm{rot}}$, удовлетворяющих главному краевому условию (4.7)).

С учётом соотношения (4.27) не представляет труда доказать, что решение $\vec{\mathbf{A}}^h$ уравнения (4.16) – *ближайшая в энергетической норме* к точному решению $\vec{\mathbf{A}}$ (уравнения (4.11)) вектор-функция среди всех вектор-функций $\vec{\mathbf{\Phi}}^h$ из $\mathbb{V}_g^{\mathrm{rot}}$.

Действительно, для любой вектор-функции $\vec{\mathbf{\Phi}}^h \in \mathbb{V}_g^{\mathrm{rot}}$ справедливо

$$\begin{aligned}
\left\|\vec{\mathbf{A}} - \vec{\mathbf{\Phi}}^h\right\|_{\text{э}}^2 &= \left(\vec{\mathbf{A}} - \vec{\mathbf{\Phi}}^h, \vec{\mathbf{A}} - \vec{\mathbf{\Phi}}^h\right)_{\text{э}} = \\
&= \left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h + \vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h, \vec{\mathbf{A}} - \vec{\mathbf{A}}^h + \vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h\right)_{\text{э}} = \\
&= \left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h, \vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right)_{\text{э}} + 2\left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h, \vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h\right)_{\text{э}} + \left(\vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h, \vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h\right)_{\text{э}} = \\
&= \left\|\vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right\|_{\text{э}}^2 + \left\|\vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h\right\|_{\text{э}}^2 + 2\left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h, \vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h\right)_{\text{э}} .
\end{aligned}$$

Поскольку $\vec{\mathbf{A}}^h$ и $\vec{\mathbf{\Phi}}^h$ являются элементами пространства $\mathbb{V}_g^{\mathrm{rot}}$ вектор-функций, удовлетворяющих на S_1 одному и тому же неоднородному краевому условию, их разность удовлетворяет на S_1 *однородному* главному краевому условию и поэтому вектор-функция $\vec{\mathbf{\Psi}}^h = \vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h$ является элементом пространства $\mathbb{V}_0^{\mathrm{rot}}$, и тогда из (4.27) сразу следует, что $(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h, \vec{\mathbf{A}}^h - \vec{\mathbf{\Phi}}^h)_{\text{э}} = 0$.

Учитывая это, получаем

$$\left\|\vec{\mathbf{A}}-\vec{\mathbf{\Phi}}^{h}\right\|_{э}^{2}=\left\|\vec{\mathbf{A}}-\vec{\mathbf{A}}^{h}\right\|_{э}^{2}+\left\|\vec{\mathbf{A}}^{h}-\vec{\mathbf{\Phi}}^{h}\right\|_{э}^{2}\quad\forall\vec{\mathbf{\Phi}}^{h}\in\mathbb{V}_{g}^{\mathrm{rot}},$$

откуда

$$\left\|\vec{\mathbf{A}}-\vec{\mathbf{A}}^{h}\right\|_{э}^{2}=\left\|\vec{\mathbf{A}}-\vec{\mathbf{\Phi}}^{h}\right\|_{э}^{2}-\left\|\vec{\mathbf{A}}^{h}-\vec{\mathbf{\Phi}}^{h}\right\|_{э}^{2}\quad\forall\vec{\mathbf{\Phi}}^{h}\in\mathbb{V}_{g}^{\mathrm{rot}},\tag{4.28}$$

т.е.

$$\left\|\vec{\mathbf{A}}-\vec{\mathbf{A}}^{h}\right\|_{э}^{2}\le\left\|\vec{\mathbf{A}}-\vec{\mathbf{\Phi}}^{h}\right\|_{э}^{2}\quad\forall\vec{\mathbf{\Phi}}^{h}\in\mathbb{V}_{g}^{\mathrm{rot}}.\tag{4.29}$$

Итак, вектор-функция $\vec{\mathbf{A}}^{h}$, полученная как решение уравнения (4.16), действительно является вектор-функцией, *ближайшей* (в энергетической норме) к $\vec{\mathbf{A}}$ (точному решению, удовлетворяющему уравнению (4.11)) среди всех вектор-функций из пространства $\mathbb{V}_{g}^{\mathrm{rot}}$.

Таким образом, при построении приближённого решения нам нужно лишь позаботиться о том, чтобы в конечномерном пространстве $\mathbb{V}_{g}^{\mathrm{rot}}$ в принципе были функции, достаточно близкие к искомому решению $\vec{\mathbf{A}}$, а метод Галёркина обеспечит выбор из всех функций пространства $\mathbb{V}_{g}^{\mathrm{rot}}$ *самой близкой* к $\vec{\mathbf{A}}$ (в энергетической норме) вектор-функции $\vec{\mathbf{A}}^{h}$. Пространство же $\mathbb{V}^{\mathrm{rot}}$ (и, естественно, его подмножество $\mathbb{V}_{g}^{\mathrm{rot}}$) в векторном МКЭ (как и в скалярном) определяется сеткой и видом определённых на ячейках сетки локальных базисных вектор-функций.

Сделаем ещё одно замечание о сути введённой нами энергетической нормы. На самом деле, квадрат энергетической нормы разности точного $\vec{\mathbf{A}}$ и приближённого $\vec{\mathbf{A}}^{h}$ решений имеет и довольно важный физический смысл. Рассмотрим его.

Будем считать, что для аппроксимации производных искомого вектор-потенциала $\vec{\mathbf{A}}$ по времени используется четырёхслойная полностью неявная схема аппроксимации по времени и значения точного и приближённого решений на предыдущих временных слоях (которые мы, по-прежнему, будем обозначать через $\vec{\mathbf{A}}^{\Leftarrow 1}$, $\vec{\mathbf{A}}^{\Leftarrow 2}$ и $\vec{\mathbf{A}}^{\Leftarrow 3}$) совпадают. При этом только для уменьшения громоздкости выкладок шаг по времени будем считать постоянным и равным Δt. Тогда энергетическая норма разности $\vec{\mathbf{A}}$ и $\vec{\mathbf{A}}^{h}$ на текущем временном слое может быть записана в следующем виде:

$$\left\|\vec{\mathbf{A}}-\vec{\mathbf{A}}^{h}\right\|_{э}^{2}=\left(\vec{\mathbf{A}}-\vec{\mathbf{A}}^{h},\vec{\mathbf{A}}-\vec{\mathbf{A}}^{h}\right)_{э}=$$

$$= \int\limits_{\Omega} \frac{1}{\mu} \operatorname{rot}\left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right) \cdot \operatorname{rot}\left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right) d\Omega + \int\limits_{\Omega} \gamma \left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right)^2 d\Omega =$$

$$= \int\limits_{\Omega} \frac{1}{\mu} \left(\operatorname{rot} \vec{\mathbf{A}} - \operatorname{rot} \vec{\mathbf{A}}^h\right)^2 d\Omega + \frac{11}{6\Delta t} \int\limits_{\Omega} \sigma \left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right)^2 d\Omega +$$

$$+ \frac{2}{\Delta t^2} \int\limits_{\Omega} \varepsilon \left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right)^2 d\Omega .$$

С учётом того, что $\operatorname{rot} \vec{\mathbf{A}} = \vec{\mathbf{B}}$, $\operatorname{rot} \vec{\mathbf{A}}^h = \vec{\mathbf{B}}^h$ и

$$\left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h\right)^2 = \left(\frac{6\Delta t}{11}\right)^2 \left(\frac{11}{6\Delta t} \vec{\mathbf{A}} - \frac{11}{6\Delta t} \vec{\mathbf{A}}^h\right)^2 =$$

$$= \left(\frac{6\Delta t}{11}\right)^2 \left(\frac{11}{6\Delta t} \vec{\mathbf{A}} - \frac{3}{\Delta t} \vec{\mathbf{A}}^{\Leftarrow 1} + \frac{3}{2\Delta t} \vec{\mathbf{A}}^{\Leftarrow 2} - \frac{1}{3\Delta t} \vec{\mathbf{A}}^{\Leftarrow 3} - \right.$$

$$\left. - \left(\frac{11}{6\Delta t} \vec{\mathbf{A}}^h - \frac{3}{\Delta t} \vec{\mathbf{A}}^{\Leftarrow 1} + \frac{3}{2\Delta t} \vec{\mathbf{A}}^{\Leftarrow 2} - \frac{1}{3\Delta t} \vec{\mathbf{A}}^{\Leftarrow 3}\right)\right)^2 = \left(\frac{6\Delta t}{11}\right)^2 \left(\vec{\mathbf{E}} - \vec{\mathbf{E}}^h\right)^2 ,$$

где $\vec{\mathbf{E}}$ и $\vec{\mathbf{E}}^h$ – аппроксимации производных по времени $-\partial \vec{\mathbf{A}} \big/ \partial t$ и $-\partial \vec{\mathbf{A}}^h \big/ \partial t$ по четырёхслойной полностью неявной схеме, получим

$$\left\| \vec{\mathbf{A}} - \vec{\mathbf{A}}^h \right\|_{\text{э}}^2 = \int\limits_{\Omega} \frac{1}{\mu} \left(\vec{\mathbf{B}} - \vec{\mathbf{B}}^h\right)^2 d\Omega + \frac{6\Delta t}{11} \int\limits_{\Omega} \sigma \left(\vec{\mathbf{E}} - \vec{\mathbf{E}}^h\right)^2 d\Omega +$$

$$+ \frac{72}{121} \int\limits_{\Omega} \varepsilon \left(\vec{\mathbf{E}} - \vec{\mathbf{E}}^h\right)^2 d\Omega . \qquad (4.30)$$

Поскольку метод Галёркина ищет ближайшую (в энергетической норме) к $\vec{\mathbf{A}}$ вектор-функцию $\vec{\mathbf{A}}^h$ среди всех вектор-функций $\vec{\mathbf{\Phi}}^h \in \mathbb{V}_g^{\operatorname{rot}}$ (см. (4.29)), из (4.30) видно, что одновременно с минимизацией $\left\| \vec{\mathbf{A}} - \vec{\mathbf{A}}^h \right\|_{\text{э}}$ обеспечивается близость вычисляемых через $\vec{\mathbf{A}}^h$ индукции магнитного поля $\vec{\mathbf{B}}^h$ и напряжённости электрического поля $\vec{\mathbf{E}}^h$ к «точным» значениям индукции $\vec{\mathbf{B}}$ и напряжённости $\vec{\mathbf{E}}$.

Таким образом, фактически в рассматриваемом нами варианте векторного МКЭ в качестве приближённого решения будет находиться такой вектор-потенциал $\vec{\mathbf{A}}^h$, который на каждом временном слое будет минимизировать функционал

$$\int\limits_{\Omega} \frac{1}{\mu} \left(\vec{\mathbf{B}} - \vec{\mathbf{B}}^h\right)^2 d\Omega + \frac{6\Delta t}{11} \int\limits_{\Omega} \sigma \left(\vec{\mathbf{E}} - \vec{\mathbf{E}}^h\right)^2 d\Omega + \frac{72}{121} \int\limits_{\Omega} \varepsilon \left(\vec{\mathbf{E}} - \vec{\mathbf{E}}^h\right)^2 d\Omega , \qquad (4.31)$$

т.е. будут минимизироваться взвешенные квадраты отклонений индукции магнитного поля и напряжённости электрического поля от своих «точных» значений.

Обратим внимание на то, что в функционале (4.31) перед вторым слагаемым есть сомножитель Δt. Вообще говоря, входящие в (4.31) слагаемые имеют смысл физических энергий – первое слагаемое может быть сопоставлено с энергией магнитного поля $\int\limits_{\Omega}\vec{\mathbf{B}}\vec{\mathbf{H}}d\Omega=\int\limits_{\Omega}\frac{1}{\mu}\vec{\mathbf{B}}^2d\Omega$, второе слагаемое – с потерями энергии электромагнитного поля $\int\limits_{\Omega}\sigma\vec{\mathbf{E}}^2d\Omega$, связанными с переходом этой энергии в тепло, и третье слагаемое – с энергией электрического поля $\int\limits_{\Omega}\varepsilon\vec{\mathbf{E}}^2d\Omega$. При этом в балансе энергий электромагнитного поля энергии магнитного и электрического поля дифференцируются по времени, а потери – нет. Фактически это мы видим и в функционале (4.31) – сомножитель Δt перед вторым слагаемым делает пропорции между входящими в (4.31) погрешностями аналогичными пропорциям соответствующих энергий в балансе энергий электромагнитного поля.

Сделаем ещё одно очень важное замечание о единственности решения задачи (4.2) и эквивалентного ей вариационного уравнения (4.11) (и соответственно уравнения (4.16) для нахождения приближённого решения).

При решении задач, в которых токи смещения являются несущественными, в уравнении (4.1) слагаемое $\varepsilon\frac{\partial^2\vec{\mathbf{A}}}{\partial t^2}$ полагается равным нулю. При этом, хотя коэффициент ε по физическому смыслу является ненулевым для любых материалов, в задачах с несущественными токами смещения удобно именно ε сделать равным нулю – в этом случае и для задачи с несущественными токами смещения остаются справедливыми рассмотренные нами аппроксимации, приводящие к вариационным задачам (4.11) и (4.16).

Коэффициент же удельной проводимости σ может принимать нулевые значения (в том числе и по его физическому смыслу) в отдельных подобластях расчётной области. Тогда при несущественных токах смещения и наличии в расчётной области непроводящих подобластей уравнение (4.1) будет иметь не единственное решение.

Действительно, если $\vec{\mathbf{A}}$ удовлетворяет уравнению (4.1), то для любой скалярной функции U, равной нулю в подобластях с ненулевым σ и отличной от нуля в подобластях, где коэффициент σ равен нулю, уравнению (4.1) будет удовлетво-

рять также и вектор-функция $\vec{\mathbf{A}} + \operatorname{grad} U$. Очевидно, то же самое можно сказать и о решении уравнения Галёркина (4.11) – если к вектор-функции $\vec{\mathbf{A}}$, удовлетворяющей этому уравнению, добавить градиент скалярной функции U такой, что U равна нулю в подобластях с ненулевым γ и отлична от нуля в тех подобластях, где γ равен нулю, полученная вектор-функция также будет удовлетворять уравнению (4.11).

Однако при несущественных токах смещения (когда в уравнении (4.1) коэффициент ε полагается равным нулю) в подобластях, где равен нулю и коэффициент σ, физический смысл имеет лишь $\operatorname{rot} \vec{\mathbf{A}}$, а производная по времени вектор-потенциала $\vec{\mathbf{A}}$ (как и сам вектор-потенциал $\vec{\mathbf{A}}$) никакого физического смысла не имеет. Поэтому (при $\varepsilon = 0$) мы можем найти *любой* из тех элементов $\vec{\mathbf{A}}^h$ пространства $\mathbb{V}_g^{\operatorname{rot}}$, которые в подобластях с нулевым σ обеспечивают максимальную близость $\vec{\mathbf{B}}^h = \operatorname{rot} \vec{\mathbf{A}}^h$ к $\vec{\mathbf{B}} = \operatorname{rot} \vec{\mathbf{A}}$.

Очевидно, что именно такой вектор-потенциал $\vec{\mathbf{A}}^h = \sum_i q_i \vec{\psi}_i$ и будет найден в результате решения системы (4.18) (с добавленными к ней уравнениями, обеспечивающими выполнение главных краевых условий). Действительно, как было показано чуть выше, приближённое решение минимизирует энергетическую норму разности точного $\vec{\mathbf{A}}$ и приближённого $\vec{\mathbf{A}}^h$ решений (см. (4.29)). А при $\varepsilon = 0$ квадрат этой нормы может быть записан в виде

$$\left\| \vec{\mathbf{A}} - \vec{\mathbf{A}}^h \right\|_э^2 = \int\limits_{\Omega_0} \frac{1}{\mu} \left(\operatorname{rot} \vec{\mathbf{A}} - \operatorname{rot} \vec{\mathbf{A}}^h \right)^2 d\Omega + \\ + \int\limits_{\Omega \setminus \Omega_0} \left(\frac{1}{\mu} \left(\operatorname{rot} \vec{\mathbf{A}} - \operatorname{rot} \vec{\mathbf{A}}^h \right)^2 + \gamma \left(\vec{\mathbf{A}} - \vec{\mathbf{A}}^h \right)^2 \right) d\Omega, \qquad (4.32)$$

где Ω_0 – объединение всех подобластей расчётной области Ω, в которых коэффициент σ равен нулю. Очевидно, что минимизация (4.32) не гарантирует близости в подобласти Ω_0 приближённого решения $\vec{\mathbf{A}}^h$ к некоторому *фиксированному* точному решению $\vec{\mathbf{A}}$. Но это и не нужно – по смыслу задачи вполне достаточно получить такой $\vec{\mathbf{A}}^h$, чтобы в подобласти Ω_0 вычисленная по $\vec{\mathbf{A}}^h$ индукция магнитного поля $\vec{\mathbf{B}}^h = \operatorname{rot} \vec{\mathbf{A}}^h$ была максимально близка к $\vec{\mathbf{B}} = \operatorname{rot} \vec{\mathbf{A}}$, что с успехом и обеспечивает метод Галёркина.

Тем не менее при численной реализации метода Галёркина (для задач с несущественными токами смещения и наличием подобластей с нулевой проводимостью) возникают определённые трудности, связанные с вырожденностью системы уравнений (4.18) (которая остаётся вырожденной даже после добавления в неё уравнений, обеспечивающих выполнение главных краевых условий). Для решения таких задач нужно либо использовать специальные методы решения СЛАУ, учитывающие её возможную вырожденность, либо изменять постановку задачи так, чтобы устранить вырожденность полученной СЛАУ (к ним относятся, например, постановки, основанные на совместном использовании векторного и узлового МКЭ).

4.5. ПРИНЦИПЫ ПОСТРОЕНИЯ БАЗИСНЫХ ВЕКТОР-ФУНКЦИЙ ИЗ ПРОСТРАНСТВА $\mathbb{H}^{\mathrm{rot}}$ НА ДВУМЕРНЫХ СЕТКАХ

4.5.1. ОБЩИЕ ПРИНЦИПЫ

Базисные функции в векторном МКЭ, как и в узловом, строятся в виде полиномов пространственных координат, определяемых на ячейках сетки. Однако векторный МКЭ имеет одно довольно существенное отличие – базисные функции в нём не должны быть полностью непрерывными. Допустимыми для вариационной задачи (4.11) являются вектор-функции, у которых на границах конечных элементов *непрерывны* только *касательные* к этим границам составляющие. Разрывность же нормальных составляющих базисных вектор-функций не только не создаёт проблем при построении конечноэлементного решения, но во многих случаях является даже необходимым свойством. Дело в том, что при решении многих практических задач довольно часто возникают ситуации, когда напряжённость электрического поля $\vec{\mathbf{E}} = -\partial\vec{\mathbf{A}}/\partial t$ имеет разрывные нормальные составляющие – тогда и точное решение $\vec{\mathbf{A}}$ краевой задачи (4.2) или вариационной задачи (4.11) будет иметь разрывы в своей нормальной составляющей (на тех границах, где разрывен коэффициент γ).

Здесь мы рассмотрим основные принципы построения базисных вектор-функций, используемых для конечноэлементной аппроксимации вариационного уравнения (4.11).

4.5.2. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ НА ПРЯМОУГОЛЬНИКЕ

Проще всего можно построить базисные вектор-функции, определяющие конечномерное пространство $\mathbb{V}^{\mathrm{rot}}$, на конечноэлементной сетке с прямоугольными ячейками.

Рассмотрим прямоугольный конечный элемент $\Omega_{rs} = \left[x_r, x_{r+1}\right] \times \left[y_s, y_{s+1}\right]$, изображённый на рис. 35. Определим на нём четыре локальные базисные вектор-функции:

$$\hat{\vec{\psi}}_1 = \begin{pmatrix} 0 \\ \dfrac{x_{r+1} - x}{h_x} \end{pmatrix}, \qquad \hat{\vec{\psi}}_2 = \begin{pmatrix} 0 \\ \dfrac{x - x_r}{h_x} \end{pmatrix},$$

$$\hat{\vec{\psi}}_3 = \begin{pmatrix} \dfrac{y_{s+1} - y}{h_y} \\ 0 \end{pmatrix}, \qquad \hat{\vec{\psi}}_4 = \begin{pmatrix} \dfrac{y - y_s}{h_y} \\ 0 \end{pmatrix}. \qquad (4.33)$$

Очевидно, что базисные вектор-функции $\hat{\vec{\psi}}_1$ и $\hat{\vec{\psi}}_2$ направлены вдоль оси y и их модули $\left|\hat{\vec{\psi}}_1\right|$ и $\left|\hat{\vec{\psi}}_2\right|$ меняются линейно вдоль оси x ($\left|\hat{\vec{\psi}}_1\right|$ – от единицы на ребре 1 до нуля на ребре 2, а $\left|\hat{\vec{\psi}}_2\right|$ – от единицы на ребре 2 до нуля на ребре 1).

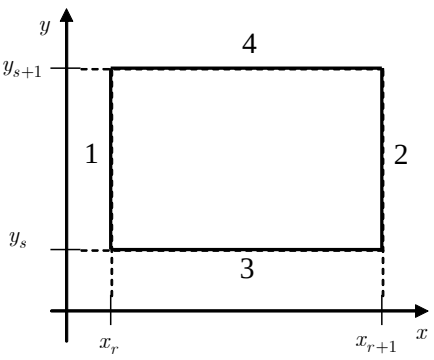


Рис. 35. Локальная нумерация рёбер и базисных вектор-функций на прямоугольном конечном элементе

Аналогично базисные вектор-функции $\hat{\vec{\psi}}_3$ и $\hat{\vec{\psi}}_4$ направлены вдоль оси x и их модули меняются линейно вдоль оси y.

Каждая из базисных вектор-функций (4.33) только на одном ребре прямоугольника Ω_{rs} имеет ненулевую касательную составляющую: $\hat{\vec{\psi}}_1$ – на ребре 1, $\hat{\vec{\psi}}_2$ – на ребре 2 и т.д. Поэтому такие базисные вектор-функции ассоциируют с соответствующими рёбрами (на которых их касательные составляющие не равны нулю) и иногда называют **рёберными** (или **edge**-) функциями, а конечные элементы с такими базисными функциями – **edge-элементами**.

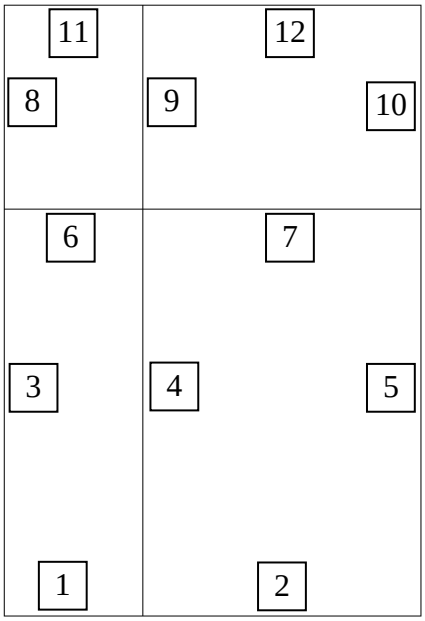


Рис. 36. Нумерация рёбер и соответствующих глобальных базисных вектор-функций

Как мы увидим чуть ниже, термин «edge» вполне уместен для базисных вектор-функций из $\mathbb{H}^{\mathrm{rot}}$ и соответствующих конечных элементов *первого порядка*. Базисные же вектор-функции высоких порядков могут быть ассоциированы уже не только с рёбрами, но и с самими элементами (или с их гранями в трёхмерном случае), и поэтому для таких функций более корректным является название «базисные вектор-функции из $\mathbb{H}^{\mathrm{rot}}$» (которое мы в основном и будем использовать).

Глобальные базисные edge-функции также ассоциируются с рёбрами (поскольку имеют ненулевую касательную составляющую только на одном ребре). Каждая из таких вектор-функций получается в результате сшивки соответствующих локальных базисных вектор-функций двух конечных элементов, содержащих это ребро, и не равна нулю только на этих (двух) элементах. Для сетки, изображённой на рис. 36, на рис. 37 показан вид двух глобальных базисных вектор-функций, одна из которых $\vec{\psi}_7$ ассоциирована с ребром, параллельным оси x, а другая $\vec{\psi}_4$ – с ребром, параллельным оси y. Очевидно, что определённые таким образом глобальные базисные вектор-функции (и соответственно любые их линейные комбинации) будут иметь непрерывные касательные составляющие во всей расчётной области.

На рис. 38 в качестве примера изображена вектор-функция

$$\vec{\mathbf{f}} = \vec{\psi}_1 + \vec{\psi}_2 - 1.5\vec{\psi}_3 - 0.6\vec{\psi}_4 + 1.5\vec{\psi}_5 - 0.2\vec{\psi}_6 - 0.2\vec{\psi}_7 - \\ - 0.5\vec{\psi}_8 - 0.2\vec{\psi}_9 + 0.5\vec{\psi}_{10} - 0.4\vec{\psi}_{11} - 0.4\vec{\psi}_{12} \ ,$$

являющаяся линейной комбинацией базисных вектор-функций, нумерация которых соответствует нумерации рёбер на рис. 36. Эта вектор-функция может быть проинтерпретирована, например, как напряжённость некоторого электрического поля в области с разрывным (на рёбрах 6 и 7, см. рис. 36) коэффициентом удельной проводимости σ (или разрывным коэффициентом диэлектрической проницаемости ε), причём отношение значений соответствующего коэффициента (взятых из разных подобластей) на этой границе равно примерно 3 (и таким же является отношение на этой границе нормальных составляющих $\vec{\mathrm{f}}$).

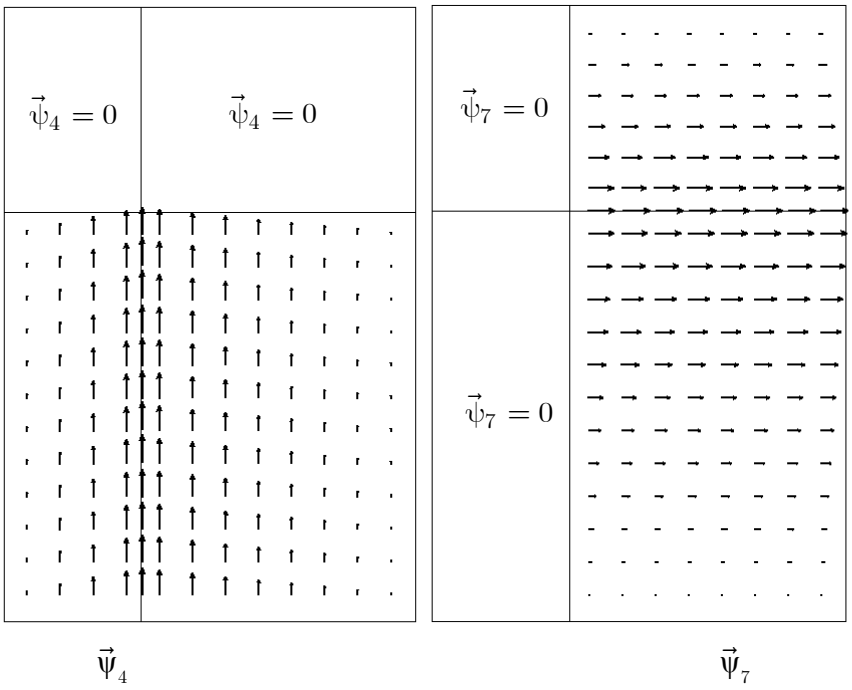


Рис. 37. Вид глобальных базисных вектор-функций на прямоугольной сетке

Рассмотренные нами глобальные базисные вектор-функции, порождённые из локальных вектор-функций вида (4.33), являются вектор-функциями наименьшей степени, которые допустимы для решения вариационной задачи (4.11) на прямоугольной сетке. В векторном МКЭ эти функции обычно называют функциями первого порядка, хотя, очевидно, они не представляют точно даже такие векторные поля, у которых x-компонента линейно зависит от x, а y-компонента линейно зависит от y. Тем не менее такие базисные вектор-функции позволяют аппроксимировать любые векторные поля с погрешностью $O\big(h\big)$ в энергетической норме (h – характерный размер конечного элемента), т.е. конечные элементы с этими базисными вектор-функциями имеют первый порядок аппроксимации.

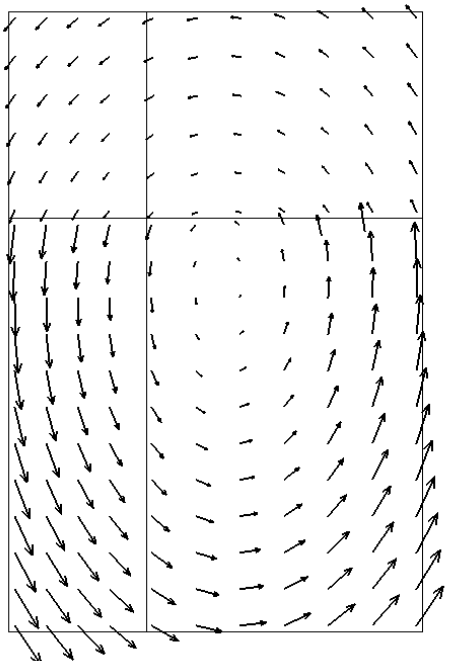


Рис. 38. Вид линейной комбинации глобальных базисных вектор-функций

Базисные вектор-функции высоких порядков определяются, как и в узловом МКЭ, с помощью полиномов соответствующей степени. При этом степень полинома у той компоненты вектор-функции, которая соответствует её направлению, берётся обычно на единицу меньше.

Покажем, как могут быть построены базисные вектор-функции высокого порядка на прямоугольном элементе. При этом мы будем использовать те принципы, по которым строились иерархические базисные функции в узловом МКЭ.

Вид иерархических базисных вектор-функций высокого порядка приведём для шаблонного элемента $\Omega^E = [-1,1]\times[-1,1]$.

На шаблонном элементе Ω^E базисные вектор-функции первого порядка (см. (4.33)) имеют следующий вид:

$$\hat{\vec{\varphi}}_1 = \begin{pmatrix} 0 \\ \dfrac{1-\xi}{2} \end{pmatrix}, \quad \hat{\vec{\varphi}}_2 = \begin{pmatrix} 0 \\ \dfrac{1+\xi}{2} \end{pmatrix}, \quad \hat{\vec{\varphi}}_3 = \begin{pmatrix} \dfrac{1-\eta}{2} \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_4 = \begin{pmatrix} \dfrac{1+\eta}{2} \\ 0 \end{pmatrix}. \qquad (4.34)$$

Для построения прямоугольного векторного элемента второго порядка необходимо добавить к локальным базисным вектор-функциям (4.34) ещё 8 вектор-функций вида

$$\hat{\vec{\varphi}}_5 = \begin{pmatrix} 0 \\ \dfrac{1-\xi}{2}\eta \end{pmatrix}, \quad \hat{\vec{\varphi}}_6 = \begin{pmatrix} 0 \\ \dfrac{1+\xi}{2}\eta \end{pmatrix}, \quad \hat{\vec{\varphi}}_7 = \begin{pmatrix} \dfrac{1-\eta}{2}\xi \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_8 = \begin{pmatrix} \dfrac{1+\eta}{2}\xi \\ 0 \end{pmatrix},$$

$$\hat{\vec{\varphi}}_9 = \begin{pmatrix} 0 \\ 1-\xi^2 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{10} = \begin{pmatrix} 0 \\ \left(1-\xi^2\right)\eta \end{pmatrix}, \quad \hat{\vec{\varphi}}_{11} = \begin{pmatrix} 1-\eta^2 \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{12} = \begin{pmatrix} \left(1-\eta^2\right)\xi \\ 0 \end{pmatrix}. \qquad (4.35)$$

Очевидно, что квадратичные базисные вектор-функции $\hat{\vec{\varphi}}_9 \div \hat{\vec{\varphi}}_{12}$ имеют нулевые касательные составляющие на всех рёбрах конечного элемента, и поэтому их имеет смысл ассоциировать с самим конечным элементом (т.е. к ним уже совершенно не подходит термин «edge»). Заметим также, что квадратичные глобальные

базисные вектор-функции, построенные из вектор-функций вида $\hat{\vec{\varphi}}_9 \div \hat{\vec{\varphi}}_{12}$, являются ненулевыми *только на одном конечном элементе*, т.е. они равны одной из локальных базисных вектор-функций внутри конечного элемента и тождественно равны нулю вне его. Это же справедливо для всех базисных вектор-функций высоких порядков, ассоциированных не с рёбрами, а с самим конечным элементом.

Приведём общий вид шаблонных иерархических базисных вектор-функций прямоугольного конечного элемента p-го порядка. В этот базис входят все $2p(p-1)$ базисных вектор-функций элемента $(p-1)$-го порядка и ещё $4p$ вектор-функций (для $p \geq 3$):

$$\hat{\vec{\varphi}}_{2p(p-1)+1} = \begin{pmatrix} 0 \\ \left(\frac{1-\xi}{2}\right)\eta^{p-3}\left(1-\eta^2\right) \end{pmatrix}, \quad \hat{\vec{\varphi}}_{2p(p-1)+2} = \begin{pmatrix} 0 \\ \left(\frac{1+\xi}{2}\right)\eta^{p-3}\left(1-\eta^2\right) \end{pmatrix},$$

$$\hat{\vec{\varphi}}_{2p(p-1)+3} = \begin{pmatrix} \left(\frac{1-\eta}{2}\right)\xi^{p-3}\left(1-\xi^2\right) \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{2p(p-1)+4} = \begin{pmatrix} \left(\frac{1+\eta}{2}\right)\xi^{p-3}\left(1-\xi^2\right) \\ 0 \end{pmatrix},$$

$$\hat{\vec{\varphi}}_{2p(p-1)+4+1} = \begin{pmatrix} 0 \\ \xi^{p-2}\left(1-\xi^2\right) \end{pmatrix}, \qquad \hat{\vec{\varphi}}_{2p(p-1)+4+2} = \begin{pmatrix} 0 \\ \eta\xi^{p-2}\left(1-\xi^2\right) \end{pmatrix},$$

$$\hat{\vec{\varphi}}_{2p(p-1)+4+3} = \begin{pmatrix} 0 \\ \left(1-\eta^2\right)\xi^{p-2}\left(1-\xi^2\right) \end{pmatrix}, \qquad \ldots,$$

$$\hat{\vec{\varphi}}_{2p(p-1)+4+p} = \begin{pmatrix} 0 \\ \eta^{p-3}\left(1-\eta^2\right)\xi^{p-2}\left(1-\xi^2\right) \end{pmatrix}, \hat{\vec{\varphi}}_{2p(p-1)+4+p+1} = \begin{pmatrix} 0 \\ \eta^{p-3}\left(1-\eta^2\right)\left(1-\xi^2\right) \end{pmatrix},$$

$$\ldots, \qquad \hat{\vec{\varphi}}_{2p(p-1)+4+2p-2} = \begin{pmatrix} 0 \\ \eta^{p-3}\left(1-\eta^2\right)\xi^{p-3}\left(1-\xi^2\right) \end{pmatrix},$$

$$\hat{\vec{\varphi}}_{2p(p-1)+2+2p+1} = \begin{pmatrix} \eta^{p-2}\left(1-\eta^2\right) \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{2p(p-1)+2+2p+2} = \begin{pmatrix} \xi\eta^{p-2}\left(1-\eta^2\right) \\ 0 \end{pmatrix},$$

$$\hat{\vec{\varphi}}_{2p(p-1)+2+2p+3} = \begin{pmatrix} \left(1-\xi^2\right)\eta^{p-2}\left(1-\eta^2\right) \\ 0 \end{pmatrix}, \quad \ldots,$$

$$\hat{\vec{\varphi}}_{2p(p-1)+2+2p+p} = \begin{pmatrix} \xi^{p-3}\left(1-\xi^2\right)\eta^{p-2}\left(1-\eta^2\right) \\ 0 \end{pmatrix}, \hat{\vec{\varphi}}_{2p(p-1)+2+3p+1} = \begin{pmatrix} \xi^{p-3}\left(1-\xi^2\right)\left(1-\eta^2\right) \\ 0 \end{pmatrix},$$

$$\ldots, \qquad \hat{\vec{\varphi}}_{2p(p-1)+4p} = \begin{pmatrix} \xi^{p-3}\left(1-\xi^2\right)\eta^{p-3}\left(1-\eta^2\right) \\ 0 \end{pmatrix}, \tag{4.36}$$

из которых первые четыре вектор-функции ассоциируются с рёбрами конечного элемента, а остальные $4p-4$ вектор-функции – с самим элементом.

Очевидно, что базисные вектор-функции $\hat{\vec{\psi}}_i\left(x,y\right)$ произвольного конечного элемента $\Omega_{rs} = \left[x_r, x_{r+1}\right] \times \left[y_s, y_{s+1}\right]$ могут быть легко получены из вектор-функций $\hat{\vec{\varphi}}_i\left(\xi,\eta\right)$ заменой переменных

$$\xi = 2\left(x - x_{r+1/2}\right)\Big/ h_x, \quad h_x = x_{r+1} - x_r, \quad x_{r+1/2} = \left(x_r + x_{r+1}\right)\Big/ 2,$$

$$\eta = 2\left(y - y_{s+1/2}\right)\Big/ h_y, \quad h_y = y_{s+1} - y_s, \quad y_{s+1/2} = \left(y_s + y_{s+1}\right)\Big/ 2,$$

т.е.

$$\hat{\vec{\psi}}_i\left(x,y\right) = \hat{\vec{\varphi}}_i\left(\frac{2\left(x - x_{r+1/2}\right)}{h_x}, \frac{2\left(y - y_{s+1/2}\right)}{h_y}\right).$$

4.5.3. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ НА ТРЕУГОЛЬНИКЕ

Рассмотрим, как можно построить базисные вектор-функции из $\mathbb{H}^{\mathrm{rot}}$ на треугольных конечных элементах. Для этого удобно использовать векторы $\operatorname{grad}\boldsymbol{\mathcal{L}}_i$, где $\boldsymbol{\mathcal{L}}_i$ – это $\boldsymbol{\mathcal{L}}$-координаты треугольника Ω_m.

Действительно, вектор-функция $\operatorname{grad}\boldsymbol{\mathcal{L}}_1$ имеет нулевую касательную составляющую на ребре с номером 2, соединяющем вершины с номерами 1 и 3 (рис. 39), поскольку линии равного уровня $\boldsymbol{\mathcal{L}}_1$ параллельны этому ребру. Тогда вектор-функция $\boldsymbol{\mathcal{L}}_2 \operatorname{grad}\boldsymbol{\mathcal{L}}_1$ будет иметь нулевые касательные составляющие уже на двух рёбрах с номерами 2 и 3 (поскольку функция $\boldsymbol{\mathcal{L}}_2$ на ребре 3 равна нулю). Таким образом, вектор-функция $\boldsymbol{\mathcal{L}}_2 \operatorname{grad}\boldsymbol{\mathcal{L}}_1$ имеет ненулевую касательную составляющую только на ребре 1. При этом значение касательной составляющей вектор-функции $\boldsymbol{\mathcal{L}}_2 \operatorname{grad}\boldsymbol{\mathcal{L}}_1$ на ребре 1 меняется линейно от нуля в точке 1 до значения $\left(\operatorname{grad}\boldsymbol{\mathcal{L}}_1\right)\cdot\vec{\mathbf{v}}_1$ (где $\vec{\mathbf{v}}_1$ – единичный вектор, направленный вдоль ребра 1) в точке 2. Покажем, что значение $\left(\operatorname{grad}\boldsymbol{\mathcal{L}}_1\right)\cdot\vec{\mathbf{v}}_1$ в точке 1 равно

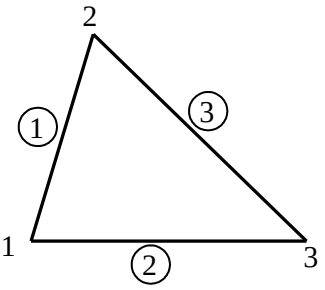


Рис. 39. Локальная нумерация узлов и рёбер на треугольном конечном элементе Ω_m

$1/l_1$ (или $-1/l_1$ в зависимости от ориентации вектора $\vec{\mathbf{v}}_1$), где $l_1 = \sqrt{\left(\hat{x}_1 - \hat{x}_2\right)^2 + \left(\hat{y}_1 - \hat{y}_2\right)^2}$ – длина ребра 1.

Заметим, что $\boldsymbol{\mathcal{L}}$-координаты являются линейными функциями декартовых координат:

$$\boldsymbol{\mathcal{L}}_i = \alpha_0^i + \alpha_1^i x + \alpha_2^i y, \;\; i = 1, \;\; 2, \;\; 3, \tag{4.37}$$

где

$$\begin{pmatrix} \alpha_0^1 & \alpha_1^1 & \alpha_2^1 \\ \alpha_0^2 & \alpha_1^2 & \alpha_2^2 \\ \alpha_0^3 & \alpha_1^3 & \alpha_2^3 \end{pmatrix} = \boldsymbol{\mathcal{A}} = \mathbf{D}^{-1} = \begin{pmatrix} 1 & 1 & 1 \\ \hat{x}_1 & \hat{x}_2 & \hat{x}_3 \\ \hat{y}_1 & \hat{y}_2 & \hat{y}_3 \end{pmatrix}^{-1}. \tag{4.38}$$

Поэтому $\operatorname{grad}\boldsymbol{\mathcal{L}}_1 = \left(\alpha_1^1, \alpha_2^1\right)$, и, учитывая, что

$$\vec{\mathbf{v}}_1 = \left(\hat{x}_1 - \hat{x}_2, \hat{y}_1 - \hat{y}_2\right)/l_1, \tag{4.39}$$

получаем

$$\left(\operatorname{grad}\boldsymbol{\mathcal{L}}_1\right) \cdot \vec{\mathbf{v}}_1 = \left(\alpha_1^1\left(\hat{x}_1 - \hat{x}_2\right) + \alpha_2^1\left(\hat{y}_1 - \hat{y}_2\right)\right)/l_1. \tag{4.40}$$

При этом

$$\begin{pmatrix} \alpha_0^1 & \alpha_1^1 & \alpha_2^1 \\ \alpha_0^2 & \alpha_1^2 & \alpha_2^2 \\ \alpha_0^3 & \alpha_1^3 & \alpha_2^3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ \hat{x}_1 & \hat{x}_2 & \hat{x}_3 \\ \hat{y}_1 & \hat{y}_2 & \hat{y}_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

т.е. в результате умножения первой строки матрицы $\boldsymbol{\mathcal{A}}$ на первый и второй столбцы матрицы $\mathbf{D}$ получаем

$$\alpha_0^1 + \alpha_1^1 \hat{x}_1 + \alpha_2^1 \hat{y}_1 = 1, \tag{4.41}$$

$$\alpha_0^1 + \alpha_1^1 \hat{x}_2 + \alpha_2^1 \hat{y}_2 = 0. \tag{4.42}$$

Вычитая (4.42) из (4.41), получаем

$$\alpha_1^1\left(\hat{x}_1 - \hat{x}_2\right) + \alpha_2^1\left(\hat{y}_1 - \hat{y}_2\right) = 1,$$

т.е. с учётом (4.40)

$$\left(\operatorname{grad}\boldsymbol{\mathcal{L}}_1\right) \cdot \vec{\mathbf{v}}_1 = 1/l_1. \tag{4.43}$$

Абсолютно аналогично можно доказать, что для вектора $\vec{\mathbf{v}}_1$, направленного вдоль первого ребра треугольника Ω_m и определяемого соотношением (4.39), будет справедливо равенство

$$\left(\operatorname{grad}\boldsymbol{\mathcal{L}}_2\right)\cdot\vec{\mathbf{v}}_1 = -1/l_1\,,$$

и поэтому касательная составляющая вектор-функции $\boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_2$ на ребре 1 будет меняться линейно от значения $-1/l_1$ в точке 1 до нуля в точке 2. При этом вектор-функция $\boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_2$ так же, как и вектор-функция $\boldsymbol{\mathcal{L}}_2\operatorname{grad}\boldsymbol{\mathcal{L}}_1$, имеет нулевые касательные составляющие на других рёбрах треугольника, поскольку $\operatorname{grad}\boldsymbol{\mathcal{L}}_2$ перпендикулярен к ребру с номером 2, а функция $\boldsymbol{\mathcal{L}}_1$ равна нулю на ребре с номером 3 (см. рис. 39).

Поэтому вектор-функция

$$\hat{\vec{\psi}}_1 = \boldsymbol{\mathcal{L}}_2\operatorname{grad}\boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_2 \qquad (4.44)$$

имеет *постоянное, равное* $1/l_1$, значение касательной составляющей на ребре 1 и *нулевые* значения касательных составляющих на ребрах 2 и 3.

На самом деле, доказать то, что вектор-функция $\hat{\vec{\psi}}_1$ имеет постоянную касательную на ребре 1, можно и другим, очень простым способом. Для этого достаточно выразить из тождества $\boldsymbol{\mathcal{L}}_1+\boldsymbol{\mathcal{L}}_2+\boldsymbol{\mathcal{L}}_3=1$ функцию $\boldsymbol{\mathcal{L}}_2$:

$$\hat{\vec{\psi}}_1 = \boldsymbol{\mathcal{L}}_2\operatorname{grad}\boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_2 = \boldsymbol{\mathcal{L}}_2\operatorname{grad}\boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1\operatorname{grad}\left(1-\boldsymbol{\mathcal{L}}_1-\boldsymbol{\mathcal{L}}_3\right)$$

и учесть, что на этом ребре $\boldsymbol{\mathcal{L}}_1+\boldsymbol{\mathcal{L}}_2\equiv 1$, т.е.

$$\hat{\vec{\psi}}_1 = \boldsymbol{\mathcal{L}}_2\operatorname{grad}\boldsymbol{\mathcal{L}}_1 + \boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_1 + \boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_3 = \operatorname{grad}\boldsymbol{\mathcal{L}}_1 + \boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_3$$

на ребре 1.

Тогда, учитывая, что функция $\operatorname{grad}\boldsymbol{\mathcal{L}}_3$ перпендикулярна ребру 1 (т.е. $\operatorname{grad}\boldsymbol{\mathcal{L}}_3\cdot\vec{\mathbf{v}}_1=0$, $\vec{\mathbf{v}}_1$ – касательный к ребру 1 единичный вектор, см. (4.39)), касательная к этому ребру составляющая вектор-функции $\hat{\vec{\psi}}_1$ будет равна

$$\hat{\vec{\psi}}_1\cdot\vec{\mathbf{v}}_1 = \operatorname{grad}\boldsymbol{\mathcal{L}}_1\cdot\vec{\mathbf{v}}_1 + \boldsymbol{\mathcal{L}}_1\operatorname{grad}\boldsymbol{\mathcal{L}}_3\cdot\vec{\mathbf{v}}_1 = \operatorname{grad}\boldsymbol{\mathcal{L}}_1\cdot\vec{\mathbf{v}}_1 =$$

$$= \alpha_1^1\frac{\hat{x}_1-\hat{x}_2}{l_1} + \alpha_2^1\frac{\hat{y}_1-\hat{y}_2}{l_1}$$

(сравните с (4.40)).

Но для нас очень важным является и тот факт, что величина $\alpha_1^1\left(\hat{x}_1-\hat{x}_2\right)+\alpha_2^1\left(\hat{y}_1-\hat{y}_2\right)$ *не зависит от координат вершины треугольника, не лежащей на ребре* 1, *с которым ассоциирована вектор-функция* $\hat{\vec{\psi}}_1$, а доказательство этого и составляет основную часть проведённых нами выше рассуждений (см. соотношения (4.41), (4.42)).

Очевидно, что на треугольнике Ω_m можно ввести ещё две локальные базисные вектор-функции, ассоциированные с ребром 2 и ребром 3 соответственно (см. рис. 39):

$$\hat{\vec{\psi}}_2 = \boldsymbol{\mathcal{L}}_3 \operatorname{grad}\boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad}\boldsymbol{\mathcal{L}}_3, \tag{4.45}$$

$$\hat{\vec{\psi}}_3 = \boldsymbol{\mathcal{L}}_3 \operatorname{grad}\boldsymbol{\mathcal{L}}_2 - \boldsymbol{\mathcal{L}}_2 \operatorname{grad}\boldsymbol{\mathcal{L}}_3. \tag{4.46}$$

Точно так же, как и для вектор-функции $\hat{\vec{\psi}}_1$, можно доказать, что вектор-функции $\hat{\vec{\psi}}_2$ и $\hat{\vec{\psi}}_3$ имеют постоянные касательные составляющие на рёбрах, с которыми они ассоциированы, и нулевые касательные составляющие на двух других рёбрах.

Из локальных вектор-функций такого вида можно построить минимально допустимый базис конечномерного подпространства $\mathbb{V}^{\mathrm{rot}}$, определяемого для сетки с треугольными ячейками. Действительно, так как локальные базисные вектор-функции вида (4.44)–(4.46), ассоциированные с ребром Γ, на двух треугольниках Ω_{m_1} и Ω_{m_2}, смежных по этому ребру, имеют касательные составляющие, равные $1/l$ (l – длина Γ), то в качестве глобальной базисной вектор-функции, ассоциированной с ребром Γ, можно взять следующую вектор-функцию. На треугольниках Ω_{m_1} и Ω_{m_2} она должна быть равна локальным базисным вектор-функциям, ассоциированным с ребром Γ, а на всех остальных треугольниках эта глобальная базисная вектор-функция должна быть тождественно равна нулю.

Очевидно, что определённые таким образом глобальные вектор-функции имеют ненулевые (и непрерывные) касательные составляющие только на одном ребре, с которым они ассоциированы, и поэтому они сами и любые их линейные комбинации являются элементами пространства $\mathbb{H}^{\mathrm{rot}}$. Пример одной из таких глобальных базисных вектор-функций (для сетки из треугольных конечных элементов) приведён на рис. 40.

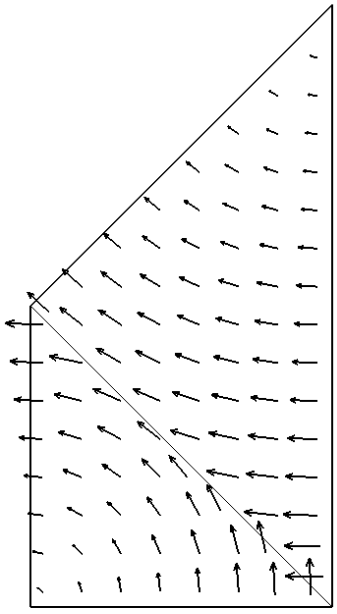


Рис. 40. Вид глобальной базисной вектор-функции на треугольной сетке

Обратим внимание на одну деталь, очень важную при работе с треугольными edge-элементами (это замечание актуально при использовании любых нерегулярных сеток с edge-элементами). Непрерывность касательной составляющей глобальной базисной вектор-функции на ребре, с которым она ассоциирована, оказывается зависящей от *локальной нумерации вершин треугольников*, смежных по этому ребру. Покажем это на примере ситуации, изображённой на рис. 41.

Рассмотрим глобальную базисную вектор-функцию, ассоциированную с ребром, соединяющим вершины с глобальными номерами 2 и 3. На треугольнике Ω_1 (с вершинами, имеющими глобальные номера 1,2,3) эта базисная вектор-функция определяется как

$$\hat{\vec{\psi}}_3^{\Omega_1} = \boldsymbol{\mathcal{L}}_3 \operatorname{grad} \boldsymbol{\mathcal{L}}_2 - \boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_3 ,$$

а на треугольнике Ω_2 (с вершинами, имеющими глобальные номера 3,2,4) – как

$$\hat{\vec{\psi}}_1^{\Omega_2} = \boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad} \boldsymbol{\mathcal{L}}_2 .$$

Но локальная вектор-функция $\hat{\vec{\psi}}_3^{\Omega_1}$ на ребре {2,3} имеет касательную составляющую, направленную от точки с глобальным номером 3 к точке с глобальным номером 2, а локальная вектор-функция $\hat{\vec{\psi}}_1^{\Omega_2}$ на этом же ребре – касательную составляющую, направленную от точки с глобальным номером 2 к точке с глобальным номером 3. Вид глобальной вектор-функции, полученной в результате «формальной» сшивки этих двух локальных вектор-функций, приведён на рис. 41,*б*. Очевидно, что полученная глобальная вектор-функция имеет *разрывную* касательную составляющую на ребре, с которым она ассоциирована.

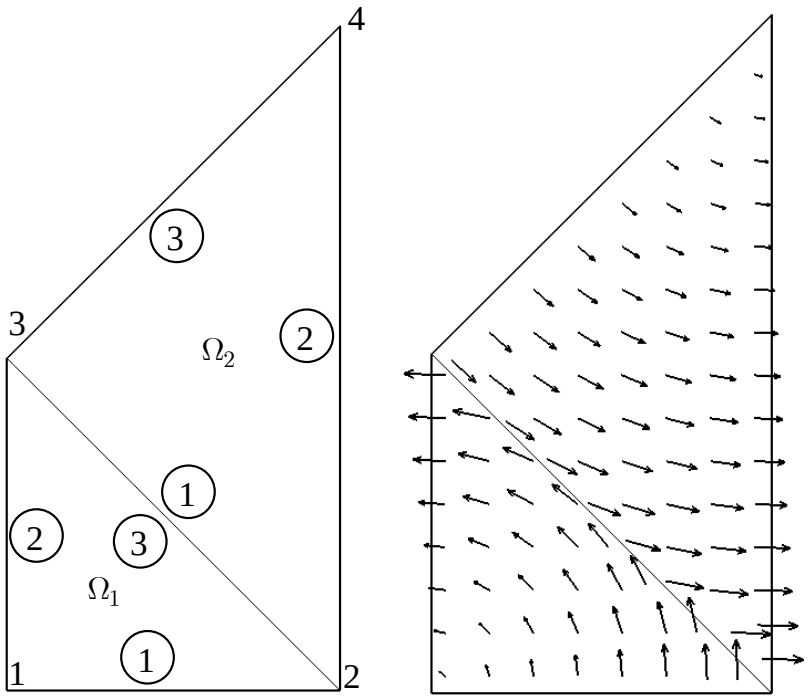


Рис. 41. Глобальная и локальная нумерация базисных вектор-функций на треугольнике (*а*) и вид глобальной базисной вектор-функции, полученной «формальной» сшивкой локальных базисных вектор-функций $\hat{\vec{\psi}}_3^{\Omega_1}$ и $\hat{\vec{\psi}}_1^{\Omega_2}$ (*б*)

Решить эту проблему можно путем специального упорядочения локальных номеров вершин на конечном элементе. Так, если на треугольных конечных элементах локальную нумерацию вершин проводить строго по возрастанию их глобальных номеров, то локальные базисные вектор-функции, примыкающие к одному и тому же ребру, всегда будут направлены в одну и ту же сторону, и касательные составляющие получаемых глобальных вектор-функций будут непрерывны.

Иногда вместо базисных вектор-функций (4.44)–(4.46) используют нормированные вектор-функции

$$\hat{\vec{\psi}}_1 = l_1\left(\boldsymbol{\mathcal{L}}_2 \operatorname{grad}\boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad}\boldsymbol{\mathcal{L}}_2\right), \tag{4.47}$$

$$\hat{\vec{\psi}}_2 = l_2\left(\boldsymbol{\mathcal{L}}_3 \operatorname{grad}\boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad}\boldsymbol{\mathcal{L}}_3\right), \tag{4.48}$$

$$\hat{\vec{\psi}}_3 = l_3\left(\boldsymbol{\mathcal{L}}_3 \operatorname{grad}\boldsymbol{\mathcal{L}}_2 - \boldsymbol{\mathcal{L}}_2 \operatorname{grad}\boldsymbol{\mathcal{L}}_3\right), \tag{4.49}$$

где l_1, l_2 и l_3 – длины соответствующих рёбер. Эти вектор-функции удобней тем, что в случае их использования не возникает никаких проблем при решении задачи *на смешанной сетке*, состоящей из прямоугольных и треугольных ячеек. Дейст-

вительно, касательная составляющая глобальной базисной вектор-функции, ассоциированной с ребром, которое является общим для прямоугольника и треугольника, будет равна единице как на прямоугольнике, так и на треугольнике. Это и обеспечит непрерывность такой глобальной вектор-функции (полученной формальной сшивкой локальных базисных вектор-функций прямоугольника и треугольника) на ребре, с которым она ассоциирована, и тем самым обеспечит принадлежность её пространству $\mathbb{H}^{\mathrm{rot}}$.

Следует отметить ещё одну особенность базисов, построенных с помощью соотношений (4.44)–(4.46) (или, что практически то же самое, (4.47)–(4.49)). На этих базисах плохо представимыми являются даже вектор-функции, линейно изменяющиеся по одной из пространственных координат. Поэтому могут возникать серьёзные трудности при решении задач со скин-эффектом, когда необходимо аппроксимировать функцию, резко меняющуюся вблизи некоторой границы. В этом случае обычно используют конечные элементы, вытянутые в направлении, касательном к границе, и очень сжатые в направлении нормали к этой границе. При использовании же вектор-функций (4.44)–(4.46) (и (4.47)–(4.49)) на вытянутых треугольниках решения задач со скин-эффектом обычно аппроксимируются с очень большой погрешностью.

В принципе, на треугольных конечных элементах совсем несложно построить такой базис, с помощью которого можно точно представить любую линейную вектор-функцию. Однако в этом случае нужно использовать шесть локальных базисных вектор-функций. Например, это могут быть вектор-функции следующего вида:

$$\hat{\vec{\psi}}_1 = \boldsymbol{\pounds}_2 \operatorname{grad} \boldsymbol{\pounds}_1, \quad \hat{\vec{\psi}}_2 = \boldsymbol{\pounds}_1 \operatorname{grad} \boldsymbol{\pounds}_2, \quad \hat{\vec{\psi}}_3 = \boldsymbol{\pounds}_3 \operatorname{grad} \boldsymbol{\pounds}_1,$$
$$\hat{\vec{\psi}}_4 = \boldsymbol{\pounds}_1 \operatorname{grad} \boldsymbol{\pounds}_3, \quad \hat{\vec{\psi}}_5 = \boldsymbol{\pounds}_3 \operatorname{grad} \boldsymbol{\pounds}_2, \quad \hat{\vec{\psi}}_6 = \boldsymbol{\pounds}_2 \operatorname{grad} \boldsymbol{\pounds}_3, \qquad (4.50)$$

где вектор-функции $\hat{\vec{\psi}}_1$ и $\hat{\vec{\psi}}_2$ имеют ненулевые касательные только на ребре 1 (см. рис. 39) и поэтому ассоциируются с этим ребром, вектор-функции $\hat{\vec{\psi}}_3$ и $\hat{\vec{\psi}}_4$ имеют ненулевые касательные только на ребре 2 (и ассоциированы с ребром 2), а вектор-функции $\hat{\vec{\psi}}_5$ и $\hat{\vec{\psi}}_6$ – только на ребре 3 (и ассоциированы с ним). При этом касательные составляющие этих вектор-функций меняются от значения $1/l$ (или $-1/l$ в зависимости от ориентации ребра, l – длина ребра) в одной вершине ребра, с которым они ассоциированы, до нуля в другой вершине этого ребра.

Глобальные базисные вектор-функции получаются (как и в случае базиса (4.44)–(4.46)) сшивкой двух локальных базисных вектор-функций, *ассоциированных с одним ребром и отличных от нуля в одной и той же вершине этого ребра*. Так, например, вектор-функция $\hat{\vec{\psi}}_1$ будет сшиваться с такой функцией смежного с Ω_m (см. рис. 39) треугольника, которая ассоциирована с ребром 1 и отлична от нуля в той вершине, которая на Ω_m имеет локальный номер 2. Поэтому каждую из таких глобальных вектор-функций можно считать *ассоциированной одновременно с одним ребром и одной вершиной*.

Эквивалентный (4.50) базис может быть построен добавкой на каждом конечном элементе к базису (4.44)–(4.46) ещё трёх линейных базисных вектор-функций (т.е. по иерархическому принципу – новый базис получается *добавкой* базисных функций к набору функций старого базиса). Можно, например, к вектор-функциям (4.44)–(4.46) добавить первую, третью и пятую вектор-функции из набора базисных вектор-функций (4.50). А можно добавить вторую, четвёртую и шестую. Но обычно к вектор-функциям (4.44)–(4.46) добавляются вектор-функции следующего вида:

$$\hat{\vec{\psi}}_4 = \boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_1 + \boldsymbol{\mathcal{L}}_1 \operatorname{grad} \boldsymbol{\mathcal{L}}_2, \tag{4.51}$$

$$\hat{\vec{\psi}}_5 = \boldsymbol{\mathcal{L}}_3 \operatorname{grad} \boldsymbol{\mathcal{L}}_1 + \boldsymbol{\mathcal{L}}_1 \operatorname{grad} \boldsymbol{\mathcal{L}}_3, \tag{4.52}$$

$$\hat{\vec{\psi}}_6 = \boldsymbol{\mathcal{L}}_3 \operatorname{grad} \boldsymbol{\mathcal{L}}_2 + \boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_3. \tag{4.53}$$

Каждая из вектор-функций (4.51)–(4.53) также имеет ненулевую касательную только на одном ребре (с которым она и ассоциируется), и эта касательная изменяется вдоль соответствующего ребра линейно от значения -1 в одной его вершине до значения 1 в другой.

Базисные вектор-функции треугольных конечных элементов высокого порядка можно построить, например, в виде

$$\left(\boldsymbol{\mathcal{L}}_1\right)^{k_1} \left(1-\boldsymbol{\mathcal{L}}_1\right)^{k_2} \left(\boldsymbol{\mathcal{L}}_2\right)^{k_3} \left(1-\boldsymbol{\mathcal{L}}_2\right)^{k_4} \left(\boldsymbol{\mathcal{L}}_3\right)^{k_5} \left(1-\boldsymbol{\mathcal{L}}_3\right)^{k_6} \operatorname{grad} \boldsymbol{\mathcal{L}}_i. \tag{4.54}$$

При построении таких базисов необходимо следить, чтобы все базисные функции были линейно независимыми и давали любую полиномиальную вектор-функцию нужного порядка. Однако на симплексах элементы высокого порядка используются довольно редко. Чаще всего, если для повышения точности расчёта выбирается путь, связанный с повышением порядка элемента, в качестве конеч-

ных элементов используются либо прямоугольники или параллелепипеды, либо четырёхугольники или шестигранники (с четырёхугольными гранями).

Сделаем ещё одно довольно важное замечание относительно использования edge-элементов на ячейках различных типов. При использовании базисов вида (4.50) или (4.44)–(4.46) и (4.51)–(4.53) число глобальных базисных вектор-функций ровно вдвое больше числа рёбер сетки. Кроме того, возрастает и количество связей между глобальными вектор-функциями, что приводит к увеличению заполненности матрицы конечноэлементной СЛАУ почти вчетверо. Поэтому в ситуациях, когда при использовании минимального базиса (4.44)–(4.46) не удаётся получить решение задачи (например, в случае скин-эффекта и необходимости использования вытянутых ячеек), треугольные edge-элементы являются далеко не самыми эффективными. В этом случае лучше использовать либо прямоугольные edge-элементы (если геометрия задачи позволяет это делать), либо четырёхугольные edge-элементы, которые, пожалуй, наиболее эффективны для решения двумерных задач в областях со сложной геометрией.

4.5.4. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ НА ЧЕТЫРЁХУГОЛЬНИКЕ

В векторном МКЭ, как и в скалярном, четырёхугольный конечный элемент сочетает в себе основные достоинства прямоугольных и треугольных элементов: высокое качество аппроксимации (как у прямоугольных элементов) и возможность работы со сложной геометрией (как у треугольных элементов).

Основной принцип построения базисных вектор-функций на четырёхугольнике остаётся тем же (как и на прямоугольнике или треугольнике): каждая базисная вектор-функция должна иметь отличную от нуля касательную составляющую не более чем на одном ребре. Наиболее удобно строить такие базисные вектор-функции через отображение на четырёхугольник базисных вектор-функций шаблонного элемента в виде квадрата $\Omega^E = [0,1]\times[0,1]$ или $\Omega^E = [-1,1]\times[-1,1]$ (при работе с лагранжевыми базисами обычно используется квадрат $\Omega^E = [0,1]\times[0,1]$, а при работе с иерархическими базисами чаще используют квадрат $\Omega^E = [-1,1]\times[-1,1]$, поскольку одномерные иерархические базисные функции удобней задавать на отрезке $[-1,1]$). Мы будем пользоваться шаблонным элементом $\Omega^E = [-1,1]\times[-1,1]$, полагая, что элементы высоких порядков будут строиться с помощью рассмотренных в разделе 4.5.2 базисных вектор-функций вида (4.36).

Рассмотрим некоторый (произвольный) четырёхугольный конечный элемент Ω_m с вершинами $\left(\hat{x}_i,\hat{y}_i\right), i=1...4$. Будем считать, что его вершины и рёбра пронумерованы так, как это показано на рис. 42 (причём в прямоугольничках стоят локальные номера вершин, а в кружочках – локальные номера рёбер).

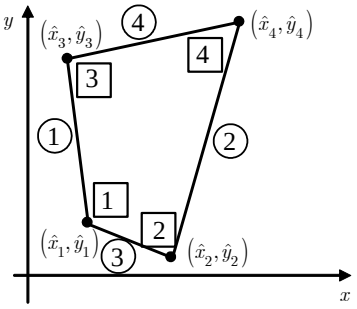


Рис. 42. Расположение узлов на четырёхугольном конечном элементе

Прежде чем определить базисные вектор-функции на Ω_m, установим между точками $\left(x,y\right)$ четырёхугольника Ω_m и точками $\left(\xi,\eta\right)$ шаблонного элемента $\Omega^E=\left[-1,1\right]\times\left[-1,1\right]$ взаимно однозначное соответствие с помощью формул

$$\begin{aligned} x &= \hat{x}_1\hat{\varphi}_1\left(\xi,\eta\right)+\hat{x}_2\hat{\varphi}_2\left(\xi,\eta\right)+\hat{x}_3\hat{\varphi}_3\left(\xi,\eta\right)+\hat{x}_4\hat{\varphi}_4\left(\xi,\eta\right),\\ y &= \hat{y}_1\hat{\varphi}_1\left(\xi,\eta\right)+\hat{y}_2\hat{\varphi}_2\left(\xi,\eta\right)+\hat{y}_3\hat{\varphi}_3\left(\xi,\eta\right)+\hat{y}_4\hat{\varphi}_4\left(\xi,\eta\right), \end{aligned} \qquad (4.55)$$

где $\hat{\varphi}_i$ – это стандартные узловые билинейные базисные функции:

$$\hat{\varphi}_1=\frac{\left(1-\xi\right)}{2}\frac{\left(1-\eta\right)}{2},\ \hat{\varphi}_2=\frac{\left(1+\xi\right)}{2}\frac{\left(1-\eta\right)}{2},\ \hat{\varphi}_3=\frac{\left(1-\xi\right)}{2}\frac{\left(1+\eta\right)}{2},\ \hat{\varphi}_4=\frac{\left(1+\xi\right)}{2}\frac{\left(1+\eta\right)}{2}.$$

Так как определяемые нами на Ω_m базисные функции являются векторами, то чтобы задать на Ω_m базисные вектор-функции через стандартные базисные вектор-функции шаблонного конечного элемента Ω^E:

$$\hat{\vec{\varphi}}_1=\begin{pmatrix}0\\ \dfrac{1-\xi}{2}\end{pmatrix},\quad \hat{\vec{\varphi}}_2=\begin{pmatrix}0\\ \dfrac{1+\xi}{2}\end{pmatrix},\quad \hat{\vec{\varphi}}_3=\begin{pmatrix}\dfrac{1-\eta}{2}\\ 0\end{pmatrix},\quad \hat{\vec{\varphi}}_4=\begin{pmatrix}\dfrac{1+\eta}{2}\\ 0\end{pmatrix} \qquad (4.56)$$

(сравните с (4.33)), надо определить, как векторы, заданные на Ω^E (в координатах $\left\{\xi,\eta\right\}$), отображаются в векторы на Ω_m (в координатах $\left\{x,y\right\}$).

Например, отображение вектора $\vec{\mathbf{w}}$, заданного в точке $\left(\xi_p,\eta_p\right)$ квадрата Ω^E (т.е. точка $\left(\xi_p,\eta_p\right)$ – начало вектора $\vec{\mathbf{w}}$), в вектор $\vec{\mathbf{v}}$ с началом в точке $\left(x_p,y_p\right)$ (точка $\left(x_p,y_p\right)$ из Ω_m является отображением точки $\left(\xi_p,\eta_p\right)$ из Ω^E) можно определить с помощью соотношения

$$\vec{\mathbf{v}}=\mathbf{J}^{\mathrm{T}}\left(\xi_p,\eta_p\right)\vec{\mathbf{w}}, \qquad (4.57)$$

где $\mathbf{J}\left(\xi,\eta\right)$ – функциональная матрица отображения (4.55), т.е.

$$\mathbf{J}=\begin{pmatrix}\dfrac{\partial x}{\partial\xi} & \dfrac{\partial y}{\partial\xi}\\ \dfrac{\partial x}{\partial\eta} & \dfrac{\partial y}{\partial\eta}\end{pmatrix}=$$

$$=\begin{pmatrix}\dfrac{(1-\eta)}{4}\left(\hat{x}_2-\hat{x}_1\right)+\dfrac{(1+\eta)}{4}\left(\hat{x}_4-\hat{x}_3\right) & \dfrac{(1-\eta)}{4}\left(\hat{y}_2-\hat{y}_1\right)+\dfrac{(1+\eta)}{4}\left(\hat{y}_4-\hat{y}_3\right)\\ \dfrac{(1-\xi)}{4}\left(\hat{x}_3-\hat{x}_1\right)+\dfrac{(1+\xi)}{4}\left(\hat{x}_4-\hat{x}_2\right) & \dfrac{(1-\xi)}{4}\left(\hat{y}_3-\hat{y}_1\right)+\dfrac{(1+\xi)}{4}\left(\hat{y}_4-\hat{y}_2\right)\end{pmatrix}. \qquad (4.58)$$

Тогда любой вектор, касательный к стороне $\xi=-1$ квадрата Ω^E, будет отображаться в вектор, касательный к ребру 1 четырёхугольника Ω_m (см. рис. 42), любой вектор, касательный к стороне $\xi=1$ квадрата Ω^E, – в вектор, касательный к ребру 2 четырёхугольника Ω_m, любой вектор, касательный к стороне $\eta=-1$ квадрата Ω^E, – в вектор, касательный к ребру 3 четырёхугольника Ω_m, а любой вектор, касательный к стороне $\eta=1$ квадрата Ω^E, – в вектор, касательный к ребру 4 четырёхугольника Ω_m.

Однако если базисные вектор-функции $\hat{\vec{\psi}}_i\left(x,y\right)$ конечного элемента Ω_m определить как

$$\hat{\vec{\psi}}_i\left(x,y\right)=\mathbf{J}^{\mathrm{T}}\left(\xi\left(x,y\right),\eta\left(x,y\right)\right)\hat{\vec{\varphi}}_i\left(\xi\left(x,y\right),\eta\left(x,y\right)\right), \qquad (4.59)$$

где $\hat{\vec{\varphi}}_i\left(\xi,\eta\right)$ – базисные вектор-функции, заданные на шаблонном элементе Ω^E (см. (4.56)), а $\xi\left(x,y\right)$ и $\eta\left(x,y\right)$ – функции, определяющие обратное к (4.55) отображение, то эти вектор-функции могут иметь ненулевые касательные не на одном ребре, с которым они ассоциированы, а ещё на двух смежных с ним рёбрах. Действительно, вектор-функция $\hat{\vec{\psi}}_1\left(x,y\right)$ (полученная из вектор-функции $\hat{\vec{\varphi}}_1\left(\xi,\eta\right)$ по формуле (4.59)) в первом узле $\left(\hat{x}_1,\hat{y}_1\right)$ четырёхугольника Ω_m (см. рис. 42) будет направлена вдоль ребра 1 и поэтому будет иметь ненулевую касательную к ребру 3 (если ребро 3 не ортогонально ребру 1). Аналогично, $\hat{\vec{\psi}}_1\left(x,y\right)$ в узле $\left(\hat{x}_3,\hat{y}_3\right)$ будет иметь ненулевую касательную к ребру 4 (если ребро 4 не ортогонально ребру 1). Поведение такой вектор-функции на Ω_m изображено на рис. 43. То же самое можно сказать и о трёх остальных вектор-функциях, определяемых соотношением (4.59).

Чтобы получить нужные нам базисные вектор-функции (с нулевыми касательными на трёх рёбрах Ω_m), воспользуемся следующим фактом. Значение скалярного произведения любой вектор-функции $\hat{\vec{\psi}}\left(x,y\right)$ на произвольный вектор $\vec{\mathbf{v}}$ в любой точке $\left(x_p,y_p\right)$ может быть вычислено как

$$\left(\hat{\vec{\psi}}\left(x_p,y_p\right),\vec{\mathbf{v}}\right)=\left(\hat{\vec{\psi}}\left(x_p,y_p\right),\mathbf{J}^{\mathrm{T}}\vec{\mathbf{w}}\right)=\left(\mathbf{J}\hat{\vec{\psi}}\left(x_p,y_p\right),\vec{\mathbf{w}}\right),$$

где $\vec{\mathbf{w}}$ – вектор, полученный из $\vec{\mathbf{v}}$ отображением, обратным к (4.57), т.е. $\vec{\mathbf{w}}=\mathbf{J}^{-\mathrm{T}}\left(\xi_p,\eta_p\right)\vec{\mathbf{v}}$, а $\left(\xi_p,\eta_p\right)$ – точка в координатах $\left\{\xi,\eta\right\}$, которая при преобразовании координат $\left\{\xi,\eta\right\}$ в координаты $\left\{x,y\right\}$ отображается в точку $\left(x_p,y_p\right)$.

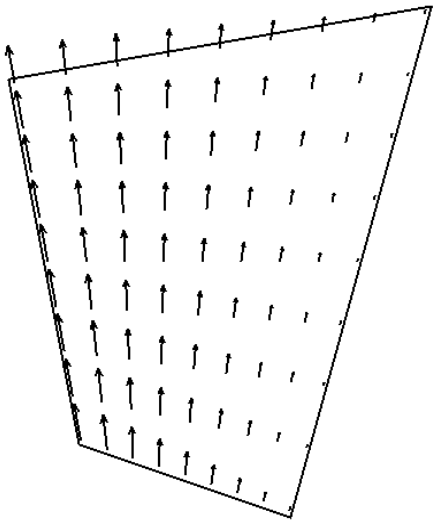


Рис. 43. Вид «неправильной» базисной вектор-функции, определяемой соотношением (4.59)

Таким образом, если в качестве локальных базисных вектор-функций четырёхугольника Ω_m взять вектор-функции

$$\hat{\vec{\psi}}_i\left(x,y\right)=\mathbf{J}^{-1}\left(\xi\left(x,y\right),\eta\left(x,y\right)\right)\hat{\vec{\varphi}}_i\left(\xi\left(x,y\right),\eta\left(x,y\right)\right), \qquad (4.61)$$

то каждая из таких вектор-функций $\hat{\vec{\psi}}_i\left(x,y\right)$ будет иметь ненулевые (и при этом постоянные) касательные только на одном ребре с номером i (с которым она и должна быть ассоциирована).

Действительно, если для вектор-функции $\hat{\vec{\psi}}_1\left(x,y\right)$ в качестве вектора $\vec{\mathbf{v}}$ взять вектор единичной длины, направленный вдоль ребра 1, то с учётом (4.60) и (4.61) в любой точке $\left(x_p,y_p\right)$, лежащей на ребре 1 четырёхугольника Ω_m,

$$\left(\hat{\vec{\psi}}_1\left(x_p,y_p\right),\vec{\mathbf{v}}\right)=\left(\mathbf{J}\hat{\vec{\psi}}_1\left(x_p,y_p\right),\vec{\mathbf{w}}\right)=\left(\hat{\vec{\varphi}}_1\left(\xi_p,\eta_p\right),\vec{\mathbf{w}}\right), \qquad (4.62)$$

где $\vec{\mathbf{w}}$ – вектор, направленный вдоль стороны $\xi=-1$ шаблонного квадрата Ω^E, т.е. $\vec{\mathbf{w}}=\left(0,d\right)^{\mathrm{T}}$. При этом значение d одинаково для всех точек ребра 1, поскольку

$$\vec{\mathbf{v}}=\mathbf{J}^{\mathrm{T}}\left(\xi_p,\eta_p\right)\vec{\mathbf{w}}=\mathbf{J}^{\mathrm{T}}\left(-1,\eta_p\right)\begin{pmatrix}0\\d\end{pmatrix}=$$

$$= \begin{pmatrix} \frac{(1-\eta_p)}{4}(\hat{x}_2-\hat{x}_1)+\frac{(1+\eta_p)}{4}(\hat{x}_4-\hat{x}_3) & \frac{1}{2}(\hat{x}_3-\hat{x}_1) \\ \frac{(1-\eta_p)}{4}(\hat{y}_2-\hat{y}_1)+\frac{(1+\eta_p)}{4}(\hat{y}_4-\hat{y}_3) & \frac{1}{2}(\hat{y}_3-\hat{y}_1) \end{pmatrix} \begin{pmatrix} 0 \\ d \end{pmatrix} = \frac{d}{2} \begin{pmatrix} (\hat{x}_3 - \hat{x}_1) \\ (\hat{y}_3 - \hat{y}_1) \end{pmatrix},$$

т.е. d не зависит от координаты η_p (координата же ξ_p равна -1 на ребре 1). С учётом того, что $\vec{\mathbf{v}}$ – вектор единичной длины,

$$d = \frac{2\left\|\vec{\mathbf{v}}\right\|}{\sqrt{\left(\hat{x}_3 - \hat{x}_1\right)^2 + \left(\hat{y}_3 - \hat{y}_1\right)^2}} = \frac{2}{l_1},$$

где l_1 – длина ребра 1.

Таким образом, значение касательной составляющей вектор-функции $\hat{\vec{\psi}}_1(x,y)$, определяемой формулой (4.61), на ребре 1 четырёхугольника Ω_m может быть вычислено как (см. (4.62))

$$\left(\hat{\vec{\psi}}_1, \vec{\mathbf{v}}\right) = \left(\hat{\vec{\varphi}}_1(-1,\eta), \vec{\mathbf{w}}\right),$$

и с учётом того, что $\hat{\vec{\varphi}}_1(-1,\eta) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ и $\vec{\mathbf{w}} = \begin{pmatrix} 0 \\ d \end{pmatrix}$, получаем

$$\left(\hat{\vec{\psi}}_1, \vec{\mathbf{v}}\right) = d = \frac{2}{l_1},$$

т.е. вектор-функция $\hat{\vec{\psi}}_1(x,y)$ имеет на ребре 1 постоянную, равную $2/l_1$, касательную составляющую.

На остальных рёбрах $\hat{\vec{\psi}}_1(x,y)$ имеет нулевые касательные составляющие. Действительно, если взять в качестве $\vec{\mathbf{v}}$ единичный вектор, направленный вдоль ребра 3, то его прообразом будет вектор $\vec{\mathbf{w}} = \begin{pmatrix} 2/l_3 \\ 0 \end{pmatrix}$, и тогда

$$\left(\hat{\vec{\psi}}_1, \vec{\mathbf{v}}\right) = \left(\hat{\vec{\varphi}}_1(\xi,-1), \vec{\mathbf{w}}\right) = \begin{pmatrix} 0 \\ \frac{1-\xi}{2} \end{pmatrix} \cdot \left(2/l_3, 0\right) = 0.$$

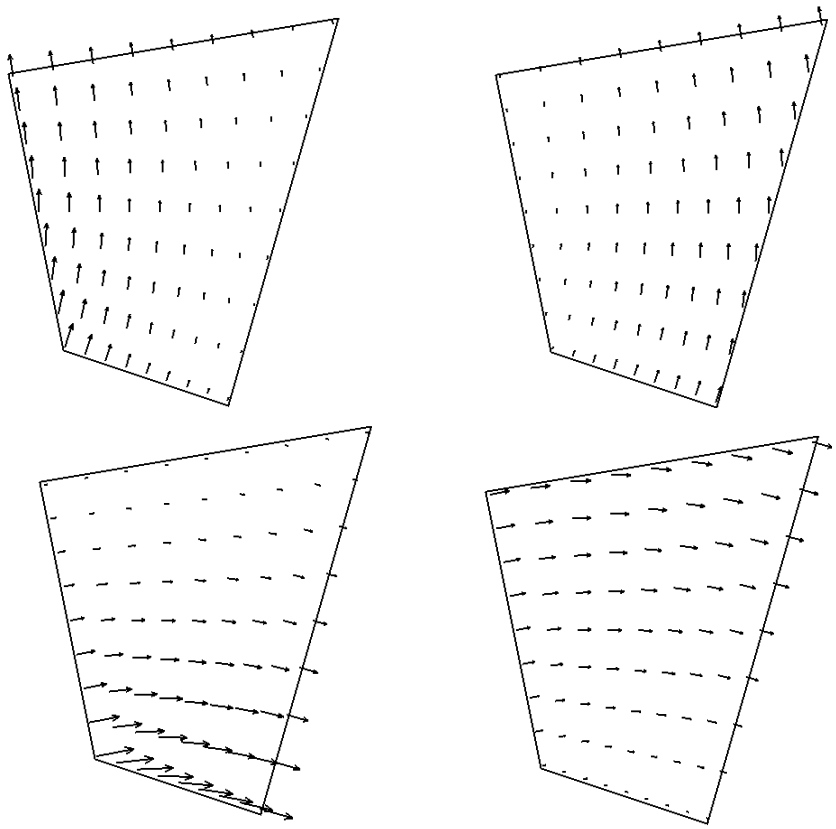


Рис. 44. Поведение локальных базисных вектор-функций, определяемых соотношением (4.61), на четырёхугольнике Ω_m

Аналогично $\left(\hat{\vec{\psi}}_1, \vec{\mathbf{v}}\right) = 0$, если вектор $\vec{\mathbf{v}}$ направлен вдоль ребра 4. На ребре же с номером 2 вектор-функция $\hat{\vec{\psi}}_1$ имеет нулевую касательную, поскольку на нём $\hat{\vec{\psi}}_1 \equiv \vec{\mathbf{0}}$.

Точно так же можно доказать, что и остальные базисные вектор-функции, определяемые соотношением (4.61), имеют ненулевые (и постоянные, равные $2/l_i$) касательные на рёбрах, с которыми они ассоциированы, и нулевые касательные на остальных рёбрах четырёхугольника Ω_m. На рис. 44 показано поведение на Ω_m локальных базисных вектор-функций. На рис. 45 показан вид одной глобальной базисной вектор-функции в расчётной области при использовании в качестве элементов дискретизации четырёхугольников.

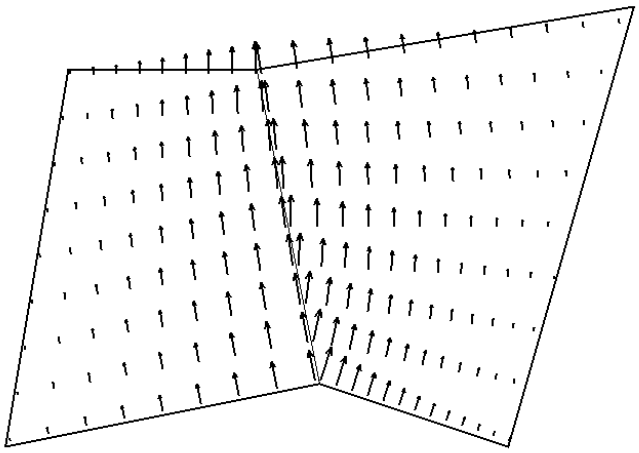


Рис. 45. Поведение одной из глобальных базисных вектор-функций при использовании сетки с четырёхугольными ячейками

Очевидно, что вместо вектор-функций (4.61), у которых касательная составляющая на ребре (с которым она ассоциирована) равна $2/l_i$, можно ввести нормированные базисные вектор-функции с единичными касательными:

$$\hat{\vec{\psi}}_i\left(x,y\right)=\frac{l_i}{2}\mathbf{J}^{-1}\left(\xi\left(x,y\right),\eta\left(x,y\right)\right)\hat{\vec{\varphi}}_i\left(\xi\left(x,y\right),\eta\left(x,y\right)\right), \qquad (4.63)$$

где l_i – длина i-го ребра.

Вектор-функции вида (4.63) имеет смысл использовать, например, в тех случаях, когда в конечноэлементной сетке наряду с четырёхугольниками есть и прямоугольники, и при этом локальные матрицы прямоугольных элементов вычисляются не через шаблонные элементы, а напрямую по функциям, определённым в разделе 4.5.2.

Рассмотрим, как можно ввести базисные вектор-функции *на криволинейном четырёхугольнике*. Задать криволинейный четырёхугольник можно (так же как и четырёхугольник с прямыми сторонами) через некоторое отображение шаблонного квадратного элемента. Например, криволинейный четырёхугольник можно задать отображением шаблонного элемента $\Omega^E=\left[-1,1\right]\times\left[-1,1\right]$ с помощью соотношений

$$x=\sum_{i=1}^{9}\hat{\varphi}_i^{\mathrm{bq}}\left(\xi,\eta\right)\hat{x}_i\,,\qquad y=\sum_{i=1}^{9}\hat{\varphi}_i^{\mathrm{bq}}\left(\xi,\eta\right)\hat{y}_i\,, \qquad (4.64)$$

где $\hat{\varphi}_i^{\mathrm{bq}}$ – стандартные лагранжевы биквадратичные (узловые) функции на Ω^E, а $\left(\hat{x}_i,\hat{y}_i\right)$ – координаты девяти узлов элемента Ω_m. Тогда определяемые соотношени-

ем (4.61) базисные вектор-функции (с функциональной матрицей $\mathbf{J}$, полученной для отображения (4.64) будут иметь ненулевые касательные составляющие также *только на одной стороне* криволинейного четырёхугольника, и эти касательные будут зависеть от координат *только тех узлов* четырёхугольника Ω_m, *которые лежат на этой стороне*.

Базисные вектор-функции высокого порядка на четырёхугольнике (в том числе и на криволинейном) также удобно вводить через базисные вектор-функции высокого порядка шаблонного элемента Ω^E. Для этого можно, например, использовать шаблонные вектор-функции иерархического типа, которые были введены нами в разделе 4.5.2 (см. соотношения (4.34)–(4.36)). Базисные же вектор-функции высокого порядка $\hat{\vec{\psi}}_i\left(x,y\right)$ конечного элемента Ω_m получаются из вектор-функций $\hat{\vec{\varphi}}_i\left(\xi,\eta\right)$ шаблонного элемента Ω^E с помощью того же соотношения (4.61)

.

И в заключение обратим внимание на то, что нумерация узлов векторного четырёхугольника (которая косвенно определяет нумерацию его базисных вектор-функций) не может быть абсолютно произвольной, поскольку это может привести к разрывности глобальных базисных вектор-функций (из-за того, что касательные составляющие локальных вектор-функций одного и того же ребра могут оказаться направленными в разные стороны). Здесь, к сожалению, проблему нельзя решить столь же просто, как для треугольников (т.е. использовать только такие локальные нумерации вершин, которые упорядочены по возрастанию соответствующих им глобальных номеров). Простейшим выходом для четырёхугольников может быть использование таких нумераций узлов и рёбер, которые подобны нумерациям в стандартных прямоугольных сетках (что и позволит для любых двух смежных четырёхугольников автоматически обеспечить одинаковую направленность локальных базисных вектор-функций, ассоциированных с ребром, по которому эти четырёхугольники являются смежными).

4.6. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ ИЗ ПРОСТРАНСТВА $\mathbb{H}^{\mathrm{rot}}$ НА ТРЁХМЕРНЫХ СЕТКАХ

Общие принципы построения базисных вектор-функций из пространства $\mathbb{H}^{\mathrm{rot}}$ на трёхмерных сетках практически ничем не отличаются от принципов построения базисных вектор-функций на двумерных сетках (см. раздел 4.5.1). Здесь мы рассмотрим, как эти базисные функции строятся на наиболее часто употребляемых типах трёхмерных конечных элементов (при этом мы будем существенным образом опираться на сведения, представленные в разделе 4.5).

4.6.1. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ НА ПАРАЛЛЕЛЕПИПЕДЕ

Базисные вектор-функции из пространства $\mathbb{H}^{\mathrm{rot}}$ на параллелепипедах строятся наиболее просто (по сравнению с трёхмерными конечными элементами других типов), и основные принципы построения этих вектор-функций практически ничем не отличаются от принципов построения базисных вектор-функций на прямоугольниках (см. раздел 4.5.2).

Рассмотрим параллелепипед $\Omega_{rsp} = \left[x_r, x_{r+1}\right] \times \left[y_s, y_{s+1}\right] \times \left[z_p, z_{p+1}\right]$. Определим на нём двенадцать (ассоциированных с рёбрами) базисных вектор-функций таких, что каждая из них имеет направление строго вдоль одной оси координат, параллельной ребру, с которым она ассоциирована. При этом модуль каждой базисной вектор-функции внутри Ω_{rsp} является билинейной функцией двух других координат, направлению осей которых она перпендикулярна.

Например, базисная вектор-функция, ассоциированная с ребром $\left\{y = y_s,\ \ z = z_p\right\}$, имеет вид

$$\vec{\psi}_1 = \begin{pmatrix} \dfrac{y_{s+1} - y}{h_y} \cdot \dfrac{z_{p+1} - z}{h_z} \\ 0 \\ 0 \end{pmatrix},$$

где $h_y = y_{s+1} - y_s$, $h_z = y_{p+1} - y_p$. Очевидно, эта базисная вектор-функция имеет ненулевые касательные составляющие только на двух гранях Ω_{rsp} – это грани $y = y_s$ и $z = z_p$. При этом на каждой из этих граней она изменяется линейно по нефиксированной (и не совпадающей с x) координате от значения 1 на ребре, с которым

она ассоциирована, до нуля на противоположном ребре. Тем самым обеспечивается *непрерывная сшивка касательных* этих базисных вектор-функций в параллелепипеидальной сетке (и принадлежность полученных функций пространству $\mathbb{H}^{\mathrm{rot}}$).

Параллелепипеидальный векторный элемент, кроме того, что он имеет определённую самостоятельную ценность для решения электромагнитных задач в областях с границами, параллельными координатным осям, является основой для построения векторного шестигранного элемента (в том числе и с искривлёнными гранями), наиболее эффективного по точности и возможностям учёта сложной геометрии расчётной области. Поэтому то, как строятся базисные вектор-функции высоких порядков, мы покажем на примере шаблонного элемента-кубика $\Omega^E = [-1,1]\times[-1,1]\times[-1,1]$, который является базовым при построении векторных базисных функций на шестигранных элементах. Заметим, что в векторном МКЭ, как и в узловом, иерархические базисы по сравнению с лагранжевыми базисами того же порядка гораздо более удобны при использовании p-технологии уточнения конечноэлементного решения (повышение точности за счет увеличения порядка элемента), могут давать меньшую заполненность конечноэлементной СЛАУ и более быструю сходимость итерационных методов её решения.

Введём одномерные скалярные иерархические функции:

$$\phi_-(\vartheta) = \frac{1-\vartheta}{2}, \qquad \phi_+(\vartheta) = \frac{1+\vartheta}{2}, \qquad \phi_p(\vartheta) = \vartheta^{p-2}\left(1-\vartheta^2\right), \;\; p \geq 2. \tag{4.65}$$

Базис первого порядка на Ω^E содержит, как уже говорилось, 12 вектор-функций, которые через одномерные функции $\phi_-(\vartheta)$ и $\phi_+(\vartheta)$ могут быть записаны в виде

$$\hat{\vec{\varphi}}_1 = \begin{pmatrix} \phi_-(\eta)\phi_-(\zeta) \\ 0 \\ 0 \end{pmatrix}, \qquad \hat{\vec{\varphi}}_2 = \begin{pmatrix} \phi_+(\eta)\phi_-(\zeta) \\ 0 \\ 0 \end{pmatrix}, \qquad \hat{\vec{\varphi}}_3 = \begin{pmatrix} \phi_-(\eta)\phi_+(\zeta) \\ 0 \\ 0 \end{pmatrix},$$

$$\hat{\vec{\varphi}}_4 = \begin{pmatrix} \phi_+(\eta)\phi_+(\zeta) \\ 0 \\ 0 \end{pmatrix}, \qquad \hat{\vec{\varphi}}_5 = \begin{pmatrix} 0 \\ \phi_-(\xi)\phi_-(\zeta) \\ 0 \end{pmatrix}, \qquad \hat{\vec{\varphi}}_6 = \begin{pmatrix} 0 \\ \phi_+(\xi)\phi_-(\zeta) \\ 0 \end{pmatrix},$$

$$\hat{\vec{\varphi}}_7 = \begin{pmatrix} 0 \\ \phi_-(\xi)\phi_+(\zeta) \\ 0 \end{pmatrix}, \qquad \hat{\vec{\varphi}}_8 = \begin{pmatrix} 0 \\ \phi_+(\xi)\phi_+(\zeta) \\ 0 \end{pmatrix}, \qquad \hat{\vec{\varphi}}_9 = \begin{pmatrix} 0 \\ 0 \\ \phi_-(\xi)\phi_-(\eta) \end{pmatrix},$$

$$\hat{\vec{\varphi}}_{10} = \begin{pmatrix} 0 \\ 0 \\ \phi_+(\xi)\phi_-(\eta) \end{pmatrix}, \qquad \hat{\vec{\varphi}}_{11} = \begin{pmatrix} 0 \\ 0 \\ \phi_-(\xi)\phi_+(\eta) \end{pmatrix}, \qquad \hat{\vec{\varphi}}_{12} = \begin{pmatrix} 0 \\ 0 \\ \phi_+(\xi)\phi_+(\eta) \end{pmatrix},$$

и все эти вектор-функции ассоциированы с рёбрами конечного элемента.

Для построения иерархического базиса второго порядка к уже имеющимся двенадцати вектор-функциям первого порядка добавляется 12 вектор-функций, ассоциированных с рёбрами, 24 вектор-функции, ассоциированные с гранями, и 6 вектор-функций, ассоциированных с самим конечным элементом.

Добавляемые вектор-функции, ассоциированные с рёбрами, получаются следующим образом. Каждая вектор-функция первого порядка умножается на координату, вдоль которой она направлена:

$$\hat{\vec{\varphi}}_{13} = \xi\hat{\vec{\varphi}}_1, \quad \hat{\vec{\varphi}}_{14} = \xi\hat{\vec{\varphi}}_2, \quad \hat{\vec{\varphi}}_{15} = \xi\hat{\vec{\varphi}}_3, \quad \hat{\vec{\varphi}}_{16} = \xi\hat{\vec{\varphi}}_4,$$
$$\hat{\vec{\varphi}}_{17} = \eta\hat{\vec{\varphi}}_5, \quad \hat{\vec{\varphi}}_{18} = \eta\hat{\vec{\varphi}}_6, \quad \hat{\vec{\varphi}}_{19} = \eta\hat{\vec{\varphi}}_7, \quad \hat{\vec{\varphi}}_{20} = \eta\hat{\vec{\varphi}}_8,$$
$$\hat{\vec{\varphi}}_{21} = \zeta\hat{\vec{\varphi}}_9, \quad \hat{\vec{\varphi}}_{22} = \zeta\hat{\vec{\varphi}}_{10}, \quad \hat{\vec{\varphi}}_{23} = \zeta\hat{\vec{\varphi}}_{11}, \quad \hat{\vec{\varphi}}_{24} = \zeta\hat{\vec{\varphi}}_{12}.$$

Очевидно, каждая из этих вектор-функций имеет ненулевые касательные составляющие только на двух гранях, смежных по ребру, с которым эта вектор-функция ассоциирована.

Ассоциированные с гранями вектор-функции второго порядка имеют следующий вид. С гранью $\xi = -1$ ассоциируются вектор-функции

$$\hat{\vec{\varphi}}_{25} = \begin{pmatrix} 0 \\ \phi_2(\zeta)\phi_-(\xi) \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{26} = \eta\hat{\vec{\varphi}}_{25}, \quad \hat{\vec{\varphi}}_{27} = \begin{pmatrix} 0 \\ 0 \\ \phi_2(\eta)\phi_-(\xi) \end{pmatrix}, \quad \hat{\vec{\varphi}}_{28} = \zeta\hat{\vec{\varphi}}_{27}.$$

С гранью $\xi = 1$ ассоциируются

$$\hat{\vec{\varphi}}_{29} = \begin{pmatrix} 0 \\ \phi_2(\zeta)\phi_+(\xi) \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{30} = \eta\hat{\vec{\varphi}}_{29}, \quad \hat{\vec{\varphi}}_{31} = \begin{pmatrix} 0 \\ 0 \\ \phi_2(\eta)\phi_+(\xi) \end{pmatrix}, \quad \hat{\vec{\varphi}}_{32} = \zeta\hat{\vec{\varphi}}_{31}.$$

С гранью $\eta = -1$ ассоциируются

$$\hat{\vec{\varphi}}_{33} = \begin{pmatrix} \phi_2(\zeta)\phi_-(\eta) \\ 0 \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{34} = \xi\hat{\vec{\varphi}}_{33}, \quad \hat{\vec{\varphi}}_{35} = \begin{pmatrix} 0 \\ 0 \\ \phi_2(\xi)\phi_-(\eta) \end{pmatrix}, \quad \hat{\vec{\varphi}}_{36} = \zeta\hat{\vec{\varphi}}_{35}.$$

С гранью $\eta = 1$ ассоциируются

$$\hat{\vec{\varphi}}_{37} = \begin{pmatrix} \phi_2(\zeta)\phi_+(\eta) \\ 0 \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{38} = \xi\hat{\vec{\varphi}}_{37}, \quad \hat{\vec{\varphi}}_{39} = \begin{pmatrix} 0 \\ 0 \\ \phi_2(\xi)\phi_+(\eta) \end{pmatrix}, \quad \hat{\vec{\varphi}}_{40} = \zeta\hat{\vec{\varphi}}_{39}.$$

С гранью $\zeta = -1$ ассоциируются

$$\hat{\vec{\varphi}}_{41} = \begin{pmatrix} \phi_2(\eta)\phi_-(\zeta) \\ 0 \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{42} = \xi\hat{\vec{\varphi}}_{41}, \quad \hat{\vec{\varphi}}_{43} = \begin{pmatrix} 0 \\ \phi_2(\xi)\phi_-(\zeta) \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{44} = \eta\hat{\vec{\varphi}}_{43}.$$

С гранью $\zeta = 1$ ассоциируются

$$\hat{\vec{\varphi}}_{45} = \begin{pmatrix} \phi_2(\eta)\phi_+(\zeta) \\ 0 \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{46} = \xi\hat{\vec{\varphi}}_{45}, \quad \hat{\vec{\varphi}}_{47} = \begin{pmatrix} 0 \\ \phi_2(\xi)\phi_+(\zeta) \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{48} = \eta\hat{\vec{\varphi}}_{47}.$$

Очевидно, что все эти вектор-функции $\hat{\vec{\varphi}}_{25} \div \hat{\vec{\varphi}}_{48}$ имеют ненулевые касательные только на одной грани, и поэтому соответствующие глобальные вектор-функции получаются сшивкой только двух локальных вектор-функций.

И, наконец, вектор-функции второго порядка, ассоциированные с самим элементом, имеют вид

$$\hat{\vec{\varphi}}_{49} = \begin{pmatrix} \phi_2(\eta)\phi_2(\zeta) \\ 0 \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{50} = \hat{\vec{\varphi}}_{49}\xi, \quad \hat{\vec{\varphi}}_{51} = \begin{pmatrix} 0 \\ \phi_2(\xi)\phi_2(\zeta) \\ 0 \end{pmatrix}, \quad \hat{\vec{\varphi}}_{52} = \hat{\vec{\varphi}}_{51}\eta,$$

$$\hat{\vec{\varphi}}_{53} = \begin{pmatrix} 0 \\ 0 \\ \phi_2(\xi)\phi_2(\eta) \end{pmatrix}, \quad \hat{\vec{\varphi}}_{54} = \hat{\vec{\varphi}}_{53}\zeta.$$

Все эти вектор-функции имеют нулевые касательные на всех гранях конечного элемента, и поэтому каждая из них фактически является прототипом глобальной вектор-функцией, отличной от нуля внутри только одного конечного элемента.

Рассмотрим, как может быть построен иерархический базис p-го порядка путём добавления недостающих вектор-функций в иерархический базис $(p-1)$-го

порядка. Пусть иерархический базис $(p-1)$-го порядка содержит k вектор-функций ($k = 3p^2(p-1)$). Покажем, что для построения базиса p-го порядка к базису $(p-1)$-го порядка должны быть добавлены 12 вектор-функций, ассоциированных с рёбрами, $24(p-1)$ вектор-функций, ассоциированных с гранями, и $9p^2 - 21p + 12$ вектор-функций, ассоциированных с самим элементом.

Для удобства записи базисных вектор-функций кроме одномерных функций $\phi_-(\vartheta)$, $\phi_+(\vartheta)$ и $\phi_p(\vartheta)$ для $p \geq 2$ (см. (4.65)) введём ещё две одномерные функции

$$\phi_0(\vartheta) = 1, \qquad \phi_1(\vartheta) = \vartheta.$$

Добавляемые к базису $(p-1)$-го порядка вектор-функции, ассоциированные с рёбрами, строятся аналогично тому, как это было сделано для вектор-функций второго порядка. Каждая вектор-функция первого порядка умножается на функцию $\phi_{p-1}(\vartheta)$, где ϑ – координата, вдоль которой направлено соответствующее этой вектор-функции ребро, т.е.

$$\hat{\vec{\varphi}}_{k+1} = \phi_{p-1}(\xi)\hat{\vec{\varphi}}_1, \ \ldots, \ \hat{\vec{\varphi}}_{k+12} = \phi_{p-1}(\zeta)\hat{\vec{\varphi}}_{12}.$$

Добавляемые к базису $(p-1)$-го порядка вектор-функции, ассоциированные с гранями, строятся по следующему принципу. На каждой грани строятся вектор-функции двух направлений, совпадающих с направлениями рёбер этой грани (по $2(p-1)$ вектор-функции каждого направления). У этих вектор-функций, как и у всех остальных, только одна ненулевая компонента. Эта компонента является произведением одномерных функций координат ξ, η или ζ, одна из которых является функцией $\phi_-(\vartheta)$ или $\phi_+(\vartheta)$ (где ϑ – то из направлений ξ, η или ζ, которое перпендикулярно рассматриваемой грани, а индекс «$-$» или «$+$» у ϕ определяет положение грани, т.е. «$-$» – это грань $\vartheta = -1$, а «$+$» – грань $\vartheta = 1$).

Остальные два сомножителя являются одномерными иерархическими функциями двух оставшихся координат. При этом для каждого из двух направлений (касательных к грани и определяемых оставшимися координатами) базисные вектор-функции являются функциями двух типов.

В вектор-функциях первого типа вторая одномерная функция-сомножитель координаты, вдоль которой направлена вектор-функция, является иерархическим полиномом степени $p-1$, а третья одномерная функция-сомножитель – полино-

мом от оставшейся третьей координаты степени от 2 до $p-1$. Например, для грани $\xi=-1$ это вектор-функции

$$\hat{\vec{\varphi}}_{k+12+i-1}=\begin{pmatrix}0\\ \phi_{-}\left(\xi\right)\phi_{p-1}\left(\eta\right)\phi_{i}\left(\zeta\right)\\ 0\end{pmatrix},\qquad i=2\ldots p-1$$

$$\hat{\vec{\varphi}}_{k+12+p-2+i-1}=\begin{pmatrix}0\\ 0\\ \phi_{-}\left(\xi\right)\phi_{p-1}\left(\zeta\right)\phi_{i}\left(\eta\right)\end{pmatrix},\qquad i=2\ldots p-1,$$

из которых, очевидно, первые $p-2$ вектор-функции направлены вдоль координаты η, а последние $p-2$ вектор-функции – вдоль координаты ζ.

В вектор-функциях второго типа вторая одномерная функция-сомножитель координаты, перпендикулярно оси которой направлена рассматриваемая вектор-функция (при этом её направление, естественно, должно быть касательным к рассматриваемой грани), является иерархическим полиномом степени p, а третья одномерная функция-сомножитель – полиномом от оставшейся третьей координаты степени от 0 до $p-1$, из которых неиерархическими являются только полиномы нулевой и первой степени. Так, для грани $\xi=-1$ это вектор-функции

$$\hat{\vec{\varphi}}_{k+12+2\left(p-2\right)+i+1}=\begin{pmatrix}0\\ \phi_{-}\left(\xi\right)\phi_{p}\left(\zeta\right)\phi_{i}\left(\eta\right)\\ 0\end{pmatrix},\qquad i=0\ldots p-1,$$

$$\hat{\vec{\varphi}}_{k+12+2\left(p-2\right)+p+i+1}=\begin{pmatrix}0\\ 0\\ \phi_{-}\left(\xi\right)\phi_{p}\left(\eta\right)\phi_{i}\left(\zeta\right)\end{pmatrix},\quad i=0\ldots p-1.$$

Очевидно, первые p из этих вектор-функций направлены вдоль координаты η, а последние p – вдоль координаты ζ. Всего же на одной грани получается $2\left(p-2\right)+2p=4\left(p-1\right)$ новых вектор-функций.

Таким образом, к иерархическому базису $\left(p-1\right)$-го порядка должны быть добавлены $24\left(p-1\right)$ новые вектор-функции, ассоциированные с гранями.

И, наконец, рассмотрим добавляемые к базису $\left(p-1\right)$-го порядка вектор-функции, ассоциированные с самим элементом, – это вектор-функции трёх на-

правлений (т.е. каждая вектор-функция по-прежнему имеет лишь одну ненулевую компоненту, являющуюся произведением трёх одномерных полиномов от координат ξ, η или ζ). Такие вектор-функции (каждого из трёх направлений) можно условно разбить на два типа по виду образующих их ненулевые компоненты одномерных функций-сомножителей.

В вектор-функциях первого типа первая одномерная функция-сомножитель – это иерархический полином ϕ_{p-1} от координаты, вдоль оси которой направлена вектор-функция, а вторая и третья одномерные функции-сомножители – это иерархические полиномы от оставшихся координат степени от 2 до $p-1$. Так, вектор-функции этого типа, направленные вдоль оси координаты ξ, – это вектор-функции вида

$$\hat{\vec{\varphi}}_{k+24p-12+(p-2)(i-2)+j-1} = \begin{pmatrix} \phi_{p-1}(\xi)\phi_i(\eta)\phi_j(\zeta) \\ 0 \\ 0 \end{pmatrix}, \qquad i,j = 2...p-1.$$

В вектор-функциях второго типа первая функция-сомножитель – это один из полиномов степени от 0 до $p-1$ (из которых неиерархическими являются только полиномы нулевой и первой степени) от той координаты, вдоль оси которой направлена вектор-функция. Вторая одномерная функция-сомножитель – это иерархический полином степени p от одной из оставшихся координат. Третья же одномерная функция-сомножитель является полиномом степени от 2 до p от последней оставшейся координаты. Вектор-функции этого типа, направленные вдоль координаты ξ, имеют следующий вид (при этом мы последними пронумеруем те вектор-функции, у которых сомножители являются полиномами p-й степени и от η, и от ζ, и поэтому эти вектор-функции мы определим отдельной строкой):

$$\hat{\vec{\varphi}}_{k+24p-12+(p-2)^2+i(p-2)+j-1} = \begin{pmatrix} \phi_i(\xi)\phi_p(\eta)\phi_j(\zeta) \\ 0 \\ 0 \end{pmatrix}, \qquad i{=}0...p{-}1, \quad j{=}2...p{-}1,$$

$$\hat{\vec{\varphi}}_{k+24p-12+(p-2)^2+p(p-2)+i(p-2)+j-1} = \begin{pmatrix} \phi_i(\xi)\phi_p(\zeta)\phi_j(\eta) \\ 0 \\ 0 \end{pmatrix}, \; i{=}0...p{-}1, \; j{=}2...p{-}1,$$

$$\hat{\vec{\varphi}}_{k+24p-12+(p-2)^2+2p(p-2)+i} = \begin{pmatrix} \phi_i(\xi)\phi_p(\eta)\phi_p(\zeta) \\ 0 \\ 0 \end{pmatrix}, \qquad i = 0...p-1.$$

Итак, к базису $(p-1)$-го порядка добавляется $(p-2)^2 + 2p(p-2) + p = 3p^2 - 7p + 4$ ассоциированных с самим элементом вектор-функций одного направления.

Аналогично строятся добавляемые к базису $(p-1)$-го порядка и ассоциированные с самим элементом вектор-функции двух других направлений, и всего получается $9p^2 - 21p + 12$ новых вектор-функций, ассоциированных с самим элементом.

Таким образом, для построения иерархического базиса p-го порядка к вектор-функциям $(p-1)$-го порядка должно быть добавлено $9p^2 + 3p$ новых вектор-функций. Общее же количество вектор-функций в базисе p-го порядка равно $3p(p+1)^2$.

В заключение приведём вид лагранжевых базисных вектор-функций высокого порядка на шаблонном элементе Ω^E.

При построении лагранжева базиса в качестве шаблонного элемента мы будем использовать единичный кубик $\Omega^E = [0,1] \times [0,1] \times [0,1]$, поскольку одномерные *лагранжевы* базисные функции удобней задавать на отрезке $[0,1]$.

Введём на отрезке $[0,1]$ стандартные одномерные базисные полиномы Лагранжа p-й степени ($p \geq 1$; для $p = 0$ $\phi_0^0(\vartheta) \equiv 1$):

$$\phi_i^p(\vartheta) = \prod_{j=0, j\neq i}^{p} \frac{(\vartheta - j/p)}{(i/p - j/p)}, \qquad i = 0...p.$$

На параллелепипеде в лагранжевом векторном базисе p-го порядка так же, как и в иерархическом, по $p(p+1)^2$ вектор-функций каждого из трёх направлений (т.е. всего $3p(p+1)^2$ вектор-функций). Единственная ненулевая компонента вектор-функций одного направления является произведением трёх одномерных лагранжевых полиномов. Два из них – это полиномы p-й степени от координат, оси которых перпендикулярны направлению рассматриваемой вектор-функции, а тре-

тий – полином $\left(p-1\right)$-й степени координаты, ось которой параллельна направлению рассматриваемой вектор-функции, т.е.

$$\hat{\vec{\varphi}}_{i(p+1)^2+j\left(p+1\right)+l+1}=\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_j^p\left(\eta\right)\phi_l^p\left(\zeta\right)\\0\\0\end{pmatrix},\qquad i=0\ldots p-1,\quad j,l=0\ldots p,$$

$$\hat{\vec{\varphi}}_{p(p+1)^2+ip(p+1)+j\left(p+1\right)+l+1}=\begin{pmatrix}0\\\phi_i^p\left(\xi\right)\phi_j^{p-1}\left(\eta\right)\phi_l^p\left(\zeta\right)\\0\end{pmatrix},\quad i,l=0\ldots p,\qquad j=0\ldots p-1,$$

$$\hat{\vec{\varphi}}_{2p(p+1)^2+ip(p+1)+jp+l+1}=\begin{pmatrix}0\\0\\\phi_i^p\left(\xi\right)\phi_j^p\left(\eta\right)\phi_l^{p-1}\left(\zeta\right)\end{pmatrix},\qquad i,j=0\ldots p,\quad l=0\ldots p-1.$$

В лагранжевом базисе с рёбрами ассоциируются вектор-функции, у которых в ненулевой компоненте нижние индексы обоих одномерных полиномов степени p принимают крайние значения 0 и p (это вектор-функции $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_0^p\left(\eta\right)\phi_0^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_p^p\left(\eta\right)\phi_0^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_0^p\left(\eta\right)\phi_p^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_p^p\left(\eta\right)\phi_p^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}0\\\phi_0^p\left(\xi\right)\phi_j^{p-1}\left(\eta\right)\phi_0^p\left(\zeta\right)\\0\end{pmatrix}$, $\begin{pmatrix}0\\\phi_p^p\left(\xi\right)\phi_j^{p-1}\left(\eta\right)\phi_0^p\left(\zeta\right)\\0\end{pmatrix}$ и т.д.). С гранями ассоциируются вектор-функции, у которых среди трёх сомножителей-полиномов ненулевой компоненты нижний индекс только одного из одномерных полиномов степени p принимает крайние значения 0 и p (это вектор-функции $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_0^p\left(\eta\right)\phi_k^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_k^p\left(\eta\right)\phi_0^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_p^p\left(\eta\right)\phi_k^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}\phi_i^{p-1}\left(\xi\right)\phi_k^p\left(\eta\right)\phi_p^p\left(\zeta\right)\\0\\0\end{pmatrix}$, $\begin{pmatrix}0\\\phi_0^p\left(\xi\right)\phi_j^{p-1}\left(\eta\right)\phi_k^p\left(\zeta\right)\\0\end{pmatrix}$, $\begin{pmatrix}0\\\phi_k^p\left(\xi\right)\phi_j^{p-1}\left(\eta\right)\phi_0^p\left(\zeta\right)\\0\end{pmatrix}$ и т.д., $k\neq 0$ и $k\neq p$).

Все остальные вектор-функции ассоциированы с самим конечным элементом.

Отметим, что информация о том, *с чем* (ребром, гранью или самим конечным элементом) *ассоциирована* локальная базисная вектор-функция, может быть очень полезна для глобальной нумерации базисных вектор-функций на сетках с шестигранными ячейками и определения соответствия между локальной и глобальной нумерациями базисных вектор-функций на каждом конечном элементе. Фактически это соответствие и определяет глобальные базисные вектор-функции как сшивку локальных и, как следствие, сборку глобальных матрицы и вектора из локальных.

Соответствие между локальной и глобальной нумерацией лагранжевых базисных вектор-функций можно установить и без привязки их к рёбрам, граням и элементам, а *по всем узлам*, в которых только одна глобальная вектор-функция отлична от нуля. Такой способ, конечно, проще и применяется для регулярных сеток, но для нерегулярных сеток он является вычислительно гораздо более затратным.

Информация о том, с каким ребром или гранью ассоциирована базисная вектор-функция, нужна при реализации технологии $\mathbf{T}$-преобразования для согласования векторных конечных элементов на несогласованных сетках с шестигранными ячейками (и, как следствие, при реализации h-, p- и $p-h$-технологий уточнения конечноэлементных решений).

Итак, рассмотренные нами здесь базисы высоких порядков на шаблонном элементе-кубике являются основой для реализации векторного МКЭ на шестигранных сетках (включая и элементы с искривлёнными гранями). А поскольку шестигранный элемент очень удобен для описания сложной трёхмерной геометрии, рассмотренный здесь материал служит основой для создания многих эффективных алгоритмов решения сложных практических задач.

4.6.2. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ НА ТЕТРАЭДРЕ

Базисные вектор-функции на тетраэдре, как и на треугольнике, удобно определить через $\mathcal{L}$-координаты. В качестве направлений этих базисных вектор-функций можно взять векторы $\operatorname{grad}\mathcal{L}_i$, $i=1\ldots4$. Нетрудно убедиться, что каждый из векторов $\operatorname{grad}\mathcal{L}_i$ перпендикулярен грани, противоположной узлу с локальным номером i. Поэтому вектор-функции вида $\mathcal{L}_j \operatorname{grad}\mathcal{L}_i$ $(i \neq j)$ имеют нулевые каса-

тельные уже на двух гранях, противоположных вершинам тетраэдров с локальными номерами i и j (поскольку на грани, противоположной вершине с локальным номером j, функция $\boldsymbol{\mathcal{L}}_j \equiv 0$).

Таким образом, вектор-функция $\boldsymbol{\mathcal{L}}_j \operatorname{grad} \boldsymbol{\mathcal{L}}_i$ имеет ненулевые касательные только на двух гранях, которые содержат ребро, соединяющее вершины с локальными номерами i и j. Покажем, что касательные вектор-функции $\boldsymbol{\mathcal{L}}_j \operatorname{grad} \boldsymbol{\mathcal{L}}_i$ к каждой из этих граней зависят только от координат вершин тетраэдра, лежащих на рассматриваемой грани (и никак не зависят от координат вершины, противоположной этой грани).

Рассмотрим в качестве примера вектор-функцию $\boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_1$. Очевидно, эта вектор-функция имеет ненулевые касательные составляющие только на гранях, содержащих вершины тетраэдра с локальными номерами 1 и 2, т.е. на гранях $\{1,2,3\}$ и $\{1,2,4\}$ (к грани $\{2,3,4\}$ вектор $\operatorname{grad} \boldsymbol{\mathcal{L}}_1$ ортогонален, а на грани $\{1,3,4\}$ функция $\boldsymbol{\mathcal{L}}_2$ равна нулю).

Покажем, что на грани $\{1,2,3\}$ касательные составляющие вектор-функции $\operatorname{grad} \boldsymbol{\mathcal{L}}_1$ не зависят от координат $(\hat{x}_4, \hat{y}_4, \hat{z}_4)$ четвёртой вершины тетраэдра. Для этого вычислим два скалярных произведения $\operatorname{grad} \boldsymbol{\mathcal{L}}_1 \cdot \vec{\mathbf{v}}_{12}$ и $\operatorname{grad} \boldsymbol{\mathcal{L}}_1 \cdot \vec{\mathbf{v}}_{13}$, в которых векторы

$$\vec{\mathbf{v}}_{12} = \left(\hat{x}_1 - \hat{x}_2, \hat{y}_1 - \hat{y}_2, \hat{z}_1 - \hat{z}_2\right)^{\mathrm{T}}, \qquad \vec{\mathbf{v}}_{13} = \left(\hat{x}_1 - \hat{x}_3, \hat{y}_1 - \hat{y}_3, \hat{z}_1 - \hat{z}_3\right)^{\mathrm{T}} \tag{4.66}$$

направлены вдоль рёбер, соединяющих вершины с локальными номерами 1 и 2 и с локальными номерами 1 и 3.

Воспользуемся тем, что коэффициенты α_j^i, определяющие $\boldsymbol{\mathcal{L}}$-координаты в виде

$$\boldsymbol{\mathcal{L}}_i = \alpha_0^i + \alpha_1^i x + \alpha_2^i y + \alpha_3^i z, \tag{4.67}$$

являются компонентами матрицы $\boldsymbol{a}$, обратной к составленной из координат вершин тетраэдра матрице $\mathbf{D}$:

$$\begin{pmatrix} \alpha_0^1 & \alpha_1^1 & \alpha_2^1 & \alpha_3^1 \\ \alpha_0^2 & \alpha_1^2 & \alpha_2^2 & \alpha_3^2 \\ \alpha_0^3 & \alpha_1^3 & \alpha_2^3 & \alpha_3^3 \\ \alpha_0^4 & \alpha_1^4 & \alpha_2^4 & \alpha_3^4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ \hat{x}_1 & \hat{x}_2 & \hat{x}_3 & \hat{x}_4 \\ \hat{y}_1 & \hat{y}_2 & \hat{y}_3 & \hat{y}_4 \\ \hat{z}_1 & \hat{z}_2 & \hat{z}_3 & \hat{z}_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.68}$$

Из соотношения (4.68) получаем (как результат умножения первой строки $\boldsymbol{a}$ на первый и второй столбец матрицы $\mathbf{D}$):

$$\alpha_0^1 + \alpha_1^1 \hat{x}_1 + \alpha_2^1 \hat{y}_1 + \alpha_3^1 \hat{z}_1 = 1, \tag{4.69}$$

$$\alpha_0^1 + \alpha_1^1 \hat{x}_2 + \alpha_2^1 \hat{y}_2 + \alpha_3^1 \hat{z}_2 = 0. \tag{4.70}$$

Вычитая (4.70) из (4.69), получаем равенство

$$\alpha_1^1 \left(\hat{x}_1 - \hat{x}_2\right) + \alpha_2^1 \left(\hat{y}_1 - \hat{y}_2\right) + \alpha_3^1 \left(\hat{z}_1 - \hat{z}_2\right) = 1,$$

которое с учётом обозначения (4.66) и представления $\boldsymbol{\mathcal{L}}_1$ в виде (4.67) может быть записано в виде

$$\operatorname{grad} \boldsymbol{\mathcal{L}}_1 \cdot \vec{\mathbf{v}}_{12} = 1, \tag{4.71}$$

т.е. проекция вектора $\operatorname{grad} \boldsymbol{\mathcal{L}}_1$ на единичный вектор $\vec{\mathbf{v}}_{12} / l_{12}$ (где $l_{12} = \sqrt{\left(\hat{x}_1 - \hat{x}_2\right)^2 + \left(\hat{y}_1 - \hat{y}_2\right)^2 + \left(\hat{z}_1 - \hat{z}_2\right)^2}$ – длина вектора $\vec{\mathbf{v}}_{12}$) *не зависит* от координат *четвёртой* вершины тетраэдра (и зависит только от координат первой и второй его вершин).

Абсолютно аналогично доказывается, что

$$\operatorname{grad} \boldsymbol{\mathcal{L}}_1 \cdot \vec{\mathbf{v}}_{13} = 1, \tag{4.72}$$

т.е. проекция вектора $\operatorname{grad} \boldsymbol{\mathcal{L}}_1$ и на единичный вектор $\vec{\mathbf{v}}_{13} / l_{13}$ (где l_{13} – длина вектора $\vec{\mathbf{v}}_{13}$) *не зависит* от координат *четвёртой* вершины тетраэдра (а зависит только от координат первой и третьей его вершин).

С учётом неколлинеарности векторов $\vec{\mathbf{v}}_{12}$ и $\vec{\mathbf{v}}_{13}$ из (4.71) и (4.72) сразу следует, что проекция вектора $\operatorname{grad} \boldsymbol{\mathcal{L}}_1$ на любой вектор, лежащий в плоскости грани $\{1,2,3\}$ (т.е. грани, определяемой вершинами тетраэдра с локальными номерами 1, 2 и 3), *не зависит от координат* $\left(\hat{x}_4, \hat{y}_4, \hat{z}_4\right)$ *четвёртой вершины* тетраэдра, а значит, и касательная составляющая вектора $\operatorname{grad} \boldsymbol{\mathcal{L}}_1$ на грани $\{1,2,3\}$ не зависит от координат вершины с номером 4. При этом в ходе доказательства мы фактически получили, что *касательная* к грани $\{1,2,3\}$ *составляющая трёхмерной* $\boldsymbol{\mathcal{L}}$-координаты $\boldsymbol{\mathcal{L}}_1$ ведёт себя абсолютно так же, как и градиент соответствующей *двумерной* $\boldsymbol{\mathcal{L}}$-координаты треугольника $\{1,2,3\}$.

Очевидно, и вектор-функция $\mathcal{L}_2 \operatorname{grad} \mathcal{L}_1$ на грани $\{1,2,3\}$ будет иметь касательную, не зависящую от четвёртой вершины тетраэдра (поскольку функция $\mathcal{L}_2$ на этой грани также не зависит от четвёртой вершины тетраэдра – она линейно меняется от единицы во второй вершине до нуля в вершинах с номерами 1 и 3).

Таким образом, вектор-функция $\mathcal{L}_2 \operatorname{grad} \mathcal{L}_1$ на грани $\{1,2,3\}$ ведёт себя следующим образом. Её касательная к этой грани не зависит от координаты четвёртой вершины (а, точнее, ведёт себя как соответствующая двумерная вектор-функция треугольника $\{1,2,3\}$), и при этом её проекция на единичный вектор $\vec{\mathbf{v}}_{12}/l_{12}$ (направленный вдоль ребра, соединяющего вершины с номерами 1 и 2) также линейно изменяется от значения $1/l_{12}$ в вершине с номером 2 до нуля в вершине с номером 1.

Абсолютно аналогично можно доказать, что и касательная к грани $\{1,2,3\}$ составляющая вектор-функция $\mathcal{L}_1 \operatorname{grad} \mathcal{L}_2$ не зависит от координаты четвёртой вершины тетраэдра, а её проекция на единичный вектор $\vec{\mathbf{v}}_{12}/l_{12}$ линейно изменяется от значения $-1/l_{12}$ в вершине с номером 1 до нуля в вершине с номером 2. Поэтому проекция вектор-функции $\mathcal{L}_2 \operatorname{grad} \mathcal{L}_1 - \mathcal{L}_1 \operatorname{grad} \mathcal{L}_2$ на ребро, соединяющее вершины с локальными номерами 1 и 2, будет постоянной на этом ребре и равной $1/l_{12}$ (как и у соответствующей функции треугольного элемента).

Очевидно, что на грани $\{1,2,4\}$ касательная к ней составляющая вектор-функция $\mathcal{L}_2 \operatorname{grad} \mathcal{L}_1 - \mathcal{L}_1 \operatorname{grad} \mathcal{L}_2$ не будет зависеть от координат третьей вершины тетраэдра (и будет вести себя на этой грани как соответствующая вектор-функция двумерного треугольного элемента).

Поэтому, если на каждом тетраэдре, содержащем некоторое ребро Γ, определить базисную вектор-функцию $\vec{\psi}_k$ в виде $\mathcal{L}_i \operatorname{grad} \mathcal{L}_j - \mathcal{L}_j \operatorname{grad} \mathcal{L}_i$ (где i и j – локальные номера, соответствующие глобальным номерам вершин рассматриваемого ребра Γ), то эта вектор-функция будет иметь непрерывные (ненулевые) касательные составляющие на всех гранях, содержащих ребро Γ. На всех же остальных гранях тех тетраэдров, которые содержат Γ, эта вектор-функция будет иметь нулевые касательные составляющие. Тем самым будет обеспечена непрерывность

касательных составляющих $\vec{\psi}_k$ во всей расчётной области (естественно, $\vec{\psi}_k \equiv \mathbf{0}$ во всех тетраэдрах, не содержащих ребро Γ).

В результате при использовании такого базиса на одном тетраэдре у нас будет определено шесть локальных базисных вектор-функций $\hat{\vec{\psi}}_p$ вида

$$\hat{\vec{\psi}}_p = \boldsymbol{\mathcal{L}}_i \operatorname{grad} \boldsymbol{\mathcal{L}}_j - \boldsymbol{\mathcal{L}}_j \operatorname{grad} \boldsymbol{\mathcal{L}}_i, \qquad (4.73)$$

ассоциированных с соответствующими рёбрами тетраэдра. Номера p этих локальных вектор-функций – это локальные номера рёбер тетраэдра, которые могут быть определены через локальные номера i и j вершин этого ребра. Например, можно ввести следующую локальную нумерацию рёбер тетраэдра: первое ребро соединяет вершины с локальными номерами 1 и 2, второе – вершины 1 и 3, третье – вершины 1 и 4, четвёртое – вершины 2 и 3, пятое – вершины 2 и 4 и, наконец, шестое ребро соединяет вершины с номерами 3 и 4.

Для обработки тетраэдрального векторного элемента (построения его локальных матриц и вектора) удобно связать локальные нумерации его вершин и рёбер заданием двух целочисленных функций $i(p)$ и $j(p)$, которые по локальному номеру p ребра позволяют определять локальные номера i и j его вершин. Для рассмотренной нами чуть выше нумерации эти функции определяются табл. 4.

Таблица 4.

p	1	2	3	4	5	6
$i(p)$	1	1	1	2	2	3
$j(p)$	2	3	4	3	4	4

Рассмотренный нами базис минимального порядка на тетраэдральных векторных конечных элементах (построенный с использованием только вектор-функций вида (4.73)) обладает теми же недостатками, что и аналогичный базис на треугольных векторных элементах. Один из наиболее принципиальных – это резкое ухудшение аппроксимационных свойств при появлении в сетке вытянутых тетраэдров (включение которых в сетку обычно требуется, например, при наличии в расчётной области вытянутых подобластей, при решении задач со скин-эффектом и т.д.). Избежать этих эффектов (при появлении в сетке вытянутых тет-

раэдров) можно путём добавления в локальный базис тетраэдра шести дополнительных (также ассоциированных с его рёбрами) вектор-функций вида

$$\hat{\vec{\psi}}_{6+p} = \boldsymbol{\mathcal{L}}_i \operatorname{grad} \boldsymbol{\mathcal{L}}_j + \boldsymbol{\mathcal{L}}_j \operatorname{grad} \boldsymbol{\mathcal{L}}_i , \qquad (4.74)$$

где $i = i\big(p\big)$ и $j = j\big(p\big)$ – целочисленные функции, определяемые (как и для вектор-функций (4.73)) табл. 4.

В принципе, эквивалентным базису (4.73)–(4.74) на тетраэдре является базис, построенный из двенадцати вектор-функций вида

$$\hat{\vec{\psi}}_p = \boldsymbol{\mathcal{L}}_i \operatorname{grad} \boldsymbol{\mathcal{L}}_j , \qquad (4.75)$$

в котором с каждым ребром, соединяющим вершины с локальными номерами i и j, также ассоциированы две вектор-функции: это вектор-функция $\boldsymbol{\mathcal{L}}_i \operatorname{grad} \boldsymbol{\mathcal{L}}_j$ и вектор-функция $\boldsymbol{\mathcal{L}}_j \operatorname{grad} \boldsymbol{\mathcal{L}}_i$.

Заметим, что базис с вектор-функциями (4.73)–(4.74) (в отличие от базиса с вектор-функциями (4.75)) построен, вообще говоря, по иерархическому принципу и позволяет использовать в одной и той же сетке как элементы с шестью локальными базисными вектор-функциями вида (4.73), так и элементы с добавленными к ним базисными вектор-функциями вида (4.74). Это, естественно, даёт такому базису заметные преимущества (по сравнению с базисом (4.75)), так как позволяет включать в него дополнительные (вида (4.74)) вектор-функции *не на всех*, а только *на вытянутых* (и, естественно, примыкающих к ним) тетраэдрах.

Обратим внимание на то, что на векторном тетраэдре (как и на векторном треугольнике) необходимо следить за *направлением* локальной базисной вектор-функции (т.е. за тем, чтобы локальные базисные вектор-функции одного и того же ребра имели *одинаково направленную* касательную к этому ребру). Решить эту проблему проще всего так же, как и на векторных треугольниках: локальную нумерацию вершин каждого тетраэдра проводить строго по возрастанию их глобальных номеров.

В целом же, даже при использовании минимального базиса (с единственной вектор-функцией на ребре вида (4.73)) тетраэдральные векторные элементы обычно уступают векторным шестигранникам (при примерно одинаковом количестве узлов в сетке) как по точности получаемых конечноэлементных решений, так и по вычислительным затратам (из-за гораздо большего количества рёбер в тетраэдральной сетке по сравнению с сеткой из шестигранников). При этом гео-

метрия расчётной области (даже при наличии в ней сложных границ) с помощью шестигранных элементов может быть описана, в общем-то, не хуже, чем при использовании тетраэдральных элементов. Единственным преимуществом тетраэдральных элементов, как мы уже говорили, является возможность разрежения шага сетки в любых направлениях. Однако использование несогласованных сеток в совокупности с технологией **T**-преобразования позволяет очень эффективно решать и эту проблему на сетках с шестигранными ячейками. К преимуществам сеток с шестигранными ячейками можно отнести и то, что на них проще реализуются элементы высоких порядков. Поэтому для решения практических задач шестигранные элементы (в совокупности с технологией **T**-преобразования) являются, пожалуй, наиболее эффективными.

4.6.3. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ НА ТРЕУГОЛЬНОЙ ПРИЗМЕ

Рассмотрим, как могут быть определены базисные вектор-функции на *прямой* призме с треугольным основанием (наклонные призмы с треугольным основанием мы рассматривать не будем, поскольку вместо них удобнее использовать шестигранные элементы).

Базисные вектор-функции первого порядка на призме (как и на параллелепипеде и тетраэдре) мы будем строить так, чтобы каждая из них имела ненулевые касательные только на двух гранях, пересекающихся по ребру, с которым эта вектор-функция ассоциирована.

Будем считать, что основания призмы перпендикулярны оси z (лежат в плоскостях $z = \mathrm{const}$). Пронумеруем рёбра призмы так, как это показано на рис. 46 (эти номера указаны в кружочках). Базисные вектор-функции, ассоциированные с рёбрами, имеющими номера от 1 до 6, определим следующим образом:

$$\hat{\vec{\psi}}_1 = \left(\boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad} \boldsymbol{\mathcal{L}}_2\right)\frac{\hat{z}_2 - z}{h}, \tag{4.76}$$

$$\hat{\vec{\psi}}_2 = \left(\boldsymbol{\mathcal{L}}_3 \operatorname{grad} \boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad} \boldsymbol{\mathcal{L}}_3\right)\frac{\hat{z}_2 - z}{h}, \tag{4.77}$$

$$\hat{\vec{\psi}}_3 = \left(\boldsymbol{\mathcal{L}}_3 \operatorname{grad} \boldsymbol{\mathcal{L}}_2 - \boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_3\right)\frac{\hat{z}_2 - z}{h}, \tag{4.78}$$

$$\hat{\vec{\psi}}_4 = \left(\boldsymbol{\mathcal{L}}_2 \operatorname{grad} \boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad} \boldsymbol{\mathcal{L}}_2\right)\frac{z - \hat{z}_1}{h}, \tag{4.79}$$

$$\hat{\vec{\psi}}_5 = \left(\boldsymbol{\mathcal{L}}_3 \operatorname{grad} \boldsymbol{\mathcal{L}}_1 - \boldsymbol{\mathcal{L}}_1 \operatorname{grad} \boldsymbol{\mathcal{L}}_3\right)\frac{z - \hat{z}_1}{h}, \tag{4.80}$$

$$\hat{\vec{\psi}}_6 = \left(\boldsymbol{\mathcal{L}}_3 \operatorname{grad}\boldsymbol{\mathcal{L}}_2 - \boldsymbol{\mathcal{L}}_2 \operatorname{grad}\boldsymbol{\mathcal{L}}_3\right)\frac{z-\hat{z}_1}{h}, \qquad (4.81)$$

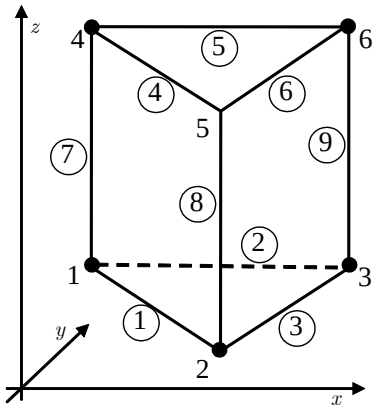


Рис. 46. Локальная нумерация узлов и рёбер на призме

где $h = \hat{z}_2 - \hat{z}_1$ – высота призмы ($\hat{z}_1$ и $\hat{z}_2$ – z-координаты нижнего и верхнего её оснований), а $\boldsymbol{\mathcal{L}}_i = \boldsymbol{\mathcal{L}}_i\left(x,y\right)$ – *двумерные* $\boldsymbol{\mathcal{L}}$-координаты треугольника, являющегося основанием призмы (коэффициенты α_j^i двумерных $\boldsymbol{\mathcal{L}}$-координат вычисляются через x- и y-координаты трёх вершин призмы любого из оснований по стандартным формулам (4.38)). Под $\operatorname{grad}\boldsymbol{\mathcal{L}}_i$ в формулах (4.76)-(4.81) понимается стандартный трёхмерный grad-вектор:

$$\operatorname{grad}\boldsymbol{\mathcal{L}}_i = \left(\frac{\partial\boldsymbol{\mathcal{L}}_i\left(x,y\right)}{\partial x}, \frac{\partial\boldsymbol{\mathcal{L}}_i\left(x,y\right)}{\partial y}, 0\right)^{\mathrm{T}}.$$

Очевидно, что каждая из вектор-функций (4.76)–(4.81) имеет нулевые z-составляющие и вектор-функции (4.76)–(4.78) на нижней грани $z = \hat{z}_1$, а вектор-функции (4.79)–(4.81) на верхней грани $z = \hat{z}_2$ полностью совпадают со стандартными базисными вектор-функциями треугольника, являющегося основанием призмы. Сомножители $\frac{\hat{z}_2 - z}{h}$ и $\frac{z-\hat{z}_1}{h}$ (в соотношениях (4.76)–(4.81)) делают эти вектор-функции убывающими от значений, равных значениям стандартных базисных вектор-функций треугольника на «своём» основании, до нуля на противоположном.

В результате каждая из вектор-функций (4.76)–(4.81) имеет ненулевые касательные только на двух гранях, которые пересекаются по ассоциированному с ней ребру (это одно из оснований призмы и боковая её грань). При этом на боковой (вертикальной) грани эти вектор-функции ведут себя как базисные вектор-функции прямоугольника, а на основании, как уже говорилось, – как базисные вектор-функции треугольника.

Базисные вектор-функции, ассоциированные с вертикальными рёбрами 7,8 и 9 (см. рис. 46), определим в виде

$$\hat{\vec{\psi}}_7 = \left(0,0,\tfrac{1}{h}\boldsymbol{\mathcal{L}}_1\right)^{\mathrm{T}}, \qquad \hat{\vec{\psi}}_8 = \left(0,0,\tfrac{1}{h}\boldsymbol{\mathcal{L}}_2\right)^{\mathrm{T}}, \qquad \hat{\vec{\psi}}_9 = \left(0,0,\tfrac{1}{h}\boldsymbol{\mathcal{L}}_3\right)^{\mathrm{T}}, \qquad (4.82)$$

где $\boldsymbol{\mathcal{L}}_i = \boldsymbol{\mathcal{L}}_i\left(x,y\right)$ – те же двумерные $\boldsymbol{\mathcal{L}}$-координаты треугольника, являющегося основанием призмы (множитель h в этих функциях поставлен для того, чтобы edge-функции призмы были определены в некотором смысле *единообразно*, т.е. их касательные составляющие на соответствующих рёбрах были равны длине ребра).

Вектор-функции (4.82), очевидно, перпендикулярны основаниям призмы, и каждая из них имеет ненулевые касательные составляющие только на двух боковых гранях, смежных по ребру, с которым она ассоциирована. При этом (по свойствам $\boldsymbol{\mathcal{L}}$-координат треугольника) модуль каждой из вектор-функций (4.82) линейно по x и y убывает от значения h на вертикальном ребре, с которым она ассоциирована, до нуля на остальных двух вертикальных рёбрах (и, естественно, на вертикальной грани, содержащей эти два вертикальных ребра). Таким образом, вектор-функции (4.82) на боковых (вертикальных) гранях ведут себя как базисные вектор-функции прямоугольника.

Заметим, что для вектор-функций (4.76)–(4.81) так же, как и для вектор-функций треугольника и тетраэдра, необходимо следить за тем, чтобы ассоциированные с одним и тем же ребром локальные вектор-функции этого типа на двух смежных призмах были направлены в одну и ту же сторону.

Конечные элементы в виде призм с векторными базисными функциями (как и с узловыми) применяются обычно для таких задач, у которых расчётные области имеют довольно сложное сечение, перпендикулярное одной из координатных осей, и это сечение очень слабо меняется при движении вдоль оси, которой оно перпендикулярно. При этом так же, как и при использовании векторных треугольников (с базисными вектор-функциями вида (4.44)–(4.46)), аппроксимационные качества треугольных призм с векторными базисными функциями вида (4.76)–(4.81) резко ухудшаются, если в основании призмы лежит вытянутый треугольник.

Устранить этот недостаток треугольной призмы можно введением на ней дополнительных базисных вектор-функций вида

$$\hat{\vec{\psi}}_p = \left(\boldsymbol{\mathcal{L}}_i \operatorname{grad} \boldsymbol{\mathcal{L}}_j + \boldsymbol{\mathcal{L}}_j \operatorname{grad} \boldsymbol{\mathcal{L}}_i\right) Z\left(z\right),$$

где $Z\left(z\right)$ – одна из линейных функций вида $\left(\hat{z}_2 - z\right)\big/h$ и $\left(z - \hat{z}_1\right)\big/h$. Но при этом и заметно возрастёт как число степеней свободы, так и заполненность матрицы конечноэлементной СЛАУ. Гораздо более выгодным вариантом может стать использование прямых призм с четырёхугольным основанием, являющихся (как и

параллелепипед) одним из частных случаев, пожалуй, наиболее удобного и эффективного (по соотношению вычислительных затрат и точности аппроксимации) векторного конечного элемента – шестигранника.

4.6.4. БАЗИСНЫЕ ВЕКТОР-ФУНКЦИИ НА ШЕСТИГРАННИКЕ

На шестиграннике так же, как и на четырёхугольнике, базисные вектор-функции удобней всего строить с помощью шаблонного элемента. Обычно в качестве шаблонного элемента берут кубик $[0,1]\times[0,1]\times[0,1]$ при использовании базиса лагранжева типа или кубик $[-1,1]\times[-1,1]\times[-1,1]$ при использовании иерархического базиса. Мы рассмотрим, как строятся базисные вектор-функции с использованием шаблонного кубика $\Omega^E=[-1,1]\times[-1,1]\times[-1,1]$.

Рассмотрим некоторый (произвольный) шестигранник Ω_m с вершинами $(\hat{x}_i,\hat{y}_i,\hat{z}_i)$, $i=1\ldots8$.

Зададим отображение шаблонного кубика Ω^E в шестигранник Ω_m соотношениями

$$x=\sum_{i=1}^{8}\hat{\varphi}_i(\xi,\eta,\zeta)\hat{x}_i,\quad y=\sum_{i=1}^{8}\hat{\varphi}_i(\xi,\eta,\zeta)\hat{y}_i,\quad z=\sum_{i=1}^{8}\hat{\varphi}_i(\xi,\eta,\zeta)\hat{z}_i, \tag{4.83}$$

где $\hat{\varphi}_i(\xi,\eta,\zeta)$ – стандартные скалярные трилинейные базисные функции, определённые на шаблонном элементе Ω^E (каждая функция $\hat{\varphi}_i(\xi,\eta,\zeta)$ является произведениями трёх линейных одномерных функций ξ, η и ζ вида $W_-(\vartheta)=\frac{1-\vartheta}{2}$ или $W_+(\vartheta)=\frac{1+\vartheta}{2}$ и равна единице в одной вершине Ω^E, которая отображается в узел $(\hat{x}_i,\hat{y}_i,\hat{z}_i)$, и нулю в остальных вершинах).

Определим отображение базисных вектор-функций $\hat{\vec{\varphi}}_i(\xi,\eta,\zeta)$ шаблонного элемента Ω^E на шестигранник Ω_m. При этом нам желательно отобразить $\hat{\vec{\varphi}}_i$ таким образом, чтобы у полученных на Ω_m вектор-функций $\hat{\vec{\psi}}_i$ касательные составляющие были ненулевыми только на тех гранях, на прообразах которых шаблонные вектор-функции $\hat{\vec{\varphi}}_i$ имеют ненулевые касательные. Этому условию будут удовлетворять вектор-функции $\hat{\vec{\psi}}_i$, полученные из $\hat{\vec{\varphi}}_i$ следующим образом:

$$\hat{\vec{\psi}}_i\left(x,y,z\right)=\mathbf{J}^{-1}\hat{\vec{\varphi}}_i\left(\xi\left(x,y,z\right),\eta\left(x,y,z\right),\zeta\left(x,y,z\right)\right), \tag{4.84}$$

где

$$\mathbf{J}=\begin{pmatrix}\dfrac{\partial x}{\partial\xi} & \dfrac{\partial y}{\partial\xi} & \dfrac{\partial z}{\partial\xi}\\ \dfrac{\partial x}{\partial\eta} & \dfrac{\partial y}{\partial\eta} & \dfrac{\partial z}{\partial\eta}\\ \dfrac{\partial x}{\partial\zeta} & \dfrac{\partial y}{\partial\zeta} & \dfrac{\partial z}{\partial\zeta}\end{pmatrix} \tag{4.85}$$

– функциональная матрица преобразования координат, переводящего кубик Ω^E в шестигранник Ω_m (сравните с функциональной матрицей (4.58) преобразования квадрата в четырёхугольник).

Доказательство того, что базисные вектор-функции (4.84) будут перпендикулярны всем тем граням шестигранника, которые являются образами граней Ω^E, где шаблонные вектор-функции $\hat{\vec{\varphi}}_i$ имеют ненулевые касательные, в целом мало отличается от соответствующего доказательства для четырёхугольника. Это доказательство базируется на том, что в любой точке $\left(x_p,y_p,z_p\right)$ шестигранника скалярное произведение любой вектор-функции $\hat{\vec{\psi}}$ на произвольный вектор $\vec{\mathbf{v}}$ может быть представлено в виде скалярного произведения вектора $\mathbf{J}\hat{\vec{\psi}}\left(x_p,y_p,z_p\right)$ на вектор $\vec{\mathbf{w}}$, связанный с $\vec{\mathbf{v}}$ соотношением $\vec{\mathbf{v}}=\mathbf{J}^{\mathrm{T}}\vec{\mathbf{w}}$, где $\mathbf{J}=\mathbf{J}\left(\xi\left(x_p,y_p,z_p\right),\eta\left(x_p,y_p,z_p\right),\zeta\left(x_p,y_p,z_p\right)\right)$.

Действительно,

$$\left(\hat{\vec{\psi}}\left(x_p,y_p,z_p\right),\vec{\mathbf{v}}\right)=\left(\hat{\vec{\psi}}\left(x_p,y_p,z_p\right),\mathbf{J}^{\mathrm{T}}\vec{\mathbf{w}}\right)=\left(\mathbf{J}\hat{\vec{\psi}}\left(x_p,y_p,z_p\right),\vec{\mathbf{w}}\right), \tag{4.86}$$

и из (4.86) сразу следует, что если вектор $\mathbf{J}\hat{\vec{\psi}}\left(x_p,y_p,z_p\right)$ перпендикулярен вектору $\vec{\mathbf{w}}$, то и $\hat{\vec{\psi}}\left(x_p,y_p,z_p\right)$ будет перпендикулярна вектору $\vec{\mathbf{v}}$. Но если в качестве $\mathbf{J}\hat{\vec{\psi}}\left(x,y,z\right)$ взять одну из базисных вектор-функций $\hat{\vec{\varphi}}_i$, а в качестве вектора $\vec{\mathbf{w}}$ произвольный вектор, касательный к тем граням Ω^E, которым перпендикулярна $\hat{\vec{\varphi}}_i$, то, во-первых, вектор $\vec{\mathbf{v}}$ будет касательным к грани шестигранника Ω_m, являющейся образом соответствующей грани кубика Ω^E, а во-вторых, вектор-функция $\hat{\vec{\psi}}_i$ на этой грани шестигранника Ω_m будет ему перпендикулярна (см. (4.86)). Если отображе-

ние Ω^E в Ω_m является взаимно однозначным, то, очевидно, для любого вектора $\vec{\mathbf{v}}$, касательного к какой-либо грани Γ шестигранника, можно найти вектор $\vec{\mathbf{w}}$, лежащий на грани-прообразе Γ и являющийся прообразом $\vec{\mathbf{v}}$ (этот вектор $\vec{\mathbf{w}} = \mathbf{J}^{-\mathrm{T}}\vec{\mathbf{v}}$). В результате из (4.86) следует, что *вектор-функция* $\hat{\vec{\psi}}_i$, *определённая как* $\mathbf{J}^{-1}\hat{\vec{\varphi}}_i$ (см. (4.84)), *будет ортогональна всем граням шестигранника* Ω_m, *являющимся образами тех граней кубика* Ω^E, *которым ортогональна* $\hat{\vec{\varphi}}_i$.

Таким образом, каждая из вектор-функций $\hat{\vec{\psi}}_i$, полученных из шаблонных вектор-функций первого порядка (это вектор-функции $\hat{\vec{\varphi}}_1 \div \hat{\vec{\varphi}}_{12}$ из раздела 4.6.1), будет иметь ненулевые касательные только на двух гранях, пересекающихся по ребру шестигранника Ω_m, являющегося образом ребра кубика Ω^E, с которым ассоциирован её прообраз $\hat{\vec{\varphi}}_i$ (к граням Ω_m, являющимся образами граней Ω^E, которым перпендикулярна $\hat{\vec{\varphi}}_i$, вектор-функция $\hat{\vec{\psi}}_i$ будет перпендикулярна в соответствии с (4.86), а на остальных двух гранях, не содержащих рассматриваемое ребро, $\hat{\vec{\psi}}_i$ будет равна нулю, поскольку на гранях, являющихся их прообразами, $\hat{\vec{\varphi}}_i$ равна нулю). Очевидно, вектор-функции $\hat{\vec{\psi}}_i = \mathbf{J}^{-1}\hat{\vec{\varphi}}_i$, $i = 1\ldots12$, как и их прообразы $\hat{\vec{\varphi}}_i$, имеет смысл ассоциировать с соответствующими рёбрами шестигранника.

Точно так же доказывается, что образы $\hat{\vec{\psi}}_i$ ассоциированных с рёбрами и гранями шаблонных вектор-функций $\hat{\vec{\varphi}}_i$ высокого порядка имеют ненулевые касательные либо только на двух смежных по одному ребру гранях (если вектор-функции ассоциированы с этим ребром), либо только на одной грани, с которой они ассоциированы (все же ассоциированные с самим конечным элементом Ω_m вектор-функции высоких порядков просто равны нулю на всех его гранях).

Не представляет особого труда определить базисные вектор-функции и на шестиграннике с изогнутыми гранями. Например, для описания трёхмерной геометрии расчётной области с изогнутыми границами удобно использовать шестигранник, полученный из шаблонного кубика $\Omega^E = \left[-1,1\right]\times\left[-1,1\right]\times\left[-1,1\right]$ отображением вида

$$x = \sum_{i=1}^{27} \hat{\varphi}_i^{tk}\left(\xi,\eta,\zeta\right)\hat{x}_i, \quad y = \sum_{i=1}^{27} \hat{\varphi}_i^{tk}\left(\xi,\eta,\zeta\right)\hat{y}_i, \quad z = \sum_{i=1}^{27} \hat{\varphi}_i^{tk}\left(\xi,\eta,\zeta\right)\hat{z}_i, \tag{4.87}$$

где $\hat{\varphi}_i^{tk}$ – это стандартные триквадратичные лагранжевы базисные функции, заданные на Ω^E, а $\left(\hat{x}_i, \hat{y}_i, \hat{z}_i\right)$, $i = 1\ldots27$, – координаты узлов криволинейного конечного элемента Ω_m, восемь из которых являются его вершинами, а остальные девятнадцать задаются для определения искривления рёбер и граней Ω_m (двенадцать узлов на «серединах» рёбер, шесть узлов в «центрах» граней и один узел в «центре» элемента).

Базисные вектор-функции на шестиграннике с изогнутыми гранями также определяются с помощью соотношения (4.84), в котором $\hat{\vec{\varphi}}_i$ – это базисные вектор-функции шаблонного элемента любого порядка (в том числе и первого), а **J** – функциональная матрица преобразования (4.87).

В заключение отметим, что все получаемые из шаблонных вектор-функций $\hat{\vec{\varphi}}_i$ по формуле (4.84) локальные базисные вектор-функции $\hat{\vec{\psi}}_i$ автоматически сшиваются в глобальные вектор-функции с непрерывными на границах между двумя шестигранниками касательными составляющими (если они, естественно, ассоциированы с одним и тем же ребром или одной и той же гранью, являются однотипными и «однонаправленными»).

Ключевым в доказательстве этого утверждения, опять же, является соотношение (4.86), а также и следующий факт.

Пусть Γ – граница между шестигранными элементами Ω_{m_1} и Ω_{m_2} (возможно, изогнутая). Если на гранях Γ_1^E и Γ_2^E шаблонного элемента Ω^E, которые переводятся в одну и ту же грань Γ при отображениях Ω^E в Ω_{m_1} и Ω_{m_2} соответственно, взять такие касательные к ним векторы $\vec{\mathbf{w}}_1$ и $\vec{\mathbf{w}}_2$, что они совпадают при совмещении Γ_1^E с Γ_2^E, то образом этих векторов будет один и тот же вектор $\vec{\mathbf{v}}$, касательный к поверхности Γ.

В таком случае из (4.86) в одной и той же точке $\left(x_p, y_p, z_p\right)$ грани Γ для любого касательного к Γ вектора $\vec{\mathbf{v}}$

$$\left(\hat{\vec{\psi}}_i^{\Omega_{m_1}}\left(x_p, y_p, z_p\right), \vec{\mathbf{v}}\right) = \left(\hat{\vec{\varphi}}_i\left(\xi_p^{\Omega_{m_1}}, \eta_p^{\Omega_{m_1}}, \zeta_p^{\Omega_{m_1}}\right), \vec{\mathbf{w}}_1\right),$$

$$\left(\hat{\vec{\psi}}_j^{\Omega_{m_2}}\left(x_p, y_p, z_p\right), \vec{\mathbf{v}}\right) = \left(\hat{\vec{\varphi}}_j\left(\xi_p^{\Omega_{m_2}}, \eta_p^{\Omega_{m_2}}, \zeta_p^{\Omega_{m_2}}\right), \vec{\mathbf{w}}_2\right),$$

где $\left(\xi_p^{\Omega_{m_1}}, \eta_p^{\Omega_{m_1}}, \zeta_p^{\Omega_{m_1}}\right)$ и $\left(\xi_p^{\Omega_{m_2}}, \eta_p^{\Omega_{m_2}}, \zeta_p^{\Omega_{m_2}}\right)$ – прообразы одной и той же точки $\left(x_p, y_p, z_p\right)$ на грани Γ, $\hat{\vec{\psi}}_i^{\Omega_{m_1}}$ и $\hat{\vec{\psi}}_j^{\Omega_{m_2}}$ – сшиваемые (по Γ) в одну глобальную вектор-функцию локальные вектор-функции элементов Ω_{m_1} и Ω_{m_2}, а $\hat{\vec{\varphi}}_i$ и $\hat{\vec{\varphi}}_j$ – их прообразы.

Тогда из условия

$$\left(\hat{\vec{\varphi}}_i\left(\xi_p^{\Omega_{m_1}}, \eta_p^{\Omega_{m_1}}, \zeta_p^{\Omega_{m_1}}\right), \vec{\mathbf{w}}_1\right) = \left(\hat{\vec{\varphi}}_j\left(\xi_p^{\Omega_{m_2}}, \eta_p^{\Omega_{m_2}}, \zeta_p^{\Omega_{m_2}}\right), \vec{\mathbf{w}}_2\right),$$

которое является естественным для прообразов сшиваемых вектор-функций, и следует сразу же равенство

$$\left(\hat{\vec{\psi}}_i^{\Omega_{m_1}}\left(x_p, y_p, z_p\right), \vec{\mathbf{v}}\right) = \left(\hat{\vec{\psi}}_j^{\Omega_{m_2}}\left(x_p, y_p, z_p\right), \vec{\mathbf{v}}\right),$$

означающее непрерывность касательных составляющих глобальных вектор-функций, получаемых сшивкой локальных вектор-функций $\hat{\vec{\psi}}_i^{\Omega_{m_1}}$ и $\hat{\vec{\psi}}_j^{\Omega_{m_2}}$.

Таким образом, из определённых нами с помощью соотношения (4.84) локальных базисных вектор-функций для любых шестигранных элементов (в том числе и с изогнутыми гранями) получаются глобальные вектор-функции любых порядков *с непрерывными на границах элементов касательными к ним составляющими*.

4.7. ГЕНЕРАЦИЯ КОНЕЧНОЭЛЕМЕНТНОЙ СЛАУ В ВЕКТОРНОМ МКЭ ПРИ РЕШЕНИИ ТРЁХМЕРНЫХ ЗАДАЧ

Мы приведём здесь локальные матрицы и векторы наиболее часто употребляемых трёхмерных edge-элементов (с вектор-функциями первого порядка) и покажем, как можно вычислять компоненты локальных матриц при использовании базисных вектор-функций высоких порядков на наиболее эффективных векторных элементах – шестигранниках. Кроме того, мы покажем, как следует обрабатывать правую часть векторного уравнения при наличии в ней слагаемых вида $d_1\vec{\mathbf{A}}^{\Leftarrow 1}$, $d_2\vec{\mathbf{A}}^{\Leftarrow 2}$ и т.д. (получаемых при аппроксимации по времени исходной нестационарной задачи, $\vec{\mathbf{A}}^{\Leftarrow k}$ – решения на предыдущих временных слоях), и рассмотрим особенности учёта краевых условий.

4.7.1. ОСНОВНЫЕ СООТНОШЕНИЯ, ОПРЕДЕЛЯЮЩИЕ КОНЕЧНОЭЛЕМЕНТНЫЕ МАТРИЦЫ И ВЕКТОРЫ

Формулы для вычисления компонент глобальных матриц жёсткости $\mathbf{G}$ и массы $\mathbf{M}$, определяющих глобальную матрицу $\mathbf{C} = \mathbf{G} + \mathbf{M}$ конечноэлементной СЛАУ, очевидно, имеют вид (см. формулу (4.18)):

$$G_{ij} = \int_{\Omega} \frac{1}{\mu} \operatorname{rot} \vec{\psi}_i \cdot \operatorname{rot} \vec{\psi}_j d\Omega, \qquad M_{ij} = \int_{\Omega} \gamma \vec{\psi}_i \cdot \vec{\psi}_j d\Omega. \tag{4.88}$$

Соответственно вклады в $\mathbf{G}$ и $\mathbf{M}$ от конечного элемента Ω_m определяются соотношениями

$$\hat{G}_{ij} = \int_{\Omega_m} \frac{1}{\mu} \operatorname{rot} \hat{\vec{\psi}}_i \cdot \operatorname{rot} \hat{\vec{\psi}}_j d\Omega, \tag{4.89}$$

$$\hat{M}_{ij} = \int_{\Omega_m} \gamma \hat{\vec{\psi}}_i \cdot \hat{\vec{\psi}}_j d\Omega, \tag{4.90}$$

где $\hat{\mathbf{G}}$ и $\hat{\mathbf{M}}$ – локальные матрицы жёсткости и массы конечного элемента Ω_m, а $\hat{\vec{\psi}}_i$ – его локальные базисные вектор-функции.

Компоненты глобального вектора $\mathbf{b}$ конечноэлементной СЛАУ (4.18), очевидно, определяются соотношением

$$b_i = \int_{\Omega} \vec{\mathbf{F}} \cdot \vec{\psi}_i d\Omega + \int_{S_2} \left(\vec{\mathbf{H}}^{\Theta} \times \vec{\mathbf{n}} \right) \cdot \vec{\psi}_i dS. \tag{4.91}$$

Таким образом, глобальный вектор правой части $\mathbf{b}$ собирается из локальных векторов конечных элементов Ω_m с компонентами вида $\int_{\Omega_m} \vec{\mathbf{F}} \cdot \hat{\vec{\psi}}_i d\Omega$ и из локальных векторов участков S_2^m границы S_2 с компонентами вида $\int_{S_2^m} \left(\vec{\mathbf{H}}^{\Theta} \times \vec{\mathbf{n}} \right) \cdot \hat{\vec{\psi}}_i dS$.